

# Probabilistic Programming & Linear Logic.

MPRI 2025 – Nov. 7 & 14

---

Christine Tasson

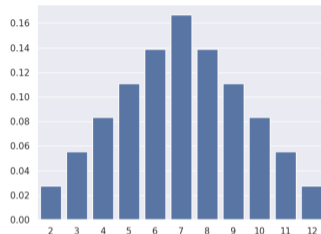
# Probabilistic Programming

---

## Programs as Random Variables

## Sum of two Dice<sup>1</sup>

```
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**

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

<sup>1</sup>Baudart 2024, *muPPL, a PPL prototype in Python*, <https://github.com/gbdr/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**  $\Omega$  : 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**  $\Omega$  : 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**  $\Omega$  : 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  **$\sigma$ -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  $\Omega$  be a set, a  $\sigma$ -algebra  $\mathcal{F}$  is a set of subsets of  $\Omega$  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  $\sigma$ -algebra **generated** by a family of subsets is the closure by complement countable intersections and unions. For instance:

The Borel  $\sigma$ -algebra is generated by finite rectangles

The Discrete  $\sigma$ -algebra is generated by singleton

## Theoretical interlude - Probability space

### Definition

Let  $\Omega$  be a set, a  $\sigma$ -algebra  $\mathcal{F}$  is a set of subsets of  $\Omega$  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  $\sigma$ -algebra **generated** by a family of subsets is the closure by complement countable intersections and unions. For instance:

The Borel  $\sigma$ -algebra is generated by finite rectangles

The Discrete  $\sigma$ -algebra is generated by singleton

### Definition

A **measure** on  $(\Omega, \mathcal{F})$  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  $\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$ .

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

## Theoretical interlude - Probability space

### Definition

Let  $\Omega$  be a set, a  $\sigma$ -algebra  $\mathcal{F}$  is a set of subsets of  $\Omega$  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  $\sigma$ -algebra **generated** by a family of subsets is the closure by complement countable intersections and unions. For instance:

The Borel  $\sigma$ -algebra is generated by finite rectangles

The Discrete  $\sigma$ -algebra is generated by singleton

### Definition

A **measure** on  $(\Omega, \mathcal{F})$  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  $\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$ .

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

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

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

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

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

## 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 take 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 take 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  $X \geq 0$ ).

# Discrete Random Variable

## Definition

A random variable  $X : \Omega \rightarrow \mathbb{R}$  is **discrete** if it take 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  $X \geq 0$ ).

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
```

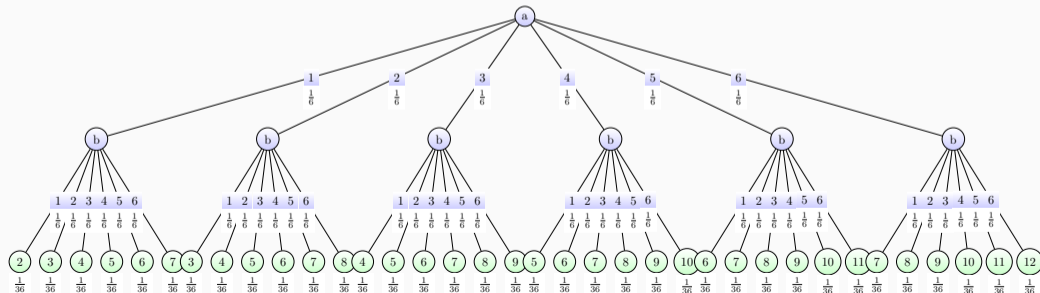
# Trace Semantics

```
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  $\{1, \dots, 6\}$ .



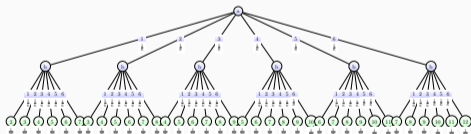
# What is the law of this program ?

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

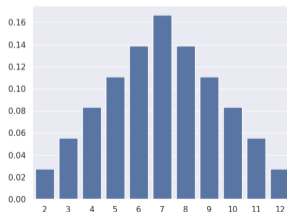
## The discrete measure:

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

Trace Semantics



Measure Semantics

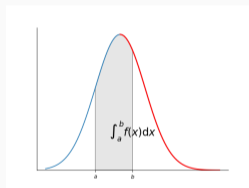


# Continuous random variable with density

## Definition

A random variable  $X : \Omega \rightarrow \mathbb{R}$  is **continuous with density** (absolutely continuous with respect to Lebesgues 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  $X$ :

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

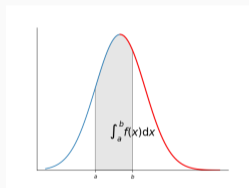


# Continuous random variable with density

## Definition

A random variable  $X : \Omega \rightarrow \mathbb{R}$  is **continuous with density** (absolutely continuous with respect to Lebesgues 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:

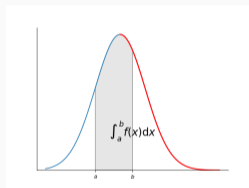
$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x f(u) du$$

# Continuous random variable with density

## Definition

A random variable  $X : \Omega \rightarrow \mathbb{R}$  is **continuous with density** (absolutely continuous with respect to Lebesgues 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:**

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

**Path:**

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

**Measure:** by integration along the path

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

## 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))
```

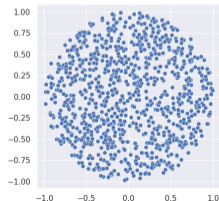
```
    d2 = x**2 + y**2
```

```
    observe(Gaussian(d2, 0.1), 0.5)
```

```
    return(x, y)
```

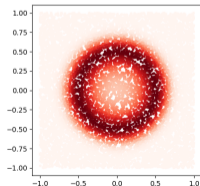
```
with ImportanceSampling(num_particles
```

