Beyond calculation: Probabilistic Computing Machines and Universal Stochastic Inference

Vikash K. Mansinghka

December 17, 2011

NIPS Workshop on Philosophy and Machine Learning

Acknowledgements

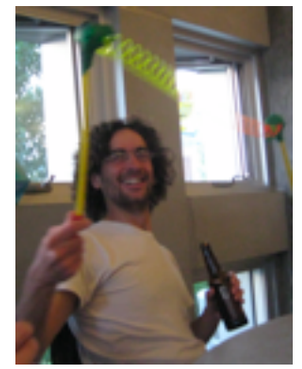
Eric Jonas

Keith Bonawitz

Cap Petschulat

Josh Tenenbaum

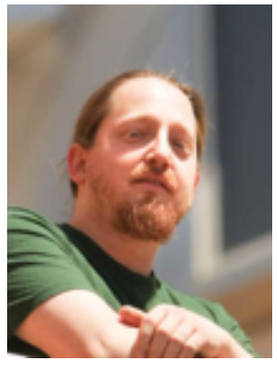
Dan Roy



Noah Goodman

Max Gasner

Beau Cronin



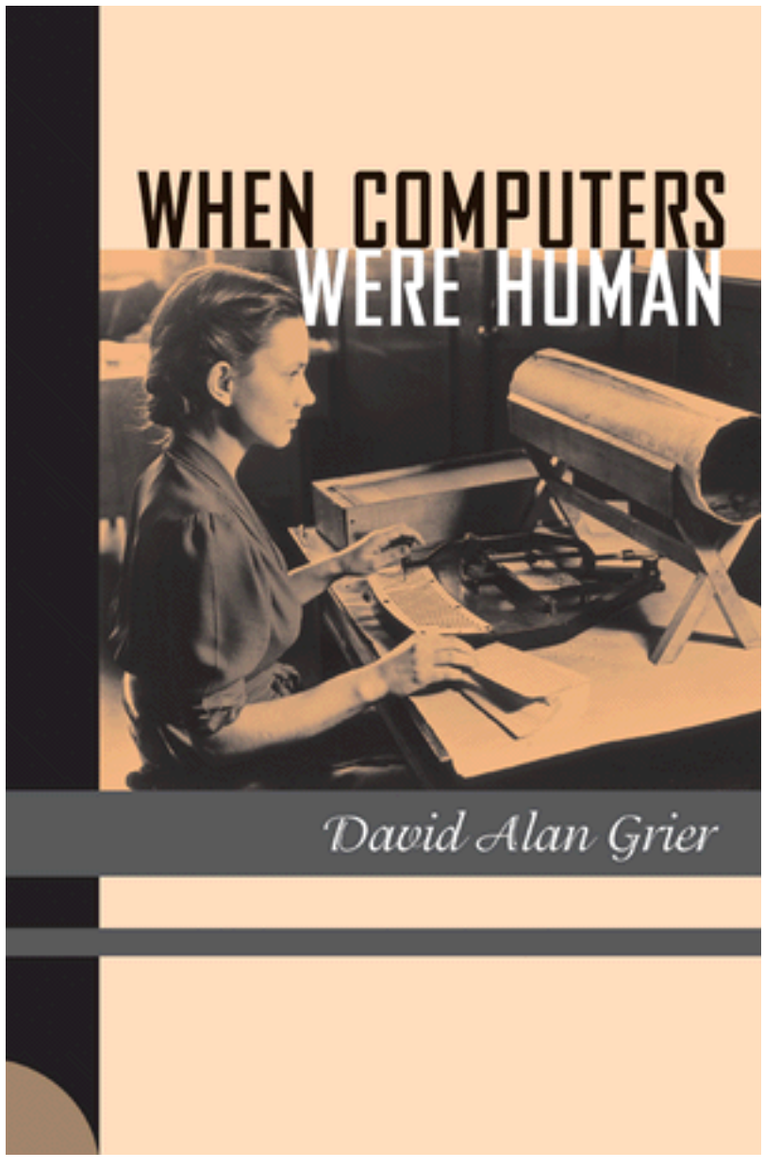
Cameron Freer

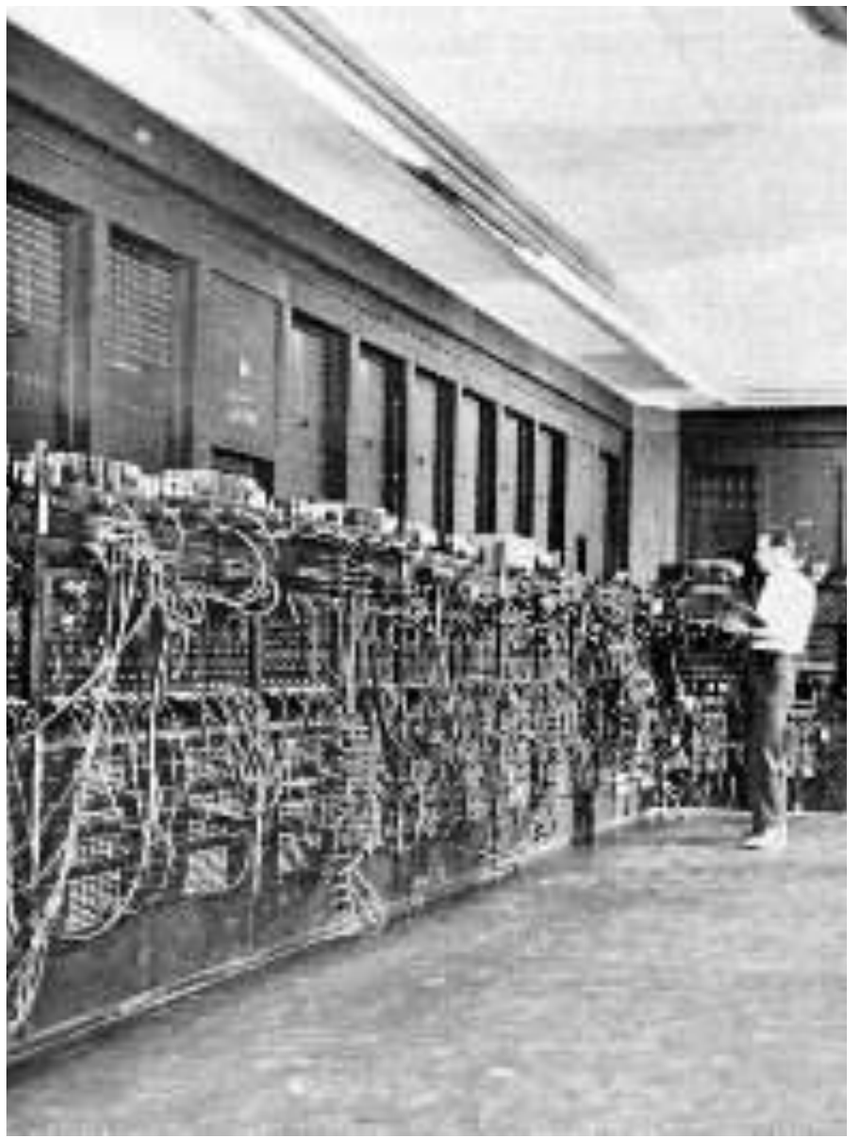
Tom Knight

Gerry Sussman

Tomaso Poggio

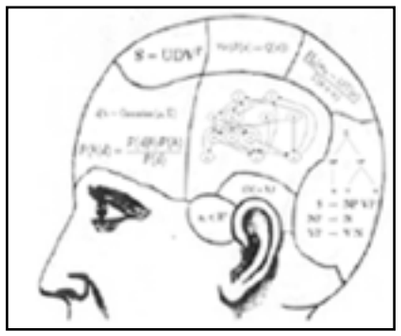
Computers were built for calculation and deduction

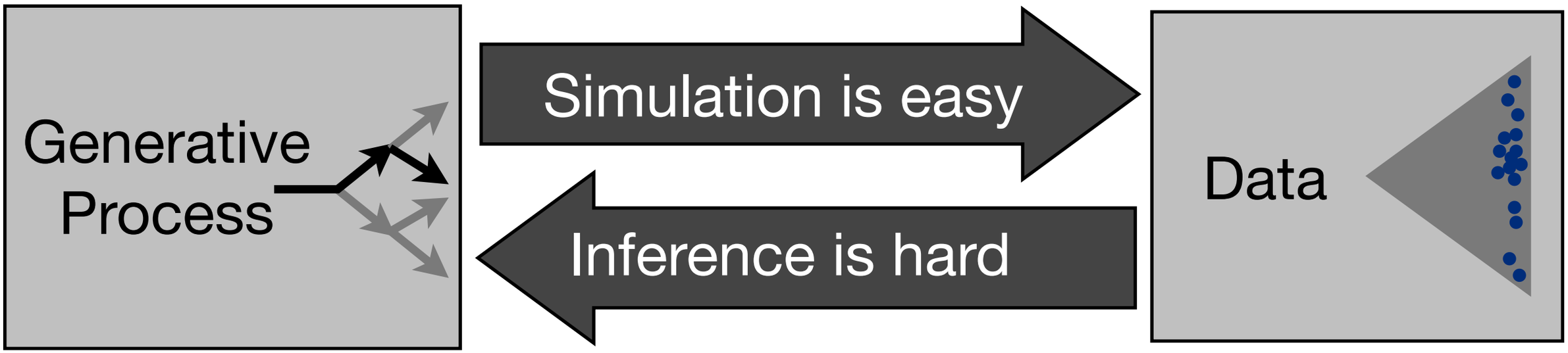




compute, v: *to determine by mathematical means; to calculate*

Probabilistic inference seems central to intelligence, but also cumbersome, intractable, so we simplify and approximate





$$P(\text{model} \mid \text{data}) = \frac{P(\text{model})\, P(\text{data} \mid \text{model})}{P(\text{data})}$$

