Probabilistic Programming Languages

Hugo Paquet

MPRI 25–26

Recap

- **Probabilistic programming**: a language for building probabilistic models, including samples and observations.
- Monte Carlo methods for Bayesian inference: importance sampling, particle filters, etc.
- With Monte Carlo methods, the result is always an **approximation.** This is often necessary because the true posterior distribution is generally uncomputable.

Today

- Semantics for **discrete** probabilistic programs using monads (including a small amount of category theory).
- We add higher-order function types, e.g. $(\mathrm{Nat} \to \mathrm{Nat}) \to \mathrm{Nat}$.
- When the distributions have finite support we can perform **exact inference**.
- Other efficient representations of discrete distributions: probabilistic graphical models.

1. Semantics for discrete probabilistic programs using monads

The need for semantics

- Semantics provides a formal connection between probabilistic programs and the statistical models they represent.
- Using semantics we can reason about program equivalence, e.g. there could be several implementations of the same probability distribution (which may not be equally efficient for inference).

```
let x = normal 0 2 in
return x
```

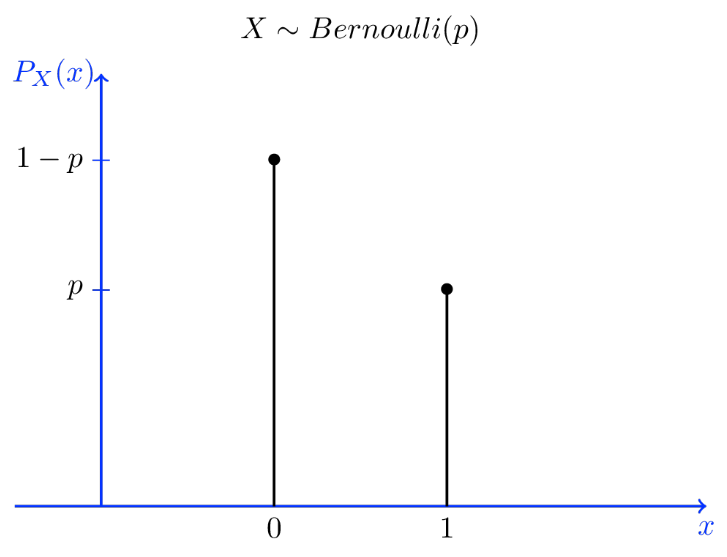
=

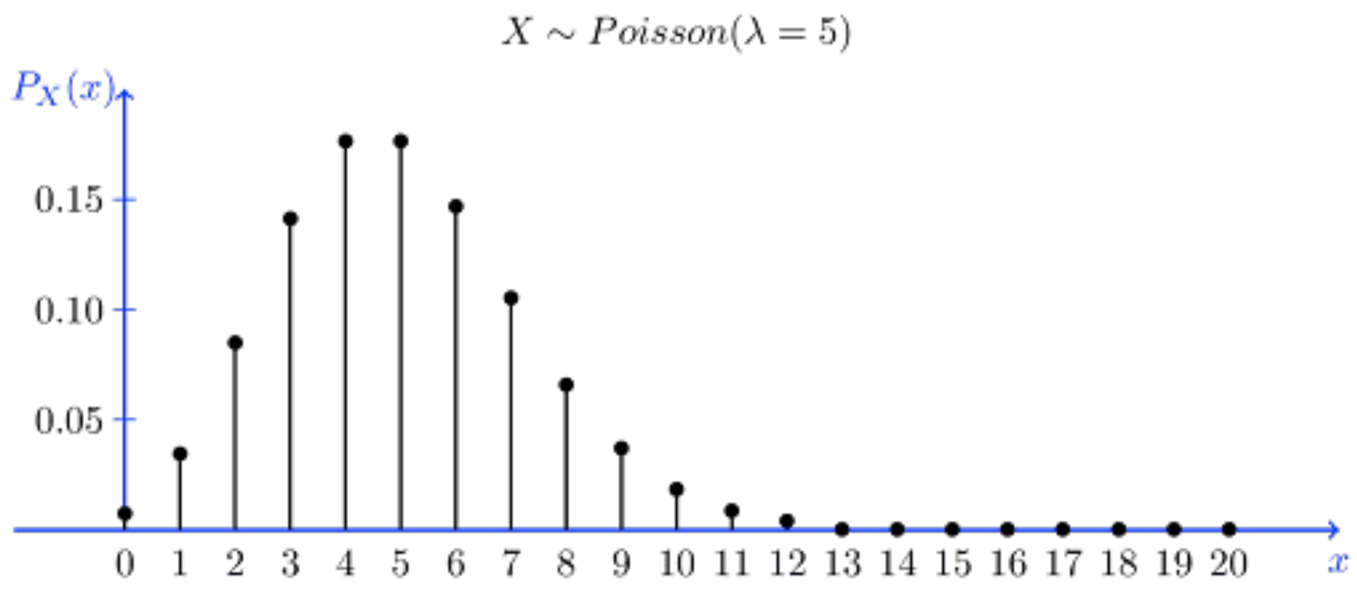
```
let y = normal 0 1 in
let z = normal 0 1 in
return (y + z)
```

- **Correctness of inference.** Inference algorithms often approximate. But what are they trying to approximate?

Discrete probability and measure theory

Discrete probability is based on sets X and distributions functions $X \to [0,1]$.





It is a special case of measure-theoretic probability:

- A measurable space (X, Σ_X) is called **discrete** if $\Sigma_X = \mathscr{P}(X)$. Discrete measurable spaces are in correspondence with sets.
- A probability measure $\mu : \Sigma_X \to [0,1]$ on a discrete space is completely determined by its value on the set of **atoms** $\{x \in X \mid \mu\{x\} > 0\}$, and therefore by the a probability mass function $f_\mu : X \to [0,1]$.
- Probability measures have at most countably many atoms. (True even for unnormalized finite measures but not for general measures.)

A probabilistic language

Types.

$$A, B ::= \text{Unit} \mid \text{Bool} \mid \text{Nat} \mid \text{Real} \mid A \times B \mid A \to B$$

Terms.

basic operations +, >, etc.

$$s, t ::= () \mid b \in \{\text{True}, \text{False}\} \mid r \in \mathbb{R} \mid n \in \mathbb{N} \mid \text{op}$$

$$\langle s, t \rangle \mid \text{fst } s \mid \text{snd } s \mid \text{if } s \text{ then } t \text{ else } t'$$

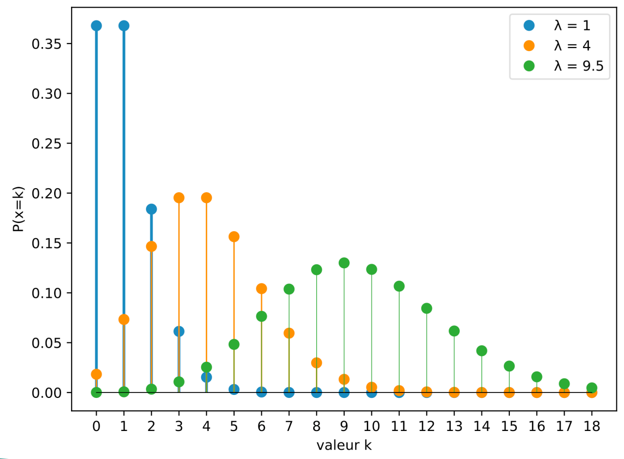
$$\mid x \mid \lambda x.\, s \mid s \ t$$

$$\text{return } s \mid \text{let } x = s \text{ in } t$$

$$\text{bernoulli } s \mid \text{poisson } s$$

poisson(λ) = a distribution on $\mathbb{N}$ with infinite support, defined when $\lambda > 0$.

