Categorical models of probability with symmetries

Sam Staton, Oxford

Categorical models of probability with symmetries

My starting point:

- Probabilistic programming is an internal language for categorical probability theory (as well as a useful practical tool in stats/ML).
- Programming languages already have mechanisms for abstraction and invariance.
- These can give new perspectives on symmetry in probability.

Plan of talk:

1. Intuitive illustrations of symmetries in
   - a. **random graphs**
   - b. beta distributions / Pólya urns
2. Models for
   - a. beta / Pólya urns

     Staton, Stein, Yang, Ackerman, Freer, Roy, ICALP 2018.
   - b. random graphs

     ongoing work with Ackerman, Freer, Roy, Yang.

Building infinite random graphs

Interface:

```
get() : node
edge(node,node) : bool
```

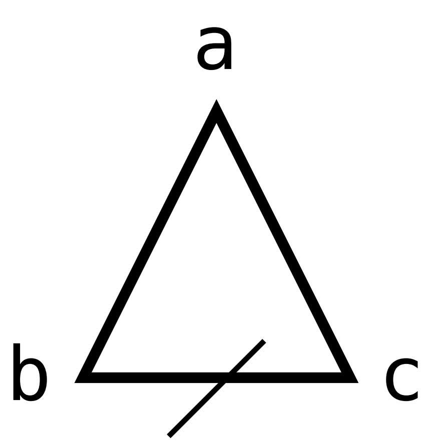
```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```

Building infinite random graphs

Interface:

```
get() : node
edge(node,node) : bool
```

```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```



Building infinite random graphs

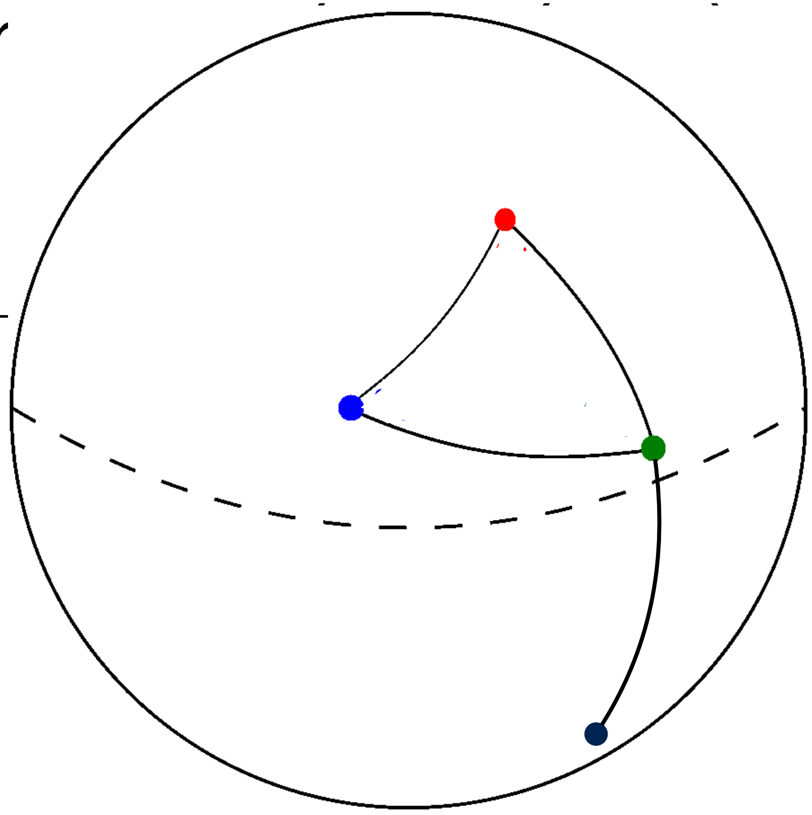
Interface:

```
get() : node

edge(node,node) : bool
```

Example:

```
get() = uniform Sn

edge(p,q)
  = if d(p,q) < π/2
    then True else False
```

```
a <- get()

b <- get()

c <- get()

r
```



Building infinite random graphs

Interface:

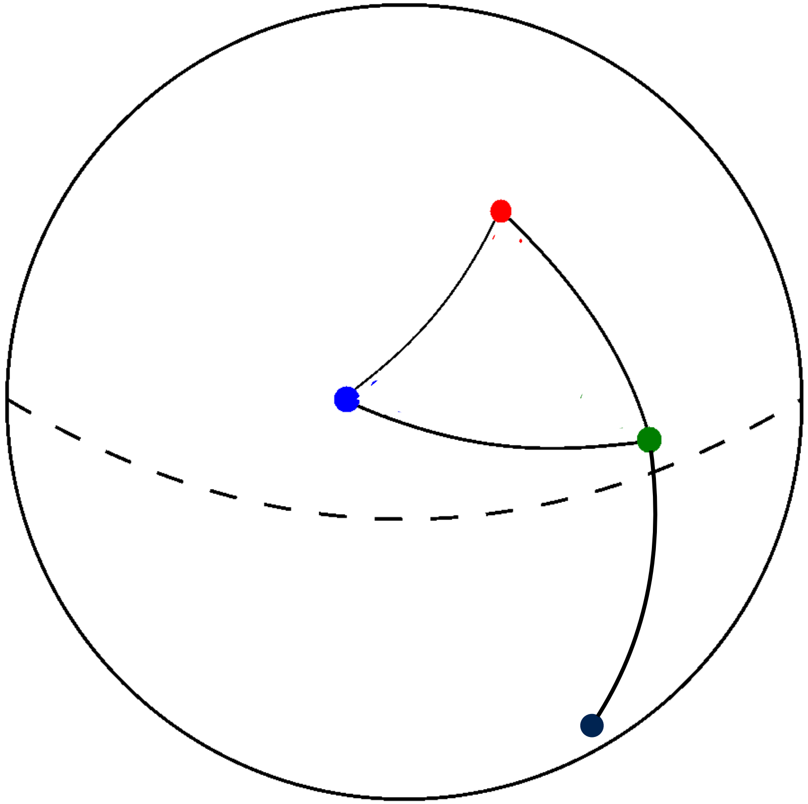
```
get() : node
edge(node,node) : bool
```

```
a <- get()
b <- get()
return (edge(a,b))
```

Example:

get() = uniform S_n

edge(p,q)
  = if d(p,q) < $\pi/2$
    then True else False



Building infinite random graphs

Interface:

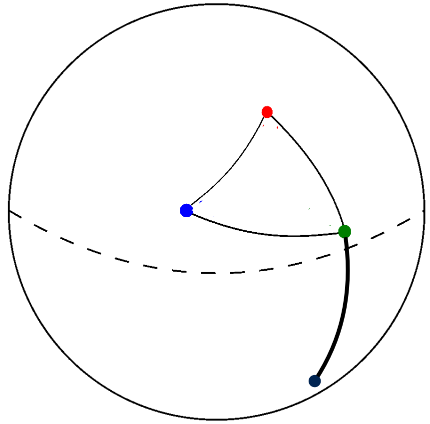
```
get() : node
edge(node,node) : bool
```

Example:

get() = uniform S_n

edge(p,q)
  = if d(p,q) < $\pi/2$
    then True else False

```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```



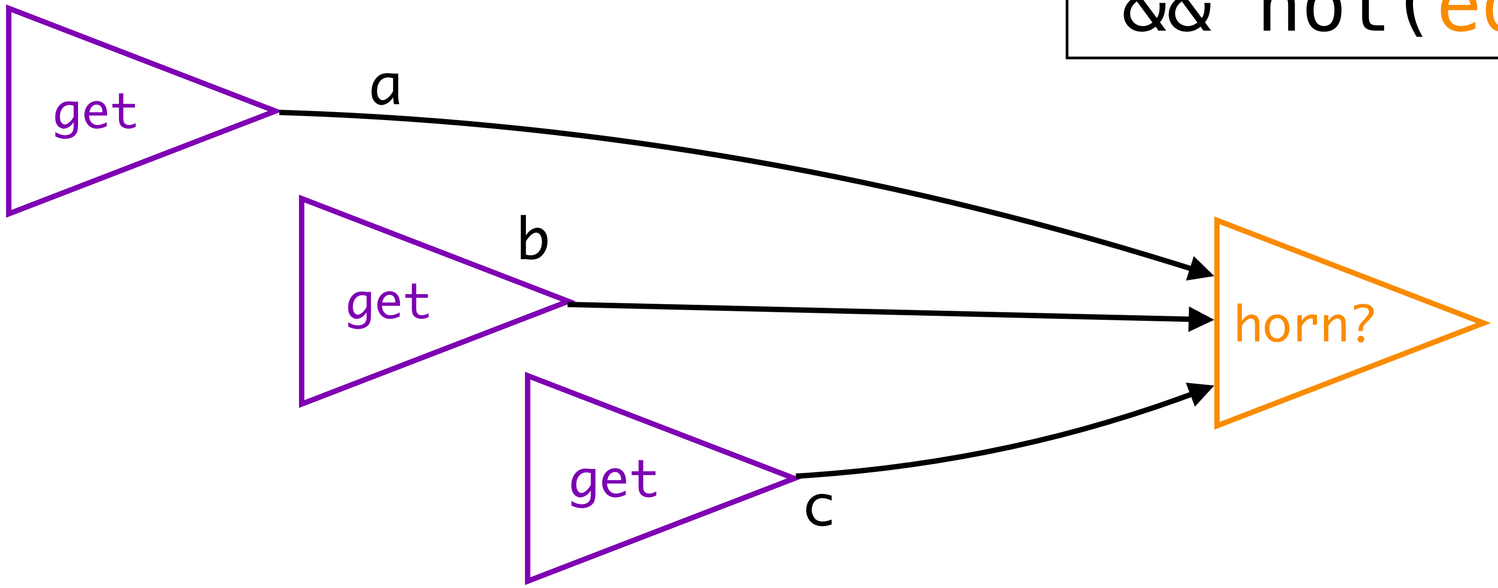
Building infinite random graphs

Interface:

