

Probabilistic Programming & Linear Logic.

MPRI 2025 – Nov. 7 & 14

Christine Tasson

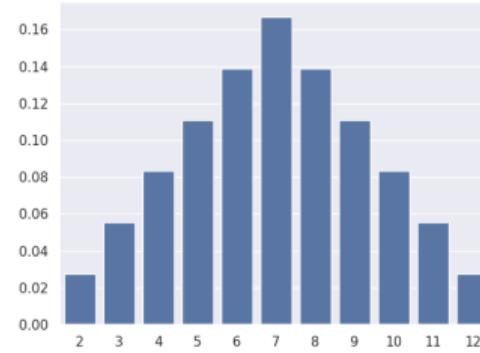


Probabilistic Programming

Programs as Random Variables

Sum of two Dice¹

```
def dice() → int:  
    a = sample(RandInt(1, 6))  
    b = sample(RandInt(1, 6))  
    return a + b  
  
with Enumeration():  
    dist: Categorical[float] = infer(dice)
```



dice is a Random Variable whose semantics is a distribution over $\{2, \dots, 12\}$ obtained by **sums**

$$\begin{aligned}\llbracket \text{dice} \rrbracket : \quad \mathbb{N} &\rightarrow \quad \mathbb{R}^+ \\ k &\mapsto \quad \sum_{a=1}^6 \sum_{b=1}^6 \frac{1}{36} \delta_{a+b}(k)\end{aligned}$$

¹ Baudart 2024, *muPPL, a PPL prototype in Python*, <https://github.com/gbdrt/mu-ppl>

Historical interlude - Kolmogorov (1930) probability

An **experiment** can produce a number of results

Example : toss a coin, roll a dice, choose a point on a line

Historical interlude - Kolmogorov (1930) probability

An **experiment** can produce a number of results

Example : toss a coin, roll a dice, choose a point on a line

A **universe** Ω : set of all possible results

Example : $\{1, 2, 3, 4, 5, 6\}$, \mathbb{R}

Historical interlude - Kolmogorov (1930) probability

An **experiment** can produce a number of results

Example : toss a coin, roll a dice, choose a point on a line

A **universe** Ω : set of all possible results

Example : $\{1, 2, 3, 4, 5, 6\}$, \mathbb{R}

An **event** : the subset of results satisfying a property whose probability is computable

Example : $\{1, 3, 5\}$ the *odd* event, $]0, 1[$, the *bounded by 0 and 1* event

Historical interlude - Kolmogorov (1930) probability

An **experiment** can produce a number of results

Example : toss a coin, roll a dice, choose a point on a line

A **universe** Ω : set of all possible results

Example : $\{1, 2, 3, 4, 5, 6\}$, \mathbb{R}

An **event** : the subset of results satisfying a property whose probability is computable

Example : $\{1, 3, 5\}$ the *odd* event, $]0, 1[$, the *bounded by 0 and 1* event

A σ -**algebra** \mathcal{F} : set of computable events

Example : $\mathcal{P}(\{1, 2, 3, 4, 5, 6\})$, intervals of \mathbb{R} .

Theoretical interlude - Probability space

Definition

Let Ω be a set, a σ -algebra \mathcal{F} is a set of subsets of Ω such that

$\emptyset \in \mathcal{F}$ and if $A_1, A_2, \dots \in \mathcal{F}$, then $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$
and if $A \in \mathcal{F}$, then $A^c \in \mathcal{F}$.

The σ -algebra **generated** by a family of subsets is the closure by complement countable intersections and unions. For instance:

The Borel σ -algebra is generated by finite rectangles

The Discrete σ -algebra is generated by singleton

Theoretical interlude - Probability space

Definition

Let Ω be a set, a σ -algebra \mathcal{F} is a set of subsets of Ω such that

$\emptyset \in \mathcal{F}$ and if $A_1, A_2, \dots \in \mathcal{F}$, then $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$
and if $A \in \mathcal{F}$, then $A^c \in \mathcal{F}$.

The σ -algebra **generated** by a family of subsets is the closure by complement countable intersections and unions. For instance:

The Borel σ -algebra is generated by finite rectangles

The Discrete σ -algebra is generated by singleton

Definition

A **measure** on (Ω, \mathcal{F}) is a function $\mu : \mathcal{F} \rightarrow [0, 1]$

such that $\mu(\emptyset) = 0$

and if $A_1, A_2, \dots \in \mathcal{F}$ are pairwise disjoint, then $\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i)$.

If moreover $\mathbb{P}(\Omega) = 1$, then it is a probability measure.

Theoretical interlude - Probability space

Definition

Let Ω be a set, a σ -algebra \mathcal{F} is a set of subsets of Ω such that

$\emptyset \in \mathcal{F}$ and if $A_1, A_2, \dots \in \mathcal{F}$, then $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$
and if $A \in \mathcal{F}$, then $A^c \in \mathcal{F}$.

The σ -algebra **generated** by a family of subsets is the closure by complement countable intersections and unions. For instance:

The Borel σ -algebra is generated by finite rectangles

The Discrete σ -algebra is generated by singleton

Definition

A **measure** on (Ω, \mathcal{F}) is a function $\mu : \mathcal{F} \rightarrow [0, 1]$

such that $\mu(\emptyset) = 0$

and if $A_1, A_2, \dots \in \mathcal{F}$ are pairwise disjoint, then $\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i)$.

If moreover $\mathbb{P}(\Omega) = 1$, then it is a probability measure.

A **probabilistic space** is given by $(\Omega, \mathcal{F}, \mathbb{P})$.

Random variable

Let $\Omega, \mathcal{F}, \mathbb{P}$ be a probability space.

Definition

A **random variable** is a measurable function $X : \Omega \rightarrow \mathbb{R}$ such that

$$\forall x \in \mathbb{R} \quad \{\omega \in \Omega \mid X(\omega) \leq x\} \in \mathcal{F}.$$

The **probability distribution** μ of X , denoted $X \sim \mu$ is the pushforward probability measure of X on its outputs:

$$\forall U \subseteq \mathbb{R}, \quad \mu(U) = \mathbb{P}(X^{-1}(U))$$

For a real random variable, the **cumulative distribution function** (CDF) of X is $F_X(x) = \mathbb{P}(X \leq x)$ characterizes its distribution μ . The inverted CDF is denoted idf_X .