$$\Pr(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}$$



A higher-order probabilistic language

$$\frac{}{\Gamma, x : A \vdash^{\mathrm{D}} x : A} \qquad \frac{\Gamma, x : A \vdash^{\mathrm{P}} t : B}{\Gamma \vdash^{\mathrm{D}} \lambda x.\, t : A \to B} \qquad \frac{\Gamma \vdash^{\mathrm{D}} t : A \to B \qquad \Gamma \vdash^{\mathrm{D}} s : A}{\Gamma \vdash^{\mathrm{P}} t\, s : B}$$

$$\frac{\Gamma \vdash^{\mathrm{D}} s : A \qquad \Gamma \vdash^{\mathrm{D}} t : B}{\Gamma \vdash^{\mathrm{D}} \langle s, t \rangle : A \times B} \qquad \frac{\Gamma \vdash^{\mathrm{D}} s : A \times B}{\Gamma \vdash^{\mathrm{D}} \mathrm{fst}\ s : A} \qquad \frac{\Gamma \vdash^{\mathrm{D}} s : A \times B}{\Gamma \vdash^{\mathrm{D}} \mathrm{snd}\ s : B}$$

$$\frac{\Gamma \vdash^{\mathrm{D}} s : \mathrm{Bool} \quad \Gamma \vdash^{\mathrm{P}} t : \mathrm{A} \quad \Gamma \vdash^{\mathrm{P}} t' : \mathrm{A}}{\Gamma \vdash^{\mathrm{P}} \mathrm{if}\ s\ \mathrm{then}\ t\ \mathrm{else}\ t' : A}$$

$$\frac{\Gamma \vdash^{\mathrm{D}} s : A}{\Gamma \vdash^{\mathrm{P}} \mathrm{return}\, s : A} \qquad \frac{\Gamma \vdash^{\mathrm{P}} s : A \qquad \Gamma, x : A \vdash^{\mathrm{P}} t : B}{\Gamma \vdash^{\mathrm{P}} \mathrm{let}\, x = s\, \mathrm{in}\, t : B}$$

$$\frac{\Gamma \vdash^{D} s : \mathrm{Real}}{\Gamma \vdash^{\mathrm{P}} \mathrm{bernoulli}\ s : \mathrm{Bool}} \qquad \frac{\Gamma \vdash^{\mathrm{D}} s : \mathrm{Real}}{\Gamma \vdash^{\mathrm{P}} \mathrm{poisson}\ s : \mathrm{Nat}}$$

Ignore out-of-bounds errors for parameters.

We also assume appropriate typing rules for constants (unit, booleans, integers, reals) and basic operations and functions.

Expressivity (1)

The type system is very restrictive, for example the program

$\langle$bernoulli 0.1, bernoulli 0.2$\rangle$

is not a well-typed program. Instead we must specify the sampling order. The following is a well-typed program of type $\mathrm{Bool} \times \mathrm{Bool}$:

```
let x = bernoulli 0.1 in
let y = bernoulli 0.2 in
⟨x, y⟩
```

Expressivity (2)

The language doesn't have an explicit type constructor dist for probability distributions.

But we can recover it as:

$$\mathrm{dist}\,A \;:=\; \mathrm{Unit} \to A$$
$$\mathrm{sample}\,d \;:=\; d\,() \qquad \text{(for } d : \mathrm{dist\ A}\text{)}$$

In fact, the type $\mathrm{Unit} \to A$ is always a monad in a call-by-value language with function types (prove it.)

Semantics

We will give formal semantics for this language in terms of sets, functions, and discrete probability distributions.

The semantics of the language consists of semantics for each kind of syntactic element:

Syntactic elements	Semantics using the probability monad
type A	set $[\![A]\!]$
$\Gamma = x_1 : A_1, \ldots, x_n : A_n$	$[\![\Gamma]\!] = [\![A_1]\!] \times \ldots \times [\![A_n]\!]$
$\Gamma \vdash^{\mathrm{D}} s : A$	$[\![s]\!] : [\![\Gamma]\!] \to [\![A]\!]$
$\Gamma \vdash^{\mathrm{P}} s : A$	$[\![s]\!] : [\![\Gamma]\!] \to \mathrm{dist}\, [\![A]\!]$

The definition is by induction on types and terms. We will see that it works because dist is a monad.

dist : a monad for discrete probability

A probability distribution on a set X is a function $d : X \to [0,1]$ such that $\sum_{x \in X} d(x) = 1$.

Let $\mathrm{dist}\, X$ denote the set of all probability distributions on X.

The dist operator comes equipped with combinators:

$$
\begin{aligned}
\mathrm{return} : X &\longrightarrow \mathrm{dist}\, X \\
x &\longmapsto \delta_x
\end{aligned}
$$

used for the semantics of return

$$
\begin{aligned}
\ggg= \,:\, \mathrm{dist}\, X \times (X \to \mathrm{dist}\, Y) &\longrightarrow \mathrm{dist}\, Y \\
(d \ggg= f) &\longmapsto \left(y \mapsto \sum_{x \in X} d(x) f(x)(y)\right)
\end{aligned}
$$

used for the semantics of $\mathrm{let} \ \ldots \ \mathrm{in}$

They satisfy the three monad equations:

$$
\begin{aligned}
d \ggg= \mathrm{return} \quad &= \quad d \\
\mathrm{return}(x) \ggg= f \quad &= \quad f(x) \\
(d \ggg= f) \ggg= g \quad &= \quad d \ggg= \lambda y \,.\, (f(y) \ggg= g)
\end{aligned}
$$

Semantics of types

Every type defines a **set.**

$$[\![\mathrm{Unit}]\!] = \{\,*\,\}$$

$$[\![\mathrm{Bool}]\!] = \{\mathrm{True}, \mathrm{False}\}$$

$$[\![\mathrm{Nat}]\!] = \mathbb{N}$$

$$[\![\mathrm{Real}]\!] = \mathbb{R}$$

$$[\![A \times B]\!] = [\![A]\!] \times [\![B]\!]$$

$$[\![A \to B]\!] = [\![A]\!] \to \mathrm{dist}\,[\![B]\!]$$

$X \to Y$:= set of all functions from X to Y

- In this language the real numbers are just a set = a discrete measurable space (this is not the usual σ-algebra on $\mathbb{R}$!).
- But it's ok to have a discrete probability distribution on an uncountable set.
- In fact, even if we remove the type Real, the semantics involves uncountable sets, like $[\![\mathrm{Unit} \to \mathrm{Bool}]\!]$ or $[\![\mathrm{Nat} \to \mathrm{Nat}]\!]$.
- A probability distribution d on an uncountable set X always has countable support, otherwise the sum $\sum_{x \in X} d(x)$ would diverge.

