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

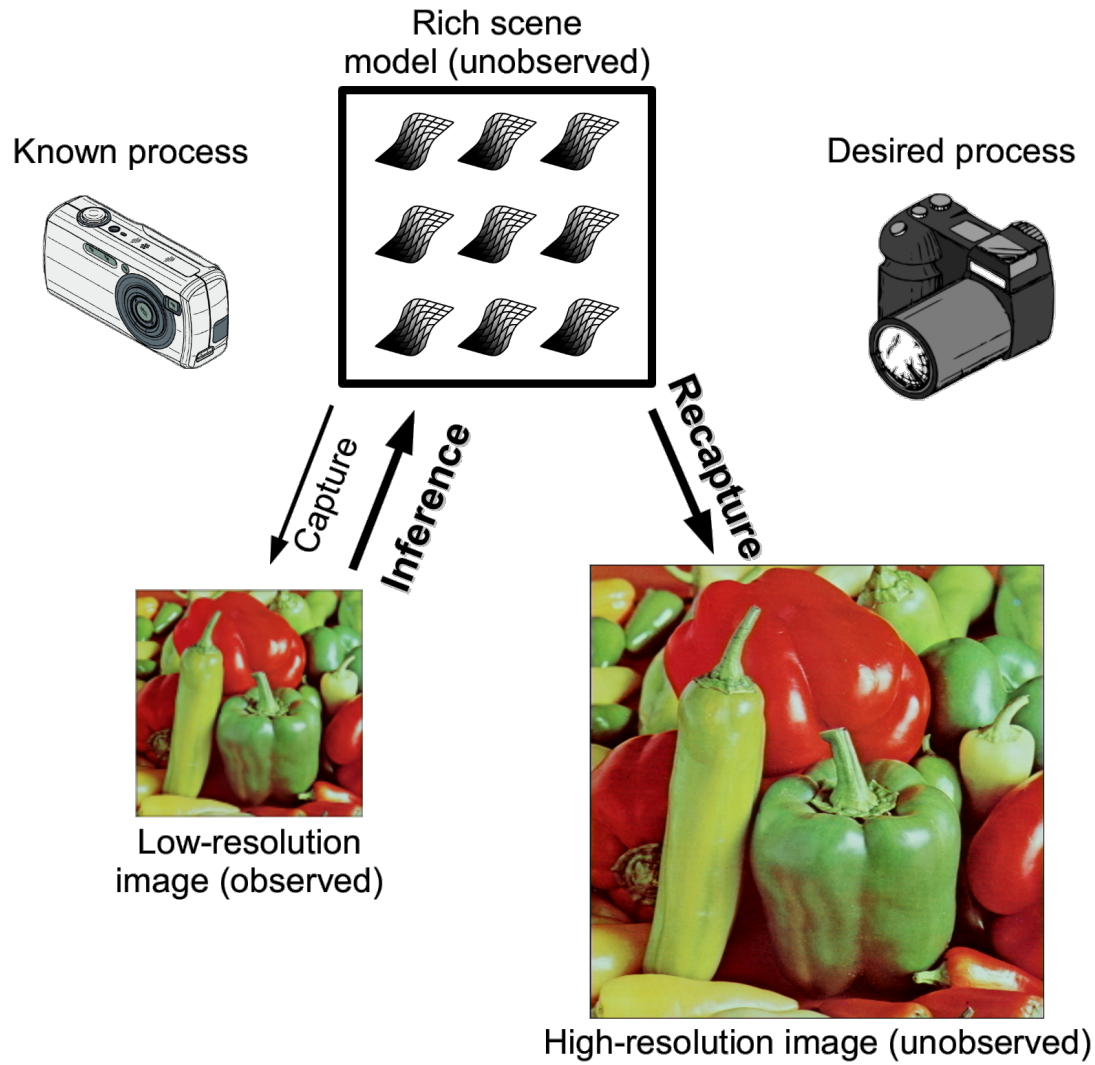
- Illustrate part of the intersection of machine learning (ML) and programming languages (PL), and the technical and cultural gaps
- Why: Everyone knows we have a hard time working together
- “Neil, that’s a dangerous talk.”

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)...

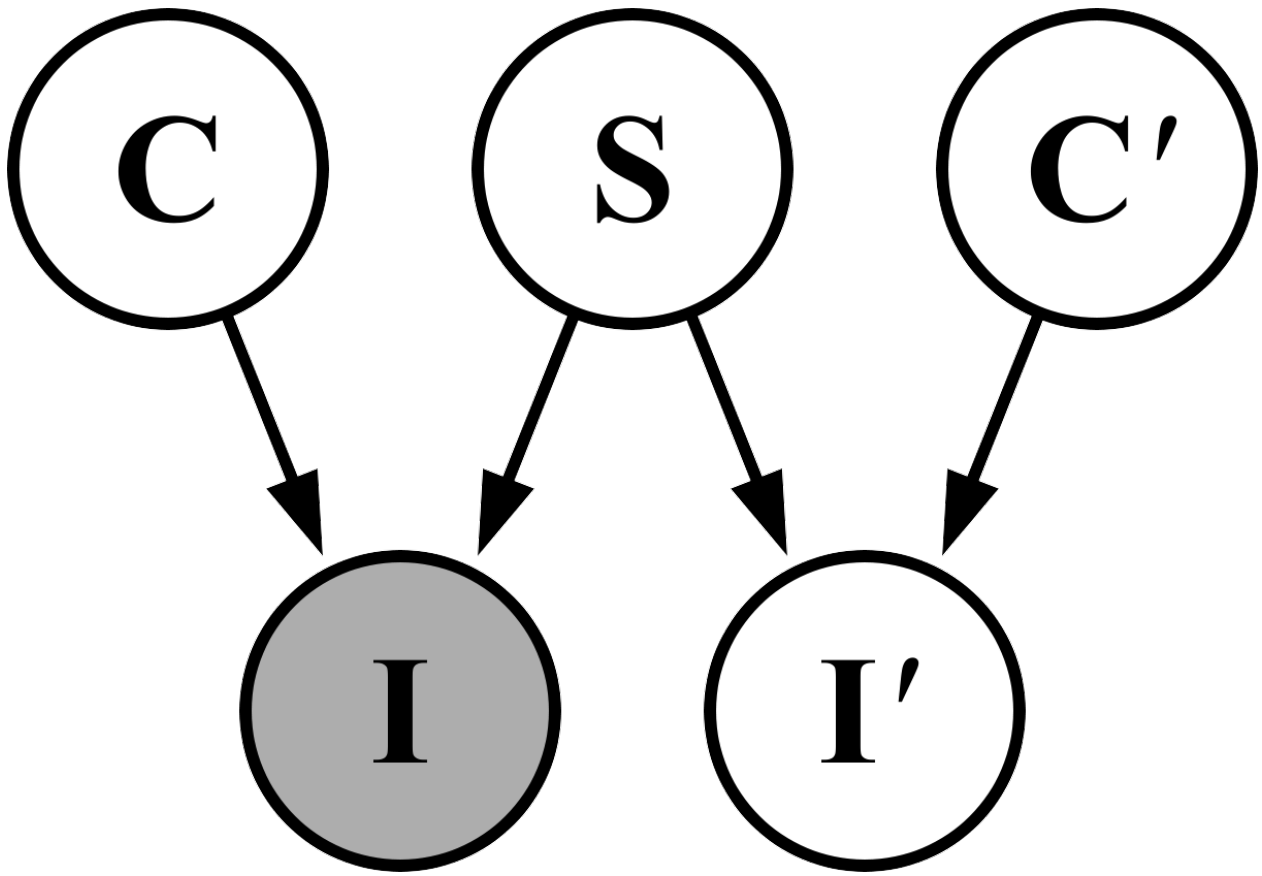
Early Years

Master's Research: Super-Resolution



Toronto, Morse, Seppi, Ventura. *Super-Resolution via Recapture and Bayesian Effect Modeling.* CVPR 2009

Master's Research: Super-Resolution



Toronto, Morse, Seppi, Ventura. *Super-Resolution via Recapture and Bayesian Effect Modeling.* CVPR 2009

Master's Research: Math

Half a page of beautiful math

$$\mathbf{C}^x_{i,j} \equiv i + \tfrac{1}{2} \qquad i \in 0..m-1$$
$$\mathbf{C}^y_{i,j} \equiv j + \tfrac{1}{2} \qquad j \in 0..n-1$$

$$\mathrm{N9}(x,y) \equiv \{i \in \mathbb{Z} \mid -1 \le i - \lfloor x \rfloor \le 1\} \times \{j \in \mathbb{Z} \mid -1 \le j - \lfloor y \rfloor \le 1\}$$

$$\mathrm{dist}(x,y,\theta,d) \equiv x\cos\theta + y\sin\theta - d$$

$$\mathrm{prof}(d,\sigma,v^+,v^-) \equiv \frac{v^+ - v^-}{2}\,\mathrm{erf}\left(\frac{d}{\sqrt{2}\sigma}\right) + \frac{v^+ + v^-}{2}$$

$$\mathrm{edge}(x,y,\theta,d,v^+,v^-,\sigma) \equiv \mathrm{prof}(\mathrm{dist}(x,y,\theta,d),\sigma,v^+,v^-)$$

$$\mathbf{S}^{\mathrm{edge}}_{i,j}(x,y) \equiv \mathrm{edge}(x - \mathbf{C}^x_{i,j}, y - \mathbf{C}^y_{i,j}, \mathbf{S}^\theta_{i,j}, \mathbf{S}^d_{i,j}, \mathbf{S}^{v^+}_{i,j}, \mathbf{S}^{v^-}_{i,j}, \mathbf{S}^\sigma_{i,j})$$

$$\mathrm{E}[h(\mathbf{S}_{x,y})] \equiv \sum_{k,l \in \mathrm{N9}(x,y)} w(x - \mathbf{C}^x_{k,l}, y - \mathbf{C}^y_{k,l})\; h(\mathbf{S}^{\mathrm{edge}}_{k,l}(x,y))$$

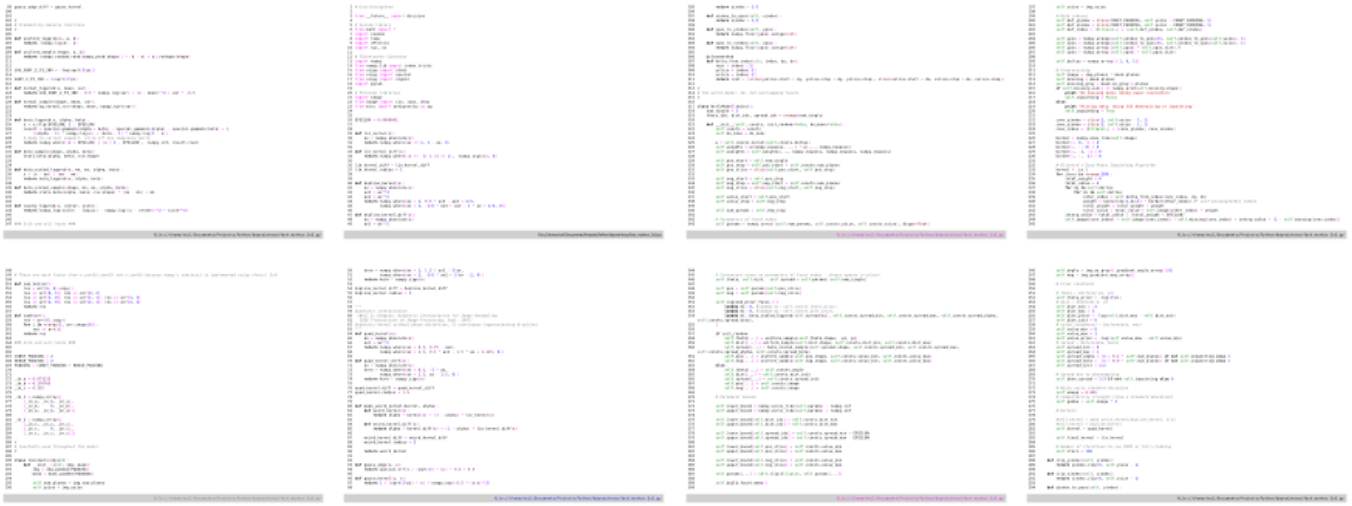
$$\mathbf{S}^\theta_{i,j} \sim \mathrm{Uniform}(-\pi,\pi) \qquad \mathbf{S}^{v^+}_{i,j} \sim \mathrm{Uniform}(0,1)$$
$$\mathbf{S}^d_{i,j} \sim \mathrm{Uniform}(-3,3) \qquad \mathbf{S}^{v^-}_{i,j} \sim \mathrm{Uniform}(0,1)$$
$$\mathbf{S}^\sigma_{i,j} \sim \mathrm{Beta}(1.6,1)$$

$$\mathbf{I}_{i,j} | \mathbf{S}_{\mathrm{N9}(i,j)} \sim \mathrm{Normal}(\mathrm{E}[\mathbf{S}_{i,j}], \omega)$$

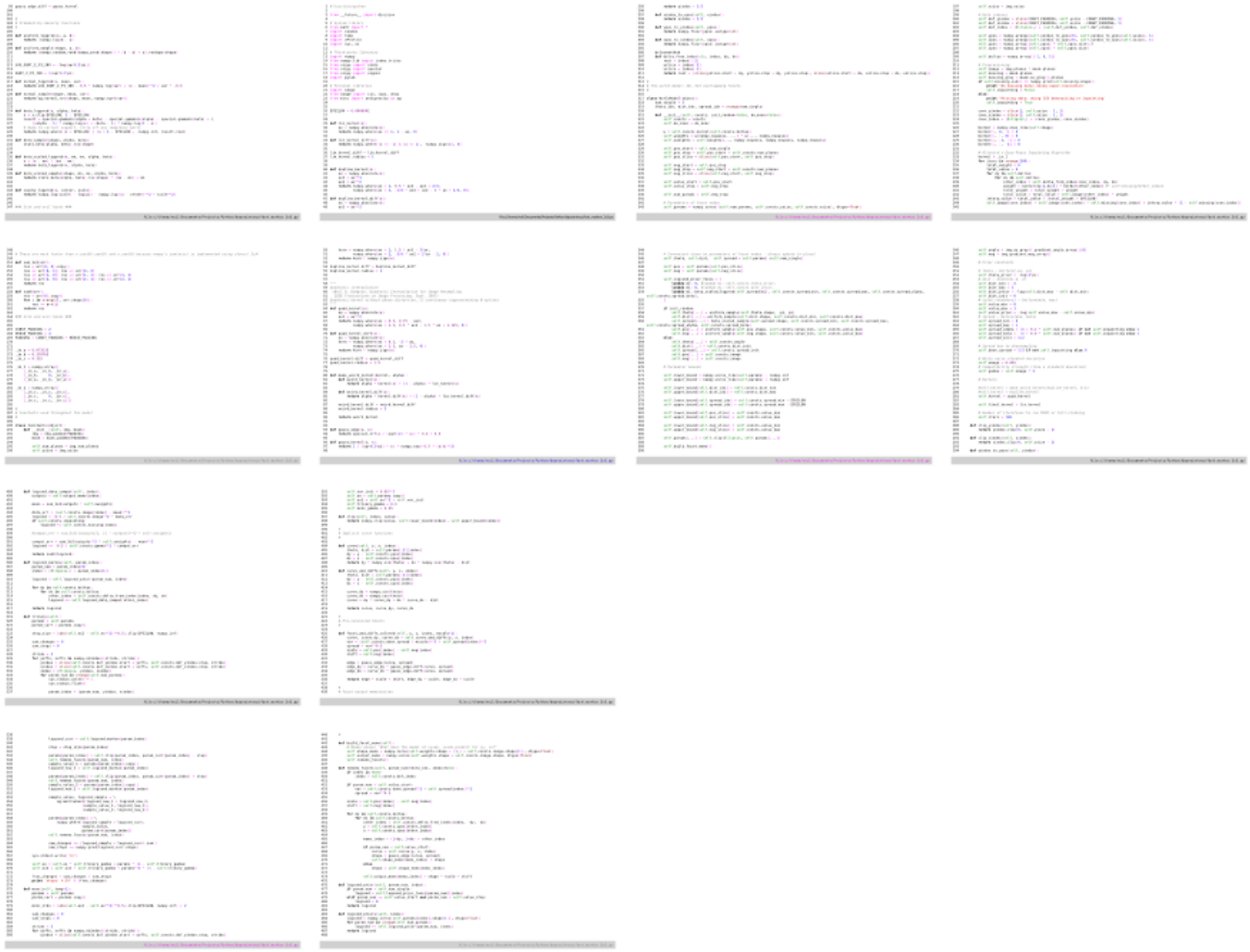
$$\Phi_{i,j}(\mathbf{S}_{\mathrm{N9}(i,j)}) \equiv \exp\left(-\frac{\mathrm{Var}[\mathbf{S}_{i,j}]}{2\gamma^2}\right)$$