Exercises: describe the universe and the random variable

```
def SqDist() → float:  
    x = sample(Uniform(-1, 1))  
    y = sample(Gaussian(0,1))  
    return x**2 + y**2
```

```
def RandomWalk(n: int, d) → int:  
    s = 0  
    for k in range(n):  
        s += sample(d)  
    return s
```

```
def StoppingTime(d) → int:  
    time = 1  
    while sample(d):  
        time = time +1  
    return time
```

```
def model():  
    m = sample(Gaussian(0.0, 3.0))  
    b = sample(Gaussian(0.0, 3.0))  
    f = lambda x: m*x +b  
    return f
```

Discrete Random Variable

Definition

A random variable $X : \Omega \rightarrow \mathbb{R}$ is discrete if it takes a countable number of values.

The probability mass function (PMF) of a discrete random variable X is a function $f : \mathbb{R} \rightarrow [0, 1]$ such that: $f(x) = \mathbb{P}(X = x) = \mathbb{P}(\{\omega \in \Omega \mid X(\omega) = x\})$.

Discrete Random Variable

Definition

A random variable $X : \Omega \rightarrow \mathbb{R}$ is discrete if it takes a countable number of values.

The probability mass function (PMF) of a discrete random variable X is a function $f : \mathbb{R} \rightarrow [0, 1]$ such that: $f(x) = \mathbb{P}(X = x) = \mathbb{P}(\{\omega \in \Omega \mid X(\omega) = x\})$.

The mean (expected value), written $\mathbb{E}(X)$ of a discrete random variable X is:

$$\mathbb{E}(X) = \sum_x x \mathbb{P}(X = x) = \sum_x x f(x)$$

when the sum exists (it is always the case when $X > 0$).

Discrete Random Variable

Definition

A random variable $X : \Omega \rightarrow \mathbb{R}$ is discrete if it takes a countable number of values.

The probability mass function (PMF) of a discrete random variable X is a function $f : \mathbb{R} \rightarrow [0, 1]$ such that: $f(x) = \mathbb{P}(X = x) = \mathbb{P}(\{\omega \in \Omega \mid X(\omega) = x\})$.

The mean (expected value), written $\mathbb{E}(X)$ of a discrete random variable X is:

$$\mathbb{E}(X) = \sum_x x \mathbb{P}(X = x) = \sum_x x f(x)$$

when the sum exists (it is always the case when $X > 0$).

If $g : \mathbb{R} \rightarrow \mathbb{R}$ measurable and X is a discrete random variable, then $g(X)$ is a discrete random variable

$$\mathbb{E}(g(X)) = \sum_x g(x) \mathbb{P}(X = x)$$

Trace Semantics

```
def dice() → int:  
    a = sample(RandInt(1, 6))  
    b = sample(RandInt(1, 6))  
    return a + b
```

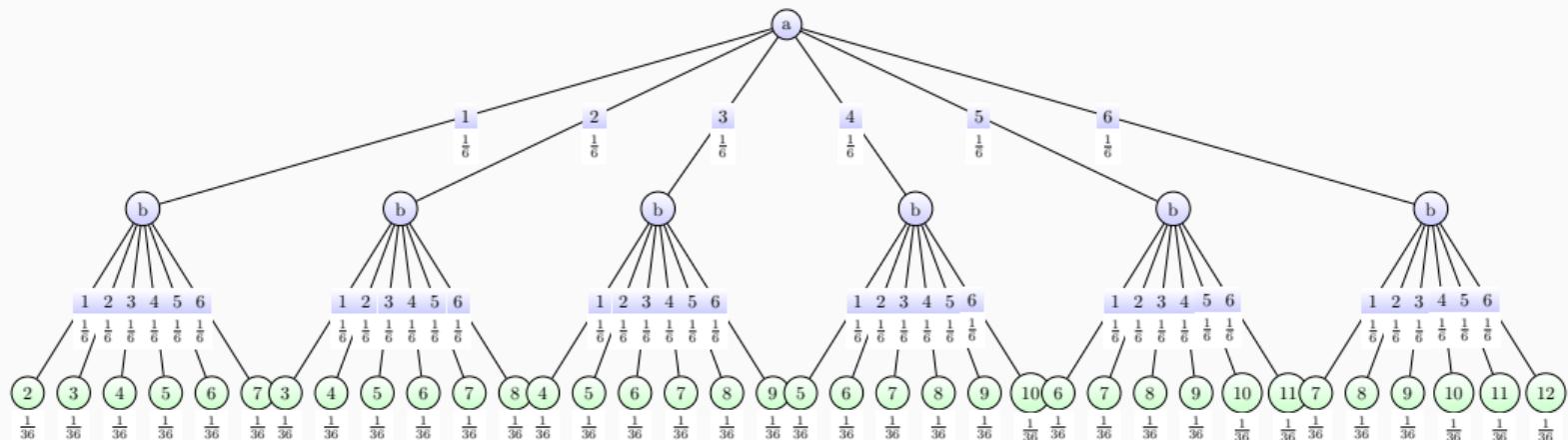
Trace Semantics

```
def dice() → int:  
    a = sample(RandInt(1, 6))  
    b = sample(RandInt(1, 6))  
    return a + b
```

This program simulates the **random variable** dice:

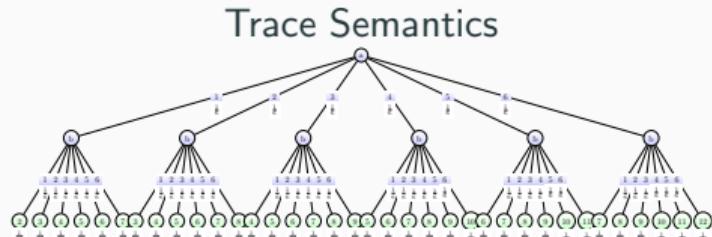
"The sum of two independent fair dice."

At each execution, the operator sample **simulates** a random variable uniform over $\{1, \dots, 6\}$.



What is the law of this program ?

