

# Tutorial



## Mental models as probabilistic programs



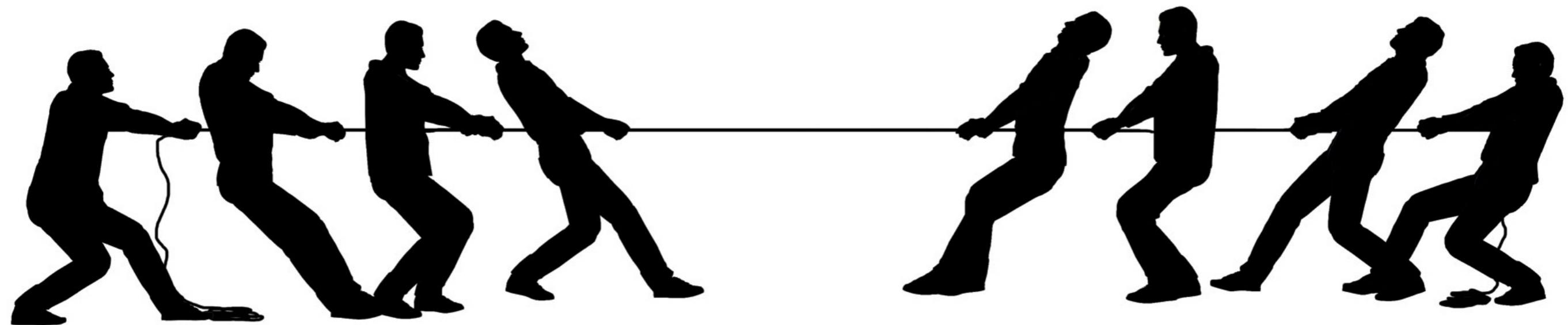
Kevin Smith

Tobias Gerstenberg



August 12, 2021

# Tug of War





Tom



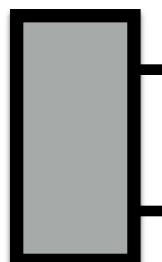
Steve



Tom



Steve

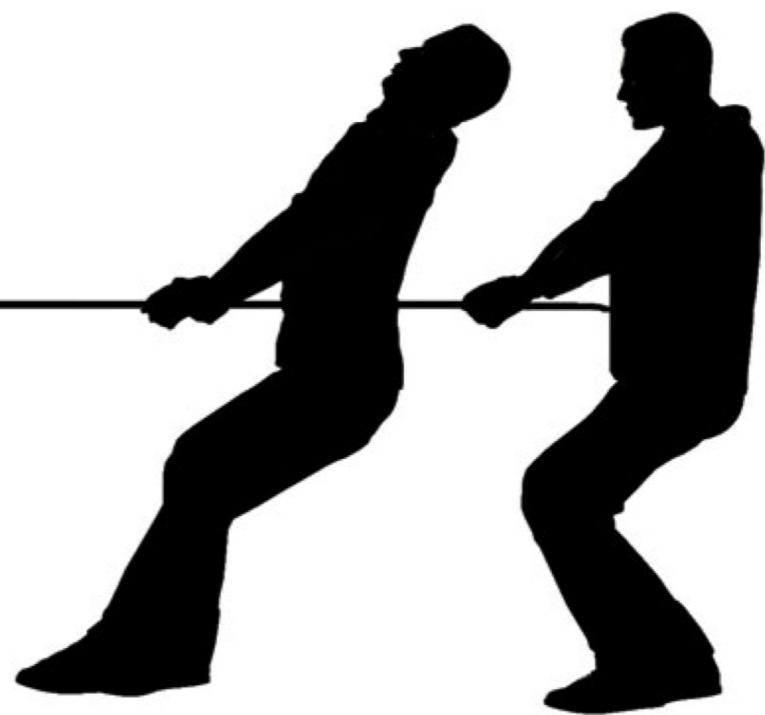


very weak

very strong

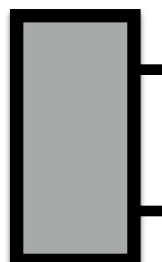


Bill



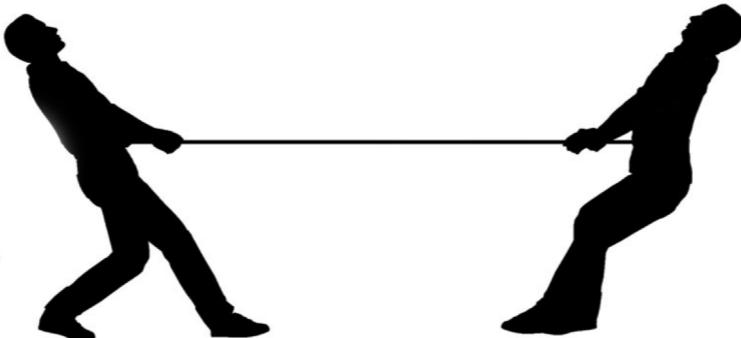
Mark

Nick



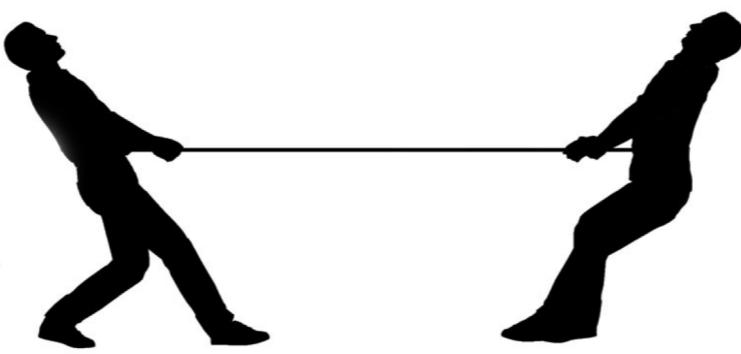
very weak

very strong



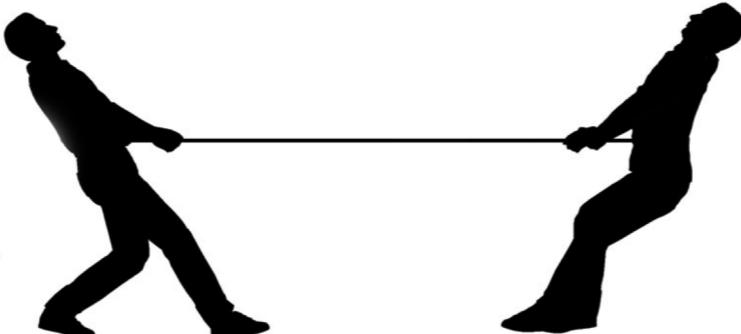
Tom

Steve



Bill

Steve



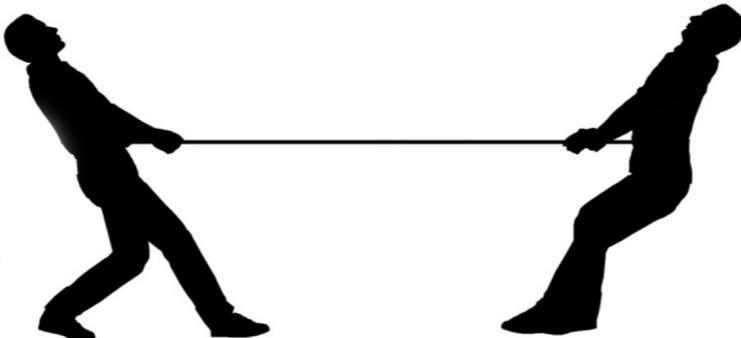
Ryan

Steve



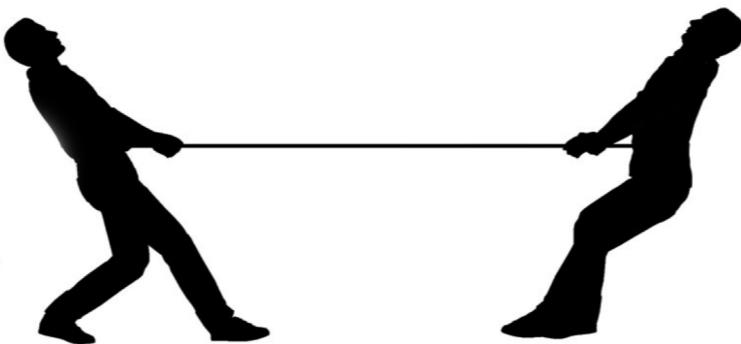
very weak

very strong



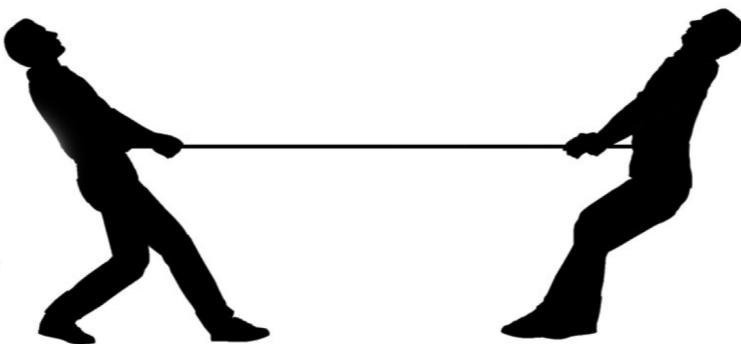
Tom

Steve



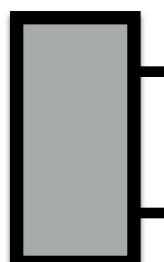
Bill

Steve



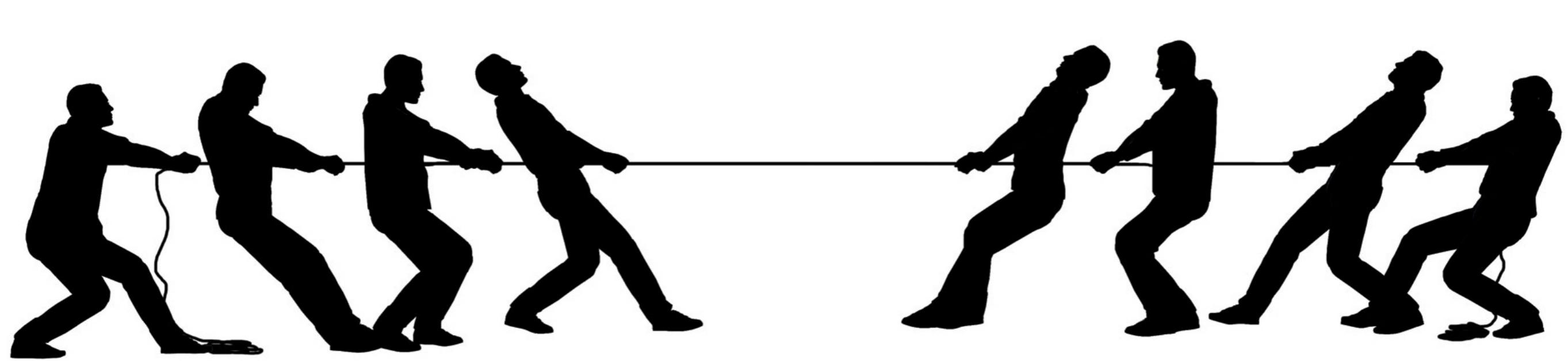
Ryan

Steve



very weak

very strong



## Concepts:

teams  
person  
strength  
effort  
pulling  
winner

# A computational model of tug of war

```