Master's Research: Code

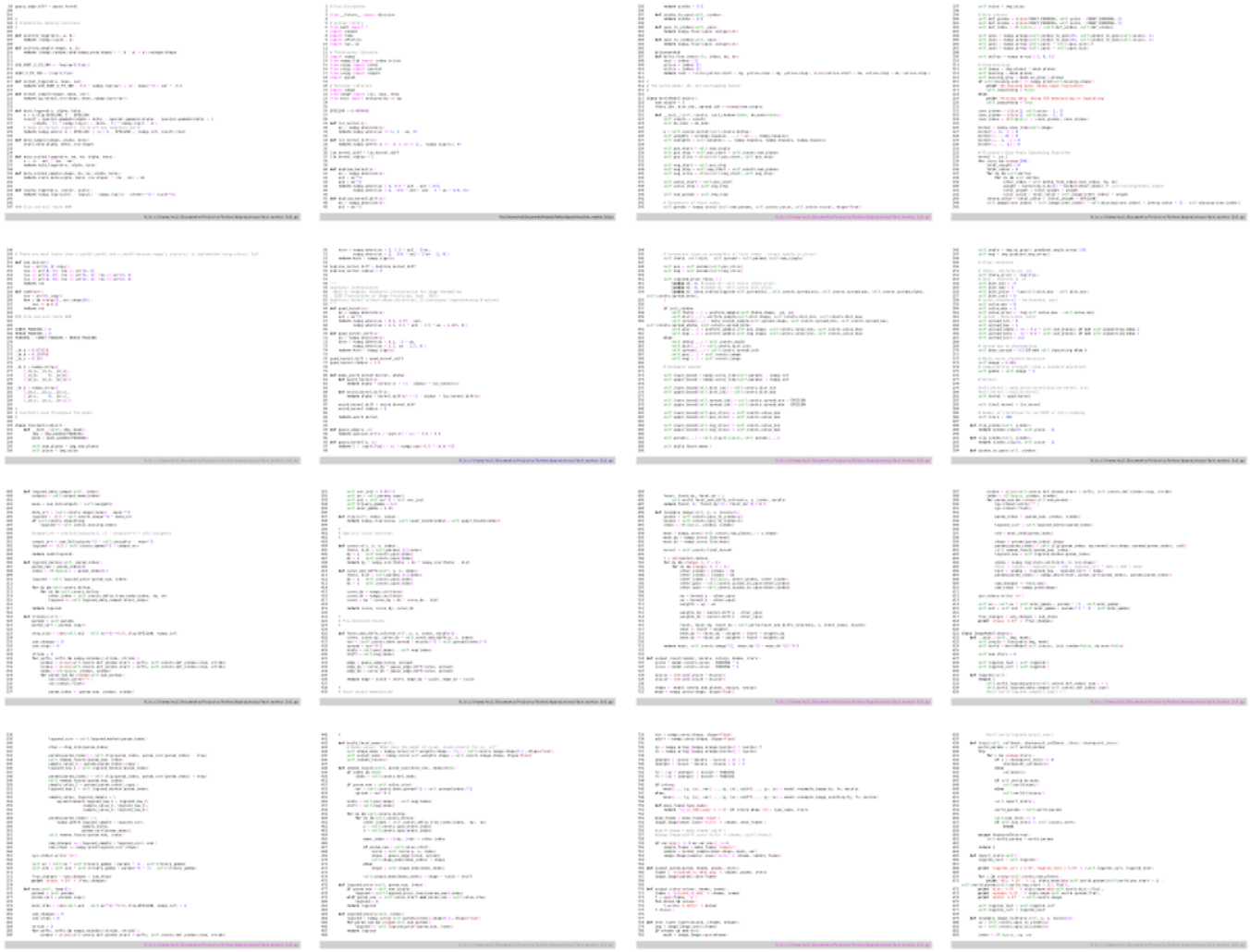
Master's Research: Code



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:

Resolution Synthesis

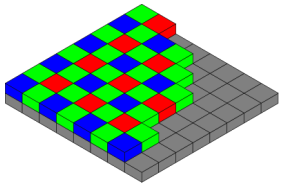
Bayesian Edge Inference

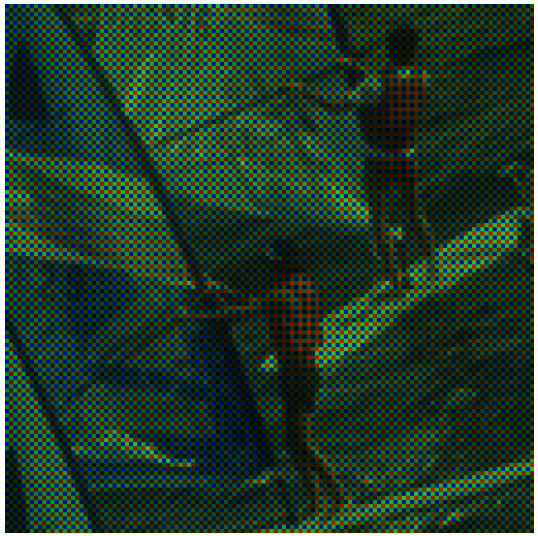
Beat state-of-the-art on “objective” measures

Results: Other Reconstruction

CCD

Demosaicing

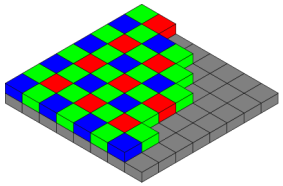


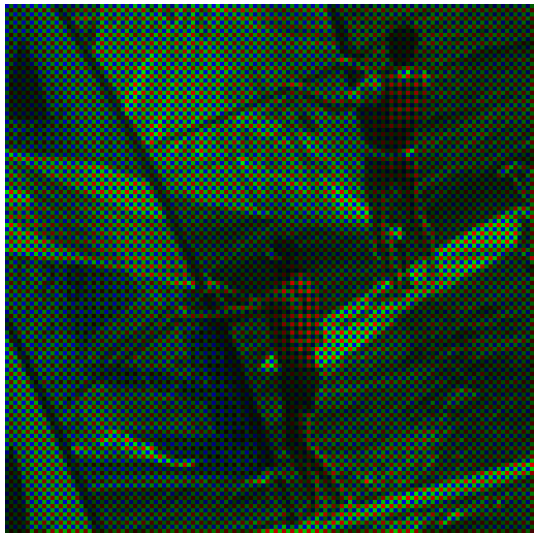


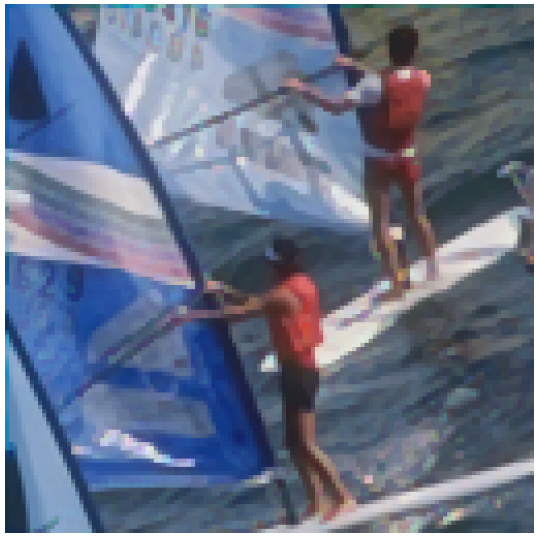
Results: Other Reconstruction

CCD

Demosaicing



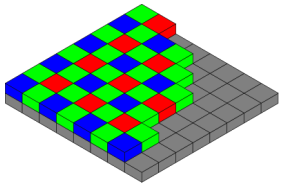


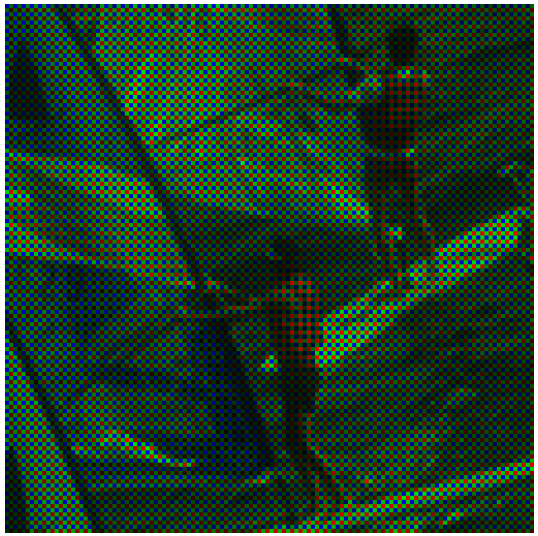


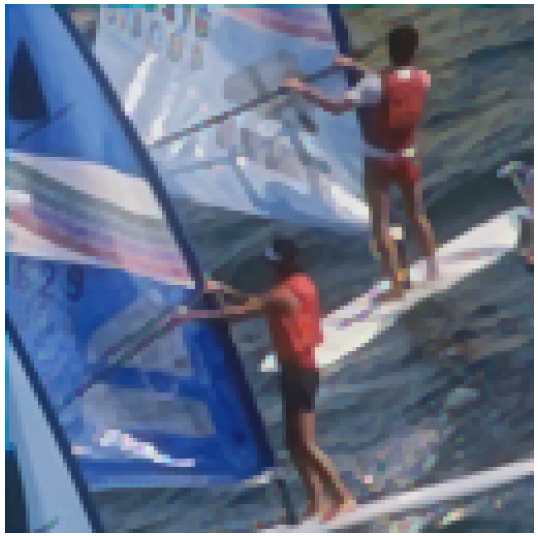
Results: Other Reconstruction

CCD

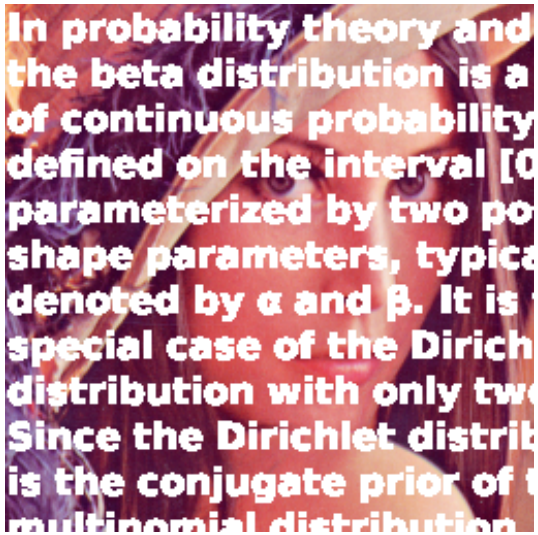
Demosaicing







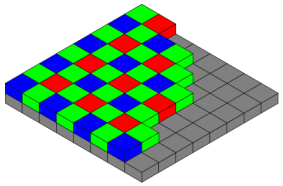
Inpainting

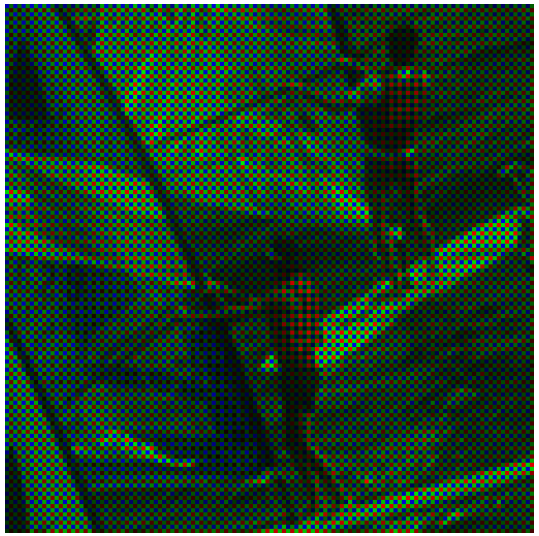


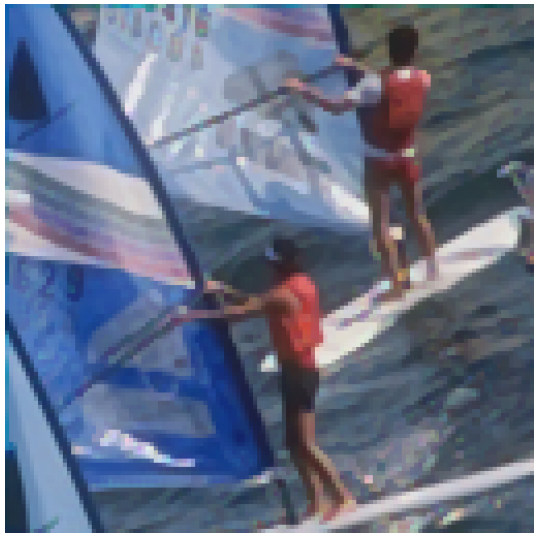
Results: Other Reconstruction

CCD

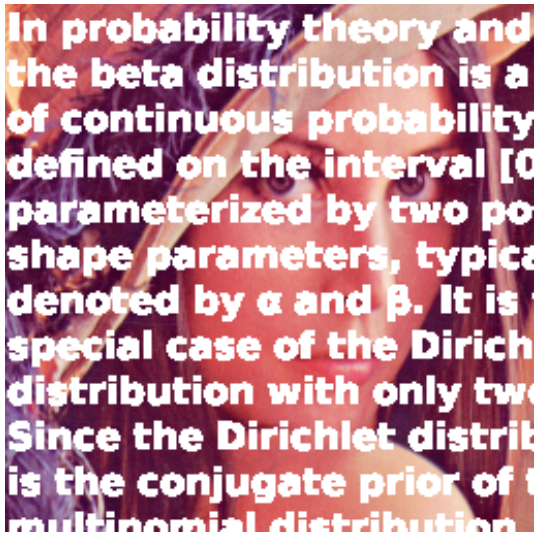
Demosaicing







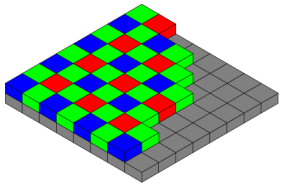
Inpainting

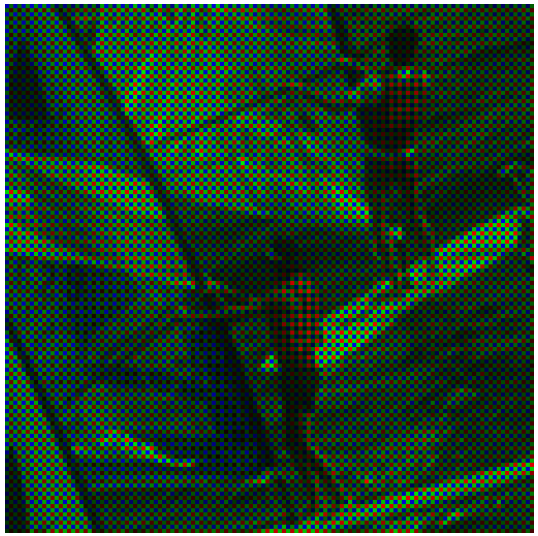


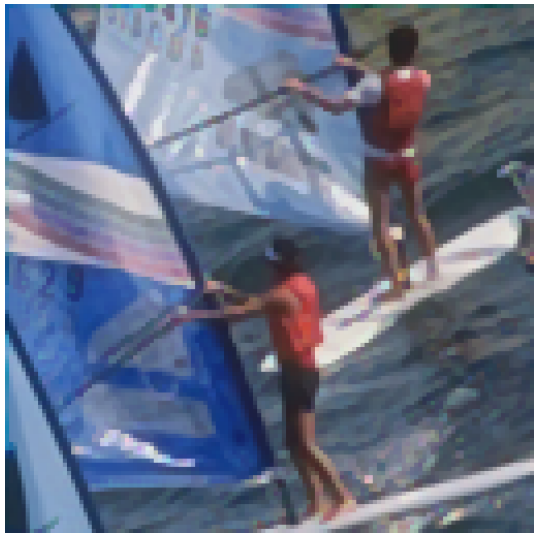
Results: Other Reconstruction

CCD

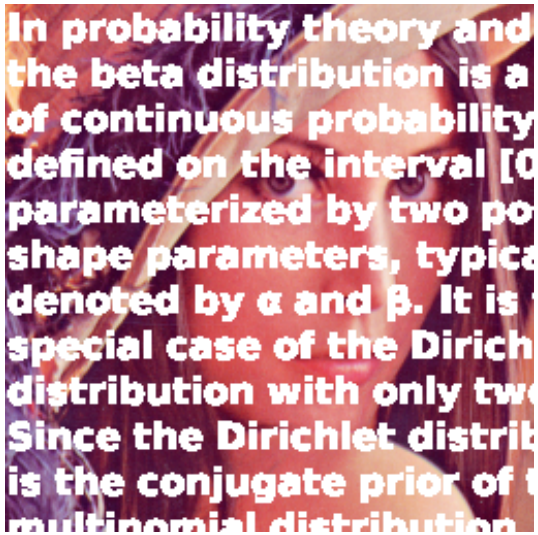
Demosaicing







Inpainting



It only *looks* like magic

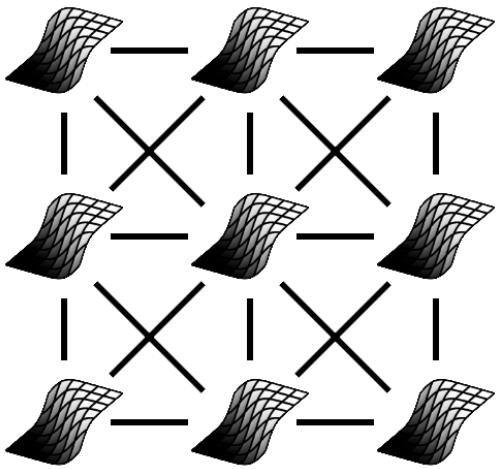
Only Mostly Satisfying

Problem 1: Still not sure the program is right

Only Mostly Satisfying

Problem 1: Still not sure the program is right

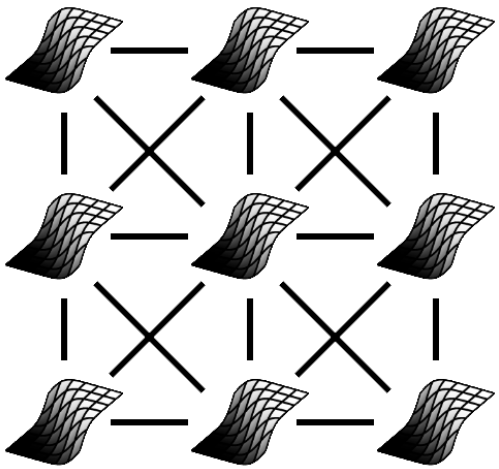
Problem 2: *Smooth* edges instead of *step* edges