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

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

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

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

## 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  $Y$

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

## 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:

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

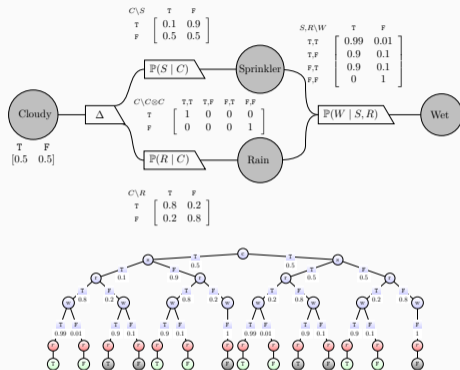
## Exercise: What are the Prior, Likelihood, Conditional measures

```
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)
```



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

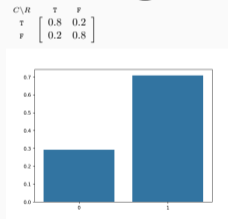
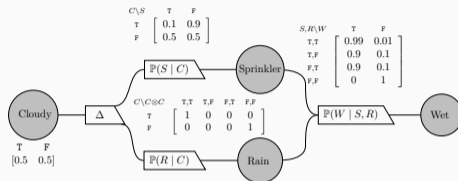
## Exercise: What are the Prior, Likelihood, Conditional measures

```
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)
```



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  $\sigma$ -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

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

$\forall V \in \mathcal{F}_{\mathcal{Y}}, \kappa(V|-) : \mathcal{X} \rightarrow \mathbb{R}^+$  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  $\mu$

## Giry Monad

$\mathcal{G}(\mathcal{X})$  is a measurable space when endowed with the  $\sigma$ -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

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

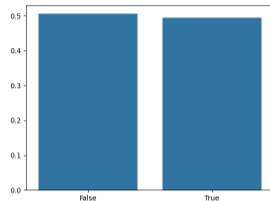
### Markov Kernels is Symmetric Monoidal

## 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 !



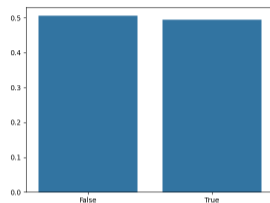
How can we prove it ?

## 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 ?

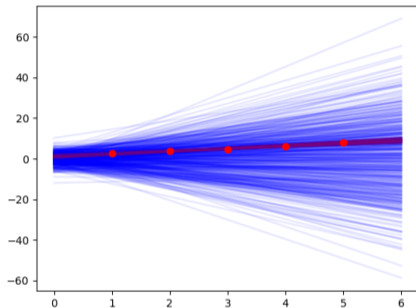
Recursive equation (if  $p \notin \{0, 1\}$ ):

$$\mathcal{F}(t) = p(1 - p)(\delta_T + \delta_F) + (1 - 2(p(1 - p))) t$$

$$\llbracket \text{FC} \rrbracket = \frac{1}{2}(\delta_T + \delta_F)$$

# 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  $\sigma$ -algebra

$\mathcal{Y} = (\mathbb{R}, \mathcal{C}(\mathbb{R}))$  the countable-cocountable  $\sigma$ -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  $\sigma$ -algebra

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

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

$h_x : y \mapsto \begin{cases} \delta_1 & \text{if } x = y \\ \delta_0 & \text{otherwise} \end{cases} \quad h_x^{-1}(\{p\delta_0 + (1-p)\delta_1 \mid p \in [0, 1] \cap [a, b]\})$  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,  $\Delta$  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)

**Contradition:** 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 2018b, “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

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

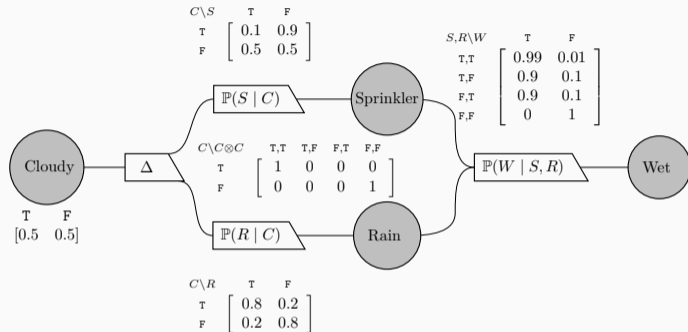
linearity and exponential are central in the semantics of Probabilistic Programming as already hinted in J.-Y. Girard 1988, "Normal functors, power series and  $\lambda$ -calculus".

# Linear Logic and Probability

---

Copy, discard

## Bayesian Network<sup>2</sup>: through copying<sup>3</sup> values



$$\mathbb{P}(C) : \Omega_{\text{bool}} = \{T, F\} \rightarrow \mathbb{R}^+$$

$$\mathbb{P}(S | C) : \Omega_{\text{bool}} \times \Omega_{\text{bool}} \rightarrow \mathbb{R}^+$$

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

<sup>2</sup>Pearl 1988, *Probabilistic reasoning in intelligent systems: networks of plausible inference*

<sup>3</sup>Cho and Jacobs 2019, "Disintegration and Bayesian inversion via string diagrams"

# Linear Logic and Probability

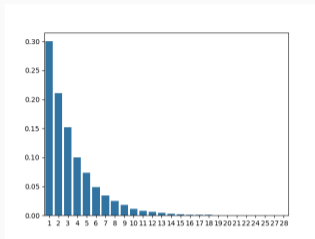
---

**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  $\mathbb{P}(\text{ST}(\text{Bernoulli}(0.9)))$

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

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

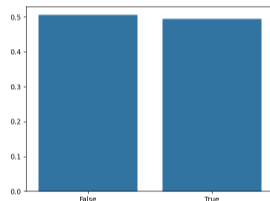
To compute the entire distribution  $\llbracket \text{StoppingTime}(\mu) \rrbracket$ , we sample **i.i.d. copies** of  $\mu$ .