Semantics of terms (1)

$$\frac{}{\Gamma, x : A \vdash^{\mathrm{D}} x : A}$$

$\Gamma \times A \longrightarrow A$
is the projection.

$$\frac{\Gamma, x : A \vdash^{\mathrm{P}} t : B}{\Gamma \vdash^{\mathrm{D}} \lambda x.\, t : A \to B}$$

Every function
$f : \Gamma \times A \longrightarrow \mathrm{dist}\ B$
determines a function

$$\begin{aligned} \Gamma &\longrightarrow (A \to \mathrm{dist}\ B) \\ \gamma &\longmapsto f(\gamma, -) \end{aligned}$$

(In fact this is a
1-1 correspondence.)

$$\frac{\Gamma \vdash^{\mathrm{D}} t : A \to B \qquad \Gamma \vdash^{\mathrm{D}} s : A}{\Gamma \vdash^{\mathrm{P}} t\, s : B}$$

Given two functions

$$\begin{aligned} g &: \Gamma \longrightarrow (A \to \mathrm{dist}\ B) \\ h &: \Gamma \longrightarrow A \end{aligned}$$

we can build the function

$$\begin{aligned} \Gamma &\longrightarrow B \\ \gamma &\longmapsto g(\gamma)(h(\gamma)) \end{aligned}$$

Semantics of terms (2)

$$\frac{\Gamma \vdash^{\mathrm{D}} s : A \qquad \Gamma \vdash^{\mathrm{D}} t : B}{\Gamma \vdash^{\mathrm{D}} \langle s, t\rangle : A \times B}$$

Given two functions

$$g : \Gamma \longrightarrow A$$
$$h : \Gamma \longrightarrow B$$

we can build the function

$$\begin{aligned} \Gamma &\longrightarrow A \times B \\ \gamma &\longmapsto (g(\gamma), h(\gamma)) \end{aligned}$$

$$\frac{\Gamma \vdash^{\mathrm{D}} s : A \times B}{\Gamma \vdash^{\mathrm{D}} \mathrm{fst}\ s : A} \qquad \frac{\Gamma \vdash^{\mathrm{D}} s : A \times B}{\Gamma \vdash^{\mathrm{D}} \mathrm{snd}\ s : B}$$

We can post-compose a function

$$g : \Gamma \longrightarrow A \times B$$

with either of the two projection

$$\pi_1 : A \times B \longrightarrow A$$
$$\pi_2 : A \times B \longrightarrow B$$

Semantics of terms (3)

$$\frac{\Gamma \vdash^{\mathrm{D}} s : \mathrm{Bool} \quad \Gamma \vdash^{\mathrm{P}} t : \mathrm{A} \quad \Gamma \vdash^{\mathrm{P}} t' : \mathrm{A}}{\Gamma \vdash^{\mathrm{P}} \text{if } s \text{ then } t \text{ else } t' : A}$$

Given functions

$$h : \Gamma \longrightarrow \mathrm{Bool}$$
$$f : \Gamma \longrightarrow \mathrm{dist}\ A$$
$$g : \Gamma \longrightarrow \mathrm{dist}\ A$$

we define a function

$$\begin{aligned} \Gamma &\longrightarrow \mathrm{dist}\, A \\ \gamma &\longmapsto \begin{cases} f(\gamma) & \text{if } h(\gamma) = \mathrm{True} \\ g(\gamma) & \text{if } h(\gamma) = \mathrm{False} \end{cases} \end{aligned}$$

Remark. We can easily add a typing rule:

$$\frac{\Gamma \vdash^{\mathrm{D}} s : \mathrm{Bool} \quad \Gamma \vdash^{\mathrm{D}} t : \mathrm{A} \quad \Gamma \vdash^{\mathrm{D}} t' : \mathrm{A}}{\Gamma \vdash^{\mathrm{D}} \text{if } s \text{ then } t \text{ else } t' : A}$$

Semantics of terms (4)

$$\frac{\Gamma \vdash^{\mathrm{D}} s : A}{\Gamma \vdash^{\mathrm{P}} \mathrm{return}\, s : A}$$

Given a function

$$g : \Gamma \longrightarrow A$$

we define the function

$$\begin{aligned} \Gamma &\longrightarrow \mathrm{dist}\ A \\ \gamma &\mapsto \mathrm{return}(g(\gamma)) \end{aligned}$$

$$\frac{\Gamma \vdash^{\mathrm{P}} s : A \qquad \Gamma, x : A \vdash^{\mathrm{P}} t : B}{\Gamma \vdash^{\mathrm{P}} \mathrm{let}\, x = s\, \mathrm{in}\, t : B}$$

Given functions

$$g : \Gamma \longrightarrow \mathrm{dist}\ A$$

$$h : \Gamma \times A \longrightarrow \mathrm{dist}\ B$$

we define the function

$$\begin{aligned} \Gamma &\longrightarrow \mathrm{dist}\ B \\ \gamma &\longmapsto g(\gamma) \mathbin{>\!\!>\!\!=} h(\gamma, -) \end{aligned}$$

Semantics of terms (5)

$$\frac{\Gamma \vdash^{D} s : \mathrm{Real}}{\Gamma \vdash^{\mathrm{P}} \mathrm{bernoulli}\ s : \mathrm{Bool}} \qquad \frac{\Gamma \vdash^{\mathrm{D}} s : \mathrm{Real}}{\Gamma \vdash^{\mathrm{P}} \mathrm{poisson}\ s : \mathrm{Nat}}$$

We have functions

$$\mathrm{bernoulli} : [0,1] \longrightarrow \mathrm{dist}\,\mathrm{Bool}$$
$$\mathrm{poisson} : \mathbb{R}_{>0} \longrightarrow \mathrm{dist}\ \mathbb{N}$$

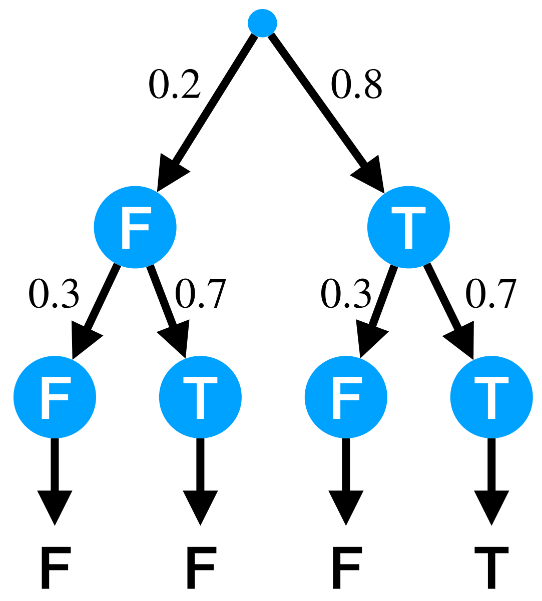
and we need

$$\mathrm{bernoulli}? : \mathbb{R} \longrightarrow \mathrm{dist}\,\mathrm{Bool}$$
$$\mathrm{poisson}? : \mathbb{R} \longrightarrow \mathrm{dist}\ \mathbb{N}$$

The behaviour is undefined but we need to define it.
(Typically, we would use a more precise type system
or add an exception monad.)

Semantics and enumeration trees

```
let x = bernoulli 0.2 in
let y = bernoulli 0.3 in
return (x ∧ y)
```



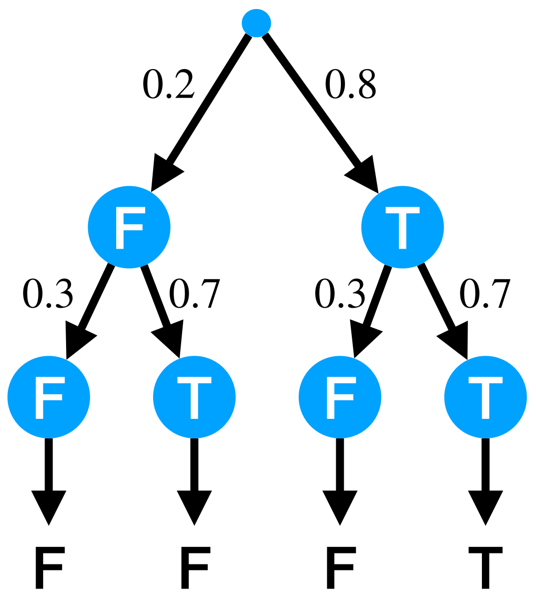
$$\dfrac{\dfrac{}{\vdash^{\mathrm{P}} \text{bernoulli } 0.2 : \text{Bool}} \qquad \dfrac{\dfrac{}{x : \text{Bool} \vdash^{\mathrm{P}} \text{bernoulli } 0.3 : \text{Bool}} \quad \dfrac{x : \text{Bool}, y : \text{Bool} \vdash^{\mathrm{D}} x \wedge y : \text{Bool}}{x : \text{Bool}, y : \text{Bool} \vdash^{\mathrm{P}} \text{return } x \wedge y : \text{Bool}}}{x : \text{Bool} \vdash \text{let } y = \text{bernoulli } 0.3 \text{ in return } x \wedge y : \text{Bool}}}{\vdash \text{let } x = \text{bernoulli } 0.2 \text{ in (let } y = \text{bernoulli } 0.3 \text{ in return } x \wedge y) : \text{Bool}}$$