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?

Games

Game 1  
PN vs. KK, PC

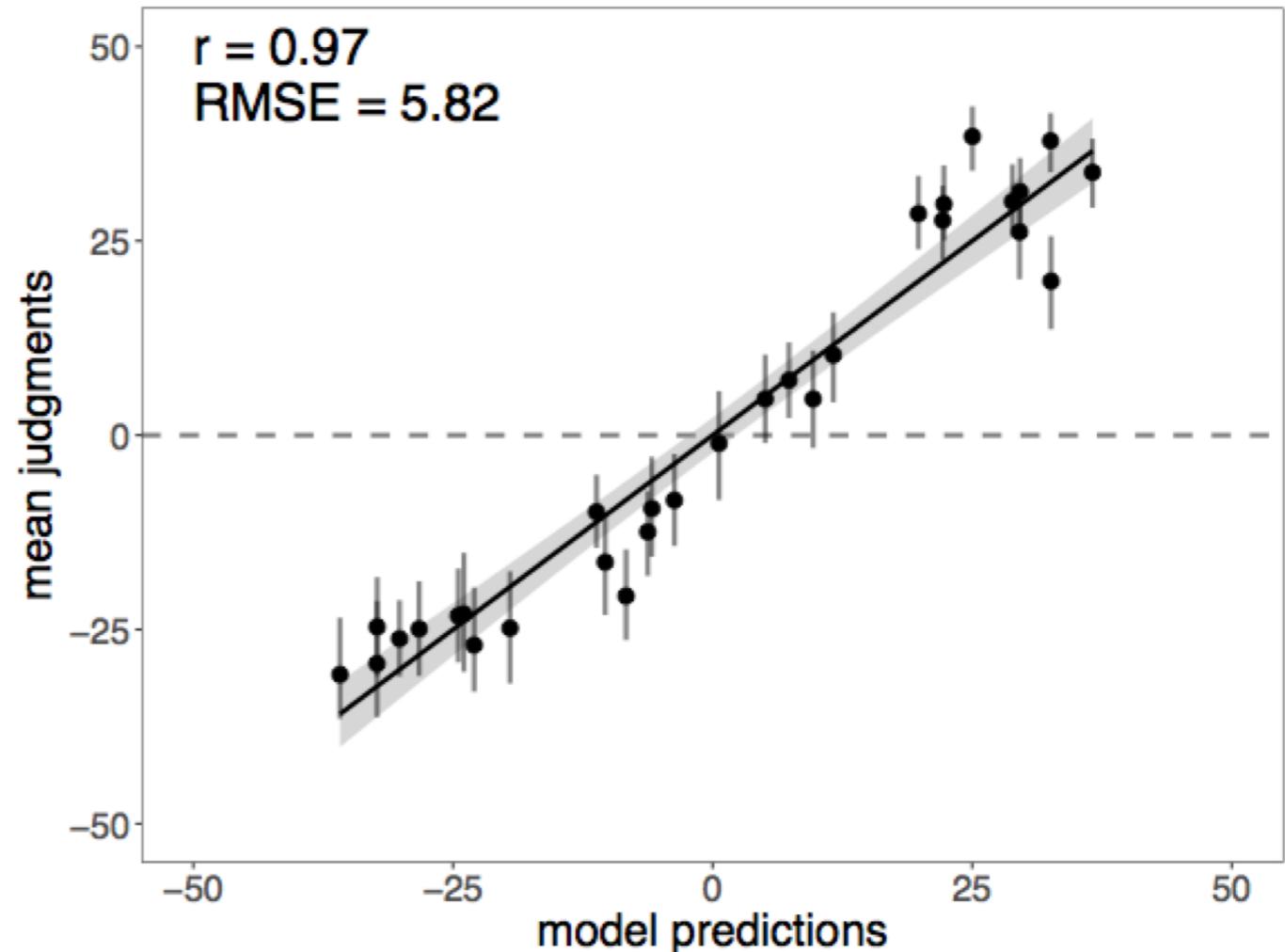
Game 2  
PN vs. IH, JG

Game 3  
PN vs. HO, JW

Please answer the following question:  
Based on the results above, how strong do you think player PN is?

very weak  very strong

Continue



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*

# Experiment

Did player X try hard in Game 1?

**Games**

Game 1  
Kevin VS. Jason 

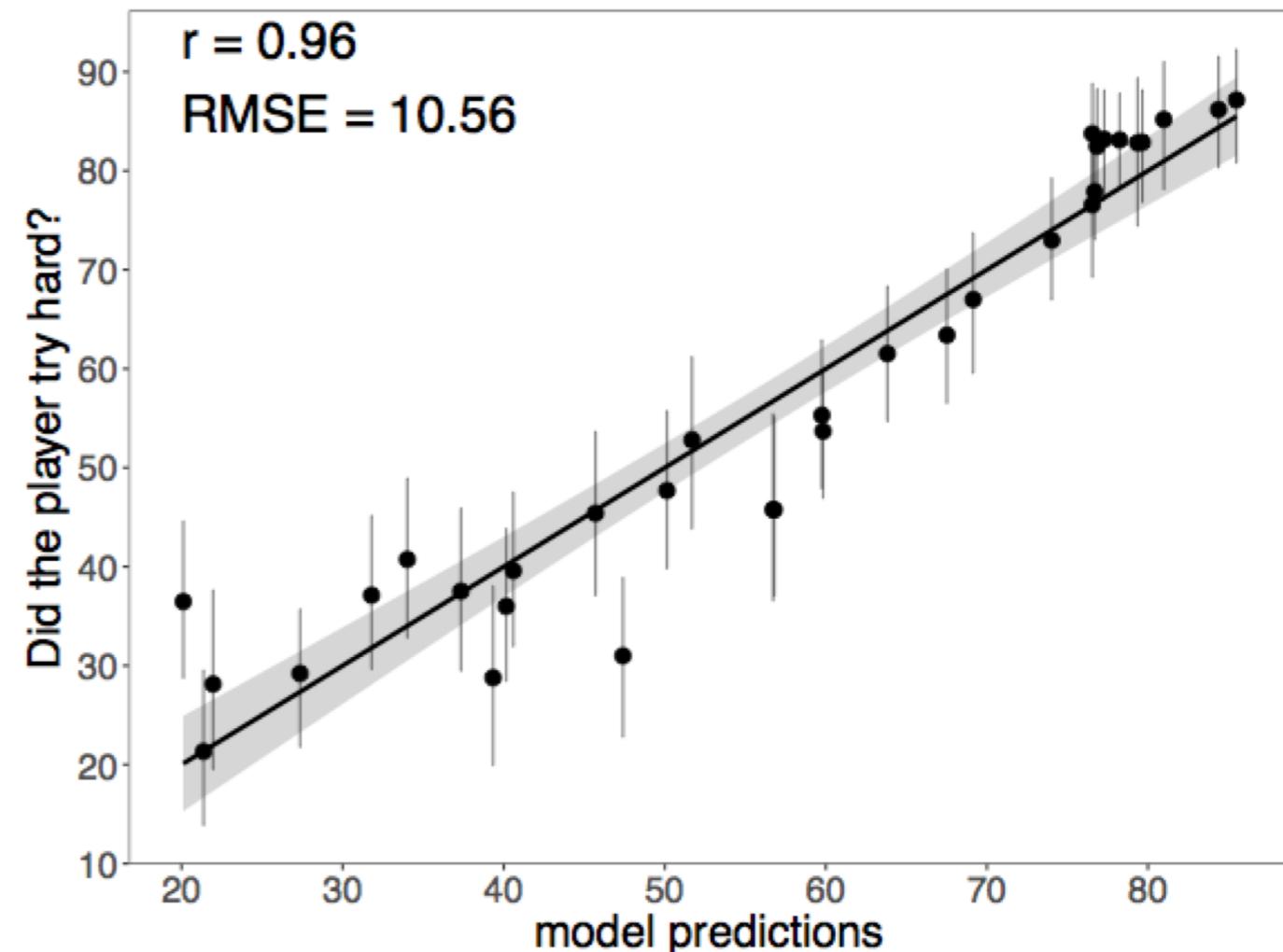
Game 2  
 Kevin VS. Jason

Game 3  
 Kevin VS. Jason

Please answer the following question:

How likely is it that Jason tried hard in Game 1?

very unlikely  very likely



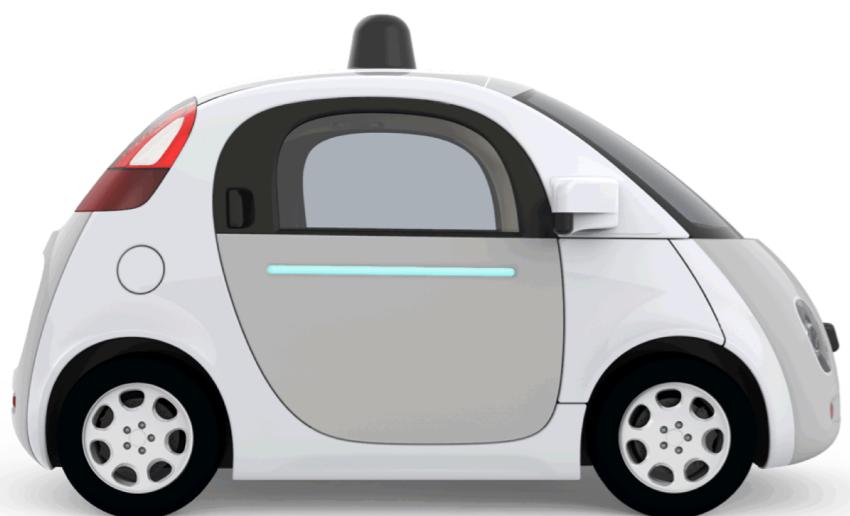
# Outline

- What is thinking?
  - What kind of a program?
  - Probabilistic language of thought hypothesis
    - Compositionality
    - Productivity
    - Inference
- **Do it yourself!**
  1. WebPPL basics
  2. Building generative models
  3. Doing inference
