CS4340: Probabilistic Programming Seminar

Lecture 1

A coin is tossed 3 times.
What is the probability that at least one head was obtained?

All face cards are removed from a pack of 52 well-shuffled cards.
From the remaining 40 cards, 4 cards are drawn randomly.
What is the probability that 4 cards are from different suits and denominations?

What is the probability of Delft flooding
after a 5-day constant rain of 50 mm in Leidschendam?

All these questions fit the same format:

What is the probability of X given Y, $p(X|Y)$?

What is the probability of X given Y, $p(X|Y)$?

The ability to answer this question is essential for trustworthy AI

What is the probability of X given Y, p(X|Y)?

This course will be

- About computing this quantity
- While making *X, Y,* and *p(X|Y)* complicated/complex
- Even without knowing what *X, Y,* and *p(X|Y)* are

Outline for today

- What this course is about: probabilistic programming
    - What are probabilistic programs?
    - The anatomy of a probabilistic program

- How will the course work

Wait a minute….

Isn't $p(X|Y)$ just a classifier? We did the in a machine learning course.

Discriminative ML

$$p(X|Y) \quad \rightarrow \quad f(x) = y$$

- Weak assumptions about the process
- Only rely on data, needs lots of data
- Difficult to impart human input, and get it from the system
- Poor at estimating uncertainty

Generative ML

$$p(X|Y) \quad \rightarrow \quad p(X,Y)$$

- Strong assumptions about the process
- Works with little or no data
- Flexible

Bayesian Inference

Stuff you can't see
but want to know
-
"Latent Variable"

"Observation"

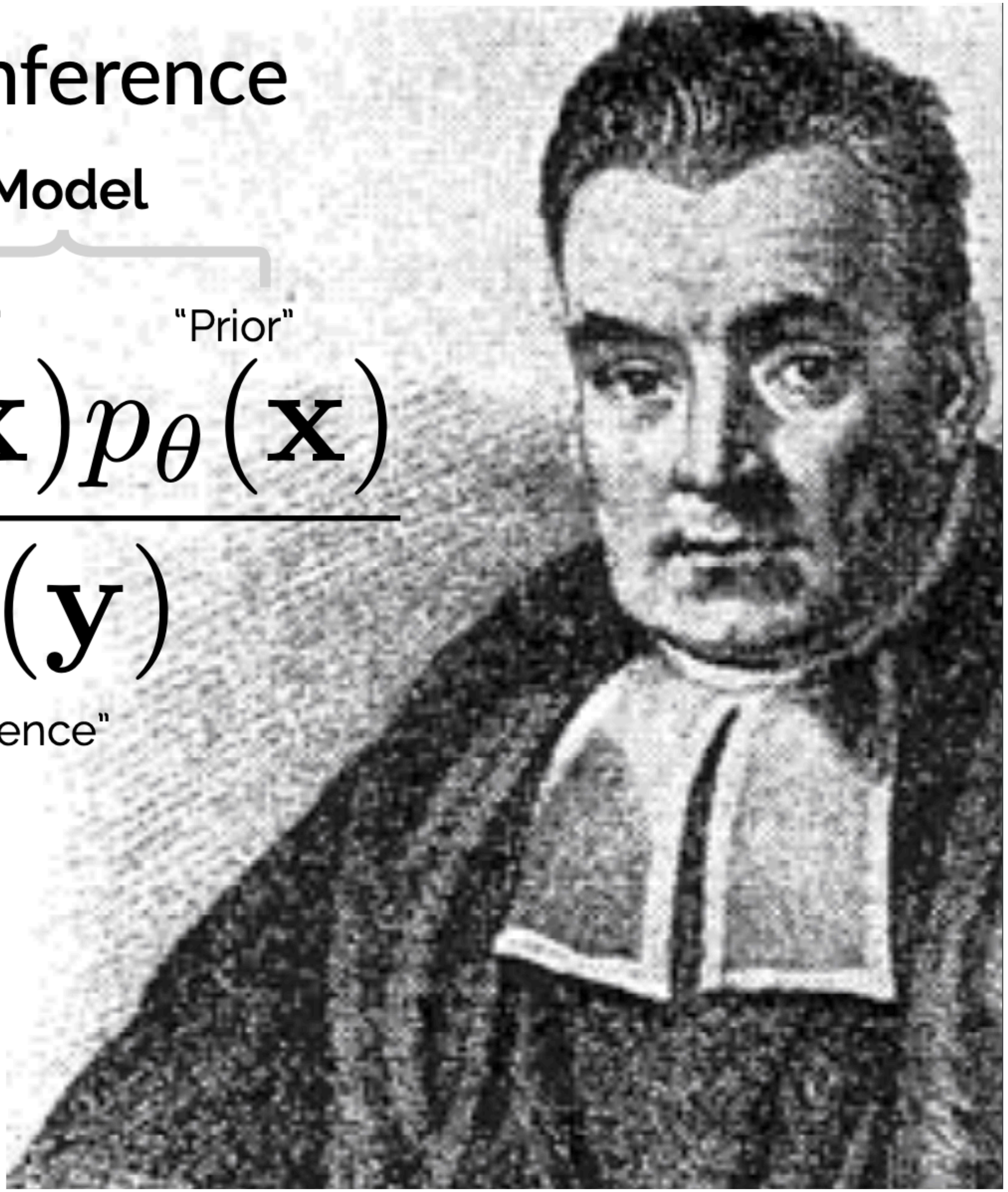
Model

"Likelihood" "Prior"

$$p_\theta(\mathbf{x}|\mathbf{y}) = \frac{\overbrace{p_\theta(\mathbf{y}|\mathbf{x})p_\theta(\mathbf{x})}^{\text{Model}}}{p_\theta(\mathbf{y})}$$

"Posterior"

"Evidence"



Probabilistic reasoning

Stuff you can observe

Stuff you can't see but want to know

How what we see gets generated

What stuff is allowed

$$p_\theta(\mathbf{y}, \mathbf{x}) = p_\theta(\mathbf{y}|\mathbf{x})p_\theta(\mathbf{x})$$

"Joint"

"Likelihood"

"Prior"

Parameters

A few more detailed examples

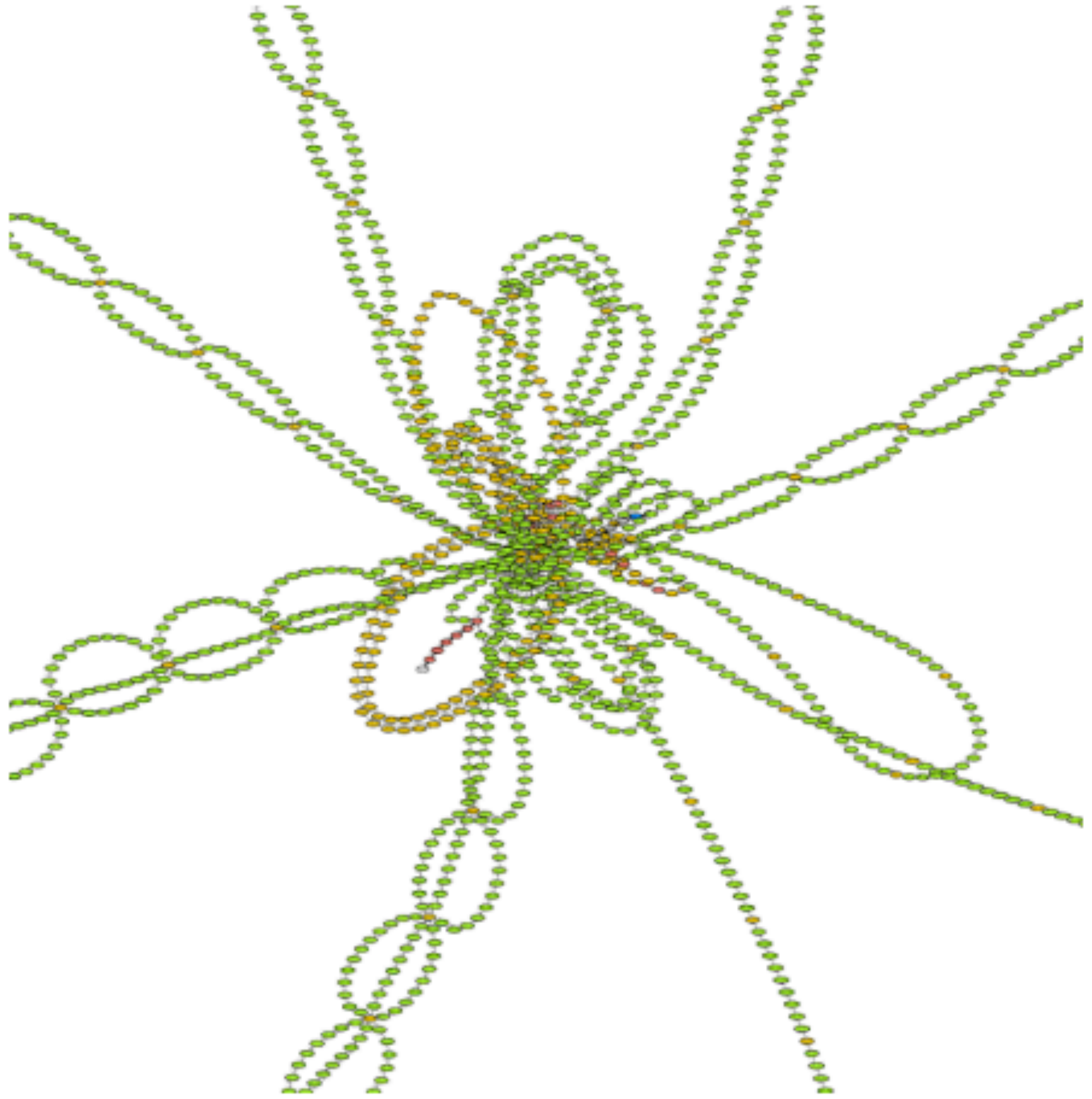
What are we interested in?

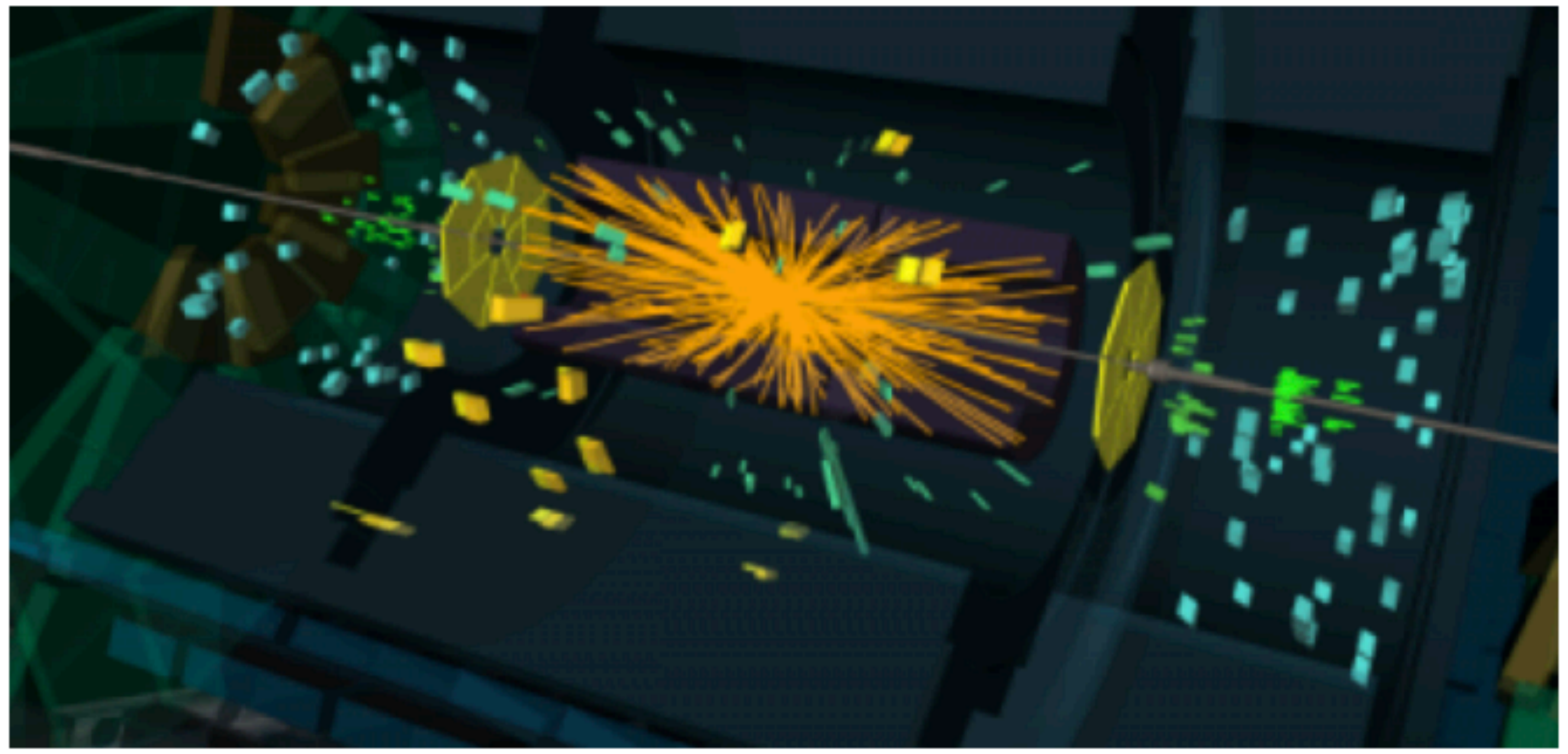
Particle types

What can we observe?