Only Mostly Satisfying

Problem 1: Still not sure the program is right

Problem 2: *Smooth* edges instead of *step* edges

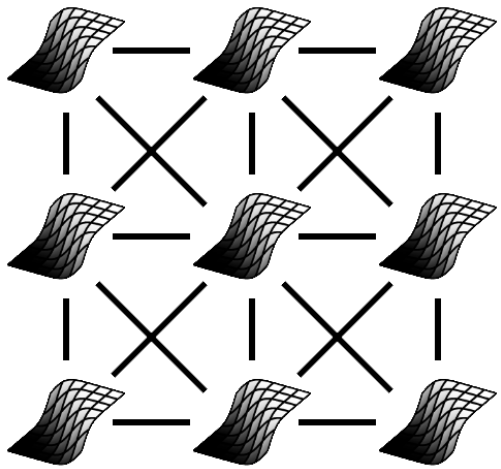


"To *approximate* blurring with a spatially varying point-spread function (PSF), we assign each facet a Gaussian PSF and convolve each analytically *before combining outputs*."

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*."

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

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

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

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
- 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

$$\mathbf{S}^{\theta}_{i,j} \sim \text{Uniform}(-\pi, \pi) \quad \mathbf{S}^{v^+}_{i,j} \sim \text{Uniform}(0, 1)$$
$$\mathbf{S}^{d}_{i,j} \sim \text{Uniform}(-3, 3) \quad \mathbf{S}^{v^-}_{i,j} \sim \text{Uniform}(0, 1)$$
$$\mathbf{S}^{\sigma}_{i,j} \sim \text{Beta}(1.6, 1)$$
$$\mathbf{I}_{i,j} | \mathbf{S}_{\text{N9}(i,j)} \sim \text{Normal}(\text{E}[\mathbf{S}_{i,j}], \omega)$$
$$\Phi_{i,j}(\mathbf{S}_{\text{N9}(i,j)}) \equiv \exp\left(-\frac{\text{Var}[\mathbf{S}_{i,j}]}{2\gamma^2}\right)$$

Culture Gap: Formalisms

- Strange vocabulary refers to even stranger math

$$\mathbf{S}^{\theta}_{i,j} \sim \text{Uniform}(-\pi, \pi) \quad \mathbf{S}^{v^+}_{i,j} \sim \text{Uniform}(0,1)$$
$$\mathbf{S}^{d}_{i,j} \sim \text{Uniform}(-3, 3) \quad \mathbf{S}^{v^-}_{i,j} \sim \text{Uniform}(0,1)$$
$$\mathbf{S}^{\sigma}_{i,j} \sim \text{Beta}(1.6, 1)$$
$$\mathbf{I}_{i,j} | \mathbf{S}_{\text{N9}(i,j)} \sim \text{Normal}(\text{E}[\mathbf{S}_{i,j}], \omega)$$
$$\Phi_{i,j}(\mathbf{S}_{\text{N9}(i,j)}) \equiv \exp\left(-\frac{\text{Var}[\mathbf{S}_{i,j}]}{2\gamma^2}\right)$$

$$\begin{aligned}
[\![f := e;\ \ldots\ ; e_b]\!]_{\mathsf{a}} &:\equiv f := [\![e]\!]_{\mathsf{a}};\ \ldots\ ; [\![e_b]\!]_{\mathsf{a}} \\
[\![\mathsf{let}\ e\ e_b]\!]_{\mathsf{a}} &:\equiv ([\![e]\!]_{\mathsf{a}}\ \&\&\&_{\mathsf{a}}\ \mathsf{arr_a\ id}) \ggg_{\mathsf{a}} [\![e_b]\!]_{\mathsf{a}} \\
[\![\mathsf{env}\ 0]\!]_{\mathsf{a}} &:\equiv \mathsf{arr_a\ fst} \\
[\![\mathsf{env}\ (n+1)]\!]_{\mathsf{a}} &:\equiv \mathsf{arr_a\ snd} \ggg_{\mathsf{a}} [\![\mathsf{env}\ n]\!]_{\mathsf{a}} \\
[\![\mathsf{if}\ e_c\ \mathsf{then}\ e_t\ \mathsf{else}\ e_f]\!]_{\mathsf{a}} &:\equiv \mathsf{ifte_a}\ [\![e_c]\!]_{\mathsf{a}}\ (\mathsf{lazy_a}\ \lambda 0.\ [\![e_t]\!]_{\mathsf{a}})\ (\mathsf{lazy_a}\ \lambda 0.\ [\![e_f]\!]_{\mathsf{a}})
\end{aligned}$$

$$\begin{aligned}
[\![\langle e_1, e_2 \rangle]\!]_{\mathsf{a}} &:\equiv [\![e_1]\!]_{\mathsf{a}}\ \&\&\&_{\mathsf{a}}\ [\![e_2]\!]_{\mathsf{a}} \\
[\![f\ e]\!]_{\mathsf{a}} &:\equiv [\![\langle e, \langle\rangle \rangle]\!]_{\mathsf{a}} \ggg_{\mathsf{a}} f \\
[\![\delta\ e]\!]_{\mathsf{a}} &:\equiv [\![e]\!]_{\mathsf{a}} \ggg_{\mathsf{a}} \mathsf{arr_a}\ \delta \\
[\![v]\!]_{\mathsf{a}} &:\equiv \mathsf{arr_a\ (const}\ v)
\end{aligned}$$

Culture Gap: Formalisms

- Strange vocabulary refers to even stranger math

$$\mathbf{S}^{\theta}_{i,j} \sim \text{Uniform}(-\pi, \pi) \quad \mathbf{S}^{v^+}_{i,j} \sim \text{Uniform}(0,1)$$
$$\mathbf{S}^{d}_{i,j} \sim \text{Uniform}(-3, 3) \quad \mathbf{S}^{v^-}_{i,j} \sim \text{Uniform}(0,1)$$
$$\mathbf{S}^{\sigma}_{i,j} \sim \text{Beta}(1.6, 1)$$
$$\mathbf{I}_{i,j} | \mathbf{S}_{\text{N9}(i,j)} \sim \text{Normal}(\text{E}[\mathbf{S}_{i,j}], \omega)$$
$$\Phi_{i,j}(\mathbf{S}_{\text{N9}(i,j)}) \equiv \exp\left(-\frac{\text{Var}[\mathbf{S}_{i,j}]}{2\gamma^2}\right)$$

$$\begin{aligned}
[\![f := e;\ \ldots\ ; e_b]\!]_{\mathsf{a}} &:\equiv f := [\![e]\!]_{\mathsf{a}};\ \ldots\ ; [\![e_b]\!]_{\mathsf{a}} \\
[\![\mathsf{let}\ e\ e_b]\!]_{\mathsf{a}} &:\equiv ([\![e]\!]_{\mathsf{a}}\ \&\&\&_{\mathsf{a}}\ \mathsf{arr_a\ id}) \ggg_{\mathsf{a}} [\![e_b]\!]_{\mathsf{a}} \\
[\![\mathsf{env}\ 0]\!]_{\mathsf{a}} &:\equiv \mathsf{arr_a\ fst} \\
[\![\mathsf{env}\ (n+1)]\!]_{\mathsf{a}} &:\equiv \mathsf{arr_a\ snd} \ggg_{\mathsf{a}} [\![\mathsf{env}\ n]\!]_{\mathsf{a}} \\
[\![\mathsf{if}\ e_c\ \mathsf{then}\ e_t\ \mathsf{else}\ e_f]\!]_{\mathsf{a}} &:\equiv \mathsf{ifte_a}\ [\![e_c]\!]_{\mathsf{a}}\ (\mathsf{lazy_a}\ \lambda 0.\ [\![e_t]\!]_{\mathsf{a}})\ (\mathsf{lazy_a}\ \lambda 0.\ [\![e_f]\!]_{\mathsf{a}})
\end{aligned}$$

$$\begin{aligned}
[\![\langle e_1, e_2\rangle]\!]_{\mathsf{a}} &:\equiv [\![e_1]\!]_{\mathsf{a}}\ \&\&\&_{\mathsf{a}}\ [\![e_2]\!]_{\mathsf{a}} \\
[\![f\ e]\!]_{\mathsf{a}} &:\equiv [\![\langle e, \langle\rangle\rangle]\!]_{\mathsf{a}} \ggg_{\mathsf{a}} f \\
[\![\delta\ e]\!]_{\mathsf{a}} &:\equiv [\![e]\!]_{\mathsf{a}} \ggg_{\mathsf{a}} \mathsf{arr_a}\ \delta \\
[\![v]\!]_{\mathsf{a}} &:\equiv \mathsf{arr_a\ (const}\ v)
\end{aligned}$$

- 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: $e \ ::= \ 0, 1, 2, \ldots \mid \texttt{add}\ e\ e$

Five-Minute Semantics

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

- What should $\texttt{add}\ \texttt{4}\ \texttt{5}$ mean?

Five-Minute Semantics

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

- What should $\texttt{add}\ \texttt{4}\ \texttt{5}$ mean? We have two main options:

Five-Minute Semantics

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

- What should $\texttt{add}\ \texttt{4}\ \texttt{5}$ mean? We have two main options:
  - Operational: $\texttt{add}\ \texttt{4}\ \texttt{5}$ means 9

Five-Minute Semantics

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

- 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: $e \ ::= \ 0, 1, 2, \ldots \mid \texttt{add}\ e\ e \mid \texttt{choose}\ e\ e$

Five-Minute Semantics

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

- What should $\texttt{choose}\ 10\ 20$ mean?

Five-Minute Semantics

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

- What should $\texttt{choose}\ 10\ 20$ mean? We have two main options:

Five-Minute Semantics

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

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

Five-Minute Semantics

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

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

Five-Minute Semantics

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

Denotational semantics: define a *semantic function* $[\![\cdot]\!]$

Five-Minute Semantics

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

Denotational semantics: define a *semantic function* $[\![\cdot]\!]$

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

Five-Minute Semantics

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

Denotational semantics: define a *semantic function* $[\![\cdot]\!]$

- Just an outfix function that operates on syntax; e.g. $[\![\texttt{choose 10 20}]\!] = \{10, 20\}$
- What should its return type be?

Five-Minute Semantics

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

Denotational semantics: define a *semantic function* $[\![\cdot]\!]$

- Just an outfix function that operates on syntax; e.g. $[\![\texttt{choose 10 20}]\!] = \{10, 20\}$
- What should its return type be?
- Uniform interpretation is good (e.g. Bayesian models always denote the same thing: a joint distribution)

Five-Minute Semantics

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

Denotational semantics: define a *semantic function* $[\![\cdot]\!]$

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

Five-Minute Semantics

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

Denotational semantics: define a *semantic function* $[\![\cdot]\!]$

- Just an outfix function that operates on syntax; e.g. $[\![\texttt{choose 10 20}]\!] = \{10, 20\}$
- 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 $[\![\texttt{choose 10 20}]\!] = \{10, 20\}$
  - But $[\![10]\!] = \{10\}$ would work

Five-Minute Semantics

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

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

Five-Minute Semantics

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

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

$$[\![n]\!] \ = \ \{n\}$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!]
\end{aligned}
$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

Test cases: $[\![\texttt{10}]\!]$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

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

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

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

$[\![\texttt{choose 10 20}]\!]$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

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

$[\![\texttt{choose 10 20}]\!] = [\![10]\!] \cup [\![20]\!]$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

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

$$
\begin{aligned}
[\![\texttt{choose 10 20}]\!] &= [\![10]\!] \cup [\![20]\!] \\
&= \{10\} \cup \{20\} = \{10, 20\}
\end{aligned}
$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

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

$$
\begin{aligned}
[\![\texttt{choose 10 20}]\!] &= [\![10]\!] \cup [\![20]\!] \\
&= \{10\} \cup \{20\} = \{10, 20\}
\end{aligned}
$$

$[\![\texttt{add 4 5}]\!]$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

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

$$
\begin{aligned}
[\![\texttt{choose 10 20}]\!] &= [\![10]\!] \cup [\![20]\!] \\
&= \{10\} \cup \{20\} = \{10, 20\}
\end{aligned}
$$

$$[\![\texttt{add 4 5}]\!] = \bigcup_{n_1 \in [\![4]\!]} \bigcup_{n_1 \in [\![5]\!]} \{n_1 + n_2\}$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

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

$$
\begin{aligned}
[\![\texttt{choose 10 20}]\!] &= [\![10]\!] \cup [\![20]\!] \\
&= \{10\} \cup \{20\} = \{10, 20\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{add 4 5}]\!] &= \bigcup_{n_1 \in [\![4]\!]} \bigcup_{n_1 \in [\![5]\!]} \{n_1 + n_2\} \\
&= \bigcup_{n_1 \in \{4\}} \bigcup_{n_1 \in \{5\}} \{n_1 + n_2\}
\end{aligned}
$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

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

$$
\begin{aligned}
[\![\texttt{choose 10 20}]\!] &= [\![10]\!] \cup [\![20]\!] \\
&= \{10\} \cup \{20\} = \{10, 20\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\texttt{add 4 5}]\!] &= \bigcup_{n_1 \in [\![4]\!]} \bigcup_{n_1 \in [\![5]\!]} \{n_1 + n_2\} \\
&= \bigcup_{n_1 \in \{4\}} \bigcup_{n_1 \in \{5\}} \{n_1 + n_2\} \\
&= \{4 + 5\} = \{9\}
\end{aligned}
$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

Distributive property:

$$[\![\texttt{add 4 (choose 10 20)}]\!] = \{14, 24\}$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

Distributive property:

$$
\begin{aligned}
[\![\texttt{add 4 (choose 10 20)}]\!] &= \{14, 24\} \\
&= [\![\texttt{choose (add 4 10) (add 4 20)}]\!]
\end{aligned}
$$

Five-Minute Semantics

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

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

$$
\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}
$$

Distributive property: for all programs e_1, e_2 and e_3,

$$[\![\texttt{add}\ e_1\ (\texttt{choose}\ e_2\ e_3)]\!] = [\![\texttt{choose}\ (\texttt{add}\ e_1\ e_2)\ (\texttt{add}\ e_1\ e_3)]\!]$$

Five-Minute Semantics

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

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

$$\begin{aligned} [\![n]\!] &= \{n\} \\ [\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\ [\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\} \end{aligned}$$

Distributive property: for all programs e_1, e_2 and e_3,

$$[\![\texttt{add}\ e_1\ (\texttt{choose}\ e_2\ e_3)]\!] = [\![\texttt{choose}\ (\texttt{add}\ e_1\ e_2)\ (\texttt{add}\ e_1\ e_3)]\!]$$