## 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

$$\llbracket \text{let } x = \text{sample } N \text{ in } M \rrbracket = \sum_{a \in \mathbb{N}} \llbracket M \rrbracket_a \llbracket N \rrbracket_a \text{ or } \int_{r \in \mathbb{R}} (\llbracket M \rrbracket \circ \delta)(r) \llbracket N \rrbracket(dr)$$

- **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

# Linear Logic

---

From the probabilistic viewpoint

# Origins of Probabilistic Coherence Spaces

J.-Y. Girard 1988, “Normal functors, power series and  $\lambda$ -calculus” define Coherence Spaces, where values of type  $A$  are given by a carrier  $|X|$  and a closed program  $\vdash t : A$  as a part  $\llbracket t \rrbracket \in \mathcal{P}(|X|)$ . An environment  $x : A \vdash s : 1$  interacts **deterministically** with any closed program  $\vdash t : A$  when:

$$\llbracket s \rrbracket \perp \llbracket t \rrbracket \iff \# \llbracket t \rrbracket \cap \llbracket s \rrbracket \leq 1.$$

J.-Y. Girard 2004, “Between Logic and Quantic: a Tract” introduces Probabilistic Coherence Spaces, as a generalization of Coherence Spaces where subsets are replaced by factors:  $\llbracket t \rrbracket : |X| \rightarrow \mathbb{R}^+$ . An environment  $x : A \vdash s : 1$  interacts **probabistically** with any closed program  $\vdash t : A$  when:

$$\llbracket s \rrbracket \perp \llbracket t \rrbracket \iff \sum_{x \in |X|} \llbracket t \rrbracket_x \cdot \llbracket s \rrbracket_x \leq 1.$$

Danos and Ehrhard 2011, “Probabilistic coherence spaces as a model of higher-order probabilistic computation” gives explicit definition of all Linear Logic connectors and **fixpoint of types**. It studies mathematical properties of Probabilistic Coherence Spaces. It gives a model of **pure lambda-calculus** and **PCF with binary choice** and proves **adequacy**.

## Measures of Ground types: unit, booleans and integers

The **semantics** of basic random variables is their probability distribution<sup>4</sup>.

$$\llbracket \vdash \text{Bernoulli}(p) : \text{bool} \rrbracket = p \cdot \delta_T + (1 - p) \cdot \delta_F \quad \llbracket \vdash \text{RandInt}(n) : \text{int} \rrbracket = \sum_{k=1}^n \frac{1}{n} \cdot \delta_k$$

The **domain**  $\Omega_A$  is the set of potential output values of *terminating* probabilistic programs.

$$|1| = \{*\}$$

$$|\text{bool}| = \{T, F\}$$

$$|\text{int}| = \mathbb{N}$$

The set  $P(A)$  of **random elements** is the set of sub-probability distributions.

$$P(1) = \{p \cdot \delta_* \mid p \leq 1\} \quad P(\text{int}) = \left\{ \sum_{k \in \mathbb{N}} x_k \cdot \delta_k \mid \sum_{k \in \mathbb{N}} x_k \leq 1 \right\}$$

$$P(\text{bool}) = \{p \cdot \delta_T + q \cdot \delta_F \mid p + q \leq 1\}$$

⚠ The probability that a probabilistic program does not terminate can be non zero. ⚠

---

<sup>4</sup>The Dirac distribution  $\delta_a$  is 1 on  $a$  and 0 otherwise.

## (sub)Stochastic Matrices

```
def linear(d) → bool:  
  if sample(d):  
    return True  
  else:  
    return sample(Bernoulli(0.2))
```

linear transforms a boolean distribution into a boolean distribution:  $\llbracket \text{linear}(d) \rrbracket = \llbracket \text{linear} \rrbracket \cdot \llbracket d \rrbracket$ , where

$$\llbracket \text{bool} \vdash \text{linear} : \text{bool} \rrbracket = \begin{pmatrix} 1 & 0.2 \\ 0 & 0.8 \end{pmatrix}$$

A **Linear Map** is a matrix  $M \in (\mathbb{R}^+)^{\Omega_A \times \Omega_B}$  that preserves random elements:

$$\forall x \in P(A) \quad M \cdot x = \sum_{b \in \Omega_B} \sum_{a \in \Omega_A} M_{a,b} x_a \cdot \delta_b \in P(B)$$

⚠ Substochastic Matrices are not subprobability distributions. ⚠

# Probabilistic Coherence Spaces (PCS) from Probabilistic testing

⚠ We do **NOT** use probabilistic monad to add externally the effect to the calculus, but probability is **intrinsic** to the model.

**Probabilistic semantics**  $P(X)$  encodes probabilistic computation intrinsically.

$P(\text{bool})$  are subprobability boolean distributions

Ensure substochastic matrices composition with random elements is well defined, i.e. sums are converging.

**Probabilistic tests** (aka orthogonality<sup>5</sup>):  $u, u' \in (\mathbb{R}^+)^{\Omega_X}$ .  $u \perp u' \iff \sum_{a \in \Omega_X} u_a u'_a \leq 1$

$$P(X)^\perp = \left\{ u' \in (\mathbb{R}^+)^{\Omega_X} \mid \forall u \in P(X) \sum_{a \in \Omega_X} u_a u'_a \leq 1 \right\} = \{ u' \mid \forall u \in P(X) u' \circ u \in P(1) \}$$

---

<sup>5</sup>Reminiscent of functional analysis curves and to logical relations.

# Linear Category Pcoh

**A PCS is  $X = (|X|, P(X))$  such that  $|X|$  is countable,  $P(X) \subseteq (\mathbb{R}^+)^{|X|}$  and  $P(X)^{\perp\perp} = P(X)$ ,**

for each  $a \in |X|$  there exists  $u \in P(X)$  such that  $x_a > 0$  (Coverage),

for each  $a \in |X|$  there exists  $A > 0$  such that  $\forall x \in P(X) \ x_a \leq A$  (Boundedness).