- Questions, Resources, some cool examples

# What is thinking?



How can we describe the intelligent  
inferences made in everyday human  
reasoning and learning?



How can we engineer  
intelligent machines?

# What is thinking?

# Computational theory of mind



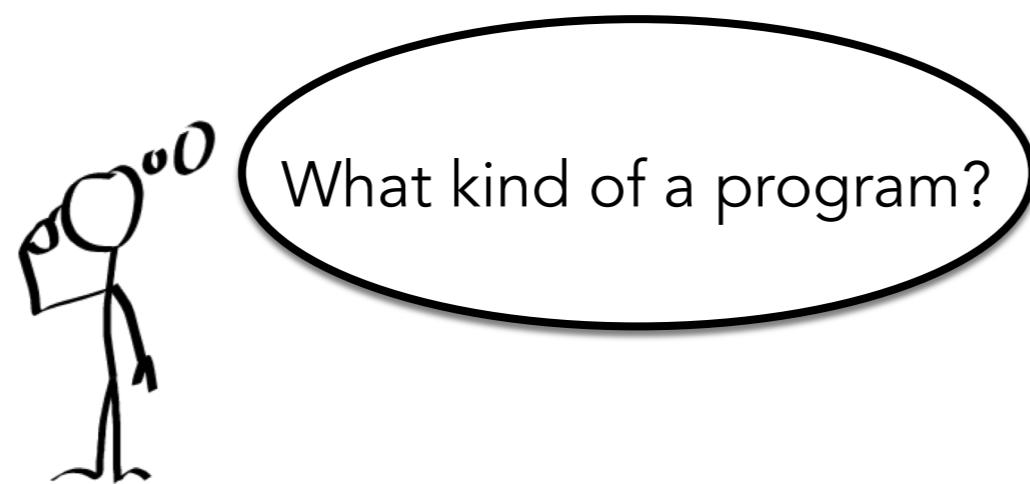
mind = computer

mental representations =  
computer programs

# run(program)

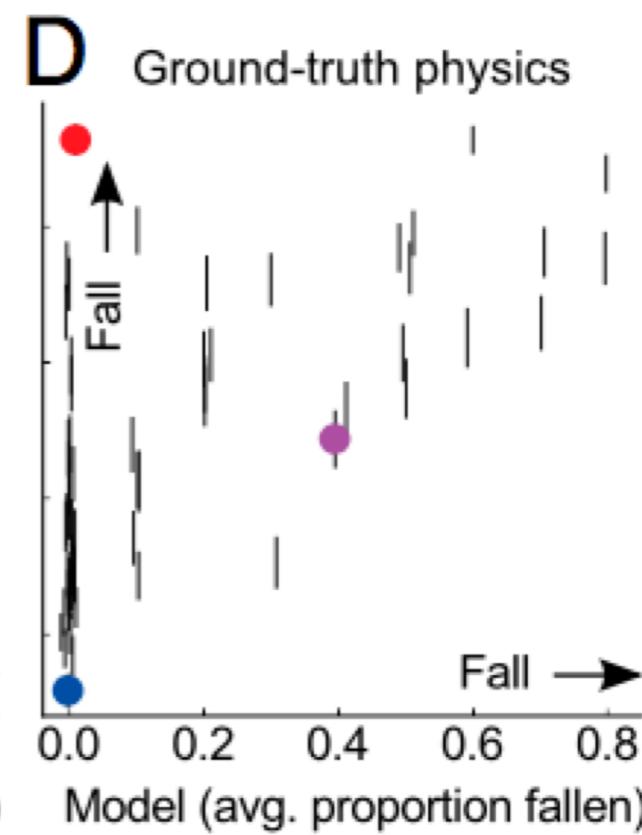
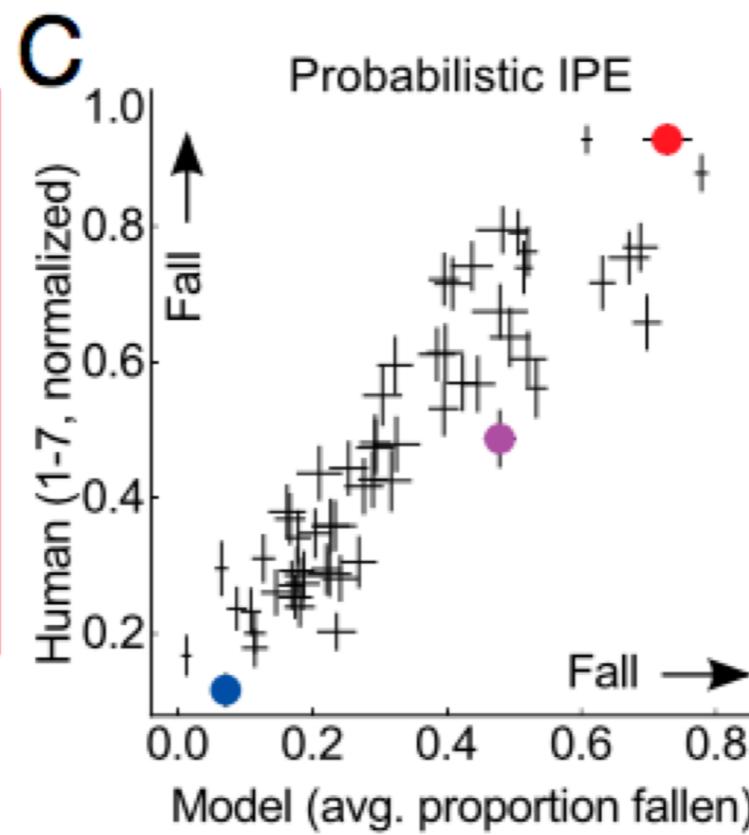
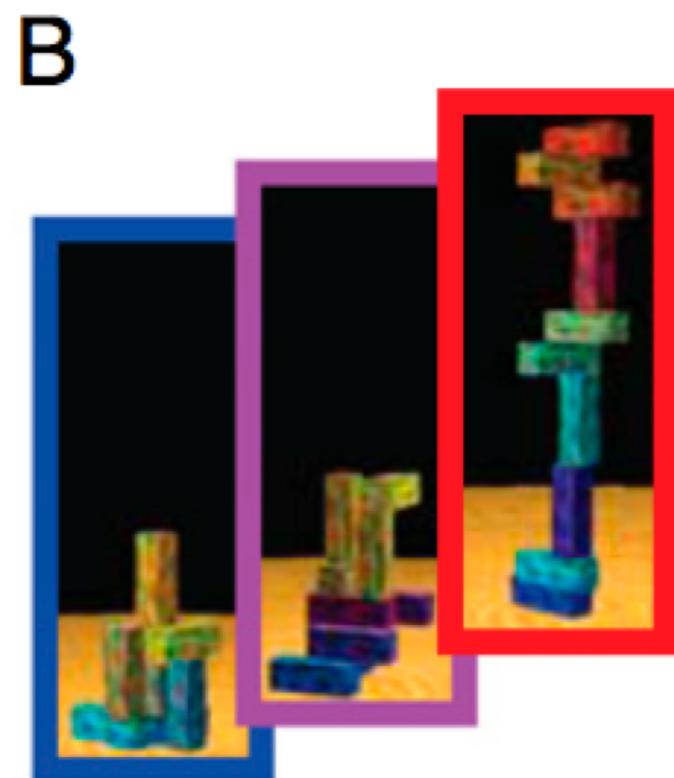
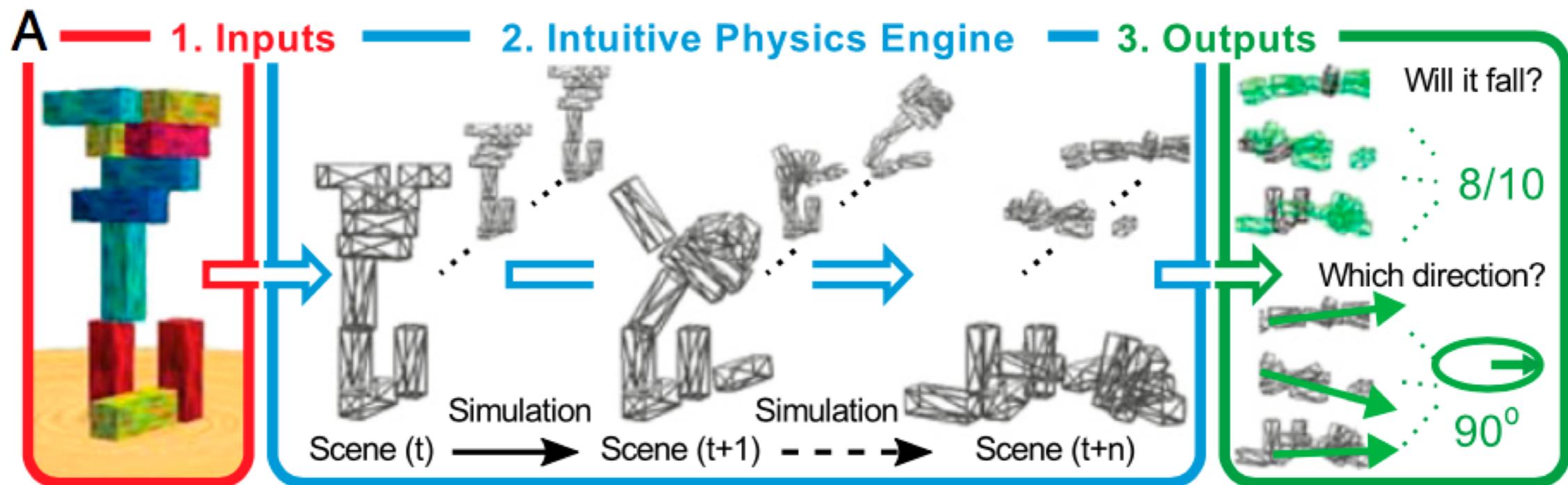
thinking =

# running a program



# What kind of a program?

## Working models of a domain

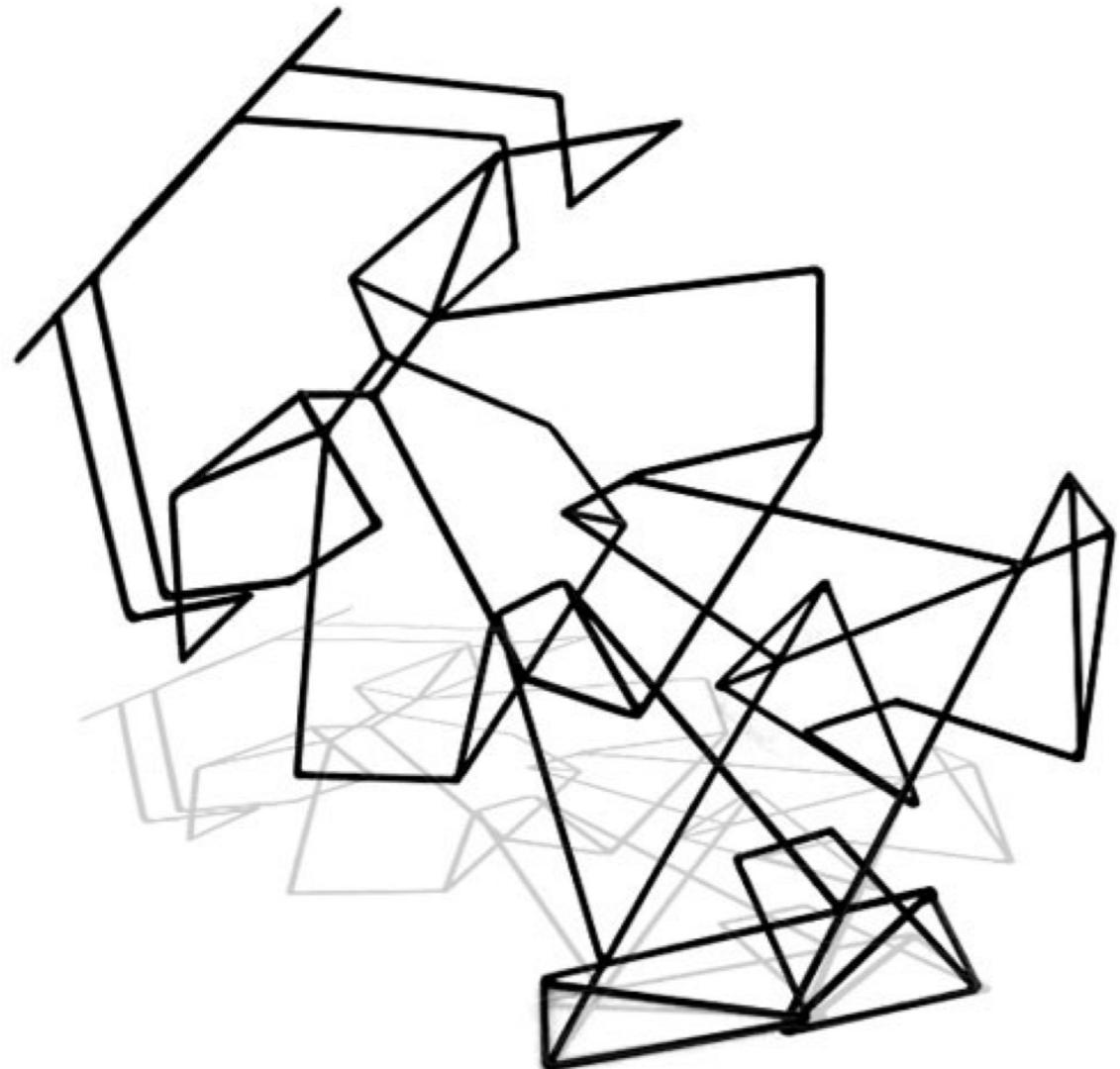


Battaglia, Hamrick, & Tenenbaum (2013) Simulation as an engine of physical scene understanding.

Proceedings of the National Academy of Sciences

# What kind of a program?

Structure



Probability



**Knowledge**

**Uncertainty**

# 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

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