```
get() : node
edge(node,node) : bool
```

```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```

Data flow graph



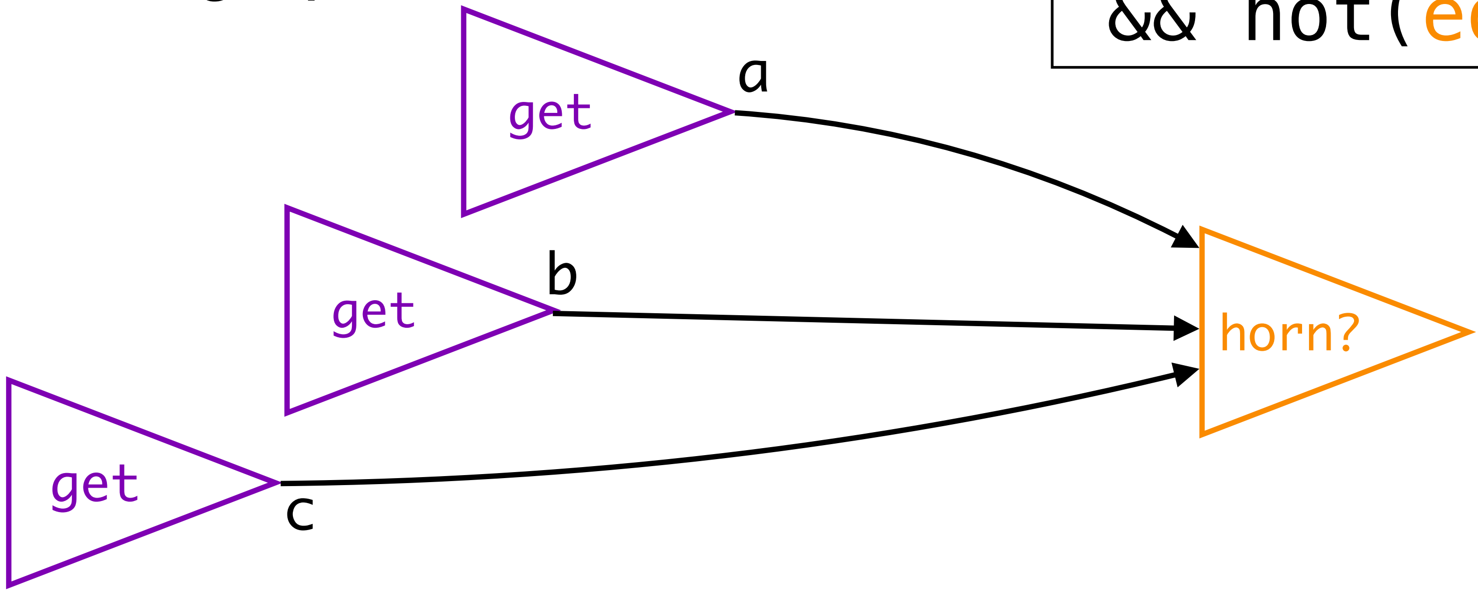
Building infinite random graphs

Interface:

```
get() : node
edge(node,node) : bool
```

```
c <- get()
b <- get()
a <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```

Data flow graph



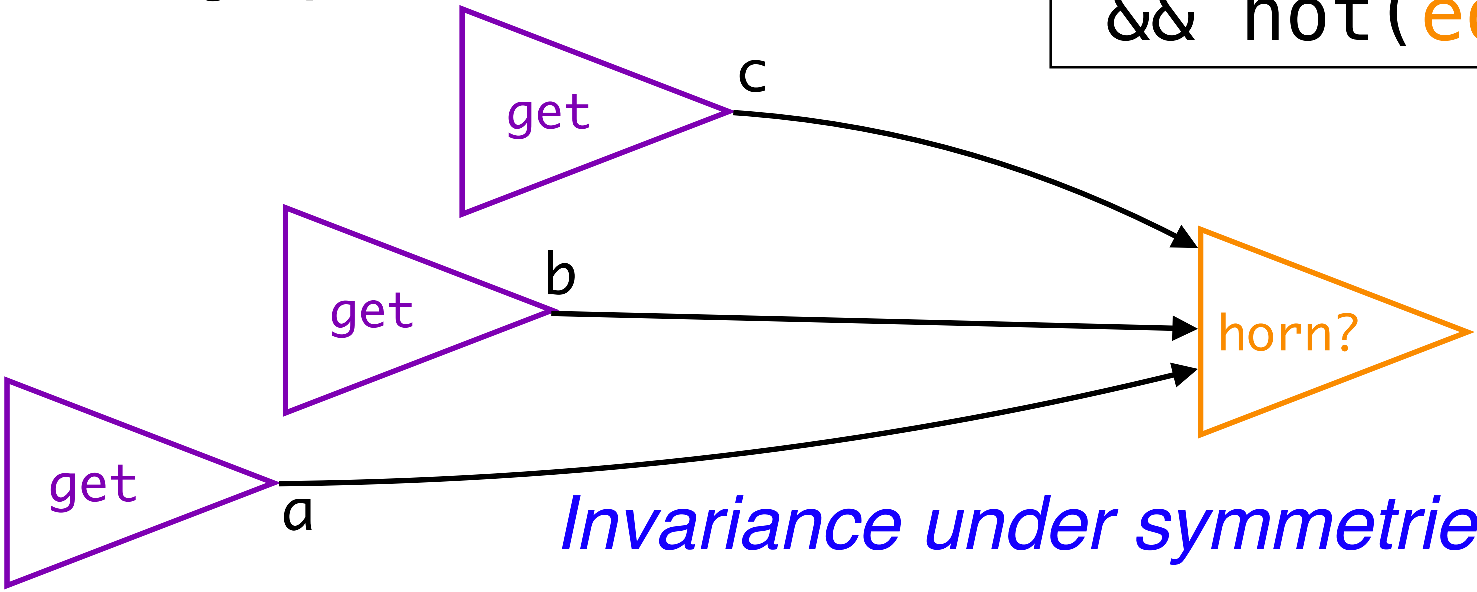
Building infinite random graphs

Interface:

```
get() : node
edge(node,node) : bool
```

```
a <- get()
b <- get()
c <- get()
return (edge(c,b)
 && edge(c,a)
 && not(edge(b,a)))
```

Data flow graph



Invariance under symmetries of data flow

= graph *exchangeability*

Building infinite random graphs

Interface:

```
get() : node
edge(node,node) : bool
```

The interface doesn't allow:

```
a <- get()
b <- get()
return (a < b)
```

```
a <- get()
b <- get()
return (sin(a) = cos(b))
```

Building infinite random graphs

Interface:

```
get() : node
edge(node,node) : bool
```

Invariance *under changes to implementation*

= *graph exchangeability*

The interface doesn't allow:

```
a <- get()
b <- get()
return (a < b)
```

```
a <- get()
b <- get()
return (sin(a) = cos(b))
```

Summary of symmetries

Interface:

```
get() : node
edge(node,node) : bool
```

Invariance under implementation details

\+ data flow symmetries

=

graph exchangeability

(Aldous-Hoover)

Another model

Interface:

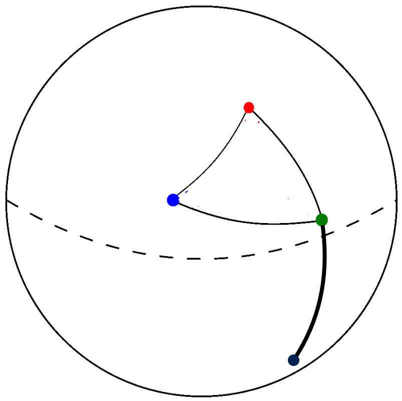
```
get() : node
edge(node,node) : bool
```

Example:

get() = uniform S_n

edge(p,q) = [d(p,q) < $\pi/2$]