**A morphism of PCSs from  $X$  to  $Y$  is a matrix  $t \in (\mathbb{R}^+)^{|X| \times |Y|}$  which maps  $P(X)$  to  $P(Y)$ .**

$$\forall u \in P(X) \ t u \in P(Y) \iff \forall v' \in P(Y)^\perp, \quad \sum_{(a,b) \in |X| \times |Y|} t_{a,b} u_a v'_b \leq 1.$$

## Identity:

The diagonal matrix  $\text{id} \in (\mathbb{R}^+)^{|X| \times |X|}$ , given by  $\text{id}_{a,b} = 1$  if  $a = b$

and  $\text{id}_{a,b} = 0$  otherwise.

$$A \vdash A$$

## Composition

Matrix multiplication, let  $s \in \mathbf{Pcoh}(X, Y)$  and  $t \in \mathbf{Pcoh}(Y, Z)$ ,

$$\text{then } (ts)_{a,c} = \sum_{b \in |Y|} s_{a,b} t_{b,c}.$$

$$\frac{\Gamma \vdash B \quad \Delta, B \vdash A}{\Gamma, \Delta \vdash A}$$

# Linear Logic

---

## Multiplicative fragment<sup>a</sup>

---

<sup>a</sup> J. Girard 1987, "Linear Logic"

# Pcoh is Symmetric Monoidal Closed: model of intuitionistic multiplicative LL

## Joint Distributions

`def joint() → int x int:` is interpreted by the  
joint distribution:  
`a = sample(x)`  
`b = sample(y)`  
`return (a, b)`  
$$x \otimes y = \sum_{\substack{a \in \Omega_X \\ b \in \Omega_Y}} x_a y_b \cdot \delta_{(a,b)}$$

$$\frac{}{\vdash 1} \quad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}$$

$$\frac{\Gamma \vdash A}{\Gamma, 1 \vdash A} \quad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}$$

## Tensor Product (*Associativity, Symmetry, unitor*)

$$|X \otimes Y| = |X| \times |Y|$$

$$P(X \otimes Y) = \{u \otimes v \mid u \in P(X) \text{ and } v \in P(Y)\}^{\perp\perp}$$

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}$$

## Closed Structure (*Evaluation and Curryfication*)

$$|X \multimap Y| = |X| \times |Y|$$

$$P(X \multimap Y) = \left\{ t \in (\mathbb{R}^+)^{|X| \times |Y|} \mid \forall u \in P(X) \, t u \in P(Y) \right\}$$

$$\frac{\Gamma \vdash A \multimap B \quad \Delta \vdash A}{\Gamma, \Delta \vdash B}$$

## Additive fragment and examples of Base Types

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \times B}$$

$$\frac{}{\Gamma \vdash \top}$$

**Product:**  $|\top| = \emptyset$

$$\left| \prod_{i \in I} X_i \right| = \bigcup_{i \in I} \{i\} \times |X_i| \quad P\left(\prod_{i \in I} X_i\right) = \{u \mid \forall i \in I \ u(i) \in P(X_i), \text{ where } \forall a \in |X_i| \ u(i)_a = u_{(i,a)}\}$$

**Enumeration:**  $\text{bool} = 1 \oplus 1$ .

$$\left| \bigoplus_{i \in I} X_i \right| = \bigcup_{i \in I} \{i\} \times |X_i| \quad P\left(\bigoplus_{i \in I} X_i\right) = \left\{ u \mid \forall i \in I \ u(i) \in P(X_i) \text{ and } \sum_{i \in I} \|u(i)\|_{X_i} \leq 1 \right\}$$

**Type Fixpoints:**  $\text{int} = 1 \oplus \text{int}$ .

PCSs is a CPO with least element  $0 = \top$  with  $|0| = \emptyset$  when ordered by

$$X \subseteq Y \quad \text{iff} \quad |X| \subseteq |Y| \quad \text{and} \quad P(X) = \{v_{||X|} \mid v \in P(Y)\}$$

## Properties of $P(X)$

$P(X)$  is **unitary**  $\|u\|_X \in [0, 1]$  for all  $u \in P(X)$  where the **norm** is defined as

$$\|u\|_X = \sup \left\{ \sum_{a \in |X|} u_a u'_a \mid u' \in P(X^\perp) \right\}$$

$P(X)$  is a **cone**

$$\forall u, v \in P(X) \forall \alpha, \beta \in \mathbb{R}^+ \quad \alpha + \beta \leq 1 \Rightarrow \alpha u + \beta v \in P(X).$$

$P(X)$  is an  **$\omega$ -continuous domain** where the **partial order** is defined as

$$u \leq v \quad \text{iff} \quad \forall a \in |X| \quad u_a \leq v_a \in \mathbb{R}^+.$$

**Pcoh** is CPO enriched indeed, **Pcoh** $(X, Y) = P(X \multimap Y)$  is a CPO with 0 as least element.

## Back to the StoppingTime Example

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

Let ST be StoppingTime(Bernoulli(p)).

After  $k$  iterations,

$$\llbracket \text{ST} \rrbracket^1 = (1 - p)\delta_1$$

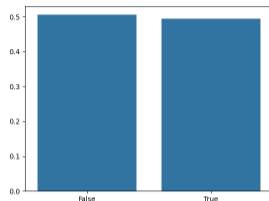
$$\llbracket \text{ST} \rrbracket_t^{k+1} = p \llbracket \text{ST} \rrbracket_{t-1}^k + (1 - p) \llbracket \text{ST} \rrbracket_t^k$$

$\llbracket \text{ST} \rrbracket$  is the lub of the increasing sequence  $\llbracket \text{ST} \rrbracket^k$  in  $P(\text{int}) \subseteq (\mathbb{R}^+)^{\mathbb{N}}$