We compute the semantics of the program using the basic building blocks:

$$1 \xrightarrow{\text{bernoulli } 0.2} \text{dist Bool} \qquad 1 \xrightarrow{\text{bernoulli } 0.3} \text{dist Bool} \qquad \text{Bool} \times \text{Bool} \xrightarrow{\wedge} \text{Bool}$$

Semantics and enumeration trees

```
let x = bernoulli 0.2 in
let y = bernoulli 0.3 in
return (x ∧ y)
```

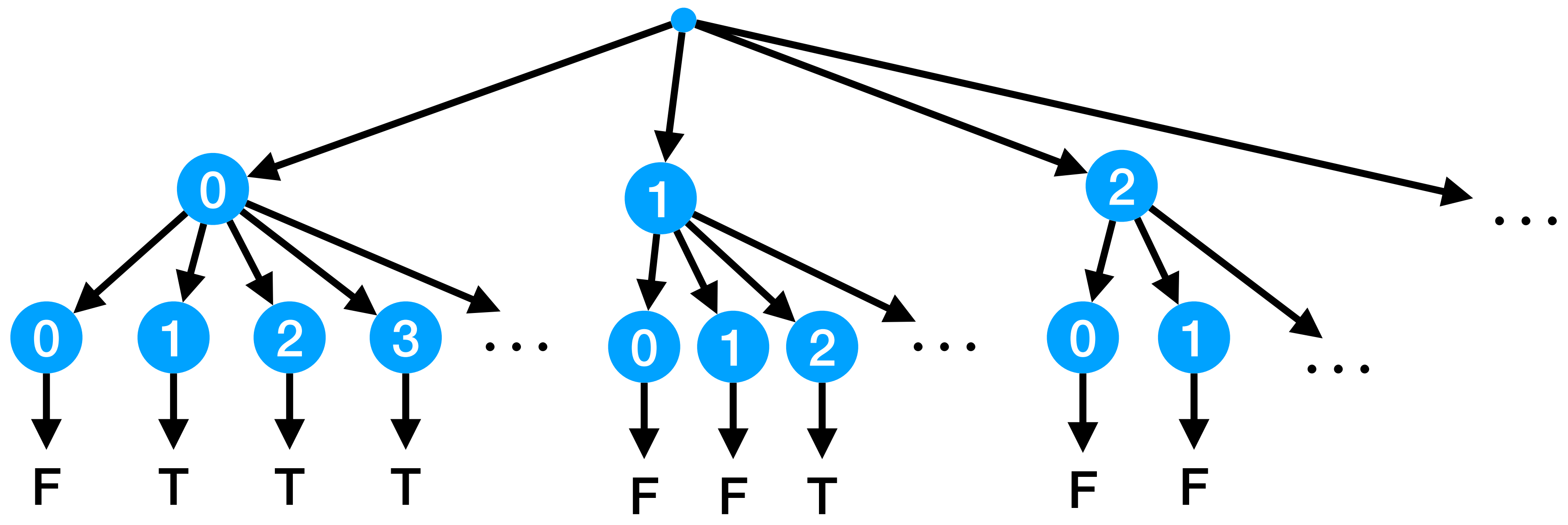


$$\dfrac{\dfrac{}{\vdash^{P} \text{bernoulli } 0.2 : \text{Bool}} \qquad \dfrac{\dfrac{}{x : \text{Bool} \vdash^{P} \text{bernoulli } 0.3 : \text{Bool}} \qquad \dfrac{x : \text{Bool}, y : \text{Bool} \vdash^{D} x \wedge y : \text{Bool}}{x : \text{Bool}, y : \text{Bool} \vdash^{P} \text{return } x \wedge y : \text{Bool}}}{x : \text{Bool} \vdash \text{let } y = \text{bernoulli } 0.3 \text{ in return } x \wedge y : \text{Bool}}}{\vdash \text{let } x = \text{bernoulli } 0.2 \text{ in } (\text{let } y = \text{bernoulli } 0.3 \text{ in return } x \wedge y) : \text{Bool}}$$

We compute the semantics of the program using the basic building blocks:

$$1 \xrightarrow{\text{bernoulli } 0.2} \text{dist Bool} \qquad 1 \xrightarrow{\text{bernoulli } 0.3} \text{dist Bool} \qquad \text{Bool} \times \text{Bool} \xrightarrow{\wedge} \text{Bool}$$

```
let x = poisson 2 in
let y = poisson 3 in
return (x < y)
```



An abstract perspective

- A **category** $\mathscr{C}$ has a collection $\mathrm{ob}\,\mathscr{C}$ of objects and a collection of morphisms $\mathscr{C}(A, B)$ between any two objects A and B, with a composition operation and identity morphisms.
- A **functor** $\mathscr{C} \to \mathscr{D}$ consists of a map $F : \mathrm{ob}\,\mathscr{C} \to \mathrm{ob}\,\mathscr{D}$ together with a maps $\mathscr{C}(A, B) \longrightarrow \mathscr{D}(FA, FB)$ for every $A, B \in \mathrm{ob}\,\mathscr{C}$.

- The category $\mathbf{Set}$ has sets as objects and $\mathbf{Set}(A, B)$is the set of functions $A \to B$.
- The category $\mathbf{Stoch}$ has sets as objects and $\mathbf{Stoch}(A, B)$ is the set of stochastic matrices, i.e. functions $A \to \mathrm{dist}\,B$.

(This is called the *Kleisli category* for the monad dist.)

An abstract perspective

What structure did we use for the semantics of our language?

- In $\mathbf{Set}$ we have finite cartesian products $\times$, function spaces $\to$, and *coproducts* $A + A$ (for $\mathbf{Bool}$ and conditionals).
- In $\mathbf{Stoch}$ we can also take finite products of objects and morphisms. This is a *monoidal product***,** not cartesian because components can be correlated.
- There is an **identity-on-objects** functor $L : \mathbf{Set} \to \mathbf{Stoch}$ which preserves the products.
- There is a functor $\mathrm{R} : \mathbf{Stoch} \to \mathbf{Set}$ which sends

$$A \in \mathbf{Stoch} \quad \longmapsto \quad \operatorname{dist} A \in \mathbf{Set}$$ (action on morphisms?)