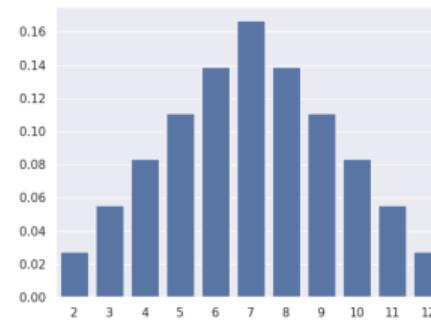
```
def dice() → int:  
    a = sample(RandInt(1, 6))  
    b = sample(RandInt(1, 6))  
    return a + b  
  
with Enumeration():  
    dist: Categorical[float] = infer(dice)
```



The discrete measure:

$$\begin{aligned}\llbracket \text{dice} \rrbracket : &= \mathbb{P}(\text{dice}() = k) \\ &= \sum_{a=1}^6 \sum_{b=1}^6 \frac{1}{36} \mathbb{1}_{\{a+b=k\}}\end{aligned}$$

Measure Semantics

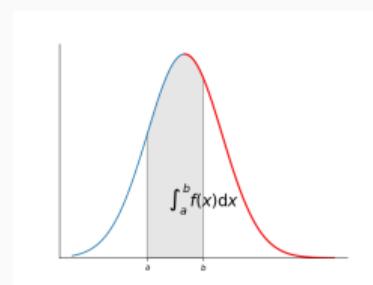


Continuous random variable with density

Definition

A random variable $X : \Omega \rightarrow \mathbb{R}$ is **continuous with density** (absolutely continuous with respect to Lebesgue's measure) if its **cumulative distribution function**: $F(x) = \mathbb{P}(\{\omega \mid X(\omega) \leq x\})$ can be described by the integral of $f : \mathbb{R} \rightarrow [0, \infty]$ called the probability **density function** (PDF) of X :

$$F(x) = \mathbb{P}(X \leq x) = \int_{-\infty}^x f(u)du$$

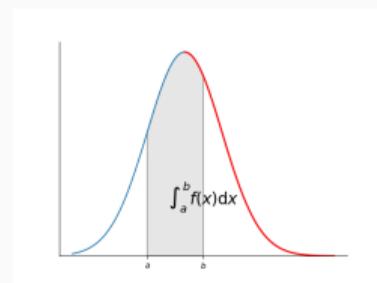


Continuous random variable with density

Definition

A random variable $X : \Omega \rightarrow \mathbb{R}$ is **continuous with density** (absolutely continuous with respect to Lebesgue's measure) if its **cumulative distribution function**: $F(x) = \mathbb{P}(\{\omega \mid X(\omega) \leq x\})$ can be described by the integral of $f : \mathbb{R} \rightarrow [0, \infty]$ called the probability **density function** (PDF) of X :

$$F(x) = \mathbb{P}(X \leq x) = \int_{-\infty}^x f(u)du$$



The **mean**, denoted $\mathbb{E}(X)$ of the continuous random variable X is defined by the integral when it exists:

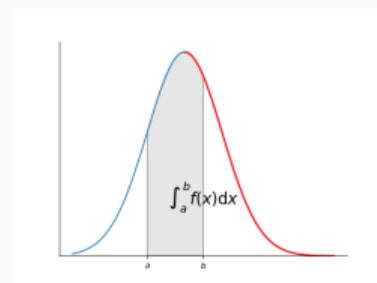
$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x f(u)du$$

Continuous random variable with density

Definition

A random variable $X : \Omega \rightarrow \mathbb{R}$ is **continuous with density** (absolutely continuous with respect to Lebesgue's measure) if its **cumulative distribution function**: $F(x) = \mathbb{P}(\{\omega \mid X(\omega) \leq x\})$ can be described by the integral of $f : \mathbb{R} \rightarrow [0, \infty]$ called the probability **density function** (PDF) of X :

$$F(x) = \mathbb{P}(X \leq x) = \int_{-\infty}^x f(u)du$$



The **mean**, denoted $\mathbb{E}(X)$ of the continuous random variable X is defined by the integral when it exists:

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x f(u)du \quad \mathbb{E}(g(X)) = \int_{-\infty}^{\infty} g(u) f(u)du$$

Density versus Measure semantics

```
def SqDist() → float:  
    x = sample(Uniform(-1, 1))  
    y = sample(Gaussian(0,1))  
    return x**2 + y**2
```

This program simulates the square distance from the origin to a point obtained from sampling in a uniform over $[-1, 1]$ and in a Gaussian of mean 0 and variance 1.

Density:

$$\text{pdf(SD)}(r_1, r_2) = \mathbb{1}_{[-1,1]}(r_1) \mathcal{N}(0, 1)(r_2)$$

Measure: by integration along the path

$$[\![\text{SD}]\!](U) = \int_{\substack{r_1 \in [0,1] \\ r_2 \in \mathbb{R}}} \mathcal{N}(0, 1)(r_2) \delta_{r_1^2 + r_2^2}(U) dr_1 dr_2$$

Path:

$$\alpha : r_1, r_2 \mapsto \delta_{r_1^2 + r_2^2}$$

Exercises: describe trace/density and measure semantics

```
def model():
    m = sample(Gaussian(0.0, 3.0))
    b = sample(Gaussian(0.0, 3.0))
    f = lambda x: m*x + b
    return f
```

```
def CondGauss() → float:
    x = sample(Uniform(-10,10))
    y = sample(Gaussian(x,1))
    return (x, y)
```

```
def FairCoin(d) → bool:
    a = sample(d)
    b = sample(d)
    if (a and not b):
        return True
    elif (b and not a):
        return False
    else:
        return FairCoin(d)
```

```
def StoppingTime(d) → int:
    time = 1
    while sample(d):
        time = time +1
    return time
```

Monte-Carlo simulation and the law of large numbers

A **Probabilistic program** is the [random variable](#) whose values are the outcome of the program execution. If the program has no [effect](#), then its [executions](#) are i.i.d. Then:

Monte-Carlo simulation and the law of large numbers

A **Probabilistic program** is the **random variable** whose values are the outcome of the program execution. If the program has no **effect**, then its **executions** are i.i.d. Then:

Law of large numbers:

Run n times the probabilistic program

Store the outputs x_1, \dots, x_n .

Compute $\frac{x_1 + \dots + x_n}{n}$ that approximates $\mathbb{E}(X)$ and $\frac{g(x_1) + \dots + g(x_n)}{n}$ that approximates $\mathbb{E}(g(X))$

Monte-Carlo simulation and the law of large numbers

A **Probabilistic program** is the **random variable** whose values are the outcome of the program execution. If the program has no **effect**, then its **executions** are i.i.d. Then:

Law of large numbers:

Run n times the probabilistic program

Store the outputs x_1, \dots, x_n .

Compute $\frac{x_1 + \dots + x_n}{n}$ that approximates $\mathbb{E}(X)$ and $\frac{g(x_1) + \dots + g(x_n)}{n}$ that approximates $\mathbb{E}(g(X))$

Monte-Carlo Simulation: histograms approximate distributions:

For a random variable X ,

$\frac{1}{n} \#(\{i | x_i = x\})$ approximates $\mathbb{E}(\chi_{X=x}) = \mathbb{P}(X = x)$

$\frac{1}{n} \#(\{i | a \leq x_i \leq b\})$ approximates $\mathbb{E}(\chi_{a \leq X \leq b}) = \mathbb{P}(a \leq X \leq b)$

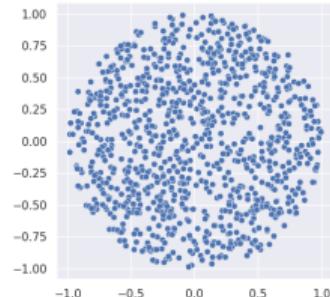
Probabilistic Programming

Bayesian Inference

Exercise: What are the Prior, Likelihood, Conditional measures

```
def HardDisk() → Tuple[float, float]:  
    x = sample(Uniform(-1, 1))  
    y = sample(Uniform(-1, 1))  
    d2 = x**2 + y**2  
    assume(d2 < 1)  
    return (x, y)
```

```
with RejectionSampling(num_samples=100):  
    dist: Empirical = infer(HardDisk)
```

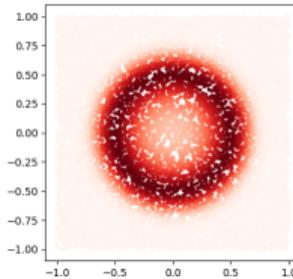


```
def SoftDisk() → Tuple[float, float]:  
    x = sample(Uniform(-1, 1))  
    y = sample(Uniform(-1, 1))  
    d2 = x**2 + y**2  
    observe(Gaussian(d2, 0.1), 0.5)  
    return(x, y)
```

```
with ImportanceSampling(num_particles  
=10000):
```

$$\frac{\int_{-1}^1 \int_{-1}^1 \text{obx}(x, y) \delta_{x,y} dx dy}{\int_{-1}^1 \int_{-1}^1 \delta_{x,y} dx dy}$$

Monte-Carlo
Simulation



Conditional Probability

Definition

The conditional probability is defined when $\mathbb{P}(B) > 0$, then

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Conditional Probability

Definition

The conditional probability is defined when $\mathbb{P}(B) > 0$, then

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Two events A and B are independent when one hence all equivalent properties are satisfied

$$\mathbb{P}(A) = \mathbb{P}(A|B) \text{ or } \mathbb{P}(B) = \mathbb{P}(B|A) \text{ or } \mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B).$$

Conditional Probability

Definition

The conditional probability is defined when $\mathbb{P}(B) > 0$, then

$$\mathbb{P}(A | B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Two events A and B are independent when one hence all equivalent properties are satisfied $\mathbb{P}(A) = \mathbb{P}(A|B)$ or $\mathbb{P}(B) = \mathbb{P}(B|A)$ or $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$.

Properties

if $\mathbb{P}(B) > 0$, then $\mathbb{P}(A) = \mathbb{P}(A|B)\mathbb{P}(B) + \mathbb{P}(A|B^c)\mathbb{P}(B^c)$.

if B_1, B_2, \dots, B_n is a partition Ω , then

$$\mathbb{P}(A) = \sum_{i=1}^n \mathbb{P}(A|B_i)\mathbb{P}(B_i)$$

Interlude theoretic - Bayes Law

Bayes Law

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B)\mathbb{P}(B)}{\mathbb{P}(A)}$$

Bayes Formula

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B)\mathbb{P}(B)}{\mathbb{P}(A|B)\mathbb{P}(B) + \mathbb{P}(A|B^c)\mathbb{P}(B^c)}$$

Generalization, if $\mathbb{P}(A) > 0$ and B_1, B_2, \dots, B_n is a partition of Ω such that $\forall i \mathbb{P}(B_i) > 0$, then

$$\mathbb{P}(B_j|A) = \frac{\mathbb{P}(A|B_j)\mathbb{P}(B_j)}{\sum_{i=1}^n \mathbb{P}(A|B_i)\mathbb{P}(B_i)}$$

Conditional Probability and random variables

Discrete

Bayes Law

$$\mathbb{P}(Y = y | X = x) = \frac{\mathbb{P}(X = x | Y = y) \mathbb{P}(Y = y)}{\mathbb{P}(X = x)}$$

Conditional Probability and random variables

Discrete

Bayes Law

$$\mathbb{P}(Y = y|X = x) = \frac{\mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(X = x)}$$

Bayes Formula

$$\mathbb{P}(Y = y|X = x) = \frac{\mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y)}{\sum_b \mathbb{P}(X = x|Y = b)\mathbb{P}(Y = b)}$$

Conditional Probability and random variables

Discrete

Bayes Law

$$\mathbb{P}(Y = y|X = x) = \frac{\mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(X = x)}$$

Bayes Formula

$$\mathbb{P}(Y = y|X = x) = \frac{\mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y)}{\sum_b \mathbb{P}(X = x|Y = b)\mathbb{P}(Y = b)}$$

Continuous

Bayes Law

$$f_{Y|X}(y|x) = \frac{f_{X|Y}(x|y)f_Y(y)}{f_X(x)}$$

Conditional Probability and random variables

Discrete

Bayes Law

$$\mathbb{P}(Y = y|X = x) = \frac{\mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(X = x)}$$

Bayes Formula

$$\mathbb{P}(Y = y|X = x) = \frac{\mathbb{P}(X = x|Y = y)\mathbb{P}(Y = y)}{\sum_b \mathbb{P}(X = x|Y = b)\mathbb{P}(Y = b)}$$

Continuous

Bayes Law

$$f_{Y|X}(y|x) = \frac{f_{X|Y}(x|y)f_Y(y)}{f_X(x)}$$

Bayes Formula

$$f_{Y|X}(y|x) = \frac{f_{X|Y}(x|y)f_Y(y)}{\int_{-\infty}^{\infty} f_{X|Y}(x|y)f_Y(y)dy}$$

Theoretical Interlude - Conditional Law

Joint Distribution: Let X and Y be two random variables. Their joint distribution function is given by

$$F(x, y) = \mathbb{P}(X \leq x, Y \leq y)$$

X and Y are jointly continuous with density if there is joint probability density function f such that

$$F(x, y) = \int_{-\infty}^y \int_{-\infty}^x f_{X,Y}(u, v) dudv$$

Theoretical Interlude - Conditional Law

Joint Distribution: Let X and Y be two random variables. Their joint distribution function is given by

$$F(x, y) = \mathbb{P}(X \leq x, Y \leq y)$$

X and Y are jointly continuous with density if there is joint probability density function f such that

$$F(x, y) = \int_{-\infty}^y \int_{-\infty}^x f_{X,Y}(u, v) dudv$$

Then, the second marginal give the density function of the law of Y

$$f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(u, y) du$$

Theoretical Interlude - Conditional Law

Joint Distribution: Let X and Y be two random variables. Their joint distribution function is given by

$$F(x, y) = \mathbb{P}(X \leq x, Y \leq y)$$

X and Y are jointly continuous with density if there is joint probability density function f such that

$$F(x, y) = \int_{-\infty}^y \int_{-\infty}^x f_{X,Y}(u, v) du dv$$

Then, the second marginal give the density function of the law of Y

$$f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(u, y) du$$

Let us define **conditional density function** by:

$$f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)}$$

Exercise: What are the Prior, Likelihood, Conditional measures

```

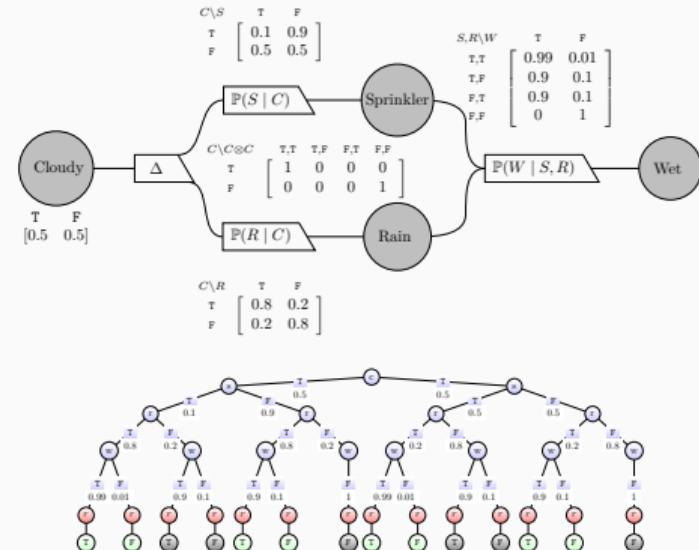
def wet() → bool:
    cloudy = sample(Bernoulli(0.5))

    p_s, p_r = (0.1, 0.8) if cloudy else (0.5, 0.2)
    sprinkle = sample(Bernoulli(p_s))
    rain = sample(Bernoulli(p_r))

    p_w = 0.99 if (sprinkle and rain) else 0.9 if
        (sprinkle != rain) else 0
    wet = sample(Bernoulli(p_w))
    assume(rain)
    return wet

with Enumeration():
    dist: Categorical[bool] = infer(wet)

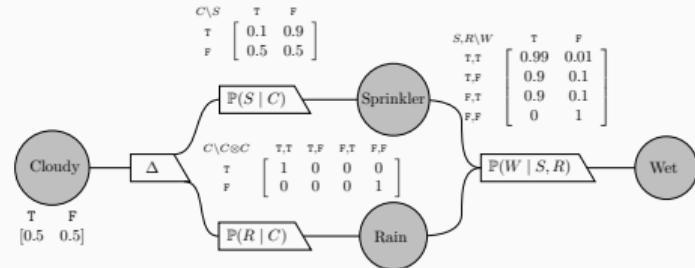
```



$$\mathbb{P}(\text{Wet} | \text{Rain}=\text{true}) = \frac{\sum_{c,s} \mathbb{P}(\text{Wet} = \text{T} | c, s, r)}{\sum_{c,s,r} \mathbb{P}(\text{Wet} = \text{T} | c, s, r)}$$

Exercise: What are the Prior, Likelihood, Conditional measures

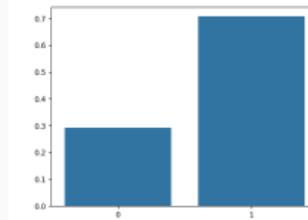
```
def wet() → bool:  
    cloudy = sample(Bernoulli(0.5))  
  
    p_s, p_r = (0.1, 0.8) if cloudy else (0.5, 0.2)  
    sprinkle = sample(Bernoulli(p_s))  
    rain = sample(Bernoulli(p_r))  
  
    p_w = 0.99 if (sprinkle and rain) else 0.9 if  
        (sprinkle != rain) else 0  
    wet = sample(Bernoulli(p_w))  
    assume(rain)  
    return wet  
  
with Enumeration():  
    dist: Categorical[bool] = infer(wet)
```



$$C \setminus S \begin{array}{cc} T & F \\ \hline T & 0.1 \quad 0.9 \\ F & 0.5 \quad 0.5 \end{array}$$

$$C \setminus R \begin{array}{cc} T & F \\ \hline T & 0.8 \quad 0.2 \\ F & 0.2 \quad 0.8 \end{array}$$

$$S, R \setminus W \begin{array}{cc} T & F \\ \hline T, T & 0.99 \quad 0.01 \\ T, F & 0.9 \quad 0.1 \\ F, T & 0.9 \quad 0.1 \\ F, F & 0 \quad 1 \end{array}$$



Exact Mass Function

Parameterized Programs

```
def sum(d) → int:  
    a = sample(d)  
    b = sample(d)  
    return a + b
```

It is a probabilistic distribution transformer.

Discrete semantics: stochastic matrix

$$\llbracket \text{sum} \rrbracket (d) : k \mapsto \sum_a \sum_b d_a d_b \delta_{a+b}(k) = \sum_{a+b=k} d_a d_b$$

Continuous semantics: stochastic kernel

$$\llbracket \text{sum} \rrbracket (\mu) : U \mapsto \int \int \mu(da) \mu(db) \delta_{a+b}(U)$$

Probabilistic Programming

The Category of Markov Kernels

Markov Kernels (History)

Lawvere 1962, “The category of probabilistic mappings” introduces stochastic kernels to denote temporally discrete Markov processes.

Kozen 1979, “Semantics of Probabilistic Programs” proposes stochastic kernels between measurable spaces to denote operational semantics of probabilistic programs.

Giry 1982, “A categorical approach to probability theory” describes Markov kernels as the Kleisli Category of the probabilistic monad.

Panangaden 1998, “The Category of Markov Kernels” relates Markov Kernels and Relations.

Markov Kernels (Definition)

Measurable space: $\mathcal{X} = (\Omega_{\mathcal{X}}, \mathcal{F}_{\mathcal{X}})$

$\Omega_{\mathcal{X}}$ set of possible experiment outcomes, with $\mathcal{F}_{\mathcal{X}}$ the σ -algebra of measurable events.

Stochastic kernel: $\kappa : \mathcal{X} \rightsquigarrow \mathcal{Y}$ is a function $\kappa : \Omega_{\mathcal{X}} \times \mathcal{F}_{\mathcal{Y}} \rightarrow \mathbb{R}^+$ such that

$\forall x \in \mathcal{X}, \kappa(\cdot|x) : \mathcal{F}_{\mathcal{Y}} \rightarrow \mathbb{R}^+$ is a measure

$\forall V \in \mathcal{F}_{\mathcal{Y}}, \kappa(V| \cdot) : \mathcal{X} \rightarrow \mathbb{R}^+$ is a measurable function

Composition: $\kappa : \mathcal{X} \rightsquigarrow \mathcal{Y}$ and $\kappa' : \mathcal{Y} \rightsquigarrow \mathcal{Z}$.

$$\forall x \in \Omega_{\mathcal{X}} \quad \forall W \in \mathcal{F}_{\mathcal{Z}}, \quad \kappa' \circ \kappa(W|x) = \int_{\mathcal{Y}} \kappa'(W|y) \kappa(dy|x)$$

Markov Kernel (Monad of Finite Measures)

Finite Measures: Let \mathcal{X} be a measurable space.

$\mathcal{G}(\mathcal{X})$ is the set of finite measures μ

Giry Monad

$\mathcal{G}(\mathcal{X})$ is a measurable space when endowed with the σ -algebra generated by the evaluations on measurable sets $\text{ev}_U : \mu \mapsto \mu(U)$, i.e. by the collection of $\text{ev}_U^{-1}([a, b])$ for $a, b \in \mathbb{R}^+$.

Remark: $\mathcal{G}(\{0, 1\}) = \{p\delta_0 + (1 - p)\delta_1 \mid 0 \leq p \leq 1\}$

with $\{\delta_1\}$ and $\{p\delta_0 + (1 - p)\delta_1 \mid p \in [0, 1] \cap [a, b]\}$ measurable sets.

Kernel and Giry: $\text{Kern}(\mathcal{X}, \mathcal{Y}) = \text{Meas}(\mathcal{X}, \mathcal{G}(\mathcal{Y}))$

Kernels are measures transformers thanks to the giry monad bind

$$\forall \mu \in \mathcal{G}(\mathcal{X}) \quad \kappa \cdot \mu : W \mapsto \int \kappa(W|x) \mu(dx) \in \mathcal{G}(\mathcal{Y})$$

Probabilistic programs with several parameters

```
def sumd(d1, d2) → int:  
    a = sample(d1)  
    b = sample(d2)  
    return a + b
```

It is interpreted as a measure parameterized on the joint measure of two independent measures.

Monoidal unit:

$$\Omega_1 = \{*\}.$$

Monoidal product

$$\Omega_{\mathcal{X} \otimes \mathcal{Y}} = \Omega_{\mathcal{X}} \times \Omega_{\mathcal{Y}}$$

$\mathcal{F}_{\mathcal{X} \otimes \mathcal{Y}}$ generated by squares of measure sets $U \times V$ for $U \in \mathcal{F}_{\mathcal{X}}$ and $V \in \mathcal{F}_{\mathcal{Y}}$

Product of kernels

$$\kappa \otimes \kappa'(U \times V|x, y) = \kappa(U|x) \kappa'(V|y)$$

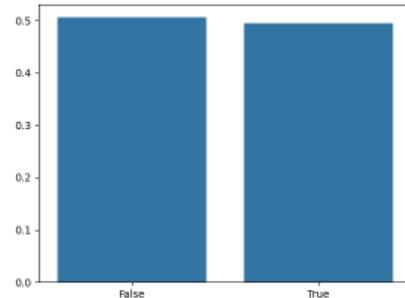
Markov Kernels is Symmetric Monoidal

Fair Coin Example: Need for Higher-Order

```
def FairCoin(d) → bool:  
    a = sample(d)  
    b = sample(d)  
    if (a and not b):  
        return True  
    elif (b and not a):  
        return False  
    else:  
        return FairCoin(d)
```

```
with ImportanceSampling(num_particles=1000):  
    FC: Categorical[bool] = infer(FairCoin,  
                                Bernoulli(0.3))
```

FC is a fair coin !



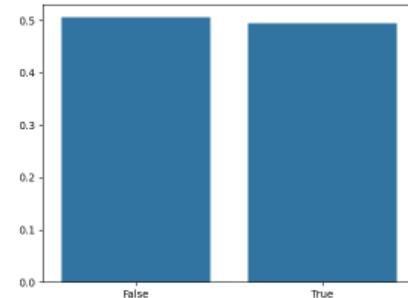
How can we prove it ?

Fair Coin Example: Need for Higher-Order

```
def FairCoin(d) → bool:  
    a = sample(d)  
    b = sample(d)  
    if (a and not b):  
        return True  
    elif (b and not a):  
        return False  
    else:  
        return FairCoin(d)
```

```
with ImportanceSampling(num_particles=1000):  
    FC: Categorical[bool] = infer(FairCoin,  
                                  Bernoulli(0.3))
```

FC is a fair coin !



How can we prove it ?

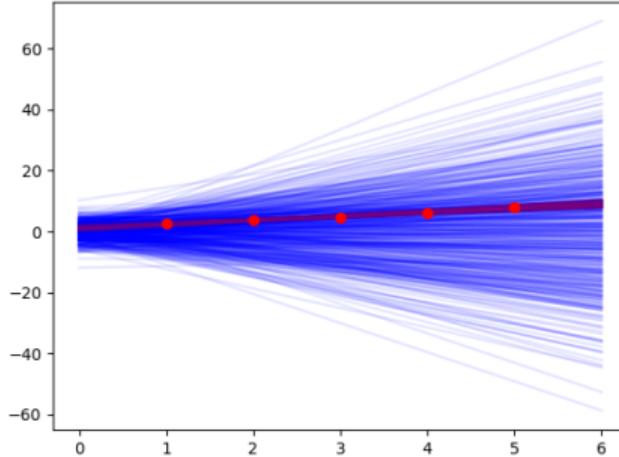
Recursive equation (if $p \notin \{0, 1\}$):

$$\begin{aligned}\mathcal{F}(t) &= p(1 - p)(\delta_T + \delta_F) \\ &\quad + (1 - 2(p(1 - p))) t \\ \llbracket \text{FC} \rrbracket &= \frac{1}{2}(\delta_T + \delta_F)\end{aligned}$$

Linear Regression

```
def model(data):
    m = sample(Gaussian(0.0, 3.0))
    b = sample(Gaussian(0.0, 3.0))
    f = lambda x: m*x +b
    for (x, y) in data:
        observe(Gaussian(f(x), 0.5), y)
    return f

data = [(1.0, 2.5), (2.0, 3.8), (3.0, 4.5),
        (4.0, 6.2), (5.0, 8.0)]
```



Wanted: Symmetric Monoidal Closed Category

- to interpret the type of parameterized programs.

