Probabilistic Programming Languages

Haluk Dogan
https://haluk.github.io/
hdogan@vivaldi.net

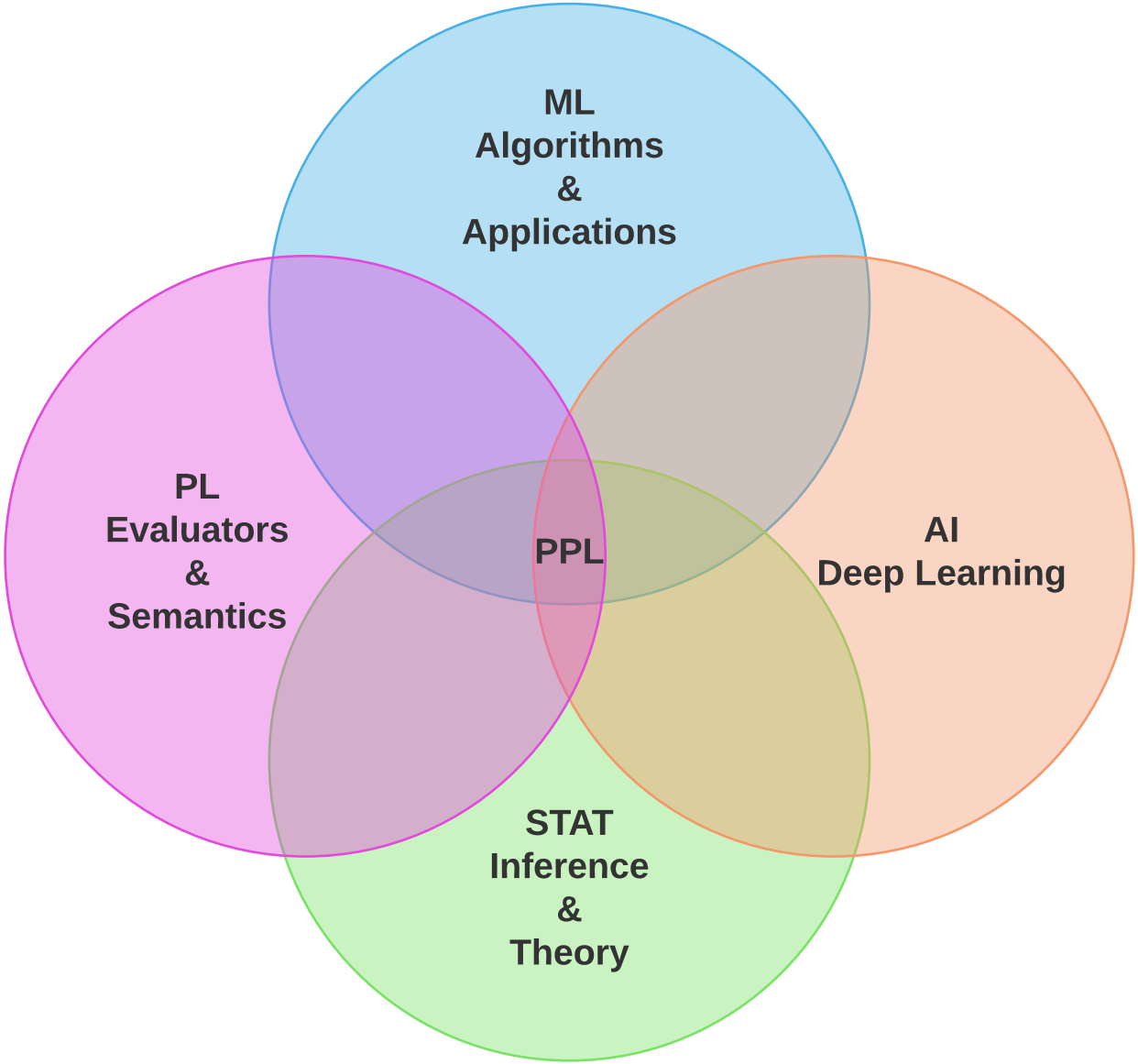
Department of Computer Science
University of Nebraska-Lincoln

March 13, 2020

Some slides copied from

Dr. Frank Wood, University of Oxford

Probabilistic Programming Languages (PPL)