- There is a well-behaved (**natural**) correspondence $\mathbf{Set}(A, \operatorname{dist} B) \cong \mathbf{Stoch}(A, B)\,.$

Extending the language with observations

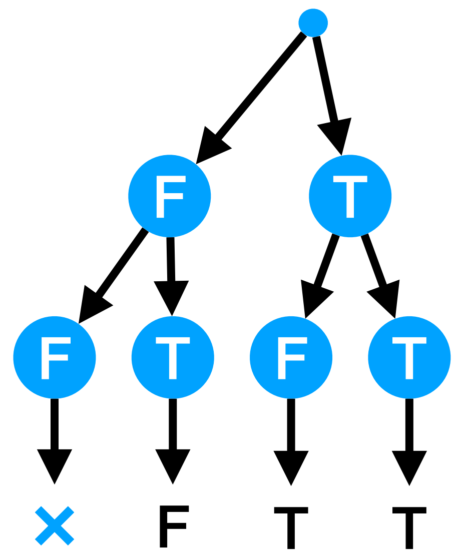
We add to the language a new syntactic primitive:

$s, t ::= \ldots \mid \mathrm{assume}\ s$

with typing rule

$$\frac{\Gamma \vdash^{\mathrm{D}} s : \mathrm{Bool}}{\Gamma \vdash^{\mathrm{P}} \mathrm{assume}\ s : \mathrm{Unit}}$$

```
let x = bernoulli 0.5 in
let y = bernoulli 0.5 in
assume (x || y);
return x
```



For the semantics of assume we need a new monad: dist is replaced by $\mathrm{dist}(- + 1)$, where $+$ is disjoint union and $1 = \{ * \}$.

Exercice. Define the return and bind.

Extending the language with observations

There is an **isomorphism of monads** $\mathrm{dist}(X+1) \cong \mathrm{dist}_{\leq}(X)$ where $\mathrm{dist}_{\leq}$ is the monad of sub-probability distributions over X. (**Why?**)

The semantics of inference is given by a normalization operator:

$$\mathrm{infer} : \mathrm{dist}_{\leq}(X) \to \mathrm{dist}(X) + 1$$

$$\mu \longmapsto \begin{cases} \mathrm{inl}\frac{\mu}{\mu(X)} & \text{if } \mu(X) > 0 \\ \mathrm{inr}\, * & \text{otherwise}\,. \end{cases}$$

Extending the language with observations

Remark. If we had a more expressive operation factor for "soft constraints" then the monad dist would not be sufficiently expressive for semantics.

Instead we would use either $\mathrm{dist}(X \times \mathbb{R}_+)$ or meas.

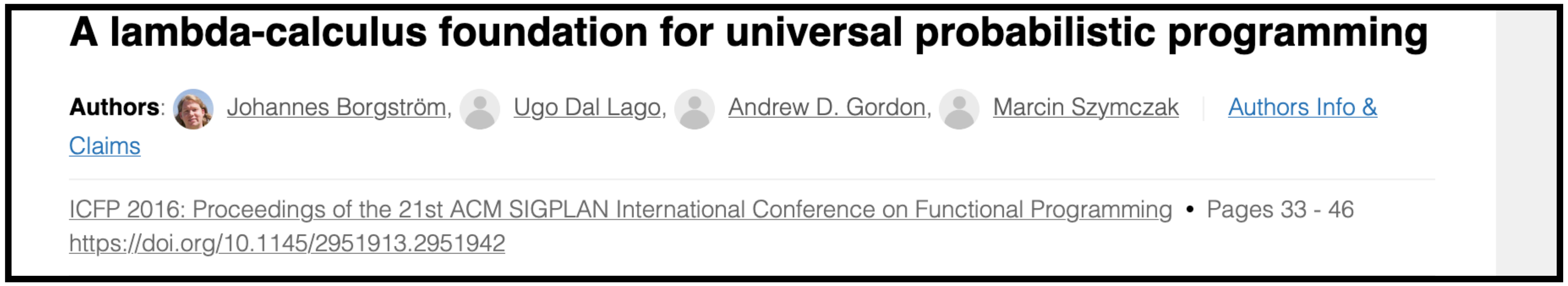
Towards a semantics with random elements

- Suppose that Ω is an idealised random number generator. Model this as the space $\Omega = [0,1]^{\mathbb{N}}$ with the product of uniform distributions.
- Then we can sample a uniform value ($\Omega \to [0,1]$) and split the infinite supply into two generators ($\Omega \to \Omega \times \Omega$), in a measure-preserving way.
- Then the set of measurable functions $\Omega \to X$ models an implementation of $\mathrm{dist}\, X$. But there are problems in making this precise:
  1. $\Omega \to X$ is not really a measurable space **Fixed in next lecture.**
  2. The associativity law of monads is only true up to measure-preserving iso: **Not really a problem.**
     $((d \ggg= f) \ggg= g) = (d \ggg= \lambda y\,.\,(f(y) \ggg= g))$

Operational semantics

- We can set up a probabilistic version of the usual reduction relation in the λ-calculus.
- E.g. $\text{bernoulli } 0.2 \xrightarrow{0.2} \text{True}$ and $\text{bernoulli } 0.2 \xrightarrow{0.8} \text{False}$
- Then we can reason about probabilities of terminating or returning a particular value.
- We will not look at operational semantics in this course because normalization in PPLs doesn't have a clear "rewriting" semantics: the operational behaviour of a program depends on the inference algorithm.

E.g. an operational approach to the Metropolis-Hastings inference algorithm:



A lambda-calculus foundation for universal probabilistic programming

Authors: Johannes Borgström, Ugo Dal Lago, Andrew D. Gordon, Marcin Szymczak | Authors Info & Claims

ICFP 2016: Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming • Pages 33 - 46
https://doi.org/10.1145/2951913.2951942

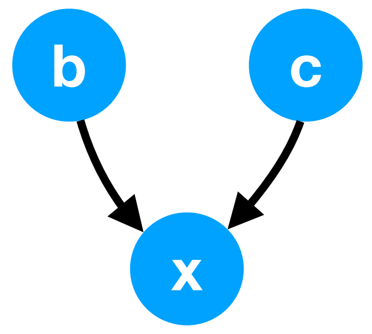
2. TD (discrete semantics and monads)

3. Probabilistic graphical models and PPLs

Key idea

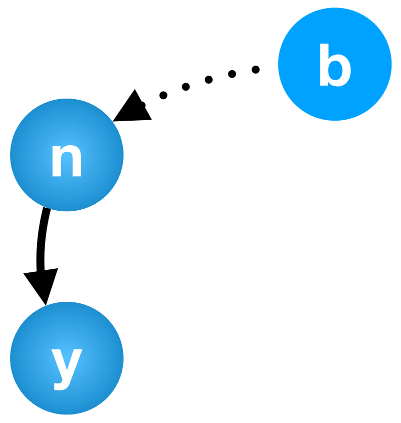
Very simple probabilistic programs can be represented by graphs of random variables. This is connected to "probabilistic graphical models".

```
let b = bernoulli 0.2 in
let c = bernoulli 0.3 in
let n = poisson (if (b ∧ c) then 1 else 2) in
return n
```

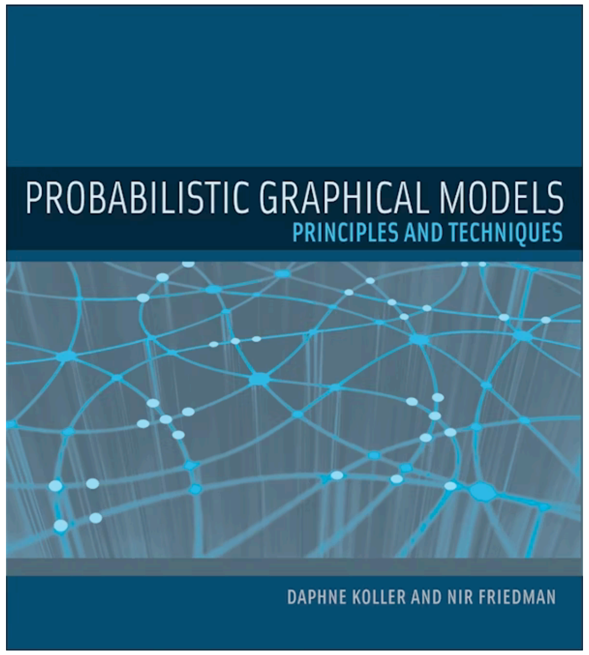


- The graphical representation has a formal meaning, and it is often close to the programmer's intuition.
- It allows for efficient inference (we will see this next).
- But the expressivity of graphical models is very limited:

```
let b = bernoulli 0.2 in
if b then
      let n = poisson 1 in
      let y = bernoulli (1/n) in
      return y
else return b
```