LHC detector response

$p_\theta(\mathbf{x})$ $\rightarrow$ $p_\theta(\mathbf{y}|\mathbf{x})$





Standard **model**/ATLAS detector **simulator**

1M+ C++ LOC

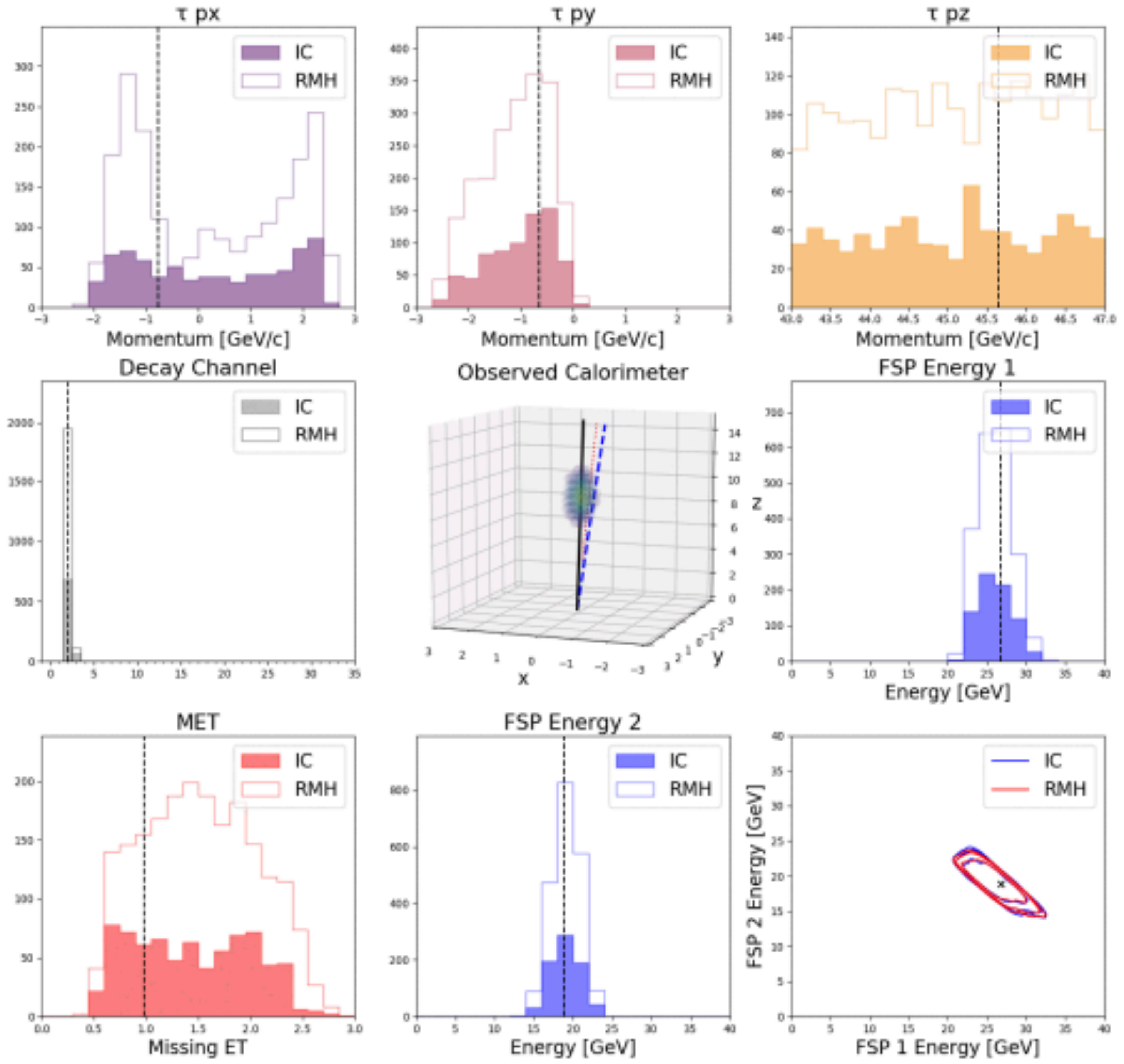
ATLAS detector **observations**

Baydin et al. Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model. NeurIPS 2019

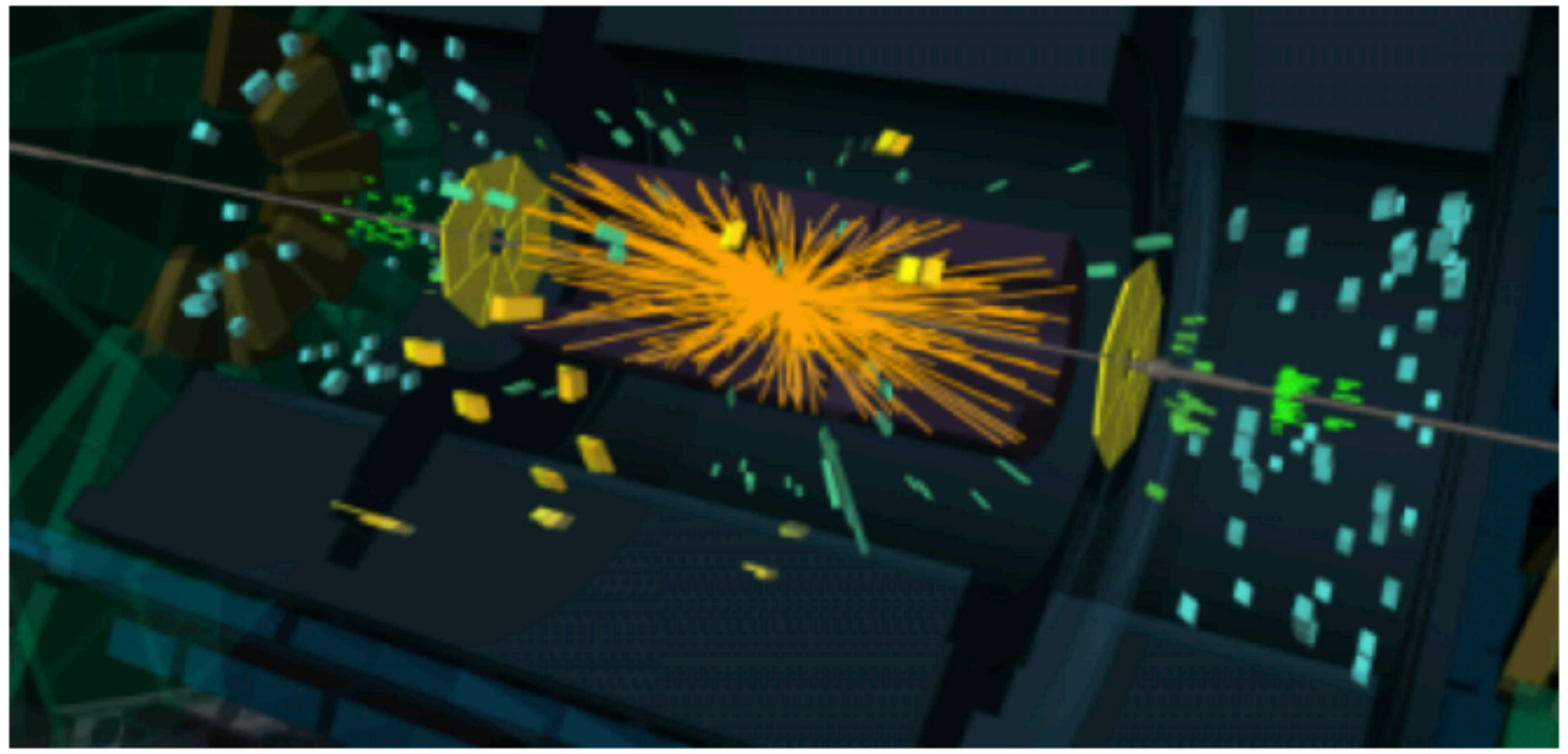
Baydin et al. Etalumis: Bringing probabilistic programming to scientific simulators at scale. In SC '19

$$q_\phi(\mathbf{x}|\mathbf{y})$$

$$\mathbf{y}$$



Particle type and momenta **posterior**



ATLAS detector **observation**

Baydin et al. Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model. NeurIPS 2019
Baydin et al. Etalumis: Bringing probabilistic programming to scientific simulators at scale. In SC '19

What are we interested in?

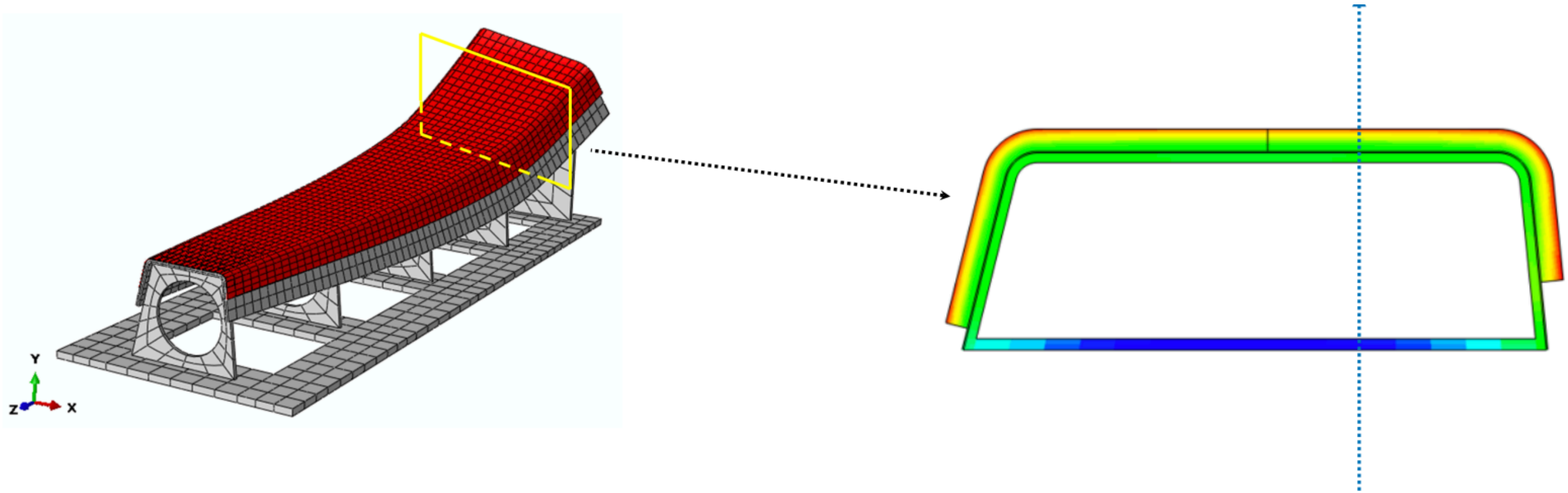
Particle types

Composite part temperature over time

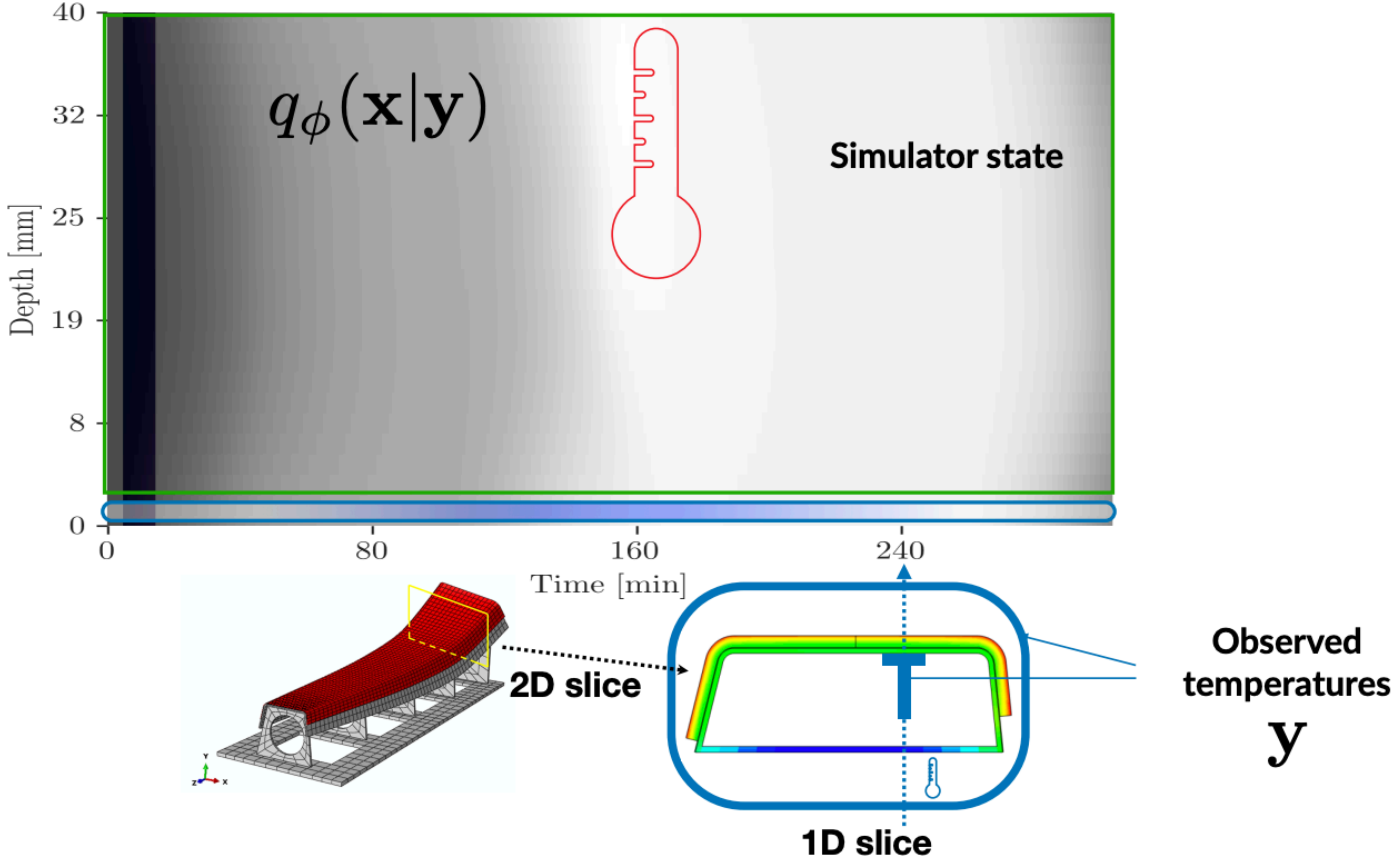
What can we observe?

LHC detector response

Over and surface temperature over time



Munk, Ścibior, Baydin, Stewart, Fernlund, Poursartip, Wood. Deep Probabilistic Surrogate Networks for Universal Simulator Approximation. arXiv 1910.11950



Munk, Ścibior, Baydin, Stewart, Fernlund, Poursartip, Wood. Deep Probabilistic Surrogate Networks for Universal Simulator Approximation. arXiv 1910.11950

What are we interested in?

Particle types

Composite part temperature over time

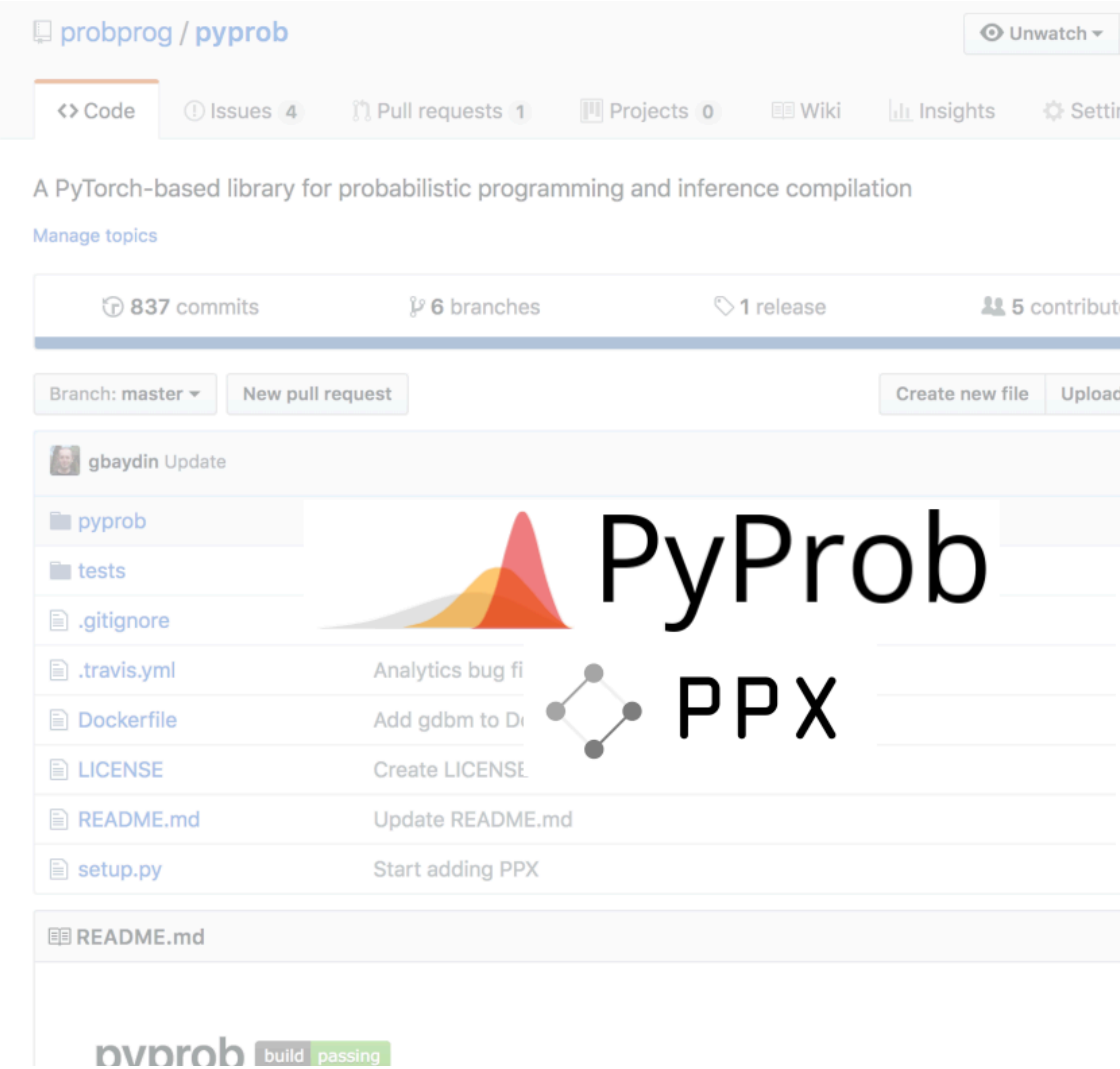
Infection rate of disease

What can we observe?