## 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))
```



Let FC be FairCoin(Bernoulli(p)).

Recursive Equation

$$\mathcal{F}(t) = p(1 - p)(\delta_T + \delta_F) + (1 - 2(p(1 - p))) t$$

$$\llbracket \text{FC} \rrbracket = \frac{1}{2}(\delta_T + \delta_F)$$

when  $p \notin \{0, 1\}$

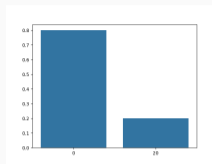
# Linear Logic

---

Exponential fragment

## Compare Linear vs Non-Linear

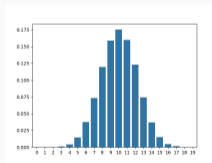
```
def FlipSum(n: int, d) → int:  
    x = sample(d)  
    sum = 0  
    for k in range(n):  
        sum += x  
    return sum
```



$$\llbracket \text{FlipSum}(n, \text{Bernoulli}(p)) \rrbracket = (1 - p) \cdot \delta_0 + p \cdot \delta_n$$

$\text{FlipSum}(n, \text{Bernoulli}(p))$  samples once  $\text{Bernoulli}(p)$  and copy its outcomes  $n$  times.

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



$$\llbracket \text{RandWalk}(n, \text{Bernoulli}(p)) \rrbracket = \sum_{k=0}^n \binom{n}{k} p^k (1 - p)^{n-k} \cdot \delta_k$$

$\text{RandomWalk}(n, \text{Bernoulli}(p))$  samples  $n$  *i.i.d.* copies of  $\text{Bernoulli}(p)$ .

# Exponential of measures

The **exponential**  $!X$  of a ground type represents the type of bags of *i.i.d.* copies of measures.

$$x^! = \sum_{m \in \mathfrak{M}_{\text{fin}} \Omega X} \prod_{a \in \text{supp}(m)} x_a^{m(a)} \cdot \delta_m$$

For instance,

$$\llbracket \text{Bernoulli}(p) \rrbracket^! = (p \cdot \delta_T + (1 - p) \cdot \delta_F)^! = \sum_{n \in \mathbb{N}} \sum_{k \in \mathbb{N}} p^k (1 - p)^{n-k} \cdot \delta_{[T \mapsto k, F \mapsto n-k]}$$

**Exponential PCS:**  $\Omega_{!X} = \mathfrak{M}_{\text{fin}} \Omega_X$  and  $P(!X) = \{x^! \mid x \in P(X)\}^{\perp\perp}$ .

## The exponential comonad in Pcoh

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}$$

$$\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}$$

$$\frac{\Gamma \vdash B}{\Gamma, !A \vdash B}$$

$$\frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}$$

**Comonad:** ! with counit  $\text{der } X \in \mathbf{Pcoh}(!X, X)$  and comultiplication  $\text{dig}^X \in \mathbf{Pcoh}(!X, !!X)$ .

**Promotion:** For  $t \in \mathbf{Pcoh}(!X, Y)$ ,  $t^! \in \mathbf{Pcoh}(!X, !Y)$  is defined such that

$$t^! u^! = (t u)^!$$

**Strength** given by isomorphisms from  $(\mathbf{Pcoh}, \times)$  to  $(\mathbf{Pcoh}, \otimes)$ ,

$$m^0 \in \mathbf{Pcoh}(!\top, 1) \quad \text{and} \quad m^2 \in \mathbf{Pcoh}(!(X \& Y), !X \otimes !Y)$$

**!X is a Commutative Comonoid.**

$$\text{contr}^{!X} \in \mathbf{Pcoh}(!X, !X \otimes !X) \quad \text{and} \quad \text{weak}^{!X} \in \mathbf{Pcoh}(!X, 1)$$

## Eilenberg Moore and Kleisli Categories: copy, delete and i.i.d. copies

**Eilenberg-Moore:** A *Coalgebra* is a PCS  $Q$  with  $h \in \mathbf{Pcoh}(Q, !Q)$  compatible with the  $!$ -comonad structure.

Every coalgebra  $Q$  comes with marginalization, copy and delete

$$\pi_Q \in \mathbf{Pcoh}(Q \otimes Q', Q) \quad \text{and} \quad \Delta_Q \in \mathbf{Pcoh}(Q, Q \otimes Q) \quad \text{and} \quad \omega_Q \in \mathbf{Pcoh}(Q, 1)$$

Value types are interpreted as coalgebras

$$\varphi, \psi := \text{unit} \mid !\sigma \mid \varphi \otimes \psi \mid \varphi \oplus \psi \mid \zeta \mid \text{Rec } \zeta \varphi$$

**Kleisli Category  $\mathbf{Pcoh}_!$**  with PCSs as objects and morphisms  $\mathbf{Pcoh}_!(X, Y) = \mathbf{Pcoh}(!X, Y)$  are power series:  $t \in \mathbf{Pcoh}_!(X, Y)$  iff  $t \in (\mathbb{R}^+)^{\mathfrak{M}_{\text{fin}}(X) \times Y}$  and

$$\forall u \in P(X) \quad t(u) = t u^! = \left( \sum_{m \in \mathfrak{M}_{\text{fin}}(X)} t_{m,b} \prod_{a \in |X|} u_a^{m(a)} \right)_{b \in |Y|} \in P(Y)$$

*A non-linear morphism creates an unbound bag of i.i.d. copies of its input distribution.*

# Linear Logic

---

## Semantics of Probabilistic Programming

# Models of Linear Logic<sup>6</sup>

## One axiomatization:

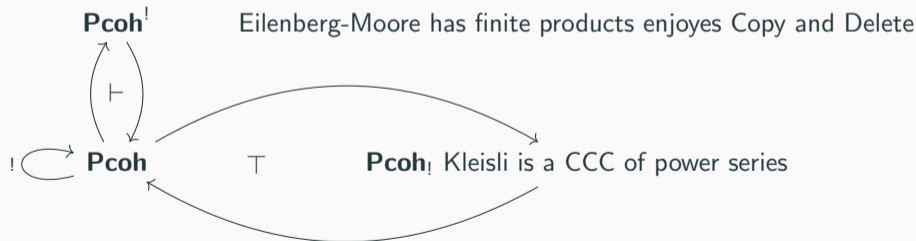
Linear Category **Pcoh**

$(\mathbf{Pcoh}, 1, \otimes, \multimap)$  is an SMCC

$(\mathbf{Pcoh}, 0, \times)$  has finite product

A comonad  $!$  which is strong monoidal from  $(\mathbf{Pcoh}, 0, \times)$  to  $(\mathbf{Pcoh}, 1, \otimes)$

## Adjunctions:



<sup>6</sup>Melliès 2009, "Categorical semantics of linear logic"

# Pcoh models probabilistic programming

## Syntaxes

Probabilistic Call-By-Push-Value.

Probabilistic Call-By-Value.  $A \rightarrow B$  encoded by  $!(A \multimap B)$ .

Probabilistic Call-By-Name.  $A \Rightarrow B$  encoded by  $(!A) \multimap B$ .

Probabilistic PCF. Fixpoint operator:  $\mathcal{Y} \in \mathbf{Pcoh}_!(X \Rightarrow X, X)$

Probabilistic Untyped Calculus. The reflexive object satisfies  $D = (!D^{\mathbb{N}})^{\perp}$ .