# 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**."

## Compositionality

"A giraffe with  
a blue ribbon  
around its neck  
eating ice  
cream"



## Productivity

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

## Language

limited number of letters  
fixed set of grammatical rules

can express an infinite number of thoughts

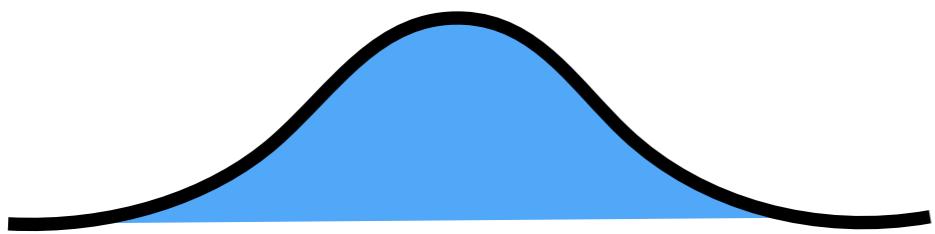
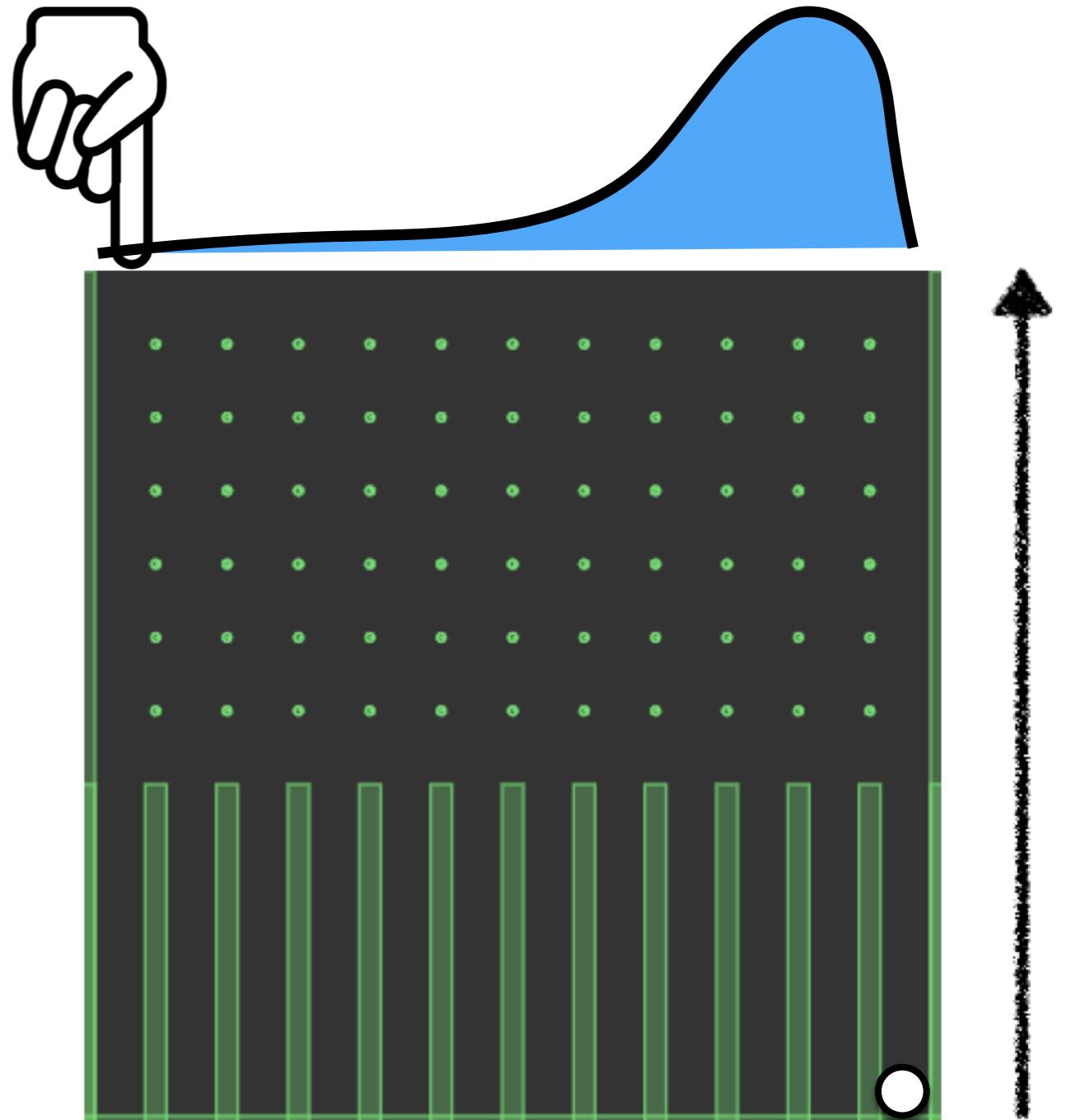
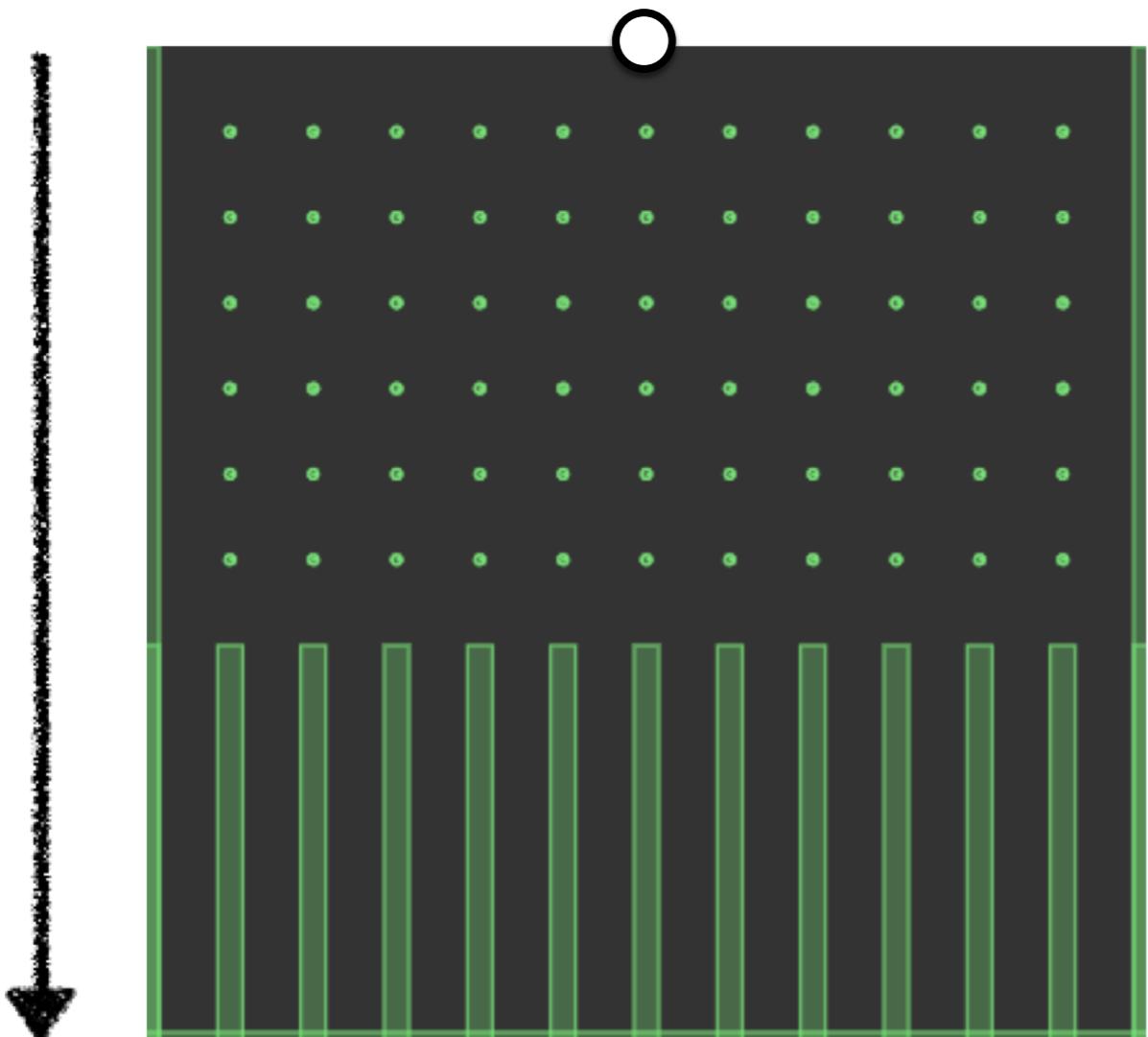
Fodor (1975). The language of thought

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

# Probabilistic inference

Run forward

Where will the ball land?



Reason backward  
Where did the ball come from?

# Probabilistic programs

D  
I  
Y  
OURSELF

# Why WebPPL?

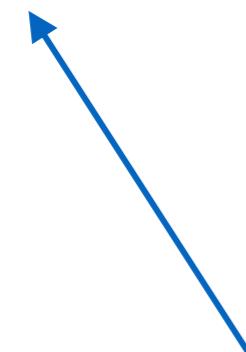
## WebPPL

## other PPLs

- written by cognitive scientists for cognitive scientists
- goal: building computational models of cognition