```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```



building on
Bubeck, Ding, Eldan, Racz, 2015
Devroye, György, Lugosi, Udina, 2011

Another model

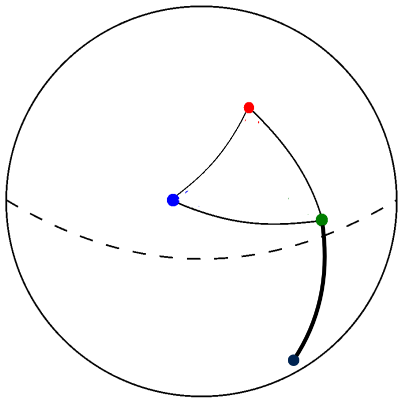
Interface:

```
get() : node
edge(node,node) : bool
```

Example:

```
get() = uniform(0,1)
edge(p,q)
  = memoize_{p,q}(bernoulli(0.5))
```

```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```



building on
Bubeck, Ding, Eldan, Racz, 2015
Devroye, György, Lugosi, Udina, 2011

Another model

Interface:

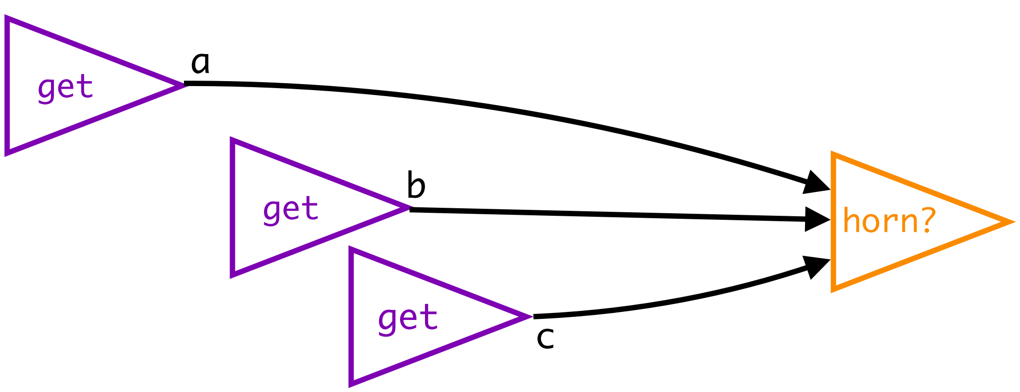
```
get() : node
edge(node,node) : bool
```

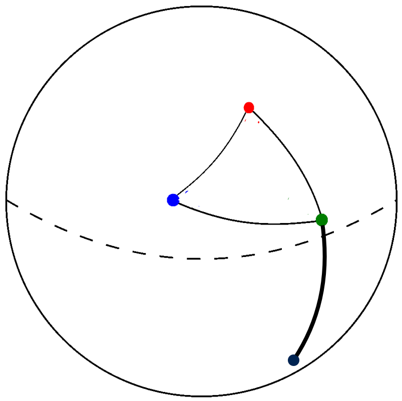
Example:

```
get() = uniform(0,1)
edge(p,q)
  = memoize_{p,q}(bernoulli(0.5))
```

```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```

Roy, Mansinghka, Goodman, Tenenbaum, ICML 2008





building on
Bubeck, Ding, Eldan, Racz, 2015
Devroye, György, Lugosi, Udina, 2011

Summary of symmetries

Interface:

```
get() : node
edge(node,node) : bool
```

Invariance under implementation details

\+ data flow symmetries

=

graph exchangeability

(Aldous-Hoover)

Plan of talk:

1. Intuitive illustrations of symmetries in
    a. random graphs
    b. **beta distributions / Pólya urns**
2. Models for
    a. beta / Pólya urns
    b. random graphs

Staton, Stein, Yang, Ackerman, Freer, Roy, ICALP 2018.

ongoing work with Ackerman, Freer, Roy, Yang.

Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```

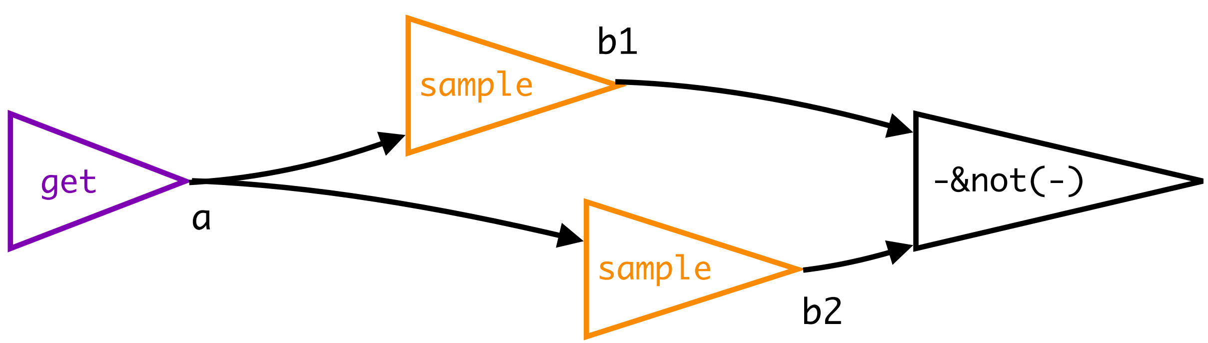
Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```



Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

<u>Interface:</u>

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```

<u>Example:</u>

```
get() = uniform(0,1)
sample(p) = bernoulli(p)
```

Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```

Example:

```
get() = uniform(0,1)
sample(p) = bernoulli(p)
```

Prob(return True) =

$$\int_0^1 p(1-p)\,\mathrm{d}p = \frac{1}{6}$$

Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

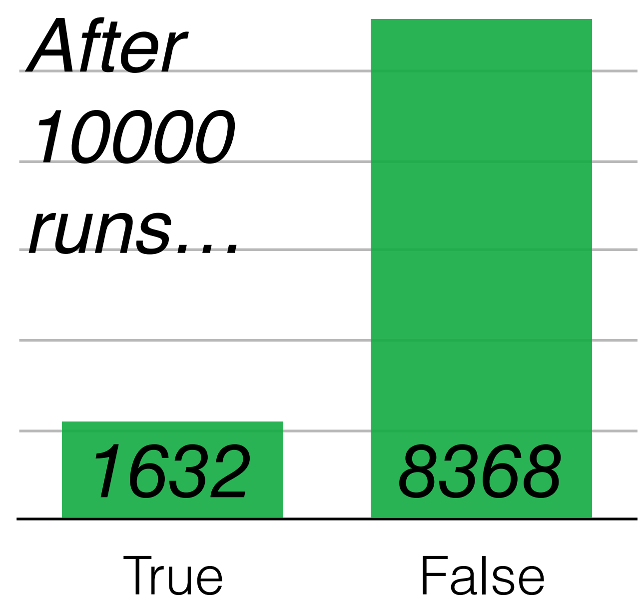
Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```

Example:

```
get() = uniform(0,1)
sample(p) = bernoulli(p)
```



Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```

Example:

```
get() = uniform(0,1)
sample(p) = bernoulli(p)
```

Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- uniform(0,1)
b1 <- bernoulli(a)
b2 <- bernoulli(a)
return (b1 & not(b2))
```

Example:

```
get() = uniform(0,1)
sample(p) = bernoulli(p)
```

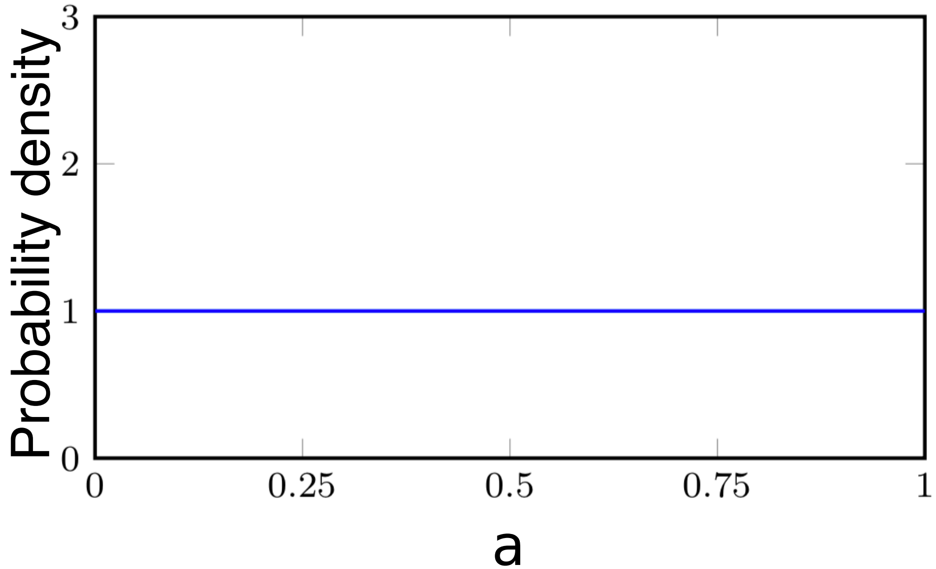
Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- uniform(0,1)
b1 <- bernoulli(a)
b2 <- bernoulli(a)
return (b1 & not(b2))
```