## Soundness

$$\llbracket M \rrbracket = \sum_{M'} \mathbb{P}(M \rightarrow M') \llbracket M' \rrbracket$$

**Adequate and Fully Abstract** for pPCF<sup>7</sup> and pCBPV<sup>8</sup>.

---

<sup>7</sup>Ehrhard, Pagani, and Tasson 2018a, “Full Abstraction for Probabilistic PCF”

<sup>8</sup>Ehrhard and Tasson 2019, “Probabilistic call by push value”

# Continuous distributions

---

Measurable space and kernels

## Wish List for Higher-Order Probabilistic Programs

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

$$\llbracket \text{let } x = \text{sample } N \text{ in } M \rrbracket = \sum_N \llbracket M \rrbracket_a \llbracket N \rrbracket_a \text{ or } \int_{\mathbb{R}} (\llbracket M \rrbracket \circ \delta)(r) \llbracket N \rrbracket(dr)$$

- **Tensor product** to interpret joint distributions
- **Copy and Discard** to duplicate Samples
- **Bags of i.i.d. copies** to duplicate distributions at will
- **Higher-order** for recursive programs

# 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  $\sigma$ -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

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

$\forall V \in \mathcal{F}_{\mathcal{Y}}, \kappa(V|-) : \mathcal{X} \rightarrow \mathbb{R}^+$  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  $\mu$

## Giry Monad

$\mathcal{G}(\mathcal{X})$  is a measurable space when endowed with the  $\sigma$ -algebra generated by the evaluations on measure sets  $\delta_U : \mu \mapsto \mu(U) : \delta_U^{-1}(a, b)$  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:**  $\mathbf{Kern}(\mathcal{X}, \mathcal{Y}) = \mathbf{Meas}(\mathcal{X}, \mathcal{G}(\mathcal{Y}))$

Kernels are measures transformers thanks to the giry monad bind

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

# Markov Kernel (Symmetric Monoidal, Not Closed)

## Monoidal unit:

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

## 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

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

**Aumann's Lemma** implies that 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.

# Continuous distributions

---

Integrable cones

## Cones to account for Measures

Following Selinger, we define a cone endowed with the structure inspired by the space of measures  $\mathcal{G}(\mathcal{X})$  on a measurable space  $\mathcal{X}$ , with a norm  $\|\mu\|_{\mathcal{G}(\mathcal{X})} = \mu(\mathcal{X})$ .

**Algebraic structure:** A cone  $P$  is an  $\mathbb{R}^+$  semi-module such that:

$$x + y = 0 \Rightarrow x, y = 0 \qquad x + y = x + y' \Rightarrow y = y'$$

**Total variation:** A cone is equipped with a norm  $\| \cdot \|_P : P \rightarrow \mathbb{R}^+$  such that:

$$\|x + x'\|_P \leq \|x\|_P + \|x'\|_P, \quad \|\alpha x\|_P = \alpha \|x\|_P \quad \|x\|_P = 0 \Rightarrow x = 0, \quad \|x\|_P \leq \|x + x'\|_P$$

**Probability:** The unit ball  $\mathcal{B}P = \{x \in P \mid \|x\|_P \leq 1\}$  account for (sub)probability distribution.

**Complete:** A cone is complete for the order:  $x \leq_P x'$  if there is  $y \in P$  such that  $x' = x + y$ .

For any monotone  $(x_n)_{n \in \mathbb{N}}$  with  $\forall n \ \|x_n\|_P \leq 1$  has a lub  $x = \sup_{n \in \mathbb{N}} x_n$  such that  $\|x\|_P \leq 1$ .

**Examples:**  $1 = \mathbb{R}^+$

$\mathcal{G}(\mathcal{X})$  is the cone of finite measures over a measurable space  $\mathcal{X}$ .

If  $\mathcal{X}$  is a PCS, then  $\widehat{\mathcal{X}} = \{x \in (\mathbb{R}^+)^{|\mathcal{X}|} \mid \exists \alpha > 0 \ \alpha x \in P(\mathcal{X})\}$  is a cone.

## Linear maps to account for Kernels: $\mathbf{ICone}(P, Q)$

Following PCSs, we define linear maps inspired by stochastic matrices and kernels.

**Linear:** a function  $f : P \rightarrow Q$  is linear if  $f(\alpha_1 x_1 + \alpha_2 x_2) = \alpha_1 f(x_1) + \alpha_2 f(x_2)$ .

**Stochastic:** A substochastic function is a linear function  $f : P \rightarrow Q$  such that  $\|f\| \leq 1$ , where

$$\|f\| = \sup_{x \in \mathcal{B}P} \|f(x)\|_Q \in \mathbb{R}^+$$

**Continuous:** If  $f : P \rightarrow Q$  is linear, then it is bounded. A **continuous** function is such that for all monotone and bounded  $(x_n \in P)_{n \in \mathbb{N}}$ ,  $f(\sup_{n \in \mathbb{N}} x_n) = \sup_{n \in \mathbb{N}} f(x_n)$ .

**Example:** Every stochastic kernel  $\kappa : \mathcal{X} \rightsquigarrow \mathcal{Y}$  between measurable spaces induces a linear stochastic continuous function between the cones of measures:

$$\begin{aligned} \hat{\kappa} : \mathcal{G}(\mathcal{X}) &\rightarrow \mathcal{G}(\mathcal{Y}) \\ \mu &\mapsto \lambda U. \int_{x \in \Omega_{\mathcal{X}}} \kappa(U \mid x) \mu(dx) \end{aligned}$$

# Continuous distributions: Random paths