LHC detector response

Over and surface temperature over time

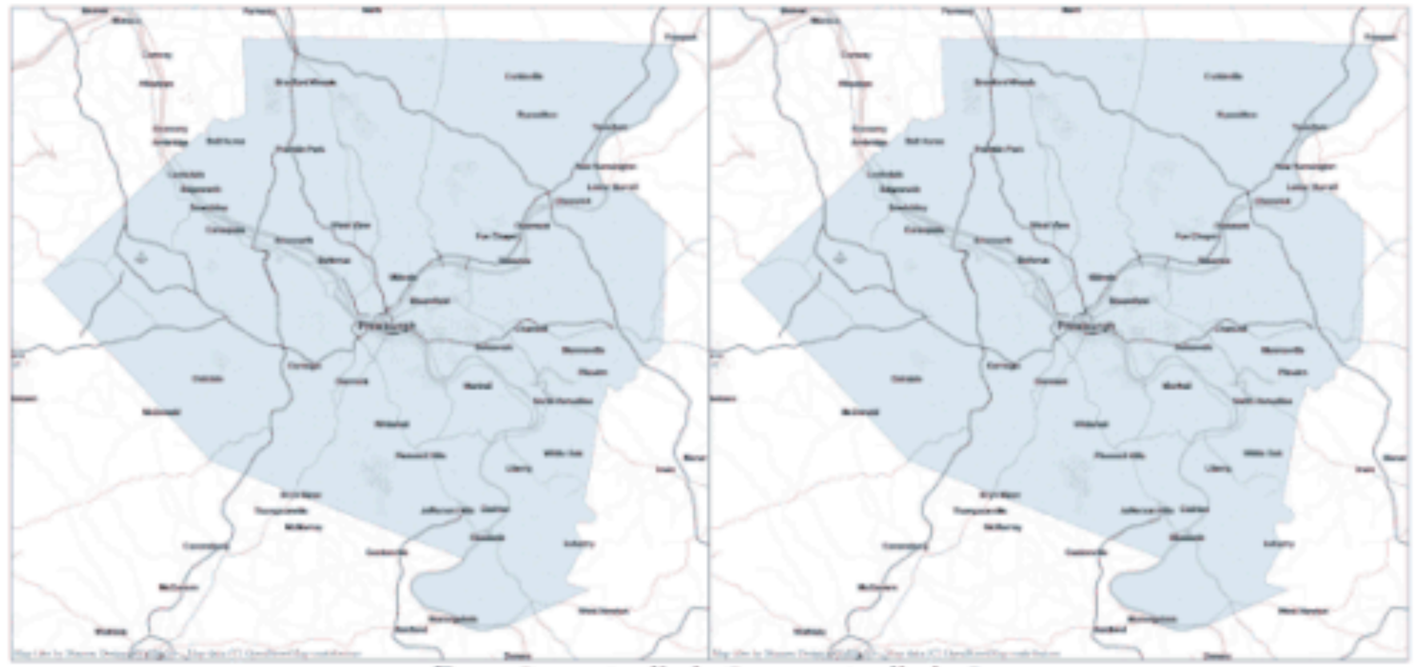
Infections over time



PyProb

PPX

FRED Epidemiological Simulator



https://github.com/plai-group/covid

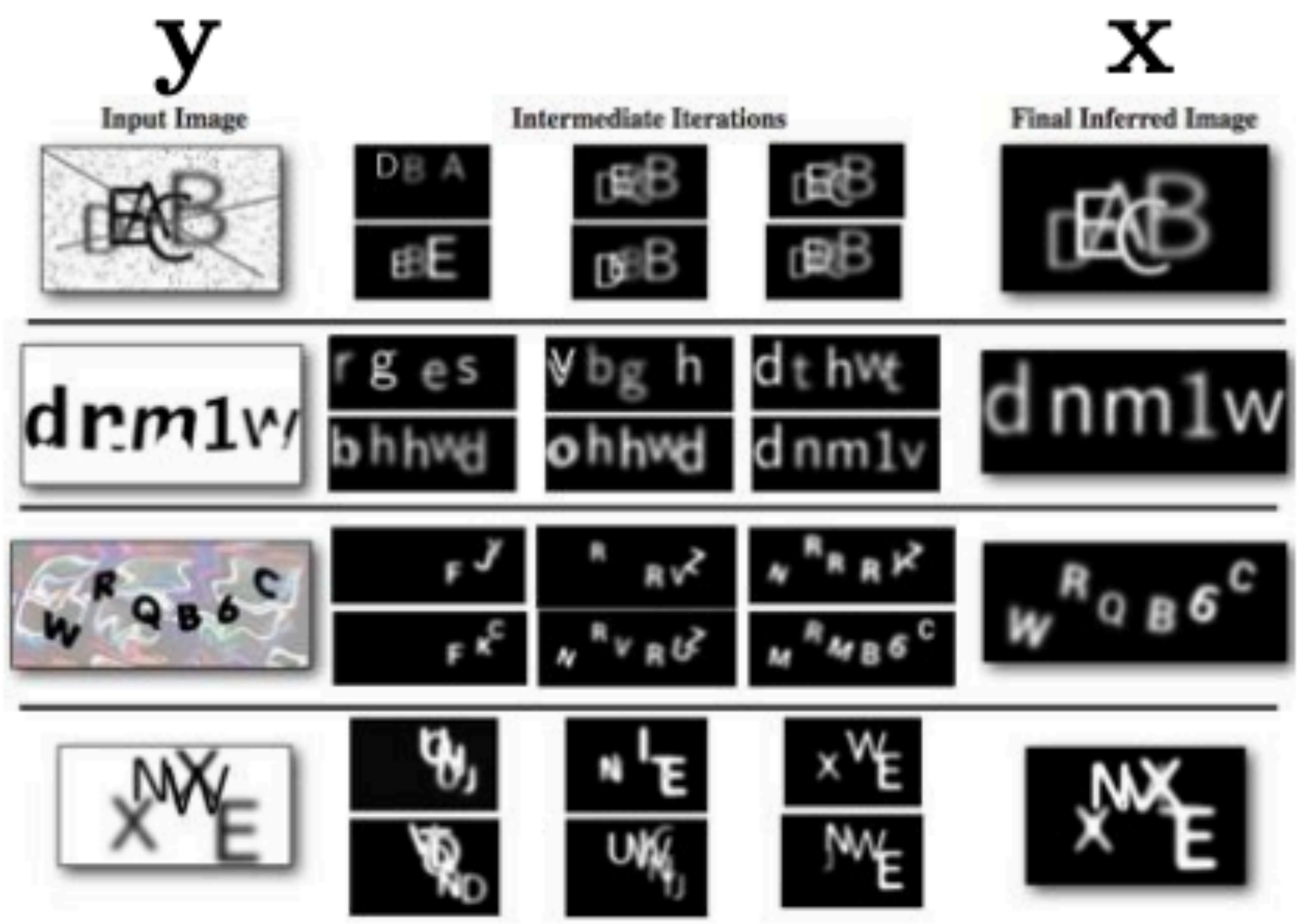
What are we interested in?	**What can we observe?**
Particle types	LHC detector response
Composite part temperature over time	Over and surface temperature over time
Infection rate of disease	Infections over time
A skill of a player	Match outcomes

TrueMatch

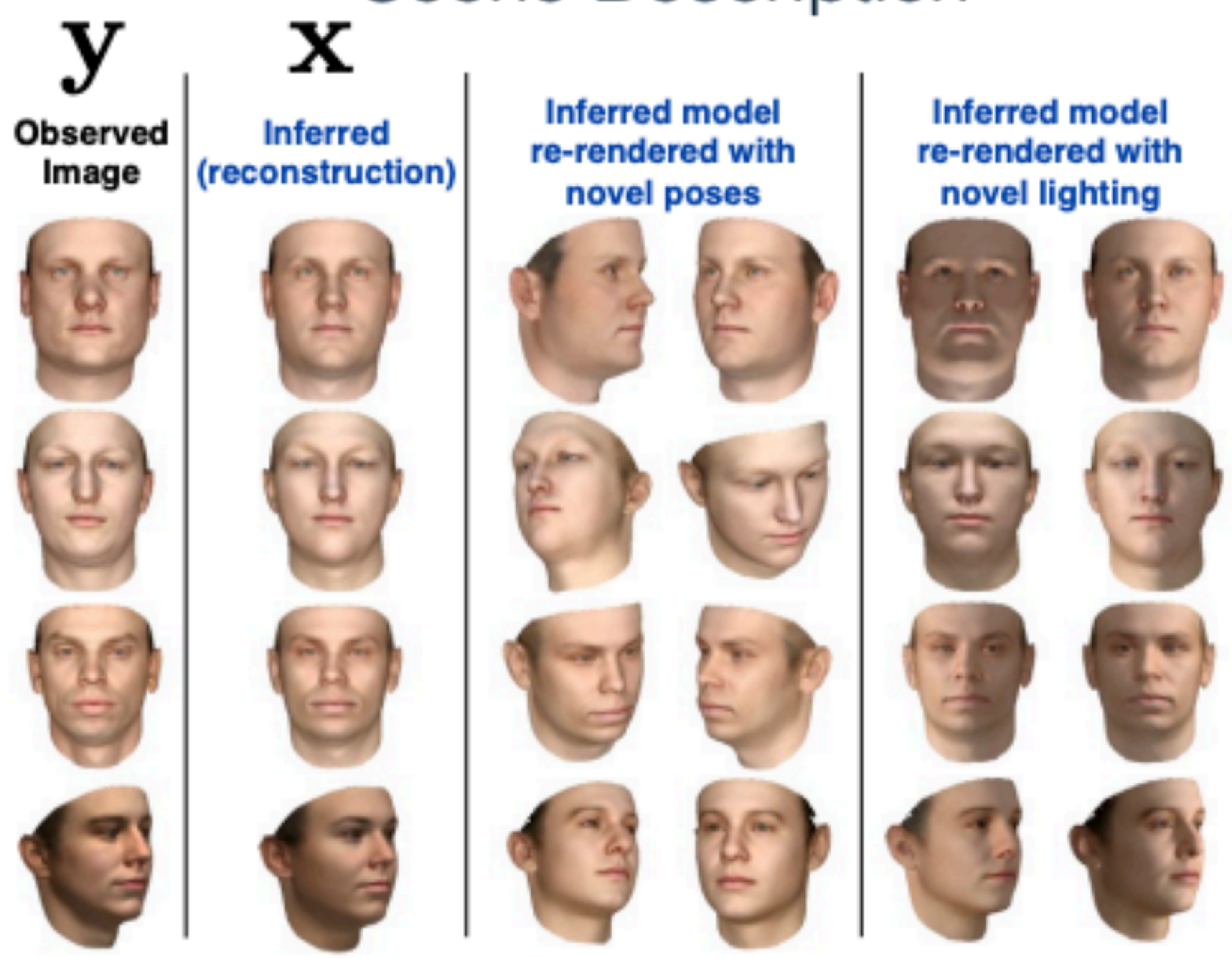
The TrueMatch matchmaking system decides which people should play together in an online multiplayer game. The Coalition have announced that Gears 5 will use TrueMatch.