- extremely flexible
- can be slow

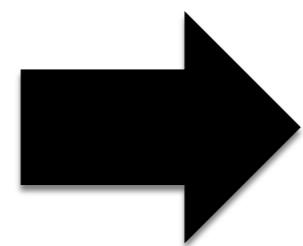
- Stan, Anglican, Alchemy, BUGS, Edward, PyMC
- goal: building rich models for data analysis
- flexible but limited
- faster (for the models that can be expressed)

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

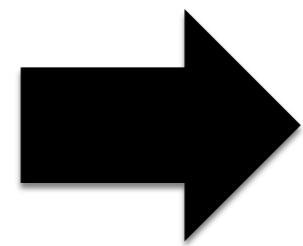


1. WebPPL basics
2. Building generative models
3. Doing inference

[https://bit.ly/bmm\\_webppl](https://bit.ly/bmm_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

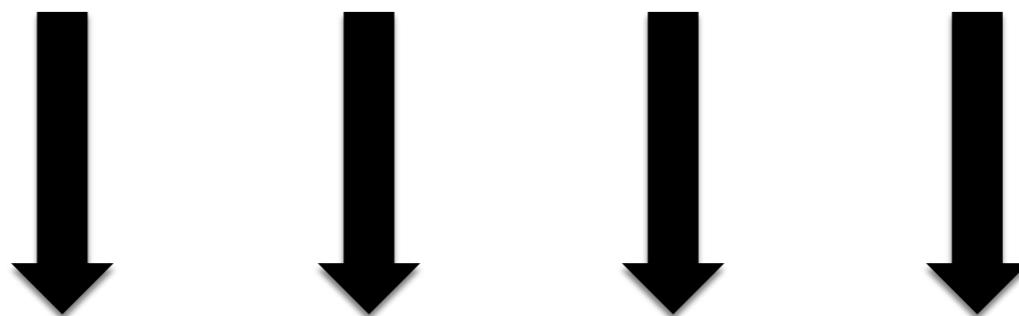
# WebPPL basics

- The `map` function

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

In:

[ 1,    2,    3,    4    ]



Out:

[    1,    4,    9,    16    ]

# 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

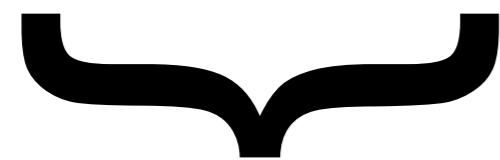
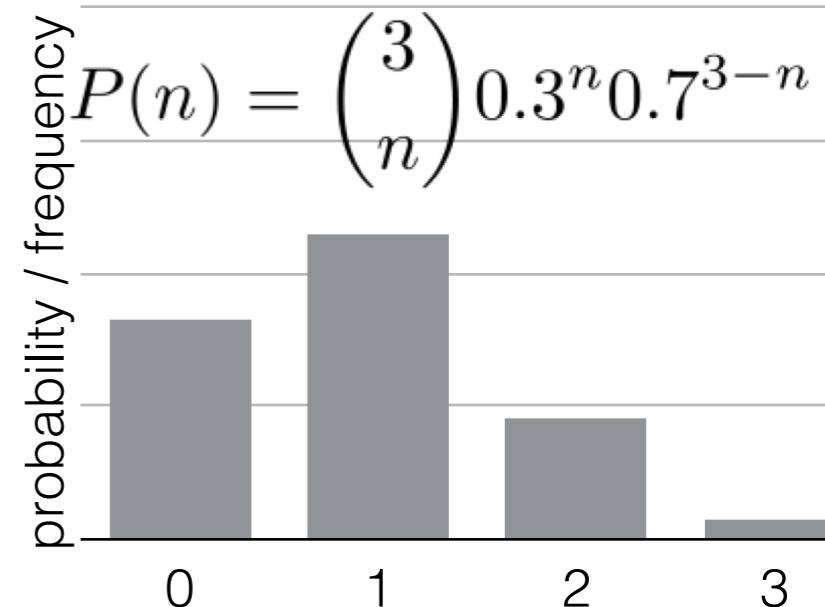


relationship between sampling and probability distributions

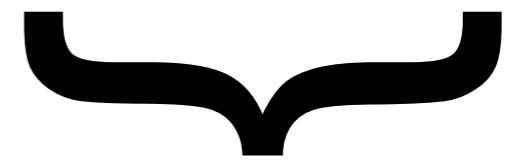
Random primitives:

```
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



Distributions

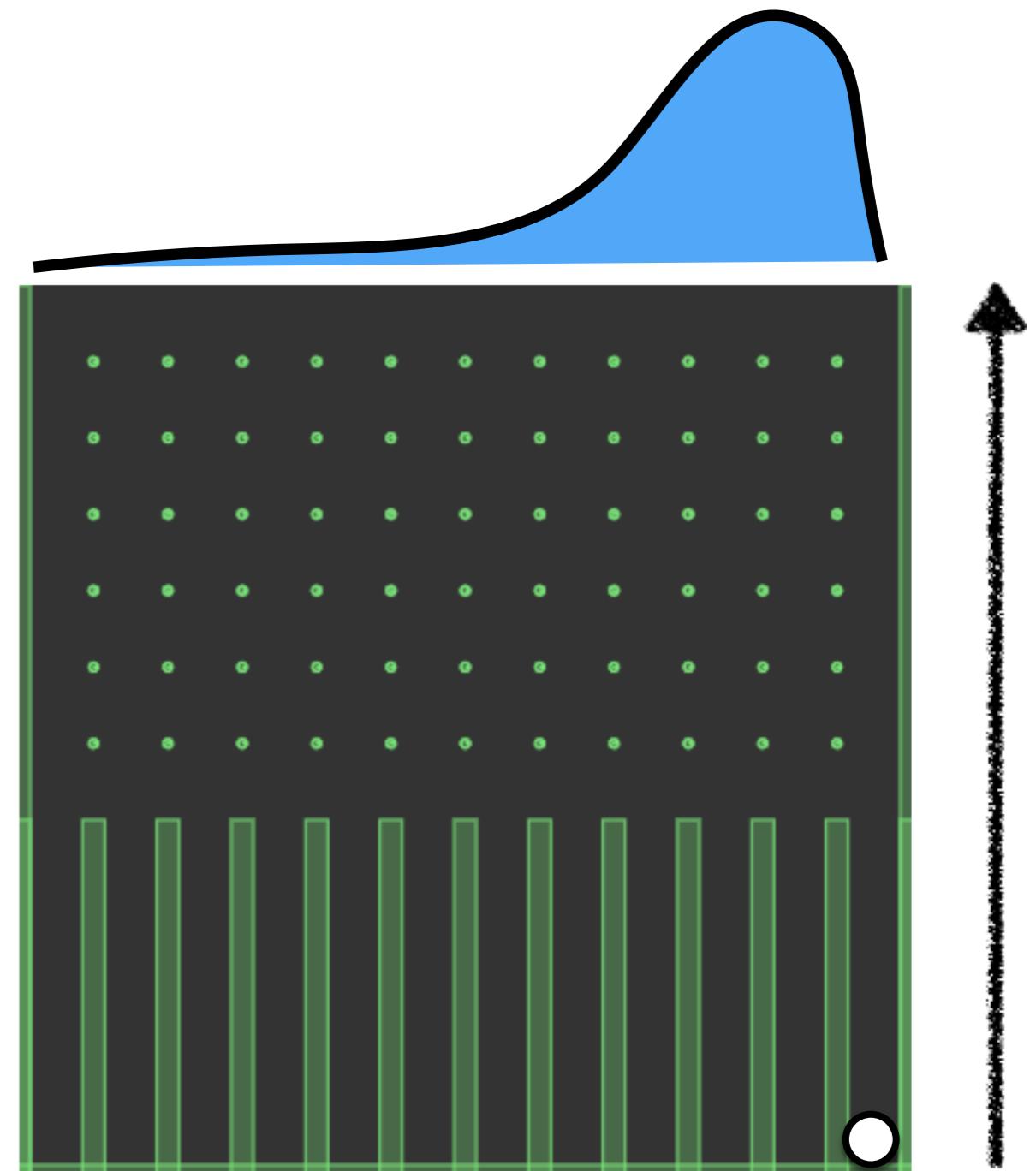
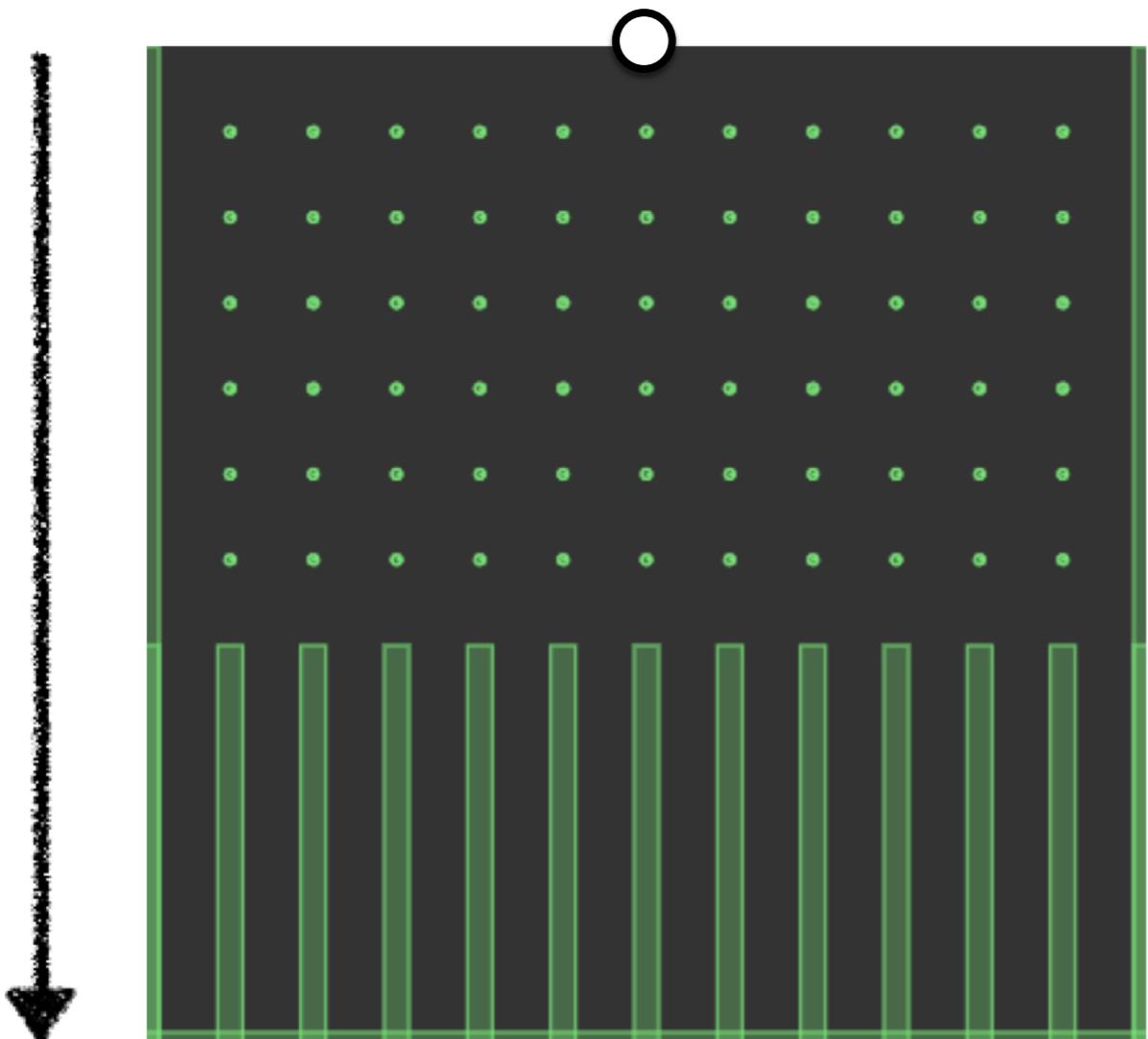
any computable distribution can be represented as the distribution induced by sampling from a probabilistic program