- Can prove this using the semantics

Five-Minute Semantics

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

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

$$\begin{aligned}
[\![n]\!] &= \{n\} \\
[\![\texttt{choose}\ e_1\ e_2]\!] &= [\![e_1]\!] \cup [\![e_2]\!] \\
[\![\texttt{add}\ e_1\ e_2]\!] &= \bigcup_{n_1 \in [\![e_1]\!]} \bigcup_{n_2 \in [\![e_2]\!]} \{n_1 + n_2\}
\end{aligned}$$

Distributive property: for all programs e_1, e_2 and e_3,

$$[\![\texttt{add}\ e_1\ (\texttt{choose}\ e_2\ e_3)]\!] = [\![\texttt{choose}\ (\texttt{add}\ e_1\ e_2)\ (\texttt{add}\ e_1\ e_3)]\!]$$

- 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)
```

0.5

0.5

Programming Coin Flips

```
(let ([x  (flip 0.5)]
      [y  (flip 0.5)])
  (list x y))
```

0.5

0.5 ⟨ 

, 

⟩

0.5 ⟨ 

, 

⟩

Programming Coin Flips

```
(let ([x  (flip 0.5)]
      [y  (flip 0.5)])
  (list x y))
```

0.5 0.5

0.5

0.5

Programming Coin Flips

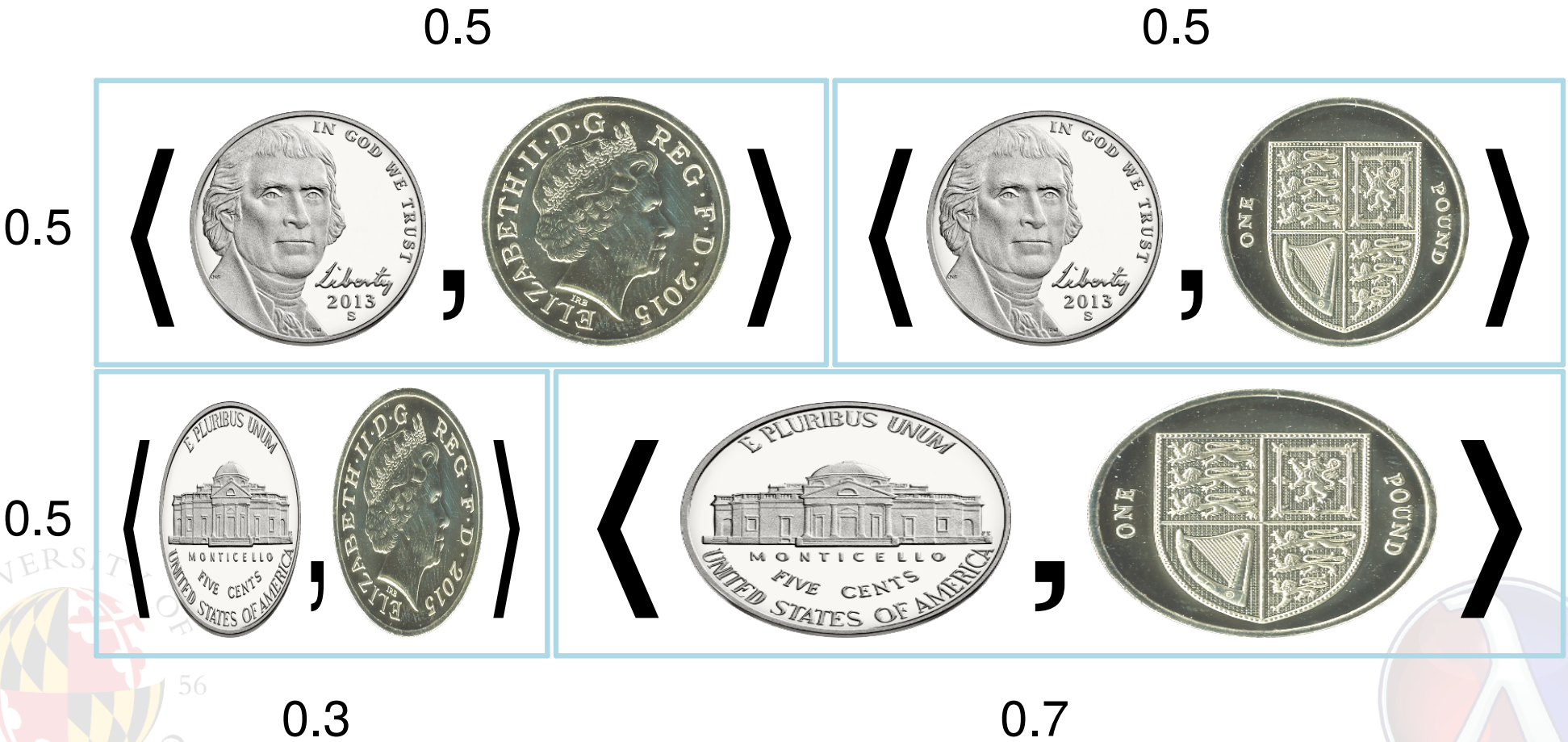
```
(let ([x  (flip 0.5)]
      [y  (flip 0.5)])
  (list x y))
```

Programming Coin Flips

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

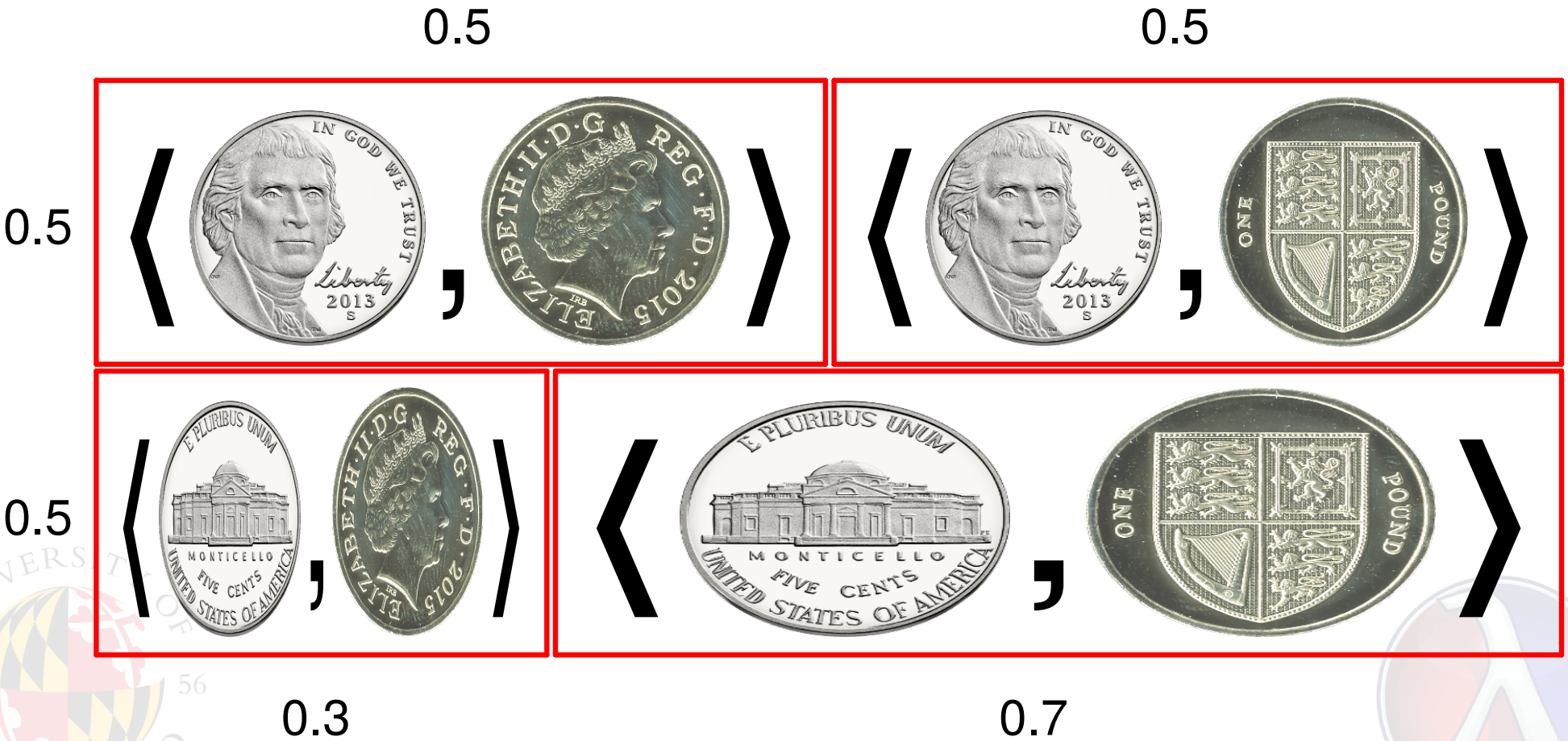
Programming Coin Flips

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



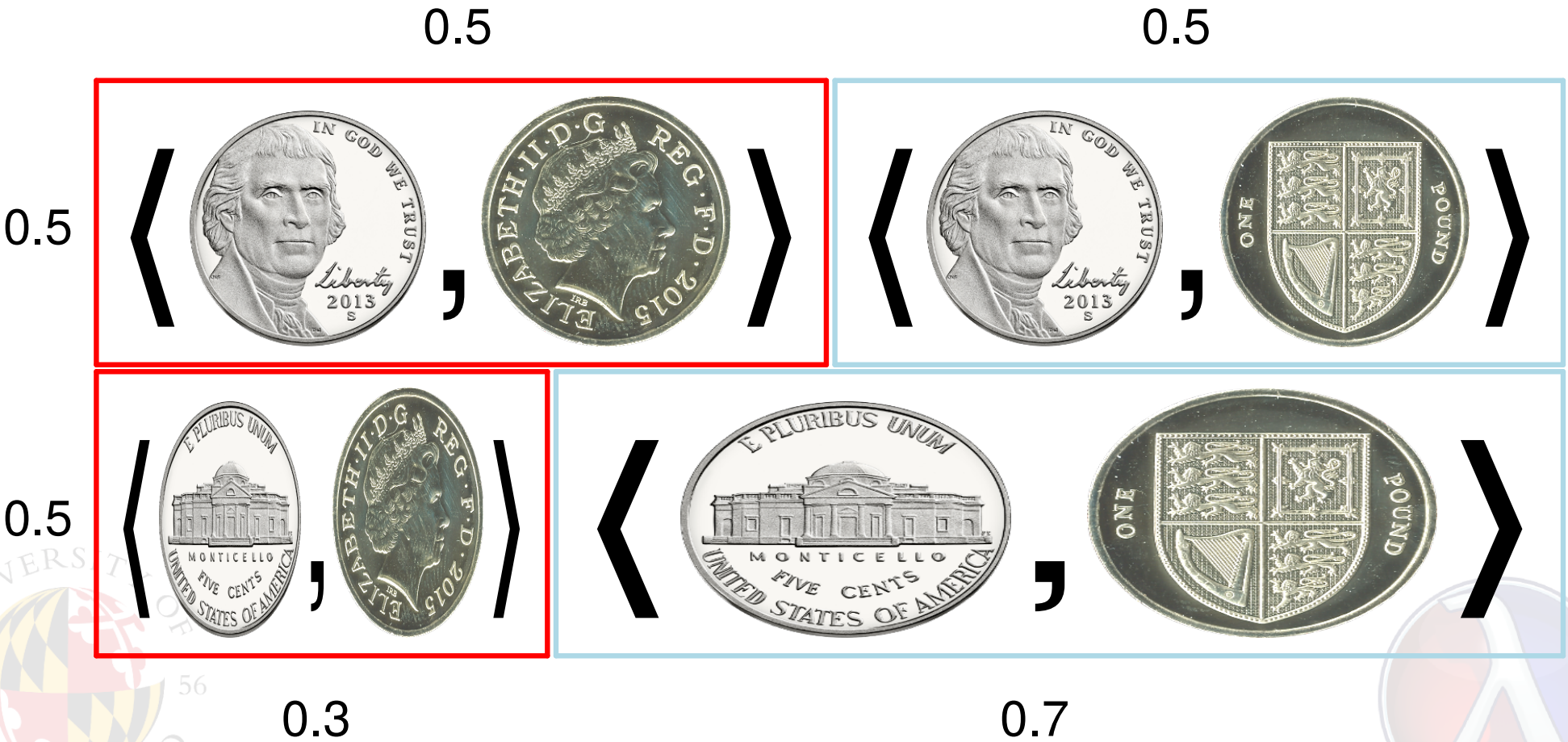
Programming Coin Flips

$$\Pr[\texttt{true}] = 0.5 \cdot 0.5 + 0.5 \cdot 0.5 + 0.5 \cdot 0.3 + 0.5 \cdot 0.7 = 1$$



Programming Coin Flips

$$\Pr[\mathrm{y} = \mathrm{heads}] = 0.5 \cdot 0.5 + 0.5 \cdot 0.3 = 0.4$$



Programming Coin Flips

$$\Pr[\mathrm{x} = \mathrm{heads} \mid \mathrm{y} = \mathrm{heads}]$$
$$= \Pr[\langle \mathrm{x}, \mathrm{y} \rangle = \langle \mathrm{heads}, \mathrm{heads} \rangle] / \Pr[\mathrm{y} = \mathrm{heads}]$$
$$= 0.25/0.4 = 0.625$$

0.5

0.5

0.5

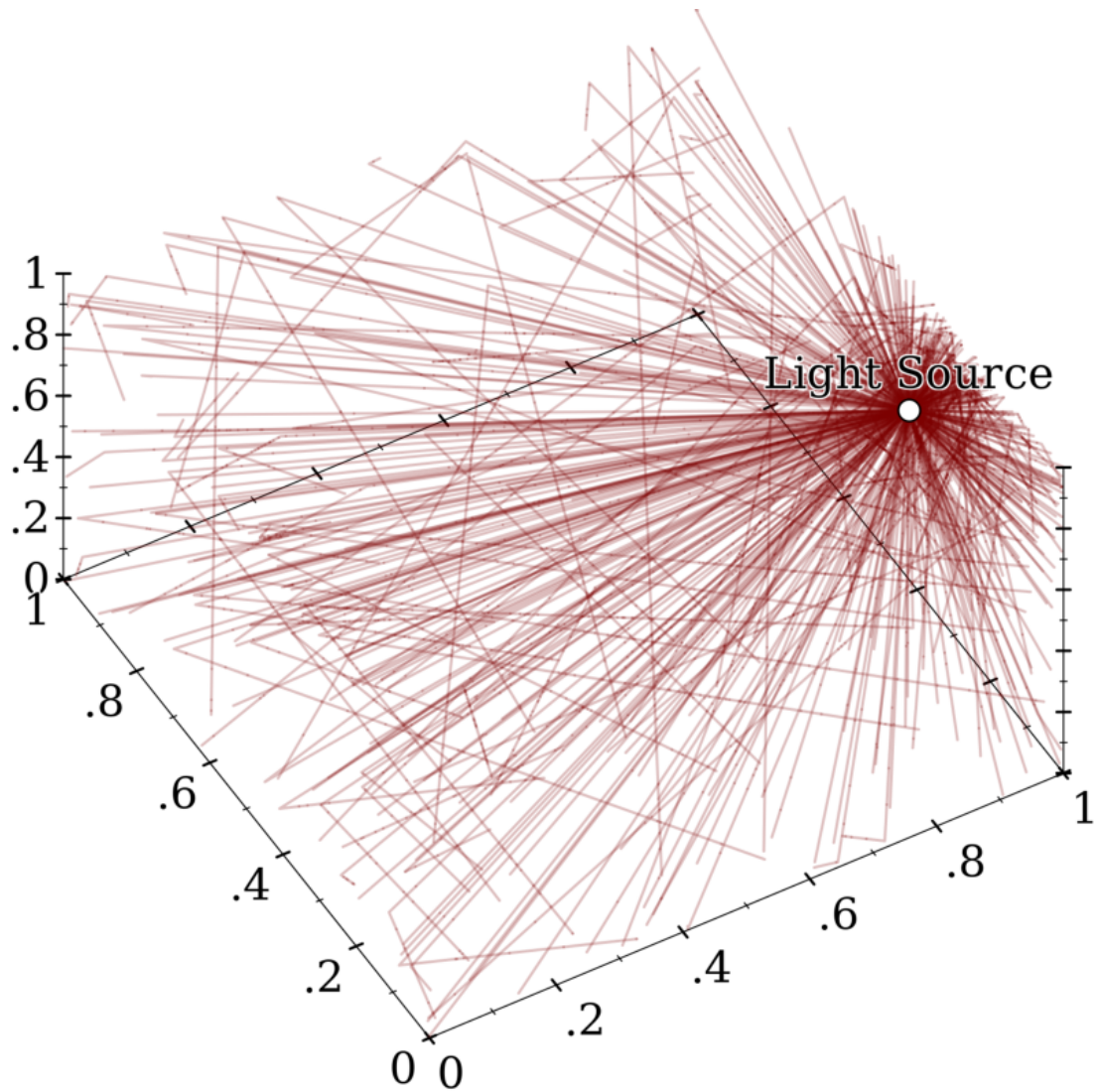
0.3

Stochastic Ray Tracing

sto·cha·stic /stō-ˈkas-tik/ *adj.* fancy word for "randomized"

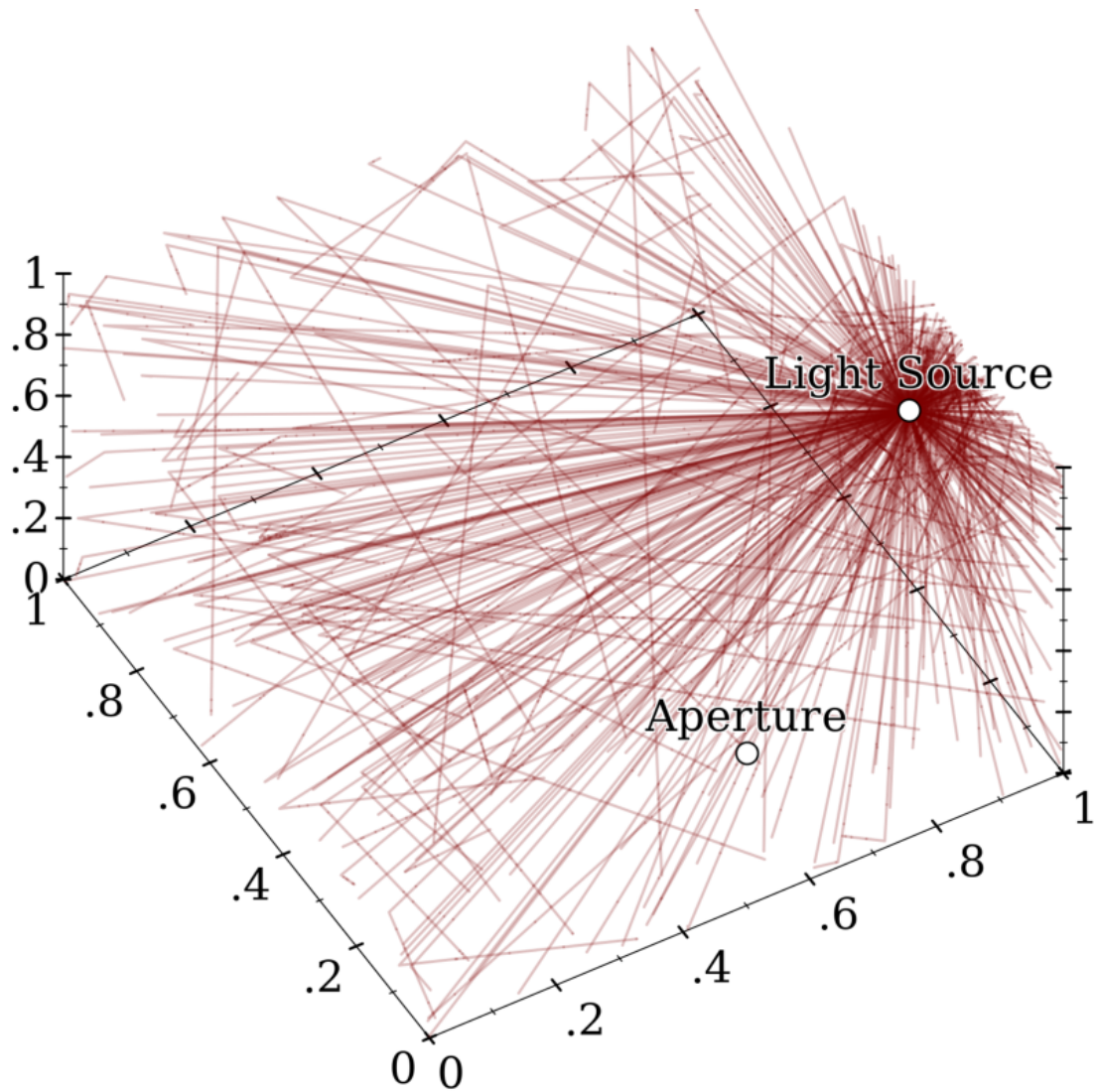
Stochastic Ray Tracing

sto·cha·stic /stō-ˈkas-tik/ *adj.* fancy word for "randomized"



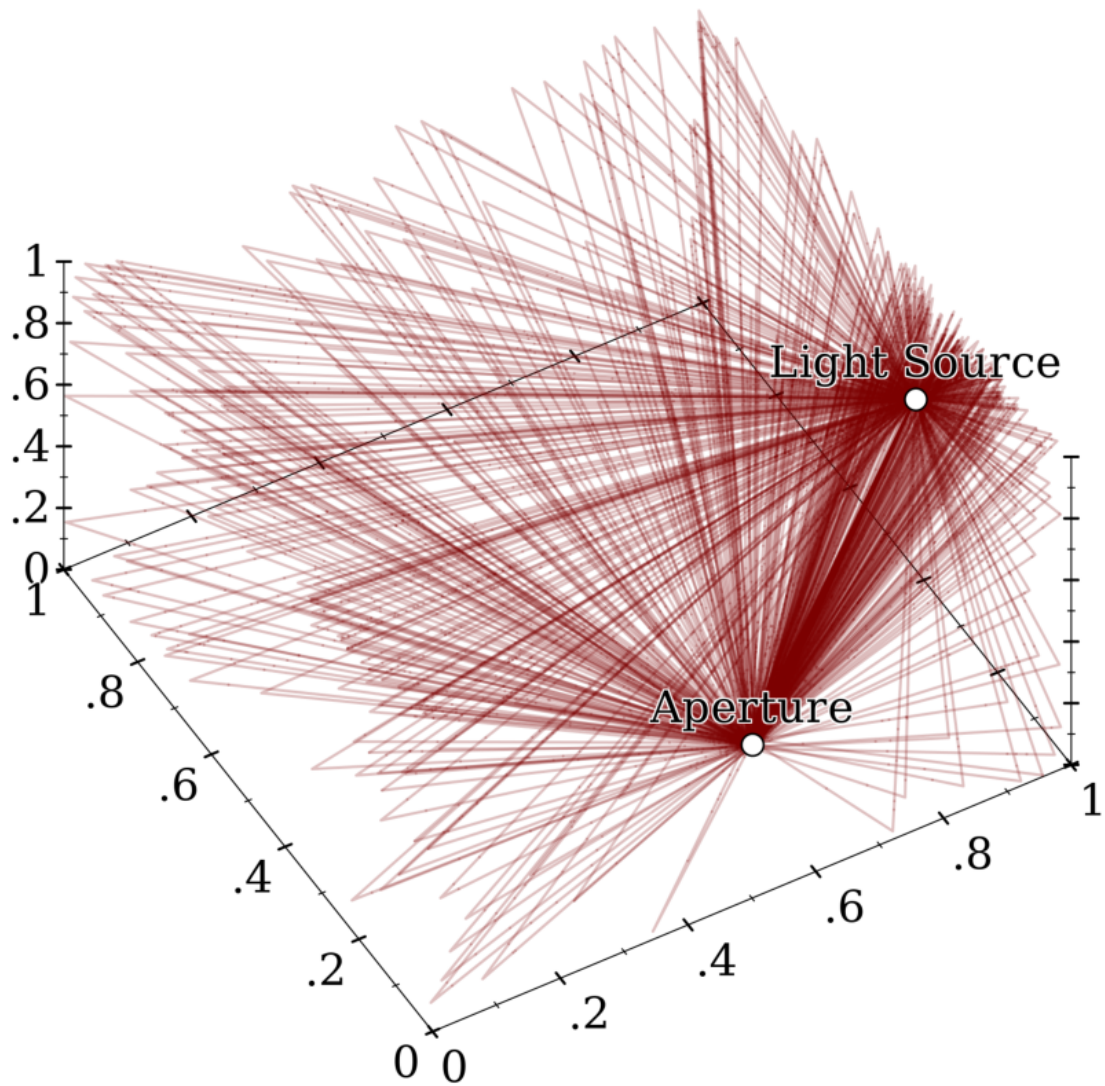
Stochastic Ray Tracing

ap·er·ture /'ap-ə(r)-chər/ *n.* fancy word for "opening"



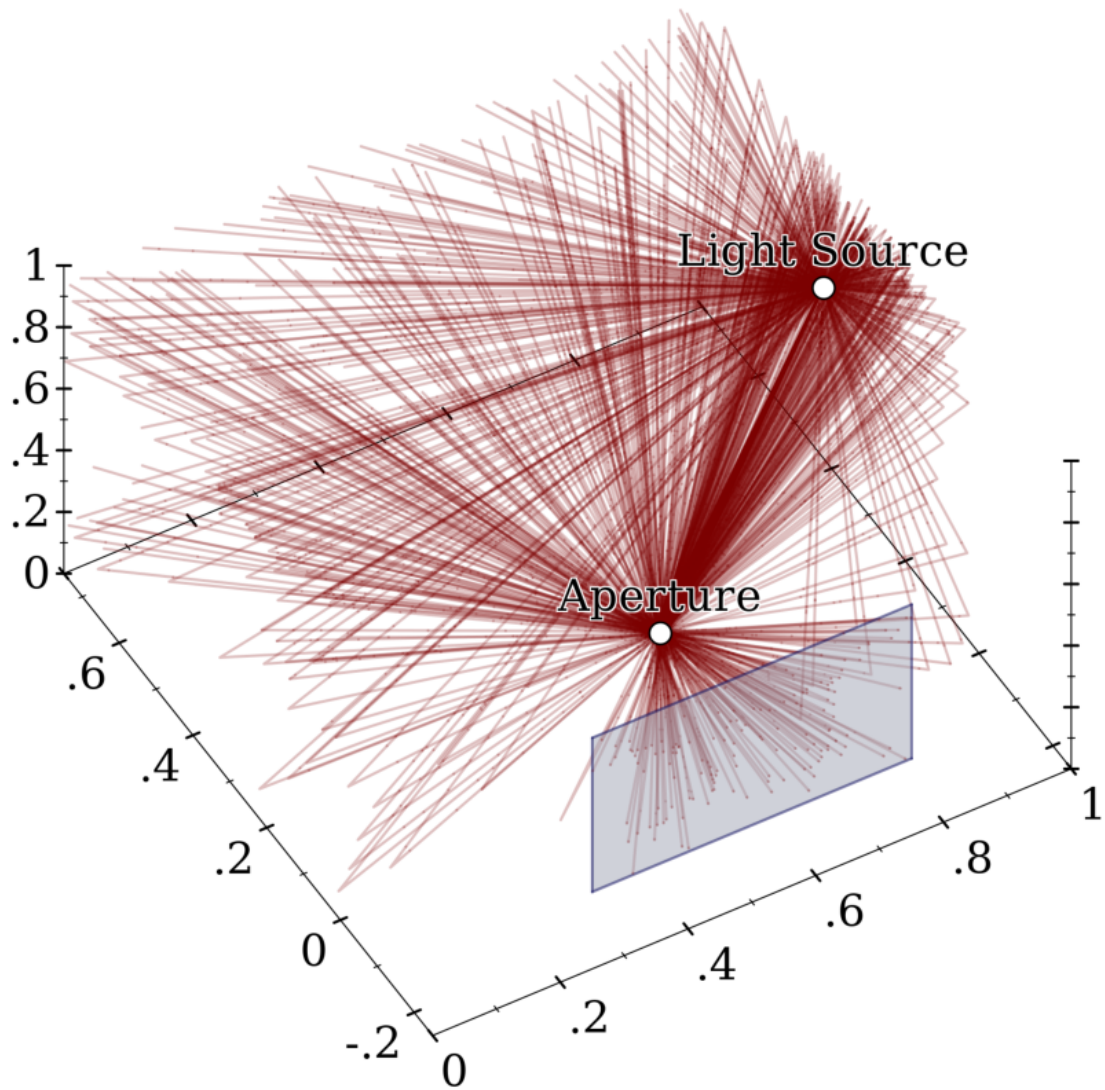
Stochastic Ray Tracing

ap·er·ture /'ap-ə(r)-chər/ *n.* fancy word for "opening"



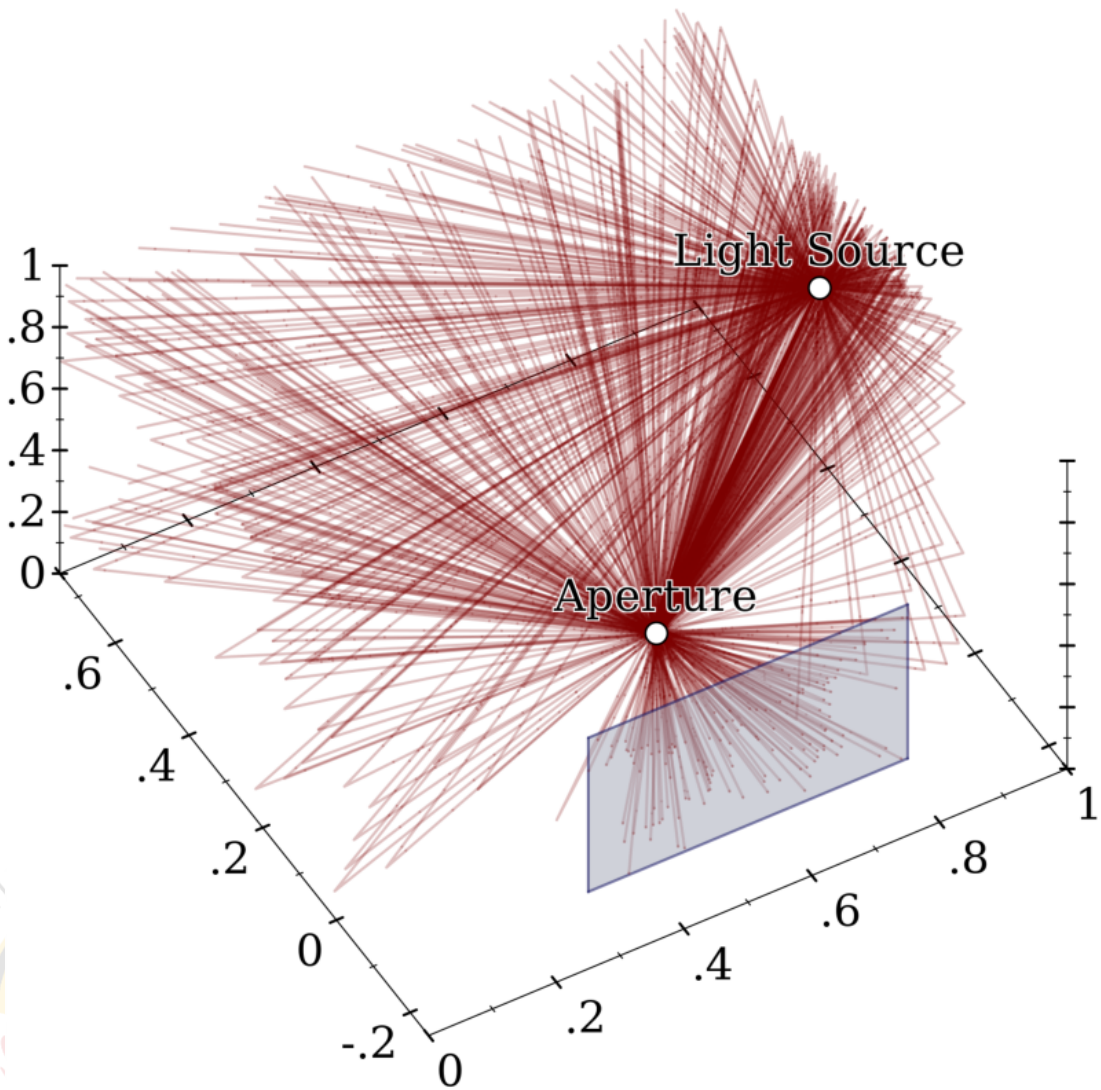
Stochastic Ray Tracing

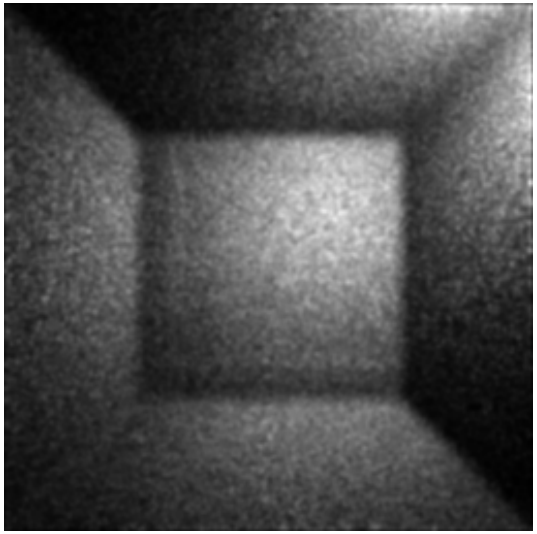
Simulate projecting rays onto a sensor...



Stochastic Ray Tracing

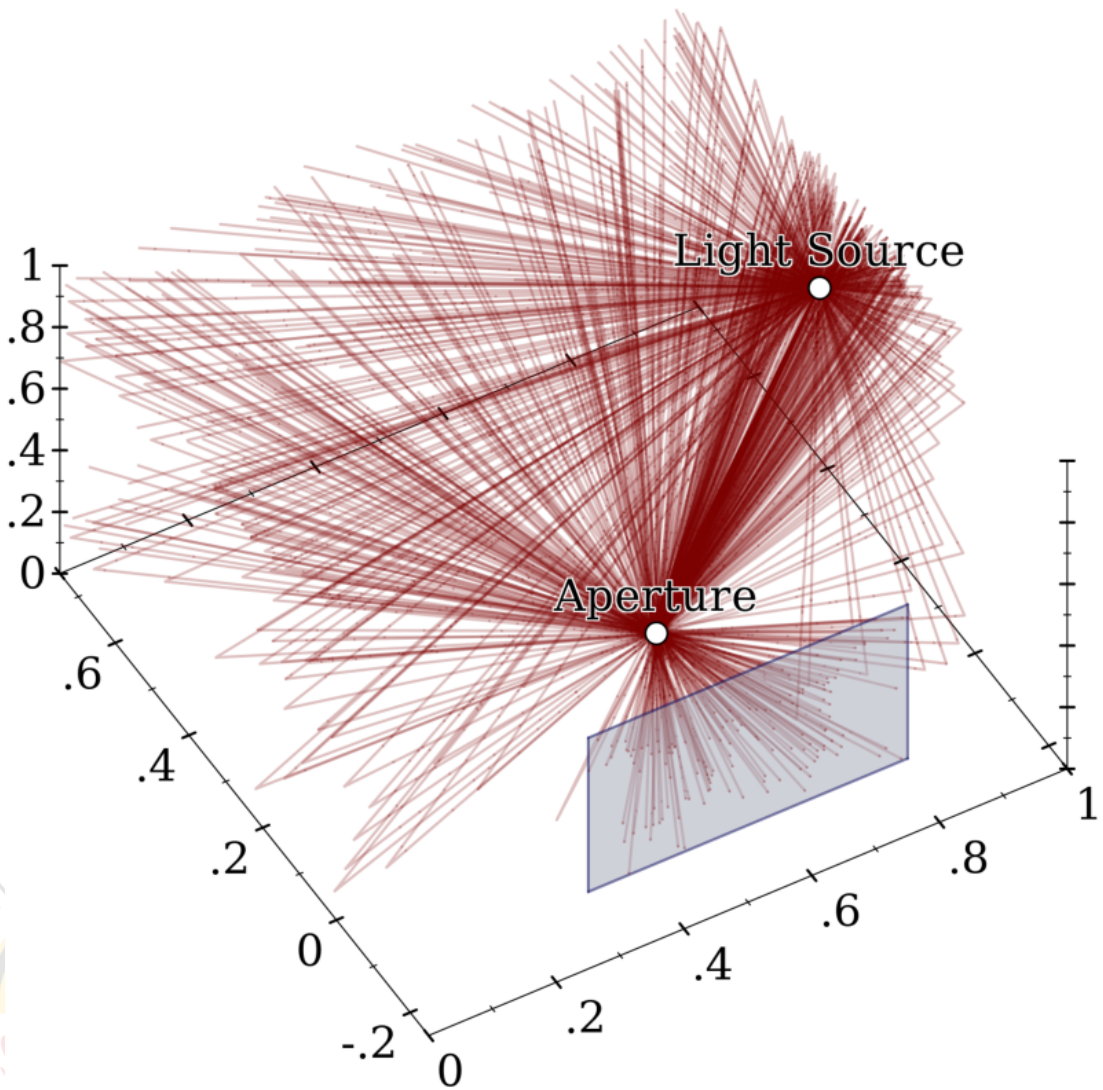
... and collect them to form an image

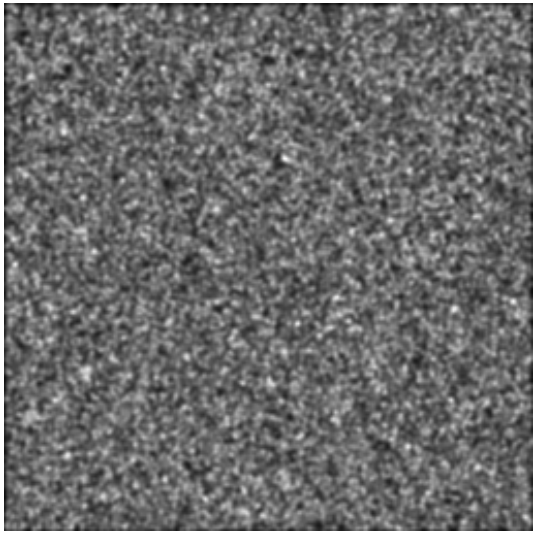




Stochastic Ray Tracing

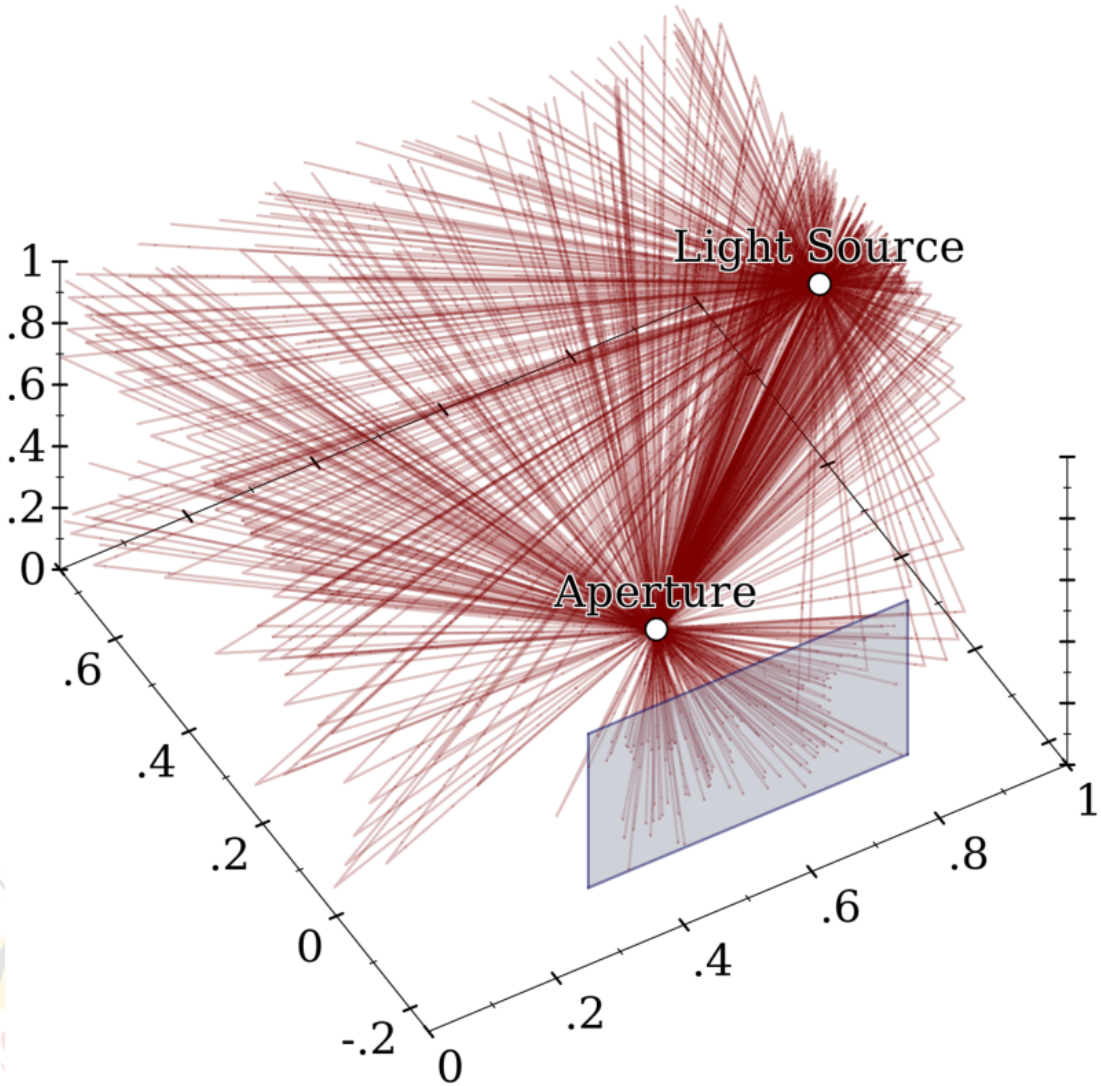
Critical: must maintain the distribution of rays





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:

```
(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  (- (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 `(max 0.5 (random))`

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 `(max 0.5 (random))`

- The reasons are almost entirely theoretical

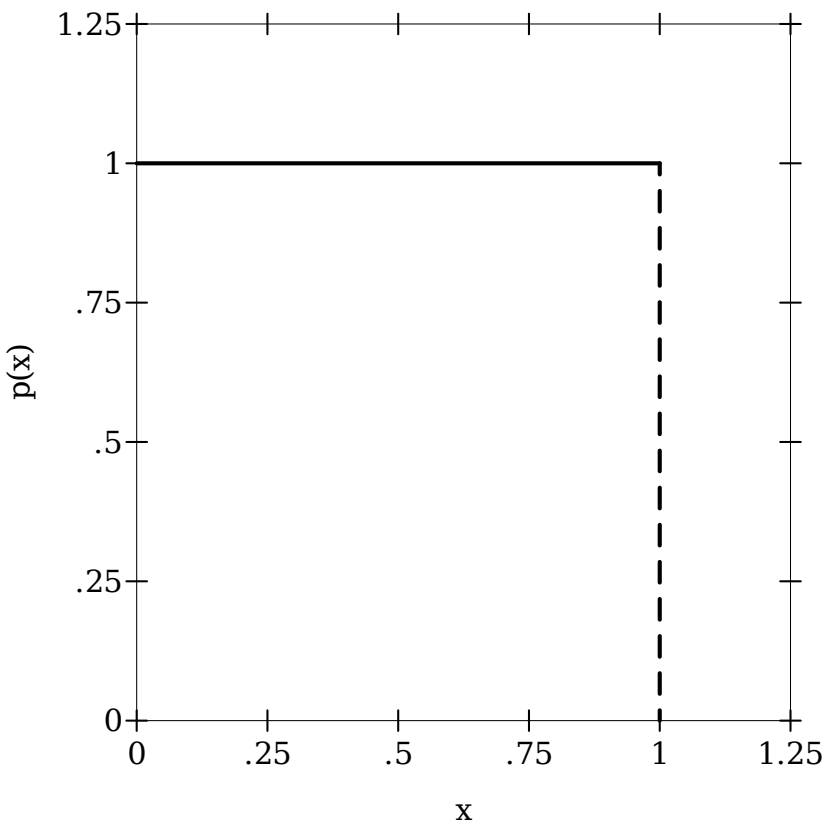
Simpler Example

- Assume `(random)` returns a value uniformly in $[0, 1]$

Simpler Example

- Assume `(random)` returns a value uniformly in $[0, 1]$

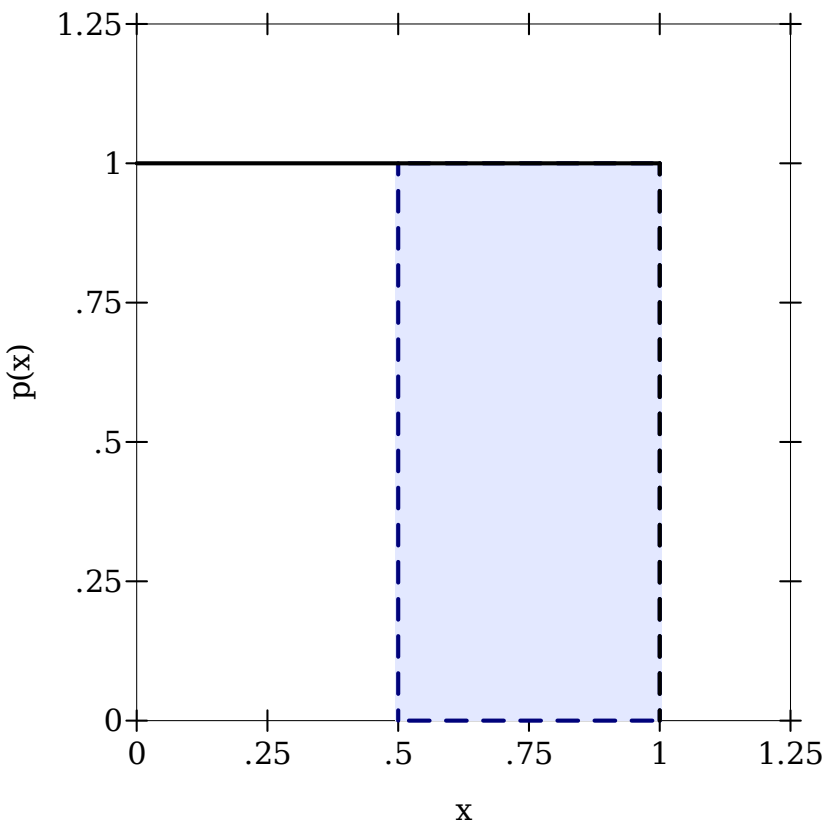
Density function p for value of `(random)`:



Simpler Example

- Assume `(random)` returns a value uniformly in $[0, 1]$

Density function p for value of `(random)`:

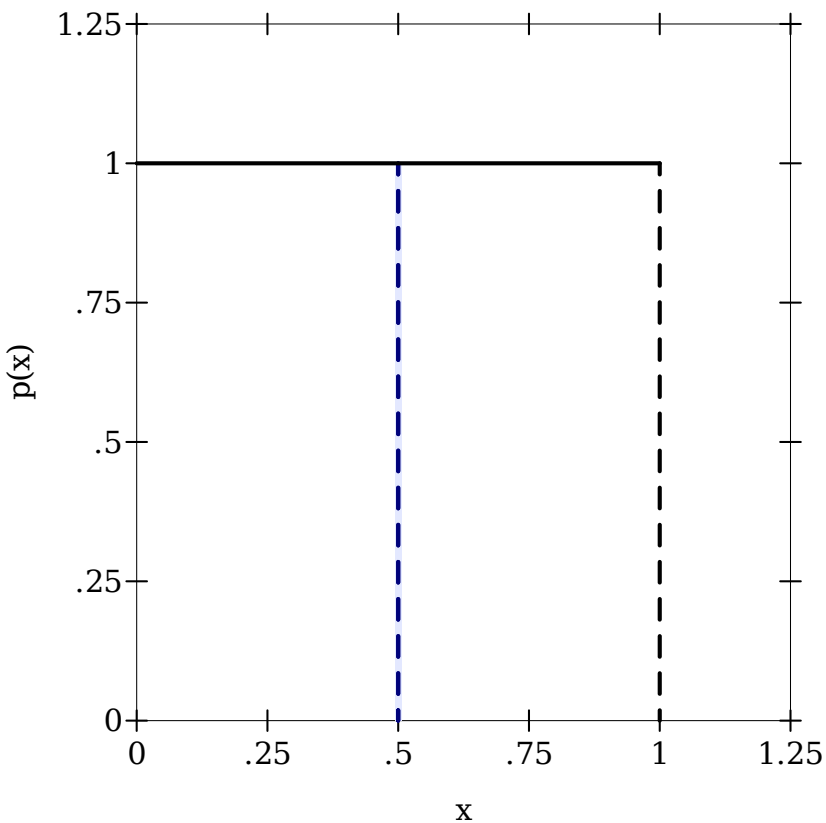


$$\Pr[(\texttt{random}) \in [0.5, 1]]$$
$$= \int_{0.5}^{1} p(x)\, dx$$
$$= 1 - 0.5$$
$$= 0.5$$

Simpler Example

- Assume `(random)` returns a value uniformly in $[0, 1]$

Density function p for value of `(random)`:

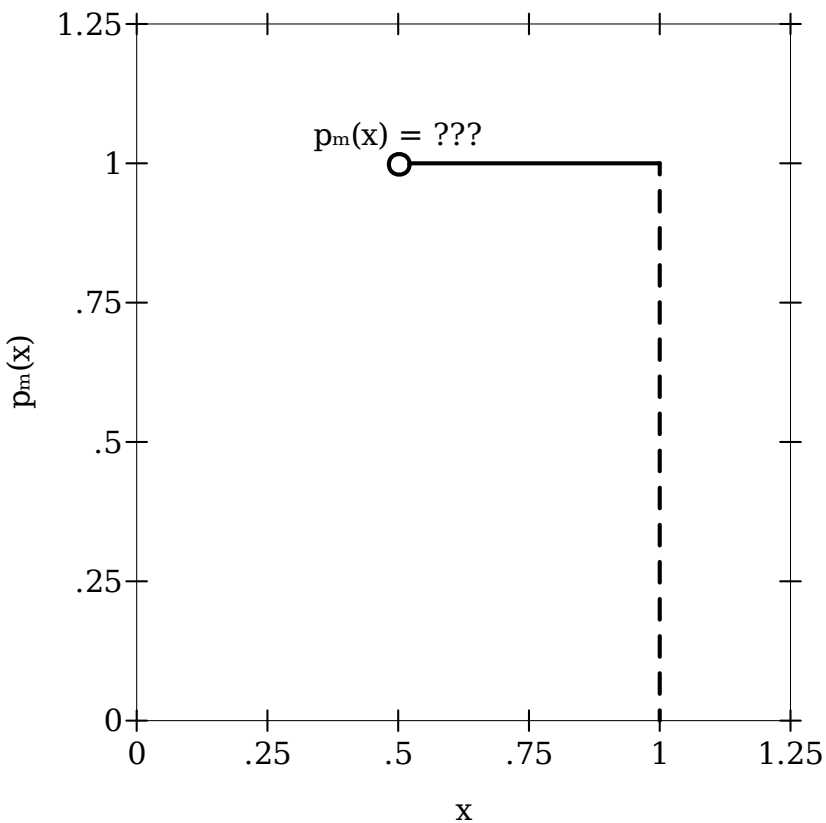


$$\Pr[(\texttt{random}) \in [0.5, 0.5]]$$
$$= \int_{0.5}^{0.5} p(x)\, dx$$
$$= 0.5 - 0.5$$
$$= 0$$

Simpler Example

- Assume `(random)` returns a value uniformly in $[0, 1]$

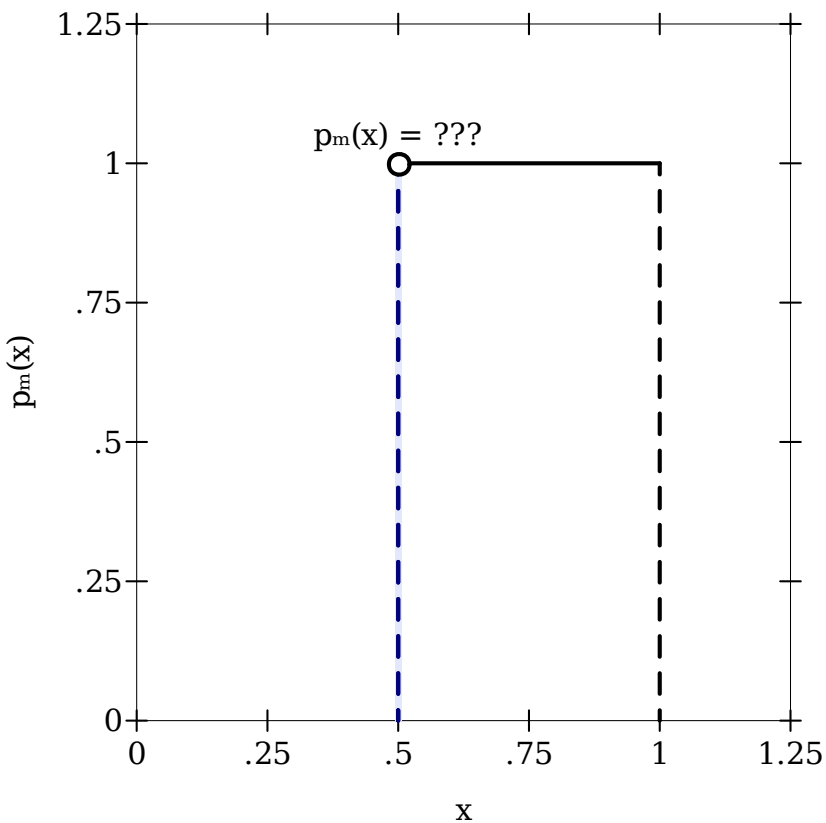
Density function p_m for value of `(max 0.5 (random))`:



Simpler Example

- Assume `(random)` returns a value uniformly in $[0, 1]$

Density function p_m for value of `(max 0.5 (random))`:

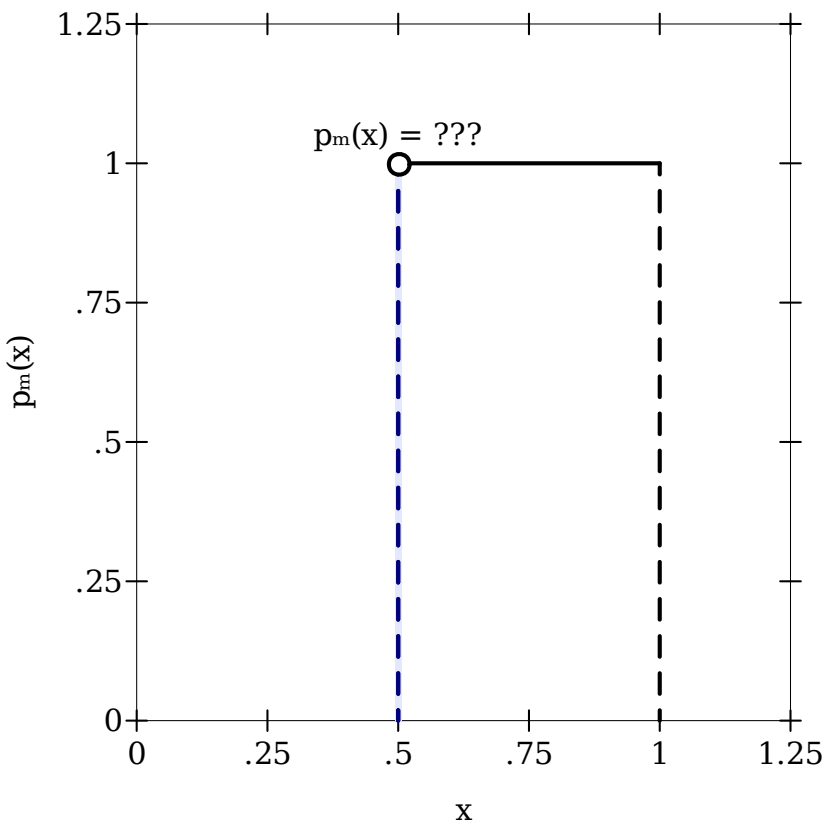


$$\Pr[(\texttt{max } 0.5 \ (\texttt{random})) \in [0.5, 0.5]] = 0.5$$

Simpler Example

- Assume `(random)` returns a value uniformly in $[0, 1]$

Density function p_m for value of `(max 0.5 (random))`:

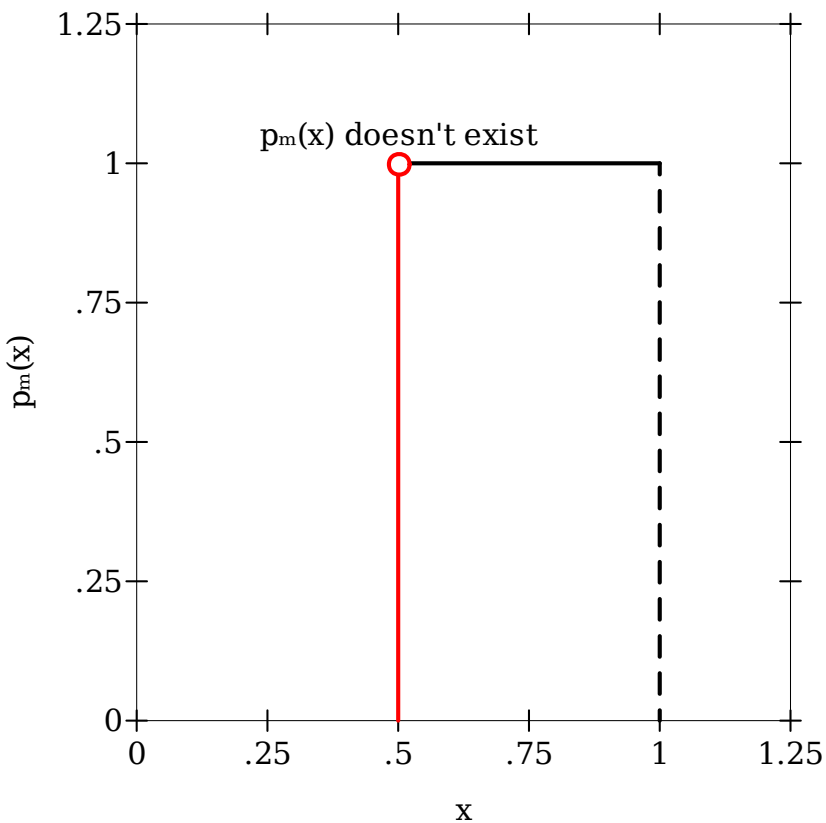


$$\Pr[(\texttt{max } 0.5 \ (\texttt{random})) \in [0.5, 0.5]]$$
$$= 0.5$$
$$= \int_{0.5}^{0.5} p_m(x)\, dx$$
$$= 0$$

Simpler Example

- Assume `(random)` returns a value uniformly in $[0, 1]$

Density function p_m for value of `(max 0.5 (random))`:



$$\Pr[(\texttt{max } 0.5 \ (\texttt{random})) \in [0.5, 0.5]]$$
$$= 0.5$$
$$= \int_{0.5}^{0.5} p_m(x)\, dx$$
$$= 0$$