Perception / Inverse Graphics

Captcha Solving



Scene Description

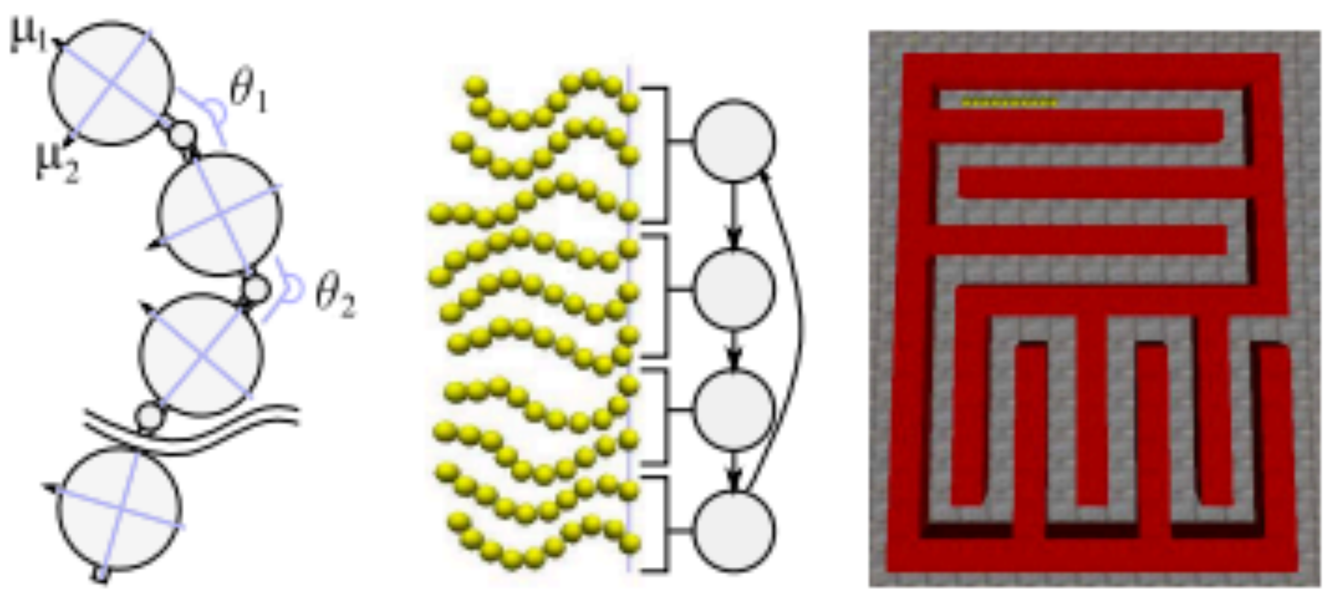


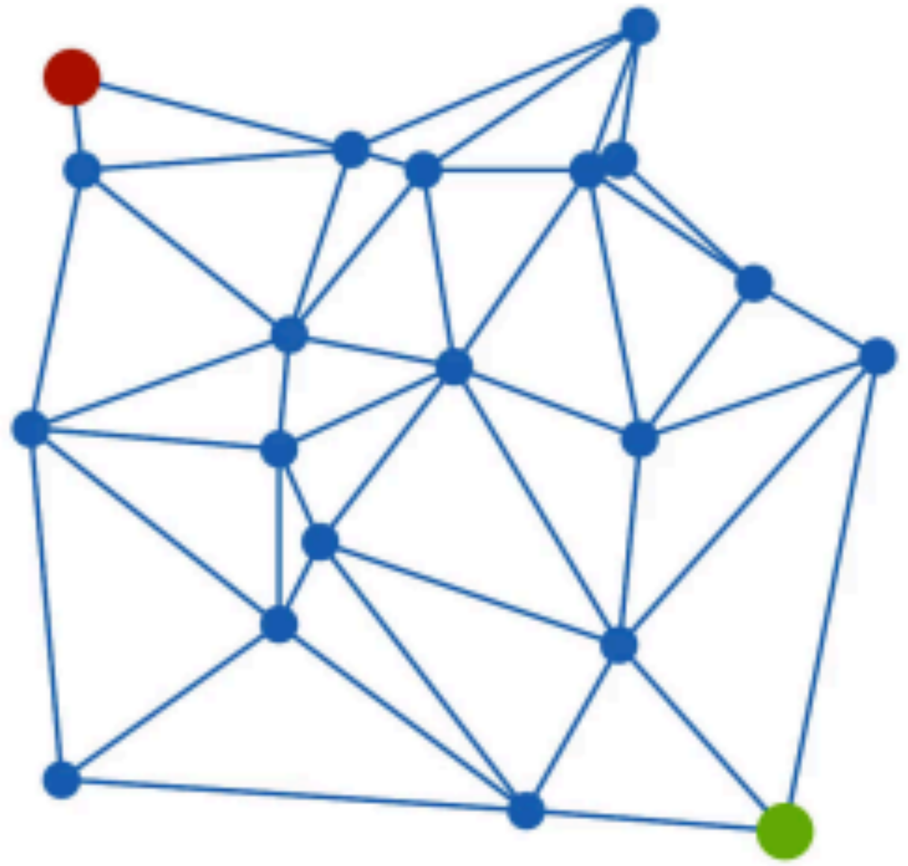
$\mathbf{x}$	$\mathbf{y}$
scene description	image

Mansinghka, Kulkarni, Perov, and Tenenbaum. "Approximate Bayesian image interpretation using generative probabilistic graphics programs." NIPS (2013).

Kulkarni, Kohli, Tenenbaum, Mansinghka "Picture: a probabilistic programming language for scene perception." CVPR (2015).

Reinforcement Learning





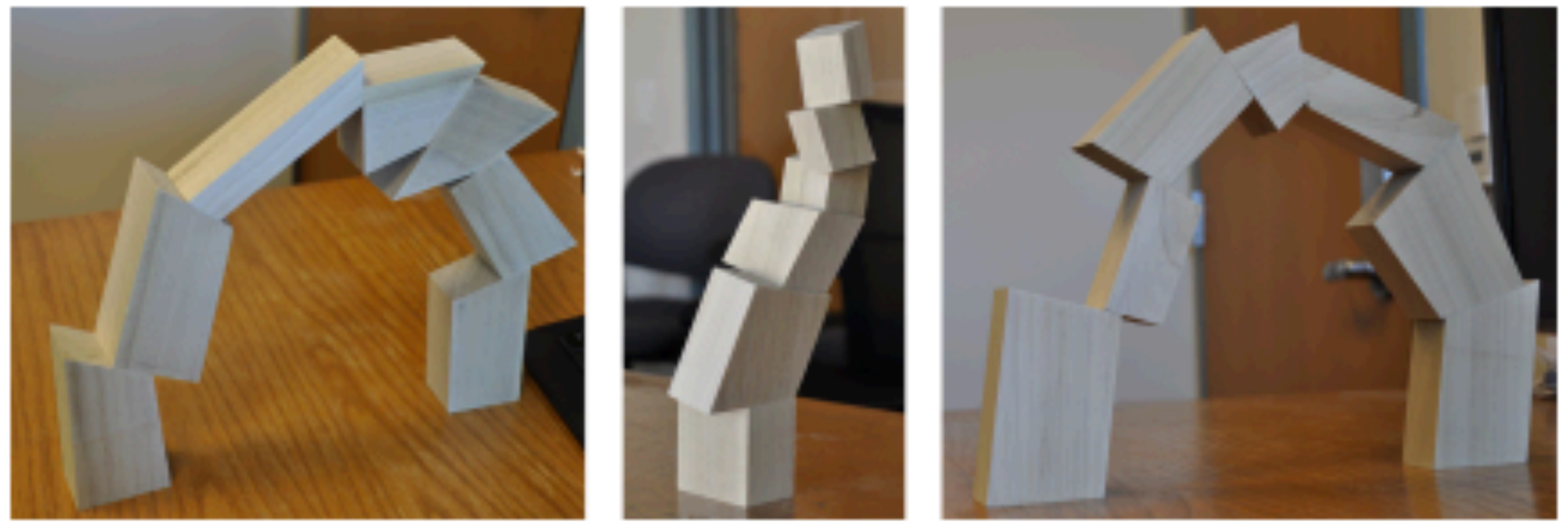
$\mathbf{x}$	$\mathbf{y}$
actions	optimality

Wingate, Goodman, Roy, Kaelbling, and Tenenbaum. "Bayesian policy search with policy priors." (IJCAI), 2011.

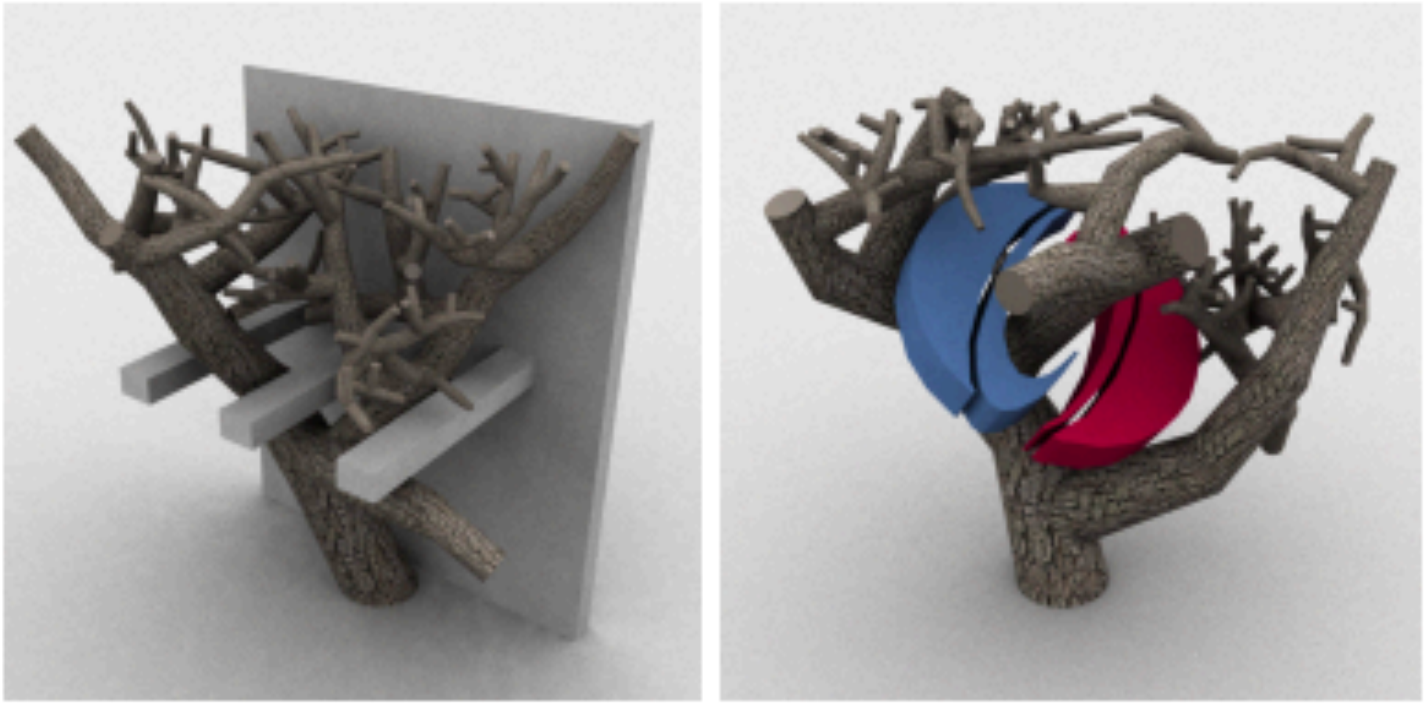
van de Meent, Tolpin, Paige, and Wood. "Black-Box Policy Search with Probabilistic Programs." (AISTATS), 2016.

Directed Procedural Graphics

Stable Static Structures



Procedural Graphics



$\mathbf{x}$	$\mathbf{y}$
simulation	constraint

Ritchie, Lin, Goodman, & Hanrahan.
Generating Design Suggestions under Tight Constraints
with Gradient-based Probabilistic Programming.
In Computer Graphics Forum, (2015)

Ritchie, Mildenhall, Goodman, & Hanrahan.
"Controlling Procedural Modeling Programs with
Stochastically-Ordered Sequential Monte Carlo."
SIGGRAPH (2015)

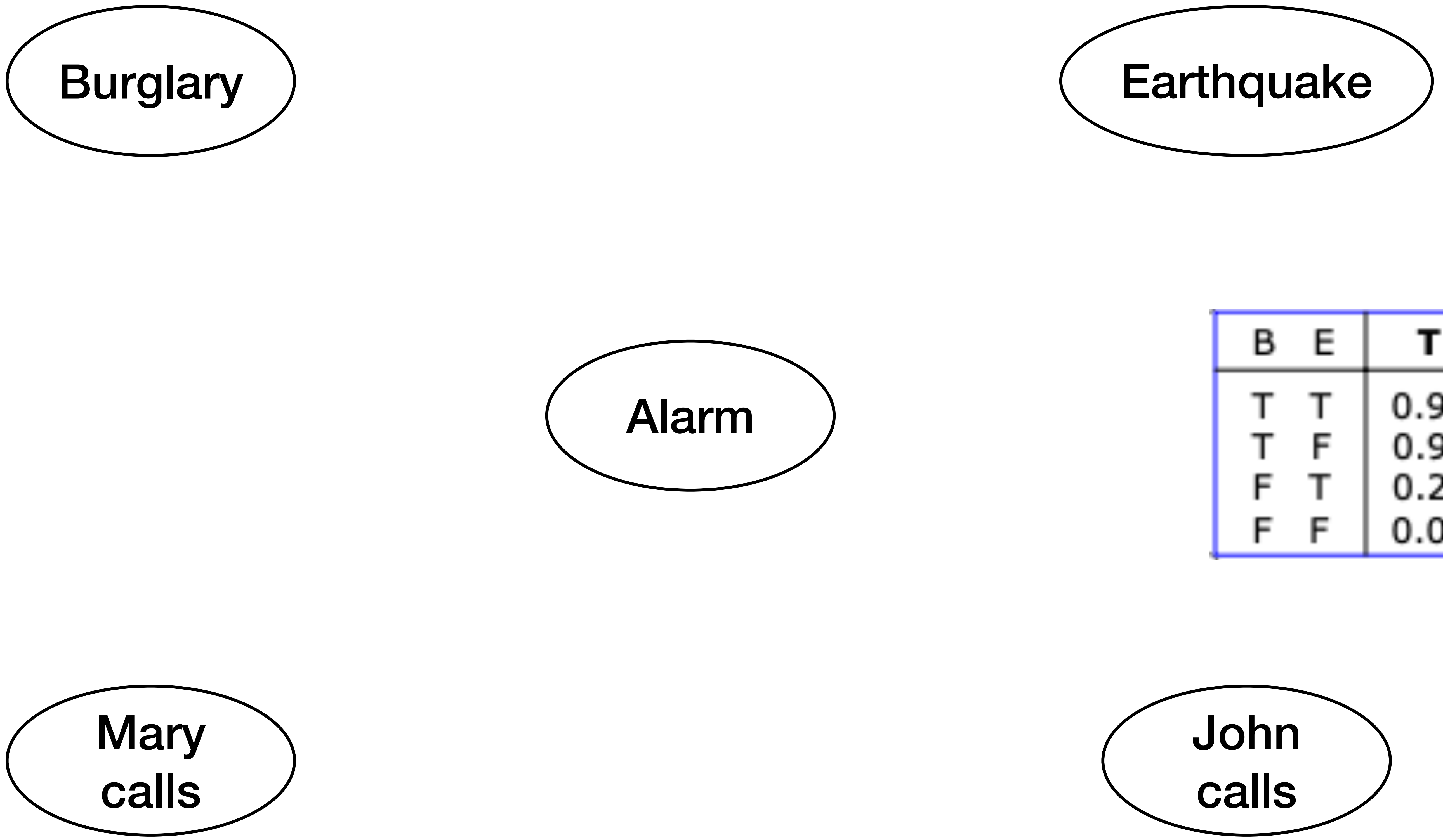
How do we represent probabilistic models?

Your house has an alarm system against burglary.

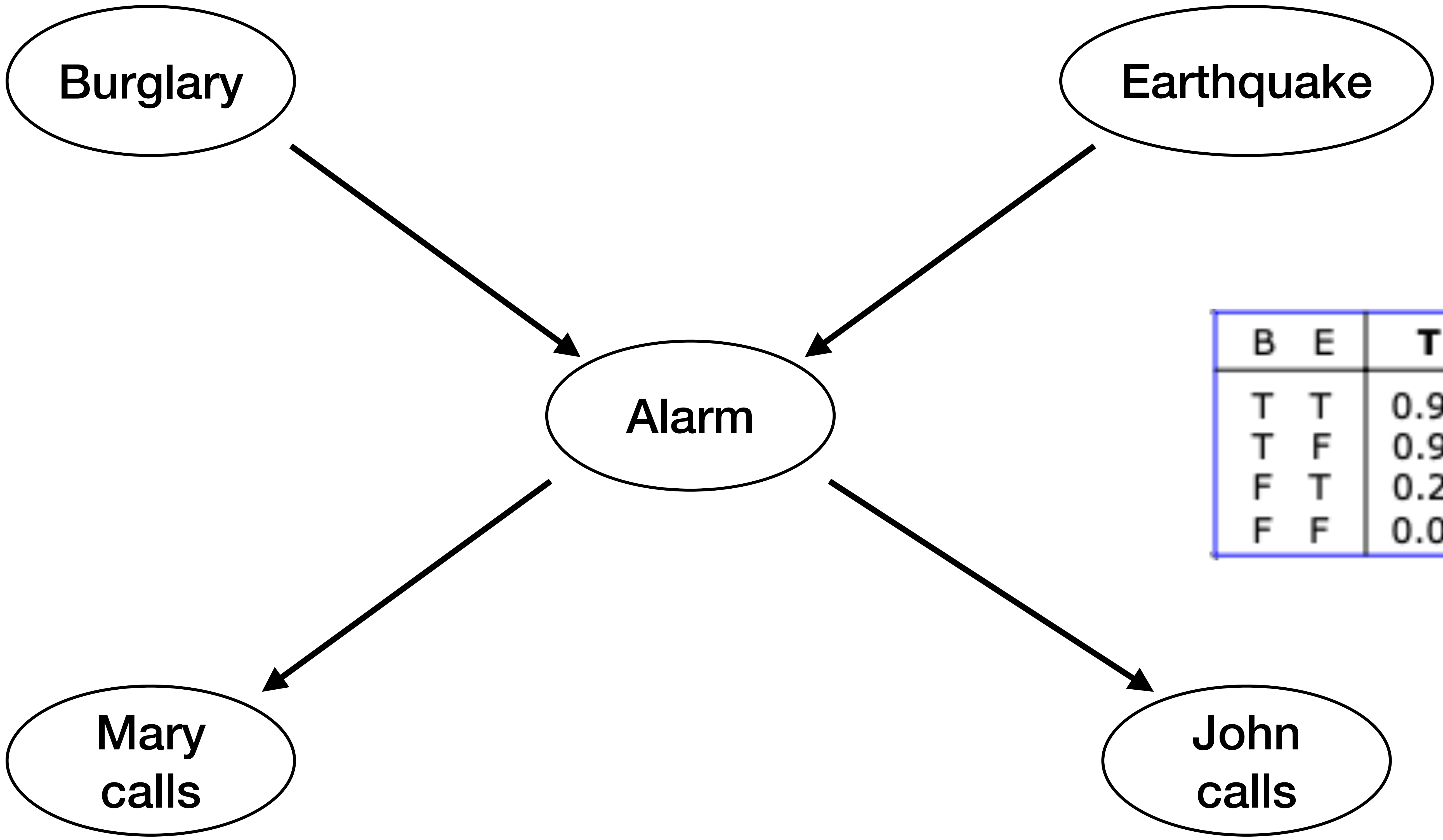
You live in a seismically active area and the alarm system can occasionally be set off by an earthquake.

You have two neighbours, Mary and John, who do not know each other.

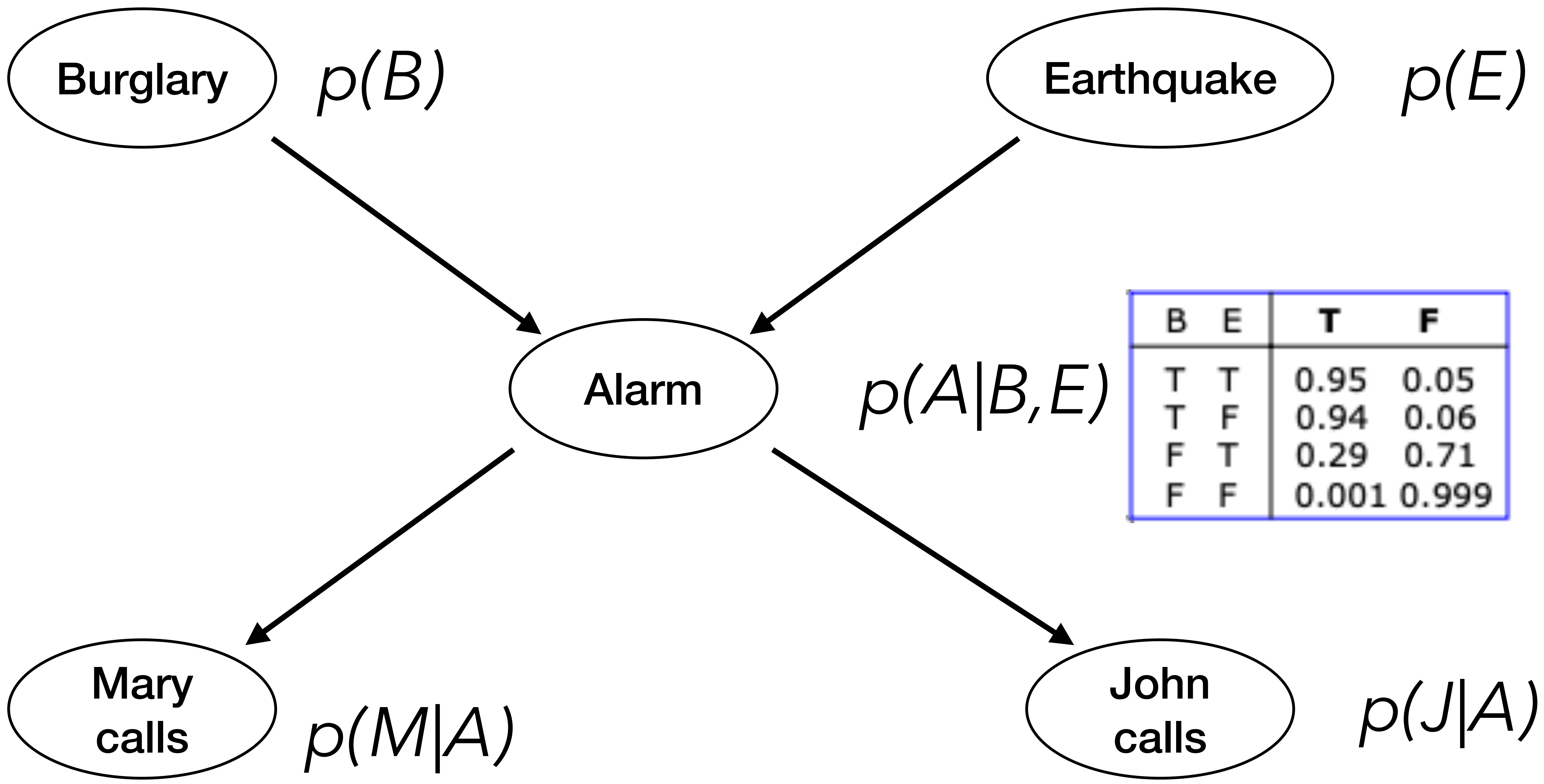
If they hear the alarm they call you, but this is not guaranteed.



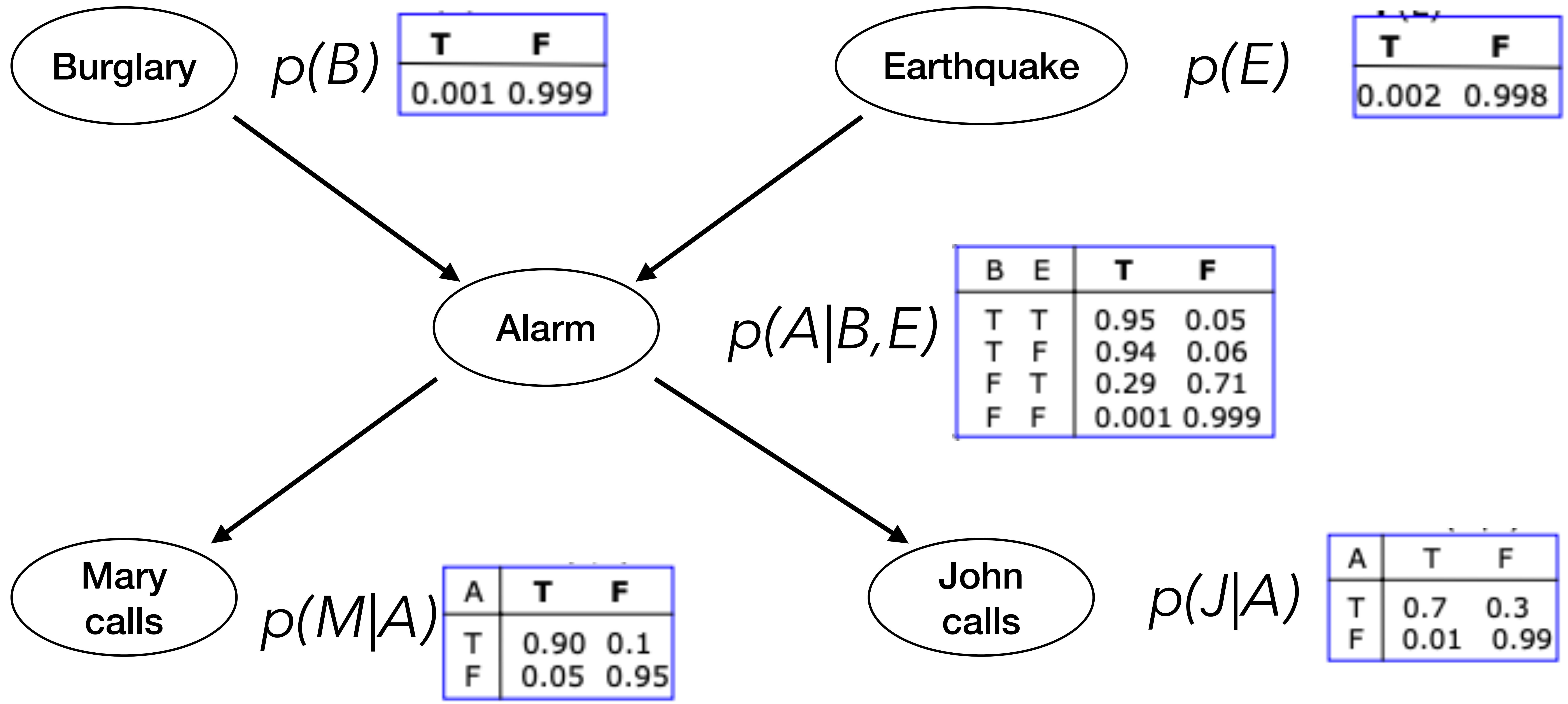
B	E	T	F
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999



B	E	T	F
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999



B	E	T	F
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999



$p(B)$

T	F
0.001	0.999

$p(E)$

T	F
0.002	0.998

$p(A|B,E)$

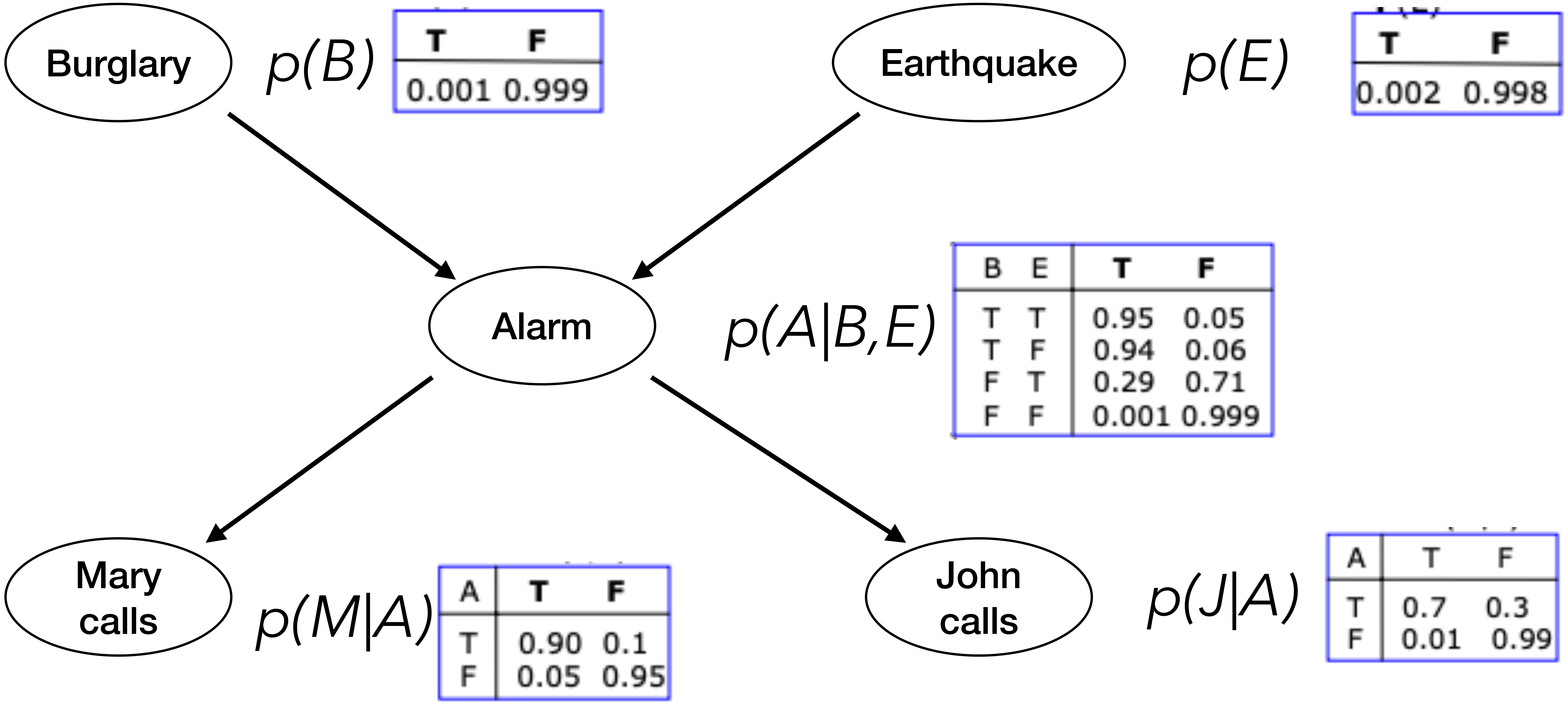
B	E	T	F
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999

$p(M|A)$

A	T	F
T	0.90	0.1
F	0.05	0.95

$p(J|A)$

A	T	F
T	0.7	0.3
F	0.01	0.99



$p(B)$

T	F
0.001	0.999

$p(E)$

T	F
0.002	0.998

$p(A|B,E)$

B	E	T	F
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	0.999

$p(M|A)$

A	T	F
T	0.90	0.1
F	0.05	0.95

$p(J|A)$

A	T	F
T	0.7	0.3
F	0.01	0.99

Bayesian networks perspective

$$p(B,E,A,M,J) = p(B)\, p(E)\, p(A|B,E)\, p(M|A)\, p(J|A)$$

What is difficult to do with Bayesian Networks?

What is difficult to do with Bayesian Networks?

Continuous values?

Changing number of variables?

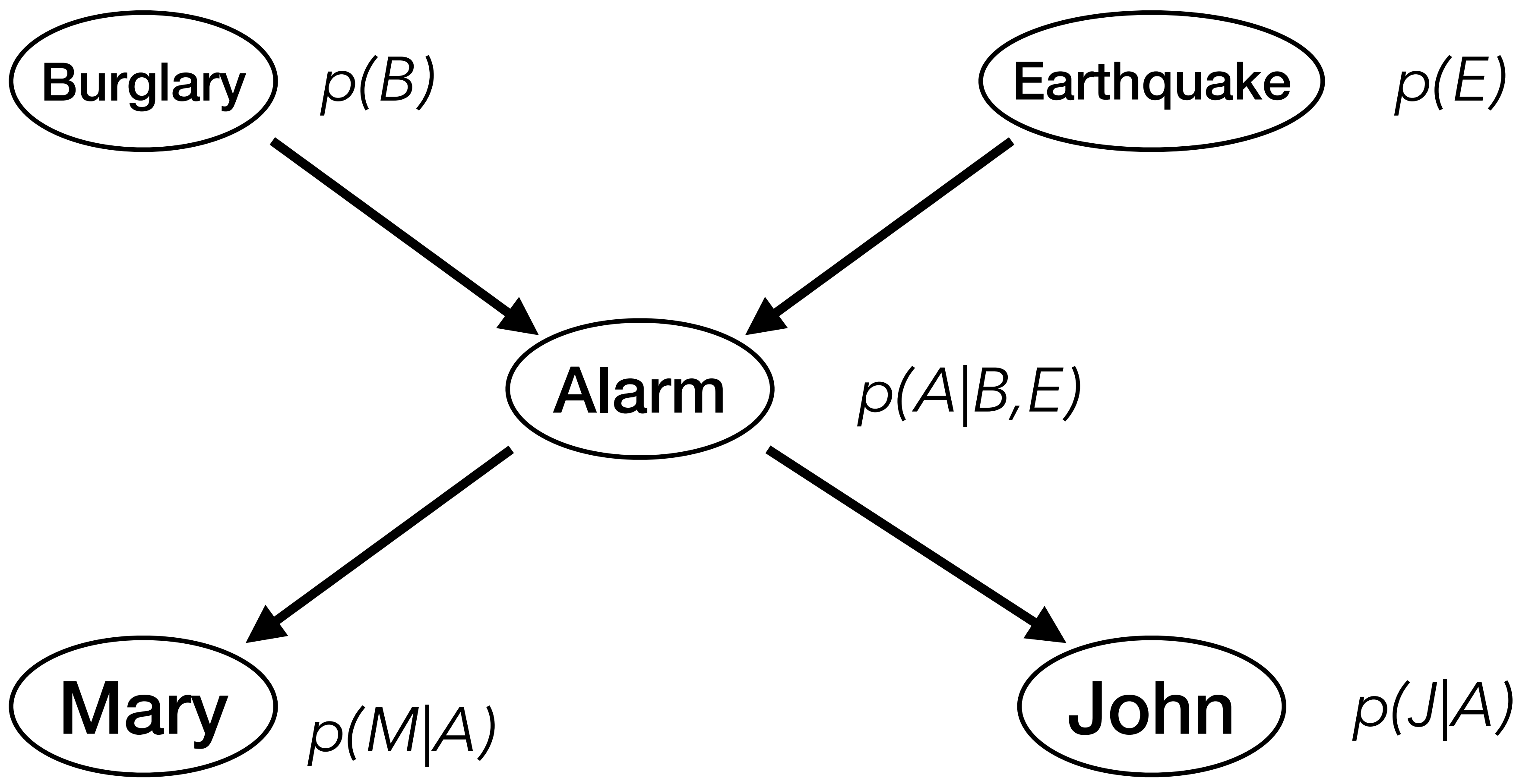
Lots of variables?

From Bayesian networks to probabilistic programs

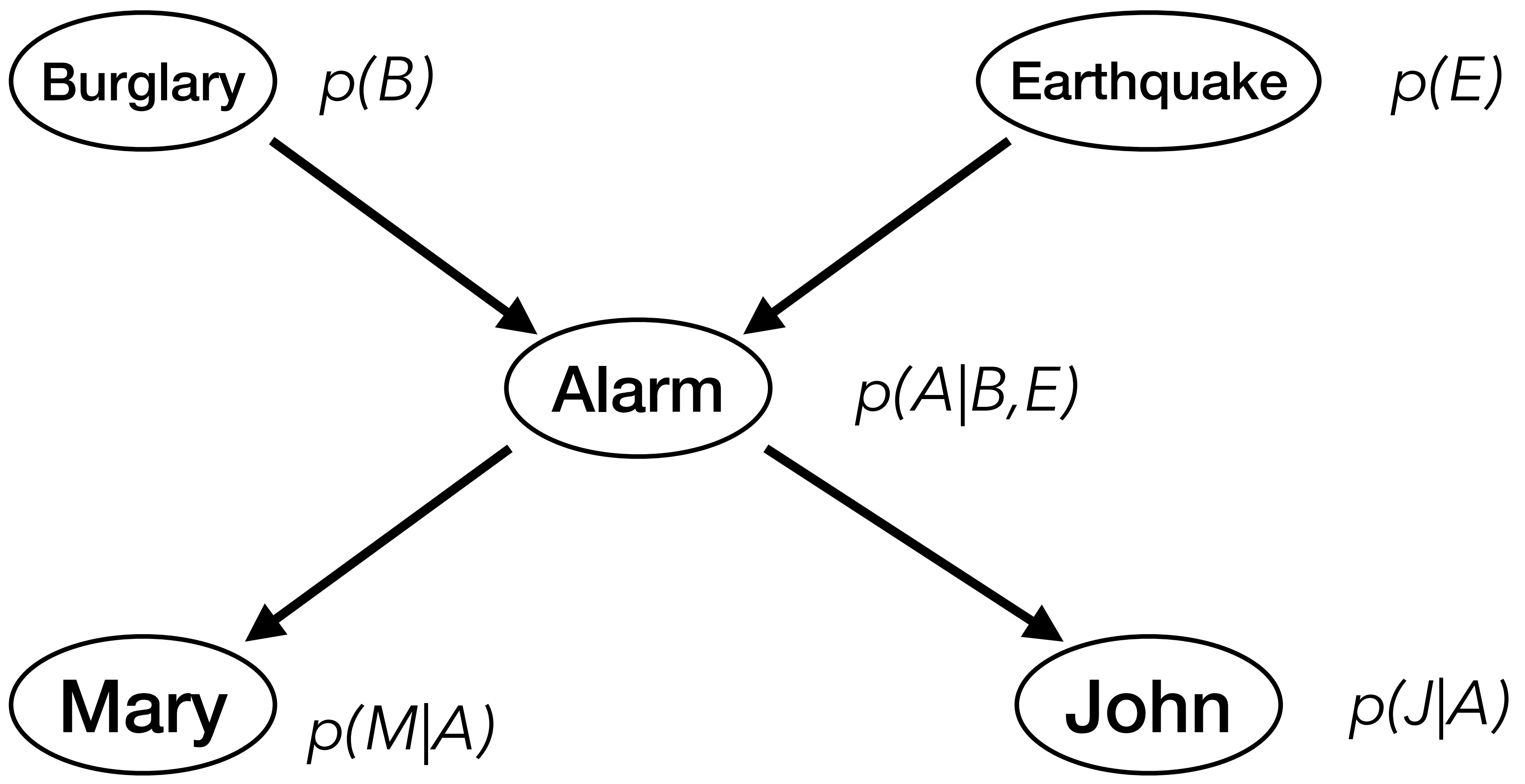
PPLs are programming languages with two special constructs

sample(Ξ)	sample a value from distribution Ξ
observe(Γ, y)	condition the variable y to have value from distribution Γ

From Bayesian networks to probabilistic programs



From Bayesian networks to probabilistic programs



Events become variables

Var burglary =

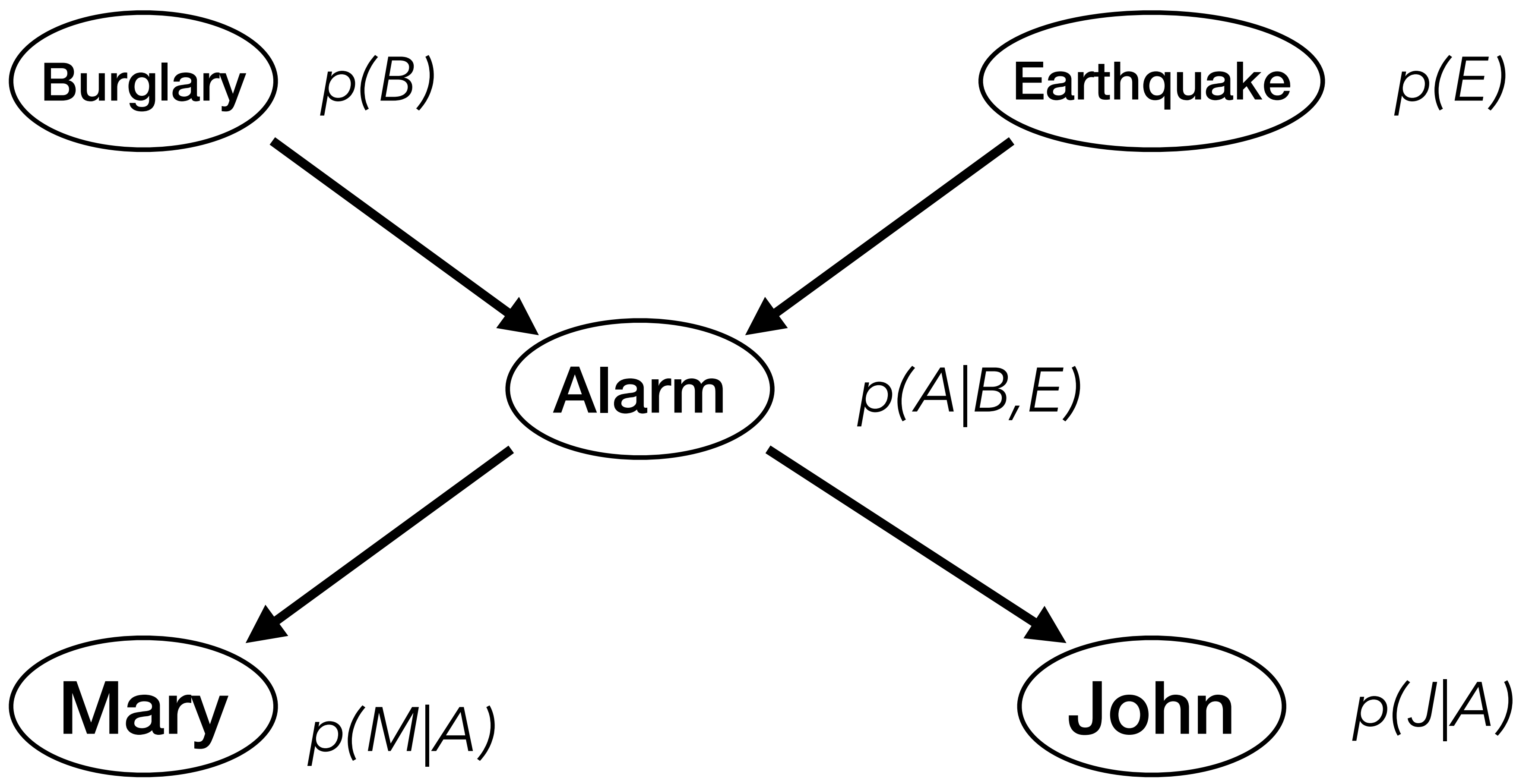
Var earthquake =

Var alarm =

Var maryCalls =

Var johnCalls =

From Bayesian networks to probabilistic programs



Events become variables from which we sample

```
Var burglary = sample(Bernoulli(0.001))

Var earthquake = sample(Bernoulli(0.001))

Var alarm = if burglary & earthquake
                  sample(Bernoulli(0.95))
            elif ....

Var maryCalls = sample(Bernoulli(0.99)) if alarm else …

Var johnCalls = sample(Bernoulli(0.8)) if alarm else …
```

From Bayesian networks to probabilistic programs

```
Def my_first_probabilistic_program():
        Var burglary = sample(Bernoulli(0.001))
        Var earthquake = sample(Bernoulli(0.001))
        Var alarm = if burglary & earthquake
                        sample(Bernoulli(0.95))
                    elif ....
        Var maryCalls = sample(Bernoulli(0.99)) if alarm else …
        Var johnCalls = sample(Bernoulli(0.8)) if alarm else …
```

Prior

From Bayesian networks to probabilistic programs

```
Def my_first_probabilistic_program():
        Var burglary = sample(Bernoulli(0.001))
        Var earthquake = sample(Bernoulli(0.001))
        Var alarm = if burglary & earthquake
                        sample(Bernoulli(0.95))
                    elif ....
        Var maryCalls = sample(Bernoulli(0.99)) if alarm else …
        Var johnCalls = sample(Bernoulli(0.8)) if alarm else …

        return johnCalls
```

Prior

Posterior

From Bayesian networks to probabilistic programs

```
Def my_first_probabilistic_program():
        Var burglary = sample(Bernoulli(0.001))
        Var earthquake = sample(Bernoulli(0.001))
        Var alarm = if burglary & earthquake
                            sample(Bernoulli(0.95))
                    elif ....
        Var maryCalls = sample(Bernoulli(0.99)) if alarm else …
        Var johnCalls = sample(Bernoulli(0.8)) if alarm else …

        observe(Bernoulli(1), alarm)

        return johnCalls
```

Prior

Likelihood

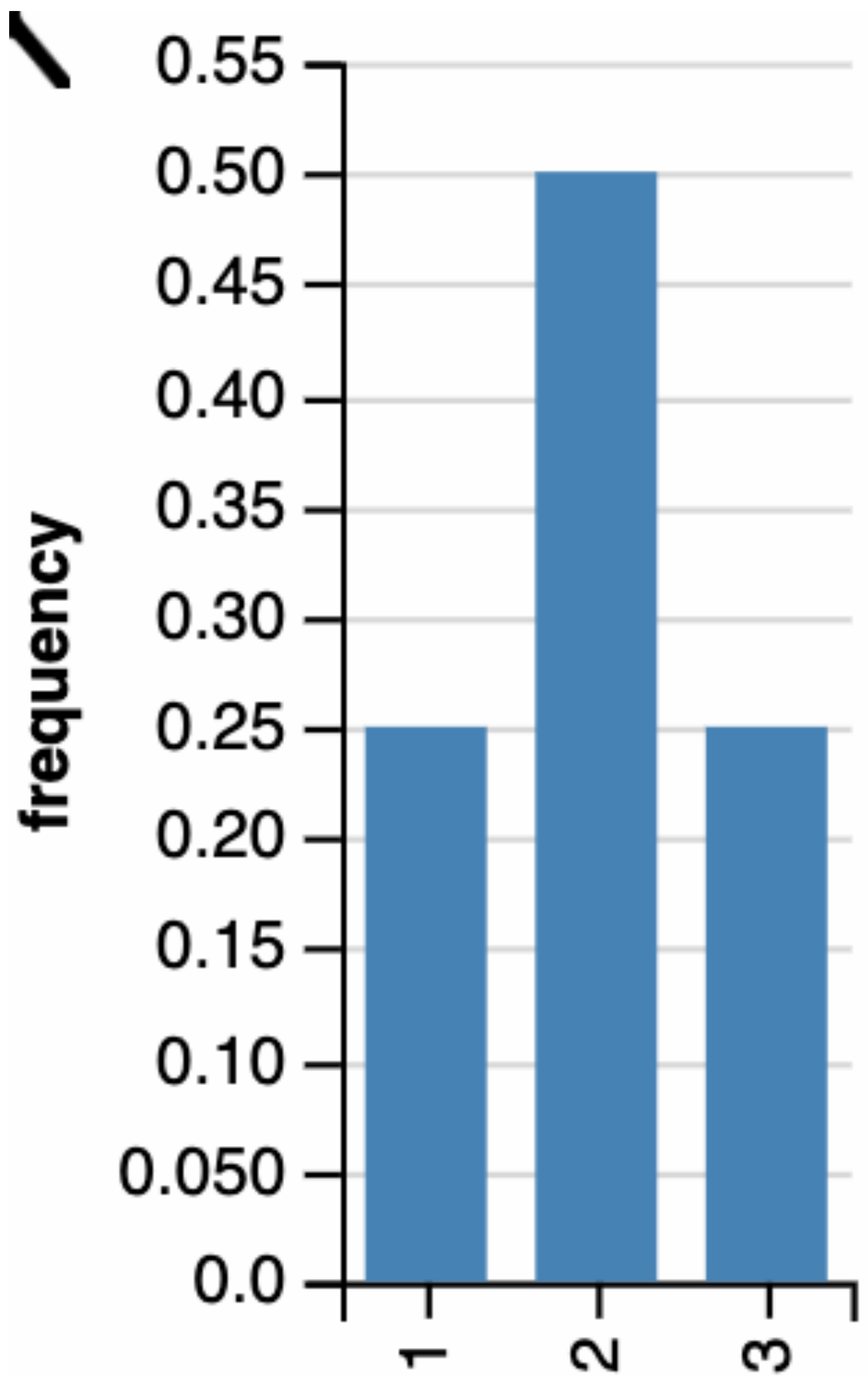
Posterior

PPLs look like 'normal' programs but return distributions

```
Def my_first_probabilistic_program():
        Var burglary = sample(Bernoulli(0.001))

        Var earthquake = sample(Bernoulli(0.001))

        Var alarm = if burglary & earthquake
                        sample(Bernoulli(0.95))
                    elif ....

        Var maryCalls = sample(Bernoulli(0.99)) if alarm else …

        Var johnCalls = sample(Bernoulli(0.8)) if alarm else …

        observe(Bernoulli(1), alarm)

        return johnCalls
```

PPLs look like ‘normal’ programs but return distributions

```
Def my_first_probabilistic_program():
        Var burglary = sample(Bernoulli(0.001))
        Var earthquake = sample(Bernoulli(0.001))
        Var alarm = if burglary & earthquake
                        sample(Bernoulli(0.95))
                    elif ….
        Var maryCalls = sample(Bernoulli(0.99)) if alarm else …
        Var johnCalls = sample(Bernoulli(0.8)) if alarm else …

        observe(Bernoulli(1), alarm)

        return johnCalls
```



Why programs?

The most expressive representation we have at the moment

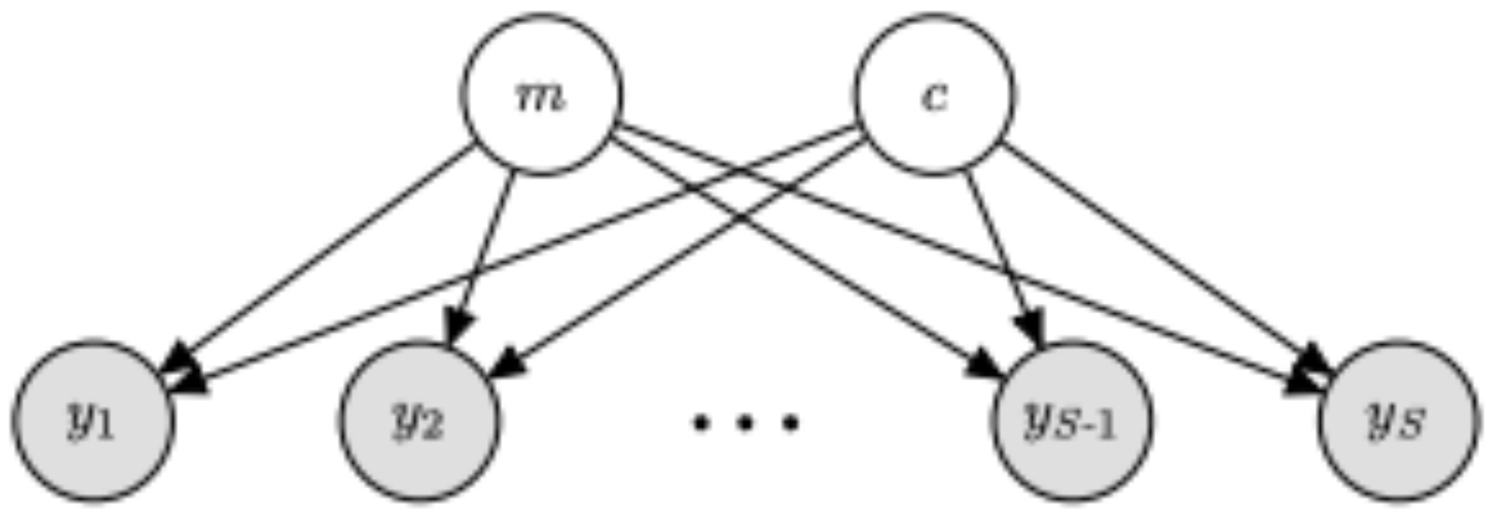
Can represent any computable process

Therefore, we can probabilistically reason about anything computable

Interpretable!

Inputs: Student-t degrees of freedom ν, error scale σ, data $y_{1:S} = \{u_s, v_s\}_{s=1}^{S}$

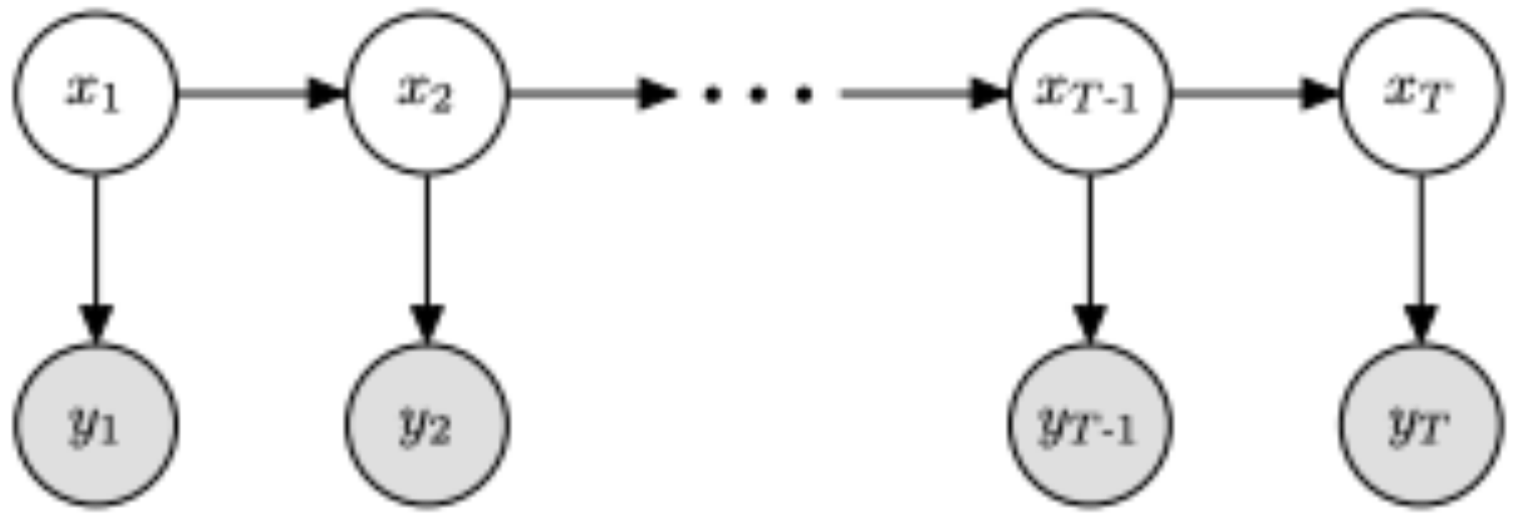
$m \leftarrow$ **sample** (**normal** (0,1))
$c \leftarrow$ **sample** (**normal** (0,1))
`obs-dist` $\leftarrow$ **student-t** (ν)
for $s = 1, \ldots, S$ **do**
  $d \leftarrow (v_s - m u_s - c)/\sigma$
  observe (`obs-dist`, d)
end for
return $m,\ c$



$$p(m, c, y_{1:S}|\nu, \sigma) = \mathcal{N}(m; 0, 1)\,\mathcal{N}(c; 0, 1)\prod_{s=1}^{S} \text{STUDENT-T}\left(\frac{v_s - m u_s - c}{\sigma}; \nu\right)$$

Inputs: Transition std-dev σ, output shape α, output rate β, data $y_{1:T}$

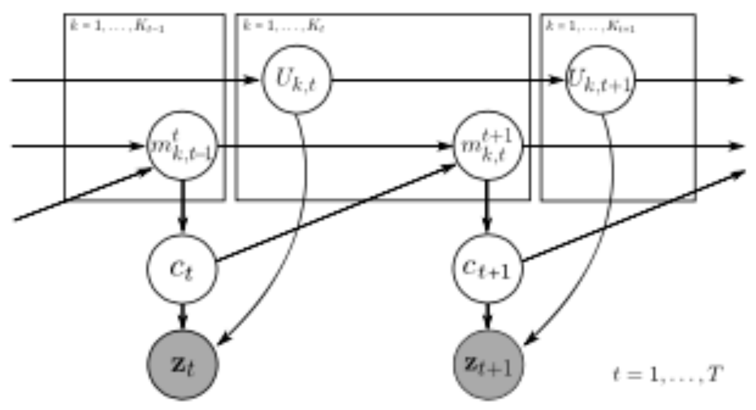
$x_0 \leftarrow 0$
`tr-dist` $\leftarrow$ **normal** $(0, \sigma)$
`obs-dist` $\leftarrow$ **gamma** (α, β)
for $t = 1, \ldots, T$ **do**
  $x_t \leftarrow x_{t-1}+$**sample** (`tr-dist`)
  observe (`obs-dist`, $y_t - x_t$)
  $z_t \leftarrow \mathbb{I}(x_t > 4)$
end for
return $z_{1:T}$

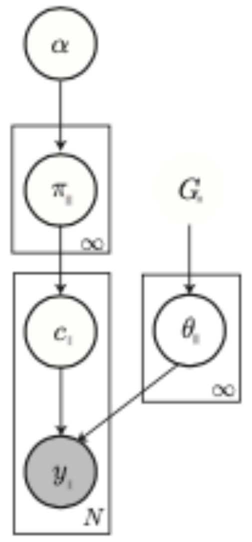


$$p(x_{1:T}, y_{1:T}|\sigma, \alpha, \beta) = \mathcal{N}(x_1; 0, \sigma^2)\ \text{GAMMA}(y_1 - x_1; \alpha, \beta)\prod_{t=2}^{T}\mathcal{N}(x_t - x_{t-1}; 0, \sigma^2)\ \text{GAMMA}(y_t - x_t; \alpha, \beta)$$

Key elements of probabilistic programs

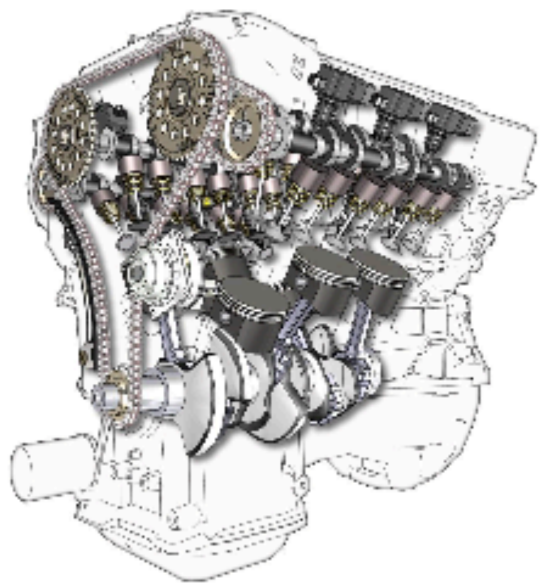
Models





$$p(\mathbf{x}, \mathbf{y})$$

Programming Language Abstraction Layer

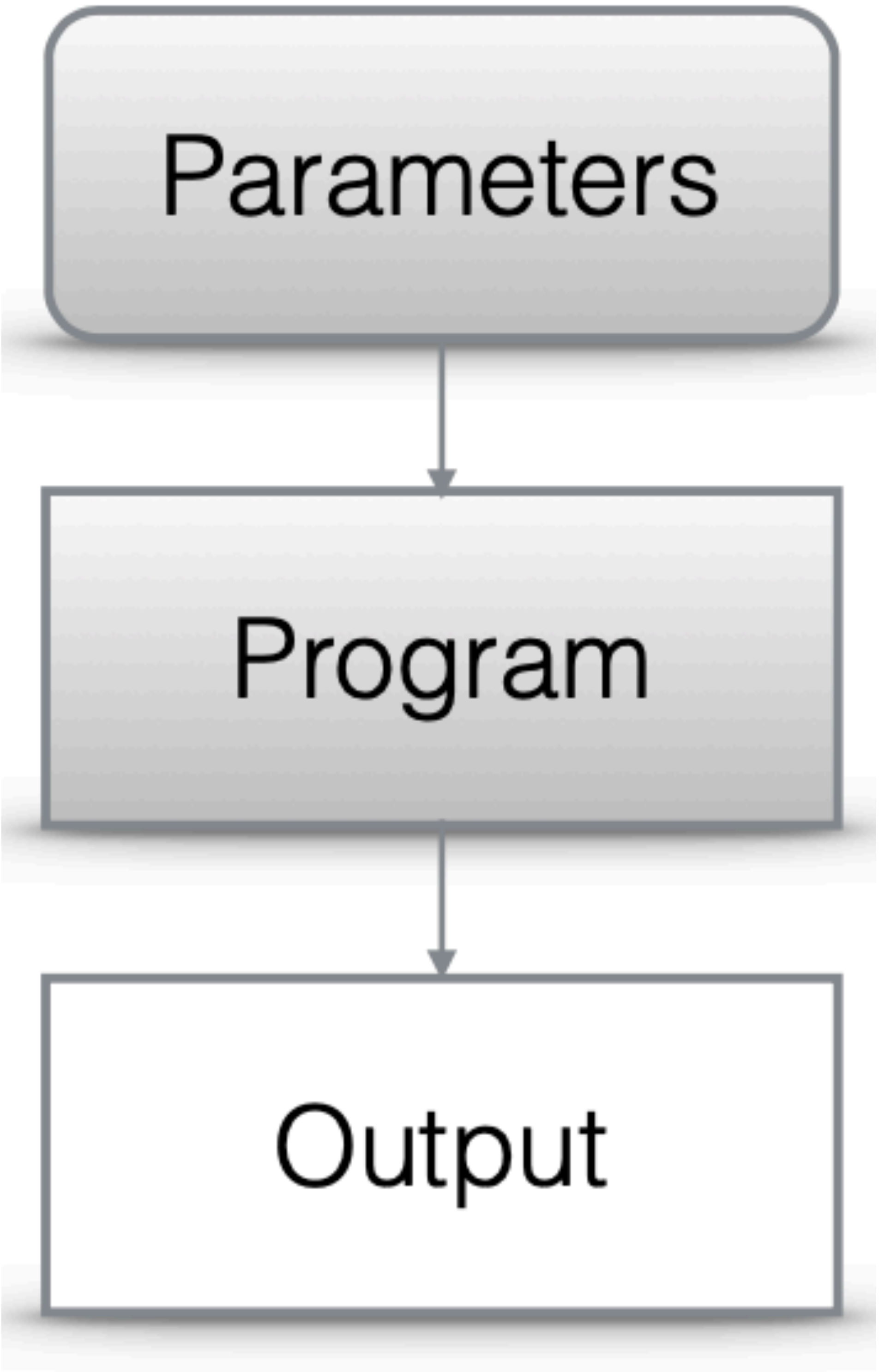


$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})}$$

Evaluators that automate Bayesian *inference*

Intuition

Intuition

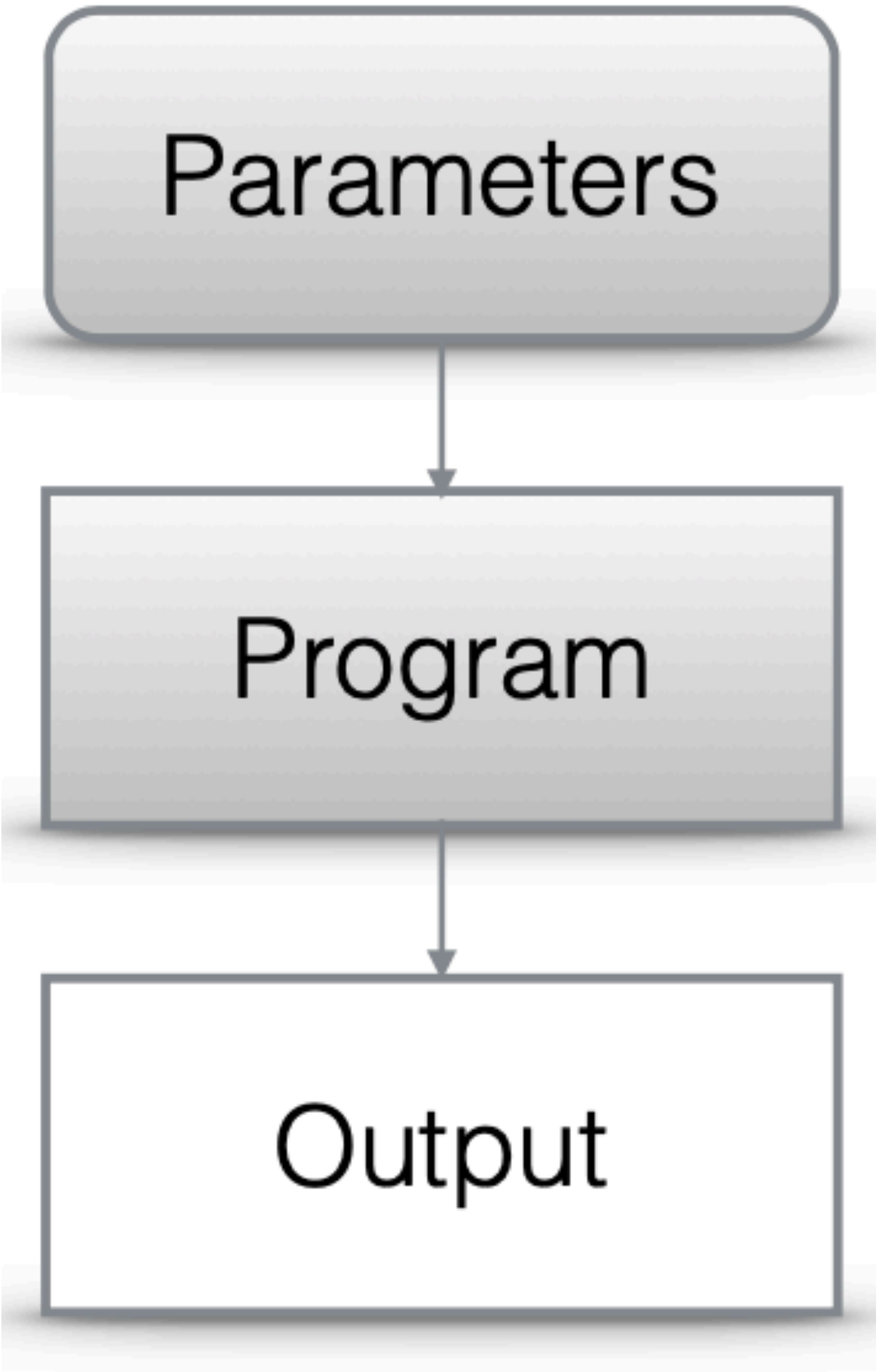


CS

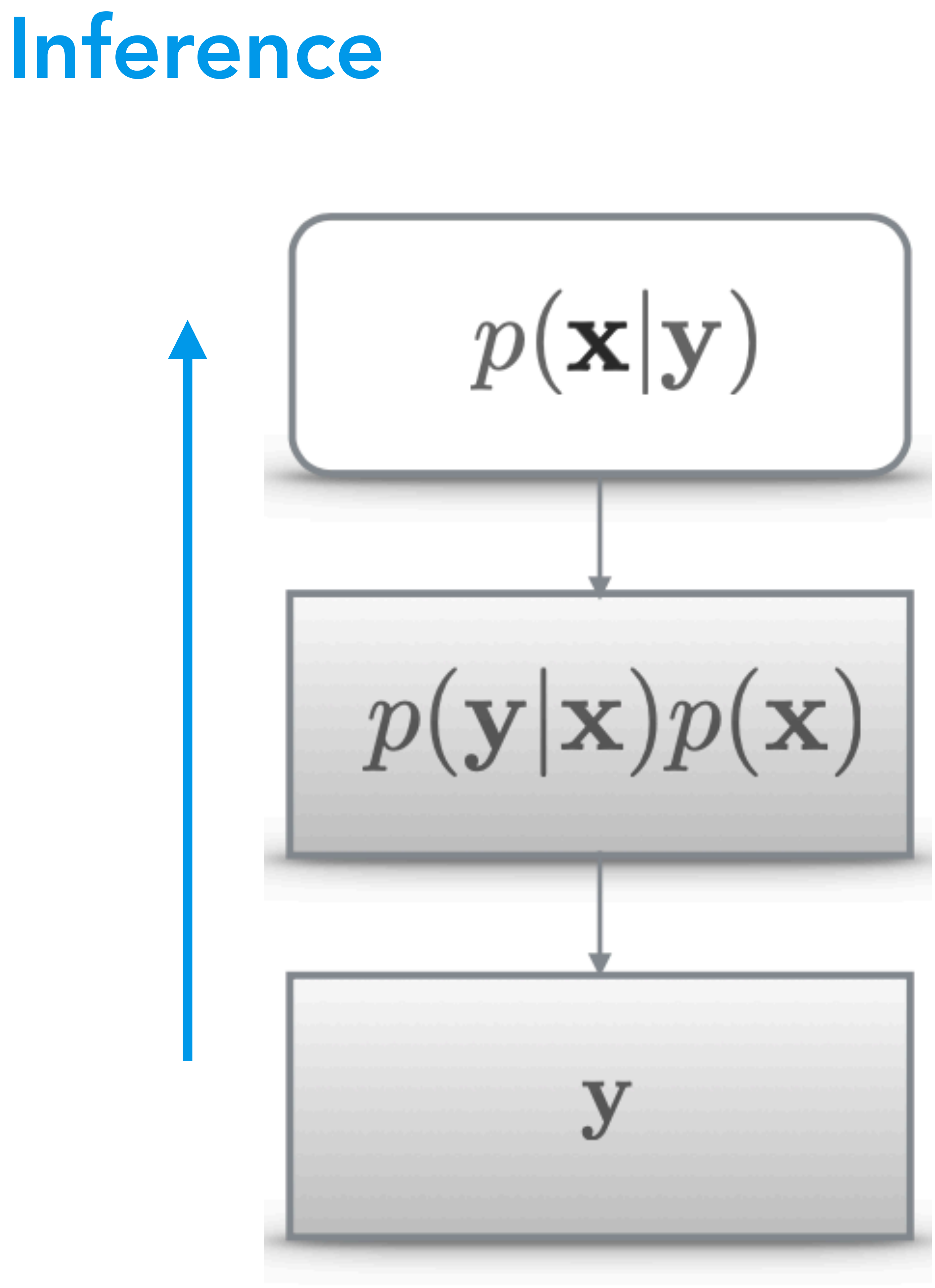
Intuition


Parameters
Program
Output
CS
$p(\mathbf{x}|\mathbf{y})$
$p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$
$\mathbf{y}$
Statistics

Intuition

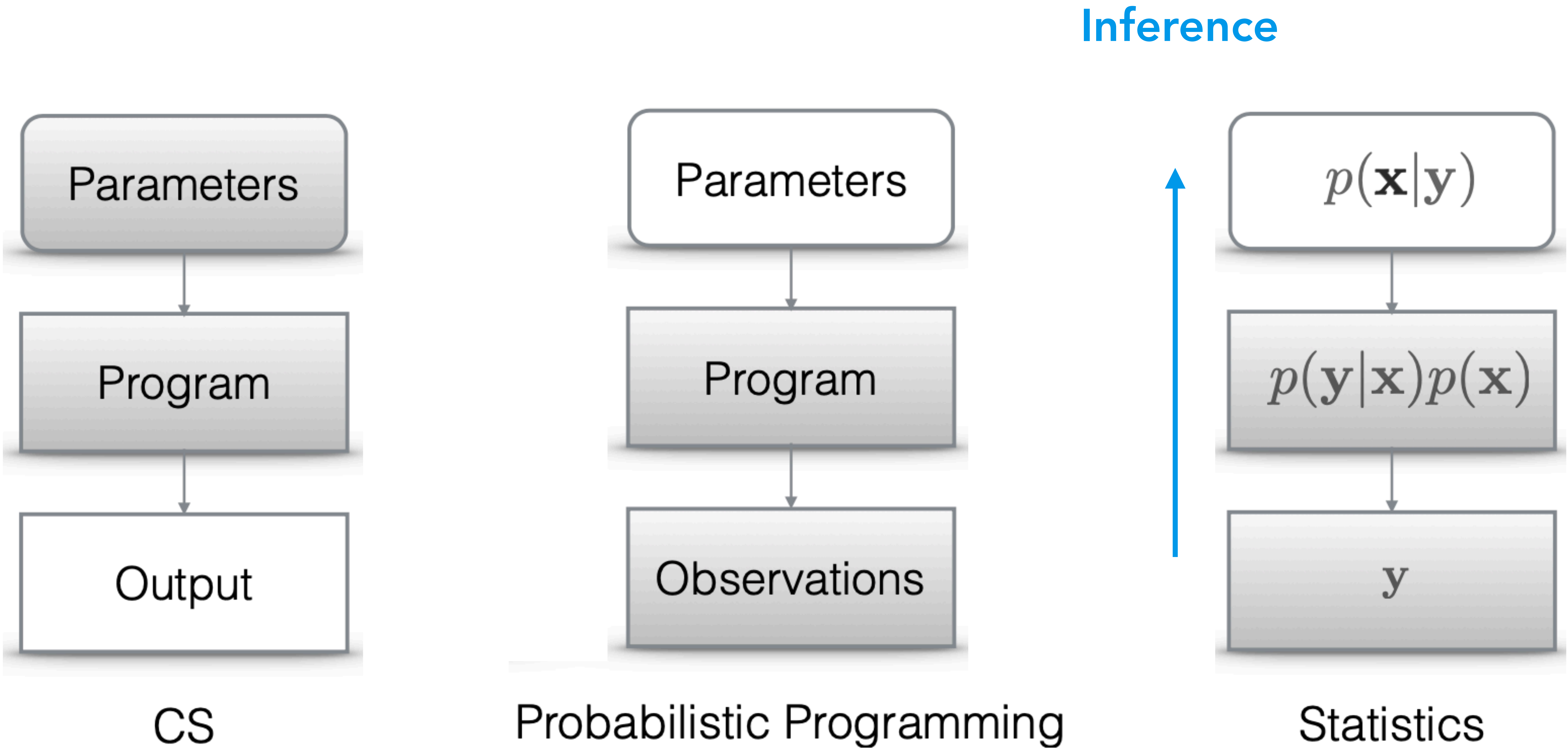


CS



Statistics

Intuition


Inference
Parameters
Program
Output
CS
Parameters
Program
Observations
Probabilistic Programming
$p(\mathbf{x}|\mathbf{y})$
$p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$
$\mathbf{y}$
Statistics

In your first-year probability course

$$P(A) = \frac{\text{events in which A happens}}{\text{All possible events}}$$

$$P(A \mid B) = \frac{\text{events in which A and B happen}}{\text{events in which B happens}}$$

In your first-year probability course

$$P(A) = \frac{\text{events in which A happens}}{\text{All possible events}}$$

$$P(A \mid B) = \frac{\text{events in which A and B happen}}{\text{events in which B happens}}$$

What is the probability that a probabilistic program defines?

A probabilistic program defines a distribution over traces

Trace: values returned by sample statements in one execution

$$p(x, y) = \prod_{t=1}^{T} f_{a_t}(x_t \mid x_{1:t-1}) \prod_{n=1}^{N} g_n(y_n \mid x_{1:\tau(n)})$$

"Prior" probs
probabilities of sample

"Likelihood" probs
probabilities of observe

Things to remember from this lecture

Probabilistic programs automate probabilistic inference

Many many problems can be naturally expressed as PP

How do we write probabilistic programs

What probabilistic programs specify

Topics: a grand tour

September 4, 2023 (W1 L1)	**What is probabilistic programming?** What is model-based reasoning? The anatomy of a probabilistic program. Course structure.
September 7, 2023 (W1 L2)	**Generative thinking.** How to write probabilistic programs? What is the distribution probabilistic program captures?
September 11, 2023 (W2 L1)	**Basic inference procedures:** Enumeration, Rejection sampling, Importance Sampling, Metropolis-Hastings MCMC, Sequential Monte Carlo (Particle filtering). Why do they work?
September 14, 2023 (W2 L2)	**Implementation strategies.** Database view. Continuations. Message passing.

Topics: a grand tour

September 18, 2023 (W3 L1)	**Gradient-directed probabilistic inference**
September 21, 2023 (W3 L2)	**Learning for inference**
September 24 2023 (W4 L1)	**Programs with stochastic support**
September 28 2023 (W4 L2)	**Programmable inference**
October 2, 2023 (W5 L1)	**Connection between probabilistic and logical reasoning**
October 5, 2023 (W5 L2)	**Probabilistic logic programming**

Topics: a grand tour

October 9, 2023 (W6 L1)	**Incremental and anytime inference**
October 12, 2023 (W6 L2)	**Deep probabilistic programming**
October 16, 2023 (W7 L1)	**Deep generative models**
October 19, 2023 (W7 L2)	**Generalised paradigms for probabilistic programming**
October 23, 2023 (W8 L1)	**No probability? No problem! Alternative sources of probabilities.**
October 26, 2023 (W8 L1)	**Learning probabilistic programs**

All course practicalities

https://sebdumancic.github.io/courses/1_prob_prog/

Course practicalities

This is a seminar course

I expect you to come prepared

I expect you to talk more than me

I expect you to do more than just learn the material

There is no textbook, we will use research papers

Course components

Paper reviews (0%)

Participation (10%)

Presentation (25%)

Research report (65%)

Course components: Presentation

Each of you will present one paper

Your goal is to present the idea as understandable as possible
(and prepare discussion points)

Schedule a meeting with me at least 2 days in advance

Course components: Report

Design a research project without executing it

Four components:

- Topic description
- Relation to other topics in the course
- Analysis of the state of the art
- Research design(s). (What, How, Why, Wrong, Experiments)

Feedback time

Why take this course?

You are interested in a principled and unifying paradigm of AI

You want to become a truly Bayesian expert so that you know what your AI models don't know

You want to develop your research skills

You are interested in research

Why not take this course?

If you are looking for an easy course

If you are not in the mood for being out of your comfort zone

Last remarks

Choose your papers by September 12

The course is suitable both for 1st and 2nd year of MSc

The official PPL for the class if Gen.jl