# 3. Doing inference

# Doing inference

Run forward

Where will the ball land?



Reason backward  
Where did the ball come from?

# Doing inference

## Conditional inference:

```
Infer(
```

```
  function() {
```

```
    var a = flip(0.3)
```

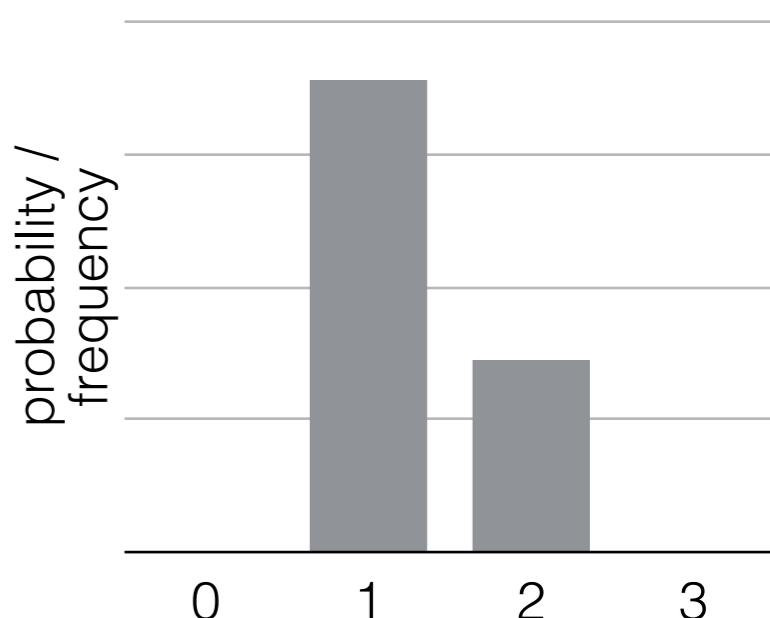
```
    var b = flip(0.3)
```

```
    var c = flip(0.3)
```

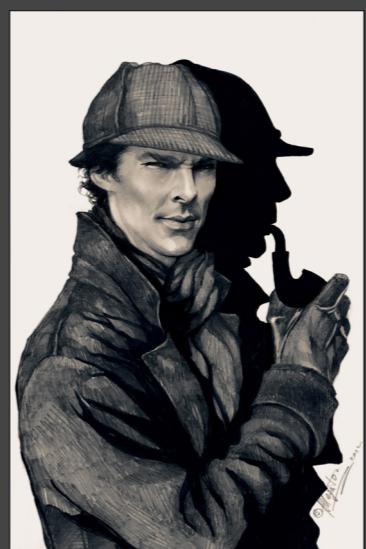
```
    condition(a + b == 1)
    return a + b + c}
```

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

Posterior distribution



“It is an old maxim of mine that when you have excluded the impossible, whatever remains, however improbable, must be the truth.”