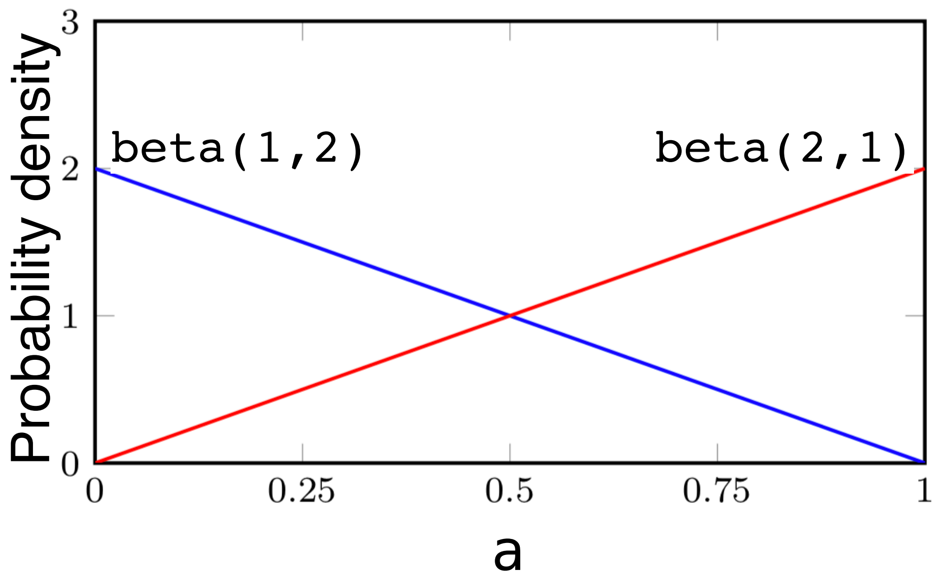
Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- uniform(0,1)
b1 <- bernoulli(a)
b2 <- bernoulli(a)
return (b1 & not(b2))
```



b1 <- bernoulli($\frac{1}{2}$)

```
a <- beta(1+b1,2-b1)
b2 <- bernoulli(a)
return (b1 & not(b2))
```

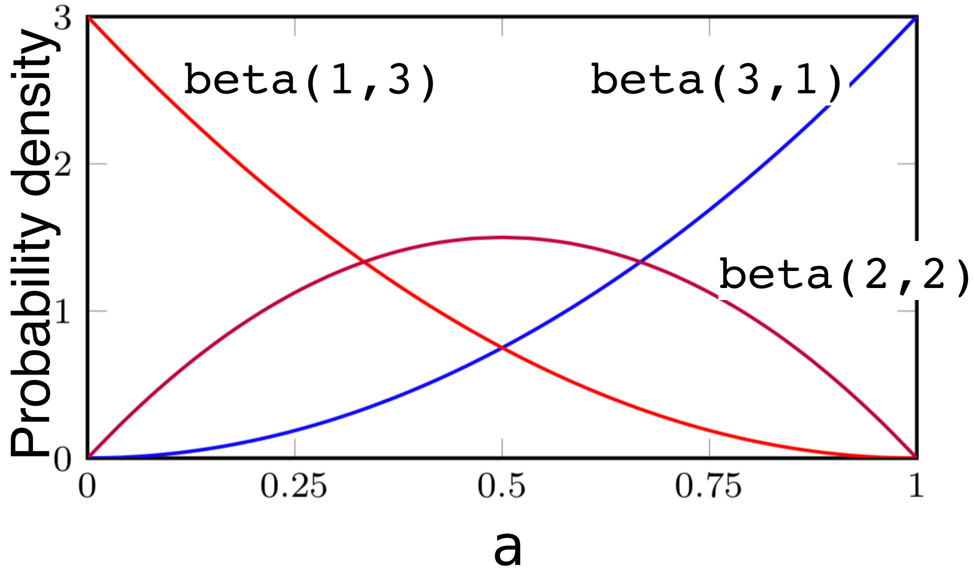
Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- uniform(0,1)
b1 <- bernoulli(a)
b2 <- bernoulli(a)
return (b1 & not(b2))
```



b1 <- bernoulli($\frac{1}{2}$)

b2 <- bernoulli($\frac{1+b1}{3}$)

a <- beta(1+b1+b2,3−b1−b2)

return (b1 & not(b2))

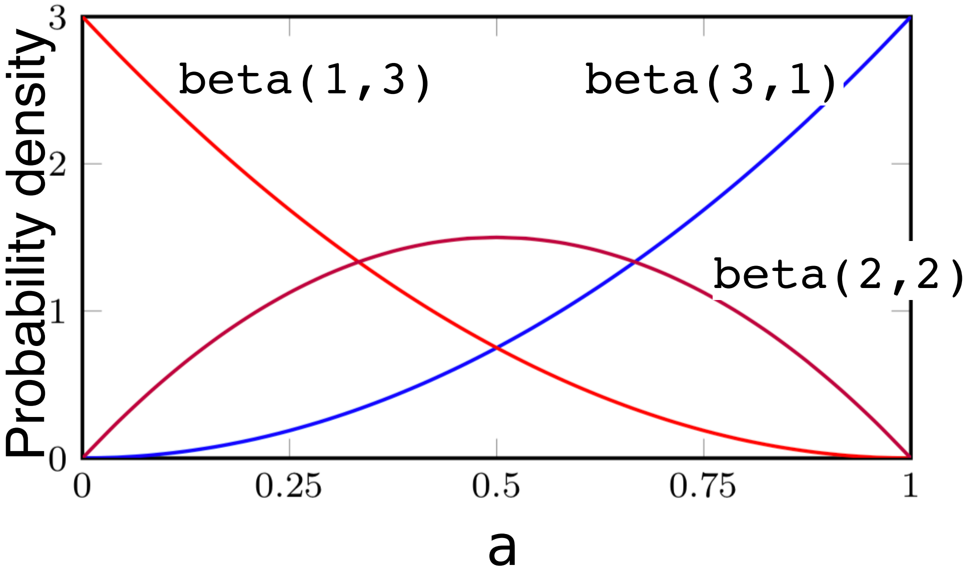
Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- uniform(0,1)
b1 <- bernoulli(a)
b2 <- bernoulli(a)
return (b1 & not(b2))
```



b1 <- bernoulli($\frac{1}{2}$)

b2 <- bernoulli($\frac{1+\mathrm{b1}}{3}$)

~~a <- beta(1+b1+b2,3-b1-b2)~~

return (b1 & not(b2))

Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- uniform(0,1)
b1 <- bernoulli(a)
b2 <- bernoulli(a)
return (b1 & not(b2))
```

Prob(return True) =

$$\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$$

No integration required!

b1 <- bernoulli($\frac{1}{2}$)

b2 <- bernoulli($\frac{1 + \mathtt{b1}}{3}$)

~~a <- beta(1+b1+b2,3-b1-b2)~~

```
return (b1 & not(b2))
```

Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

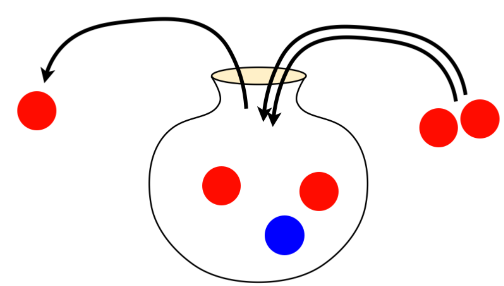
Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```

Another example:

`get()` = *new urn*



`sample(p)` = *Pólya draw: one out, two in*

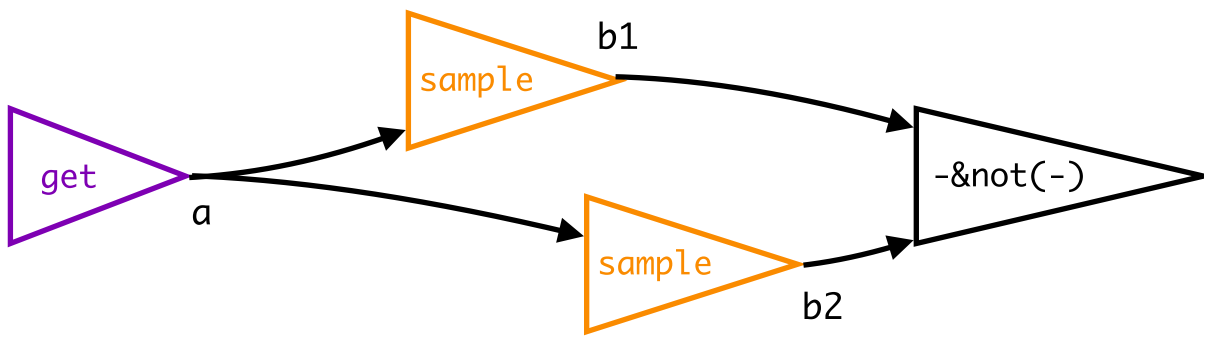
Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```



Example 2: Beta / bernoulli

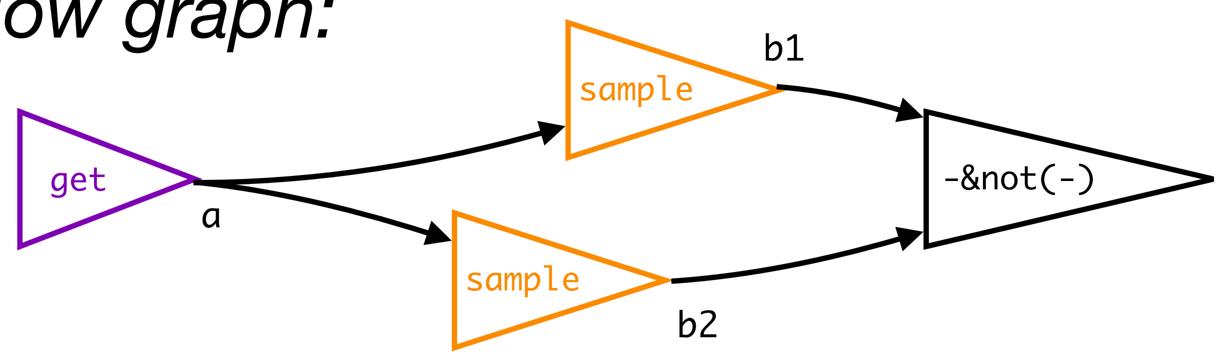
Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b2 <- sample(a)
b1 <- sample(a)
return (b1 & not(b2))
```

Data flow graph:



Example 2: Beta / bernoulli

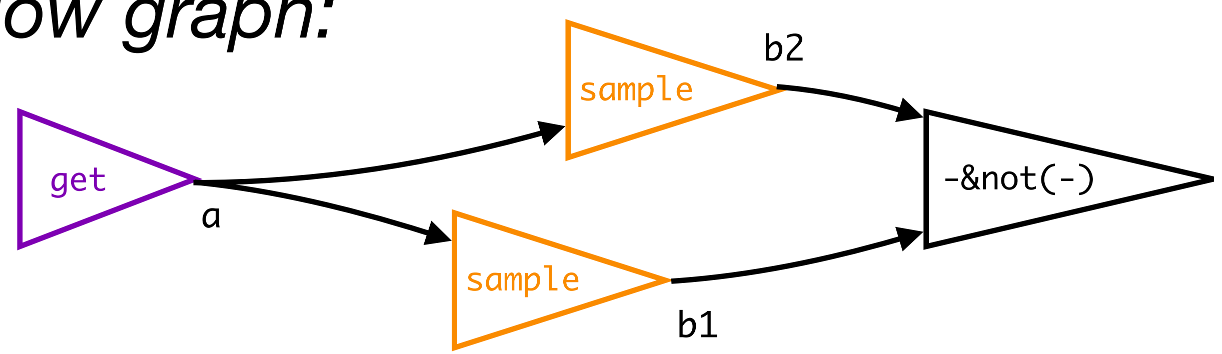
Given an unknown coin, what is the probability of heads then tails?

Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b2 & not(b1))
```

Data flow graph:



Summary of symmetries

Interface:

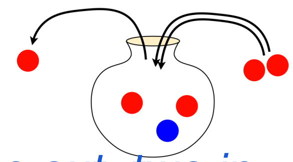
```
get() : I
sample(I) : bool
```

Two implementations:

```
get() = uniform(0,1)
sample(p) = bernoulli(p)
```

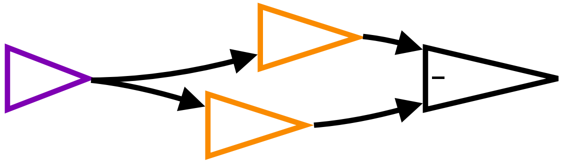
`get()` = *new urn*

`sample(p)` = *Pólya draw: one out, two in*



Invariance under implementation details

+ data flow symmetries



=

sequence exchangeability (de Finetti)

Plan of talk:

1. Intuitive illustrations of symmetries in
    a. random graphs
    b. beta distributions / Pólya urns
2. Models for
    a. beta / Pólya urns — Staton, Stein, Yang, Ackerman, Freer, Roy, ICALP 2018.
    b. random graphs — ongoing work with Ackerman, Freer, Roy, Yang.

Standard model of finite probability

Objects: natural numbers (e.g. `Bool=2`)

Deterministic morphisms: functions

Probabilistic morphisms (conditional probabilities): stochastic matrices
i.e. families $m \to P(n)$ = prob. distributions on n

Standard model of finite probability

Objects/types/sets: natural numbers (e.g. `Bool=2`)

Deterministic morphisms: functions

Probabilistic morphisms (conditional probabilities): stochastic matrices
i.e. families $m \to P(n)$ = prob. distributions on n

Problem. We can freely extend this to arbitrary sets, but we cannot work here with

```
p <- uniform      p <- beta(2,1)
```

etc.

Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

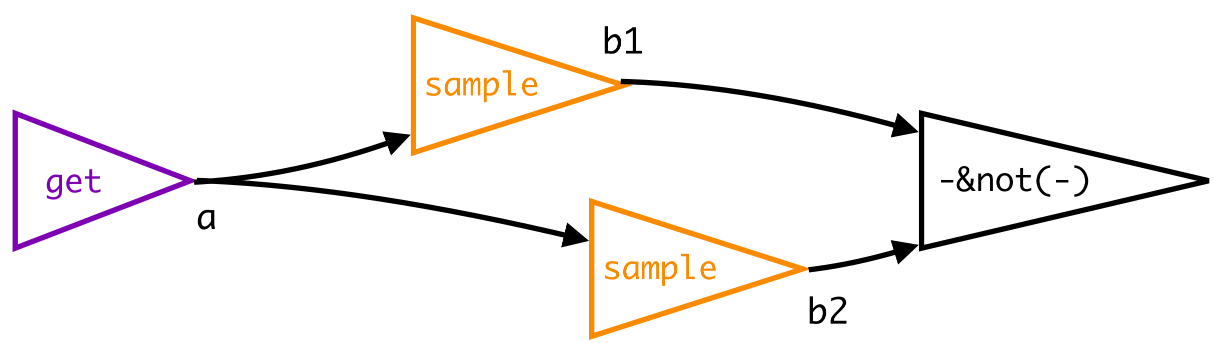
Interface:

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```

Example:

```
get() = uniform(0,1)
sample(p) = bernoulli(p)
```



Measure-based model of probability

($\Sigma_X \subseteq Powerset(X)$ closed under countable unions and complements)

Objects: Borel spaces (X, Σ_X)
e.g. countable discrete; I=$(\mathbb{R}, \mathscr{B}\mathrm{orel})$

Deterministic morphisms: measurable functions.

Probabilistic morphisms (conditional probabilities):
probability kernels

$$X \times \Sigma_Y \to [0,1]$$

i.e. measurably-parameterized probability measures.

Composition is by integration.

Example 2: Beta / bernoulli

Given an unknown coin, what is the probability of heads then tails?

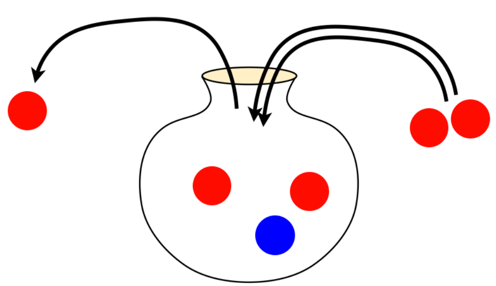
<u>Interface:</u>

```
get() : I
sample(I) : bool
```

```
a <- get()
b1 <- sample(a)
b2 <- sample(a)
return (b1 & not(b2))
```

<u>Another example:</u>

`get()` = *new urn*



`sample(p)` = *Pólya draw: one out, two in*

Operational model of probability

Objects: syntactic types
e.g. `I, bool` ...

Conditional probabilities:
programs mod contextual equivalence.

$$x : X \vdash P =_{\text{ctx}} Q : Y$$

if $\forall \mathscr{C} \,.\, \vdash \mathscr{C}[P], \mathscr{C}[Q] : \underline{n} \implies \mathscr{C}[P] = \mathscr{C}[Q]$

Combinatorial model of probability

Objects/types/sets: indexed sets $X \colon \mathbf{FinSet} \to \mathbf{Set}$

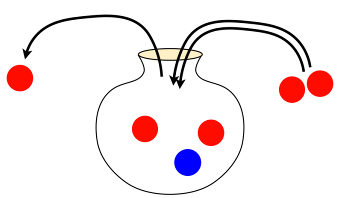
in particular, for each number n, a set $X(n)$.

e.g. $\underline{2}(n) = 2$; $I(n) = n$

Deterministic morphisms: natural families of functions.

Yoneda lemma: $X(n) = \mathrm{Nat}(I^n \to X)$

Intuition: I is a space of *urns*.



$\mathrm{Nat}(I \times I \to \underline{2}) = 2$

(no comparisons of urns)

Staton, Stein, Yang, Ackerman, Freer, Roy, ICALP 2018.

Combinatorial model of probability

Objects/types/sets: indexed sets $X : \mathbf{FinSet} \to \mathbf{Set}$
in particular, for each number n, a set $X(n)$.
e.g. $\underline{2}(n) = 2$; $I(n) = n$

Deterministic morphisms: natural families of functions.
Yoneda lemma: $X(n) = \mathrm{Nat}(I^n \to X)$

Given $X : \mathbf{FinSet} \to \mathbf{Set}$, generate $P(X) : \mathbf{FinSet} \to \mathbf{Set}$ by

$$1 \xrightarrow{\texttt{get}(i,j)} P(I)$$

$$I \xrightarrow{\texttt{sample}} P(\underline{2})$$

$$1 \xrightarrow{\texttt{bernoulli}(r)} P(\underline{2})$$

```
p <- get(i,j)
b <- sample p
```

=

```
b <- bernoulli (i/(i+j))
p <- get(i+b,j+1-b)
```

Combinatorial model of probability

Axioms:

- Conjagacy (*),
- axioms for finite probability,
- commutativity and discarding (affine).

Theorem: These axioms are Hilbert-Post complete.

Staton, Stein, Yang, Ackerman, Freer, Roy, ICALP 2018.

Given $X: \mathbf{FinSet} \to \mathbf{Set}$, generate $P(X): \mathbf{FinSet} \to \mathbf{Set}$ by

$$1 \xrightarrow{\texttt{get}(i,j)} P(I)$$

$$I \xrightarrow{\texttt{sample}} P(\underline{2})$$

$$1 \xrightarrow{\texttt{bernoulli}(r)} P(\underline{2})$$

```
p <- get(i,j)
b <- sample p
```

=

```
b <- bernoulli (i/(i+j))
p <- get(i+b,j+1-b)
```

Combinatorial model of probability

Objects/types/sets: indexed sets $X\colon \mathbf{FinSet} \to \mathbf{Set}$
in particular, for each number n, a set $X(n)$.
e.g. $\underline{2}(n) = 2$; $I(n) = n$

Deterministic morphisms: natural families of functions.
Yoneda lemma: $X(n) = \mathrm{Nat}(I^n \to X)$

Probabilistic morphisms (conditional probabilities):
e.g.
$\underline{n} \to P(\underline{m})$ = stochastic matrices;

$I^n \to P(\underline{2})$ = Bernstein polynomials in n variables.

Staton, Stein, Yang, Ackerman, Freer, Roy, ICALP 2018.

Three models of beta/bernoulli

Theorem: *For program expressions P,Q involving only* `get` *and* `sample`*, the following are equivalent:*

1. *P and Q are contextually equivalent.*

   $$\forall \mathscr{C}\,.\ \vdash \mathscr{C}[P], \mathscr{C}[Q] : \underline{n} \implies \mathscr{C}[P] = \mathscr{C}[Q]$$

2. *P and Q have the same measure-theoretic semantics as kernels* $X \times \Sigma_Y \to [0,1]$.

3. *P and Q have the same combinatorial semantics as natural transformations* $X \dot{\to} P(Y) : \mathbf{FinSet} \to \mathbf{Set}$.

Plan of talk:

1. Intuitive illustrations of symmetries in
    a. random graphs
    b. beta distributions / Pólya urns
2. Models for
    a. beta / Pólya urns — Staton, Stein, Yang, Ackerman, Freer, Roy, ICALP 2018.
    b. **random graphs** — ongoing work with Ackerman, Freer, Roy, Yang.

Building infinite random graphs

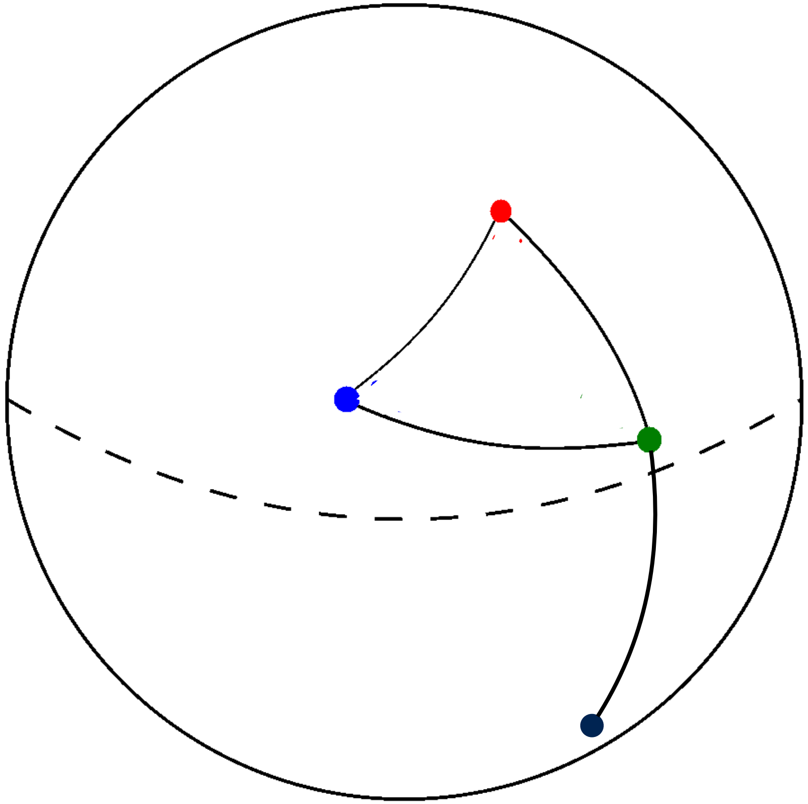
Interface:

```
get() : node
edge(node,node) : bool
```

```
a <- get()
b <- get()
return (edge(a,b))
```

Example:

```
get() = uniform Sₙ
edge(p,q)
  = if d(p,q) < π/2
    then True else False
```



Graphons

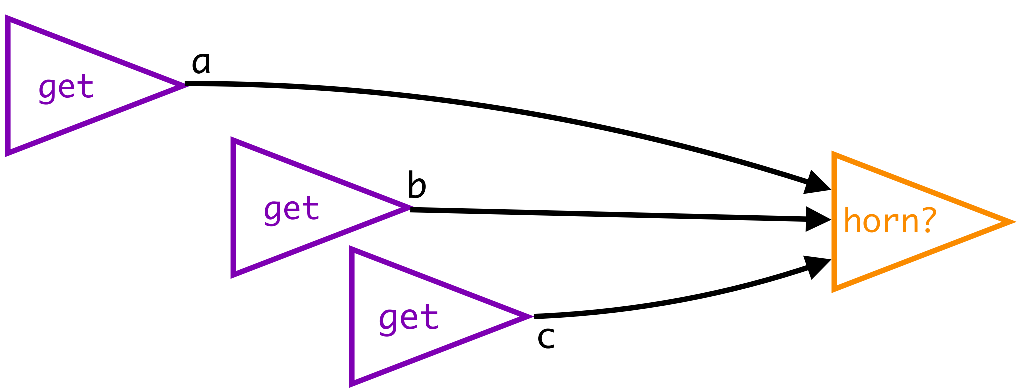
Interface:

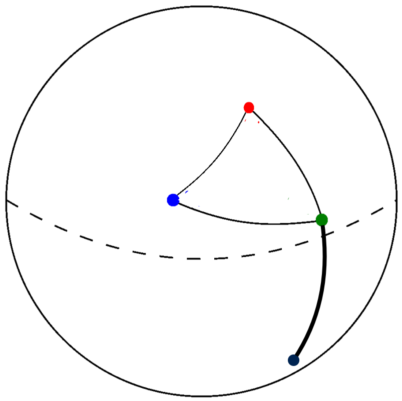
```
get() : node
edge(node,node) : bool
```

Example:

```
get() = uniform(0,1)
edge(p,q)
  = memoize_{p,q}(bernoulli(0.5))
```

```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
  && edge(a,c)
  && not(edge(b,c)))
```





building on
Bubeck, Ding, Eldan, Racz, 2015
Devroye, György, Lugosi, Udina, 2011

Graphons

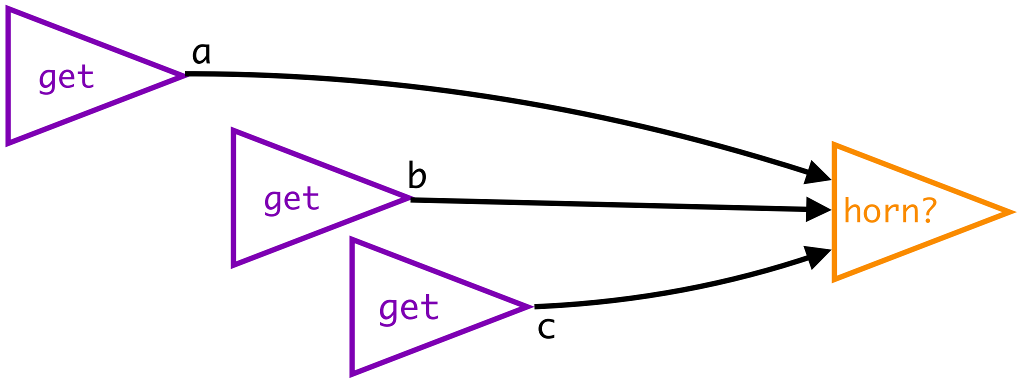
Interface:

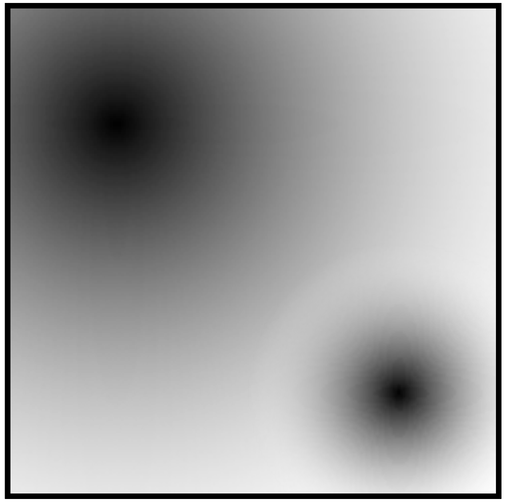
```
get() : node
edge(node,node) : bool
```

Example:

```
get() = uniform(0,1)
edge(p,q)
  = memoize_{p,q}(bernoulli(G(p,q)))
```

```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```





Graphons

Interface:

```
get() : node
edge(node,node) : bool
```

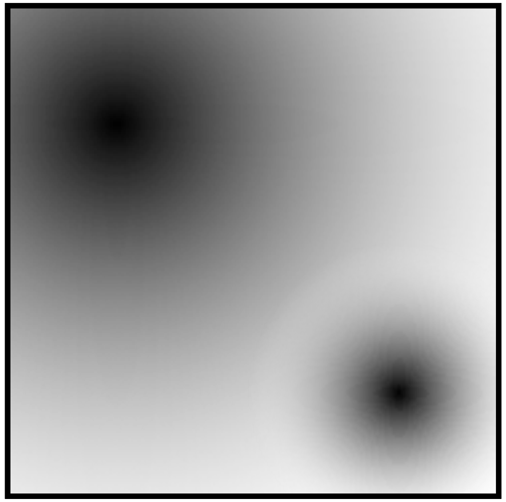
Example:

```
get() = uniform(0,1)
edge(p,q)
  = memoize_{p,q}(bernoulli(G(p,q)))
```

```
a <- get()
b <- get()
c <- get()
return (edge(a,b)
 && edge(a,c)
 && not(edge(b,c)))
```

A graphon is a measurable function

$$G : [0,1] \times [0,1] \to [0,1].$$



Lovász and Szegedy, J. Combin. Theory Ser. B., 2006.

Graphons

Interface:

```
get() : node
edge(node,node) : bool
```

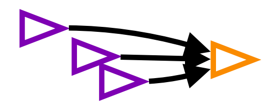
Example:

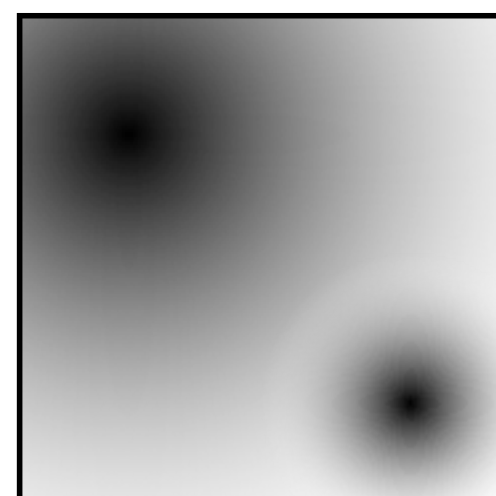
```
get() = uniform(0,1)
edge(p,q)
  = memoize_{p,q}(bernoulli(G(p,q)))
```

A graphon is a measurable function

$$G : [0,1] \times [0,1] \to [0,1].$$

Theorem. TFAE for an implementation:

- It satisfies data flow symmetry; 
- It is a graphon implementation (mod ctx equivalence).



Lovász and Szegedy, J. Combin. Theory Ser. B., 2006.

Combinatorial model of probability

Objects: indexed sets $X\colon \mathbf{FinSet} \to \mathbf{Set}$

in particular, for each number n, a set $X(n)$.

e.g. $\underline{2}(n) = 2$; $I(n) = n$

Deterministic morphisms: natural families of functions.

Yoneda lemma: $X(n) = \mathrm{Nat}(I^n \to X)$

Intuition: I is a space of *urns*.

$\mathrm{Nat}(I \times I \to \underline{2}) = 2$

(no comparisons of urns)

Staton, Stein, Yang, Ackerman, Freer, Roy, ICALP 2018.

Rado topos model of probability

Objects: indexed sets $X\colon \mathbf{FinGrph} \to \mathbf{Set}$ (+ sheaf condition)

in particular, for each graph g, a set $X(g)$.

e.g. $\underline{2}(g) = 2$; $V(g) = |g|$

e.g.
Johnstone OUP 2002
Pitts CUP 2013
Caramello, arXiv:1301.0300
Garner, notes 2014
Bojańczyk, Toruńczyk, LICS 2018

Deterministic morphisms: natural families of functions.

Yoneda lemma: $X(g) = \mathrm{Nat}(V^g \to X)$

Rado topos model of probability

Objects: indexed sets $X\colon \mathbf{FinGrph} \to \mathbf{Set}$ (+ sheaf condition)

in particular, for each graph g, a set $X(g)$.

e.g. $\underline{2}(g) = 2$; $V(g) = |g|$

e.g.
Johnstone OUP 2002
Pitts CUP 2013
Caramello, arXiv:1301.0300
Garner, notes 2014
Bojańczyk, Toruńczyk, LICS 2018

Deterministic morphisms: natural families of functions.

e.g.: $\mathtt{edge} : V \times V \to \underline{2}$

$\mathtt{edge}_g : |g| \times |g| \to 2$

Intuition: V is the vertex set of the Rado graph.

Rado topos model of probability

Objects: indexed sets $X\colon \mathbf{FinGrph} \to \mathbf{Set}$ (+ sheaf condition)

in particular, for each graph g, a set $X(g)$.

e.g. $\underline{2}(g) = 2$; $V(g) = g$

e.g.
Johnstone OUP 2002
Pitts CUP 2013
Caramello, arXiv:1301.0300
Garner, notes 2014
Bojańczyk, Toruńczyk, LICS 2018

Deterministic morphisms: natural families of functions.

e.g.: $\mathtt{edge} : V \times V \to \underline{2}$

Equivalently,

Objects/types/sets: continuous actions

$$\mathrm{Aut}(\mathrm{Rado}) \times A \to A$$

Deterministic morphisms: equivariant functions.

Rado topos model of probability

Objects: indexed sets $X\colon \mathbf{FinGrph} \to \mathbf{Set}$ (+ sheaf condition)

in particular, for each graph g, a set $X(g)$.

e.g. $\underline{2}(g) = 2$; $V(g) = g$

e.g.
Johnstone OUP 2002
Pitts CUP 2013
Caramello, arXiv:1301.0300
Garner, notes 2014
Bojańczyk, Toruńczyk, LICS 2018

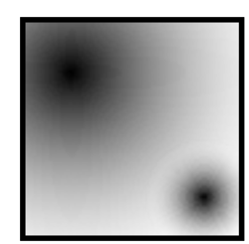
Deterministic morphisms: natural families of functions.

e.g.: $\mathtt{edge} : V \times V \to \underline{2}$

Given $X\colon \mathbf{FinGph} \to \mathbf{Set}$, generate $P(X)\colon \mathbf{FinGph} \to \mathbf{Set}$
by

$1 \xrightarrow{\mathtt{get}_W} P(V)$ for each graphon $W : [0,1]^2 \to [0,1]$

$\underline{m} \to P(\underline{n})$ for each stochastic matrix



Rado topos model of probability

Objects: indexed sets $X\colon \mathbf{FinGrph} \to \mathbf{Set}$ (+ sheaf condition)

in particular, for each graph g, a set $X(g)$.

e.g. $\underline{2}(g) = 2$; $V(g) = g$

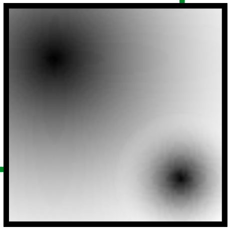
e.g.
Johnstone OUP 2002
Pitts CUP 2013
Caramello, arXiv:1301.0300
Garner, notes 2014
Bojańczyk, Toruńczyk, LICS 2018

Deterministic morphisms: natural families of functions.

e.g.: $\mathtt{edge} : V \times V \to \underline{2}$

Proposition: Each graphon induces an internal probability measure, i.e. a countably additive equivariant morphism

$$2^V \dot{\to} [0,1].$$ *Converse?*



Rado topos model of probability

Objects: indexed sets $X\colon \mathbf{FinGrph} \to \mathbf{Set}$ (+ sheaf condition)

in particular, for each graph g, a set $X(g)$.

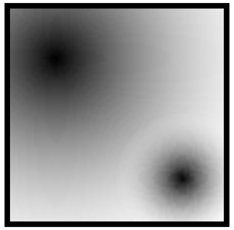
e.g. $\underline{2}(g) = 2$; $V(g) = g$

Theorem: The following data are equivalent.

1. A graphon

2. An probability measure on V satisfying Fubini.

Ackerman, Freer, Roy, Staton, Yang, unpublished.



Rado topos model of probability

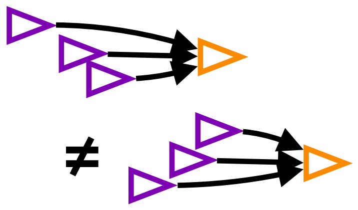
Objects: indexed sets $X\colon \mathbf{FinGrph} \to \mathbf{Set}$ (+ sheaf condition)
in particular, for each graph g, a set $X(g)$.
e.g. $\underline{2}(g) = 2$; $V(g) = g$

Defn. A probability measure on X is $2^X \dot{\to} [0,1]$ count'ly additive.

Measures on $X \times X$ are $2^{X \times X} \dot{\to} [0,1]$ (*not* product σ-algebra)

Fubini doesn't always hold but
for all $f: X \times Y \to \mathbb{R}^+$, $y \mapsto \int f(x,y)\,\mu(\mathrm{d}x)$ is a morphism.

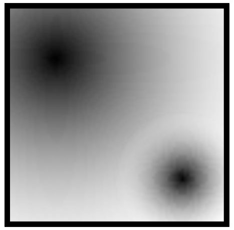
cf 'one-way Fubini' in non-standard analysis



Theorem: The following data are equivalent.

1. A graphon

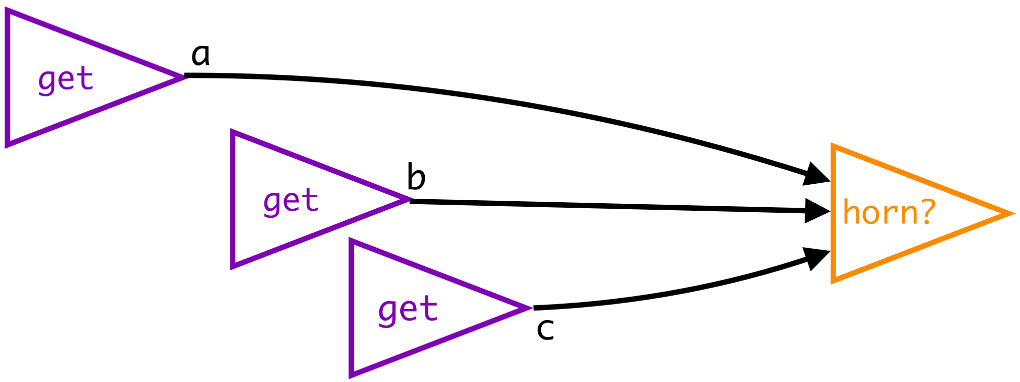
2. An probability measure on V satisfying Fubini.



Ackerman, Freer, Roy, Staton, Yang, unpublished.

Rado topos model of probability

Idea for building a model:
pick a graphon and use the probability submonad generated by it.

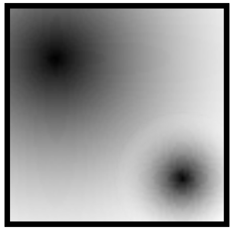


Theorem: The following data are equivalent.

1. A graphon

2. An probability measure on V satisfying Fubini.

Ackerman, Freer, Roy, Staton, Yang, unpublished.



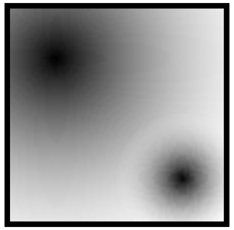
Proof of $2 \implies 1$. **General categorical proof:**

Theorem: The following data are equivalent.

1. A graphon

2. An probability measure on V satisfying Fubini.

Ackerman, Freer, Roy, Staton, Yang, unpublished.



Proof of $2 \implies 1$. **General categorical proof:**

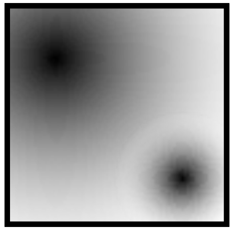
Consider an extensive category with a strong monad P*, such that* $\mathrm{Hom}(m, P(n))$ *corresponds to stochastic relations.*

Theorem: The following data are equivalent.

1. A graphon

2. An probability measure on V satisfying Fubini.

Ackerman, Freer, Roy, Staton, Yang, unpublished.



Proof of $2 \implies 1$. **General categorical proof:**

Consider an extensive category with a strong monad P*, such that* $\mathrm{Hom}(m, P(n))$ *corresponds to stochastic relations.*

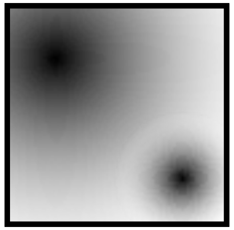
Let $(V, E : V^2 \to 2)$ *be an internal graph and* $g : 1 \to P(V)$ *be a morphism satisfying Fubini.*

Theorem: The following data are equivalent.

1. A graphon

2. An probability measure on V satisfying Fubini.

Ackerman, Freer, Roy, Staton, Yang, unpublished.



Proof of $2 \implies 1$. **General categorical proof:**

Consider an extensive category with a strong monad P, *such that* $\mathrm{Hom}(m, P(n))$ *corresponds to stochastic relations.*

Let $(V, E : V^2 \to 2)$ *be an internal graph and* $g : 1 \to P(V)$ *be a morphism satisfying Fubini.*

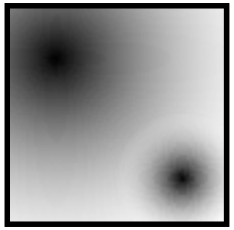
Then the finite random graphs $1 \xrightarrow{g^n} P(V^n) \xrightarrow{E^n} P(2^{n^2})$ *form a consistent local graph model.* Lovász and Szegedy, J. Combin. Theory Ser. B., 2006.

Theorem: The following data are equivalent.

1. A graphon

2. An probability measure on V satisfying Fubini.

Ackerman, Freer, Roy, Staton, Yang, unpublished.



Plan of talk:

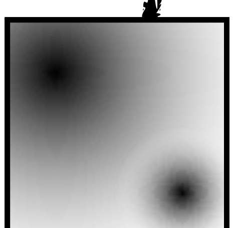
1. Intuitive illustrations of symmetries in
   a. random graphs
   b. beta distributions / Pólya urns
2. Models for
   a. beta / Pólya urns

      Staton, Stein, Yang, Ackerman, Freer, Roy, ICALP 2018.

      combinatorial model in $[\mathbf{FinSet}, \mathbf{Set}]$
   b. random graphs

      ongoing work with Ackerman, Freer, Roy, Yang.

      graphons as measures in $\mathrm{Cts}(\mathrm{Aut}(\mathrm{Rado}))$



Categorical models of probability with symmetries

What's next?

1. What about other interfaces, e.g.
   a. hierarchical graphs? Jung, Lee, Staton, Yang, Annales Henri Lebesgue, 2020.
   b. higher order functions? Heunen, Kammar, Staton, Yang, LICS 2017
2. What about recovering limiting structures from finite ones? Jacobs, Staton, CMCS 2020. Dahlqvist, Danos, Garnier. CONCUR 2016
3. What is synthetic probability theory? Fritz, Jacobs, Simpson, +++ …
4. What are the connections with non-standard approaches?