Exponential domain

Exponential normalizer

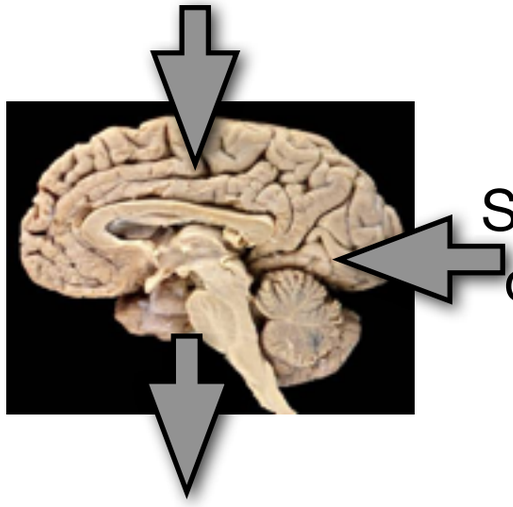
The mind and brain accomplish far more than our smartest computer systems, and they do it with far less. We need greater *expressiveness* and *tractability*, for making both *inferences and decisions*.

---

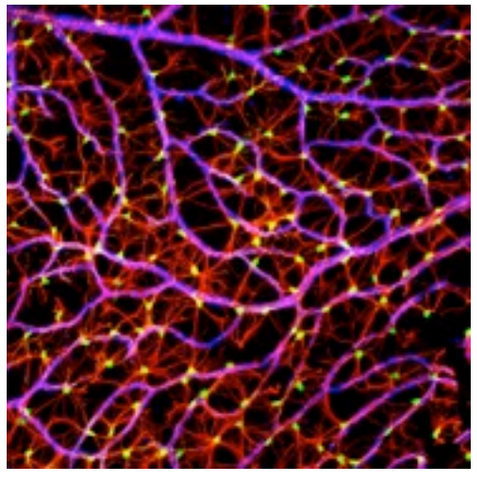
30 watts, 100Hz, sees, hears, navigates, negotiates relationships, led team that built Watson

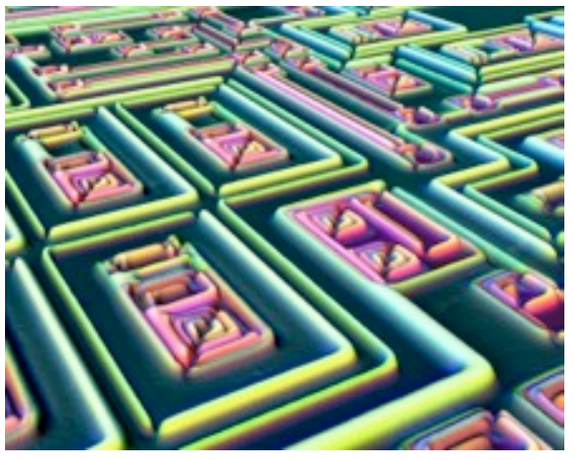
80 kilowatts, 3.55 GHz, world Jeopardy! champion, via statistical calculation

Genetic & physical constraints

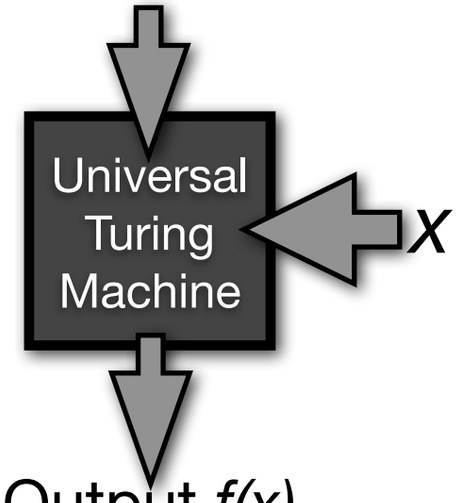


Cognition, perception, action





Function *f()*, as program that calculates



Output *f(x)*

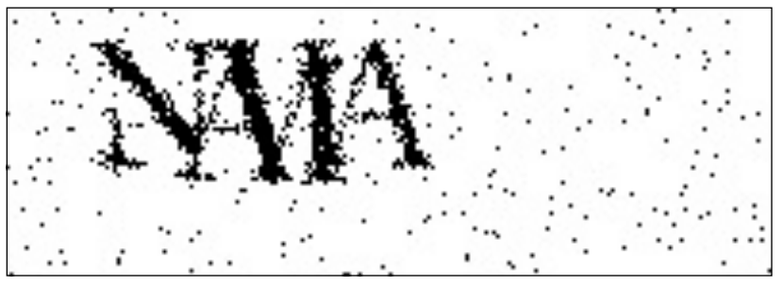
Outline

- **Motivation**: the capability and efficiency gaps with biology
- **Phenomenon**: examples of probabilistic programming systems
- **Philosophy**: a new mathematical model of computation
- **Potential:** computing machines for which induction, abduction are natural

CAPTCHAs are easy to make, hard to break

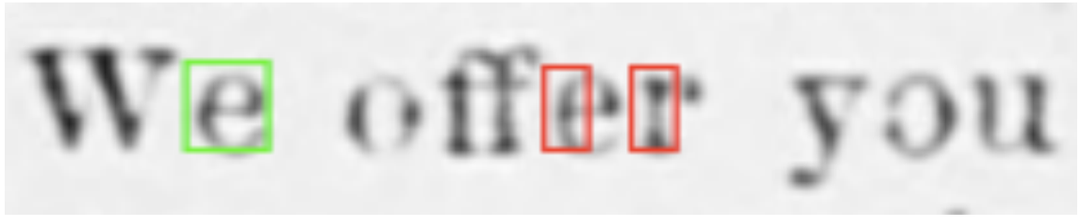
Generating CAPTCHAs: easy

{N,A,V,I,A}



Breaking CAPTCHAs: hard

Google CAPTCHA



OCR (CVPR 2010)

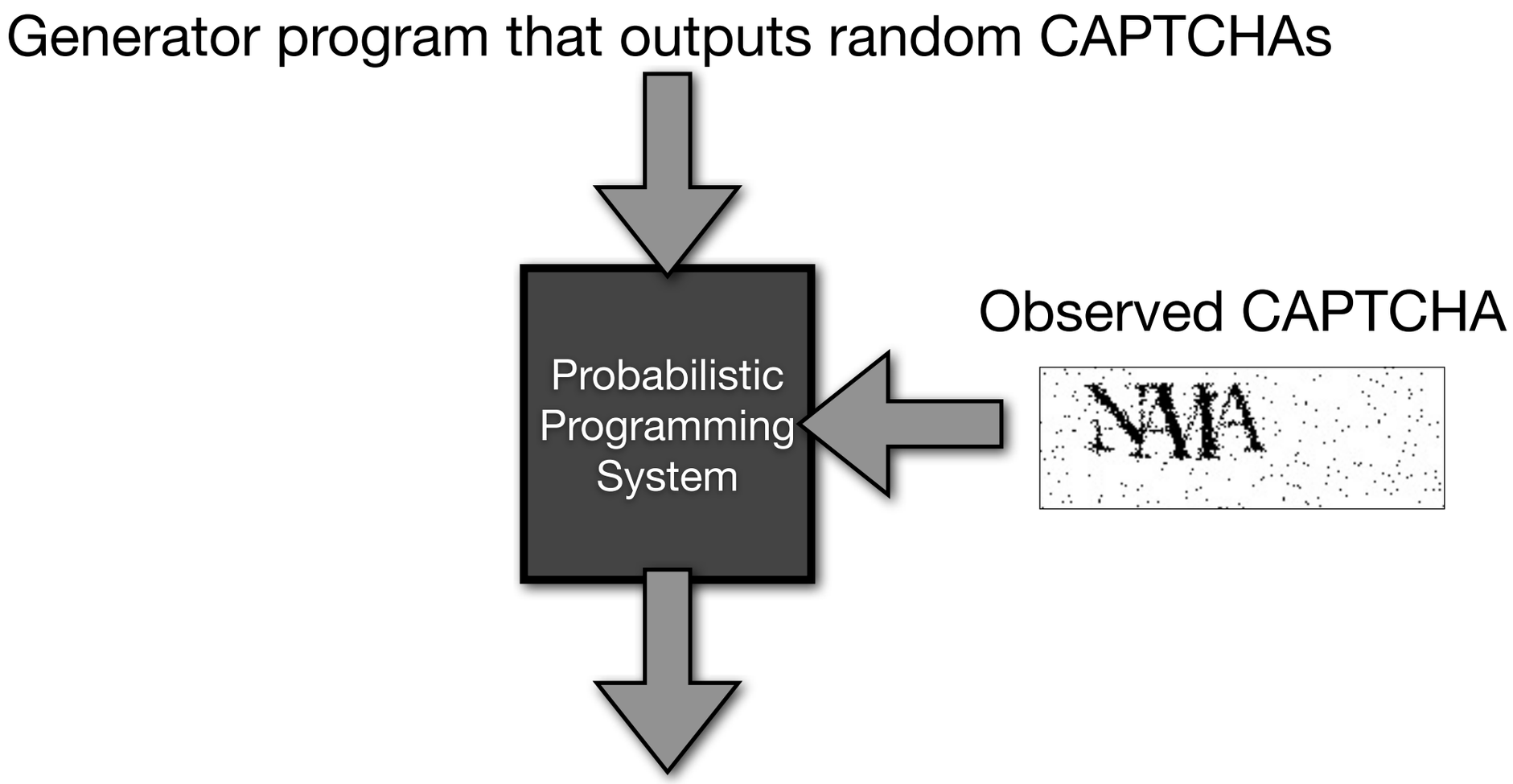
Breaking simple CAPTCHAs by running a randomized CAPTCHA generator backwards

---

Input Captcha

Guessed Explanation

Breaking simple CAPTCHAs by running a randomized CAPTCHA generator backwards



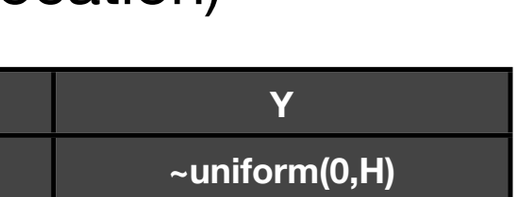
Breaking simple CAPTCHAs by running a randomized CAPTCHA generator backwards

1. Randomly choose some glyphs (with font and location)

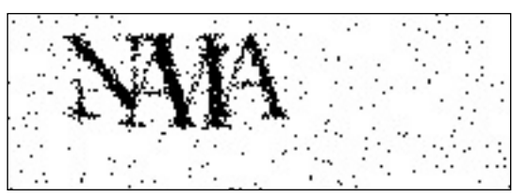
Glyph	Font	X	Y
~uniform(A,Z)	~uniform(0,2)	~uniform(0,W)	~uniform(0,H)
A	1	98	19
J	2	140	10
Q	1	43	7
S	0	98	3
J	1	80	15

2. Render to an image

3. Add some noise
(spatial noise + pixelwise errors)



4. Observe that the output = image we're interpreting



Inference

Breaking simple CAPTCHAs by running a randomized CAPTCHA generator backwards

```
(define w 200)
(define h 70)
(define rate 0.5)
(define maxglyphs 12)

(define blank (image/make w h))
```

1. Randomly choose some glyphs

```
(define maybe_sample_glyph
  (lambda ()
    (if (bernoulli rate)
      (image/glyph/sample w h)
      #f
    )
  )
)

(define glyphs
  (vector/remove
    (vector/draw
       maybe_sample_glyph
       maxglyphs)
    #f
  )
)
```

2. Render to an image

```
(define rendered
  (image/scene/draw blank glyphs))
```

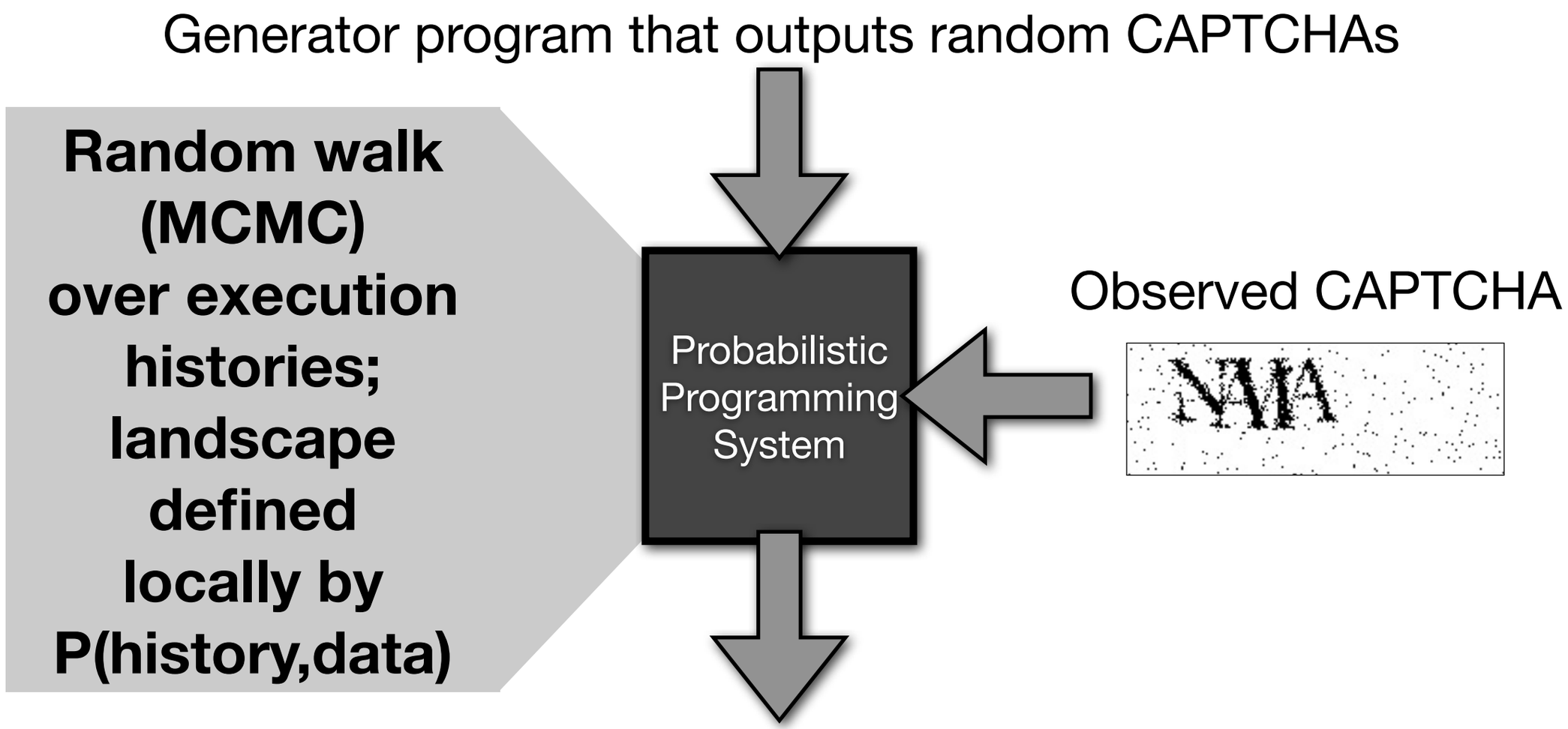
3. Add some noise

```
(define blur_radius
  (continuous-uniform:double-drift 0 10))
(define blurred
  (image/blur rendered blur_radius))

(define constant0
  (discrete-uniform 1 127))
(define constant255
  (discrete-uniform 128 254))

(define blurred_with_noise
  (image/interpolate
    blurred constant0 constant255
  )
)
```

4. Observe that the output matches the target image

```
(define observed (image/load "image.png"))
(observe
  (image/stochastic_binarize
     blurred_with_noise)
  observed
)
```

Breaking simple CAPTCHAs by running a randomized CAPTCHA generator backwards

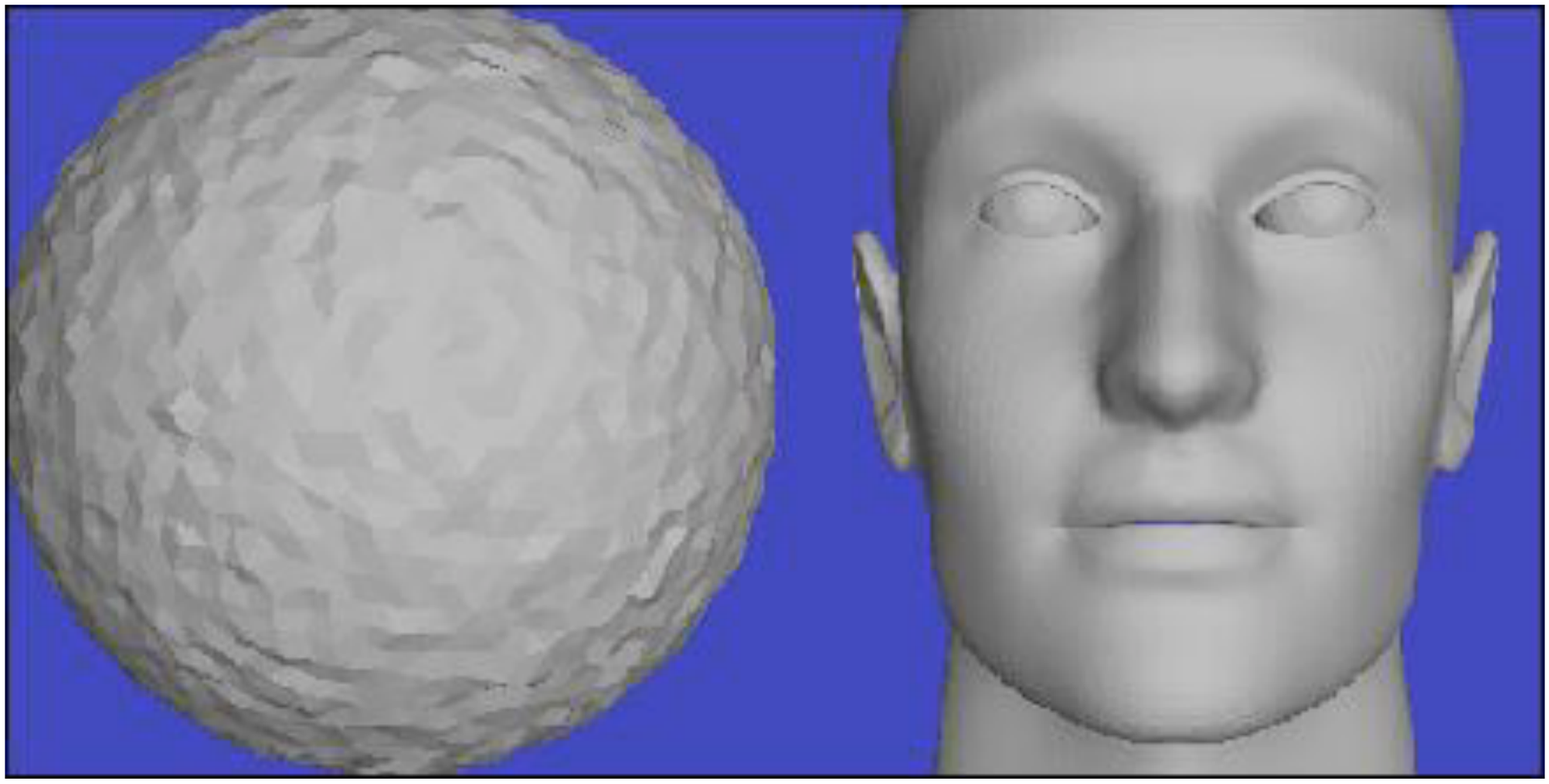


Converges well due to **inclusion of (overlooked) randomness**
Fast iterations due to **conditional independence (asymptotics, locality, parallelism)** and **software+systems engineering (small state, fast updates)**

Computer vision as “inverse Pixar”, using stochastic MATLAB

Posterior geometry, rendered (known lighting, unknown mesh; weak smoothness prior)

Target Image



(Wingate et al, 2011)

Applications of Probabilistic Programming Systems, including Church

1. **Nuclear safety via CTBT monitoring** (Arora, Russell, Sudderth et al, 2009-2011)

2. **Tracking multiple targets from video, radar** (Arora et al 2010; Oh et al 2009)

3. **Automatic control system synthesis** (Wingate et al 2011)

4. **Information extraction from unstructured text** (McCallum et al 2009)

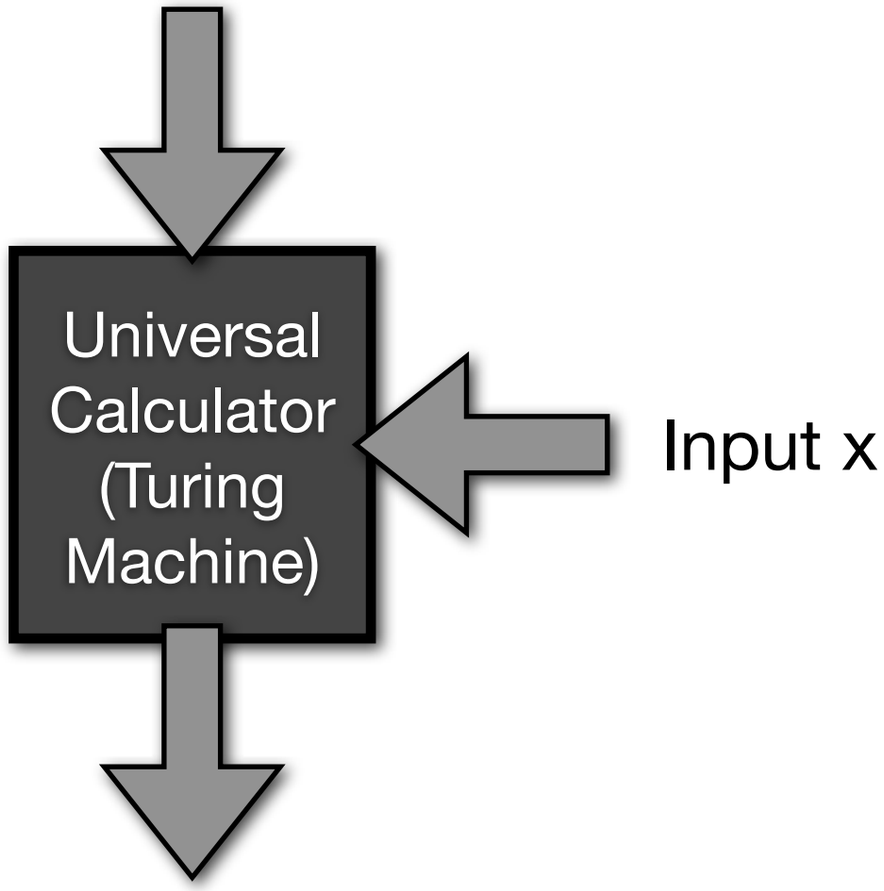
5. **Automatic statistical analyst** (Mansinghka et al 2009; 2011 in prep)

6. **Clinical reasoning and pharmacokinetics** (Mansinghka et al in prep)

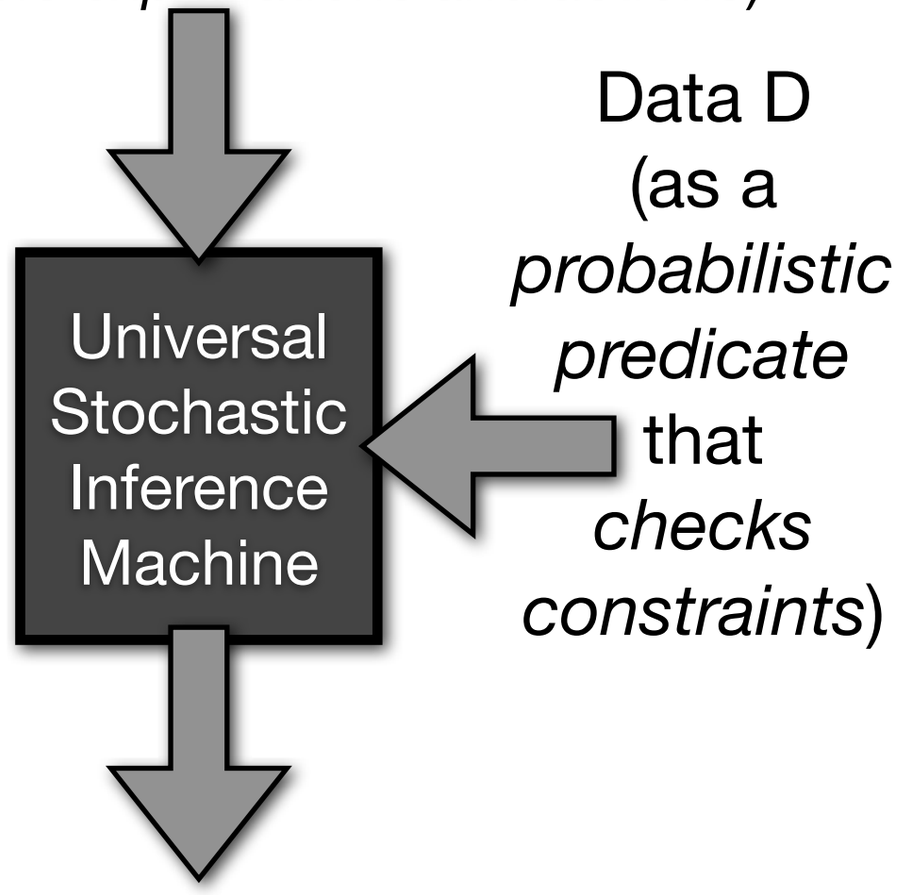
7. **Human learning as probabilistic program induction** (Stuhlmuller et al 2011, Tenenbaum; Kemp; Griffiths; Goodman)

Probabilistic computing: Computation as *stochastic inference*, not deterministic calculation

Function f()
(as a *program* that *calculates*)



Space of possibilities ~P(H)
(as a *probabilistic program* that *guesses possible explanations and actions*)



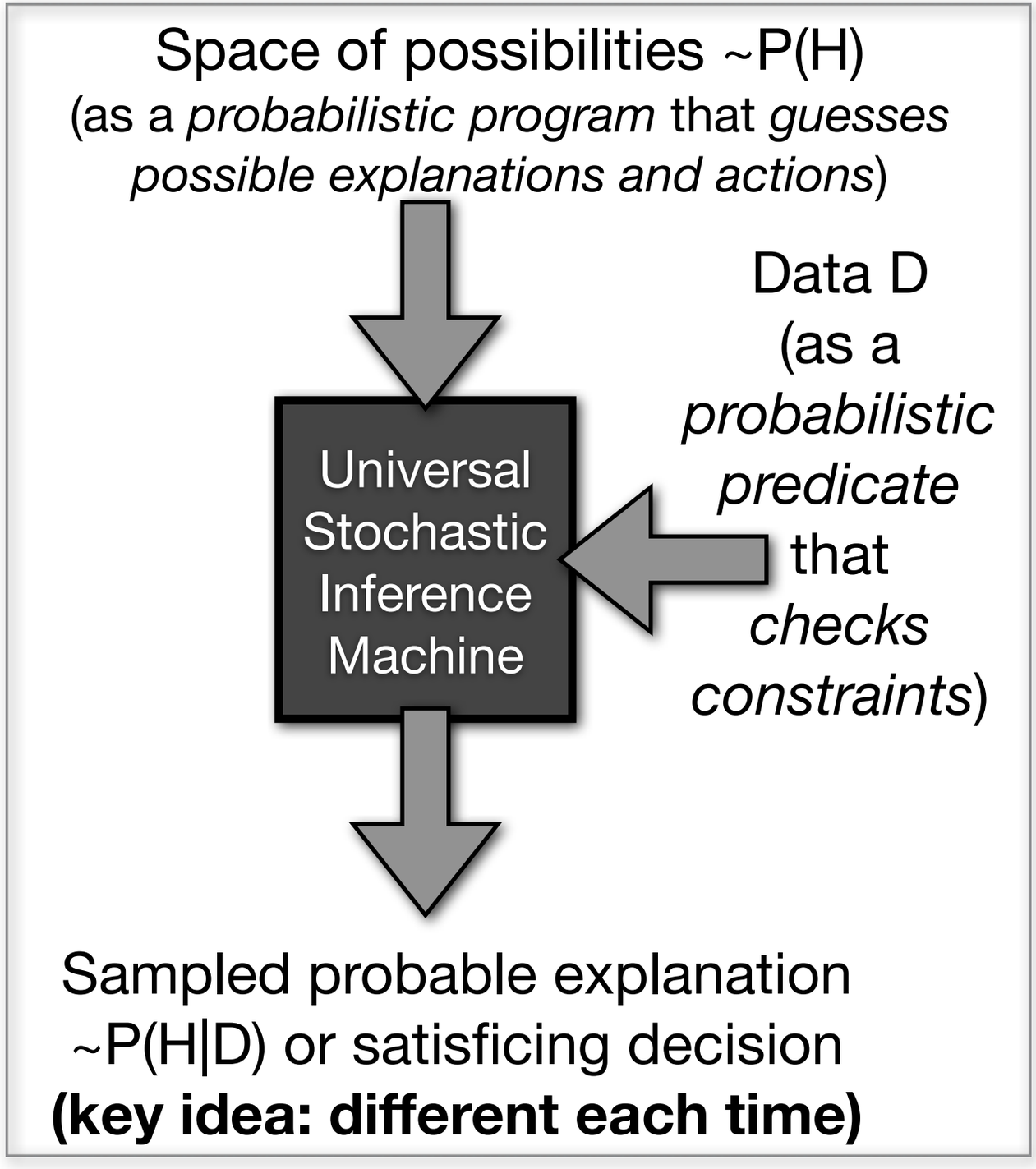
Probabilistic computing: Computation as *stochastic inference*, not deterministic calculation

Turing embedding:
H = (x, f(x))
D: H_x == x

Universality: ~P(H), D contain arbitrary stochastic inference

Prob. program induction:
H = <prob. program text>
D: <spec checker>

Meta-reasoning:
H = <model of agent>
D: <agent's behavior>



Probabilistic computing: Computation as *stochastic inference*, not deterministic calculation

AI systems,
models of cognition,
perception and action

Specialized Inference Modules

Universal Inference Machines

Parallel Stochastic Finite State Machines

Probabilistic Hardware

Commodity Hardware

Mansinghka 2009

Space of possibilities ~P(H)
(as a *probabilistic program* that *guesses possible explanations and actions*)

Data D
(as a *probabilistic predicate* that *checks constraints*)

Sampled probable explanation
~P(H|D) or satisficing decision
(key idea: different each time)

Snapshot of the field: new languages, systems, architectures, theory

10+ research prototype languages, **2 universal**

...

Church: 5 implementations

BLOG

Figaro

New Probabilistic Architectures for Universal Inference

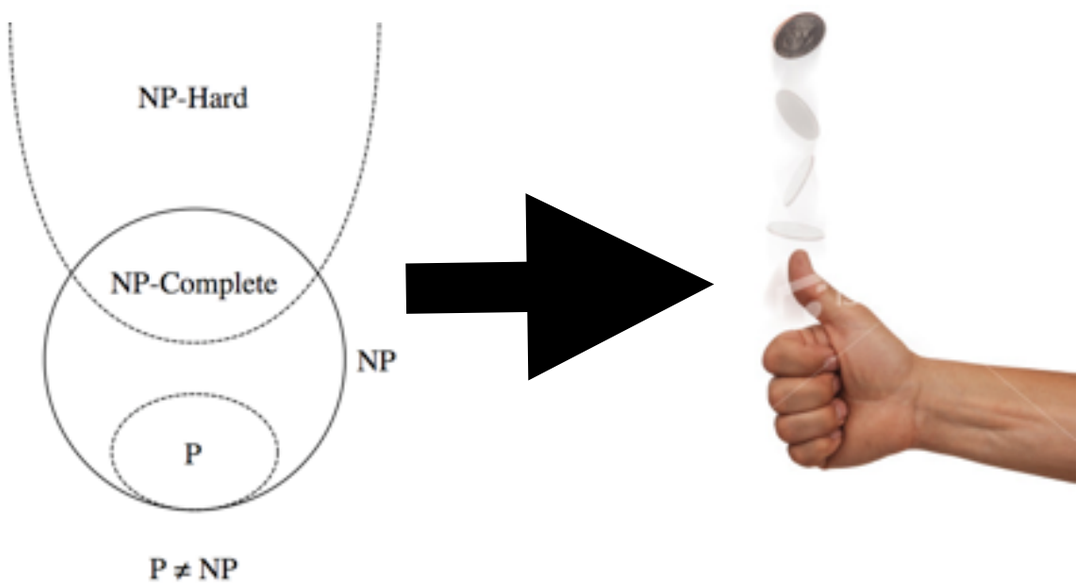
```
def is-trick? (bernoulli 0.01)
def weight (if is-trick? (uniform 0 1) 0.5)
obs (bernoulli weight) #t
```

Type	Symbol	Expression	Value
def	is-trick?	(bernoulli 0.01)	#f
def	weight	(if is-trick? (uniform 0 1) 0.5)	0.5
		if #t #f (uniform 0 1) 0.5	
obs		(bernoulli weight)	#t

Type	Symbol	Expression	Value
def	is-trick?	(bernoulli 0.01)	#t
def	weight	(if is-trick? (uniform 0 1) 0.5)	0.23
		if #t #f (uniform 0 1) 0.5	
obs		(bernoulli weight)	#t

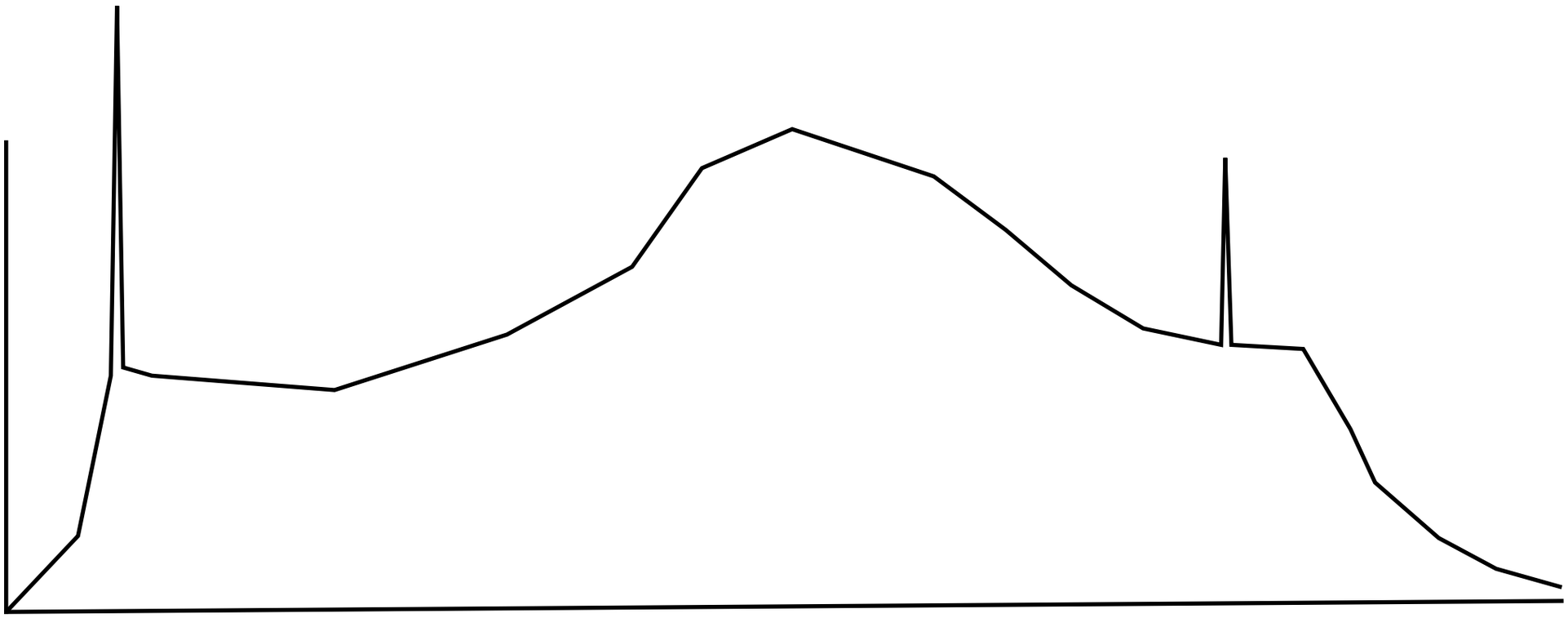
Wingate, Stuhlmuller, Goodman 2011
Arora, Russell et al 2010
Goodman, Mansinghka et al 2008

New Theorems in Probabilistic Computability and Complexity



Ackerman, Freer, Roy 2011 (& in prep)
Freer, Mansinghka, Roy 2010
Haeupler, Saha, Srinivasan 2010, Propp & Wilson 1996

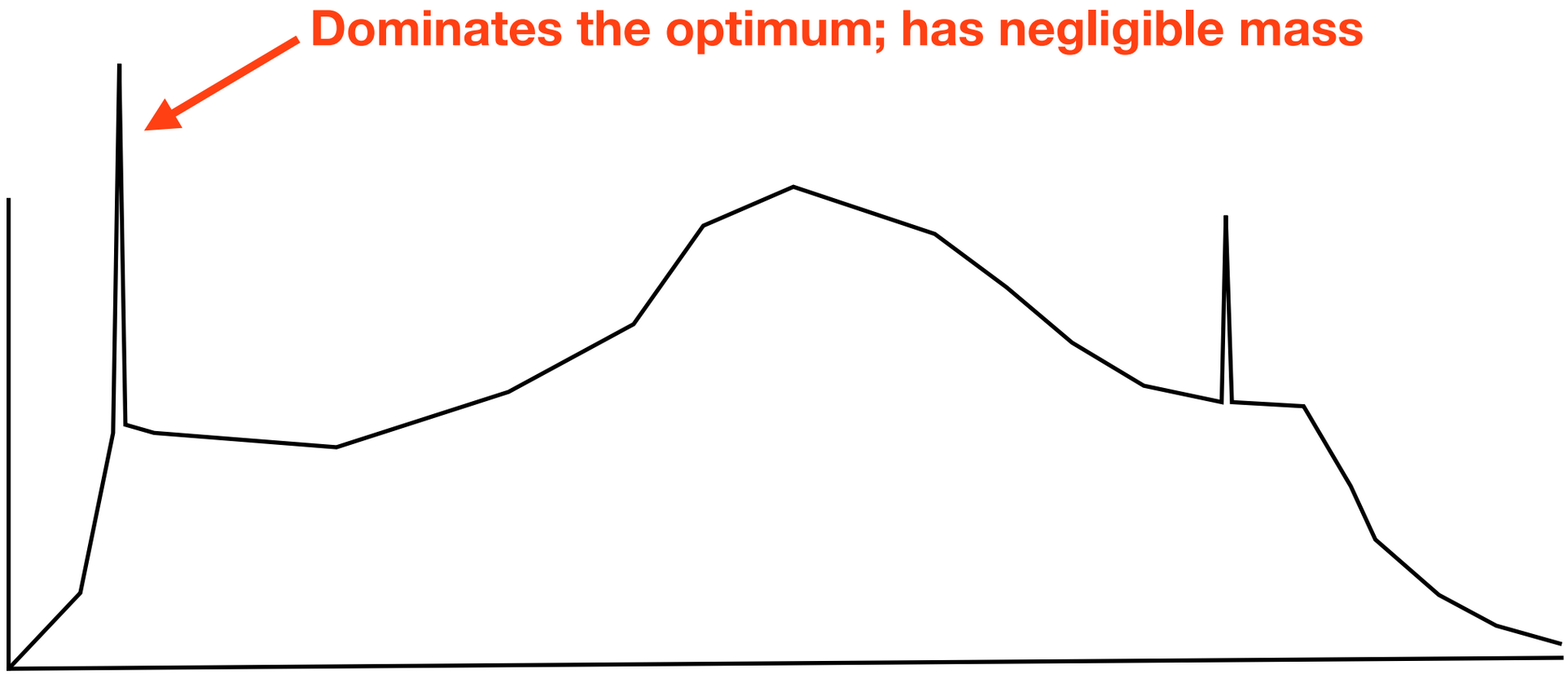
These machines make sampling more natural than optimization and integration



Claim: sampling is *easier* than both *optimization* and *integration*

Explaining away becomes natural (and a question of convergence), but calculating low probabilities exactly may be nearly impossible

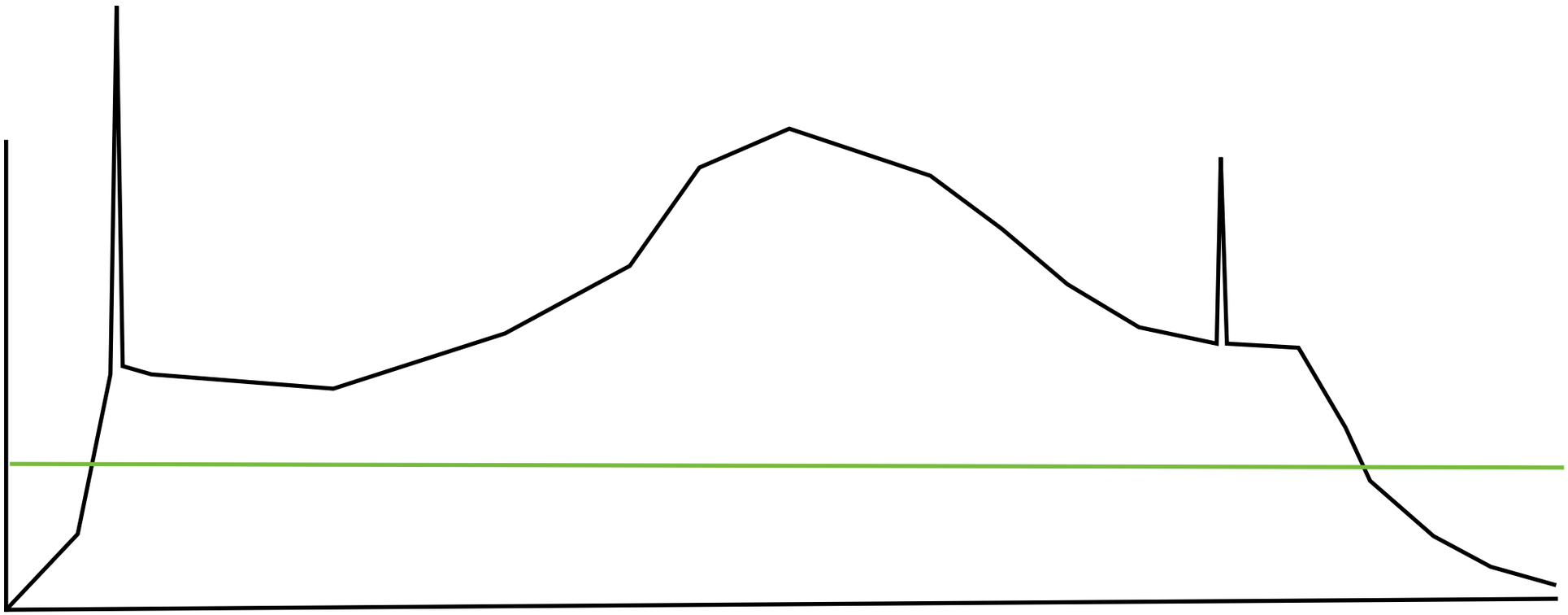
These machines make sampling more natural than optimization and integration



Claim: sampling is *easier* than both *optimization* and *integration*

Explaining away becomes natural (and a question of convergence), but calculating low probabilities exactly may be nearly impossible