Currying: $\mathbf{Kern}(\mathcal{X} \otimes \mathcal{Y}, \mathcal{Z}) = \mathbf{Kern}(\mathcal{X}, \mathcal{Y} \multimap \mathcal{Z})$

Evaluation: $\text{ev} \in \mathbf{Kern}(\mathcal{X} \otimes (\mathcal{X} \multimap \mathcal{Y}), \mathcal{Y})$

$\mathcal{X} \multimap \mathcal{Y}$ can be also denoted $\mathcal{Y}^{\mathcal{X}}$

Aumann's Lemma (1961), revisited (thanks to Ohad Kammar and Thomas Ehrhard.)

Markov kernel is not closed as there is no measurability structures that can be put on real measurable functions such that the evaluation is measurable.

Assume, by contradiction, that Markov Kernel is an SMCC:

$$\mathbf{Kern}(\mathcal{X} \otimes \mathcal{Y}, \mathcal{Z}) = \mathbf{Kern}(\mathcal{X}, \mathcal{Z}^{\mathcal{Y}})$$

where $\mathcal{Z}^{\mathcal{Y}}$ is a measurable space and the evaluation $\text{ev} : \mathcal{Z}^{\mathcal{Y}} \otimes \mathcal{Y} \rightsquigarrow \mathcal{Z}$ is a kernel

Consequence: $\mathbf{Kern}(\mathcal{Y}, \mathcal{Z}) = \mathcal{G}(\mathcal{Z}^{\mathcal{Y}})$

Proof: $\mathbf{Kern}(\mathcal{Y}, \mathcal{Z}) = \mathbf{Kern}(1 \otimes \mathcal{Y}, \mathcal{Z}) = \mathbf{Kern}(1, \mathcal{Z}^{\mathcal{Y}}) = \mathbf{Meas}(1, \mathcal{G}(\mathcal{Z}^{\mathcal{Y}})) = \mathcal{G}(\mathcal{Z}^{\mathcal{Y}})$

Consequence: the diagonal $\Delta = \{(x, y) \in \mathbb{R}^2 \mid x = y\}$ is measurable in $\mathcal{X} \otimes \mathcal{Y}$ where

$\mathcal{X} = (\mathbb{R}, \mathcal{P}(\mathbb{R}))$ the discrete σ -algebra

$\mathcal{Y} = (\mathbb{R}, \mathcal{C}(\mathbb{R}))$ the countable-cocountable σ -algebra (*generated by countable parts and parts whose complement are countable, closed by countable unions and intersections.*)

Contradiction: the diagonal cannot be measurable in $\mathcal{X} \otimes \mathcal{Y}$

Aumann's Lemma, revisited (thanks to Ohad Kammar and Thomas Ehrhard.)

Consequence: the diagonal $\Delta = \{(x, y) \in \mathbb{R}^2 \mid x = y\}$ is measurable in $\mathcal{X} \otimes \mathcal{Y}$ where

$\mathcal{X} = (\mathbb{R}, \mathcal{P}(\mathbb{R}))$ the discrete σ -algebra

$\mathcal{Y} = (\mathbb{R}, \mathcal{C}(\mathbb{R}))$ the countable-cocountable σ -algebra (*generated by countable parts and parts whose complement are countable, closed by countable unions and intersections.*)

Proof: Let $x \in \mathbb{R}$ and $h_x \in \mathbf{Meas}(\mathcal{Y}, \mathcal{G}(\{0, 1\})) = \mathbf{Kern}(\mathcal{Y}, \{0, 1\})$

$h_x : y \mapsto \begin{cases} \delta_1 & \text{if } x = y \\ \delta_0 & \text{otherwise} \end{cases}$ $h_x^{-1}(\{p\delta_0 + (1-p)\delta_1 \mid p \in [0, 1] \cap [a, b]\})$ can be $\emptyset, \mathbb{R}, \{x\}$ thus countable, or $\mathbb{R} \setminus \{x\}$ thus cocountable.

Since in \mathcal{X} any part is measurable, $\lambda x. h_x \in \mathbf{Meas}(\mathcal{X}, \mathbf{Kern}(\mathcal{Y}, \{0, 1\}))$, which is isomorphic to

$$\mathbf{Meas}(\mathcal{X}, \mathcal{G}(\{0, 1\}^{\mathcal{Y}})) = \mathbf{Kern}(\mathcal{X}, \{0, 1\}^{\mathcal{Y}}) = \mathbf{Kern}(\mathcal{X} \otimes \mathcal{Y}, \{0, 1\}) = \mathbf{Meas}(\mathcal{X} \otimes \mathcal{Y}, \mathcal{G}(\{0, 1\}))$$

We have built a measurable function \tilde{h} such that $\Delta = \tilde{h}^{-1}(\{\delta_1\})$.

Thus, Δ is measurable in $\mathcal{X} \otimes \mathcal{Y}$

Aumann's Lemma, revisited (thanks to Ohad Kammar and Thomas Ehrhard.)

Proposition: If W is measurable in $\mathcal{X} \otimes \mathcal{Y}$, then there is $B \subseteq \mathbb{R}$ countable such that

$$\text{If there is } (x, x') \in W \text{ such that } x' \notin B, \text{ then } \forall y \notin B, (x, y) \in W. \quad (1)$$

Proof: it is satisfied by all basic measurable sets and closed by countable union and countable intersection.

The Diagonal does not satisfy this proposition.

Proof: By contradiction, assume there is $B \subseteq \mathbb{R}$ countable satisfying (1).

Since B is countable, there are $x \notin B$ with $(x, x) \in \Delta$ and $y \notin B$ with $y \neq x$, thus $(x, y) \notin \Delta$.

