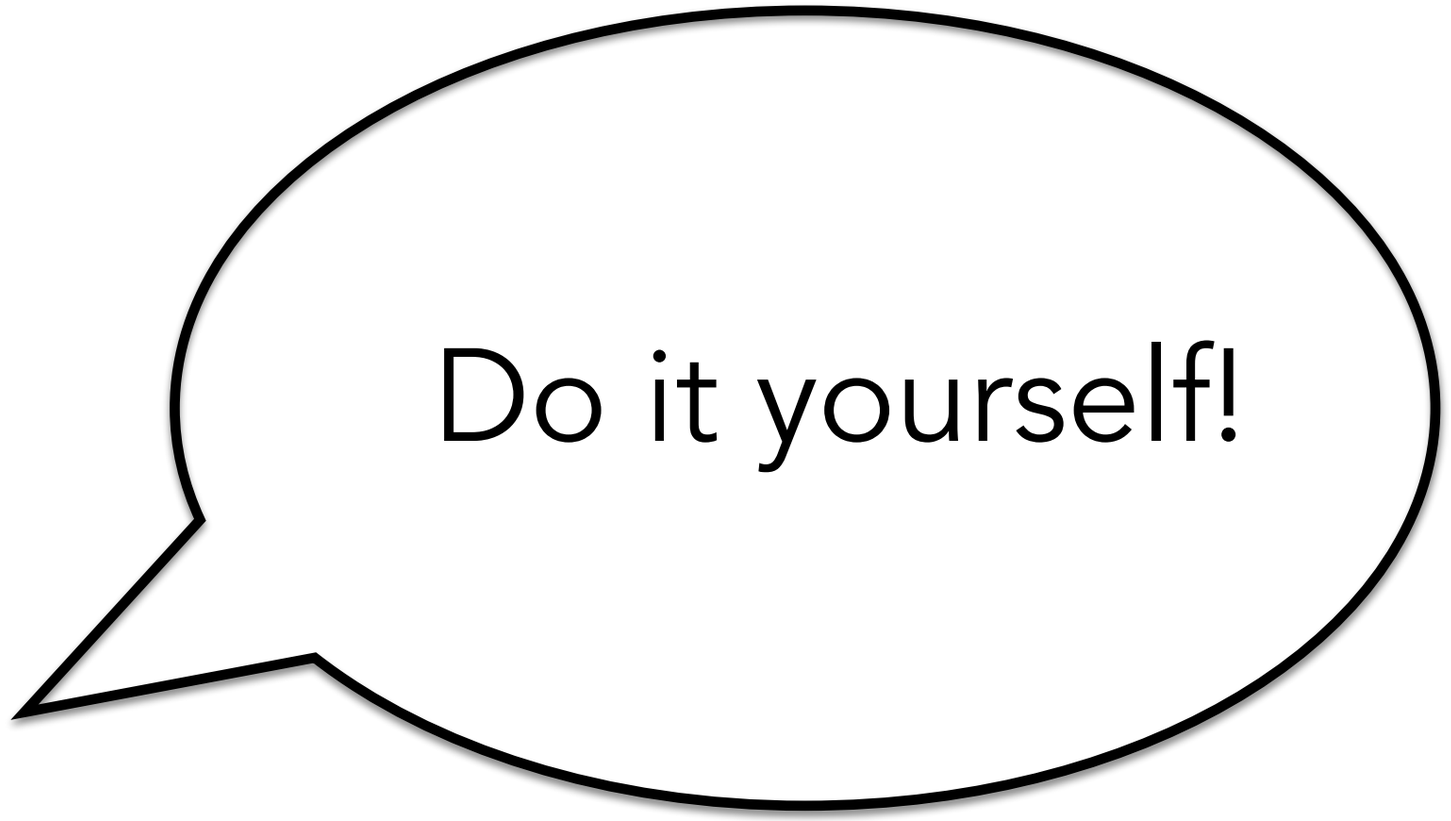
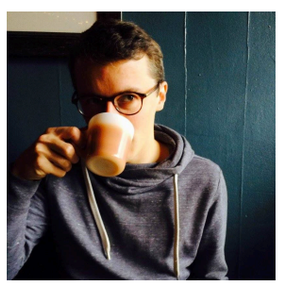
Tutorial

Mental models as probabilistic programs





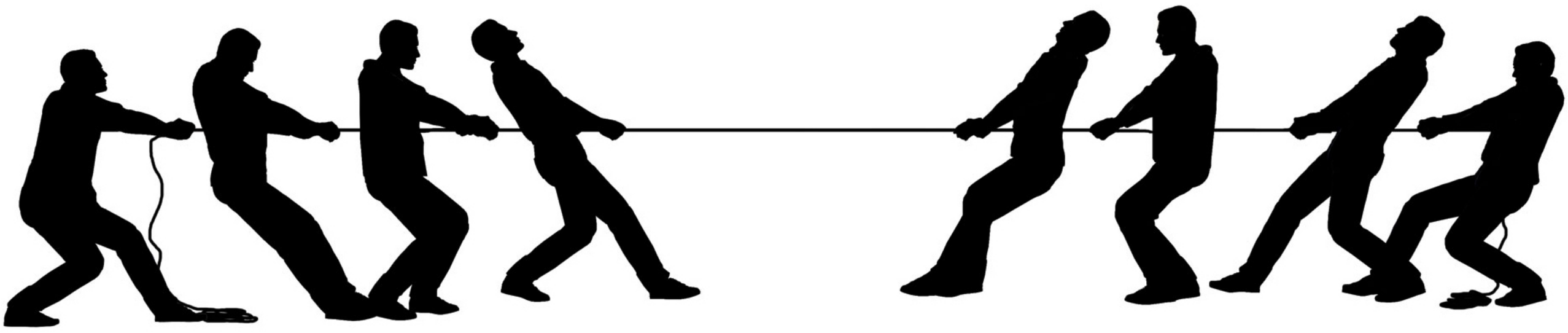
Marta Kryven
mkryven@mit.edu



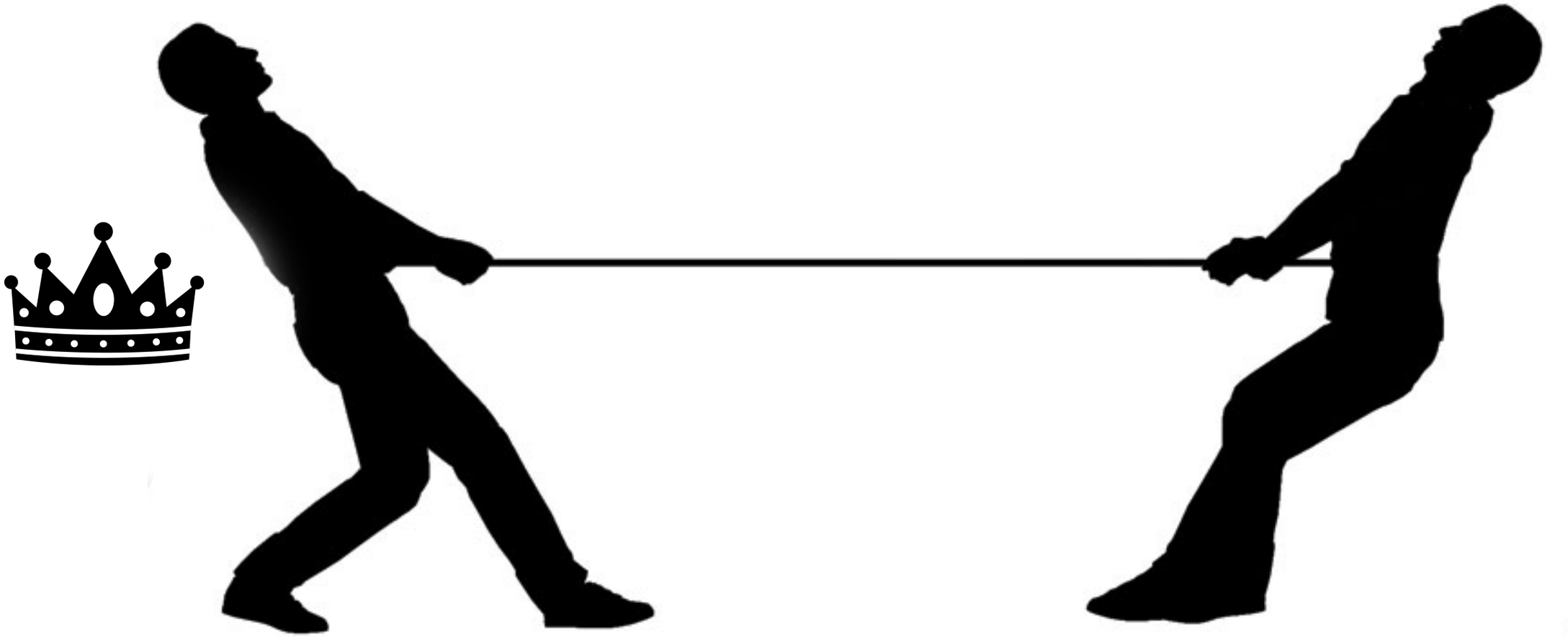
Thanks to Tobi Gerstenberg for the slides