```
def sum(d) → int:
```

```
    a = sample(d)
```

```
    b = sample(d)
```

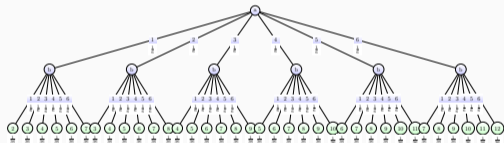
```
    return a + b
```

```
plt.show()
```

```
def SqDist() → float:
```

```
    x = sample(Uniform(-1, 1))
```

**Measure:** Weighted Random traces:



**Sum** over all random traces:

$$\llbracket \text{Dice} \rrbracket_k = \sum_{1 \leq a, b \leq 6} \frac{1}{36} \delta_{a+b}(k)$$

**Measures<sup>a</sup>:** Random paths and real measures:

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

$$\mu(dr_1 \otimes dr_2) = \mathbb{1}_{[-1,1]}(r_1) dr_1 \mathcal{N}(0,1)(r_2) dr_2$$

**Integration** along the path:

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

<sup>a</sup>Reminiscent of QBS random elements and measures.

## Random Paths and Measurability tests

**Examples:** If  $\mathcal{X}$  is a PCS,  $P(\mathcal{X}) = P(\mathcal{X})^{\perp\perp}$  implies that measurability is determined by tests against  $P(\mathcal{X})^{\perp} = P(\mathcal{X}) \multimap 1$ .

**Random path:**  $\text{Path}(\mathbb{R}^\ell, P)$ :  $\gamma : \mathbb{R}^\ell \rightarrow P$  functions such that for any  $k$ -measurability test  $m$  :,

$$(s, r) \mapsto m(\gamma(r) \mid s) \in \mathbf{Meas}(\mathbb{R}^k \times \mathbb{R}^\ell, 1)$$

**Measurability tests:**  $m : \mathbb{R}^k \times P \multimap 1$  with  $k$  random sources such that:

$$\forall r \in \mathbb{R}^k \quad m(- \mid r) \in \mathbf{Cone}(P, \mathbb{R}^+) \quad \forall x \in P \quad m(x \mid -) \in \mathbf{Meas}(\mathbb{R}^n, \mathbb{R}^+)$$

**Measurability structure**  $\mathcal{M} = (\mathcal{M}_k)_{k \in \mathbb{N}}$  a collection of measurability tests such that:

**Precomposition:** if  $m \in \mathcal{M}_n$  and  $\varphi \in \mathbf{Meas}(\mathbb{R}^k, \mathbb{R}^n)$ , then  $m(\varphi(-) \mid -) \in \mathcal{M}_k$

**Separation:** if  $x \neq y \in P$ , there is  $m \in \mathcal{M}_0$  such that  $m(x) \neq m(y)$

**Norm:**  $\|x\|_P = \sup_{m \in \mathcal{M}_0} \left( \frac{m(x)}{\|m\|} \right)$

## Measurable functions and Integration<sup>9</sup>: ICone

**Measurable function:** linear continuous functions  $f : P \rightarrow Q$  preserving random paths:

$$\forall \gamma \in \text{Path}(\mathbb{R}^\ell, P) \quad f \circ \gamma \in \text{Path}(\mathbb{R}^\ell, Q)$$

**Measure:**  $[\mu, \alpha]_d$  where  $\mu \in \mathcal{G}(\mathbb{R}^\ell)$  and  $\gamma \in \text{Path}(\mathbb{R}^\ell, P)$  is **integrable** iff there is

$$x = \int \alpha(r) \mu(dr) \quad \text{such that} \quad \forall x' \in P \multimap \mathbb{R}^+ \quad \langle x', x \rangle = \int_{\mathbb{R}^+} x' \circ \alpha(r) \mu(dr)$$

**Integrable Cone C:** complete cone with measurability structure, where all  $[\mu, \alpha]_d$  are integrable.

**Integrable linear map of cones:** measurable functions  $f \in \mathbf{ICone}(C, D)$  preserving integrals:

$$f \left( \int \alpha(r) \mu(dr) \right) = \int f(\alpha(r)) \mu(dr)$$

**Example:**  $\mathcal{G}(\mathbb{R}^\ell)$  is integrable. If  $\kappa \in \text{Path}(\mathbb{R}^\ell, \mathcal{G}(\mathbb{R}^k)) = \mathbf{Kern}(\mathbb{R}^\ell, \mathbb{R}^k)$  and  $\mu \in \mathcal{G}(\mathbb{R}^\ell)$ , then  $\nu = \int \kappa(- \mid s) \mu(ds) \in \mathcal{G}(\mathbb{R}^k)$

---

<sup>9</sup>Pettis 1938, "On Integration in Vector Spaces"

# Wish List for Higher-Order Probabilistic Programs with continuous distributions

## ICone, a solution to the measurability problem

- ✓ **Measures:**  $\mathcal{G}(\mathbb{R})$  to interpret close programs of type float as measures
- ✓ **Stochastic kernels** (parameterized measures) to interpret programs sampling in measures given as parameters
- ✓ **Integration** to interpret sampling in real distributions

$$\llbracket \text{let } x = \text{sample } d \text{ in } t \rrbracket = \int_{\mathbb{R}} (f \circ \delta)(r) \mu(dr)$$

- ☐ **Tensor product** to interpret joint distribution
- ☐ **SMCC** to account for Higher-Order
- ☐ **Copy and Discard** of distributions
- ☐ **Bags of i.i.d. copies** of distributions

# Linear, stable and non-linear maps of integrable cones

Let  $P$  and  $Q$  be integrable cones.

**Internal Linear Hom:**  $P \multimap Q$  the complete cone of linear, continuous, measurable and integrable functions with  $\|f\|_{P \multimap Q} = \sup_{x \in \mathcal{B}P} \|f(x)\|_Q$ . Equipped with the measurable structure of  $Q$  and random paths of  $P$  by pre and postcomposition.

Integrals are defined pointwise: for any path  $\gamma : \mathbb{R}^k \rightarrow P \multimap Q$ ,  $f = \int_{P \multimap Q} \gamma d\mu$  is such that

$$f(x) = \int_{\mathbb{R}^k} \gamma(x \mid r) \mu(dr)$$