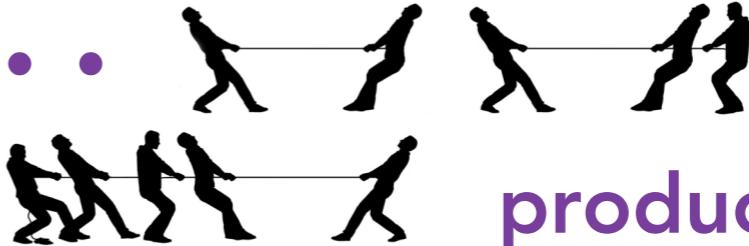


# Doing inference



- 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

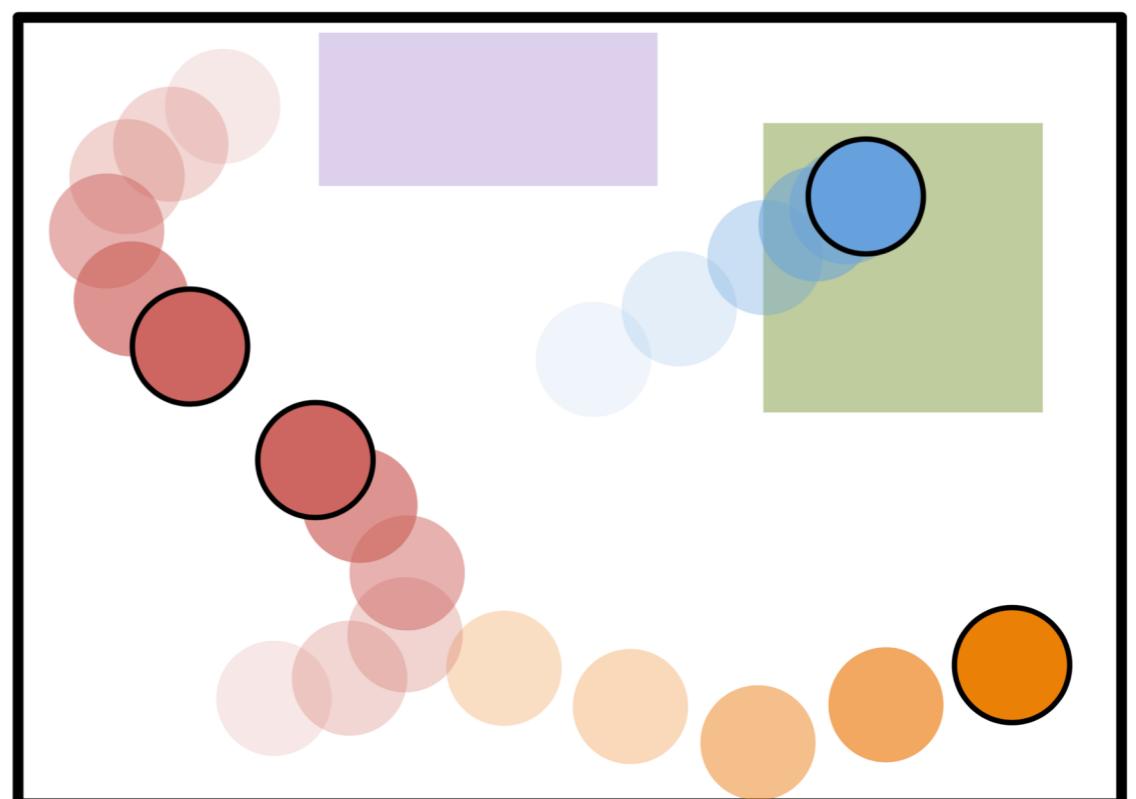
"Concepts have a language-like **compositionality** and encode probabilistic knowledge. These features allow them to be extended **productively** to new situations and support flexible reasoning and learning by **probabilistic inference.**"

```
var towModel = function() {  
    var strength = mem(function (person) {return gaussian(50, 10)})  
  
    var lazy = function(person) {return flip(0.1)} compositionality  
  
    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') • • • productivity  
}
```

# Some more cool examples: Program induction

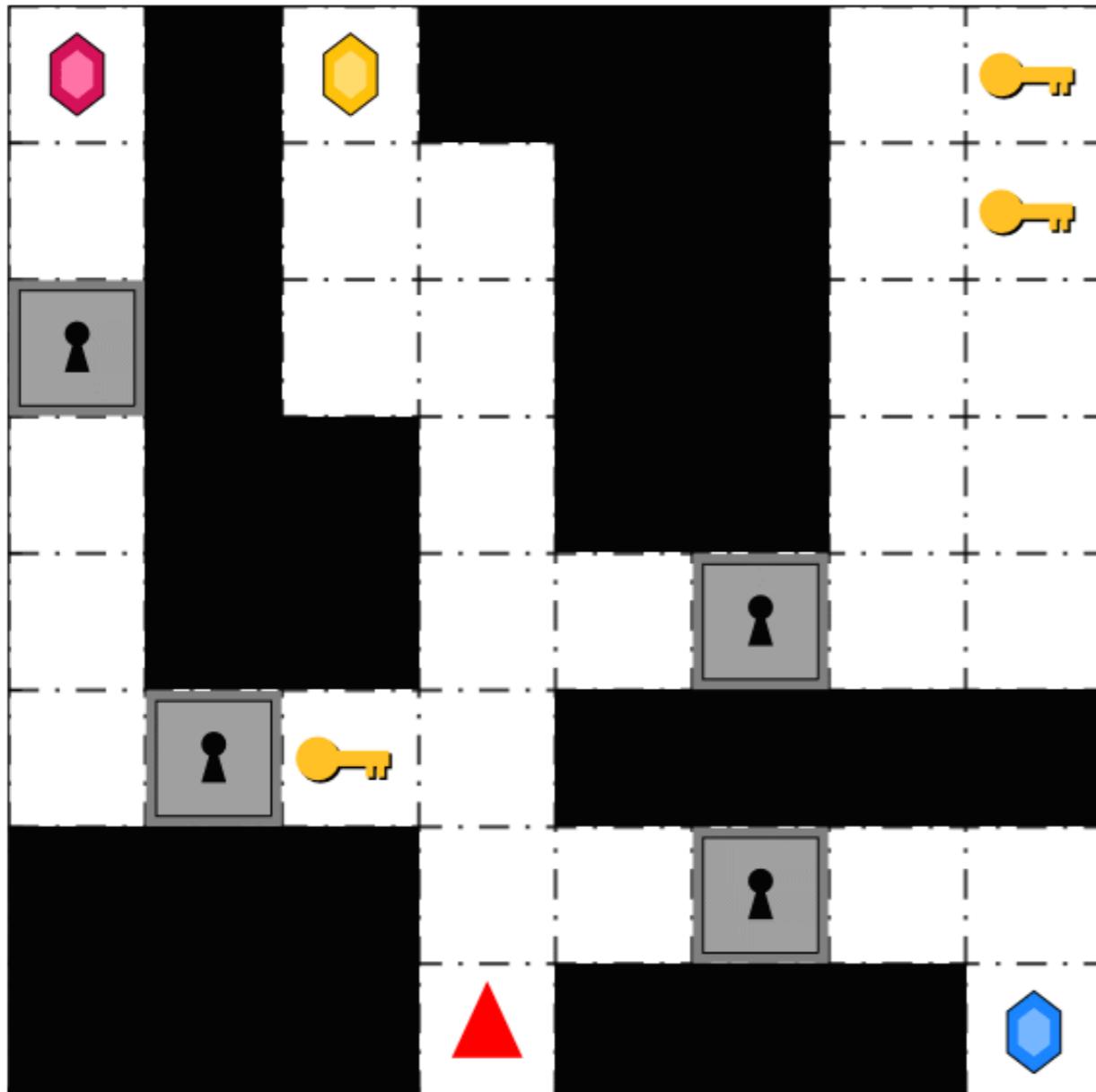
	(i)	(ii)	(iii)
<b>Level N</b>			
Innate concepts			
<b>Level 2</b>			
Entity types	<i>Entity</i>	<pre>(define (make-entity property1 property2 ... )         (list property1 property2 ... ))</pre>	
	<i>Newtonian Dynamics</i>	<pre>(define (run-dynamics entities forces                            init-cond steps dt)     (if (= steps 0) '()         (let* ((m (get-mass entities))                (F (apply-forces forces entities))                (a (/ F m)))           (new-cond (integrate init-cond                                a dt noise)))         (pair new-cond (run-dynamics entities  forces new-cond (- 1 step) dt))))</pre>	
Properties	<i>Puck:</i> ○	<pre>(define puck (make-entity                   pos shape mass vel ... ))</pre>	
	<i>Surface:</i> □	<pre>(define surface (make-entity                   pos shape friction ... ))</pre>	
	<i>Mass</i>	<pre>(define (mass)         (pair "mass" (uniform '(1 3 9))))</pre>	
	<i>Friction</i>	<pre>(define (friction)         (pair "friction" '(uniform '(0 5 20))))</pre>	
Force classes	<i>Pairwise:</i> ○ →	<pre>(define (pairwise-force c1 c2)     (let* ((a (uniform-draw '(-1 0 1))))       (lambda (o1 o2)         (let ((r (euc-dist o1 o2)))           (/ (* a del(o1,col(o1)) del(o2,col(o2)))               (power r 2))))))</pre>	②
	<i>Global:</i> ○ →	<pre>(define (global-force)     (let* ((d (uniform-draw compass-dir)))       (lambda (o) (* k d))))</pre>	
<b>Level 1</b>			
Property values	<ul style="list-style-type: none"> <li>● : large mass</li> <li>○ : medium mass</li> <li>■ : small mass</li> <li>■ : high friction</li> <li>■ : no friction</li> </ul>	<pre>(define world-entities   (map sample-values entity-list))</pre>	
Force parameters	Force b/w reds: attract	<pre>(define world-forces   (map sample-parameters force-list))</pre>	
<b>Level 0 (data)</b>	<i>Initial conditions</i>	<pre>(define scenario   (let* (     (init-cond (sample-init world-entities))     (run-dynamics world-entities                  world-forces init-cond steps dt)))     (init-cond (sample-init world-entities))     (run-dynamics world-entities                  world-forces init-cond steps dt)))</pre>	

Learning at theory of physics



Ullman, Stuhlmüller, Goodman, & Tenenbaum (2018) Learning physical parameters from dynamic scenes. Cognitive Psychology

# Some more cool examples: Goal inference



Inventory

-							

## Goal Noise

```
model agent_step(t, goal, plan, state):
```

Corrupt or repair `goal` with probability  $\epsilon g$ .

**if** `goal` is reached:

**return** nothing

**else if** `length(plan) < t` || `state` is unexpected:

Sample search budget for planner.

Search for partial plan from current `state` to `goal`.

Extend `plan` with partial plan.

**with probability**  $1-\epsilon a$ :

Select `action` from timestep `t` of `plan`.

**else**:

Randomly choose an `action` available in `state`.

**return** `action`

## Partial Planning

## Action Selection

# 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](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](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