August 17, 2020

Probabilistic problem example



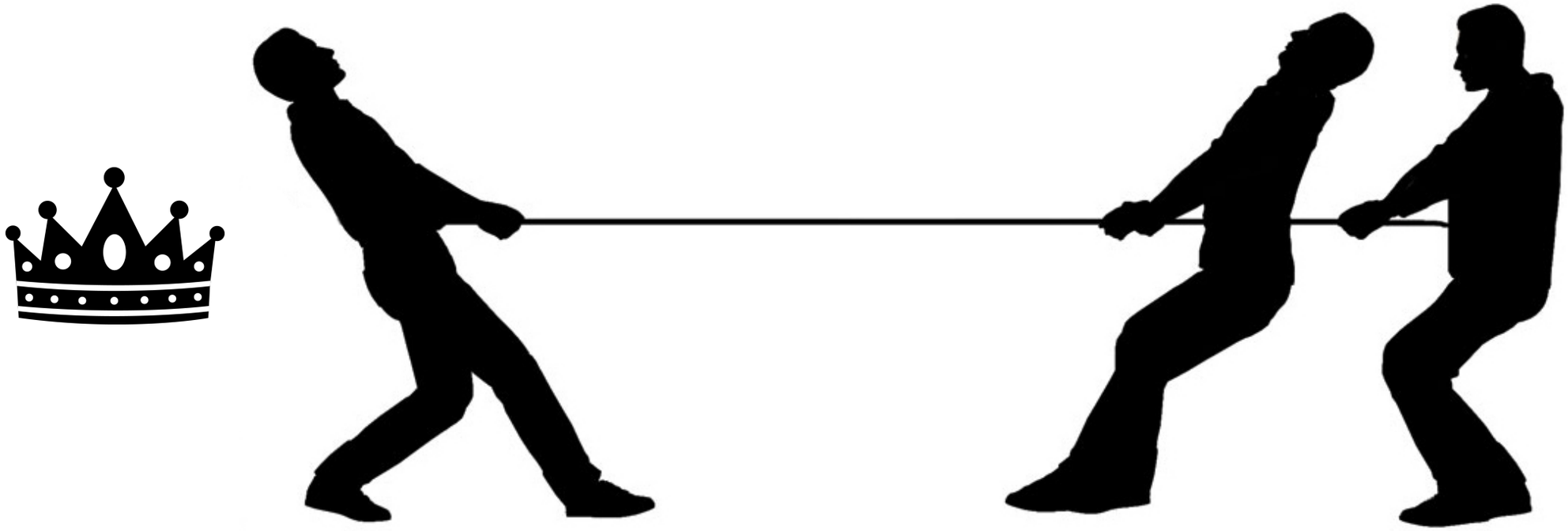
Tug of war is a sport that pits two teams against each other in a test of strength



Tom

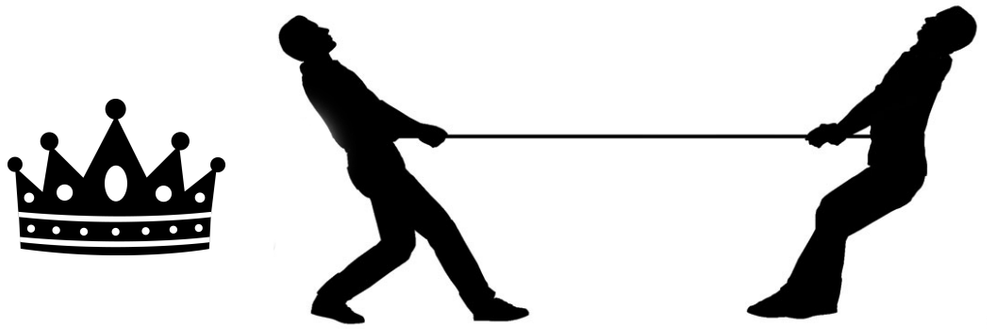
Steve

How strong is Tom (on the scale 1 to 10)?

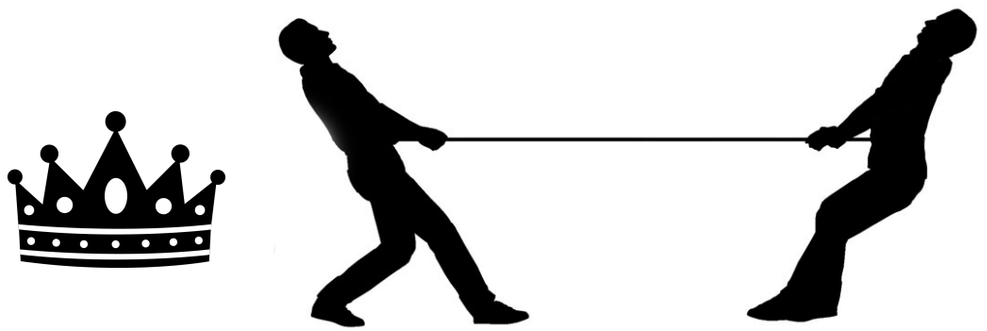


Bill Mark Nick

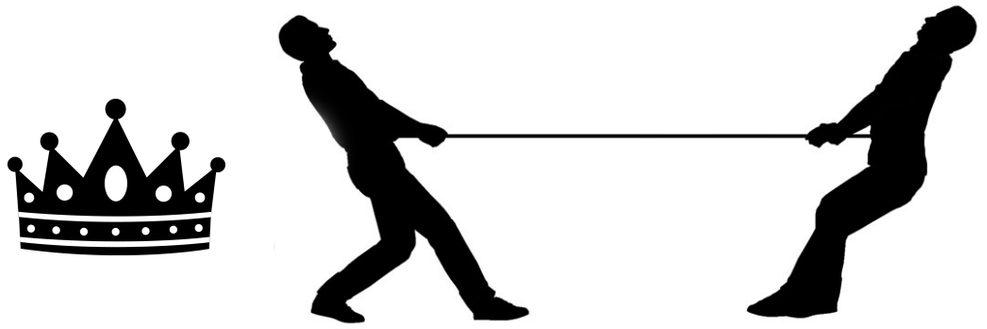
How strong is Bill?



Tom Steve

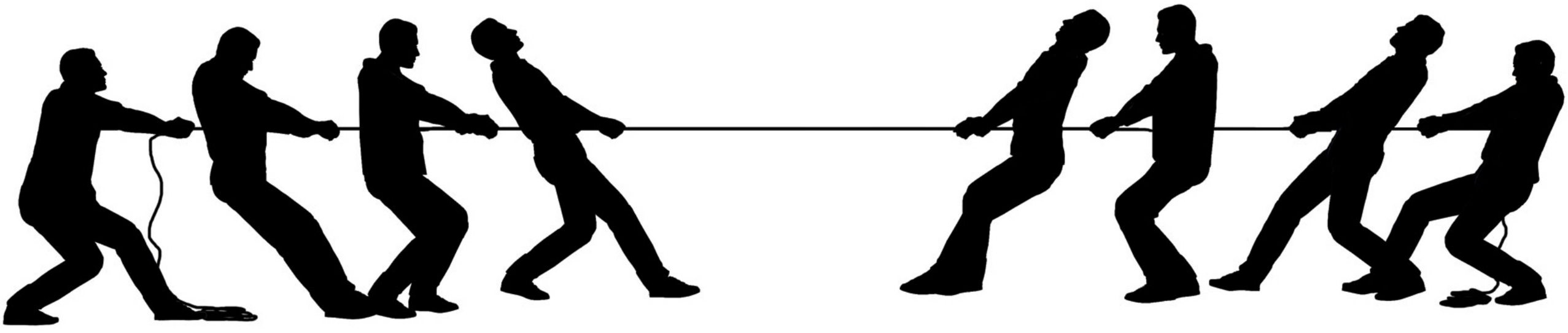


Bill Steve



Ryan Steve

How strong is Tom?



