Probability and Prejudice

Bridging the Gap Between Machine Learning and Programming Languages

Neil Toronto

University of Maryland

2015/04/17
Bridging the Gap Between Machine Learning and Programming Languages
Bridging the Gap Between Machine Learning and Programming Languages

- Illustrate part of the intersection of machine learning (ML) and programming languages (PL), and the technical and cultural gaps
Bridging the Gap Between Machine Learning and Programming Languages

• Illustrate part of the intersection of machine learning (ML) and programming languages (PL), and the technical and cultural gaps

• Why: Everyone knows we have a hard time working together
Bridging the Gap Between Machine Learning and Programming Languages

• Illustrate part of the intersection of machine learning (ML) and programming languages (PL), and the technical and cultural gaps

• Why: Everyone knows we have a hard time working together

• “Neil, that’s a dangerous talk.”
Bridging the Gap Between Machine Learning and Programming Languages

• Illustrate part of the intersection of machine learning (ML) and programming languages (PL), and the technical and cultural gaps

• Why: Everyone knows we have a hard time working together

• “Neil, that’s a dangerous talk.”

• Best presented by telling my story (cue wavy blur cut)...

Early Years
Master's Research: Super-Resolution

Toronto, Morse, Seppi, Ventura. *Super-Resolution via Recapture and Bayesian Effect Modeling*. CVPR 2009
Toronto, Morse, Seppi, Ventura. Super-Resolution via Recapture and Bayesian Effect Modeling. CVPR 2009
Half a page of beautiful math

\[ C_{i,j}^x \equiv i + \frac{1}{2} \quad i \in 0..m - 1 \]
\[ C_{i,j}^y \equiv j + \frac{1}{2} \quad j \in 0..n - 1 \]
\[ N_9(x, y) \equiv \{ i \in \mathbb{Z} \mid -1 \leq i - \lfloor x \rfloor \leq 1 \} \times \{ j \in \mathbb{Z} \mid -1 \leq j - \lfloor y \rfloor \leq 1 \} \]
\[ \text{dist}(x, y, \theta, d) \equiv x \cos \theta + y \sin \theta - d \]
\[ \text{prof}(d, \sigma, v^+, v^-) \equiv \frac{v^+-v^-}{2} \text{erf} \left( \frac{d}{\sqrt{2\sigma}} \right) + \frac{v^+ + v^-}{2} \]
\[ \text{edge}(x, y, \theta, d, v^+, v^-, \sigma) \equiv \text{prof}(\text{dist}(x, y, \theta, d), \sigma, v^+, v^-) \]

\[ S_{i,j}^{\text{edge}}(x, y) \equiv \text{edge}(x - C_{i,j}^x, y - C_{i,j}^y, S_{i,j}^\theta, S_{i,j}^d, S_{i,j}^{v+}, S_{i,j}^{v-}, S_{i,j}^\sigma) \]
\[ E[h(S_{x,y})] = \sum_{k,l \in N_9(x,y)} w(x - C_{k,l}^x, y - C_{k,l}^y) h(S_{k,l}^{\text{edge}}(x, y)) \]

\[ S_{i,j}^\theta \sim \text{Uniform}(-\pi, \pi) \quad S_{i,j}^{v+} \sim \text{Uniform}(0, 1) \quad I_{i,j} | S_{N_9(i,j)} \sim \text{Normal}(E[S_{i,j}], \omega) \]
\[ S_{i,j}^d \sim \text{Uniform}(-3, 3) \quad S_{i,j}^{v-} \sim \text{Uniform}(0, 1) \quad \Phi_{i,j}(S_{N_9(i,j)}) \equiv \exp \left( -\frac{\text{Var}[S_{i,j}]}{2\gamma^2} \right) \]

\[ S_{i,j}^\sigma \sim \text{Beta}(1.6, 1) \]
Master’s Research: Code
Master’s Research: Code
Master’s Research: Code
Master’s Research: Code

600 lines of evil parallelized Python code
Results: Super-Resolution

Competitor and BEI on 4x super-resolution:

Resolution Synthesis
Results: Super-Resolution

Competitor and BEI on 4x super-resolution:

Resolution Synthesis  
Bayesian Edge Inference
Results: Super-Resolution

Competitor and BEI on 4x super-resolution:

Beat state-of-the-art on “objective” measures
Results: Other Reconstruction

CCD
Demosaicing
Results: Other Reconstruction

CCD Demosaicing
Results: Other Reconstruction

CCD
Demosaicing

Inpainting

In probability theory and statistical inference, the beta distribution is a family of continuous probability distributions defined on the interval [0, 1] parameterized by two positive shape parameters, typically denoted by \( \alpha \) and \( \beta \). It is a special case of the Dirichlet distribution with only two parameters. Since the Dirichlet distribution is the conjugate prior of the multinomial distribution,
In probability theory and the beta distribution is a family of continuous probability distributions defined on the interval [0,1], parameterized by two positive shape parameters, typically denoted by \( \alpha \) and \( \beta \). It is a special case of the Dirichlet distribution with only two parameters. Since the Dirichlet distribution is the conjugate prior of the multinomial distribution,
Results: Other Reconstruction

CCD Demosaicing

In probability theory and the beta distribution is a family of continuous probability distributions defined on the interval [0, 1] parameterized by two positive shape parameters, typically denoted by $\alpha$ and $\beta$. It is a special case of the Dirichlet distribution with only two parameters. Since the Dirichlet distribution is the conjugate prior of the multinomial distribution...

It only looks like magic
Only Mostly Satisfying

Problem 1: Still not sure the program is right
Only Mostly Satisfying

Problem 1: Still not sure the program is right

Problem 2: *Smooth* edges instead of *step* edges
Only Mostly Satisfying

Problem 1: Still not sure the program is right

Problem 2: Smooth edges instead of step edges

“To approximate blurring with a spatially varying point-spread function (PSF), we assign each facet a Gaussian PSF and convolve each analytically before combining outputs.”
Problem 1: Still not sure the program is right

Problem 2: Smooth edges instead of step edges

“To approximate blurring with a spatially varying point-spread function (PSF), we assign each facet a Gaussian PSF and convolve each analytically before combining outputs.”

Without the fancy talk: We can't figure out how to model blur as part of image capture, so we hacked it into the scene model.
The First Crossing

Heard a PL professor, Jay McCarthy, was coming to BYU and asked him to take me on
The First Crossing

Heard a PL professor, Jay McCarthy, was coming to BYU and asked him to take me on

“Hey, I can use PL to fix my old problems!”
Communication Issues

What we say to dogs

Okay, Ginger! I've had it! You stay out of the garbage! Understand Ginger? Stay out of the garbage, or else!

What they hear

blah blah GINGER, blahblah, blah! blah GINGER, blah! blah, blah blah blah...
Human Communication Issues

Jay says... The Curry-Howard correspondence is only a general observation, except in the calculus of inductive constructions and similar languages.
Human Communication Issues

Jay says... The Curry-Howard correspondence is only a general observation, except in the calculus of inductive constructions and similar languages.

I hear... Indian cuisine correlates with integration, and possibly something about making learning algorithms.
**Human Communication Issues**

**Jay says...** The Curry-Howard correspondence is only a general observation, except in the calculus of inductive constructions and similar languages.

**I hear...** Indian cuisine correlates with integration, and possibly something about making learning algorithms.

**I say...** Metropolis-Hastings algorithms are inherently **sequential** because they **step** according to a distribution **conditioned** on their last location.
Human Communication Issues

Jay says... The Curry-Howard correspondence is only a general observation, except in the calculus of inductive constructions and similar languages.

I hear... Indian cuisine correlates with integration, and possibly something about making learning algorithms.

I say... Metropolis-Hastings algorithms are inherently sequential because they step according to a distribution conditioned on their last location.

Jay hears... You can’t parallelize algorithms for walking around in cities, and possibly something about hair care.
Bridging the Gap

• ML and PL are separated by fairly wide gaps
Bridging the Gap

• ML and PL are separated by fairly wide gaps

• Some of them make it hard to work together
Bridging the Gap

• ML and PL are separated by fairly wide gaps

• Some of them make it hard to work together

• Objectives:
Bridging the Gap

• ML and PL are separated by fairly wide gaps

• Some of them make it hard to work together

• Objectives:
  ◦ Identify major gaps
Bridging the Gap

• ML and PL are separated by fairly wide gaps

• Some of them make it hard to work together

• Objectives:
  ○ Identify major gaps
  ○ Classify as technical or cultural
Bridging the Gap

- ML and PL are separated by fairly wide gaps
- Some of them make it hard to work together
- Objectives:
  - Identify major gaps
  - Classify as technical or cultural
  - Show how to close or bridge them
Culture Gap: Vocabulary (PL)

• PL theorists have 30 terms for meaning
Culture Gap: Vocabulary (PL)

- PL theorists have 30 terms for **meaning**
  - Denotational, operational, big-step, small-step, axiomatic, concrete, collecting, abstract, etc.
Culture Gap: Vocabulary (PL)

• PL theorists have 30 terms for meaning
  ◦ Denotational, operational, big-step, small-step, axiomatic, concrete, collecting, abstract, etc., semantics
Culture Gap: Vocabulary (PL)

• PL theorists have 30 terms for **meaning**
  ◦ Denotational, operational, big-step, small-step, axiomatic, concrete, collecting, abstract, etc., **semantics**

• Word used depends on abstraction level, target language, finiteness, use, philosophy
Culture Gap: Vocabulary (PL)

- PL theorists have 30 terms for **meaning**
  - Denotational, operational, big-step, small-step, axiomatic, concrete, collecting, abstract, etc., **semantics**
- Word used depends on abstraction level, target language, finiteness, use, philosophy
- I (usually) stick with **operational** and **denotational**
Culture Gap: Vocabulary (PL)

• PL theorists have 30 terms for meaning
  • Denotational, operational, big-step, small-step, axiomatic, concrete, collecting, abstract, etc., semantics

• Word used depends on abstraction level, target language, finiteness, use, philosophy

• I (usually) stick with operational and denotational

• More precise: “the meaning of programs as X”
Culture Gap: Vocabulary (ML)

• Statisticians have 30 terms for distribution
Culture Gap: Vocabulary (ML)

- Statisticians have 30 terms for **distribution**
  - Prior, likelihood, joint, marginal, posterior, posterior predictive, pdf, pmf, measure, transition kernel, etc.
Culture Gap: Vocabulary (ML)

- Statisticians have 30 terms for distribution
  - Prior, likelihood, joint, marginal, posterior, posterior predictive, pdf, pmf, measure, transition kernel, etc.

- Word used depends on countable vs. uncountable support, specified vs. inferred, continuity, use, philosophy
• Statisticians have 30 terms for **distribution**
  - Prior, likelihood, joint, marginal, posterior, posterior predictive, pdf, pmf, measure, transition kernel, etc.

• Word used depends on countable vs. uncountable support, specified vs. inferred, continuity, use, philosophy

• I (usually) stick with **distribution** and **conditional distribution**
• Statisticians have 30 terms for **distribution**
  - Prior, likelihood, joint, marginal, posterior, posterior predictive, pdf, pmf, measure, transition kernel, etc.

• Word used depends on countable vs. uncountable support, specified vs. inferred, continuity, use, philosophy

• I (usually) stick with **distribution** and **conditional distribution**

• More precise: “the distribution of X given Y”
Culture Gap: Formalisms

• Strange vocabulary refers to even stranger math
Culture Gap: Formalisms

- Strange vocabulary refers to even stranger math

\[
S_{i,j}^\theta \sim \text{Uniform}(\pi, \pi) \quad S_{i,j}^{\mu^+} \sim \text{Uniform}(0, 1) \\
S_{i,j}^d \sim \text{Uniform}(-3, 3) \quad S_{i,j}^\nu \sim \text{Uniform}(0, 1) \\
S_{i,j}^\sigma \sim \text{Beta}(1.6, 1)
\]

\[I_{i,j} \mid S_{N9(i,j)} \sim \text{Normal}(E[S_{i,j}], \omega)\]

\[
\Phi_{i,j}(S_{N9(i,j)}) \equiv \exp \left( -\frac{\text{Var}[S_{i,j}]}{2\gamma^2} \right)
\]
Culture Gap: Formalisms

• Strange vocabulary refers to even stranger math

\( S^\theta_{i,j} \sim \text{Uniform}(\pi, \pi) \quad S^+_{i,j} \sim \text{Uniform}(0, 1) \)
\( S^d_{i,j} \sim \text{Uniform}(-3, 3) \quad S^-_{i,j} \sim \text{Uniform}(0, 1) \)
\( S^\sigma_{i,j} \sim \text{Beta}(1.6, 1) \)

\[ L_{i,j} \mid S_{N9(i,j)} \sim \text{Normal}(E[S_{i,j}], \omega) \]

\[ \Phi_{i,j}(S_{N9(i,j)}) \equiv \exp \left( -\frac{\text{Var}[S_{i,j}]}{2\gamma^2} \right) \]

\[
[f := e; \ldots ; e_b]_a \quad \equiv \quad f := [e]_a; \ldots ; [e_b]_a
\]
\[
[\text{let } e \ e_b]_a \quad \equiv \quad ([e]_a \&\&_a \text{ arr } \text{id}) \gggg_a [e_b]_a
\]
\[
[\text{env } 0]_a \quad \equiv \quad \text{arr}_a \text{ fst}
\]
\[
[\text{env } (n + 1)]_a \quad \equiv \quad \text{arr}_a \text{ snd} \gggg_a [\text{env } n]_a
\]
\[
[\text{if } e_c \text{ then } e_l \text{ else } e_f]_a \quad \equiv \quad \text{ifte}_a [e_c]_a \ (\text{lazy}_a \lambda 0. [e_l]_a) \ (\text{lazy}_a \lambda 0. [e_f]_a)
\]

\[
[\langle e_1, e_2 \rangle]_a \quad \equiv \quad [e_1]_a \&\&_a [e_2]_a
\]
\[
[f \ e]_a \quad \equiv \quad [\langle e, \langle \rangle \rangle]_a \gggg_a f
\]
\[
[\delta \ e]_a \quad \equiv \quad [e]_a \gggg_a \text{ arr}_a \delta
\]
\[
[v]_a \quad \equiv \quad \text{arr}_a \ (\text{const } v)
\]
Culture Gap: Formalisms

• Strange vocabulary refers to even stranger math

\[ S^\theta_{i,j} \sim \text{Uniform}(-\pi, \pi) \quad S^{\nu^+}_{i,j} \sim \text{Uniform}(0, 1) \]
\[ S^d_{i,j} \sim \text{Uniform}(-3, 3) \quad S_{i,j}^{-\nu} \sim \text{Uniform}(0, 1) \]
\[ S^\sigma_{i,j} \sim \text{Beta}(1.6, 1) \]

\[ I_{i,j}| S_{N|9(i,j)} \sim \text{Normal}(E[S_{i,j}], \omega) \]

\[ \Phi_{i,j}(S_{N|9(i,j)}) \equiv \exp \left( -\frac{\text{Var}[S_{i,j}]}{2\gamma^2} \right) \]

\[ [f := e; \ldots; e_b]_a \equiv f := [e]_a; \ldots; [e_b]_a \]
\[ [\text{let } e e_b]_a \equiv ([e]_a \&\&_a \text{ arr}_a \text{ id}) \gggg_a [e_b]_a \]
\[ [\text{env } 0]_a \equiv \text{ arr}_a \text{ fst} \]
\[ [\text{env } (n + 1)]_a \equiv \text{ arr}_a \text{ snd} \gggg_a [\text{env } n]_a \]

\[ ([f e]_a \equiv [[(e, \langle\rangle)]_a \gggg_a f \]
\[ [\delta e]_a \equiv [e]_a \gggg_a \text{ arr}_a \delta \]
\[ [v]_a \equiv \text{ arr}_a (\text{ const } v) \]

\[ [\text{if } e_c \text{ then } e_t \text{ else } e_f]_a \equiv \text{ ifte}_a [e_c]_a \text{ (lazy}_a \lambda.0.[e_t]_a) \text{ (lazy}_a \lambda.0.[e_f]_a) \]

• Closing the gap: only option is to take time to explain
Explaining Semantics To ML
Explaining Semantics To ML

• Avoid:
  ○ Lambdas, continuations, type systems, other favorites
Explaining Semantics To ML

• Avoid:
  ○ Lambdas, continuations, type systems, other favorites
  ○ Philosophy (e.g. “Programs mean nothing until the language has a semantics”)

Explaining Semantics To ML

• Avoid:
  ◦ Lambdas, continuations, type systems, other favorites
  ◦ Philosophy (e.g. “Programs mean nothing until the language has a semantics”)

• Do:
  ◦ Use arithmetic, draw on intuition
Explaining Semantics To ML

• Avoid:
  ◦ Lambdas, continuations, type systems, other favorites
  ◦ Philosophy (e.g. “Programs mean nothing until the language has a semantics”)

• Do:
  ◦ Use arithmetic, draw on intuition
  ◦ Add a nontrivial language feature
Explaining Semantics To ML

• Avoid:
  ◦ Lambdas, continuations, type systems, other favorites
  ◦ Philosophy (e.g. “Programs mean nothing until the language has a semantics”)

• Do:
  ◦ Use arithmetic, draw on intuition
  ◦ Add a nontrivial language feature
  ◦ Emphasize the need for an explicit mathematical model
Why Semantics?

Q. Why do Bayesians create models of processes?
Why Semantics?

Q. Why do Bayesians create models of processes?

A. Reproducibility, formal manipulation, proof of properties
Why Semantics?

Q. Why do Bayesians create models of processes?
   A. Reproducibility, formal manipulation, proof of properties

Q. Why do PL researchers create models of languages?
Why Semantics?

Q. Why do Bayesians create models of processes?
   A. Reproducibility, formal manipulation, proof of properties

Q. Why do PL researchers create models of languages?
   A. Reproducibility, formal manipulation, proof of properties
Why Semantics?

Q. Why do Bayesians create models of processes?

A. Reproducibility, formal manipulation, proof of properties

Q. Why do PL researchers create models of languages?

A. Reproducibility, formal manipulation, proof of properties

Common ideal: explicit is better than implicit
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \]
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, ... \mid \text{add } e \; e \]
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \]

• What should \text{add } 4 \ 5 \text{ mean?}
Five-Minute Semantics

Grammar: $e ::= 0, 1, 2, \ldots \mid \text{add } e\ e$

- What should $\text{add } 4\ 5$ mean? We have two main options:
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \]

• What should \text{add } 4 \ 5 mean? We have two main options:
  
  ○ Operational: \text{add } 4 \ 5 \text{ means } 9
Grammar:  \[ e ::= 0, 1, 2, \ldots \mid \text{add } e\ e \]

- What should \texttt{add 4 5} mean? We have two main options:
  - **Operational:** \texttt{add 4 5} means 9
  - **Denotational:** \texttt{add 4 5} means \(4 + 5\)
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e e \mid \text{choose } e e \]
Five-Minute Semantics

Grammar: $e ::= 0, 1, 2, ... \mid \text{add } e e \mid \text{choose } e e$

• What should choose 10 20 mean?
Grammar: \[ e ::= 0, 1, 2, \ldots | \text{add } e \ e | \text{choose } e \ e \]

• What should \texttt{choose 10 20} mean? We have two main options:
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, ... \mid \text{add \ } e\ e \mid \text{choose \ } e\ e \]

- What should \textit{choose 10 20} mean? We have two main options:
  - Operational: either 10 or 20 (implementation decides)
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

- What should \text{choose } 10 \ 20 mean? We have two main options:
  - Operational: either 10 or 20 (implementation decides)
  - Denotational: the set \( \{10, 20\} \)
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: define a *semantic function* $\llbracket \cdot \rrbracket$
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: define a \textit{semantic function} \( [\cdot] \)

- Just an outfix function that operates on syntax; e.g.
  \[ [\text{choose } 10 \ 20] = \{10, 20\} \]
Five-Minute Semantics

Grammar:  \[ e ::= 0, 1, 2, \ldots \mid \text{add} \ e \ e \mid \text{choose} \ e \ e \]

Denotational semantics: define a \textit{semantic function} \([\cdot]\)

- Just an outfix function that operates on syntax; e.g.
  \[ [\text{choose} \ 10 \ 20] = \{10, 20\} \]

- What should its return type be?
Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \; e \mid \text{choose } e \; e \]

Denotational semantics: define a *semantic function* \( \cdot \)

- Just an outfit function that operates on syntax; e.g.
  \[
  \llbracket \text{choose } 10 \; 20 \rrbracket = \{10, 20\}
  \]

- What should its return type be?

- Uniform interpretation is good (e.g. Bayesian models always denote the same thing: a joint distribution)
Five-Minute Semantics

Grammar: \[ e ::= \ 0, 1, 2, \ldots \ | \ add \ e \ e \ | \ choose \ e \ e \]

Denotational semantics: define a \textit{semantic function} \( [\cdot] \)

- Just an outfix function that operates on syntax; e.g.
  \[ [\text{choose}\ 10\ 20] = \{10, 20\} \]

- What should its return type be?

- Uniform interpretation is good (e.g. Bayesian models always denote the same thing: a joint distribution)
  - Don’t want \( [10] = 10 \) but \( [\text{choose}\ 10\ 20] = \{10, 20\} \)
Grammar:  \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: define a *semantic function* \([\cdot]\)

- Just an outfix function that operates on syntax; e.g.
  \[ [\text{choose } 10 \ 20] = \{10, 20\} \]

- What should its return type be?

- Uniform interpretation is good (e.g. Bayesian models always denote the same thing: a joint distribution)
  - Don’t want \([10] = 10\) but \([\text{choose } 10 \ 20] = \{10, 20\}\)
  - But \([10] = \{10\}\) would work
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, ... \mid \text{add } e e \mid \text{choose } e e \]

Denotational semantics: \([\cdot] : e \rightarrow \mathcal{P}(\mathbb{N})\), defined by
Five-Minute Semantics

Grammar: \( e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \)

Denotational semantics: \( \llbracket \cdot \rrbracket : e \rightarrow \mathcal{P}(\mathbb{N}) \), defined by

\[
\llbracket n \rrbracket = \{ n \}
\]
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: \[ \llbracket \cdot \rrbracket : e \rightarrow P(\mathbb{N}) \], defined by

\[
\begin{align*}
\llbracket n \rrbracket &= \{n\} \\
\llbracket \text{choose } e_1 \ e_2 \rrbracket &= \llbracket e_1 \rrbracket \cup \llbracket e_2 \rrbracket
\end{align*}
\]
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add} \; e \; e \mid \text{choose} \; e \; e \]

Denotational semantics: \([\cdot]: e \rightarrow \mathcal{P}(\mathbb{N})\), defined by

\[
\begin{align*}
\llbracket n \rrbracket &= \{n\} \\
\llbracket \text{choose} \; e_1 \; e_2 \rrbracket &= \llbracket e_1 \rrbracket \cup \llbracket e_2 \rrbracket \\
\llbracket \text{add} \; e_1 \; e_2 \rrbracket &= \bigcup_{n_1 \in \llbracket e_1 \rrbracket} \bigcup_{n_2 \in \llbracket e_2 \rrbracket} \{n_1 + n_2\}
\end{align*}
\]
Five-Minute Semantics

Grammar: \[ e \ ::= \ 0, 1, 2, \ldots \ | \ add \ e \ e \ | \ choose \ e \ e \]

Denotational semantics: \( [\cdot] : e \rightarrow \mathcal{P}(\mathbb{N}) \), defined by

\[
\begin{align*}
[ n ] &= \ \{ n \} \\
[ \text{choose } e_1 \ e_2 ] &= [ e_1 ] \cup [ e_2 ] \\
[ \text{add } e_1 \ e_2 ] &= \bigcup_{n_1 \in [e_1]} \bigcup_{n_2 \in [e_2]} \{ n_1 + n_2 \}
\end{align*}
\]

Test cases: \([10]\)
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: \( [\cdot] : e \rightarrow P(\mathbb{N}) \), defined by

\[
\begin{align*}
[n] &= \{n\} \\
[\text{choose } e_1 e_2] &= [e_1] \cup [e_2] \\
[\text{add } e_1 e_2] &= \bigcup_{n_1 \in \[e_1\]} \bigcup_{n_2 \in \[e_2\]} \{n_1 + n_2\}
\end{align*}
\]

Test cases: \([10] = \{10\}\)
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, ... \mid \text{add } e\ e \mid \text{choose } e\ e \]

Denotational semantics: \[ [\cdot] : e \rightarrow \mathcal{P}(\mathbb{N}),\ \text{defined by} \]

\[
[ n ] = \{ n \}
\]

\[
[\text{choose } e_1\ e_2] = [ e_1 ] \cup [ e_2 ]
\]

\[
[\text{add } e_1\ e_2] = \bigcup_{n_1 \in [ e_1 ]} \bigcup_{n_2 \in [ e_2 ]} \{ n_1 + n_2 \}
\]

Test cases: \[ [10] = \{10\} \]

\[ [\text{choose } 10\ 20] \]
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \; e \mid \text{choose } e \; e \]

Denotational semantics: \([\cdot] : e \rightarrow \mathcal{P}(\mathbb{N})\), defined by

\[
\begin{align*}
[n] &= \{n\} \\
[\text{choose } e_1 \; e_2] &= [e_1] \cup [e_2] \\
[\text{add } e_1 \; e_2] &= \bigcup_{n_1 \in [e_1]} \bigcup_{n_2 \in [e_2]} \{n_1 + n_2\}
\end{align*}
\]

Test cases: \([10] = \{10\}\)  
\([\text{choose } 10 \; 20] = [10] \cup [20]\)
Five-Minute Semantics

Grammar: \[ e ::= \ 0, 1, 2, \ldots \ | \ add \ e \ e \ | \ choose \ e \ e \]

Denotational semantics: \[ [\cdot] : e \rightarrow \mathcal{P}(\mathbb{N}), \text{defined by} \]

\[
\begin{align*}
[n] &= \{n\} \\
[\text{choose } e_1 \ e_2] &= [e_1] \cup [e_2] \\
[\text{add } e_1 \ e_2] &= \bigcup_{n_1 \in [e_1]} \bigcup_{n_2 \in [e_2]} \{n_1 + n_2\}
\end{align*}
\]

Test cases: \[ [10] = \{10\} \]

\[
\begin{align*}
[\text{choose } 10 \ 20] &= [10] \cup [20] \\
&= \{10\} \cup \{20\} = \{10, 20\}
\end{align*}
\]
Five-Minute Semantics

Grammar: \( e ::= 0, 1, 2, \ldots | \text{add } e \ e | \text{choose } e \ e \)

Denotational semantics: \([\cdot] : e \rightarrow \mathcal{P}(\mathbb{N})\), defined by

\[
\begin{align*}
[n] &= \{n\} \\
[\text{choose } e_1 \ e_2] &= [e_1] \cup [e_2] \\
[\text{add } e_1 \ e_2] &= \bigcup_{n_1 \in [e_1]} \bigcup_{n_2 \in [e_2]} \{n_1 + n_2\}
\end{align*}
\]

Test cases: \([10] = \{10\}\)

\[
\begin{align*}
[\text{choose } 10 \ 20] &= [10] \cup [20] \\
&= \{10\} \cup \{20\} = \{10, 20\}
\end{align*}
\]

\([\text{add } 4 \ 5]\)
Five-Minute Semantics

Grammar: \( e ::= 0, 1, 2, \ldots \mid \text{add } e\ e \mid \text{choose } e\ e \)

Denotational semantics: \( [\cdot] : e \rightarrow \mathcal{P}(\mathbb{N}) \), defined by

\[
\begin{align*}
[n] &= \{n\} \\
[\text{choose } e_1\ e_2] &= [e_1] \cup [e_2] \\
[\text{add } e_1\ e_2] &= \bigcup_{n_1 \in [e_1]} \bigcup_{n_2 \in [e_2]} \{n_1 + n_2\}
\end{align*}
\]

Test cases:

\( [10] = \{10\} \)

\( [\text{choose } 10\ 20] = [10] \cup [20] \)

\( = \{10\} \cup \{20\} = \{10, 20\} \)

\( [\text{add } 4\ 5] = \bigcup_{n_1 \in [4]} \bigcup_{n_1 \in [5]} \{n_1 + n_2\} \)
Five-Minute Semantics

Grammar: \( e ::= 0, 1, 2, \ldots \mid \text{add} \ e \ e \mid \text{choose} \ e \ e \)

Denotational semantics: \( [\cdot] : e \rightarrow \mathcal{P}(\mathbb{N}) \), defined by

\[
\begin{align*}
[n] & = \{n\} \\
\left[\text{choose} \ e_1 \ e_2\right] & = \left[e_1\right] \cup \left[e_2\right] \\
\left[\text{add} \ e_1 \ e_2\right] & = \bigcup_{n_1 \in \left[e_1\right]} \bigcup_{n_2 \in \left[e_2\right]} \{n_1 + n_2\}
\end{align*}
\]

Test cases:

\[
\begin{align*}
[10] & = \{10\} \\
\left[\text{choose} \ 10 \ 20\right] & = \left[10\right] \cup \left[20\right] \\
& = \{10\} \cup \{20\} = \{10, 20\} \\
\left[\text{add} \ 4 \ 5\right] & = \bigcup_{n_1 \in \left[4\right]} \bigcup_{n_1 \in \left[5\right]} \{n_1 + n_2\} \\
& = \bigcup_{n_1 \in \{4\}} \bigcup_{n_1 \in \{5\}} \{n_1 + n_2\}
\end{align*}
\]
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: \[ \cdot : e \rightarrow P(\mathbb{N}), \text{ defined by} \]
\[
\begin{align*}
\llbracket n \rrbracket &= \{n\} \\
\llbracket \text{choose } e_1 \ e_2 \rrbracket &= \llbracket e_1 \rrbracket \cup \llbracket e_2 \rrbracket \\
\llbracket \text{add } e_1 \ e_2 \rrbracket &= \bigcup_{n_1 \in \llbracket e_1 \rrbracket} \bigcup_{n_2 \in \llbracket e_2 \rrbracket} \{n_1 + n_2\}
\end{align*}
\]

Test cases:
\[
\begin{align*}
\llbracket 10 \rrbracket &= \{10\} \\
\llbracket \text{choose } 10 \ 20 \rrbracket &= \llbracket 10 \rrbracket \cup \llbracket 20 \rrbracket \\
&= \{10\} \cup \{20\} = \{10, 20\} \\
\llbracket \text{add } 4 \ 5 \rrbracket &= \bigcup_{n_1 \in \llbracket 4 \rrbracket} \bigcup_{n_1 \in \llbracket 5 \rrbracket} \{n_1 + n_2\} \\
&= \bigcup_{n_1 \in \{4\}} \bigcup_{n_1 \in \{5\}} \{n_1 + n_2\} \\
&= \{4 + 5\} = \{9\}
\end{align*}
\]
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \; e \mid \text{choose } e \; e \]

Denotational semantics: \[ [\cdot] : e \rightarrow \mathcal{P}(\mathbb{N}), \text{ defined by} \]
\[
[\! n \!] = \{n\} \\
[\text{choose } e_1 \; e_2] = [e_1] \cup [e_2] \\
[\text{add } e_1 \; e_2] = \bigcup_{n_1 \in [e_1]} \bigcup_{n_2 \in [e_2]} \{n_1 + n_2\}
\]
Five-Minute Semantics

Grammar:  \( e ::= 0, 1, 2, \ldots \mid \text{add } e e \mid \text{choose } e e \)

Denotational semantics: \( [\cdot] : e \rightarrow \mathcal{P}(\mathbb{N}) \), defined by

\[
\begin{align*}
[n] &= \{n\} \\
[\text{choose } e_1 e_2] &= [e_1] \cup [e_2] \\
[\text{add } e_1 e_2] &= \bigcup_{n_1 \in [e_1]} \bigcup_{n_2 \in [e_2]} \{n_1 + n_2\}
\end{align*}
\]

Distributive property:

\( [\text{add } 4 (\text{choose } 10 20)] = \{14, 24\} \)
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: \( [\cdot] : e \to \mathcal{P}(\mathbb{N}) \), defined by

\[
\begin{align*}
  [n] &= \{n\} \\
  [\text{choose } e_1 e_2] &= [e_1] \cup [e_2] \\
  [\text{add } e_1 e_2] &= \bigcup_{n_1 \in [e_1]} \bigcup_{n_2 \in [e_2]} \{n_1 + n_2\}
\end{align*}
\]

Distributive property:

\[
[\text{add } 4 \ (\text{choose } 10 \ 20)] = \{14, 24\} = [\text{choose } (\text{add } 4 \ 10) \ (\text{add } 4 \ 20)]
\]
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: \[ [\cdot] : e \rightarrow \mathcal{P}(\mathbb{N}), \text{ defined by} \]

\[
\begin{align*}
[n] & = \{n\} \\
[\text{choose } e_1 e_2] & = [e_1] \cup [e_2] \\
[\text{add } e_1 e_2] & = \bigcup_{n_1 \in [e_1]} \bigcup_{n_2 \in [e_2]} \{n_1 + n_2\}
\end{align*}
\]

Distributive property: for all programs \(e_1, e_2\) and \(e_3\),

\[ [\text{add } e_1 (\text{choose } e_2 e_3)] = [\text{choose } (\text{add } e_1 e_2) (\text{add } e_1 e_3)] \]
Five-Minute Semantics

Grammar: \[ e ::= 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: \[ \llbracket \cdot \rrbracket : e \to \mathcal{P}(\mathbb{N}), \text{ defined by} \]
\[
\llbracket n \rrbracket = \{n\}
\]
\[
\llbracket \text{choose } e_1 \ e_2 \rrbracket = \llbracket e_1 \rrbracket \cup \llbracket e_2 \rrbracket
\]
\[
\llbracket \text{add } e_1 \ e_2 \rrbracket = \bigcup_{n_1 \in \llbracket e_1 \rrbracket} \bigcup_{n_2 \in \llbracket e_2 \rrbracket} \{n_1 + n_2\}
\]

Distributive property: for all programs \( e_1, e_2 \) and \( e_3 \),
\[
\llbracket \text{add } e_1 \ (\text{choose } e_2 \ e_3) \rrbracket = \llbracket \text{choose } (\text{add } e_1 \ e_2) \ (\text{add } e_1 \ e_3) \rrbracket
\]

- Can prove this using the semantics
Five-Minute Semantics

Grammar: \[ e ::= \ 0, 1, 2, \ldots \mid \text{add } e \ e \mid \text{choose } e \ e \]

Denotational semantics: \([\cdot] : e \rightarrow \mathcal{P}(\mathbb{N})\), defined by

\[
\begin{align*}
\llbracket n \rrbracket & = \{n\} \\
\llbracket \text{choose } e_1 \ e_2 \rrbracket & = \llbracket e_1 \rrbracket \cup \llbracket e_2 \rrbracket \\
\llbracket \text{add } e_1 \ e_2 \rrbracket & = \bigcup_{n_1 \in \llbracket e_1 \rrbracket} \bigcup_{n_2 \in \llbracket e_2 \rrbracket} \{n_1 + n_2\}
\end{align*}
\]

Distributive property: for all programs \(e_1, e_2\) and \(e_3\),

\[
\llbracket \text{add } e_1 \ (\text{choose } e_2 \ e_3) \rrbracket = \llbracket \text{choose } (\text{add } e_1 \ e_2) \ (\text{add } e_1 \ e_3) \rrbracket
\]

- Can prove this using the semantics
- Could adding a new kind of expression falsify the property?
Dissecting Add+Choose Semantics

• Easy to imagine implementing the language
Dissecting Add+Choose Semantics

- Easy to imagine implementing the language
- Relies only on knowledge of grammars, arithmetic and sets
Dissecting Add+Choose Semantics

- Easy to imagine implementing the language
- Relies only on knowledge of grammars, arithmetic and sets
- Distributive property is just non-obvious enough to motivate having a semantics
Dissecting Add+Choose Semantics

• Easy to imagine implementing the language

• Relies only on knowledge of grammars, arithmetic and sets

• Distributive property is just non-obvious enough to motivate having a semantics

• Shows that semantics aren’t just for running programs
Dissecting Add+Choose Semantics

• Easy to imagine implementing the language

• Relies only on knowledge of grammars, arithmetic and sets

• Distributive property is just non-obvious enough to motivate having a semantics

• Shows that semantics aren’t just for running programs

• Easy to imagine replacing choose with probabilistic choice
Explaining Probability To PL
Explaining Probability To PL

• Avoid:
  ○ Normal, gamma, beta, Dirichlet, other favorites
Explaining Probability To PL

• Avoid:
  ◦ Normal, gamma, beta, Dirichlet, other favorites
  ◦ Philosophy (e.g. “Bayes’ law says what we should believe...”)
Explaining Probability To PL

• Avoid:
  ◦ Normal, gamma, beta, Dirichlet, other favorites
  ◦ Philosophy (e.g. “Bayes’ law says what we should believe...”)
  ◦ Zero-probability conditions (philosophical issues, mathematical baggage)
Explaining Probability To PL

• Avoid:
  ○ Normal, gamma, beta, Dirichlet, other favorites
  ○ Philosophy (e.g. “Bayes’ law says what we should believe...”)
  ○ Zero-probability conditions (philosophical issues, mathematical baggage)

• Do:
  ○ Use coin flips and uniform distributions
Explaining Probability To PL

• Avoid:
  ◦ Normal, gamma, beta, Dirichlet, other favorites
  ◦ Philosophy (e.g. “Bayes’ law says what we should believe...”)
  ◦ Zero-probability conditions (philosophical issues, mathematical baggage)

• Do:
  ◦ Use coin flips and uniform distributions
  ◦ Illustrate with physical processes
Explaining Probability To PL

• Avoid:
  ◦ Normal, gamma, beta, Dirichlet, other favorites
  ◦ Philosophy (e.g. “Bayes’ law says what we should believe...”)
  ◦ Zero-probability conditions (philosophical issues, mathematical baggage)

• Do:
  ◦ Use coin flips and uniform distributions
  ◦ Illustrate with physical processes
  ◦ Draw upon intuition for area and volume
Programming Coin Flips

(let ([x (flip 0.5)]) x)
Programming Coin Flips

(let ([x (flip 0.5)])
  x)

0.5
Programming Coin Flips

(let ([x (flip 0.5)])
  x)
(let ([x (flip 0.5)]
    [y (flip 0.5)])
  (list x y))
(let ([x (flip 0.5)]
    [y (flip 0.5)]))
(list x y)
Programming Coin Flips

(let ([x (flip 0.5)]
      [y (flip 0.5)])
  (list x y))
Programming Coin Flips

(let* ([x (flip 0.5)]
        [y (flip (if (equal? x heads) 0.5 0.3))])
  (list x y))
Programming Coin Flips

```
(let* ([x (flip 0.5)]
       [y (flip (if (equal? x heads) 0.5 0.3))])
  (list x y))
```
Programming Coin Flips

\[ \Pr[\text{true}] = 0.5 \cdot 0.5 + 0.5 \cdot 0.5 + 0.5 \cdot 0.3 + 0.5 \cdot 0.7 = 1 \]
Programming Coin Flips

\[ \Pr[y = \text{heads}] = 0.5 \cdot 0.5 + 0.5 \cdot 0.3 = 0.4 \]
Programming Coin Flips

\[ \Pr[x = \text{heads} \mid y = \text{heads}] \]
\[ = \Pr[\langle x, y \rangle = \langle \text{heads}, \text{heads} \rangle] / \Pr[y = \text{heads}] \]
\[ = 0.25 / 0.4 = 0.625 \]
Stochastic Ray Tracing

**stochastic** /ʃtəˈkæstɪk/ *adj.* fancy word for "randomized"
Stochastic Ray Tracing

**stochastic** /stō-ˈkas-tik/  adj. fancy word for "randomized"
Stochastic Ray Tracing

ap·er·ture  /ˈap-ə(r)-chər/  n. fancy word for "opening"
Stochastic Ray Tracing

ap·er·ture  /ˈap-ə(r)-chər/  n. fancy word for "opening"
Stochastic Ray Tracing

Simulate projecting rays onto a sensor...
Stochastic Ray Tracing

... and collect them to form an image
Stochastic Ray Tracing

Critical: must maintain the distribution of rays
Stochastic Ray Tracing

Critical: must maintain the distribution of rays
Dissecting Stochastic Ray Tracing

• Visual, draws on intuition
Dissecting Stochastic Ray Tracing

• Visual, draws on intuition

• Motivates conditioning
Dissecting Stochastic Ray Tracing

• Visual, draws on intuition

• Motivates conditioning

• Illustrates how rare events make inference hard
Dissecting Stochastic Ray Tracing

- Visual, draws on intuition
- Motivates conditioning
- Illustrates how rare events make inference hard
- Shows why the conditional distribution matters
Dissecting Stochastic Ray Tracing

- Visual, draws on intuition
- Motivates conditioning
- Illustrates how rare events make inference hard
- Shows why the conditional distribution matters
- Motivates sampling methods
Dissecting Stochastic Ray Tracing

- Visual, draws on intuition
- Motivates conditioning
- Illustrates how rare events make inference hard
- Shows why the conditional distribution matters
- Motivates sampling methods
- Motivates PPLs: hand-coded stochastic ray tracer is complicated and hard to get right
Dissecting Stochastic Ray Tracing

- Visual, draws on intuition
- Motivates conditioning
- Illustrates how rare events make inference hard
- Shows why the conditional distribution matters
- Motivates sampling methods
- Motivates PPLs: hand-coded stochastic ray tracer is complicated and hard to get right
- Personal: allows me to distinguish my work
Stochastic Ray Tracing in PPLs

• In DrBayes, it’s simple physics simulation:

```scheme
(define/drbayes (ray-plane-intersect p0 v n d)
 (let ([denom (- (dot v n))])
   (if (> denom 0)
     (let ([t (/ (+ d (dot p0 n)) denom)])
       (if (> t 0)
         (collision t (vec+ p0 (vec* v t)) n)
         #f))
     #f)))
```
Stochastic Ray Tracing in PPLs

• In DrBayes, it’s simple physics simulation:

```
(define/drbayes (ray-plane-intersect p0 v n d)
  (let ([denom ( (dash (dot v n)))]
    (if (> denom 0)
      (let ([t (/ (+ d (dot p0 n)) denom)])
        (if (> t 0)
          (collision t (vec+ p0 (vec* v t)) n)
          #f))
      #f)))
```

• Other PPLs: not possible, or just as hard as in a general-purpose language
Stochastic Ray Tracing in PPLs

• In DrBayes, it’s simple physics simulation:

```
(define/drbayes (ray-plane-intersect p0 v n d)  
  (let ([denom (- (dot v n))])  
    (if (> denom 0)  
      (let ([t (/ (+ d (dot p0 n)) denom)])  
        (if (> t 0)  
          (collision t (vec+ p0 (vec* v t)) n)  
          #f))  
      #f)))
```

• Other PPLs: not possible, or just as hard as in a general-purpose language

• My first PPL: got stuck long before trying this, on simple expressions like \((\text{max } 0.5 \ (\text{random}))\)
Stochastic Ray Tracing in PPLs

• In DrBayes, it’s simple physics simulation:

\[
\text{(define/drbayes (ray-plane-intersect p0 v n d)}
\text{(let ([denom (- (dot v n))])}
\text{(if (> denom 0)}
\text{(let ([t (/ (+ d (dot p0 n)) denom)])}
\text{(if (> t 0)}
\text{(collision t (vec+ p0 (vec* v t)) n) #f))}
\text{#f))})
\]

• Other PPLs: not possible, or just as hard as in a general-purpose language

• My first PPL: got stuck long before trying this, on simple expressions like \(\text{(max 0.5 (random))}\)

• The reasons are almost entirely theoretical
Simpler Example

• Assume \textit{(random)} returns a value uniformly in \([0, 1]\)
• Assume \textbf{(random)} returns a value uniformly in \([0, 1]\)

\textit{Density function} \(p\) for value of \textbf{(random)}:

\begin{itemize}
  \item \begin{tikzpicture}
    \draw[->] (0,0) -- (5,0) node[anchor=north west] {x};
    \draw[->] (0,0) -- (0,5) node[anchor=south east] {p(x)};
    \fill[fill=black!20,draw=black] (0,0) rectangle (5,3);
    \draw[dashed] (5,0) -- (5,3);
    \end{tikzpicture}
\end{itemize}
Simpler Example

• Assume \textbf{(random)} returns a value uniformly in \([0, 1]\)

\textit{Density function} \(p\) for value of \textbf{(random)}:

\[
\Pr[(\text{random}) \in [0.5, 1]] = \int_{0.5}^{1} p(x) \, dx
\]

\[
= 1 - 0.5
\]

\[
= 0.5
\]
Simpler Example

- Assume \( \textbf{(random)} \) returns a value uniformly in \([0, 1]\)

**Density function** \( p \) for value of \( \textbf{(random)} \):

\[
\Pr[\textbf{(random)} \in [0.5, 0.5]] = \int_{0.5}^{0.5} p(x) \, dx
\]

\[
= 0.5 - 0.5
\]

\[
= 0
\]
Simpler Example

• Assume \textbf{(random)} returns a value uniformly in \([0, 1]\)

Density function \(p_m\) for value of \((\text{max} \ 0.5 \ \text{(random)})\):

\[
p_m(x) = ???
\]
Simpler Example

- Assume \textit{(random)} returns a value uniformly in \([0, 1]\).

Density function \(p_m\) for value of \((\max 0.5 \textit{(random)})\):

\[
\Pr[(\max 0.5 \textit{(random)}) \in [0.5, 0.5]] = 0.5
\]
Simpler Example

- Assume \((\text{random})\) returns a value uniformly in \([0, 1]\)

Density function \(p_m\) for value of \((\text{max 0.5 (random)})\):

\[
Pr[(\text{max 0.5 (random)}) \in [0.5, 0.5]] = 0.5
\]

\[
= \int_{0.5}^{0.5} p_m(x) \, dx
\]

\[
= 0
\]
Simpler Example

• Assume \(\text{(random)}\) returns a value uniformly in \([0, 1]\)

Density function \(p_m\) for value of \((\max 0.5 \text{ (random)})\):

\[
\Pr[(\max 0.5 \text{ (random)}) \in [0.5, 0.5]] = 0.5
\]

\[
= \int_{0.5}^{0.5} p_m(x) \, dx
\]

\[
= 0
\]
Technical Gap: Limited Theory of Probability
Technical Gap: Limited Theory of Probability

• Densities can’t model...

  ○ Discontinuous functions (no bounded measuring devices)

    (let ([temperature (normal 99 1)])
        (min 100 temperature))
Technical Gap: Limited Theory of Probability

• Densities can’t model...
  
  ◦ Discontinuous functions (no bounded measuring devices)
    
    (let ([temperature (normal 99 1)])
      (min 100 temperature))
  
  ◦ Variable-dimensional things (no union types)
    
    (cond [(flip 0.5) (list (random))]
      [else (list (random) (random))])
Technical Gap: Limited Theory of Probability

• Densities can’t model...
  
  ◦ Discontinuous functions (no bounded measuring devices)
    
    (let ([temperature (normal 99 1)])
    (min 100 temperature))

  ◦ Variable-dimensional things (no union types)
    
    (cond [(flip 0.5) (list (random))]
    [else (list (random) (random))])

  ◦ Infinite-dimensional things (no streams, recursion)
Technical Gap: Limited Theory of Probability

• Densities can’t model...

  ◦ Discontinuous functions (no bounded measuring devices)

    \[
    \text{(let ([temperature (normal 99 1)])}\\
    \quad (\text{min 100 temperature}))
    \]

  ◦ Variable-dimensional things (no union types)

    \[
    \text{(cond [(flip 0.5) (list (random))]}\\
    \quad [\text{else} (\text{list (random) (random))}])
    \]

  ◦ Infinite-dimensional things (no streams, recursion)

• Measure theory handles them all, but its generality makes finding computational content difficult
Probability Measures

• Like already-integrated densities, but a primitive concept
Probability Measures

• Like already-integrated densities, but a primitive concept

• Measure of (random) is $P : \mathcal{P}([0, 1]) \rightarrow [0, 1]$, defined by

$$P([a, b]) = \int_{a}^{b} p(x) \, dx = b - a$$
Probability Measures

• Like already-integrated densities, but a primitive concept

• Measure of \textbf{(random)} is $P : \mathcal{P}([0, 1]) \rightarrow [0, 1]$, defined by

$$P([a, b]) = b - a$$
Probability Measures

- Like already-integrated densities, but a primitive concept

- Measure of \textbf{(random)} is \( P : \mathcal{P}([0, 1]) \rightarrow [0, 1] \), defined by
  \[
P([a, b]) = b - a
  \]

- Measure of \textbf{(max 0.5 (random))} defined by
  \[
P_m([a, b]) = \max(0.5, b) - \max(0.5, a) + \begin{cases} 
  0.5 & \text{if } a \leq 0.5 \leq b \\
  0 & \text{otherwise}
\end{cases}
  \]
Probability Measures

• Like already-integrated densities, but a primitive concept

• Measure of (random) is $P : \mathcal{P}([0, 1]) \rightarrow [0, 1]$, defined by

$$P([a, b]) = b - a$$

• Measure of (max 0.5 (random)) defined by

$$P_m([a, b]) = \max(0.5, b) - \max(0.5, a) + \begin{cases} 
0.5 & \text{if } a \leq 0.5 \leq b \\
0 & \text{otherwise}
\end{cases}$$

This term assigns $[0.5, 0.5]$ probability 0.5
Probability Measures

• Like already-integrated densities, but a primitive concept

• Measure of (random) is $P : \mathcal{P}([0, 1]) \rightarrow [0, 1]$, defined by

$$P([a, b]) = b - a$$

• Measure of (max 0.5 (random)) defined by

$$P_m([a, b]) = \max(0.5, b) - \max(0.5, a) + \begin{cases} 
0.5 & \text{if } a \leq 0.5 \leq b \\
0 & \text{otherwise} 
\end{cases}$$

This term assigns $[0.5, 0.5]$ probability 0.5

• Bridge the gap: derive measures from code, then compute them
Probability Measures Via Preimages

• Interpret \((\max \ 0.5 \ (\text{random}))\) as \(f : [0, 1] \rightarrow \mathbb{R}\), defined

\[
    f(r) = \max(0.5, r)
\]
Probability Measures Via Preimages

• Interpret \((\max 0.5 \text{ (random)})\) as \(f : [0, 1] \rightarrow \mathbb{R}\), defined

\[ f(r) = \max(0.5, r) \]

• Derive measure of \((\max 0.5 \text{ (random)})\) as

\[ P_{m}(B) = P(f^{-1}(B)) \]
Probability Measures Via Preimages

• Interpret \((\max 0.5 \text{ (random)})\) as \(f : [0, 1] \rightarrow \mathbb{R}\), defined

\[
f(r) = \max(0.5, r)
\]

• Derive measure of \((\max 0.5 \text{ (random)})\) as

\[
P_m(B) = P(f^{-1}(B))
\]

where \(f^{-1}(B) = \{r \in [0, 1] \mid f(r) \in B\}\)
Probability Measures Via Preimages

• Interpret \((\max 0.5 \text{ (random)})\) as \(f : [0, 1] \rightarrow \mathbb{R}\), defined

\[
f(r) = \max(0.5, r)
\]

• Derive measure of \((\max 0.5 \text{ (random)})\) as

\[
P_m(B) = P(f^{-1}(B))
\]

where \(f^{-1}(B) = \{r \in [0, 1] \mid f(r) \in B\}\)

• Factored into random and deterministic parts:

\[
P_m = P \circ f^{-1}
\]
Probability Measures Via Preimages

- Interpret \((\max 0.5 (\text{random}))\) as \(f : [0, 1] \rightarrow \mathbb{R}\), defined
  \[f(r) = \max(0.5, r)\]

- Derive measure of \((\max 0.5 (\text{random}))\) as
  \[P_m(B) = P(f^{-1}(B))\]
  where \(f^{-1}(B) = \{r \in [0, 1] \mid f(r) \in B\}\)

- Factored into random and deterministic parts:
  \[P_m = P \circ f^{-1}\]

- In other words, compute measures of expressions by running them backwards
Probability Measures Via Preimages

• Interpret \( \text{max} \ 0.5 \ (\text{random}) \) as \( f : [0, 1] \rightarrow \mathbb{R} \), defined

\[
f(r) = \max(0.5, r)
\]

• Derive measure of \( \text{max} \ 0.5 \ (\text{random}) \) as

\[
P_m(B) = P(f^{-1}(B))
\]

where \( f^{-1}(B) = \{ r \in [0, 1] \mid f(r) \in B \} \)

• Factored into random and deterministic parts:

\[
P_m = P \circ f^{-1}
\]

• In other words, compute measures of expressions by running them backwards... on possibly uncountable sets
What About Approximating?

Conservative approximation with rectangles:
What About Approximating?

Conservative approximation with rectangles:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Restricting preimages to rectangular subdomains:
What About Approximating?

Sampling: exponential to quadratic (e.g. days to minutes)
What About Approximating?

Sampling: exponential to quadratic (e.g. days to minutes)
Culture Gap: Ideals and Motivation

• Jay: “Just write something for practice, and I’ll review it.”
Culture Gap: Ideals and Motivation

• Jay: “Just write something for practice, and I’ll review it.”

• Chosen topics
Culture Gap: Ideals and Motivation

- Jay: “Just write something for practice, and I’ll review it.”

- Chosen topics
  - Probability is the only good way to model uncertainty
Culture Gap: Ideals and Motivation

- Jay: “Just write something for practice, and I’ll review it.”

- Chosen topics
  - Probability is the only good way to model uncertainty
  - Why sound approximations don’t scale, but sampling does
Culture Gap: Ideals and Motivation

- Jay: “Just write something for practice, and I’ll review it.”

- Chosen topics
  - Probability is the only good way to model uncertainty
  - Why sound approximations don’t scale, but sampling does

- Jay: “This is terrible.”
Ideals and Motivation: PL vs. ML
### Ideals and Motivation: PL vs. ML

<table>
<thead>
<tr>
<th>PL</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.999% certainty isn’t enough</td>
<td></td>
</tr>
</tbody>
</table>
## Ideals and Motivation: PL vs. ML

<table>
<thead>
<tr>
<th>PL</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.999% certainty isn’t enough</td>
<td>Even 99% certainty isn’t feasible</td>
</tr>
<tr>
<td><strong>PL</strong></td>
<td><strong>ML</strong></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>99.999% certainty isn’t enough</td>
<td>Even 99% certainty isn’t feasible</td>
</tr>
<tr>
<td>Only sound approximations are admissible: we must provide guarantees</td>
<td></td>
</tr>
</tbody>
</table>
# Ideals and Motivation: PL vs. ML

<table>
<thead>
<tr>
<th>PL</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.999% certainty isn’t enough</td>
<td>Even 99% certainty isn’t feasible</td>
</tr>
<tr>
<td>Only sound approximations are admissible: we must provide guarantees</td>
<td>Logical reasoning doesn’t scale, so we can’t provide guarantees</td>
</tr>
<tr>
<td><strong>Ideals and Motivation: PL vs. ML</strong></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>PL</strong></td>
<td><strong>ML</strong></td>
</tr>
<tr>
<td>99.999% certainty isn’t enough</td>
<td>Even 99% certainty isn’t feasible</td>
</tr>
<tr>
<td>Only sound approximations are</td>
<td>Logical reasoning doesn’t scale, so</td>
</tr>
<tr>
<td>admissible: we must provide guarantees</td>
<td>we can’t provide guarantees</td>
</tr>
<tr>
<td>Correctness on all inputs is</td>
<td></td>
</tr>
<tr>
<td>desirable and achievable</td>
<td></td>
</tr>
</tbody>
</table>
# Ideals and Motivation: PL vs. ML

<table>
<thead>
<tr>
<th>PL</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.999% certainty isn’t enough</td>
<td>Even 99% certainty isn’t feasible</td>
</tr>
<tr>
<td>Only sound approximations are admissible: we must provide guarantees</td>
<td>Logical reasoning doesn’t scale, so we can’t provide guarantees</td>
</tr>
<tr>
<td>Correctness on all inputs is desirable and achievable</td>
<td>Perfection in one domain usually implies poor generalization</td>
</tr>
</tbody>
</table>
## Ideals and Motivation: PL vs. ML

<table>
<thead>
<tr>
<th>PL</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.999% certainty isn’t enough</td>
<td>Even 99% certainty isn’t feasible</td>
</tr>
<tr>
<td>Only sound approximations are admissible: we must provide guarantees</td>
<td>Logical reasoning doesn’t scale, so we can’t provide guarantees</td>
</tr>
<tr>
<td>Correctness on all inputs is desirable and achievable</td>
<td>Perfection in one domain usually implies poor generalization</td>
</tr>
</tbody>
</table>

All of the above statements are true
# Ideals and Motivation: PL vs. ML

<table>
<thead>
<tr>
<th>PL</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.999% certainty isn’t enough</td>
<td>Even 99% certainty isn’t feasible</td>
</tr>
<tr>
<td>Only sound approximations are admissible: we must provide guarantees</td>
<td>Logical reasoning doesn’t scale, so we can’t provide guarantees</td>
</tr>
<tr>
<td>Correctness on all inputs is desirable and achievable</td>
<td>Perfection in one domain usually implies poor generalization</td>
</tr>
</tbody>
</table>

All of the above statements are true **in context**
PL Research Context Example

- Example: Creating a language for writing EULAs
PL Research Context Example

• Example: Creating a language for writing EULAs

• Named entities such as THE SOFTWARE and ACME CORPORATION, based on well-known research on contracts
PL Research Context Example

• Example: Creating a language for writing EULAs

• Named entities such as THE SOFTWARE and ACME CORPORATION, based on well-known research on contracts

• Compiler creates daemons that answer queries about actions

  Can file X be copied by program Y? Can personal data Z be uploaded to ACME CORPORATION?
PL Research Context Example

- Example: Creating a language for writing EULAs

- Named entities such as **THE SOFTWARE** and **ACME CORPORATION**, based on well-known research on contracts

- Compiler creates daemons that answer queries about actions
  
  Can file **X** be copied by program **Y**? Can personal data **Z** be uploaded to **ACME CORPORATION**?

- Query language meticulously defined in terms of contracts
PL Research Context Example

• Example: Creating a language for writing EULAs

• Named entities such as **THE SOFTWARE** and **ACME CORPORATION**, based on well-known research on contracts

• Compiler creates daemons that answer queries about actions

  Can file **X** be copied by program **Y**? Can personal data **Z** be uploaded to **ACME CORPORATION**?

• Query language meticulously defined in terms of contracts

• Soundness means “always denies a contract-breaking action”
PL Research Context Example

- Example: Creating a language for writing EULAs

- Named entities such as THE SOFTWARE and ACME CORPORATION, based on well-known research on contracts

- Compiler creates daemons that answer queries about actions

  Can file X be copied by program Y? Can personal data Z be uploaded to ACME CORPORATION?

- Query language meticulously defined in terms of contracts

- Soundness means “always denies a contract-breaking action”

- Becomes used as target language for compilers for all kinds of human contracts
PL Research Context

- Research context characterized by
PL Research Context

- Research context characterized by
  1. Well-defined problems
PL Research Context

• Research context characterized by
  1. Well-defined problems
  2. Languages are used in the middle of a pile of abstractions
PL Research Context

- Research context characterized by
  1. Well-defined problems
  2. Languages are used in the middle of a pile of abstractions
- Therefore:
PL Research Context

• Research context characterized by

  1. Well-defined problems

  2. Languages are used in the middle of a pile of abstractions

• Therefore:

  1. Correctness is achievable
PL Research Context

• Research context characterized by

  1. Well-defined problems
  2. Languages are used in the middle of a pile of abstractions

• Therefore:

  1. Correctness is achievable
  2. Languages must provide upward guarantees
ML Research Context Example

• Example: Creating a classifier that determines whether actions violate ACME’s EULAs
ML Research Context Example

• Example: Creating a classifier that determines whether actions violate ACME’s EULAs

• Data sources: keystrokes, hard drive contents, running processes, webcam, examples of users violating and not violating EULAs
ML Research Context Example

• Example: Creating a classifier that determines whether actions violate ACME’s EULAs

• Data sources: keystrokes, hard drive contents, running processes, webcam, examples of users violating and not violating EULAs

• Hand-written watchdog daemon tries to determine when users are doing something shady
ML Research Context Example

• Example: Creating a classifier that determines whether actions violate ACME’s EULAs

• Data sources: keystrokes, hard drive contents, running processes, webcam, examples of users violating and not violating EULAs

• Hand-written watchdog daemon tries to determine when users are doing something shady

• Correct iff its reported violations are real violations
ML Research Context Example

• Example: Creating a classifier that determines whether actions violate ACME’s EULAs

• Data sources: keystrokes, hard drive contents, running processes, webcam, examples of users violating and not violating EULAs

• Hand-written watchdog daemon tries to determine when users are doing something shady

• Correct iff its reported violations are real violations

• This system is clearly AI-complete
ML Research Context

• Research context characterized by
ML Research Context

- Research context characterized by
  1. Ridiculously underconstrained (i.e. really hard) problems
ML Research Context

- Research context characterized by

  1. Ridiculously underconstrained (i.e. really hard) problems

  2. Algorithms used near the top of the pile of abstractions
ML Research Context

• Research context characterized by

  1. Ridiculously underconstrained (i.e. really hard) problems
  2. Algorithms used near the top of the pile of abstractions

• Therefore:
ML Research Context

• Research context characterized by
  1. Ridiculously underconstrained (i.e. really hard) problems
  2. Algorithms used near the top of the pile of abstractions

• Therefore:
  1. Correctness is almost never achievable, but strong assumptions enable good performance anyway
ML Research Context

• Research context characterized by
  1. Ridiculously underconstrained (i.e. really hard) problems
  2. Algorithms used near the top of the pile of abstractions

• Therefore:
  1. Correctness is almost never achievable, but strong assumptions enable good performance anyway
  2. Errors can be tolerated
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

\[ [\cdot] : e \rightarrow (R \rightarrow B) \]
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

\[
\cdot : e \rightarrow (R \rightarrow B) 
\]

assumed
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

\[
\begin{align*}
[\cdot]_{\text{pre}} &: \epsilon \rightarrow \text{preimage functions} & \text{correct} \\
[\cdot] &: \epsilon \rightarrow (R \rightarrow B) & \text{assumed}
\end{align*}
\]
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

\[
\begin{align*}
\llbracket \cdot \rrbracket_{\text{pre}} : e & \rightarrow \text{approximating preimage functions} \\
\llbracket \cdot \rrbracket_{\text{pre}} : e & \rightarrow \text{preimage functions} \\
\llbracket \cdot \rrbracket & : e \rightarrow (R \rightarrow B)
\end{align*}
\]
Closing the Gap in PPLs

- PL ideals: Languages must be correct, provide guarantees
- ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

\[
\begin{align*}
\mathbb{[} \cdot \mathbb{]}_{\text{pre}} & : e \to \text{approximating preimage functions} & \text{sound} \\
\mathbb{[} \cdot \mathbb{]}_{\text{pre}} & : e \to \text{preimage functions} & \text{correct} \\
\mathbb{[} \cdot \mathbb{]} & : e \to (R \to B) & \text{assumed}
\end{align*}
\]
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

Rectangular set library, inverse interval arithmetic

\[
\begin{array}{ll}
\preimage : e \to \text{approximating preimage functions} & \text{sound} \\
\preimage : e \to \text{preimage functions} & \text{correct} \\
\cdot : e \to (R \to B) & \text{assumed}
\end{array}
\]
Closing the Gap in PPLs

- PL ideals: Languages must be correct, provide guarantees
- ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

| Rectangular set library, inverse interval arithmetic | sound |
| [\cdot]_{\text{pre}} : e \rightarrow \text{approximating preimage functions} | sound |
| [\cdot]_{\text{pre}} : e \rightarrow \text{preimage functions} | correct |
| [\cdot] : e \rightarrow (R \rightarrow B) | assumed |
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

<table>
<thead>
<tr>
<th>Rectangular set library, inverse interval arithmetic</th>
<th>sound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[\cdot]_{\text{pre}} : e \rightarrow \text{approximating preimage functions}$</td>
<td>sound</td>
</tr>
<tr>
<td>$[\cdot]_{\text{pre}} : e \rightarrow \text{preimage functions}$</td>
<td>correct</td>
</tr>
<tr>
<td>$[\cdot] : e \rightarrow (R \rightarrow B)$</td>
<td>assumed</td>
</tr>
</tbody>
</table>
Closing the Gap in PPLs

• PL ideals: Languages must be correct, provide guarantees

• ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

<table>
<thead>
<tr>
<th>SOUNDNESS BARRIER</th>
<th>DO NOT CROSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangular set library, inverse interval arithmetic</td>
<td>sound</td>
</tr>
<tr>
<td>$\mathcal{L}_{\text{pre}} : e \to \text{approximating preimage functions}$</td>
<td>sound</td>
</tr>
<tr>
<td>$\mathcal{L}_{\text{pre}} : e \to \text{preimage functions}$</td>
<td>correct</td>
</tr>
<tr>
<td>$\mathcal{L} : e \to (R \to B)$</td>
<td>assumed</td>
</tr>
</tbody>
</table>
Closing the Gap in PPLs

- PL ideals: Languages must be correct, provide guarantees
- ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

Sampling algorithms

---

**SOUNDNESS BARRIER**

- Rectangular set library, inverse interval arithmetic: **sound**
- $\preimage{\cdot} : e \rightarrow \text{approximating preimage functions}$: **sound**
- $\preimage{\cdot} : e \rightarrow \text{preimage functions}$: **correct**
- $\cdot : e \rightarrow (R \rightarrow B)$: **assumed**
Closing the Gap in PPLs

- **PL ideals**: Languages must be correct, provide guarantees
- **ML ideals**: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

<table>
<thead>
<tr>
<th>Sampling algorithms</th>
<th>converge in probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangular set library, inverse interval arithmetic</td>
<td>sound</td>
</tr>
<tr>
<td>$\lbrack \cdot \rbrack_{\text{pre}} : e \rightarrow$ approximating preimage functions</td>
<td>sound</td>
</tr>
<tr>
<td>$\lbrack \cdot \rbrack_{\text{pre}} : e \rightarrow$ preimage functions</td>
<td>correct</td>
</tr>
<tr>
<td>$\lbrack \cdot \rbrack : e \rightarrow (R \rightarrow B)$</td>
<td>assumed</td>
</tr>
</tbody>
</table>
Motivating Soundness To ML
Motivating Soundness To ML

![Diagram of two axes with shaded regions](image)
Motivating Soundness To ML
Motivating Soundness To ML
Motivating Soundness To ML
Motivating Soundness To ML
Motivating Soundness To ML

![Graph](image-url)
Motivating Soundness To ML
Motivating Soundness To ML
Motivating Soundness To ML
Motivating Soundness To ML

Convergence requires soundness of $\widehat{\mathcal{C}}_{\text{pre}}$
Motivating Unsoundness to PL

• Harder than motivating soundness to ML
Motivating Unsoundness to PL

• Harder than motivating soundness to ML

• Actual conversation (last week):

  Me: “Jay, when, if ever, did you accept sampling as a viable way to represent the meaning of programs?”
Motivating Unsoundness to PL

• Harder than motivating soundness to ML

• Actual conversation (last week):

  Me: “Jay, when, if ever, did you accept sampling as a viable way to represent the meaning of programs?”

  Jay: “I believe in sampling because you believe in sampling.”
Motivating Unsoundness to PL

• Harder than motivating soundness to ML

• Actual conversation (last week):

  Me: “Jay, when, if ever, did you accept sampling as a viable way to represent the meaning of programs?”

  Jay: “I believe in sampling because you believe in sampling.”

• Start small: “There are people out there that want this.”
Culture Gap: Prejudice

pre·ju·dice  /ˈpre-ʒu-dəs/  n. preconceived opinion that is not based on reason or actual experience
Culture Gap: Prejudice

pre·ju·dice  /ˈpre-ʒə-dəs/  n. preconceived opinion that is not based on reason or actual experience

Requires only applying a *good ideal* in the wrong context
Culture Gap: Prejudice

pre·ju·dice  /ˈpre-ju-dəs/  n. preconceived opinion that is not based on reason or actual experience

Requires only applying a *good ideal* in the wrong context

We must provide sound guarantees

Sound approximations are the only useful approximations
Culture Gap: Prejudice

**pre-jú-dice** /ˈpre-ʒə-dəs/  *n.* preconceived opinion that is not based on reason or actual experience

Requires only applying a *good ideal* in the wrong context

We must provide sound guarantees

Sound approximations are the only useful approximations

I know what my programs mean well enough to write great programs

Semantics are unnecessary to make a reliable probabilistic language
Addressing Prejudice

• Sound approximations are the only useful approximations
Addressing Prejudice

• Sound approximations are the only useful approximations
  ◦ Overwhelming evidence against
Addressing Prejudice

- Sound approximations are the only useful approximations
  - Overwhelming evidence against at the top of the abstraction stack
Addressing Prejudice

• Sound approximations are the only useful approximations
  ◦ Overwhelming evidence against *at the top of the abstraction stack*

• Semantics are unnecessary to make a probabilistic language
Addressing Prejudice

• Sound approximations are the only useful approximations
  ◦ Overwhelming evidence against *at the top of the abstraction stack*

• Semantics are unnecessary to make a probabilistic language
  ◦ Reminder: Monty Hall paradox, two envelopes paradox, friendship paradox, explaining away, etc., etc., etc.
Addressing Prejudice

• Sound approximations are the only useful approximations
  ◦ Overwhelming evidence against *at the top of the abstraction stack*

• Semantics are unnecessary to make a probabilistic language
  ◦ Reminder: Monty Hall paradox, two envelopes paradox, friendship paradox, explaining away, etc., etc., etc.
  ◦ If your PPL never surprises you, it’s almost certainly wrong
New: Interval Paradox

- DrBayes doesn’t allow zero-probability conditions: must use intervals instead of equality
New: Interval Paradox

- DrBayes doesn’t allow zero-probability conditions: must use intervals instead of equality

- Standard normal-normal model with two observations:

```scheme
(let* ([x (normal 0 1)]
       [y1 (normal x 1)]
       [y2 (normal x 1)])
  (condition x (and (<= (- 2 \(\epsilon\)1) y1 (+ 2 \(\epsilon\)1))
                   (<= (- -1 \(\epsilon\)2) y2 (+ -1 \(\epsilon\)2)))))
```
New: Interval Paradox

- DrBayes doesn’t allow zero-probability conditions: must use intervals instead of equality

- Standard normal-normal model with two observations:

```scheme
(let* ([x (normal 0 1)]
       [y1 (normal x 1)]
       [y2 (normal x 1)])
  (condition x (and (<= (- 2 e1) y1 (+ 2 e1))
                   (<= (- -1 e2) y2 (+ -1 e2)))))
```

- $e1$ and $e2$ control widths of intervals
New: Interval Paradox

• DrBayes doesn’t allow zero-probability conditions: must use intervals instead of equality

• Standard normal-normal model with two observations:

\[
\text{(let* ([x (normal 0 1)]
[\ y1 (normal x 1)])
[\ y2 (normal x 1)])
\text{(condition x (and (<= (- 2 \varepsilon 1) y1 (+ 2 \varepsilon 1))
(<= (- -1 \varepsilon 2) y2 (+ -1 \varepsilon 2))))})
\]

• \(\varepsilon_1\) and \(\varepsilon_2\) control widths of intervals

• Seems like width or at least proportionality should matter...
Interval Paradox

- True Density
- $\varepsilon_1 = 0.2, \varepsilon_2 = 0.01$
- $\varepsilon_1 = 0.01, \varepsilon_2 = 0.2$
Interval Paradox
Interval Paradox

- Width matters a little, but proportionality doesn’t
Interval Paradox

- Width matters a little, but proportionality doesn’t
- General property due to Lebesgue differentiation theorem
Technical Gap: Algorithms

• Probabilistic program distributions are weird
Technical Gap: Algorithms

• Probabilistic program distributions are weird
  
  ◦ One point on support of program domain distribution:

```
  0.12351...
   /     \
0.52198...  0.33780...
   /     \
0.92462...  0.52309...  0.00143...  0.99264...
```
Technical Gap: Algorithms

• Probabilistic program distributions are weird
  
  ◦ One point on support of program domain distribution:

• Current inference algorithms developed for very different distributions
Technical Gap: Algorithms

• Probabilistic program distributions are weird
  ○ One point on support of program domain distribution:

• Current inference algorithms developed for very different distributions
• PPLs can provide extra information
Summary

• Moving from ML to PL was a culture shock due to culture gaps
Summary

• Moving from ML to PL was a culture shock due to culture gaps
  ◦ Vocabulary, formalisms, ideals and motivations, prejudice
Summary

• Moving from ML to PL was a culture shock due to culture gaps
  ◦ Vocabulary, formalisms, ideals and motivations, prejudice
• These very gaps keep us from working together, but can be closed and bridged
Summary

• Moving from ML to PL was a culture shock due to culture gaps
  ○ Vocabulary, formalisms, ideals and motivations, prejudice

• These very gaps keep us from working together, but can be closed and bridged

• PPLs are hard to create due to technical gaps
Summary

• Moving from ML to PL was a culture shock due to culture gaps
  ○ Vocabulary, formalisms, ideals and motivations, prejudice
• These very gaps keep us from working together, but can be closed and bridged
• PPLs are hard to create due to technical gaps
  ○ Limited theory of probability, limitations of PL formalisms (didn’t address today), existing sampling algorithms struggle
Summary

• Moving from ML to PL was a culture shock due to culture gaps
  ◦ Vocabulary, formalisms, ideals and motivations, prejudice

• These very gaps keep us from working together, but can be closed and bridged

• PPLs are hard to create due to technical gaps
  ◦ Limited theory of probability, limitations of PL formalisms (didn’t address today), existing sampling algorithms struggle

• But these very gaps are opportunities for foundational research