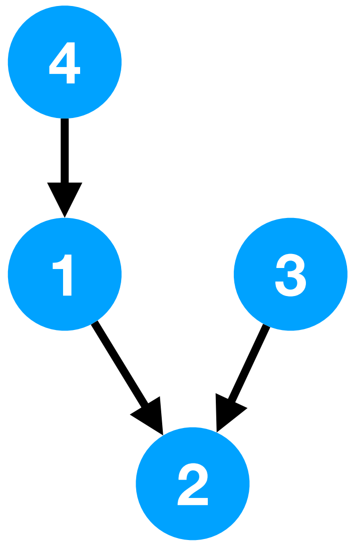
Bayesian networks


PROBABILISTIC GRAPHICAL MODELS
PRINCIPLES AND TECHNIQUES
DAPHNE KOLLER AND NIR FRIEDMAN

The best-known kind of graphical model.

Informal idea: a directed graph which encodes conditional independence properties for a joint probability distribution.

An example:



$$\begin{tikzcd} 4 \arrow[d] & \\ 1 \arrow[dr] & 3 \arrow[d] \\ & 2 \end{tikzcd}$$

- Sets X_1, X_2, X_3, X_4.
- $p \in \text{dist}(X_1 \times X_2 \times X_3 \times X_4)$
  $p(x_1, x_2, x_3, x_4) = p(x_4)\, p(x_1 \mid x_4)\, p(x_3)\, p(x_2 \mid x_1, x_3)$

Bayesian networks

Informal idea: a directed graph which encodes conditional independence properties for a joint probability distribution.

Definition. A **Bayesian network** consists of:

- An irreflexive directed acyclic graph $(V, E \subseteq V \times V)$
- A family of sets $\{X_v\}_{v \in V}$
- A distribution

  $P \in \mathrm{dist}\left(\prod_{v \in V} X_v\right)$

  such that for every $\boldsymbol{x} = (x_v)_{v \in V} \in \prod_{v \in V} X_v$,

  $P(\boldsymbol{x}) = \prod_{v \in V} P(x_v \mid \boldsymbol{x}_{\mathrm{pa}(v)})$.

Equivalently: the distribution of a variable v only depends on the values of its parents $\mathrm{pa}(v)$ in the graph: $P(x_v \mid \boldsymbol{x}_{V \setminus v}) = P(x_v \mid \boldsymbol{x}_{\mathrm{pa}(v)})$

Remark. There is no canonical way to represent a given distribution as a graph.

For example, the following graphs express the same independence assumptions (none) and are both compatible with **every** distribution on $X_1 \times X_2$.

$$\begin{tikzcd} 1 & 2 \arrow[l, bend left] & & 1 \arrow[r, bend right] & 2 \end{tikzcd}$$

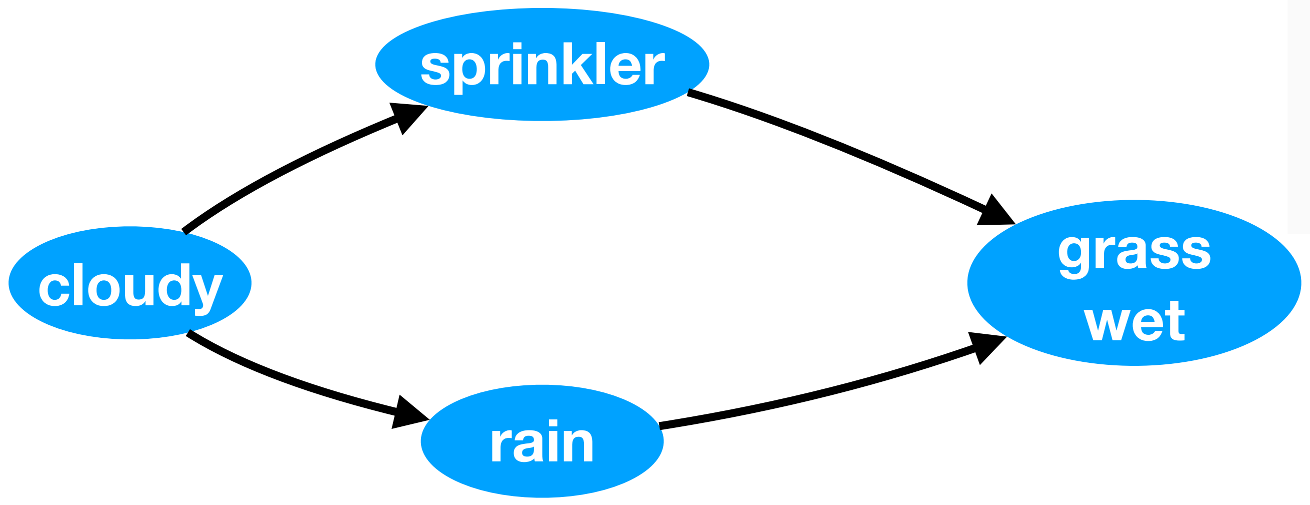
Consider the following model with four Boolean variables.

We observe that the grass is wet, and we want to know if it has been raining or if the sprinkler has been turned on.

Our prior distributions depend on whether the sky is cloudy.

$$\begin{array}{c|cc} C\backslash S & \texttt{t} & \texttt{f} \\ \hline \texttt{t} & 0.1 & 0.9 \\ \texttt{f} & 0.5 & 0.5 \end{array}$$

$$\begin{array}{cc} \texttt{t} & \texttt{f} \\ {}^3/_5 & {}^2/_5 \end{array}$$

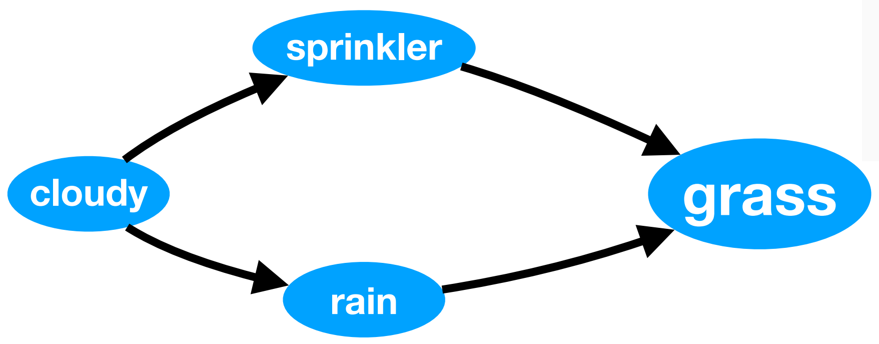


$$\begin{array}{c|cc} S,R\backslash G & \texttt{t} & \texttt{f} \\ \hline \texttt{t,t} & 0.99 & 0.01 \\ \texttt{t,f} & 0.9 & 0.1 \\ \texttt{f,t} & 0.9 & 0.1 \\ \texttt{f,f} & 0 & 1 \end{array}$$