**Stable maps** continuous totally monotone (ultra-convex).

**Analytic maps**  $\lambda x. \sum_{n=0}^{\infty} f_n(x, \dots, x)$  such that  $f_n$  is  $n$ -linear continuous, measurable and separately integrable of norm  $\leq 1$ .

**Internal Non-Linear Hom:**  $P \Rightarrow_a Q$  the complete cone of analytic maps is an integrable cone.

## Linear Logic model:

**ICone is an SMCC:** Integrable cones and linear maps is complete.

$P \multimap -$  preserves all limits. By the Special Adjoint Functor Theorem, it has a left adjoint, the tensor product  $- \otimes P$ .

**ACone is a CCC** as the evaluation and curryfication are analytic.

**Exponential:** the embedding **ICone**  $\hookrightarrow$  **ACone** preserves all limits. By the SAFT, it has a left adjoint.

The *adjunction* induces a comonad  $!$  on **ICone** together with

a unit:  $\eta \in \mathbf{ACone}(P, !P)$       a counit:  $\varepsilon \in \mathbf{ICone}(!P, P)$       a comultiplication:  $\vartheta : !P \rightarrow !!P$



The comonad  $(!, \varepsilon, \vartheta)$  is strong monoidal, thus:

$$\omega \in \mathbf{ICone}(!P, 1) \quad \Delta \in \mathbf{ICone}(!P, !P \otimes !P)$$

## Copy, delete and bags of i.i.d. copies

**Eilenberg-Moore**  $!$ -coalgebras  $(P, h)$  are symmetric comonoids: there are coalgebraic morphisms

$$\omega \in \mathbf{ICone}(C, 1) \quad \Delta \in \mathbf{ICone}(C, C \otimes C)$$

**Example:** Bags of *i.i.d.* copies of distributions.

For any measurable space  $\mathcal{X}$ ,  $\mathcal{G}(\mathcal{X})$  is a  $!$ -coalgebra, with

$$\begin{aligned} \mathcal{G}(\mathcal{X}) &\rightarrow !\mathcal{G}(\mathcal{X}) \\ \mu &\mapsto \int_{R^k} \eta \circ \delta(r) \mu(dr) \end{aligned}$$

It is well defined as the dirac  $\delta \in \mathbf{Path}(\mathbb{R}^k, \mathcal{G}(\mathcal{X}))$  and  $\eta \in \mathbf{ACone}(\mathcal{G}(\mathcal{X}), !\mathcal{G}(\mathcal{X}))$  is measurable, thus  $\eta \circ \delta \in \mathbf{Path}(\mathbb{R}^k, !\mathcal{G}(\mathcal{X}))$ .

# Wish List for Higher-Order Probabilistic Programs

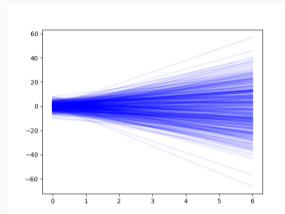
- ✓ **Measures**: to interpret close programs of ground types unit, booleans, integers, reals
- ✓ **Stochastic matrices or kernels** (parameterized measures) to interpret programs sampling in measures given as parameters
- ✓ **Sum or Integration** to interpret sampling

$$\llbracket \text{let } x = \text{sample } N \text{ in } M \rrbracket = \sum_{a \in \mathbb{N}} \llbracket M \rrbracket_a \llbracket N \rrbracket_a \text{ or } \int_{r \in \mathbb{R}} (\llbracket M \rrbracket \circ \delta)(r) \llbracket N \rrbracket(dr)$$

- ✓ **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

## Example - Higher-Order

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



**Measure space:**  $\mathbb{R}^2$  with borelians

**Probability**  $\mu \in \text{Meas}(\mathbb{R}^2) : m, b \sim \mathcal{N}(0, 2) \otimes \mathcal{N}(0, 2)$

**Random variable:**  $\alpha : (m, b) \mapsto \lambda x. m * x + b \in \text{Path}(\mathbb{R}^2, \mathbb{R} \Rightarrow_a \mathbb{R})$

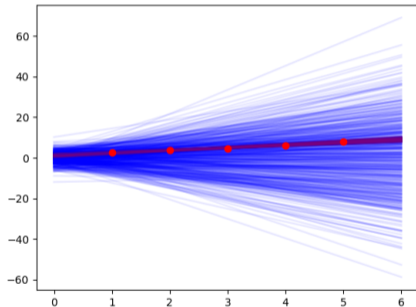
**Distribution:**  $\llbracket \text{model} \rrbracket = \int_{r \in \mathbb{R}^2} \alpha(r) \mu(dr)$  is in  $\mathbb{R} \Rightarrow_a \mathbb{R}$

$$\llbracket \text{let fs} = \text{sample}(\text{dist}) \text{in } 2 * \text{fs} \rrbracket = \int_{\mathbb{R}^2} 2 * \alpha(m, b) \mu(dm, db)$$

$$\llbracket \text{let fs} = \text{sample}(\text{dist}) \text{in } 2 * \text{fs} \rrbracket (4) = \int_{\mathbb{R}^2} (4 * m + b) \mathcal{N}(0, 2)(dm) \mathcal{N}(0, 2)(db)$$

# Linear Regression<sup>10</sup>

```
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)]
```



<sup>10</sup>Heunen et al. 2017, "A convenient category for higher-order probability theory"

**Aumann's Lemma:** Higher-order and effects is subtle mixture.

**Probabilistic Coherent Spaces** is a great model to study higher-order probabilistic programming with countable distributions.

**Cones** is a great model to study higher-order probabilistic programming with continuous distributions.

**Linear Logic** is a great tool box to understand resources.

**Semantics** helps understanding mathematical structure of computations leading to the design of elegant programming languages.

# Bibliography

Kozen 1979, “Semantics of Probabilistic Programs”; Danos and Ehrhard 2011, “Probabilistic coherence spaces as a model of higher-order probabilistic computation”; J.-Y. Girard 1988, “Normal functors, power series and  $\lambda$ -calculus”; Panangaden 1998, “The Category of Markov Kernels”; Giry 1982, “A categorical approach to probability theory”; Lawvere 1962, “The category of probabilistic mappings”; Ehrhard and Geoffroy 2025, “Integration in Cones”; Ehrhard, Pagani, and Tasson 2018b, “Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming”