Systems that Model their Environments (and Systems that Plan)

Nathaniel Virgo

Earth-Life Science Institute (ELSI),
Tokyo, Japan

funded by the John Templeton Foundation

Introduction

Philosophically motivated:
I research origins of life, and wanted to know: **what is an agent?**

i.e. what makes a system **try to do something?**

As part of that, I'm interested in what it means for **one system to model another**, which will be most of the talk.

In particular, **where does the model live in relation to the system?**

Outline

- Background: category-theoretic approach to conjugate priors
- Extension to Bayesian filtering
- How does this arise more abstractly?
  → strongly representable Markov categories
- very brief sketches of approaches to "agents that plan"

Conjugate Priors

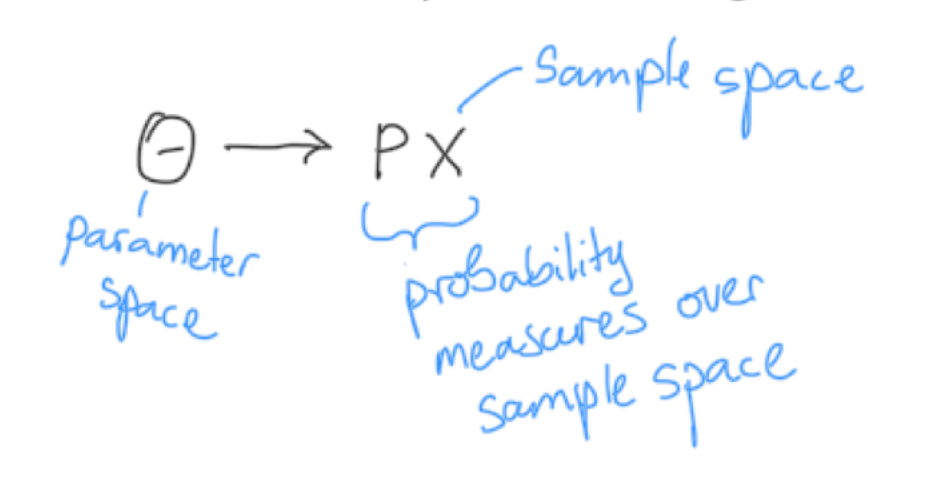
reference: Bart Jacobs 2020
A Channel-Based Perspective on Conjugate Priors
Mathematical Structures in Computer Science 30, p. 44–61

"The main result of this paper... is mathematically trivial. But it is not entirely trivial to see that this result is trivial" ← this applies to most of the talk.

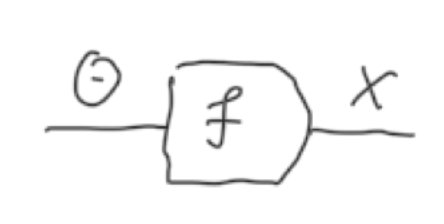
A useful perspective came from Jacobs' paper on conjugate priors.

What is a conjugate prior?

Start with a statistical model, which is just a (measurable) function



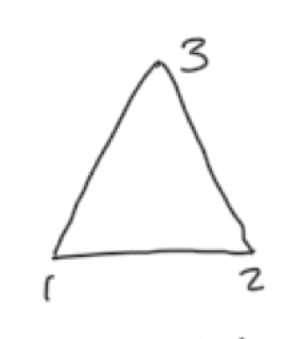
If we work in the Kleisli category of a distribution monad (or any other Markov category) then this is just a morphism



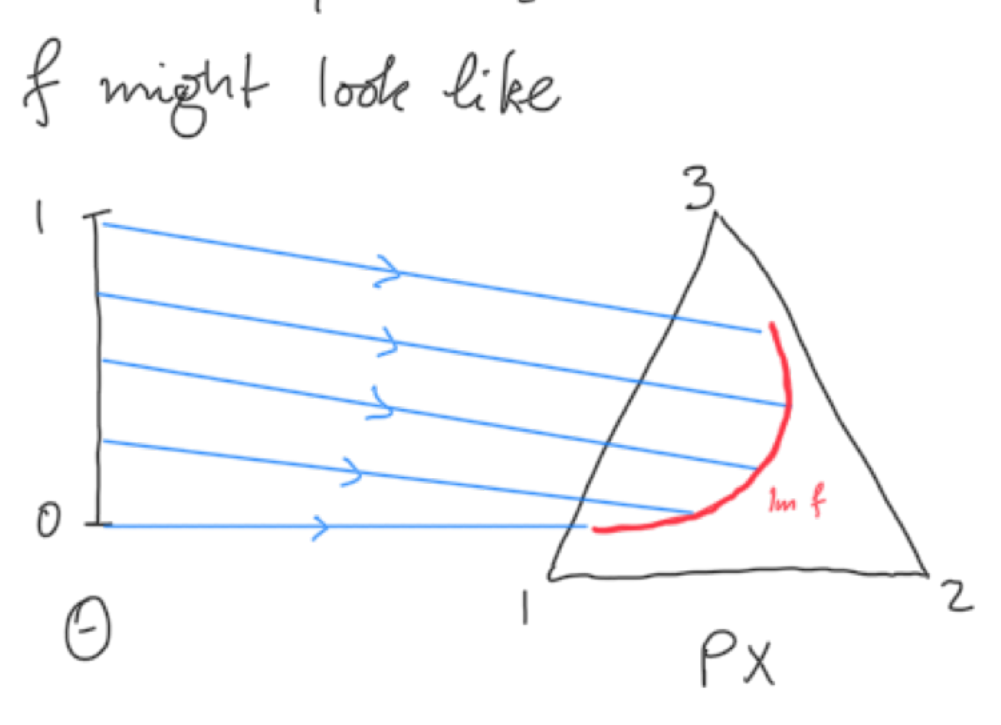
Think of this as a family of distributions

e.g. if $X = \{1, 2, 3\}$ and $\Theta = [0, 1]$

then

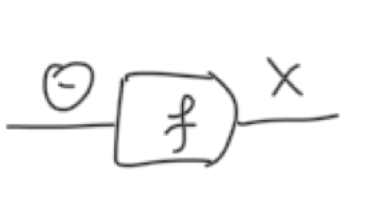


and f might look like



So we can think of the image of f as a subset of PX, parametrised by Θ.

Typical inference problem:

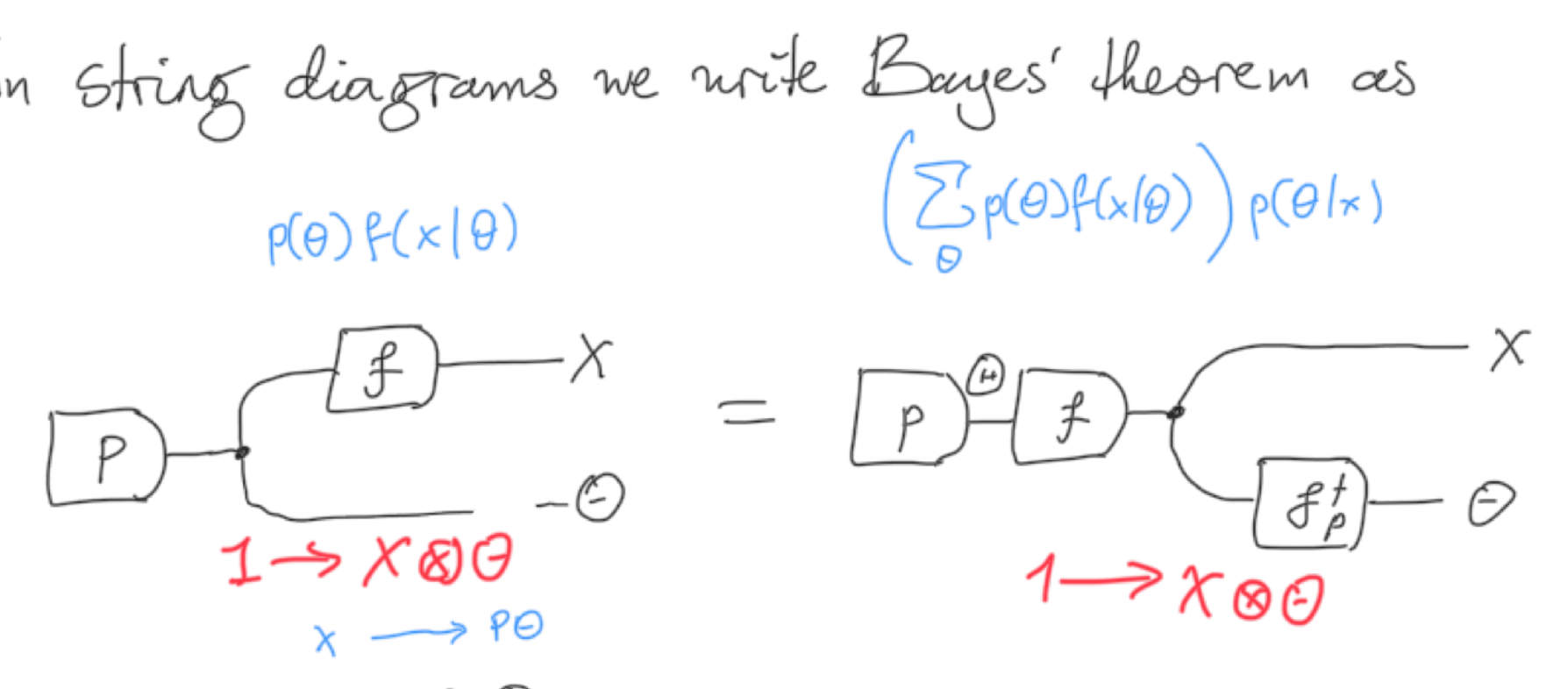
we have a statistical model 

we have some prior $p \in P\Theta$ (i.e. the "true" $\theta \in \Theta$ is unknown)

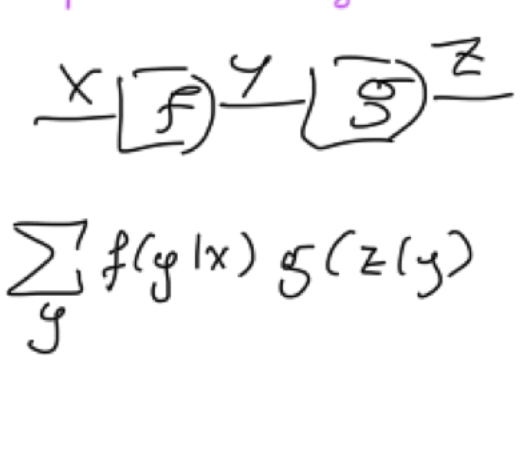
and some data $x \in X$, which is a sample from $f(\theta)$ ← the unknown "true" θ

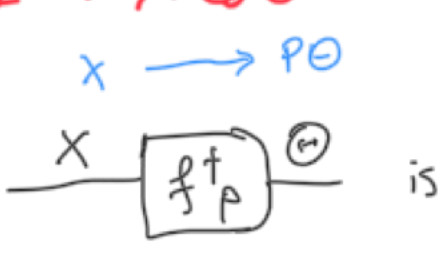
and we want to update to a posterior $\in P\Theta$.

In string diagrams we write Bayes' theorem as

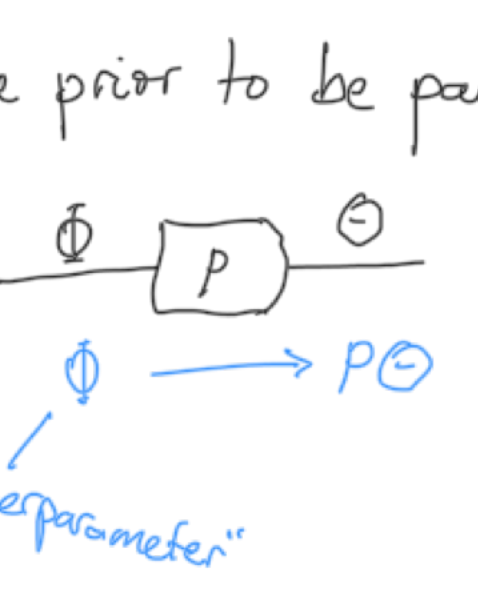


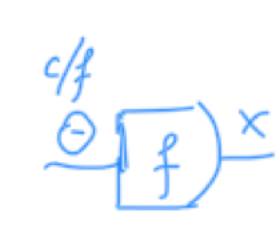
note on how Markov kernels compose, added during the talk:



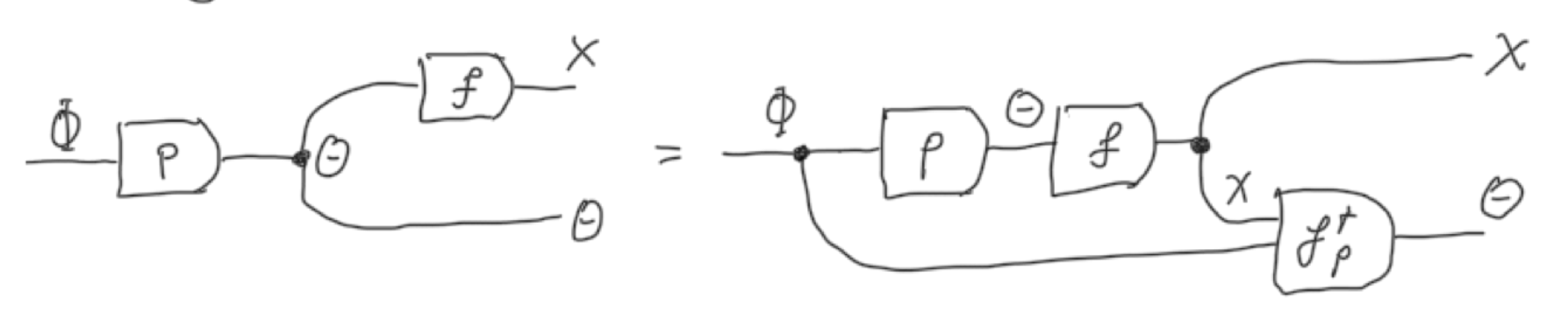
Here,  is the "Bayesian inverse" — it takes in the data $x \in X$ and returns the posterior $\in P\Theta$.

Sometimes we want the prior to be parametrised as well.

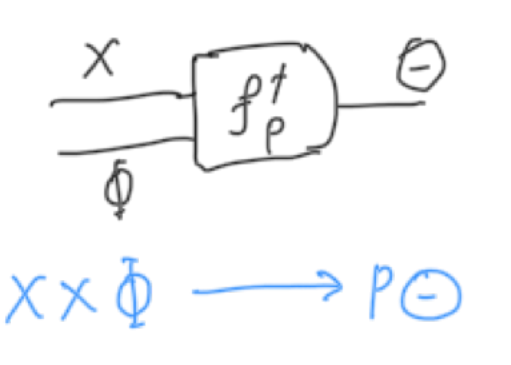


c/f 

In this case Bayes' theorem is

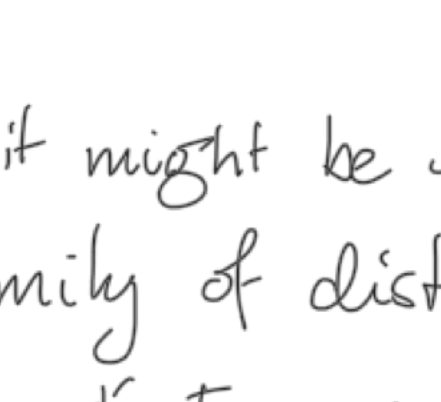


This time

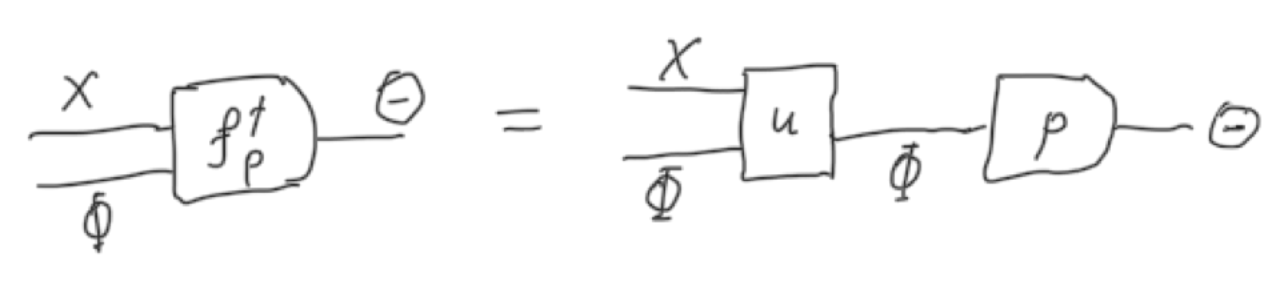


takes the data $x \in X$ and $\phi \in \Phi$ parametrising the prior and returns the posterior in $P\Theta$.

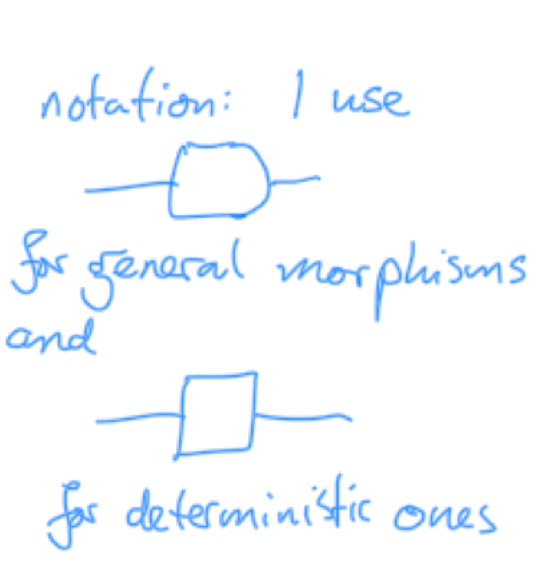
But if we are lucky then it might be that the posterior always lies in the family of distributions defined by p. i.e. there exists $\phi' \in \Phi$ such that the posterior is $p(\phi')$.



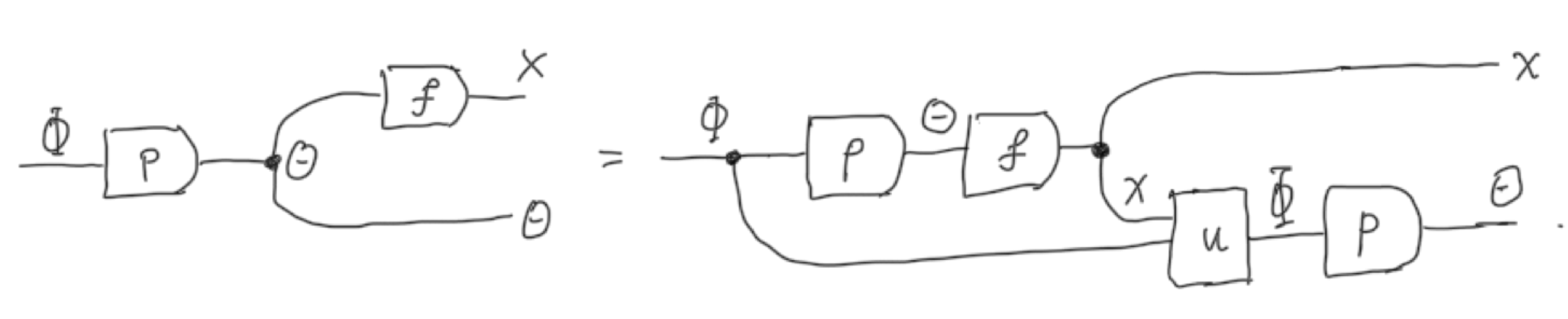
This means



for some (measurable) function $u : X \times \Phi \to \Phi$.



In this case Bayes' theorem becomes



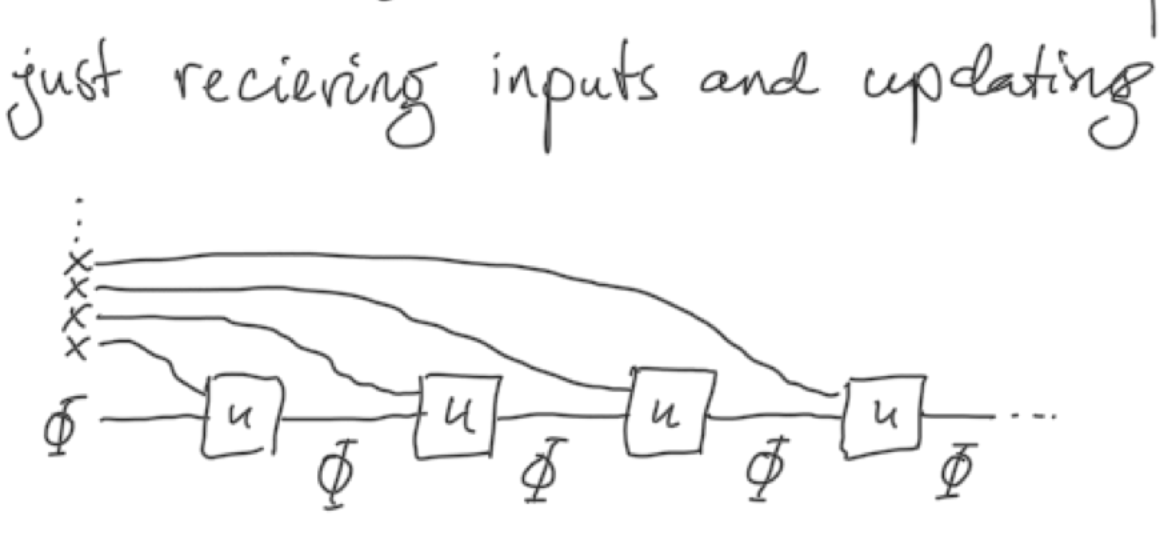
Given a model f, a conjugate prior is (u, p) satisfying this equation.

(This string diagram equation is one of the main points of Jacobs' paper.)

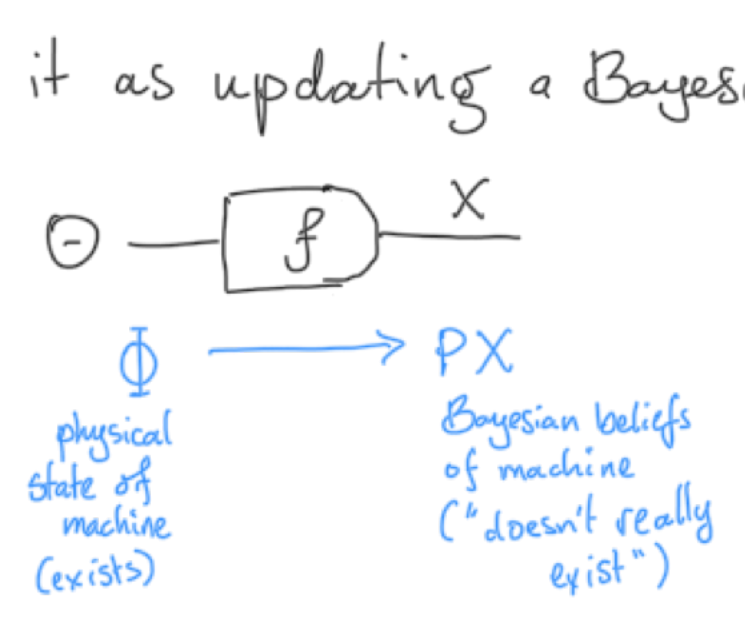
The point: if you want to implement Bayes on a computer, and if you can find a conjugate prior, then **the only thing you need to implement is the function u.**

In practice this is often a very simple function.

So we have a situation where the actual physical system is just recieving inputs and updating its state

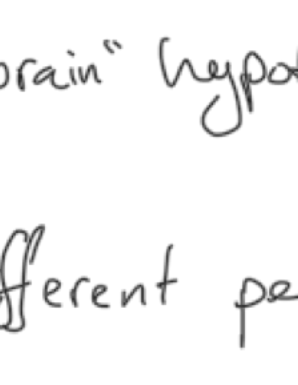


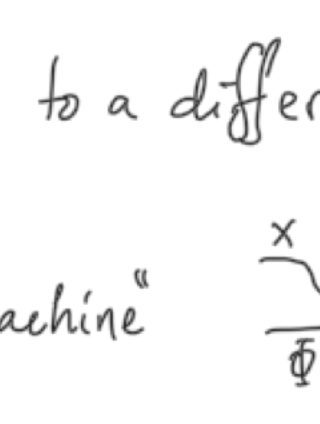
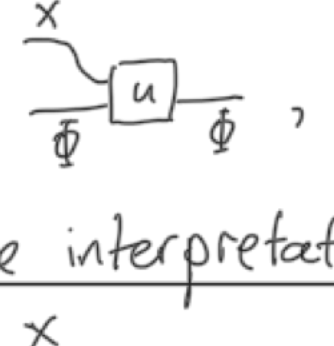
But we interpret it as updating a Bayesian belief via the map

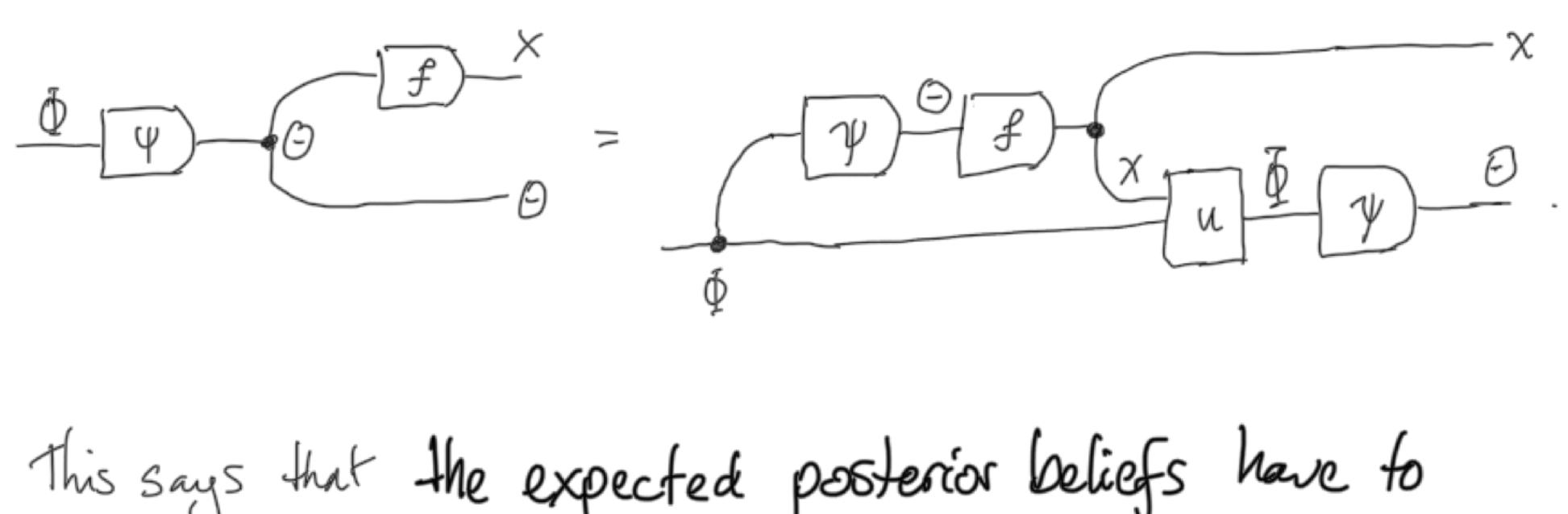


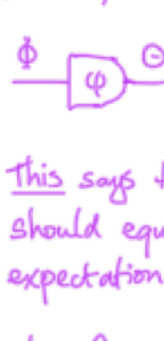
Similar ideas exist in neuroscience. (the "Bayesian brain" hypothesis)

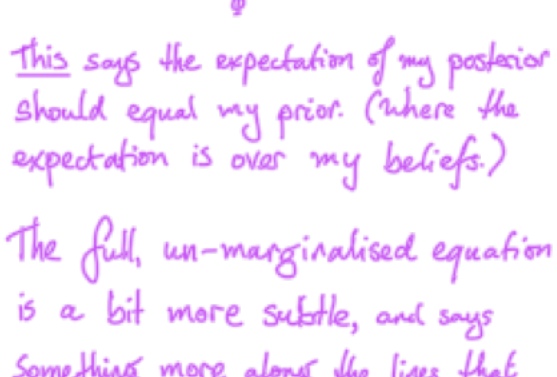
So we switch to a different perspective:

Given a "machine"  ,

a Bayesian inference interpretation is a model  and an "interpretation map"  satisfying the same equation as above,



Note from after the talk: I was a bit confused here – if we take a marginal of this equation, by post-composing both sides with  , then we get



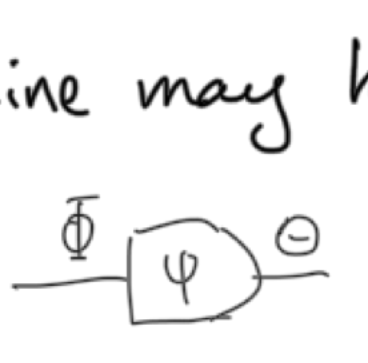
This says the expectation of my posterior should equal my prior. (where the expectation is over my beliefs.)

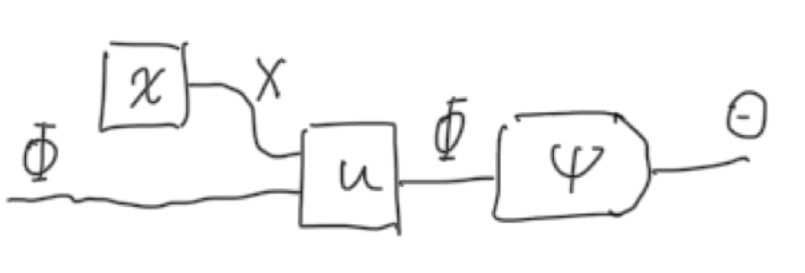
The full, un-marginalised equation is a bit more subtle, and says something more along the lines that given an observation x, my posterior beliefs should equal my current beliefs conditioned on X taking that value. (Up to an "almost surely", since conditionals are only almost surely defined in the first place.)

This says that **the expected posterior beliefs have to match the prior beliefs**, where the expectation is over the prior beliefs. (c/f martingales)

(side point: it matters that the machine is deterministic — this equation isn't strong enough otherwise.)

In general, **a machine may have many interpretations.**

In particular, if  is a valid interpretation map

then so is  , for any $x \in X$

(assuming has full support).

With the right definitions of morphisms, interpretations form a functor $\text{Machines}^{op} \to \text{Cat}$.

Taking the Grothendieck construction gives us a category of "Bayesian reasoners" = machines equipped with interpretations.

Before returning to this, I want to talk about the extension to Bayesian filtering.

Bayesian Filtering Interpretations

work with Martin Biehl, Simon McGregor

End goal: design (or reason about) agents that model their environments in order to achieve a goal.

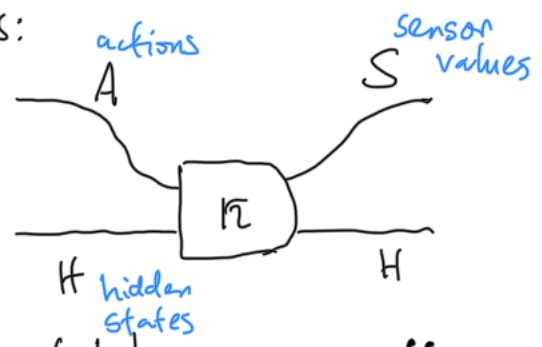
→ keep track of environment state given data

→ reason about how environment will respond to actions.

(N.B. We're assuming we already have a model
—this is POMDPs rather than reinforcement learning.)

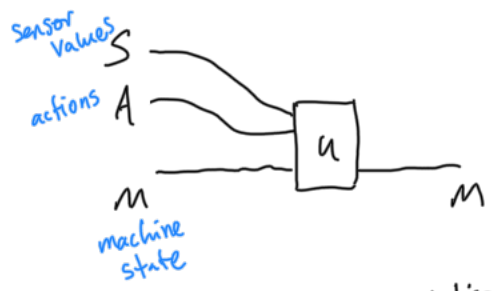
The first point can be achieved by "Bayesian filtering" but I will present a generalised version of filtering that also includes actions.

Suppose we have an environment model that looks like this:



this is meant to be **an agent's model of its environment** — it need not match the actual environment.

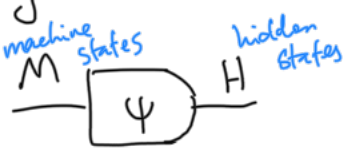
Then consider a "machine"



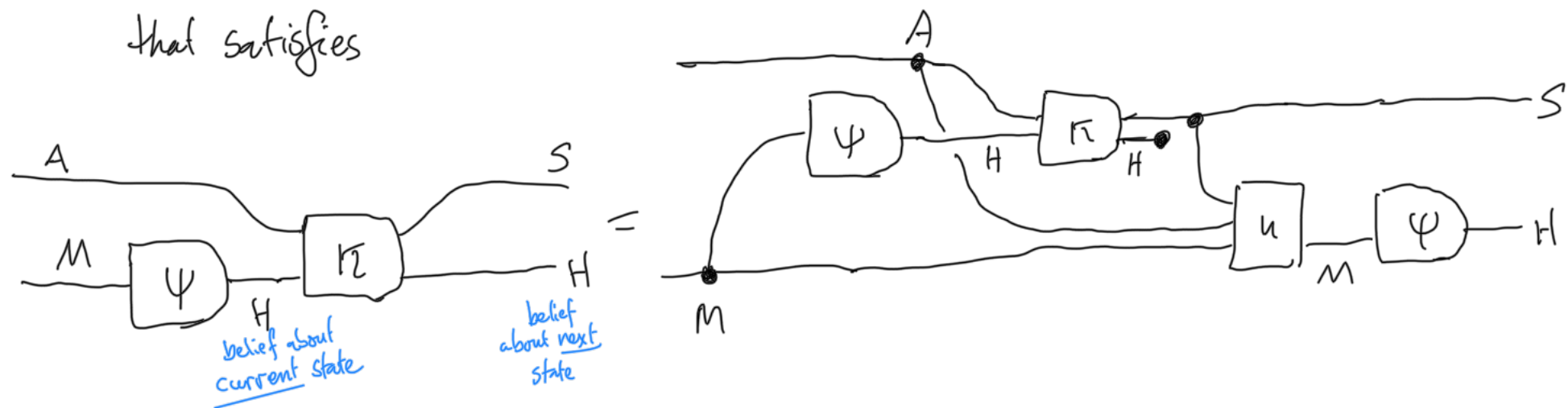
the machine's job is to answer (probabilistically) "if I take action $a \in A$ and recieve sensor value $s \in S$, then what state will the environment be in?"

It takes actions as inputs because the choice of what action to take has to be taken by something else.

Finally consider an interpretation map



that satisfies



We call (ψ, π) a Bayesian filtering interpretation of u.

This is the same principle as before. It says that for every action $a \in A$, the agent's prior beliefs about the next hidden state have to match its posterior beliefs about what will then be the current hidden state, if the data $s \in S$ is sampled from a distribution that matches its current beliefs.

Returning to the more abstract picture

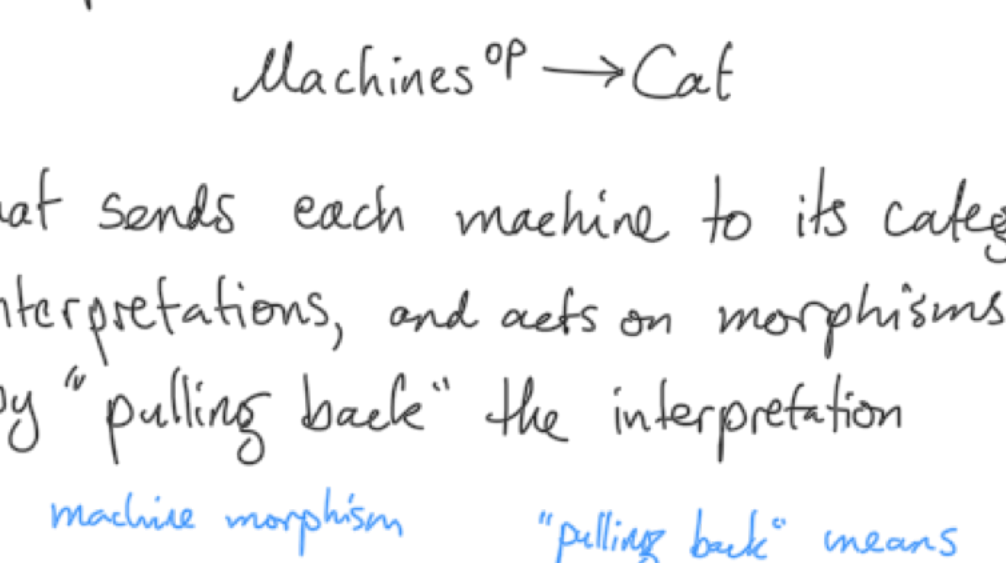
(stuff I figured out last weekend.)

Note: this section is fairly involved, but in the end the only punchline is that Bayesian filtering "pops out" of the machinery of "strongly representable Markov categories". ("...it's not entirely trivial to see that this result is trivial.")

As with conjugate priors, Bayesian filtering interpretations can be made into a functor

$$\text{Machines}^{op} \to \text{Cat}$$

that sends each machine to its category of interpretations, and acts on morphisms by "pulling back" the interpretation



We can also fix κ when doing this, so that each machine is interpreted as reasoning about the same environment. (If such an interpretation exists.) In that case there is no useful notion of morphism of interpretations, and we just get a functor

$$\text{Machines}^{op} \to \text{Set}.$$

Taking the category of elements yields a category of "filtering reasoners" = machines equipped with filtering interpretations, with model given by κ.

What we would **like** is for this category to have a terminal object. The idea is that the state space of the terminal object would be PH, with the trivial interpretation

$$PH \xrightarrow{id_{PH}} PH.$$

Then for any other object (= machine + interpretation) there would be a unique map into the terminal object given by its interpretation map, seen as a map $\psi: M \to PH$ (machine states → states of terminal reasoner)

Unfortunately, **this is not true**, for subtle reasons that involve the behaviour of machines when given "subjectively impossible inputs", i.e. values for S that are not in the support of their beliefs.

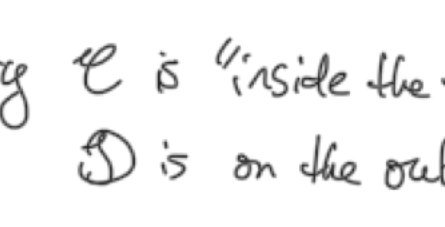
However, there is a way to make it true. For this we need some machinery from

Representable Markov Categories and Comparison Experiments in Categorical Probability

Fritz, Gonda, Perrone, Rischel
arXiv:2010.07416 (2020)

I will use "functor tube"/"functor box" notation (see, e.g. Fritz, Perrone – Bimonoidal Structure of Probability Monads)

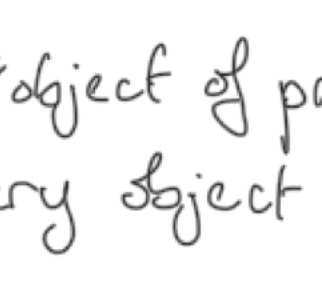
for $F: \mathcal{C} \to \mathcal{D}$,
$FX \xrightarrow{Ff} FY$ is written as



the category $\mathcal{C}$ is "inside the tube", $\mathcal{D}$ is on the outside.

A Markov category is representable if it is the Kleisli category of a probability monad.

In this case it has an "object of probability measures" PX, for every object X, which I'll write



There is also a "sampling map" $PX \to X$, which I'll write

(This is the identity id_{PX}, interpreted as a Kleisli morphism.)
We interpret it as taking a distribution as input and returning a sample from it, stochastically.

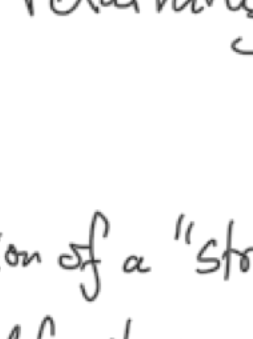
We need the definition of a "strongly representable Markov category" — to define it we need a couple of other definitions.

Definition (Fritz)

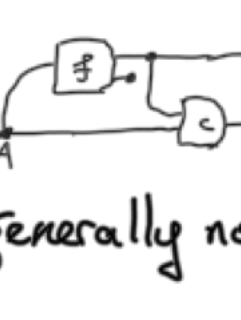
A synthetic approach to Markov kernels [...]
Tobias Fritz
Advances in Mathematics 370 (2020)

A Markov category has conditionals if

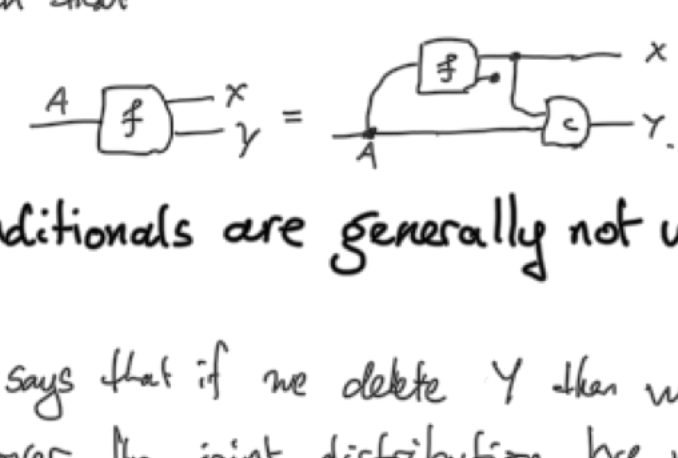
for every morphism



there exists a morphism



such that



Conditionals are generally not unique.

This says that if we delete Y then we can recover the joint distribution by multiplying by the conditional. In the discrete case,

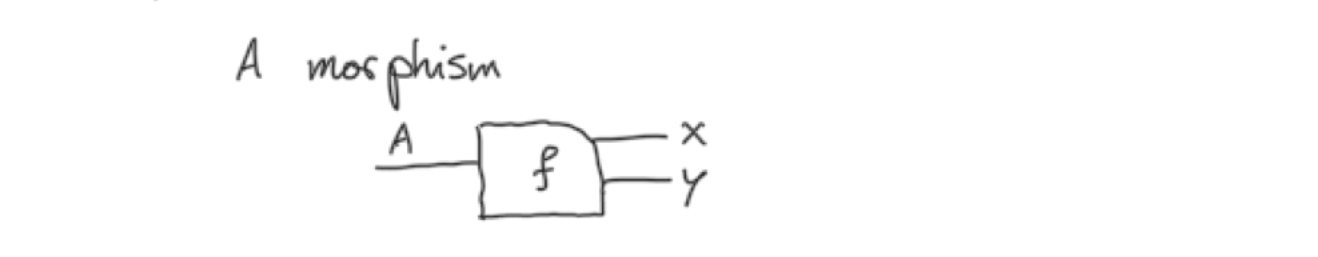
$$f(x, y \mid a) = f(x \mid a)\, c(y \mid x, a)$$

a generalisation of $p(x,y) = p(x)\, p(y \mid x)$.

Definition (Fritz, Gonda, Perrone, Rischel)

A morphism

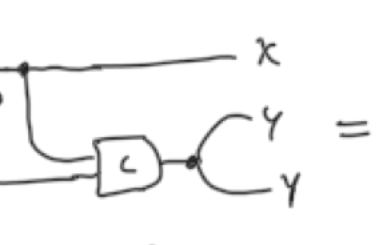
is deterministic given X if for any choice of conditional,



(if this is true for one conditional it is true for all of them.)

This says that conditionals are almost surely deterministic.

Now suppose we have a morphism



, deterministic in X.

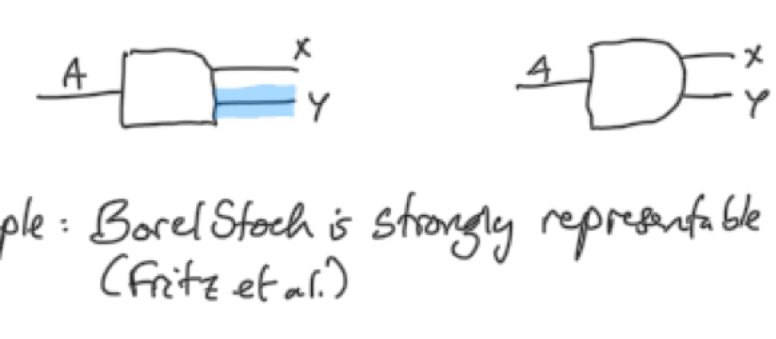
$A \to X \otimes PY$

Then we can compose with the sampling map to get

of type $A \to X \otimes Y$.

A strongly representable Markov category is one where this establishes a bijection

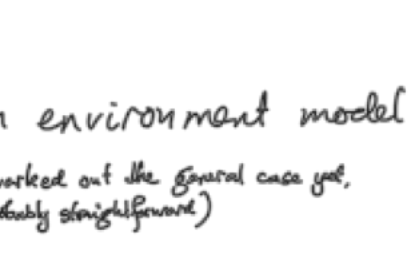
$$\mathcal{C}_{det|X}(A, X \otimes PY) \cong \mathcal{C}(A, X \otimes Y).$$



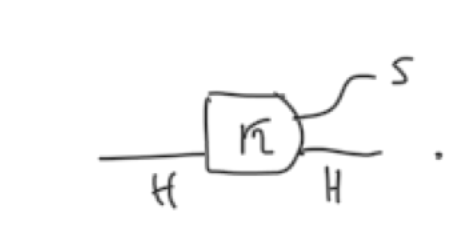
Known example: BorelStoch is strongly representable (Fritz et al.)

Back to machines and interpretations.

Consider an environment model with trivial input, (I haven't worked out the general case yet, but it's probably straightforward)



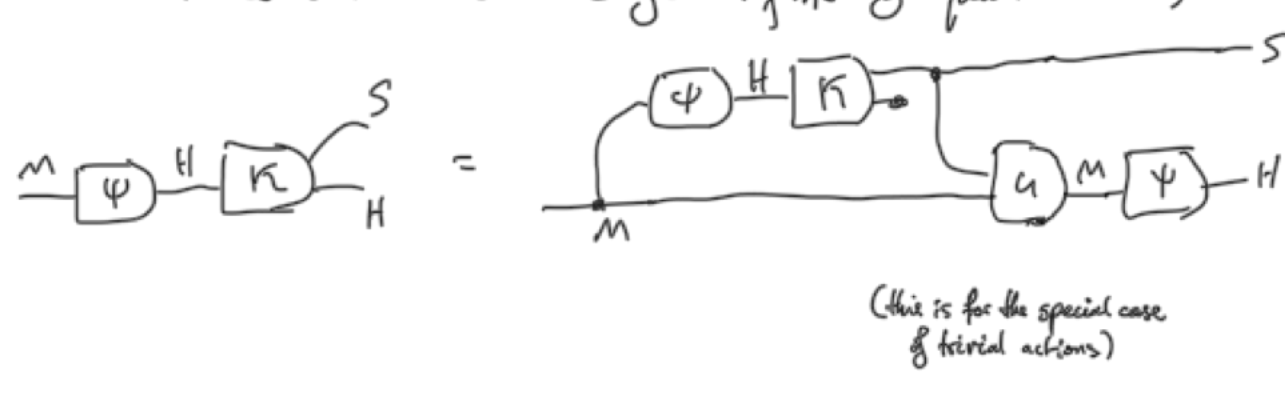
Now pre-compose this with the sampling map



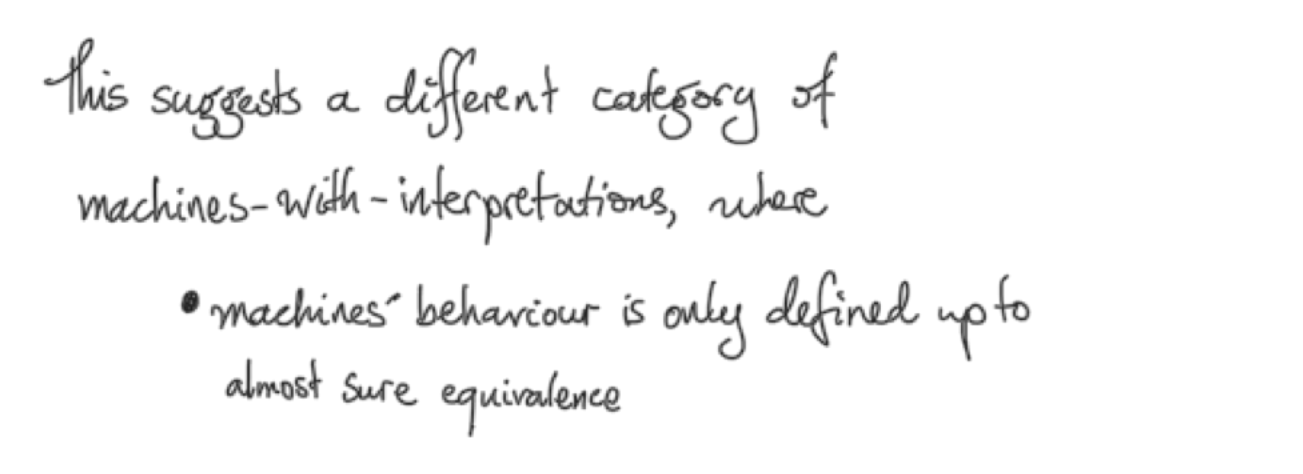
then we can use the bijection above to get

where c is both almost-surely deterministic in the sense above, and only defined up to almost sure equality.

c here is not arbitrary — it is performing Bayesian updates. Indeed we have



which looks a lot like the Bayesian filtering equation above,

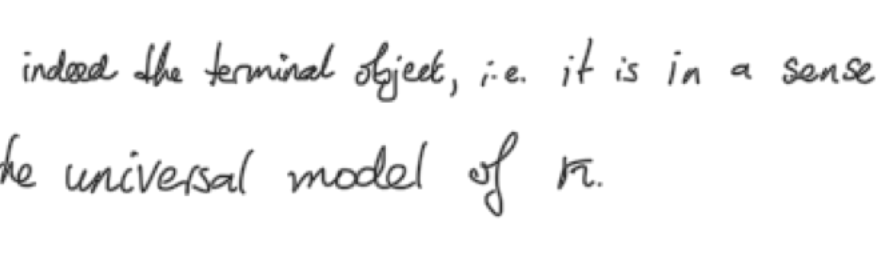


(this is for the special case of trivial actions)

This suggests a different category of machines-with-interpretations, where

- machines' behaviour is only defined up to almost sure equivalence
- machines are only required to be almost-surely deterministic

In this category the object



is indeed the terminal object, i.e. it is in a sense the universal model of κ.

In the end, the main point here is that Bayesian filtering "pops out" of strongly representable Markov categories in a simple way, and the same machinery might be useful for other similar questions about Bayesian modelling.

Systems that Plan

I only talked about systems that model their environment, but an obvious next step is systems that use their model to plan actions. Here I will only sketch approaches that we are taking to that.

As with inference, we're interested not in "what does it mean for a system to actually have a plan/goal", but "what does it mean to consistently interpret a system as planning/reaching for a goal."

1. Interpreting systems as solutions to POMDPs

work with Martin Biehl

Idea: we can interpret a system as trying to solve a POMDP task if it is an optimal solution to that task (details worked out in a similar framework to the one I presented)

2. Coalgebraic approach

ongoing work with Simon McGregor, Timo rl

uses a technically different framework from today's one

main result: **if a system is an optimal solution to a certain kind of planning task, then it can be interpreted as doing Bayesian filtering.**

(in a sense that looks like what I defined if you squint a bit.)

3 Systems theory

ongoing work with Matteo Cappucci

→ forget about internal models entirely and figure out what it means for a system to optimally regulate its actual environment, in a very general sense using categorical systems theory.

Conclusions & thanks