Thus, B does not satisfy (1)

Contradiction: the diagonal is measurable and not measurable in $\mathcal{X} \otimes \mathcal{Y}$

Wish List for Higher-Order Probabilistic Programs with continuous distributions

Markov category:

- Measures:** $\mathcal{G}(\mathbb{R})$ to interpret close programs of type float as measures
- Stochastic kernels** (parameterized measures) to interpret programs
- Integration** to interpret sampling
- Tensor product** to interpret joint distribution
- Higher-Order**

Higher-order models of Probabilistic Programming

- Quasi-Borel Spaces: Heunen et al. 2017, “A convenient category for higher-order probability theory”
- Banach Spaces: Dahlqvist and Kozen 2020, “Semantics of higher-order probabilistic programs with conditioning”
- Cones:** Ehrhard, Pagani, and Tasson 2018, “Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming” and Ehrhard and Geoffroy 2025, “Integration in Cones”

Linear Logic and Probability

Semantics

Semantics, Probabilistic Programming and Linear Logic

Why studying semantics ?

- Beauty and nobility
- Correctness of Programs
- Sound transformation of programs
- Design new languages

Linear Logic, an inspiration for resource aware languages

In the 80's, Jean-Yves Girard recognizes, in a concrete model of the simply typed lambda-calculus, the *linear and exponential decomposition* of object of morphisms

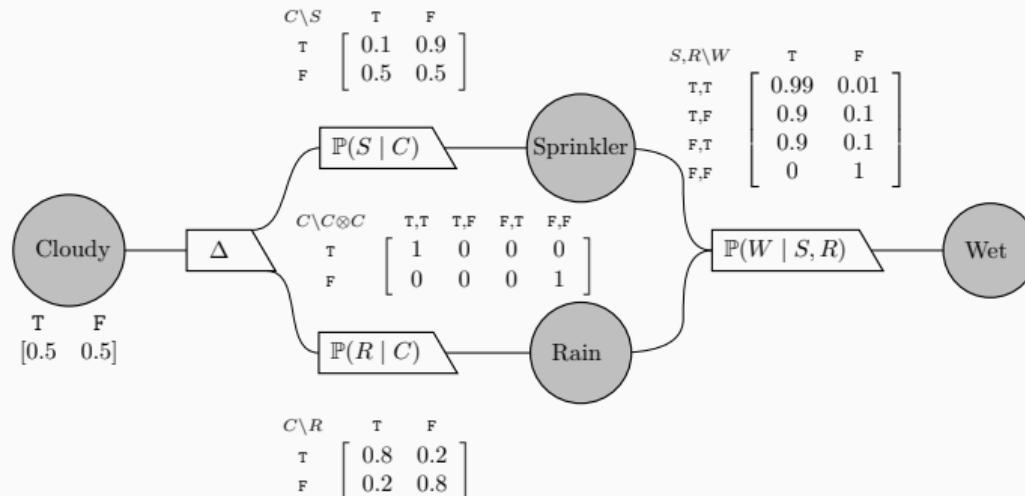
$$A \Rightarrow B \text{ versus } !A \multimap B$$

linearity and exponential are central in the semantics of Probabilistic Programming as already hinted in Girard 1988, "Normal functors, power series and λ -calculus".

Linear Logic and Probability

Copy, discard

Bayesian Network²: through copying³ values



$$P(C) ; \Delta ; (P(S | C) \otimes P(R | C)) ; P(W | S, R) = P(W)$$

²Pearl 1988, *Probabilistic reasoning in intelligent systems: networks of plausible inference*

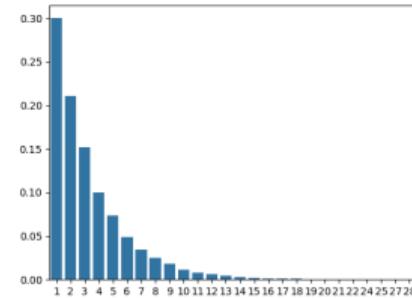
³Cho and Jacobs 2019, "Disintegration and Bayesian inversion via string diagrams"

Linear Logic and Probability

IID copies

Stopping Time: Need for bag of i.i.d. copies of a distribution

```
def StoppingTime(d) → int:  
    time = 1  
    while sample(d):  
        time = time +1  
    return time  
  
with ImportanceSampling(num_particles=10000):  
    ST: Categorical[float] = infer(StoppingTime,  
        Bernoulli(0.7))
```



Approximated Mass Function of
 $\mathbb{P}(\text{ST}(\text{Bernoulli}(0.7)))$

$\text{ST} = \text{StoppingTime}(\mu)$ is a Random Variable whose semantics is a (sub)probabilistic distribution over \mathbb{N} , with infinite support: $\text{supp}([\text{ST}]) = \mathbb{N}$,

$$\begin{aligned}\llbracket \text{StoppingTime}(\mu) \rrbracket : \quad & \Omega_{\text{int}} \rightarrow \mathbb{R}^+ \\ & k \mapsto \mathbb{P}(\text{ST} = k \mid d = \mu)\end{aligned}$$

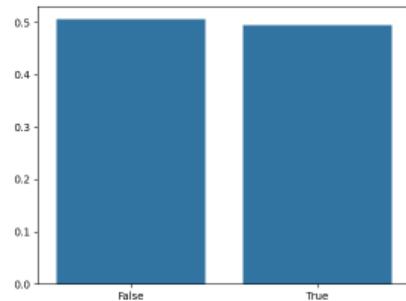
To compute the entire distribution $\llbracket \text{StoppingTime}(\mu) \rrbracket$, we sample **i.i.d. copies** of μ .

Fair Coin Example: Need for Higher-Order

```
def FairCoin(d) → bool:  
    a = sample(d)  
    b = sample(d)  
    if (a and not b):  
        return True  
    elif (b and not a):  
        return False  
    else:  
        return FairCoin(d)
```

```
with ImportanceSampling(num_particles=1000):  
    FC: Categorical[bool] = infer(FairCoin,  
                                  Bernoulli(0.3))
```

FC is a fair coin !



How can we prove it ?

Wish List for Higher-Order Probabilistic Programs

- Measures:** to interpret close programs of ground types unit, booleans, integers, reals
- Stochastic matrices or kernels** (parameterized measures) to interpret programs sampling in measures given as parameters
- Sum or Integration** to interpret sampling

$$[\![\text{let } x = \text{sample } N \text{ in } M]\!] = \sum_{a \in \mathbb{N}} [\![M]\!]_a [\![N]\!]_a \text{ or } \int_{r \in \mathbb{R}} ([\![M]\!] \circ \delta)(r) [\![N]\!](dr)$$

- Tensor product** for joint distributions
- Copy and Discard** to duplicate Samples
- Bags of i.i.d. copies** to duplicate distributions
- Higher-Order** for recursive programs and compositionality
- CPO-enriched** for loops