Concepts:

teams

person

winner

strength

pulling

effort

A computational model to infer strength from observation

```
var towModel = function() {
  var strength = mem(function (person) {return gaussian(50, 10)})

  var lazy = function(person) {return flip(0.1) }

  var pulling = function(person) {
    return lazy(person) ? strength(person) / 2 : strength(person) }

  var totalPulling = function (team) {return sum(map(pulling, team))}

  var winner = function (team1, team2) {
    totalPulling(team1) > totalPulling(team2) ? team1 : team2 }

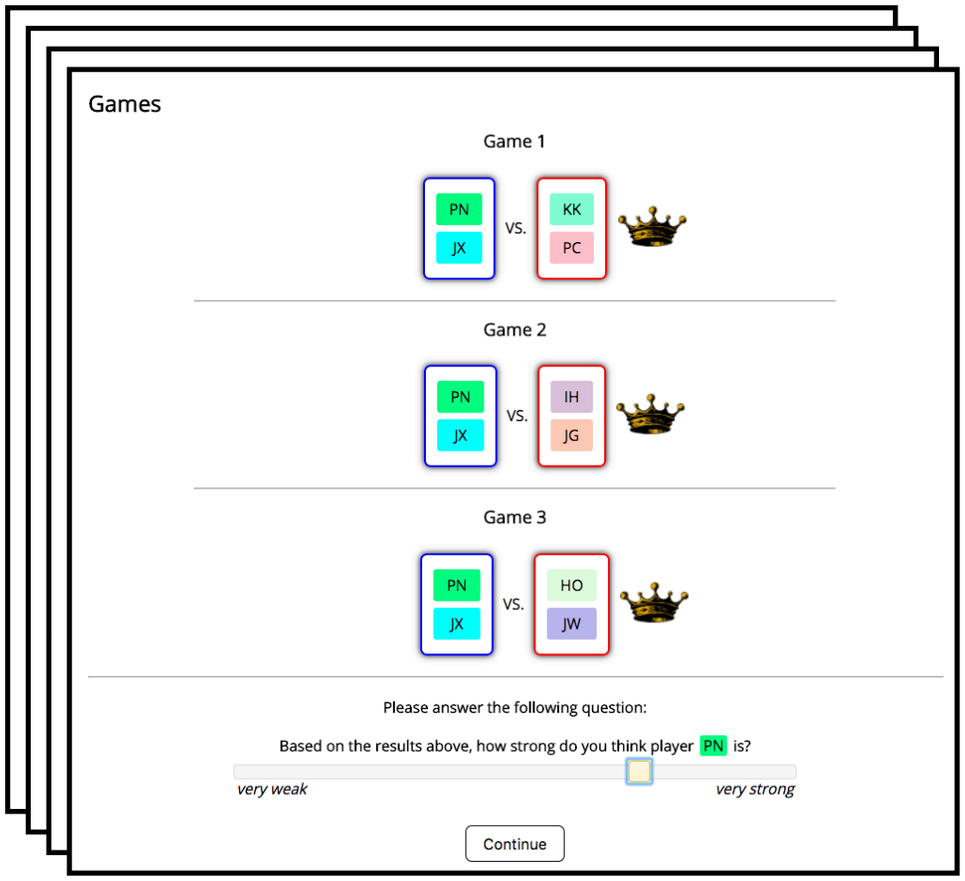
  var beat = function(team1,team2){winner(team1,team2) == team1}

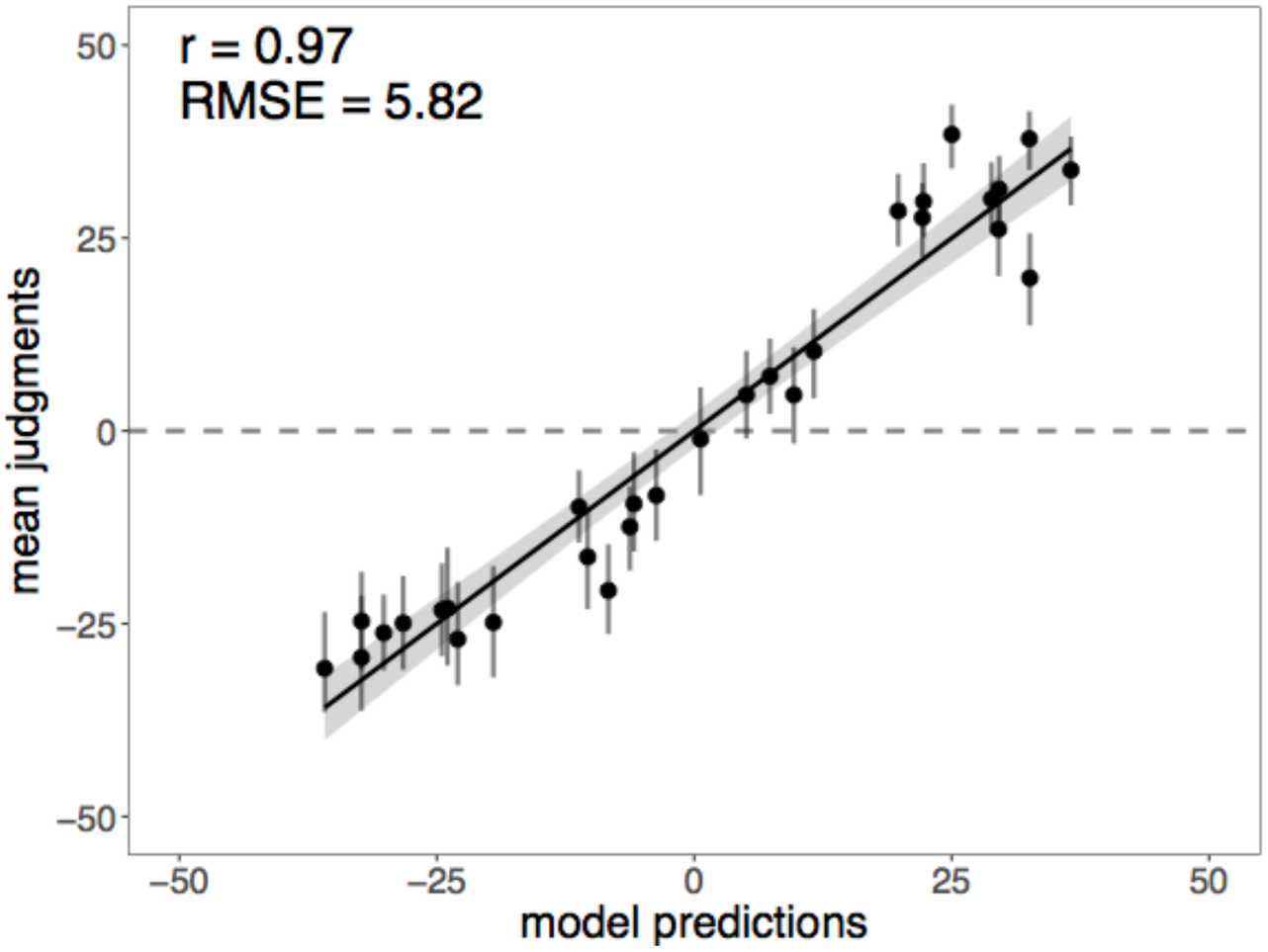
  condition(beat(['bob'], ['tom']))

  return strength('bob')
}
```

Experiment

How strong is Player X?





Gerstenberg & Goodman (2012) Ping Pong in Church: Productive use of concepts in human probabilistic inference. Cognitive Science Proceedings

Gerstenberg & Tenenbaum (2017) Intuitive Theories. Oxford Handbook of Causal Reasoning

Goodman, Tenenbaum, & Gerstenberg (2015) Concepts in a probabilistic language of thought. The Conceptual Mind: New Directions in the Study of Concepts

Outline

- How can a program model human thought? What kinds of programs?
- Probabilistic language of thought hypothesis
    - Compositionality
    - Productivity
    - Inference
- **Do it yourself!**
    1. WebPPL basics
    2. Building generative models
    3. Doing inference

How can a program model human thought?