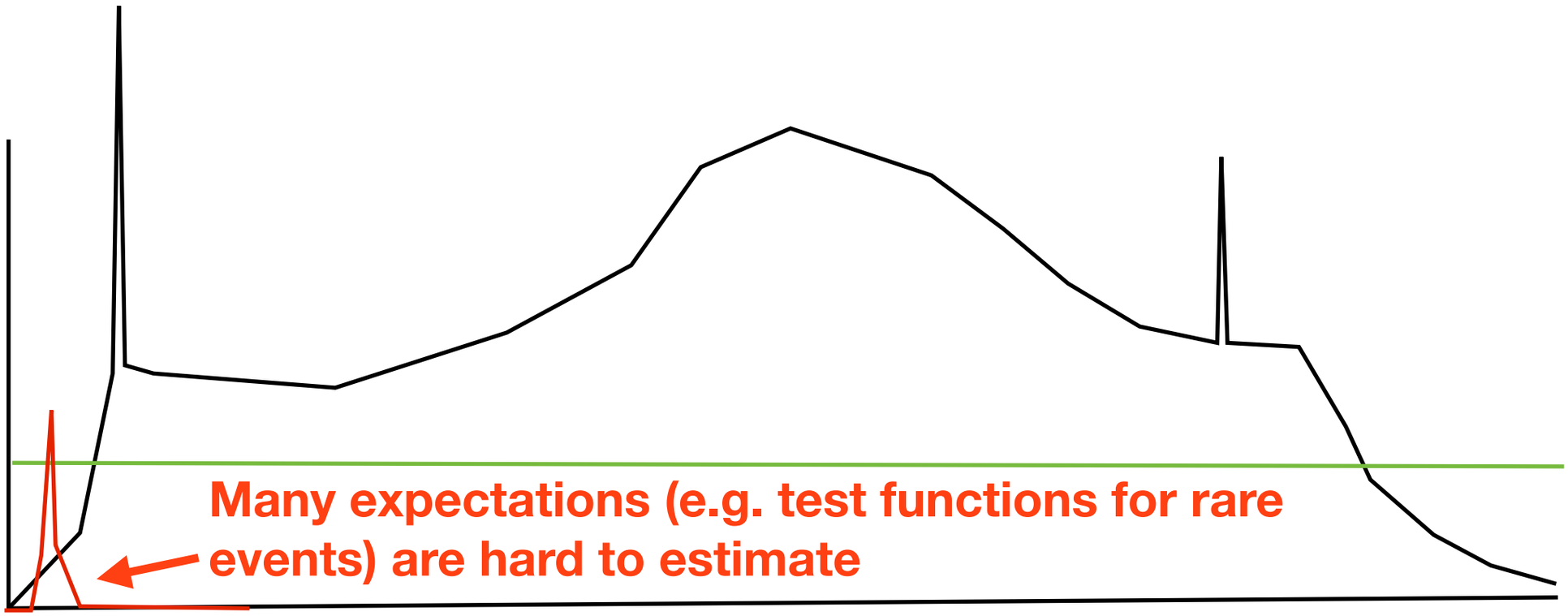
These machines make sampling more natural than optimization and integration



Claim: sampling is *easier* than both *optimization* and *integration*

Explaining away becomes natural (and a question of convergence), but calculating low probabilities exactly may be nearly impossible

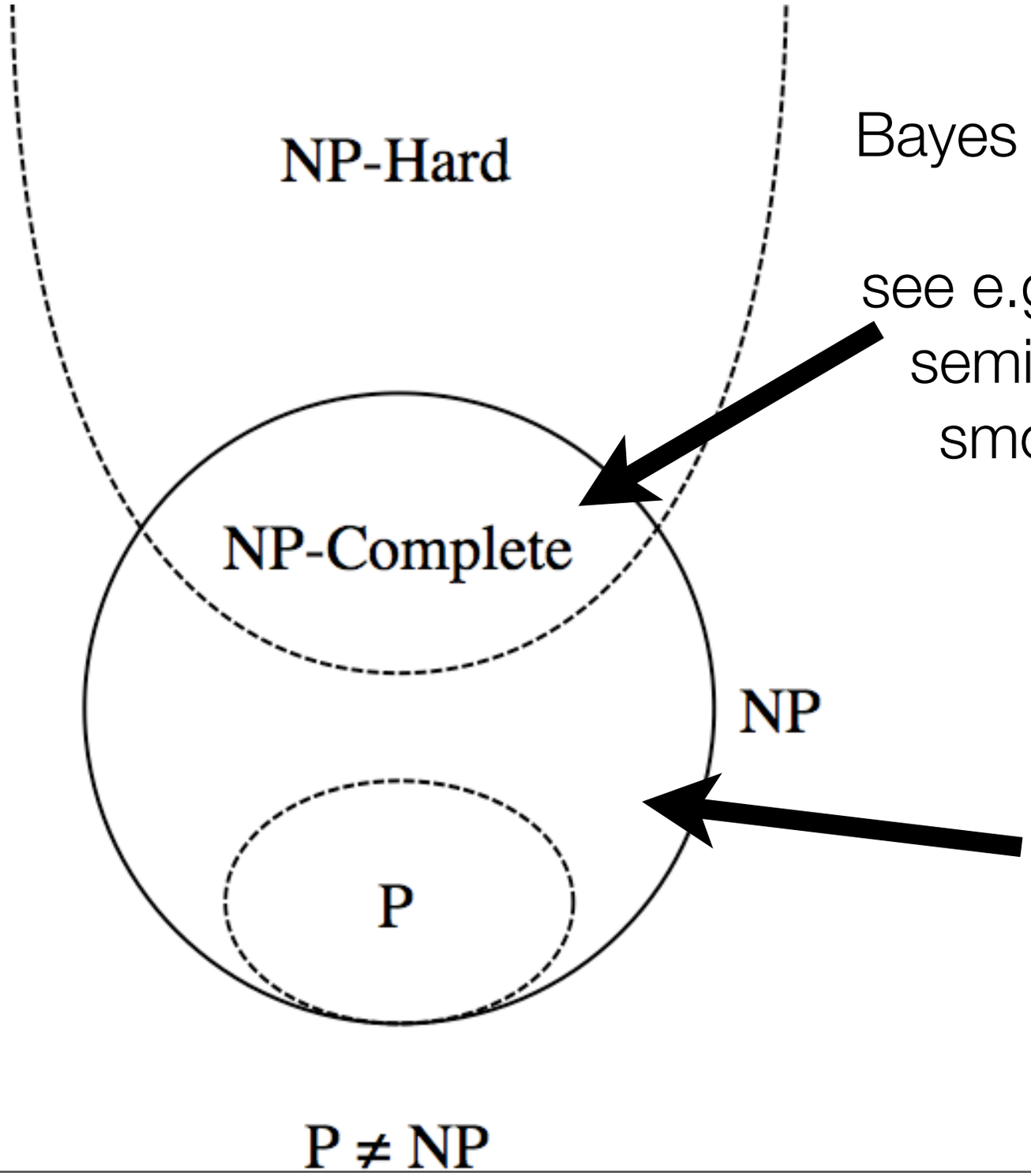
These machines make sampling more natural than optimization and integration



Claim: sampling is *easier* than both *optimization* and *integration*

Explaining away becomes natural (and a question of convergence), but calculating low probabilities exactly may be nearly impossible

What is the computational complexity of stochastic inference? (not P, NP, #P, BPP, ...)



Bayes reduction targets are usually easy:
SAT, k-color, ...
see e.g. phase transitions (Selman et al),
semi-random sources (Vazirani et al),
smoothed analysis (Spielman et al)

Usually hard, basis of crypto
(via generation of instances
that are hard in practice):
factoring, graph iso

What is the computational complexity of stochastic inference? (not P, NP, #P, BPP, ...)

```
define x1 (flip)
define x2 (flip)
define x3 (flip)
...
observe (or (not x1) x2 x3)
observe (or x2 (not x4) x5)
...
```

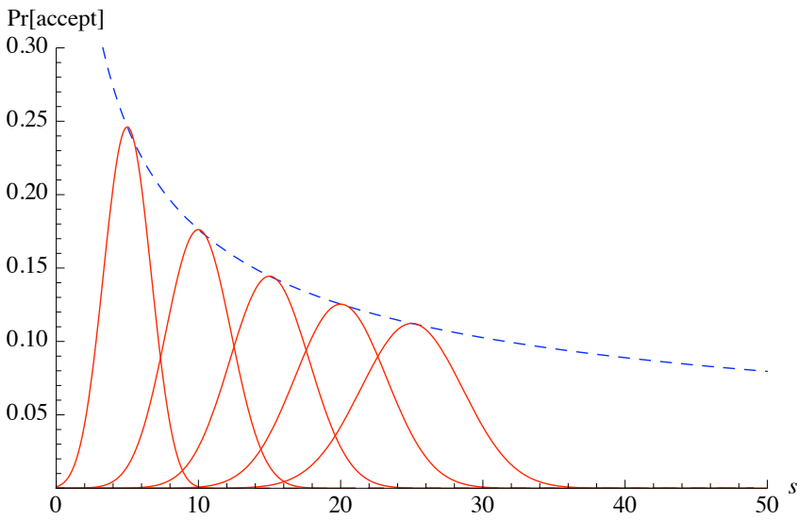
```
define x (multivariate-normal 0vec (* eps1 I))
define y (multivariate-normal (* A x) (* eps2 I))
observe (< (norm (- y b))) eps3

<20 lines code for Latent Dirichlet Allocation>
observe (get-word “doc1” 0) “hello”
observe (get-word “doc1” 1) “there”
observe (get-word “doc2” 0) “church”
...
```