$$\begin{array}{c|cc} C\backslash R & \texttt{t} & \texttt{f} \\ \hline \texttt{t} & 0.8 & 0.2 \\ \texttt{f} & 0.2 & 0.8 \end{array}$$

$$\begin{array}{c|cc} C\backslash S & \texttt{t} & \texttt{f} \\ \hline \texttt{t} & 0.1 & 0.9 \\ \texttt{f} & 0.5 & 0.5 \end{array}$$

$$\begin{array}{cc} \texttt{t} & \texttt{f} \\ {}^{3}/_{5} & {}^{2}/_{5} \end{array}$$



$$\begin{array}{c|cc} S,R\backslash G & \texttt{t} & \texttt{f} \\ \hline \texttt{t,t} & 0.99 & 0.01 \\ \texttt{t,f} & 0.9 & 0.1 \\ \texttt{f,t} & 0.9 & 0.1 \\ \texttt{f,f} & 0 & 1 \end{array}$$

$$\begin{array}{c|cc} C\backslash R & \texttt{t} & \texttt{f} \\ \hline \texttt{t} & 0.8 & 0.2 \\ \texttt{f} & 0.2 & 0.8 \end{array}$$

Observation 1. Efficient representation of the joint distribution.

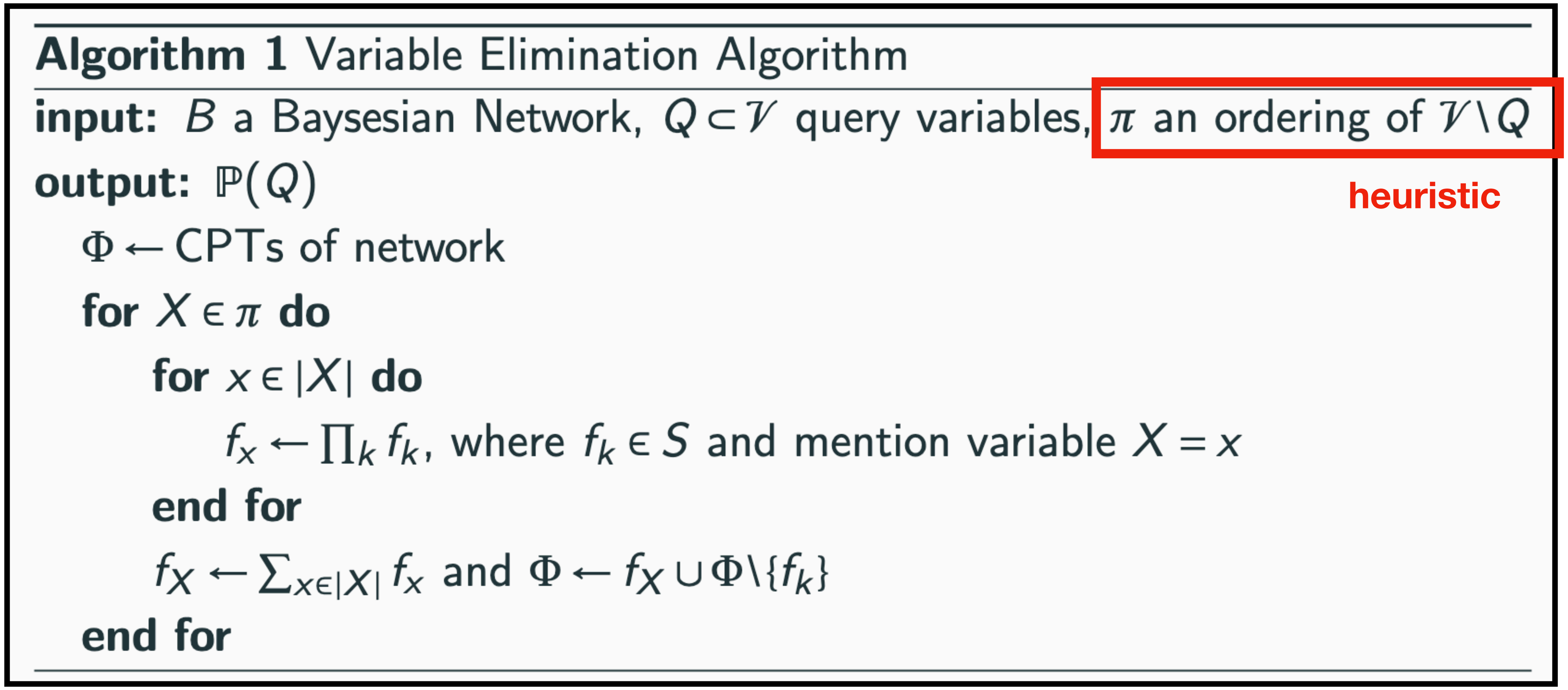
Observation 2. Efficient computation of the marginal distributions, as needed for exact inference. For example:

$$\Pr(G) = \sum_{R,S,C} \Pr(G,R,S,C)$$ ***naive formula***

$$= \sum_{R,S,C} \Pr(G \mid S,R)\Pr(R \mid C)\Pr(S \mid C)\Pr(C)$$ ***definition of the joint distribution***

$$\left.\begin{aligned} &= \sum_{C} \Pr(C) \sum_{S} \Pr(S \mid C)\left(\sum_{R} \Pr(R \mid C)\Pr(G \mid S,R) \right) \\ &= \sum_{R} \sum_{S} \Pr(G \mid S,R) \sum_{C} \Pr(C)\Pr(S \mid C)\Pr(R \mid C) \end{aligned}\right\}$$ ***two possible factorisations***

thanks to C. Tasson



```
Algorithm 1 Variable Elimination Algorithm
input: $B$ a Baysesian Network, $Q \subset \mathcal{V}$ query variables, $\pi$ an ordering of $\mathcal{V} \setminus Q$
output: $\mathbb{P}(Q)$
  $\Phi \leftarrow$ CPTs of network
  for $X \in \pi$ do
      for $x \in |X|$ do
          $f_x \leftarrow \prod_k f_k$, where $f_k \in S$ and mention variable $X = x$
      end for
      $f_X \leftarrow \sum_{x \in |X|} f_x$ and $\Phi \leftarrow f_X \cup \Phi \setminus \{f_k\}$
  end for
```

heuristic

$$\Pr(G) = \sum_{R,S,C} \Pr(G,R,S,C)$$

naive formula

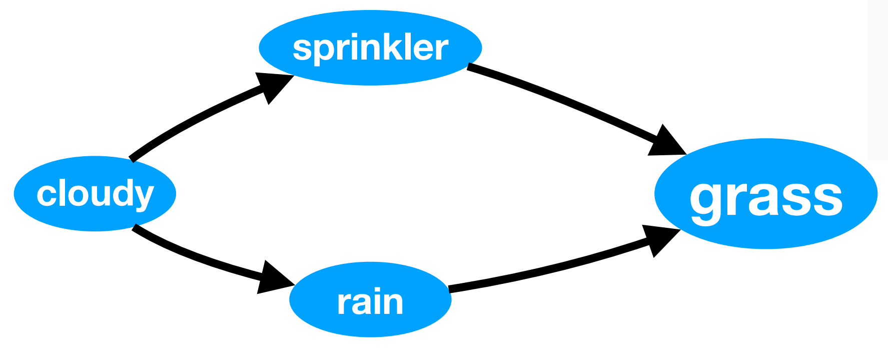
$$= \sum_{R,S,C} \Pr(G \mid S,R)\Pr(R \mid C)\Pr(S \mid C)\Pr(C)$$

definition of the joint distribution

$$\left.\begin{aligned} &= \sum_C \Pr(C) \sum_S \Pr(S \mid C)\left(\sum_R \Pr(R \mid C)\Pr(G \mid S,R)\right) \\ &= \sum_R \sum_S \Pr(G \mid S,R) \sum_C \Pr(C)\Pr(S \mid C)\Pr(R \mid C) \end{aligned}\right\}$$

two possible factorisations

Bayesian networks and PPLs



$$\begin{array}{c} \texttt{t} & \texttt{f} \\ [^3/_5 & ^2/_5] \end{array}$$

$$\begin{array}{c|cc} C\backslash S & \texttt{t} & \texttt{f} \\ \texttt{t} & 0.1 & 0.9 \\ \texttt{f} & 0.5 & 0.5 \end{array}$$

$$\begin{array}{c|cc} C\backslash R & \texttt{t} & \texttt{f} \\ \texttt{t} & 0.8 & 0.2 \\ \texttt{f} & 0.2 & 0.8 \end{array}$$