What and Why

Probabilistic Programming is Not

About writing software that behaves probabilistically

- rand as in cryptographic key generator

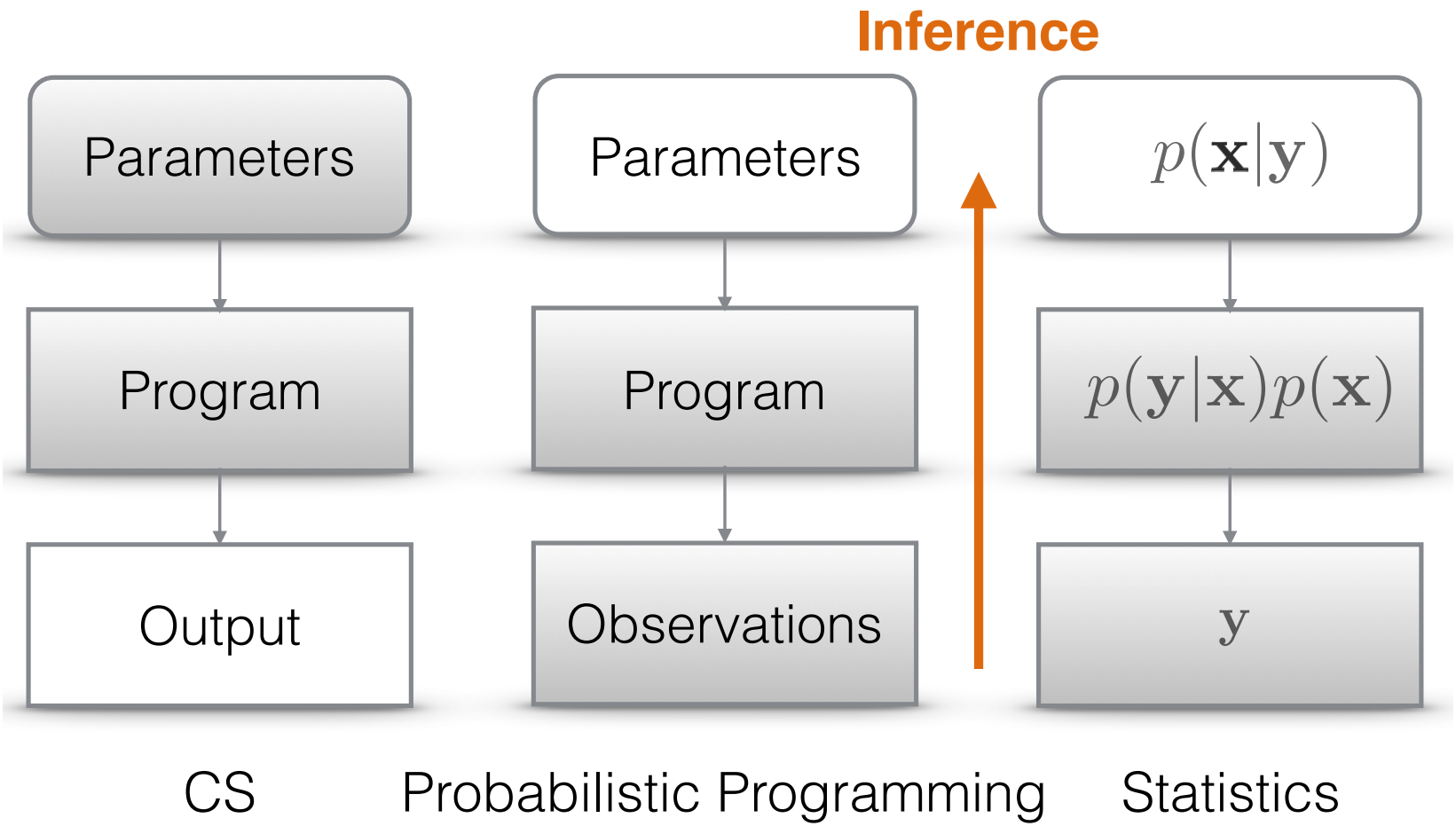
Probabilistic Programming Is

- A tool for statistical modeling
- rand and a great big pile of related tools

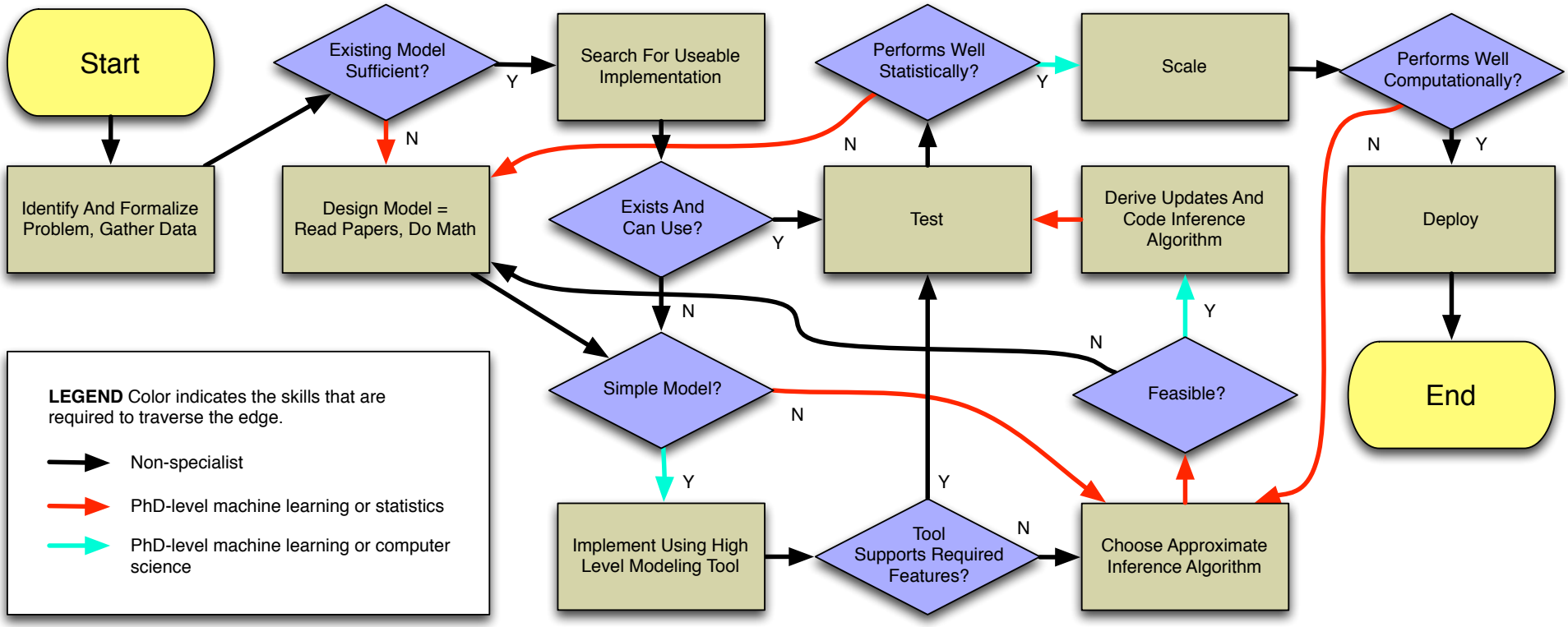
Goal

Bring the old and powerful magic of programming languages, which you already know and love, to the world of statistics.

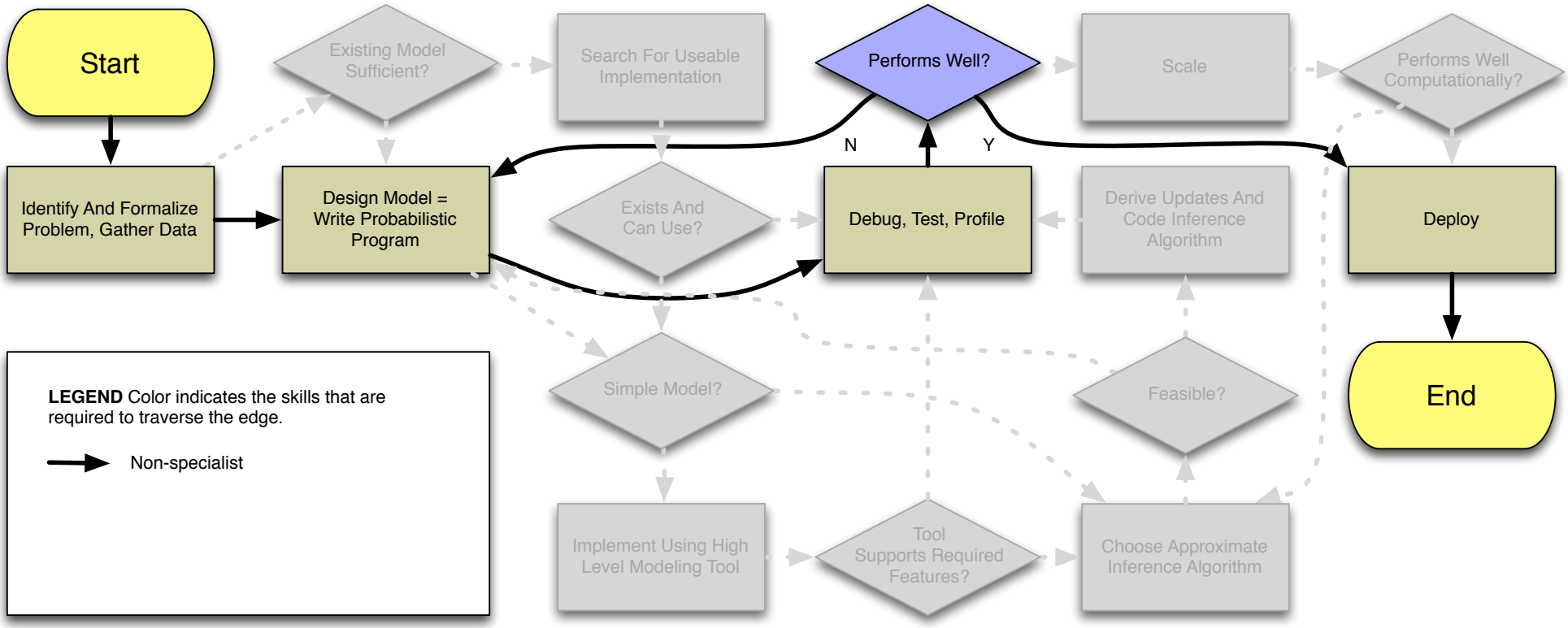
Intuition


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

The Way Machine Learning Is



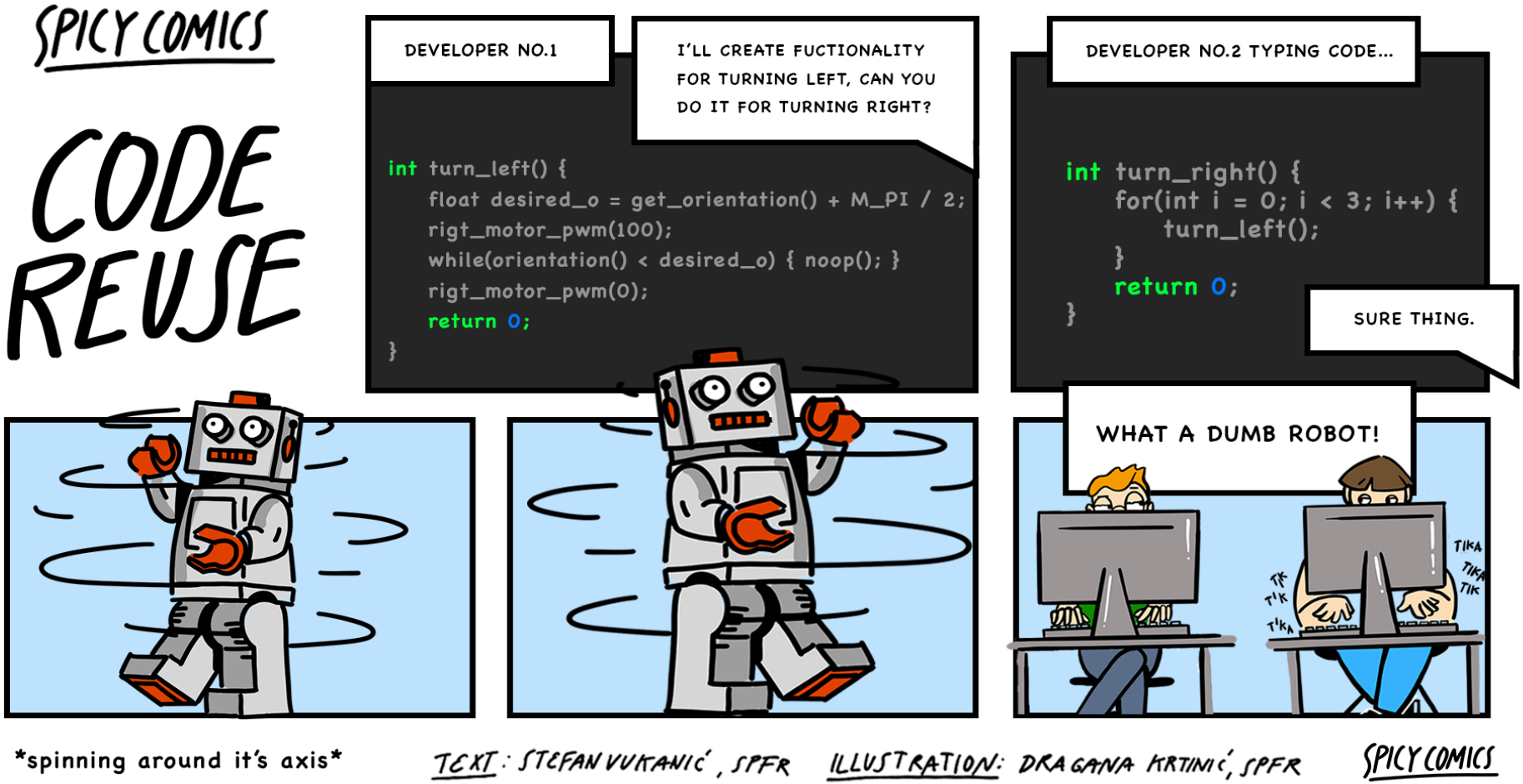
The Way Machine Learning Will Be



PL Concepts to Statistical Modeling

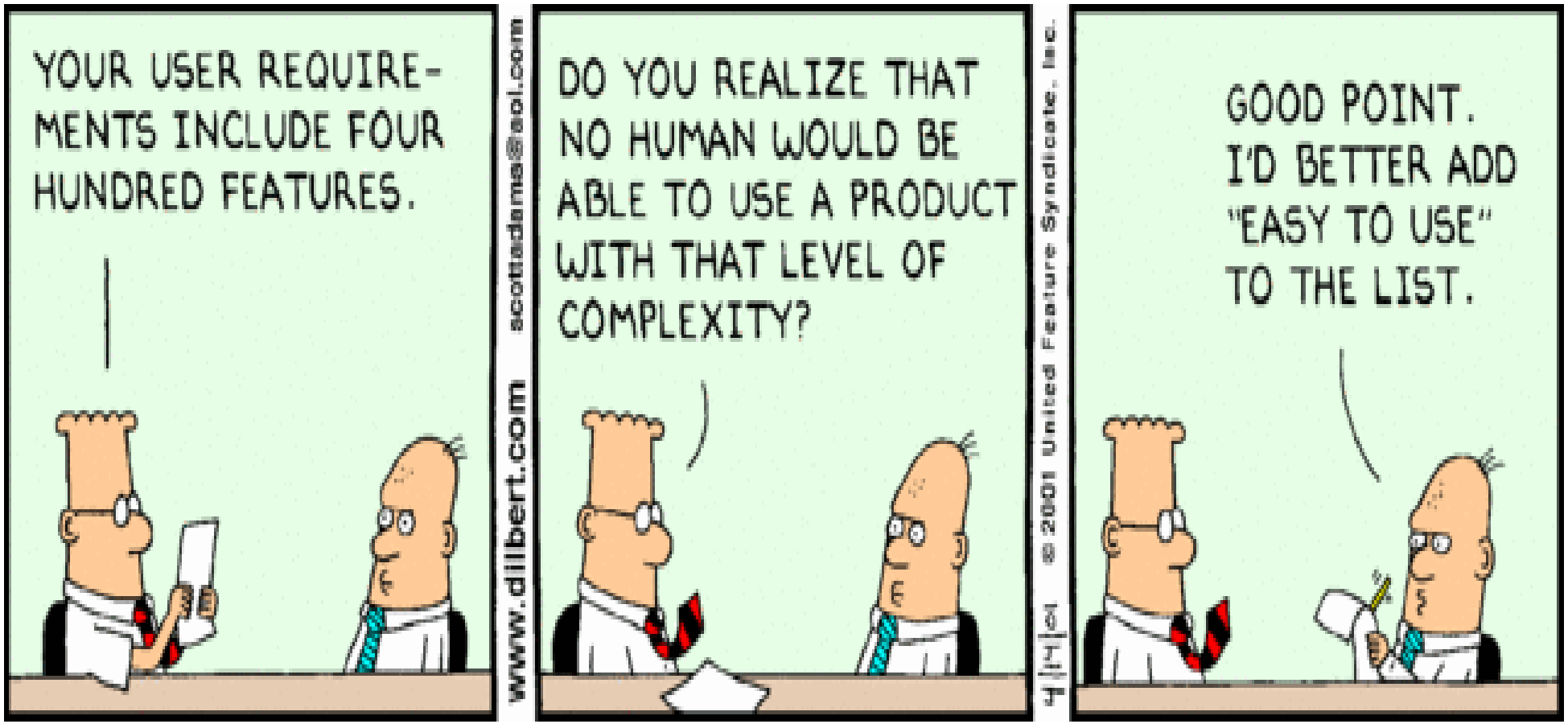
Abstraction

PL Concepts to Statistical Modeling (cont'd)

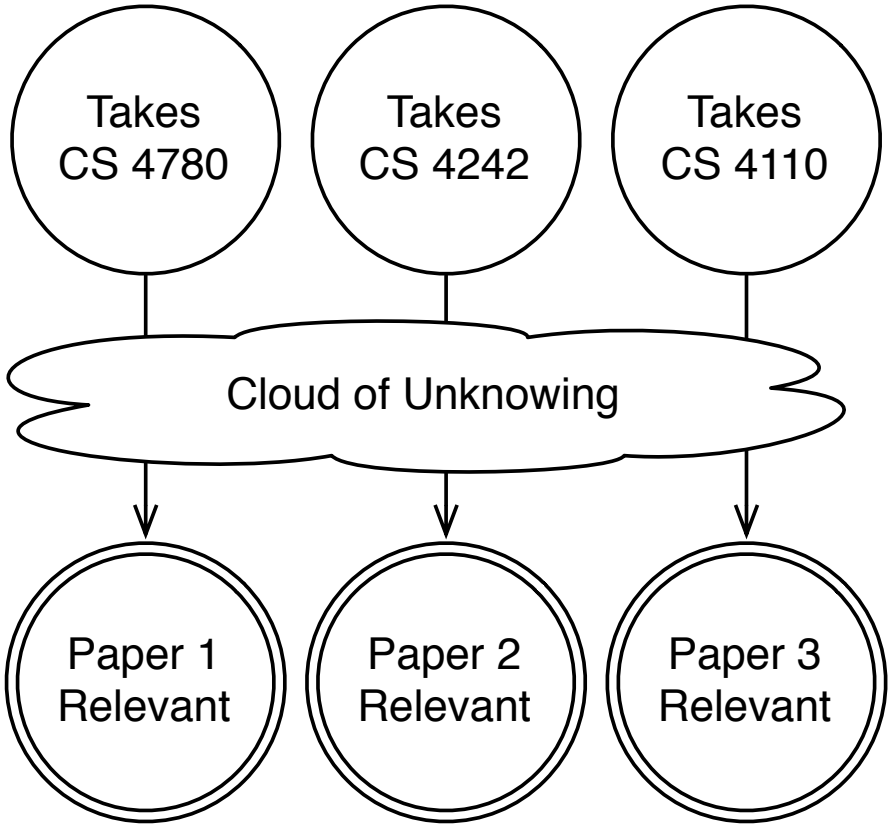


PL Concepts to Statistical Modeling (cont'd)

Simplicity



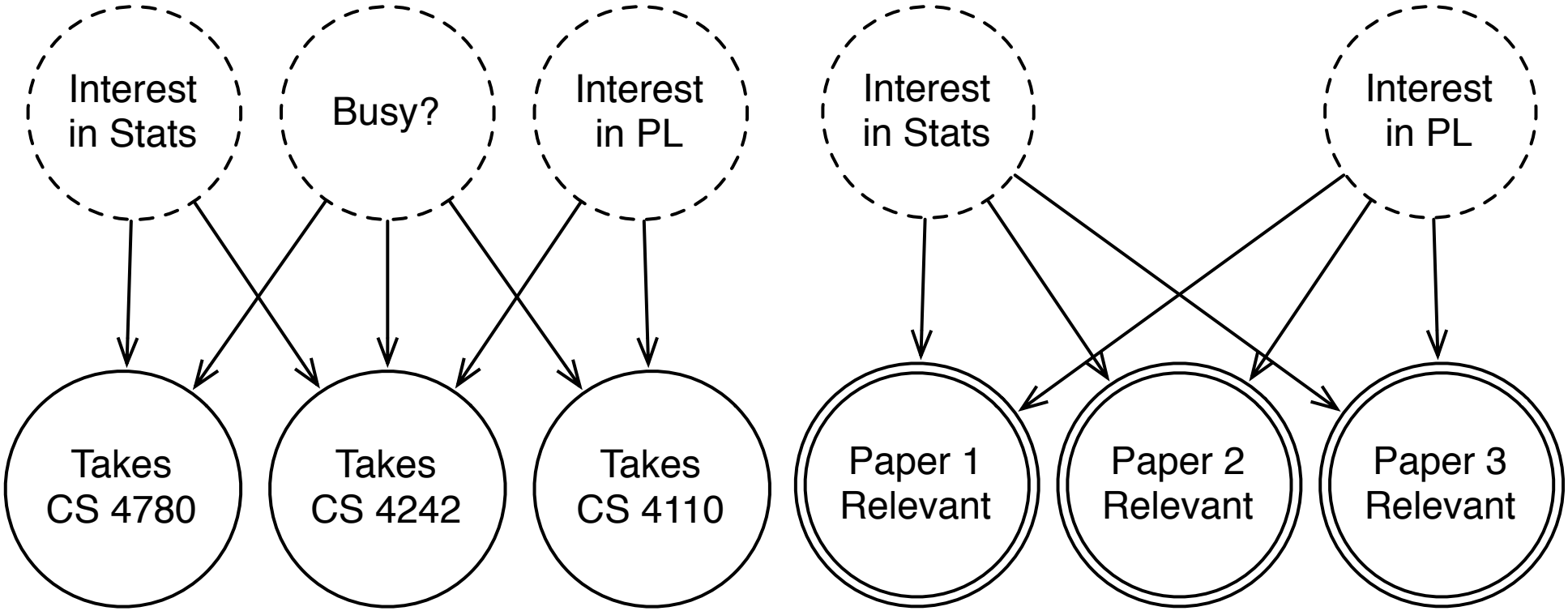
Paper Recommender System[1]



- CS 4110: PL
- CS 4780: ML
- CS 4242: PPL
- Taking 4780 means interested in statistics?
- Taking 4242 and neither 4110 nor 4780 means 50/50 PL/stats?

[1] http://adriansampson.net/doc/ppl.html

Paper Recommender System (cont'd)



Paper Recommender System (cont'd)

$$\Pr[A_{4780}|I_{\text{stats}} \wedge B] = 0.3$$
$$\Pr[A_{4780}|I_{\text{stats}} \wedge \neg B] = 0.8$$
$$\Pr[A_{4780}|\neg I_{\text{stats}}] = 0.1$$
$$\vdots$$
$$\Pr[A_{4242}|I_{\text{stats}} \wedge I_{\text{PL}}] = 0.3$$
$$\Pr[A_{4242}|I_{\text{stats}} \wedge I_{\text{PL}} \wedge \neg B] = 0.8$$
$$\Pr[A_{4242}|\neg(I_{\text{stats}} \vee I_{\text{PL}})] = 0.1$$
$$\vdots$$
$$R_1 \sim I_{\text{PL}} \wedge I_{\text{stats}}$$
$$R_2 \sim I_{\text{PL}}$$
$$R_3 \sim I_{\text{stats}}$$

- no abstraction
- no reuse
- no descriptive variable names,
- no comments
- no debugger
- no type systems

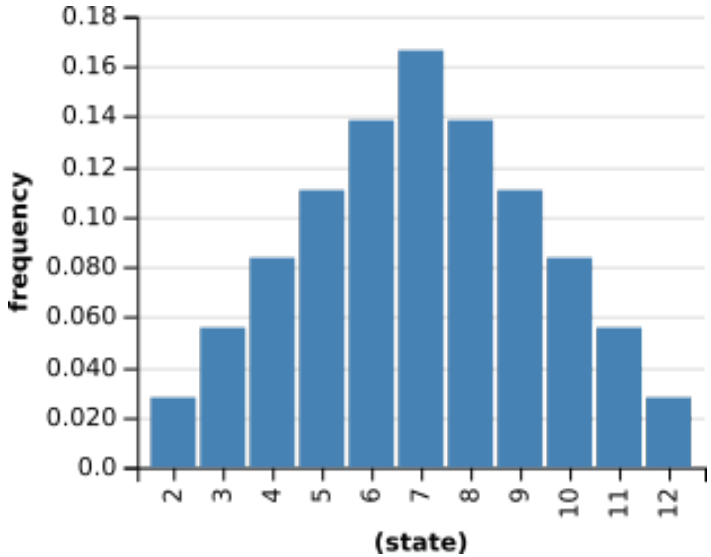
Basic Concepts

Random primitives

```
var b = flip(0.5);
b ? "yes" : "no"
```

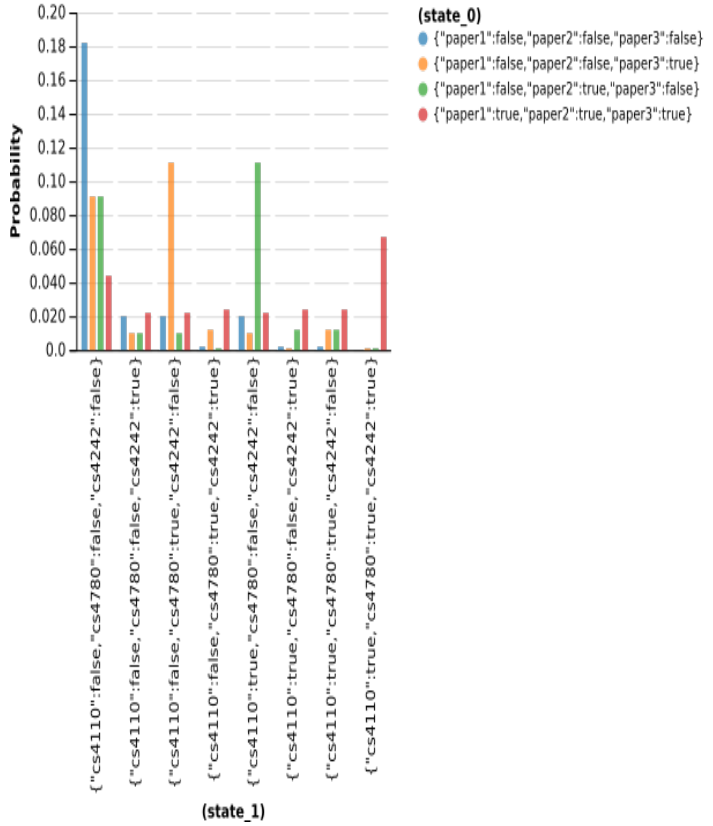
```
var roll = function () {
  var die1 = randomInteger(6) + 1;
  var die2 = randomInteger(6) + 1;
  return die1 + die2;
}

var dist = Enumerate(roll);
print(dist);
viz.auto(dist);
```



Basic Concepts (cont'd)

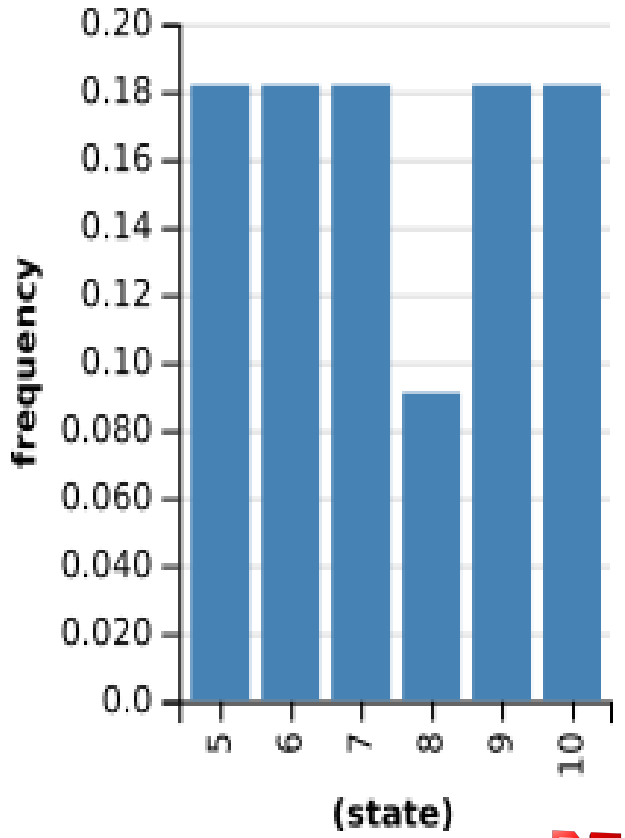
```
1  // Class attendance model.
2  var attendance = function(i_pl, i_stats, busy) {
3    var attendance = function (interest, busy) {
4      if (interest) {
5        return busy ? flip(0.3) : flip(0.8);
6      } else {
7        return flip(0.1);
8      }
9    }
10   var a_4110 = attendance(i_pl, busy);
11   var a_4780 = attendance(i_stats, busy);
12   var a_4242 = attendance(i_pl && i_stats, busy);
13
14   return {cs4110: a_4110, cs4780: a_4780, cs4242: a_4242};
15 }
16
17 // Relevance of our three papers.
18 var relevance = function(i_pl, i_stats) {
19   var rel1 = i_pl && i_stats;
20   var rel2 = i_pl;
21   var rel3 = i_stats;
22
23   return {paper1: rel1, paper2: rel2, paper3: rel3};
24 }
25
26 // A combined model.
27 var model = function() {
28   // Some even random priors for our "student profile."
29   var i_pl = flip(0.5);
30   var i_stats = flip(0.5);
31   var busy = flip(0.5);
32
33   return [relevance(i_pl, i_stats), attendance(i_pl, i_stats, busy)];
34 }
35
36 var dist = Enumerate(model);
37 viz.auto(dist);
```



Basic Concepts (cont'd)