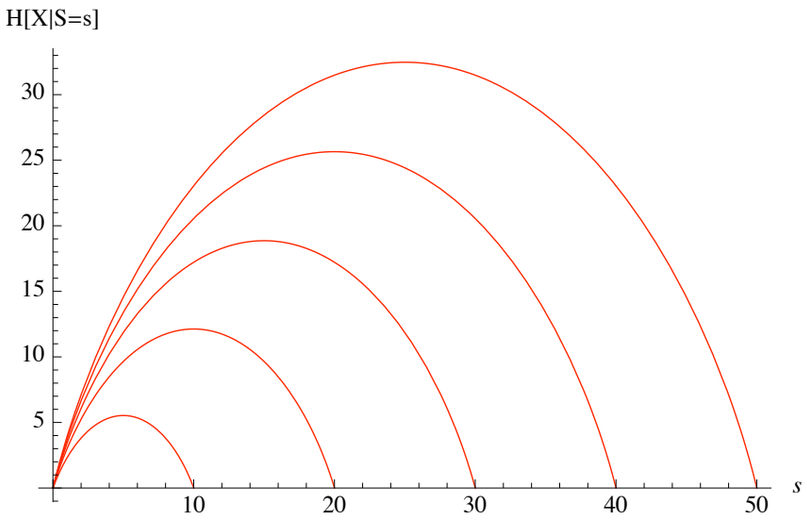
Conditional Distribution to be (KL-approx) Simulated

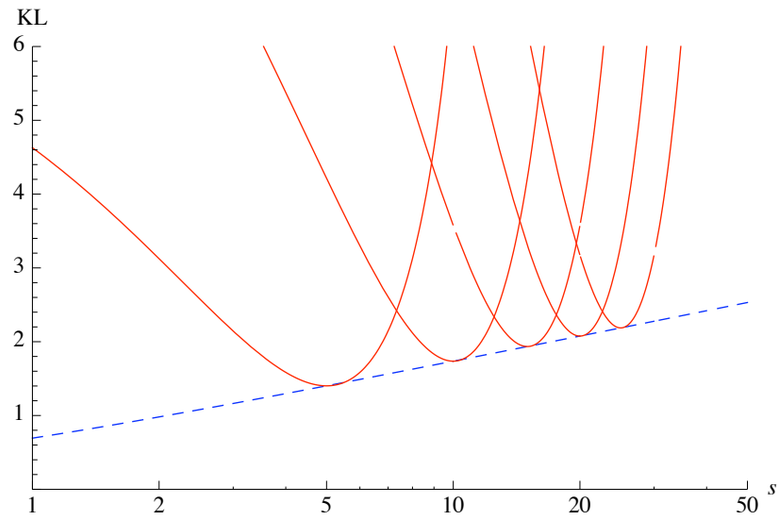
(0-entropy and uniform limits)

The Complexity of Exact Stochastic Inference by Rejection

Proposition. *Let N be the number of attempts before a rejection sampler for* `(query <exp> <pred>)` *succeeds when using samples from the distribution induced by* `<exp>` *and* `<pred>`*. Assume the application of* `<pred>` *to any input consumes no randomess, i.e.,* `<pred>` *is deterministic. Then N is geometrically distributed with mean* $\exp(D_{KL}(\texttt{(query <exp> <pred>)}\|\texttt{(eval <exp>)}))$.



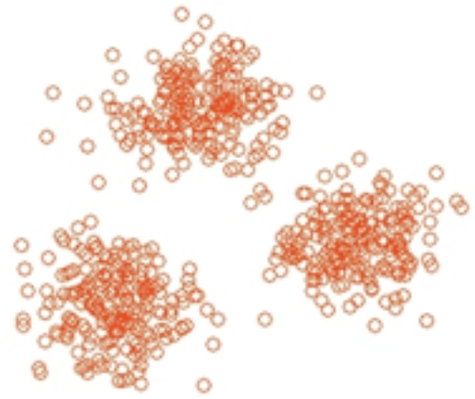


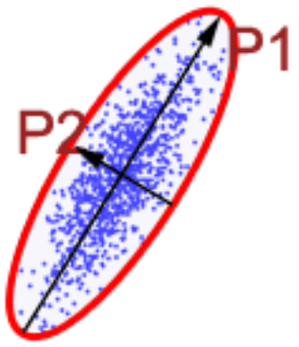


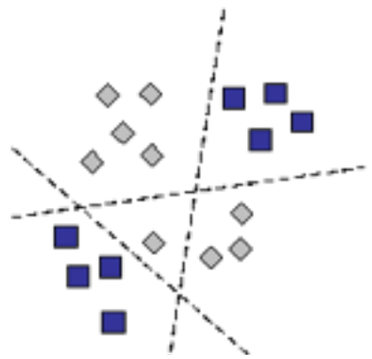
Semantic --- KL(posterior, prior) --- not syntactic (treewidth, dimensionality, ...)
Same KL as in PAC-Bayes: if easy to sample, then easy to learn (almost iff)
See Freer, Mansinghka, Roy 2010 for more results and details (incl. Markov chains)

Probabilistic Programming and Machine Learning

Big data and ML emphasis is on scaling flat statistical models to GB/TB. These models require 1-20 PLOC (probabilistic lines of code)

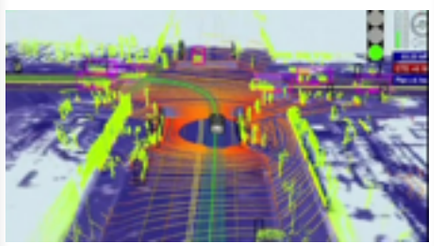






Bigger payoff: ***somewhat stochastic systems*** **that** ***marry structure and abstraction with statistics*****: 100-1K+ probabilistic LOC** ***and*** **GB/TB**

Allocating scarce resources with learned models, real constraints



Machine perception and control: vision as inv. graphics; control as inv. dynamics

NLP/MT/IX/...: jointly model lexicon, syntax, alignment, entities, coref, topics

Database Fusion & Cleanup: treat messy, incomplete rows as *evidence*, not reality

Modeling and Simulation: infer accurate parameters, realistic structure

Engineering stochastic machines

"With our artificial automata we are moving much more in the dark than nature appears to be with its organisms. We are, and apparently, at least at present, have to be much more 'scared' by the occurrence of an isolated error and by the malfunction which must be behind it. Our behavior is clearly that of overcaution, generated by ignorance."

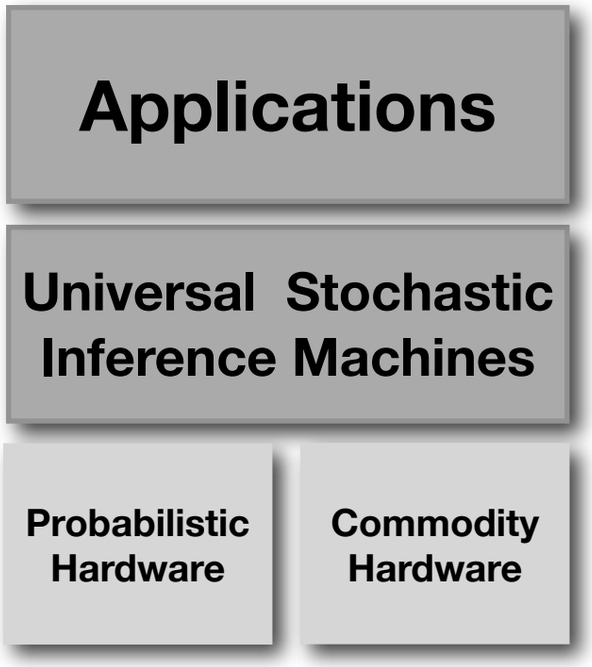
– John von Neumann, ***The General and Logical Theory of Automata***

"For over two millennia, Aristotle's logic has ruled over the thinking of western intellectuals. All precise theories, all scientific models, even models of the process of thinking itself, have in principle conformed to the straight-jacket of logic. But from its shady beginnings devising gambling strategies and counting corpses in medieval London, probability theory and statistical inference now emerge as better foundations for scientific models, especially those of the process of thinking and as essential ingredients of theoretical mathematics, even the foundations of mathematics itself. We propose that this sea change in our perspective will affect virtually all of mathematics in the next century."

— David Mumford, ***The Dawning of the Age of Stochasticity***

What if stochastic inference was as fast, cheap and ubiquitous as deterministic calculation?

mm km



Mansinghka 2009