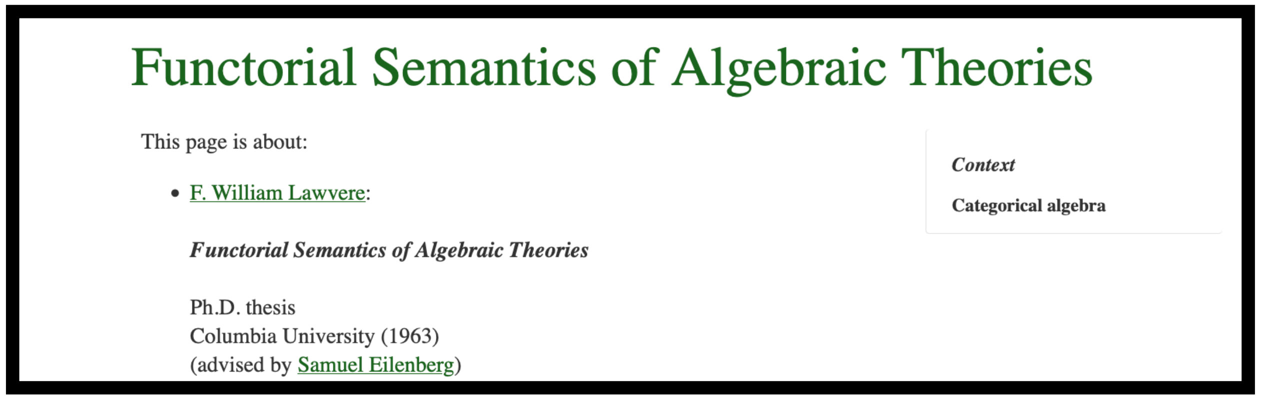
$$\begin{array}{c|cc} S,R\backslash G & \texttt{t} & \texttt{f} \\ \texttt{t,t} & 0.99 & 0.01 \\ \texttt{t,f} & 0.9 & 0.1 \\ \texttt{f,t} & 0.9 & 0.1 \\ \texttt{f,f} & 0 & 1 \end{array}$$

To implement this in a PPL, we must choose a topological ordering of the DAG.

```
let C = bernoulli (3/5) in
let S = if C then (bernoulli 0.1) else (bernoulli 0.5) in
let R = if C then (bernoulli 0.8) else (bernoulli 0.2) in
let G = if (S ∧ T) then (bernoulli 0.99) else ... in
assume G;
return R
```

Exercice: all implementations have the same semantics.

A categorical perspective


Functorial Semantics of Algebraic Theories

This page is about:

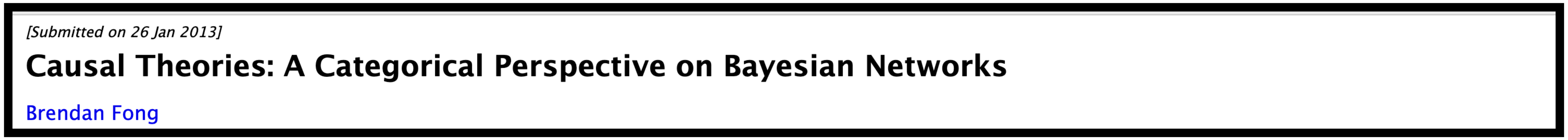
- F. William Lawvere:

  Functorial Semantics of Algebraic Theories

  Ph.D. thesis
  Columbia University (1963)
  (advised by Samuel Eilenberg)

Context

Categorical algebra


[Submitted on 26 Jan 2013]

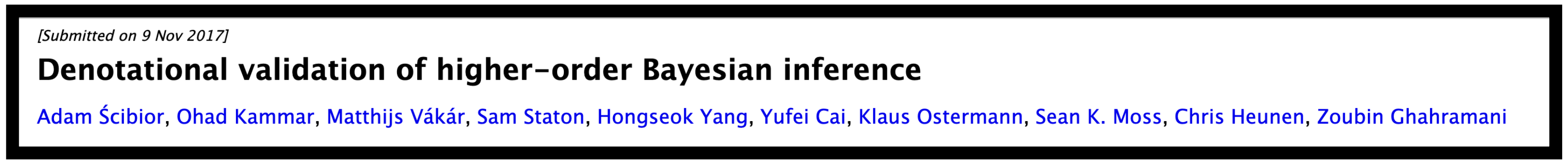
Causal Theories: A Categorical Perspective on Bayesian Networks

Brendan Fong

- We can define a Bayesian network as a DAG G together with a structure preserving functor $\mathscr{C}_G \longrightarrow \mathbf{Stoch}$ where $\mathscr{C}_G$ is a category generated from G.
- <u>*Informal idea, omitting technical details*</u>: The objects of $\mathscr{C}_G$ are lists of vertices of G, and the morphisms are generated from:
  1. "Copy" and "discard" morphisms
  2. Morphisms $v_1, \ldots, v_n \longrightarrow v$ for each vertex v, where $\{v_i\}_{i \leq n} = \mathrm{pa}(v)$.
  3. The operations of symmetric monoidal categories.

Conclusion

- *Sets and probability monads*: semantics for a simple call-by-value PPL. Corresponds to enumeration.
- This method will also extend to continuous probability with measurable spaces. Can also choose an appropriate monad depending on the inference algorithm.

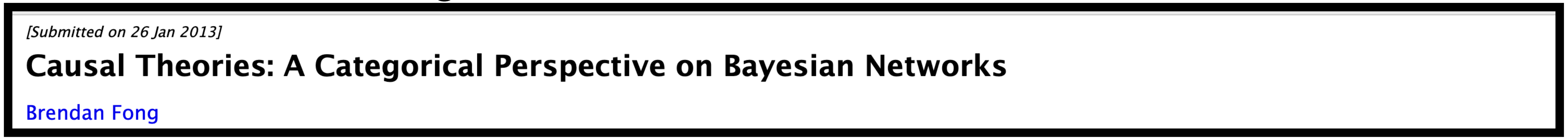


[Submitted on 9 Nov 2017]

Denotational validation of higher-order Bayesian inference

Adam Ścibior, Ohad Kammar, Matthijs Vákár, Sam Staton, Hongseok Yang, Yufei Cai, Klaus Ostermann, Sean K. Moss, Chris Heunen, Zoubin Ghahramani

- *Bayesian networks:* an efficient graphical representation of finite multidimensional distributions, with faster inference.
- We looked at the categorical foundations.



[Submitted on 26 Jan 2013]

Causal Theories: A Categorical Perspective on Bayesian Networks

Brendan Fong

- Other ongoing research: efficient inference methods for discrete variables in probabilistic programming (cf. Pyro, Stan, Dice, Pluck).