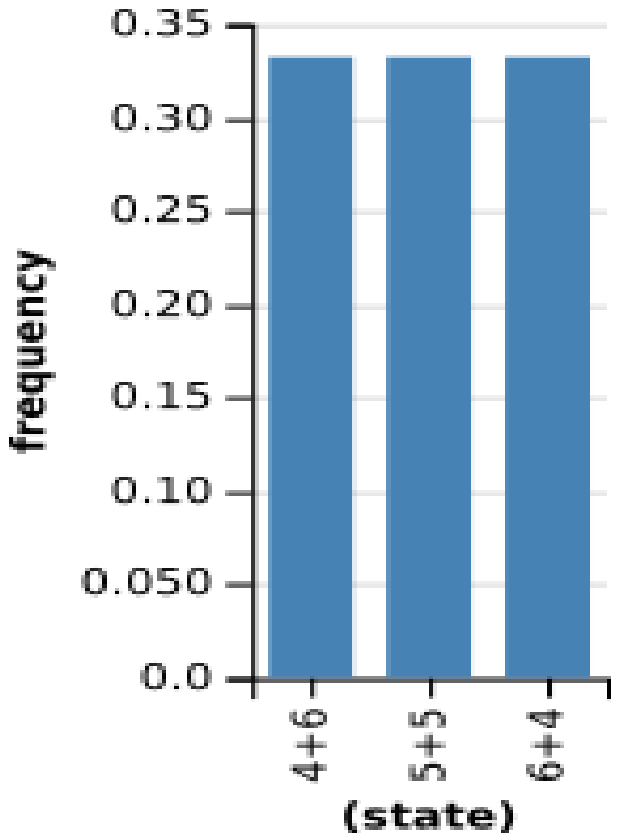
Conditioning

```
1  var roll_condition4 = function () {
2    var die1 = randomInteger(6) + 1;
3    var die2 = randomInteger(6) + 1;
4
5    // Only keep executions where at least one die is a 4.
6    if (!(die1 === 4 || die2 === 4)) {
7      factor(-Infinity);
8    }
9
10   return die1 + die2;
11 }
12
13 var dist = Enumerate(roll_condition4);
14 print(dist);
15 viz.auto(dist);
```



Basic Concepts (cont'd)

```
1  var roll_condition10 = function () {
2    var die1 = randomInteger(6) + 1;
3    var die2 = randomInteger(6) + 1;
4
5    // Discard any executions that don't sum to 10.
6    var out = die1 + die2;
7    if (out !== 10) {
8      factor(-Infinity);
9    }
10
11   // Return the values on the dice.
12   return die1 + "+" + die2;
13 }
14 var dist = Enumerate(roll_condition10);
15 print(dist);
16 viz.auto(dist);
```



Actually Recommending Papers Example

Attend CS 4110 (PL), and CS 4242 (PPL), but not the CS 4780 (ML)

```
1  // A model query that describes my class attendance.
2  // attendance and relevance models were defined previously
3
4
5  // A wrapper for `factor` for requiring conditions to be true.
6  var require = function(cond) {
7    if (!cond) {
8      factor(-Infinity);
9    }
10 }
11
12 var recommend = function() {
13   var i_pl = flip(0.5);
14   var i_stats = flip(0.5);
15   var busy = flip(0.5);
16
17   // Require my class attendance.
18   var att = attendance(i_pl, i_stats, busy);
19   require(att.cs4110 && att.cs4242 && !att.cs4780);
20
21   return relevance(i_pl, i_stats);
22 }
23
24 var dist = Enumerate(recommend);
25 viz.table(dist);
```

paper1	paper2	paper3	prob.
true	true	true	0.603
false	true	false	0.312
false	false	false	0.057
false	false	true	0.028

Inference

- Enumerate
- Rejection Sampling
  - works for small examples
  - drawing different random values for each random primitive on each execution
  - apply the program's conditioning to weight each sample and total them all up.
  - waste a lot of work taking samples that don't matter if conditioning is present

```
1 var sampled = ParticleFilter(rec('paper1'), 1000);
```

- MCMC
  - random walk over executions
  - Metropolis Hastings
  - reusing the trace
  - particle filters with rejuvenation

A Zoo of Probabilistic Programming Frameworks

- WebPPL (http://webppl.org/)
- Stan (https://mc-stan.org/)
- Tensorflow Probability (aka TFP https://www.tensorflow.org/probability)
- PyMC3 (relies on Theano https://docs.pymc.io/)
- PyMC4 (under active development, relies on TFP, https://github.com/pymc-devs/pymc4)
- Pyro (relies on PyTorch http://pyro.ai/)
- Greta (https://greta-stats.org/)
- Infer.NET (https://dotnet.github.io/infer/)
- BUGS (https://www.mrc-bsu.cam.ac.uk/software/bugs/)
- JAGS (http://mcmc-jags.sourceforge.net/)
- Anglican (https://probprog.github.io/anglican/index.html)
- Figaro (https://www.cra.com/work/case-studies/figaro)

People to Watch

- Frank Wood (http://www.robots.ox.ac.uk/~fwood/)
- Vikash Mansinghka (http://probcomp.csail.mit.edu/principal-investigator/)
- Noah D. Goodman (https://cocolab.stanford.edu/ndg.html)
- David Wingate (https://cs.byu.edu/faculty/dw87)
- Avi Pfeffer https://dblp.org/pers/p/Pfeffer:Avi.html
- Robert Zinkov (https://www.zinkov.com/)
- Andy Gordon (https://www.microsoft.com/en-us/research/people/adg/)
- John Winn (https://www.microsoft.com/en-us/research/people/jwinn/)
- Dan Roy (http://danroy.org/)

Questions

A repository for generative models http://forestdb.org/

Questions?