Formally describe intelligent inferences in human reasoning

Can be applied to engineer intelligent machines

Computational theory of mind



mind = computer



mental representations =

computer programs

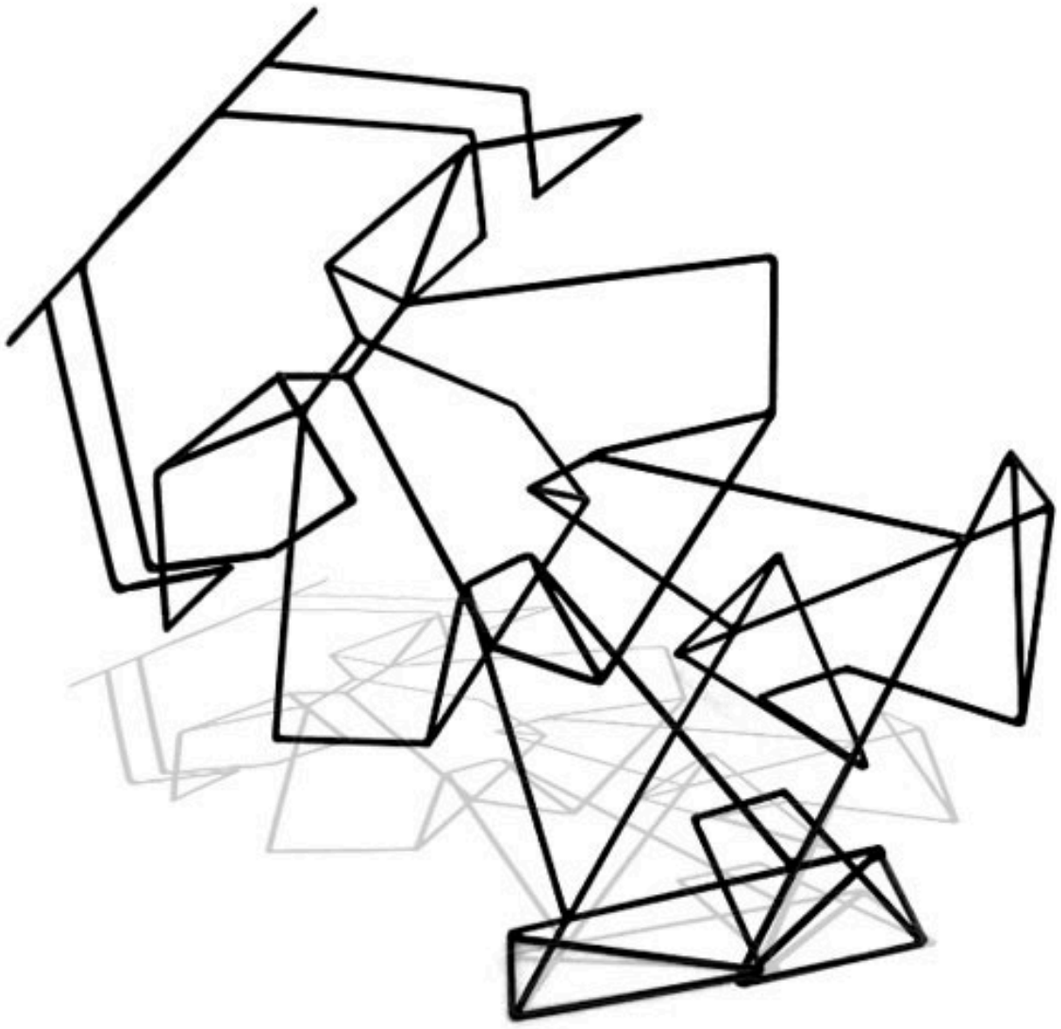


formal models of

thinking = programs

What kind of a program?

Structure



Probability

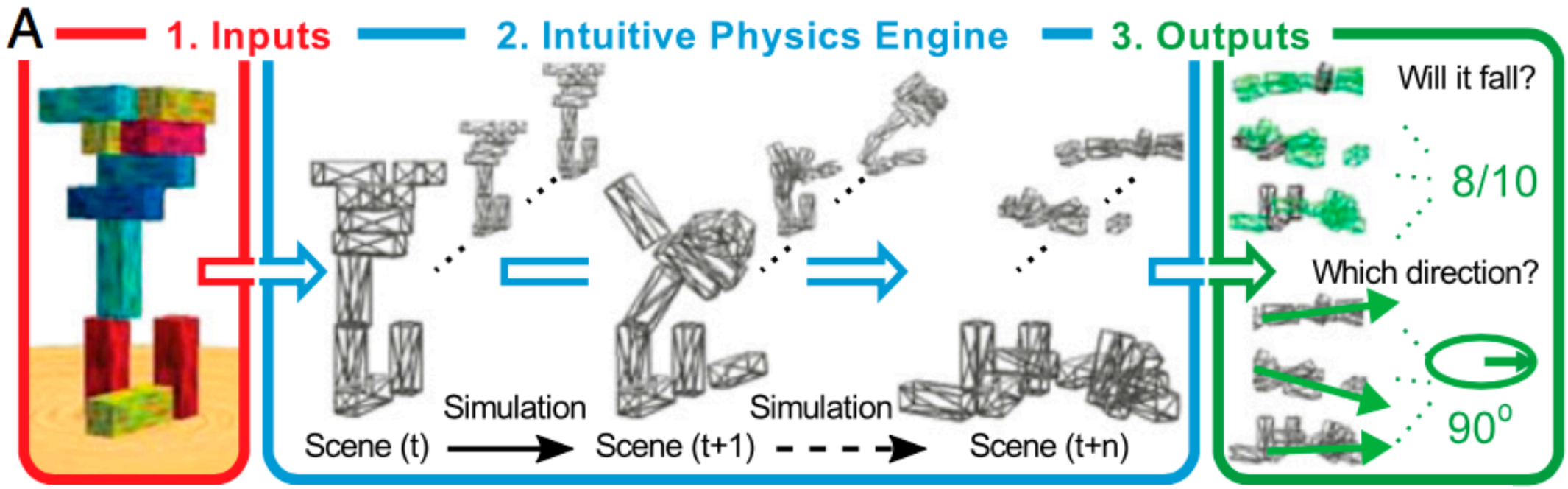


Knowledge

Uncertainty

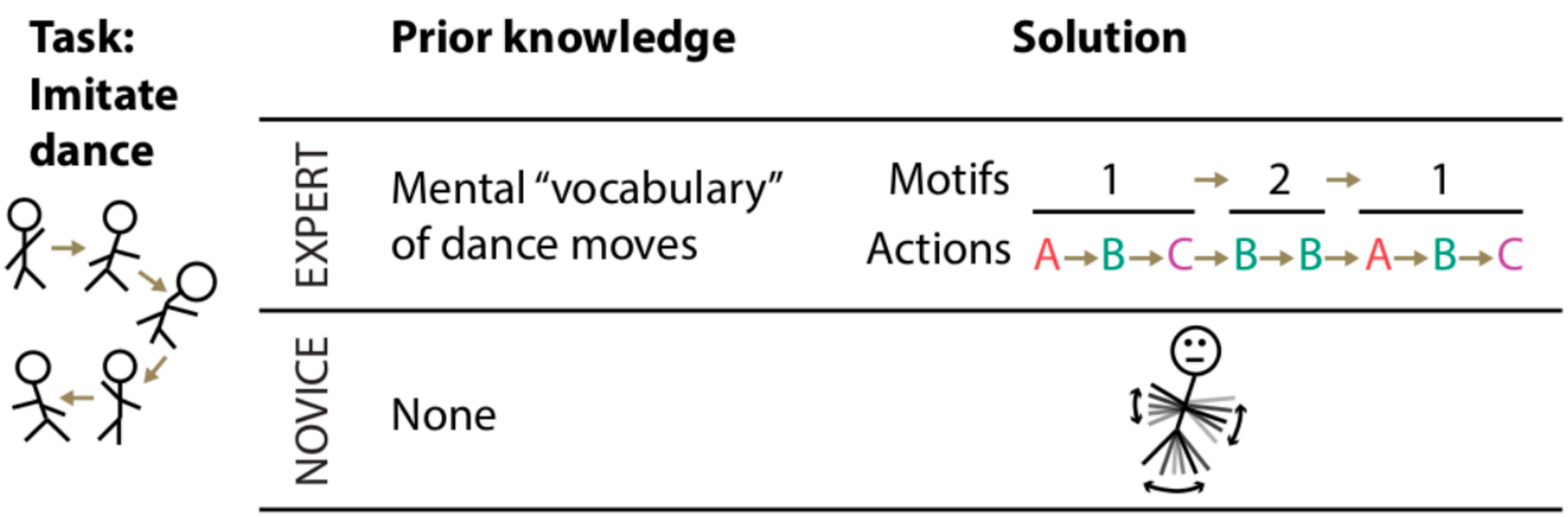
Tenenbaum, Kemp, Griffiths, & Goodman (2011) How to grow a mind: Statistics, structure, and abstraction. Science

Example of structure - knowledge of physics; Probability - stochastic simulations



Battaglia, Hamrick, & Tenenbaum (2013) Simulation as an engine of physical scene understanding. Proceedings of the National Academy of Sciences

Example of structure - motor programs in drawing; Probability - minimizing motor cost

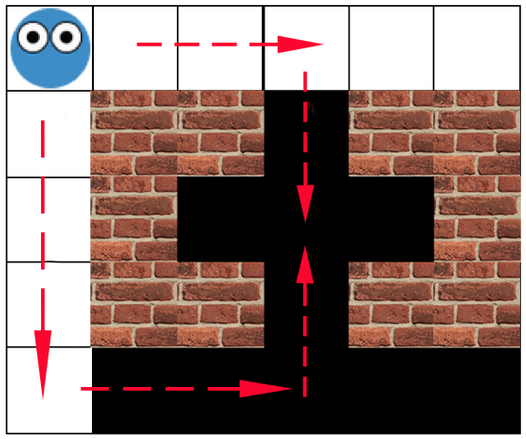


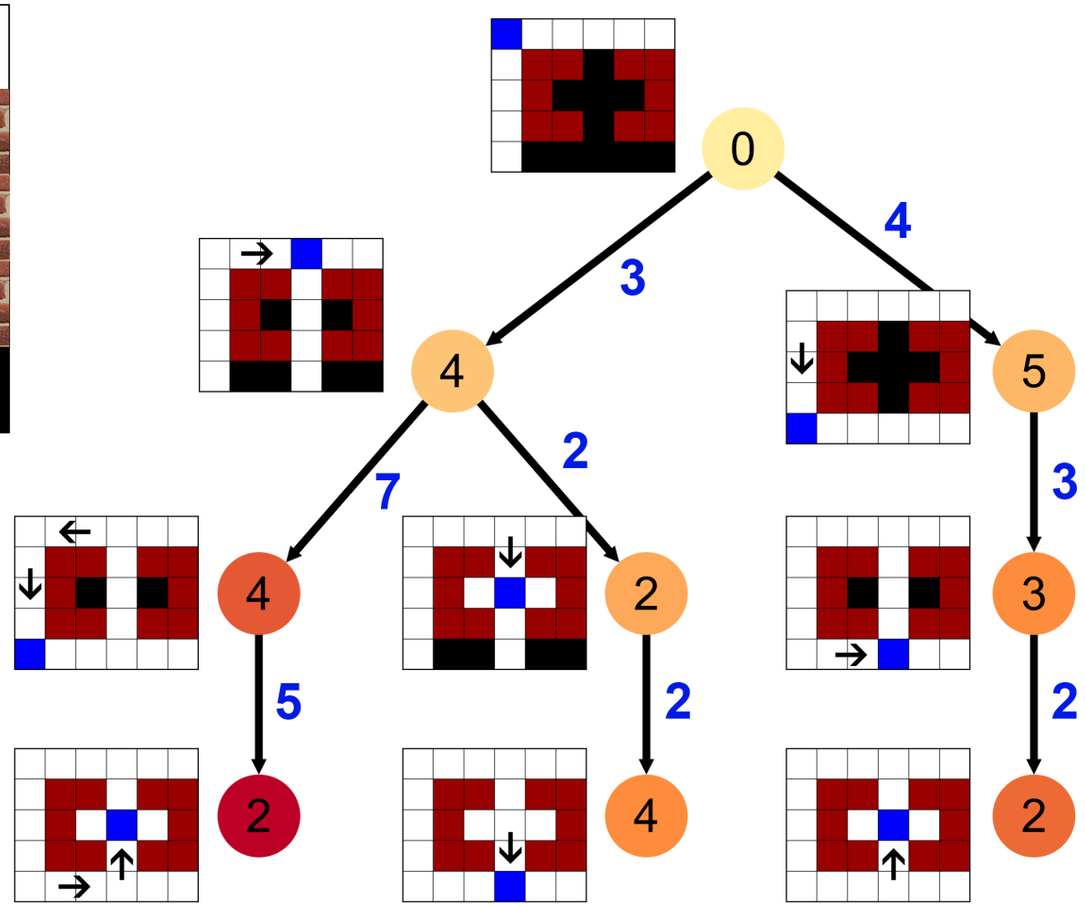
https://arxiv.org/abs/2008.03519

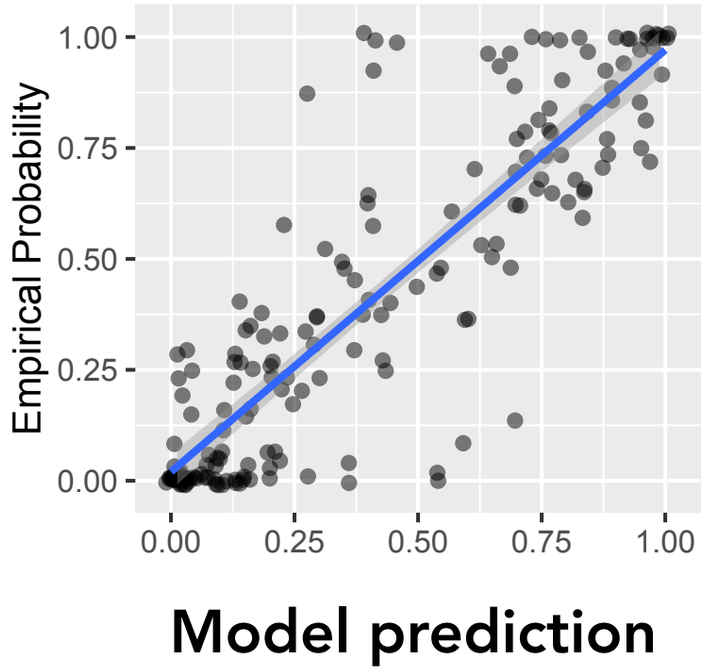
Learning abstract structure for drawing by efficient motor program induction

Lucas Y. Tian, Kevin Ellis, Marta Kryven, Joshua B. Tenenbaum

Example of structure - maps / representations of the environment; Probability - maximizing expected utility of actions







Kryven, M., Ullman, T., Cowan, W., Tenenbaum, J. B. Outcome or Strategy? A Bayesian Model of Intelligence Attribution, *Proceedings of the 38th Annual Meeting of the Cognitive Sciences Society*

Kryven, M., Ullman, T., Cowan, W., Tenenbaum, J. B. Thinking and Guessing: Model-Based and Empirical Analysis of Human Exploration, *Proceedings of the 39th Annual Meeting of the Cognitive Sciences Society*

Probabilistic language of thought hypothesis

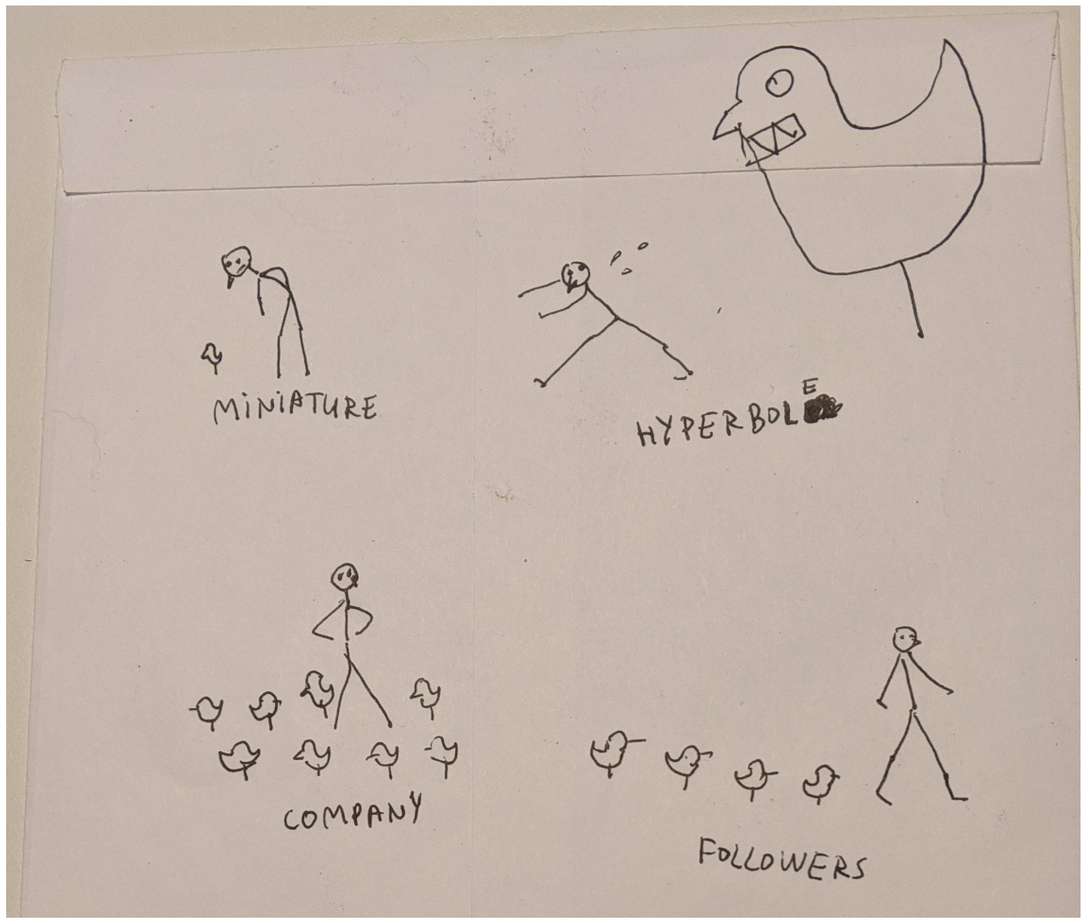
Informal version:

"Concepts have a language-like compositionality and encode probabilistic knowledge. These features allow them to be extended productively to new situations and support flexible reasoning and learning by probabilistic inference."

Fodor (1975). The language of thought

Probabilistic language of thought hypothesis

compositionality creative generation of complex actions by recombining relevant components of prior knowledge, in order to solve problems.



https://papers.nips.cc/paper/8006-learning-libraries-of-subroutines-for-neurallyguided-bayesian-program-induction
Learning Libraries of Subroutines for Neurally–Guided Bayesian Program Induction. NeurIPS 2018. Ellis, Morales, Solar-Lezama, Tenenbaum.
https://papers.nips.cc/paper/7845-learning-to-infer-graphics-programs-from-hand-drawn-images
Learning to Infer Graphics Programs from Hand-Drawn Images. NeurIPS 2018. Ellis, Ritchie, Solar-Lezama, Tenenbaum.

Probabilistic language of thought hypothesis

Productivity "infinite use of finite means" (Wilhelm von Humboldt)



Probabilistic language of thought hypothesis

Language limited number of letters, fixed set of grammatical rules

can express an infinite number of thoughts

Physics Formula Sheet

Mechanics

$x = x_0 + v_{x0}t + \frac{1}{2}a_x t^2$

$v = v_0 + at$

$v_x^2 - v_{x0}^2 = 2a(x - x_0)$

$\vec{a} = \frac{\Sigma \vec{F}}{m} = \frac{\vec{F}_{net}}{m}$

$|\vec{F}_{friction}| \leq \mu |\vec{F}_{Normal}|$

$\vec{p} = m\vec{v}$

$\Delta\vec{p} = \vec{F}\Delta t$

$KE = \frac{1}{2}mv^2$

$\Delta PE = mg\Delta y$

$\Delta E = W = Fd\cos\theta$

$a_c = \frac{v^2}{r}$

$\theta = \theta_0 + \omega_0 t + \frac{1}{2}\alpha t^2$

$\omega = \omega_0 + \alpha t$

$T = \frac{2\pi}{\omega} = \frac{1}{f}$

$\nu = f\lambda$

$x = A\cos(2\pi f t)$

$\vec{\alpha} = \frac{\Sigma \vec{\tau}}{I} = \frac{\vec{\tau}_{net}}{I}$

$\vec{\tau} = r \times F$

$L = I\,\omega$

$\Delta L = \tau \Delta t$

$|\vec{F}_{spring}| = k\,|\vec{x}|$

$PE_{spring} = \frac{1}{2}kx^2$

$T_{spring} = 2\pi\sqrt{\frac{m}{k}}$

$T_{pendulum} = 2\pi\sqrt{\frac{\ell}{g}}$

$|\vec{F}_{gravity}| = G\frac{m_1 m_2}{r^2}$

$|\vec{F}_{gravity}| = m\vec{g}$

$PE_{gravity} = -G\frac{m_1 m_2}{r}$

$\rho = \frac{m}{V}$

$KE = \frac{1}{2}I\,\omega^2$

Electricity

$|\vec{F}_E| = k\left|\frac{q_1 q_2}{r^2}\right|$

$\Delta V = IR$

$R = \frac{\rho \ell}{A}$

$I = \frac{\Delta q}{\Delta t}$

$P = I\Delta V$

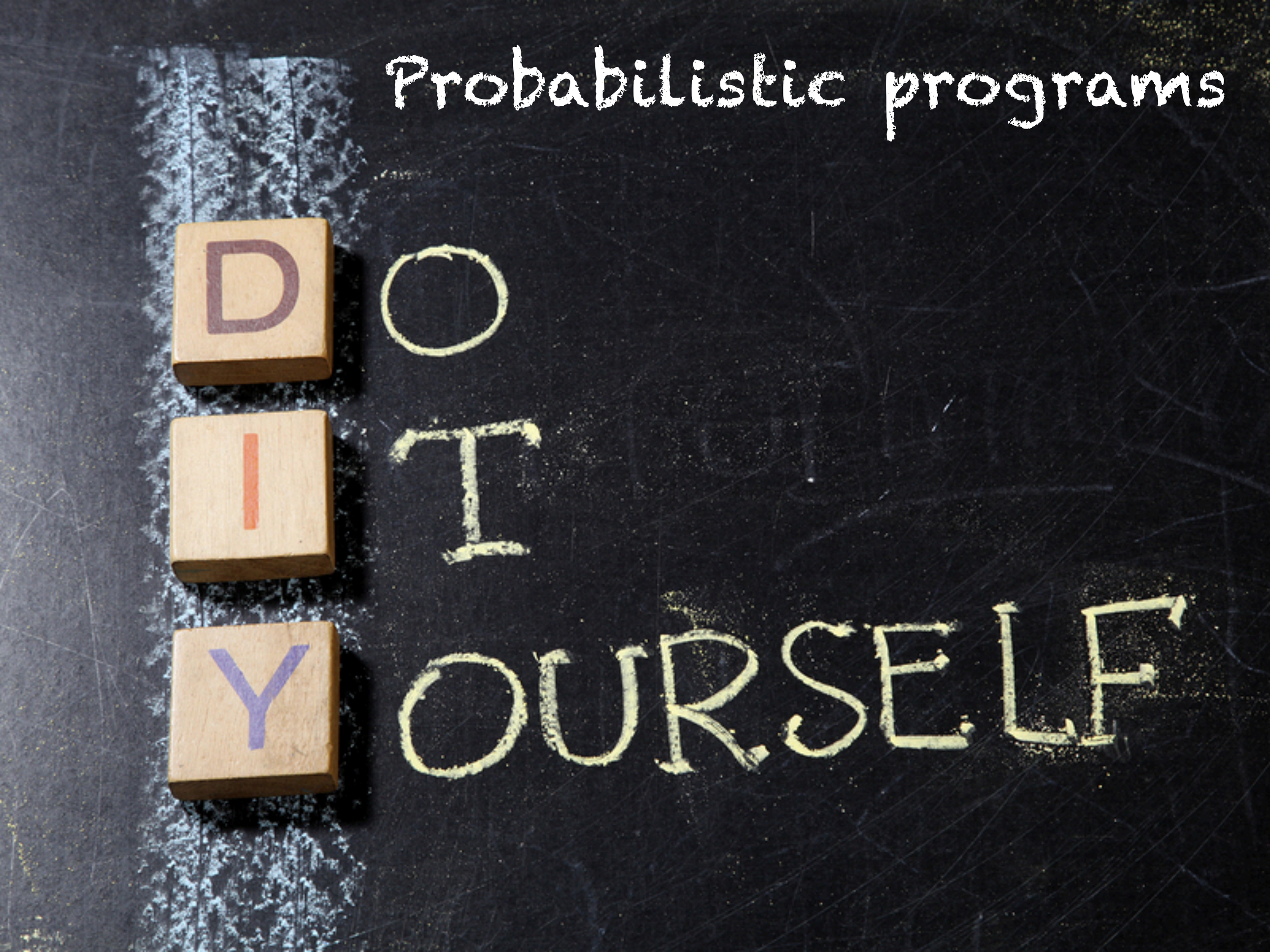
$R_{series} = R_1 + R_2 + \ldots + R_n$

$\frac{1}{R_{Parallel}} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_n}$

Variables

a = acceleration
A = amplitude
A = Area
b = base length
C = circumference
d = distance
E = energy
f = frequency
F = force
h = height
I = current
I = rotational inertia
KE = kinetic energy
k = spring constant
L = angular momentum
ℓ = length
m = mass
P = power
p = momentum
q = charge
r = radius
R = resistance
S = surface area
T = period
t = time
PE = potential energy
V = electric potential

Probabilistic programs



Why WebPPL?

WebPPL

- written by cognitive scientists for cognitive scientists
- goal: building computational models of cognition

- extremely flexible
- can be slow

other PPLs

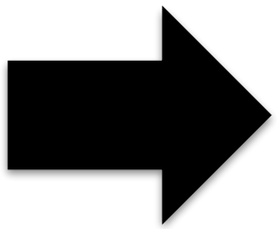
- Stan, Anglican, Alchemy, BUGS, Edward, PyMC
- goal: building rich models for data analysis

- flexible but limited
- faster (for the models that can be expressed)

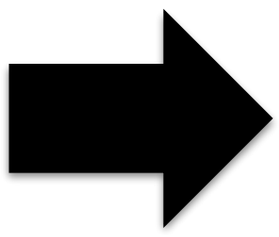
Recent PPLs (Pyro, Gen) have a steeper learning curve, require more thought about inference, but keep the flexibility of WebPPL and work faster

1. WebPPL basics
2. Building generative models
3. Doing inference

http://bit.do/webppl



notes/1_webppl_basics.md



http://webppl.org

1. WebPPL basics

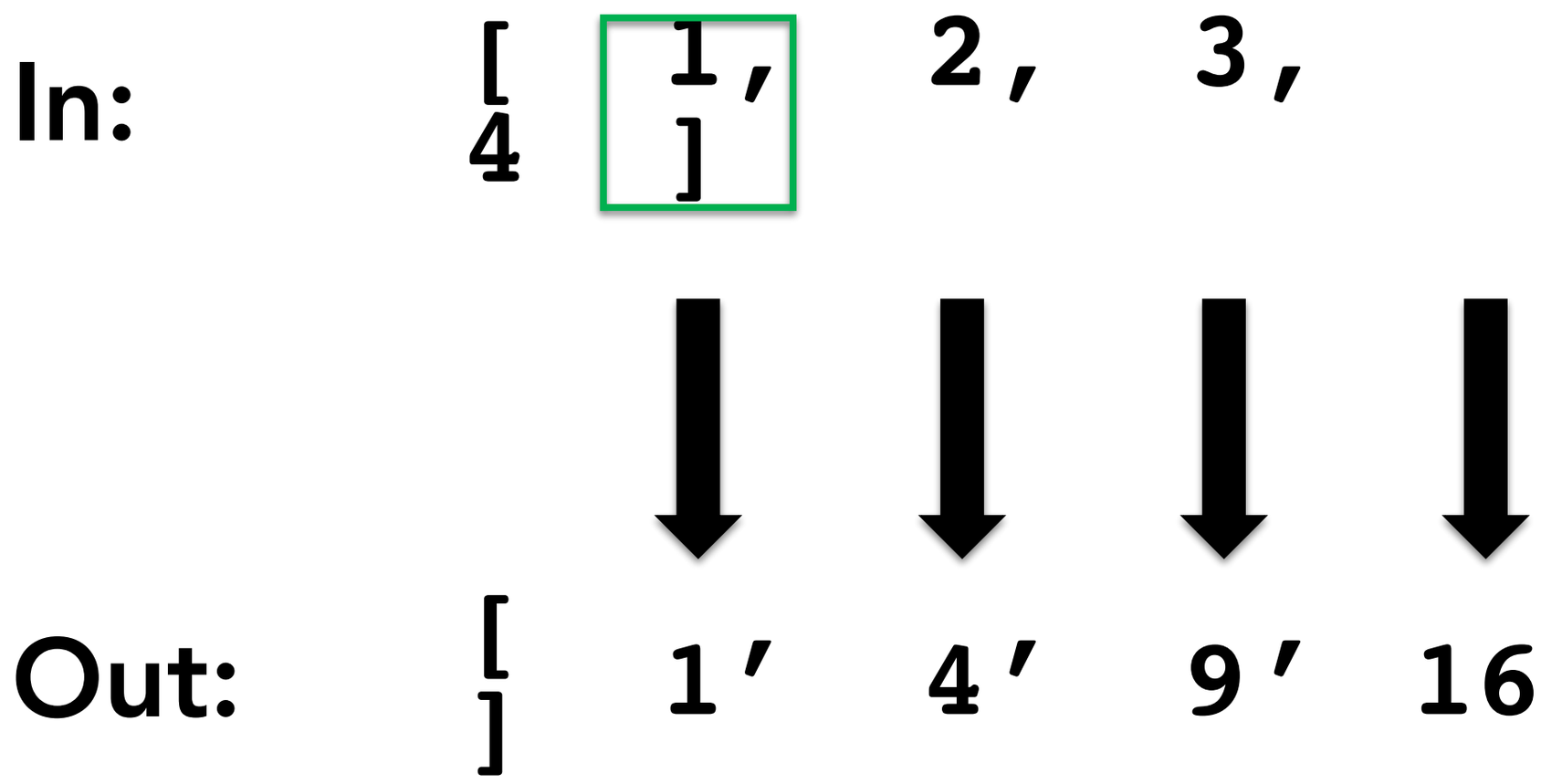
WebPPL basics

- declare variables
- data formats: numbers, strings, logicals, objects, arrays
- if-then-statements
- defining functions
- higher order functions
  - `map`

WebPPL basics

- The `map` function

```
var mySquare = function(x){return x*x}
var someNumbers = [1,2,3,4]
display(map(mySquare, someNumbers))
```



WebPPL basics



- WebPPL = purely functional programming language
  - **can't** write `for` loops or `while` statements
  - **can** create higher-order functions and recursive functions
    - for example: `map` = apply a function to each element of a list (like a `for` loop)

2. Building generative models

Building generative models

- forward sampling and random primitives
- building simple models
- sample from probability distributions
- memoization: `mem`
- recursive functions

Building generative models



- WebPPL is a language to formally describe how the world works
- random choices capture our uncertainty or ignorance
- the language is **universal**: it can express any computable process
- causal dependence is important: the program describes what influences what

Stuhlmüller, Tenenbaum, & Goodman (2010) Learning Structured Generative Concepts. Cognitive Science Proceedings

Stuhlmüller & Goodman (2014) Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. Cognitive Systems Research

Building generative models



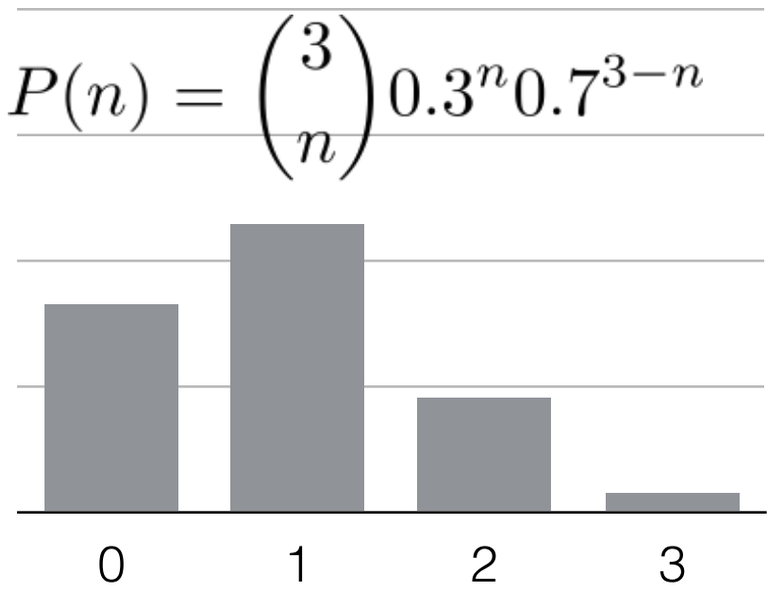
relationship between sampling and probability distributions

Random primitives:

```
var a = flip(0.3)
var b = flip(0.3)
var c = flip(0.3)
return a + b + c
```

```
=> 1 0 0
=> 0 0 0
=> 1 0 1
=> 2 0 1
```

...



Sampling $\cong$ Distributions

any computable distribution can be represented as the distribution induced by sampling from a probabilistic program

Goodman, Mansinghka, Roy, Bonawitz, & Tenenbaum (2008) Church: A language for generative models. Uncertainty in Artificial Intelligence

3. Doing inference

Doing inference

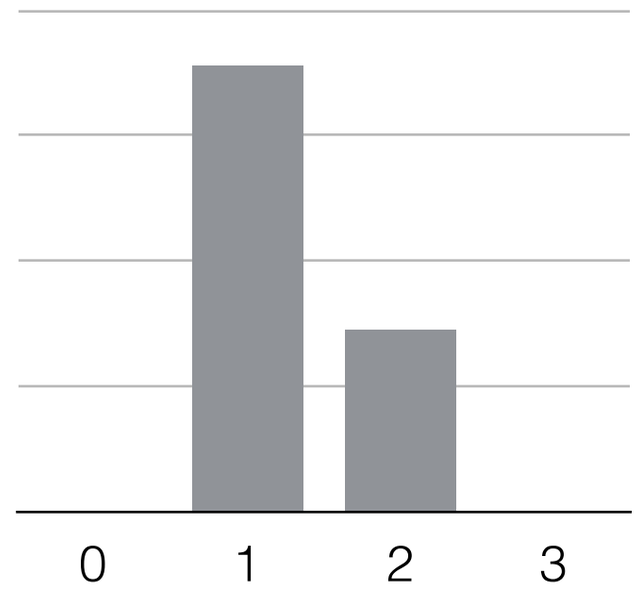
Conditional inference:

```
Infer(
 function(){
  var a = flip(0.3)
  var b = flip(0.3)
  var c = flip(0.3)
  condition(a + b == 1)
  return a + b + c})
```

```
=> 1 0 0 1
=> 0 0 0 0
=> 1 0 1 0
=> T F F T
=> 2 0 1 1
```

probability / frequency

Posterior distribution



"when you have excluded the impossible, whatever remains, however improbable, must be the truth."



Doing inference



- conditioning on variables
  - rejection sampling
  - WebPPL's inference procedures
- conditioning on arbitrary expressions
- other inference procedures

- `forward`, `rejection`, `enumerate`, `MH`, ...

- you don't have to worry about inference. **very nice!**
- WebPPL allows us to parsimoniously describe rich generative model structures and explore the inference patterns that emerge from the interaction of model and data

"Concepts have a language-like **compositionality** and encode probabilistic knowledge. These features allow them to be extended **productively** to new situations and support flexible reasoning and learning by **probabilistic inference**."

```
var towModel = function() {
  var strength = mem(function (person) {return gaussian(50, 10)})

  var lazy = function(person) {return flip(0.1) }

  var pulling = function(person) {
    return lazy(person) ? strength(person) / 2 : strength(person) }

  var totalPulling = function (team) {return sum(map(pulling, team))}

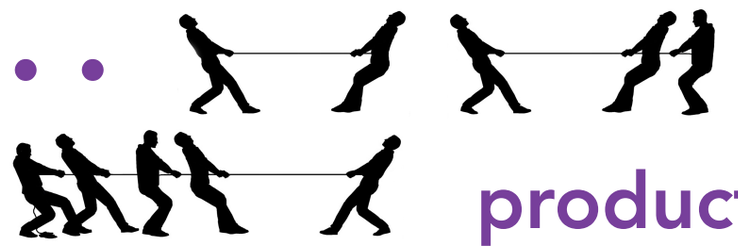
  var winner = function (team1, team2) {
    totalPulling(team1) > totalPulling(team2) ? team1 : team2 }

  var beat = function(team1,team2){winner(team1,team2) == team1}

  condition(beat(['bob'], ['tom']))

  return strength('bob')
}
```

compositionality



productivity

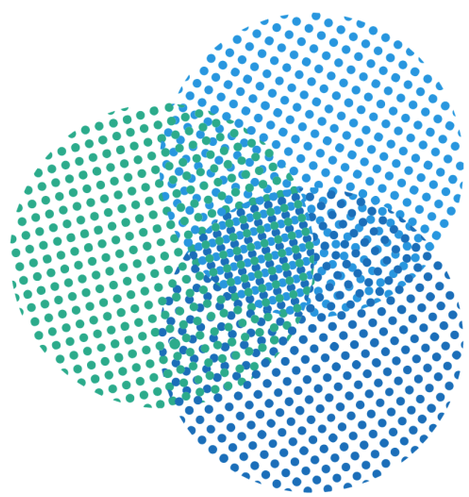
Resources: theory

- Goodman, Mansinghka, Roy, Bonawitz, & Tenenbaum (2008) Church: A language for generative models. Uncertainty in Artificial Intelligence
  - ➡ https://stanford.edu/~ngoodman/papers/churchUAI08_rev2.pdf
- Goodman, Tenenbaum, & Gerstenberg (2015) Concepts in a probabilistic language of thought. The Conceptual Mind: New Directions in the Study of Concepts
  - ➡ web.mit.edu/tger/www/papers/Concepts in a probabilistic language of thought (Goodman, Tenenbaum, Gerstenberg, 2014).pdf
- Freer, Roy, & Tenenbaum (2012) Towards common-sense reasoning via conditional simulation: legacies of Turing in Artificial Intelligence. arXiv preprint arXiv:1212.4799
  - ➡ https://arxiv.org/pdf/1212.4799.pdf
- Lake, Ullman, Tenenbaum, & Gershman (2016) Building machines that learn and think like people. arXiv preprint arXiv:1604.00289
  - ➡ https://arxiv.org/pdf/1604.00289.pdf
- Gerstenberg & Tenenbaum (2017) Intuitive Theories. Oxford Handbook of Causal Reasoning
  - ➡ web.mit.edu/tger/www/papers/Intuitive Theories, Gerstenberg, Tenenbaum, 2017.pdf
- Chater & Oaksford (2013) Programs as causal models: Speculations on mental programs and mental representation. Cognitive Science
  - ➡ http://onlinelibrary.wiley.com/doi/10.1111/cogs.12062/abstract
- Ghahramani (2015) Probabilistic machine learning and artificial intelligence. Nature
  - ➡ http://www.nature.com/nature/journal/v521/n7553/full/nature14541.html

Resources: practice

- https://probmods.org/
  - many more cool chapters to play around with
- http://webppl.org/
  - Editor to play around with code
- http://dippl.org/
  - Details about WebPPL
- https://github.com/probmods/webppl
  - github repository with latest developments
- http://webppl.readthedocs.io/en/master/
  - function reference for the webppl language
- http://agentmodels.org/
  - Great web book that focuses on how to model agents and inferences about agents
- http://probabilistic-programming.org/wiki/Home
  - homepage comparing different probabilistic programming languages

Thanks!



CENTER FOR
Brains
Minds+
Machines