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)

```
(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)
- 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 $P : \mathcal{P}([0,1]) \rightharpoonup [0,1]$, 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 `(random)` is $P : \mathcal{P}([0,1]) \rightharpoonup [0,1]$, defined by

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

Probability Measures

- Like already-integrated densities, but a primitive concept
- Measure of `(random)` is $P : \mathcal{P}([0,1]) \rightharpoonup [0,1]$, defined by

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

- Measure of `(max 0.5 (random))` defined by

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

Probability Measures

- Like already-integrated densities, but a primitive concept
- Measure of `(random)` is $P : \mathcal{P}([0,1]) \rightharpoonup [0,1]$, 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 $P : \mathcal{P}([0,1]) \rightharpoonup [0,1]$, 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 (random))` as $f : [0, 1] \to \mathbb{R}$, defined

$$f(r) = \max(0.5, r)$$

Probability Measures Via Preimages

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

$$f(r) \;=\; \max(0.5, r)$$

- *Derive* measure of `(max 0.5 (random))` as

$$P_m(B) = P(f^{-1}(B))$$

Probability Measures Via Preimages

- Interpret `(max 0.5 (random))` as $f : [0, 1] \to \mathbb{R}$, defined

$$f(r) \;=\; \max(0.5, r)$$

- *Derive* measure of `(max 0.5 (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 (random))` as $f : [0, 1] \to \mathbb{R}$, defined

$$f(r) \;=\; \max(0.5, r)$$

- *Derive* measure of `(max 0.5 (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 (random))` as $f : [0, 1] \rightarrow \mathbb{R}$, defined

$$f(r) \;=\; \max(0.5, r)$$

- *Derive* measure of `(max 0.5 (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 `(max 0.5 (random))` as $f : [0,1] \to \mathbb{R}$, defined

$$f(r) \;=\; \max(0.5, r)$$

- *Derive* measure of `(max 0.5 (random))` as

$$P_m(B) = P(f^{-1}(B))$$

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

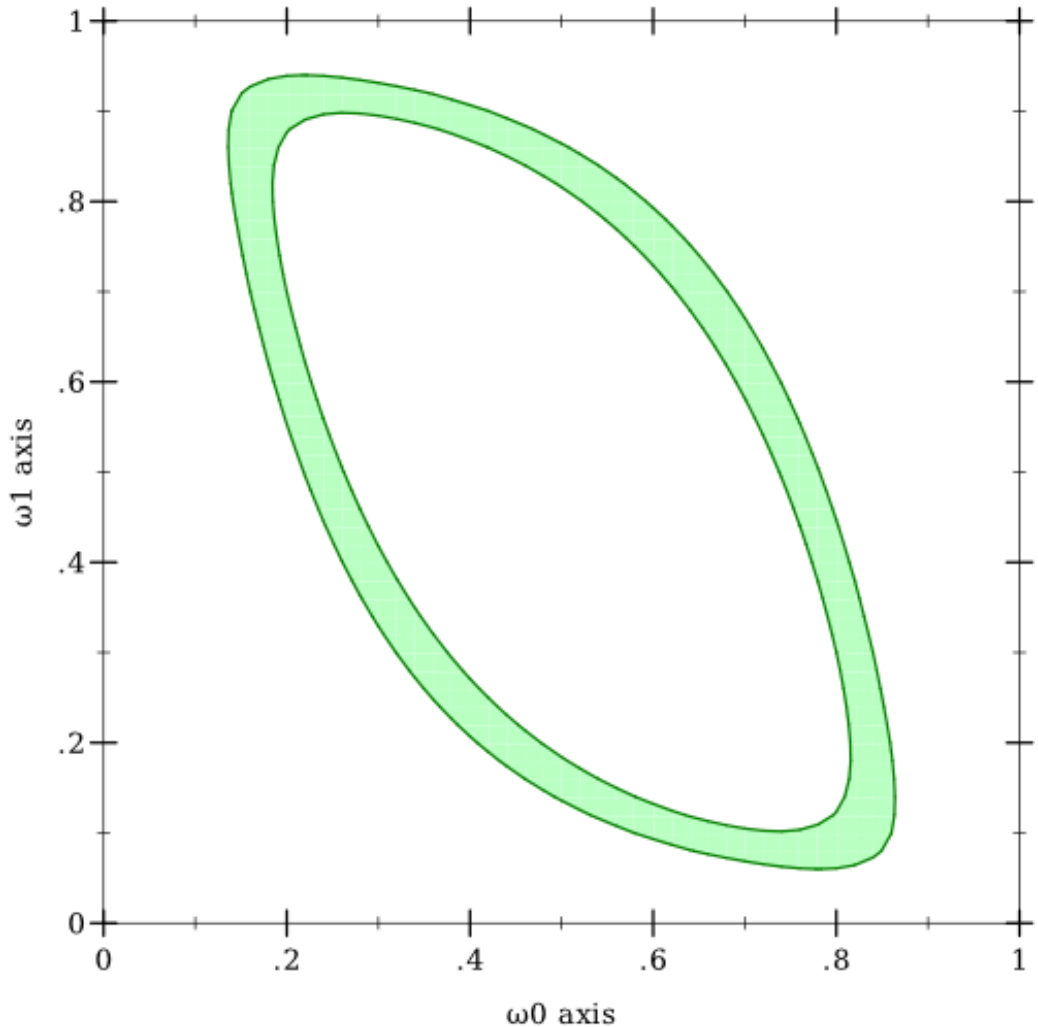
- Factored into random and deterministic parts:

$$P_m = P \circ f^{-1}$$

- In other words, compute measures of expressions by *running them backwards*... **on possibly uncountable sets**

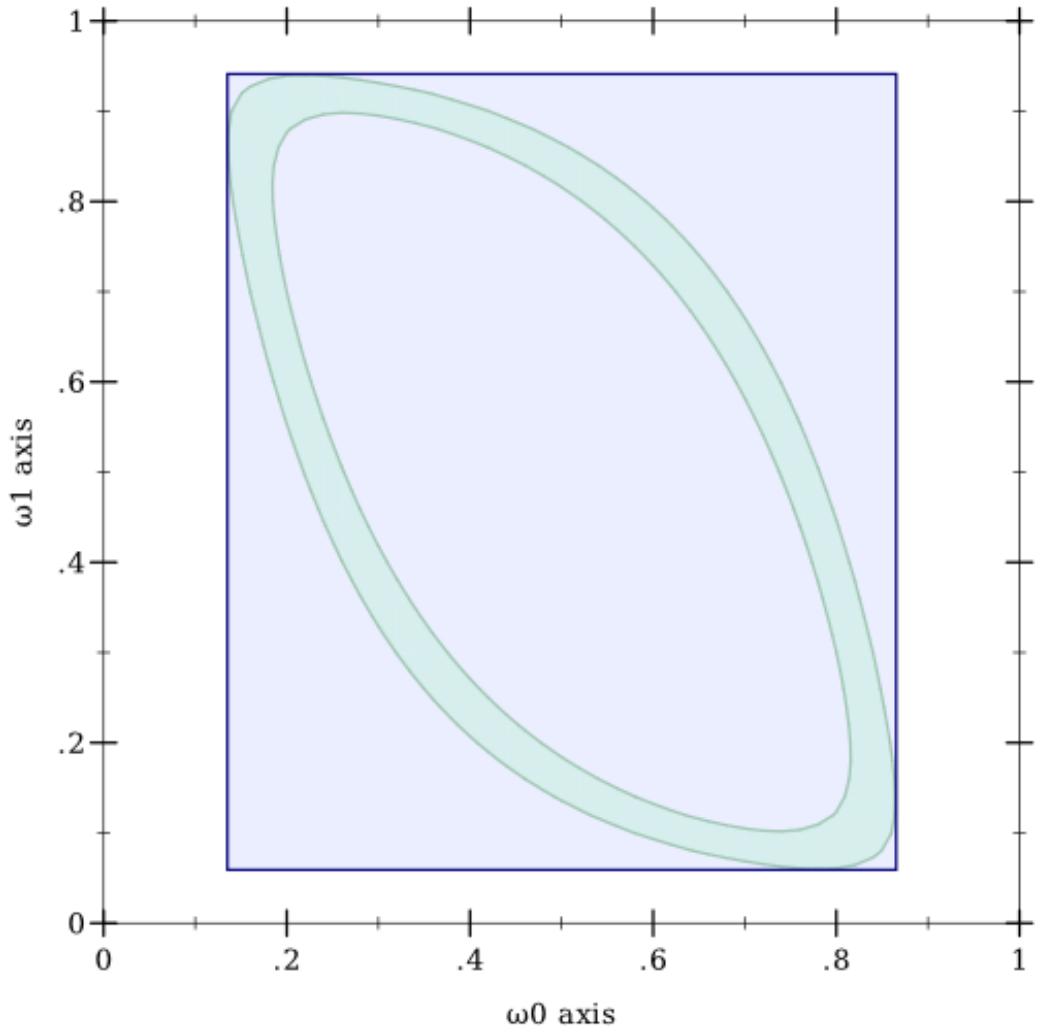
What About Approximating?

Conservative approximation with rectangles:



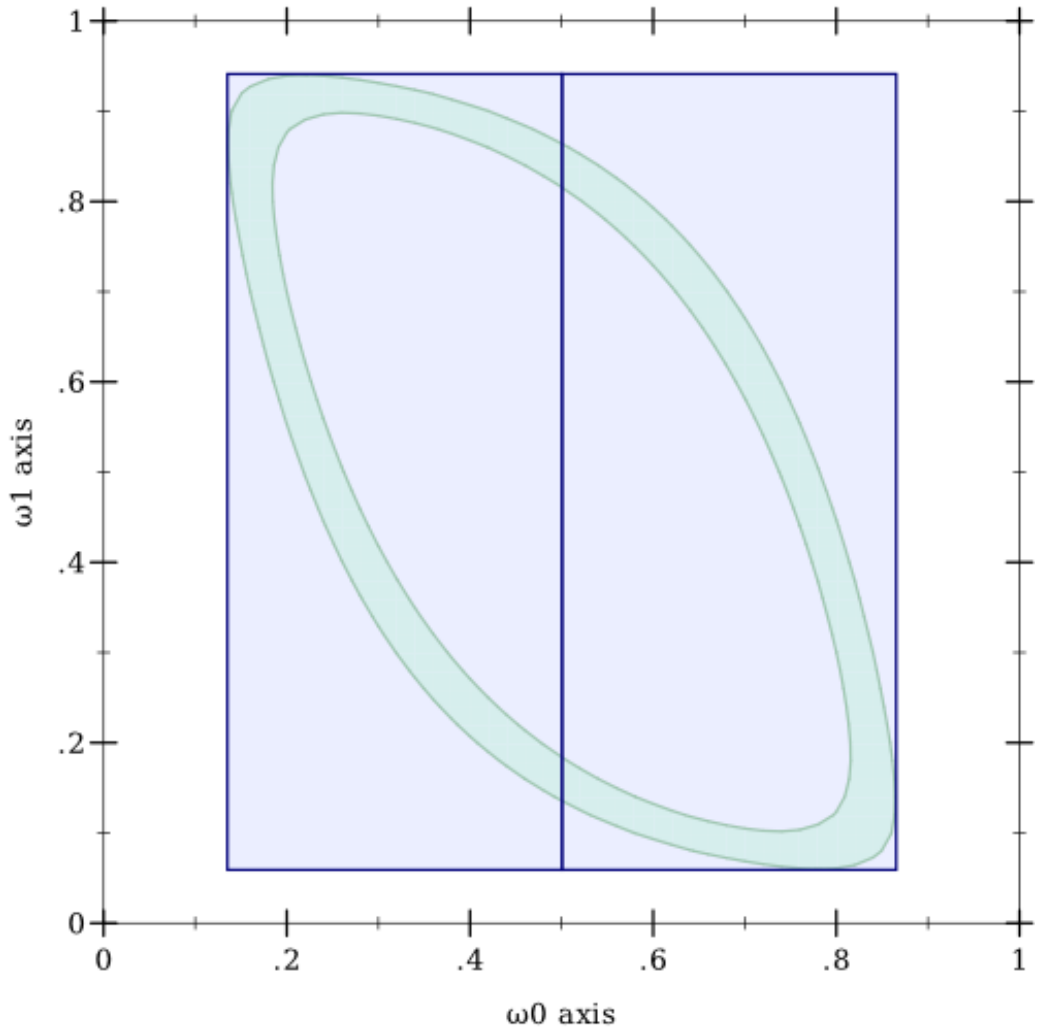
What About Approximating?

Conservative approximation with rectangles:



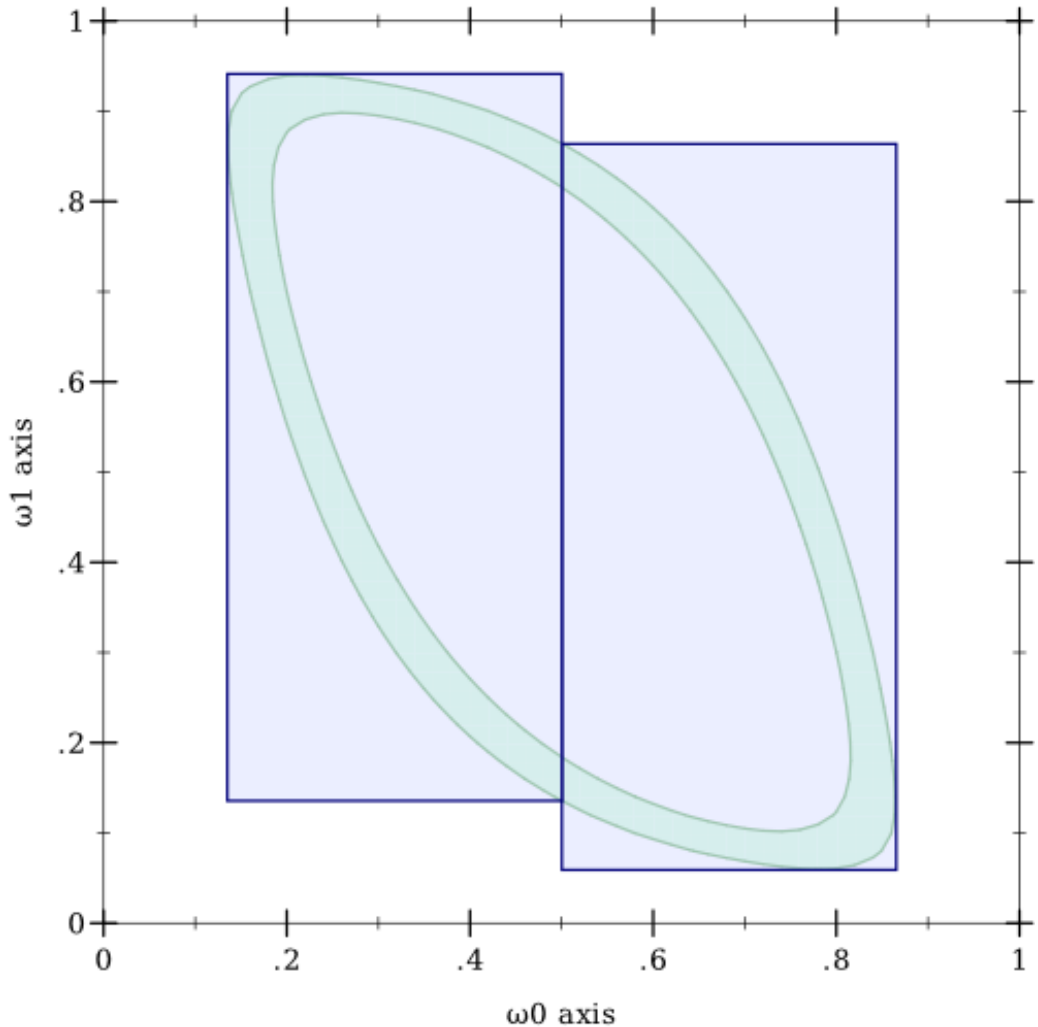
What About Approximating?

Restricting preimages to rectangular subdomains:



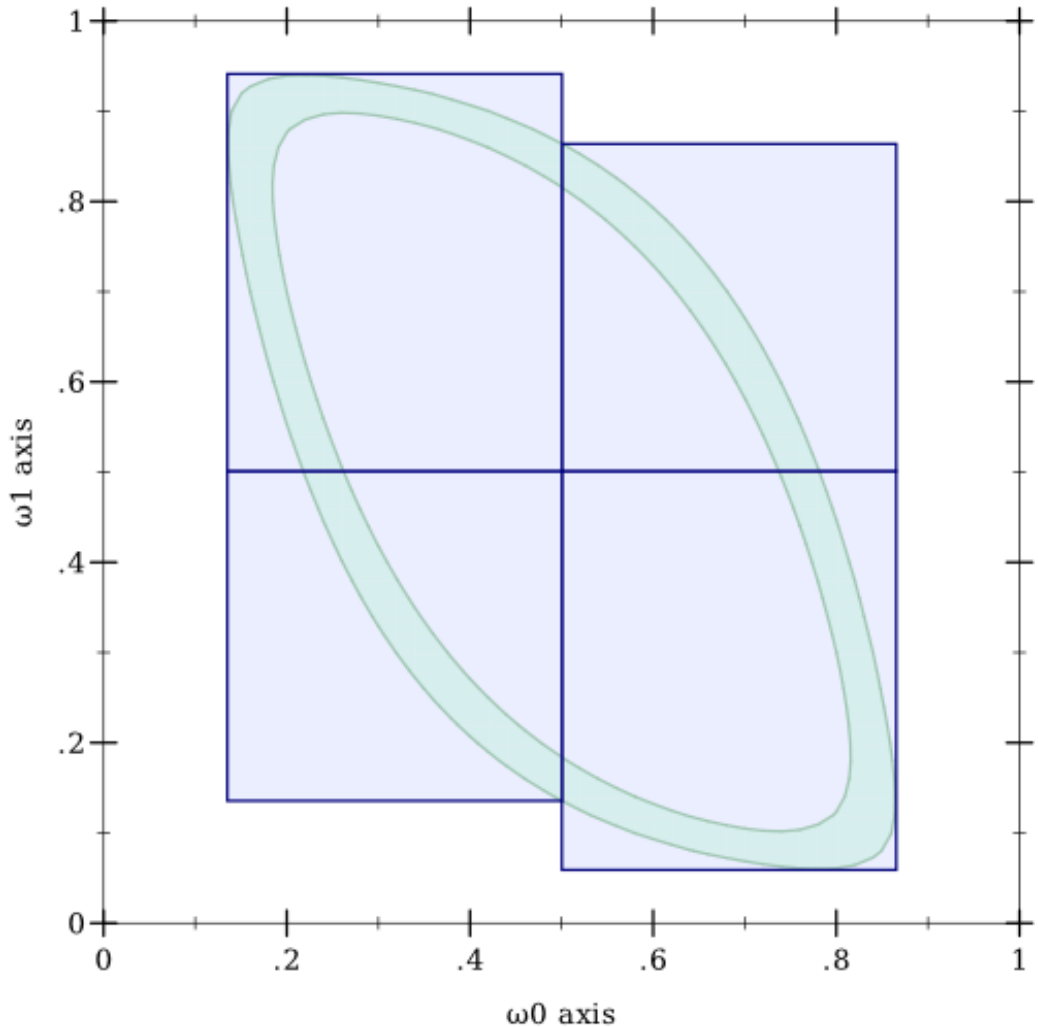
What About Approximating?

Restricting preimages to rectangular subdomains:



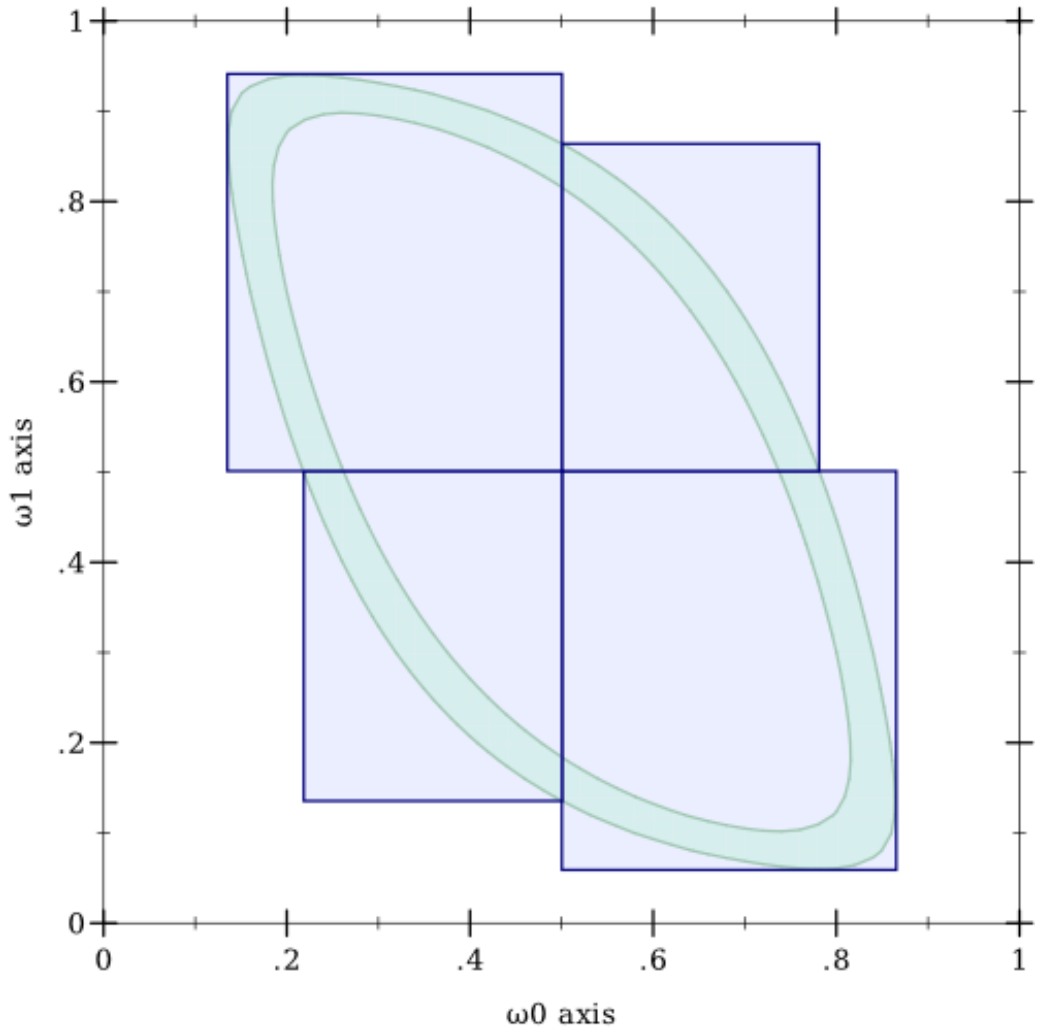
What About Approximating?

Restricting preimages to rectangular subdomains:



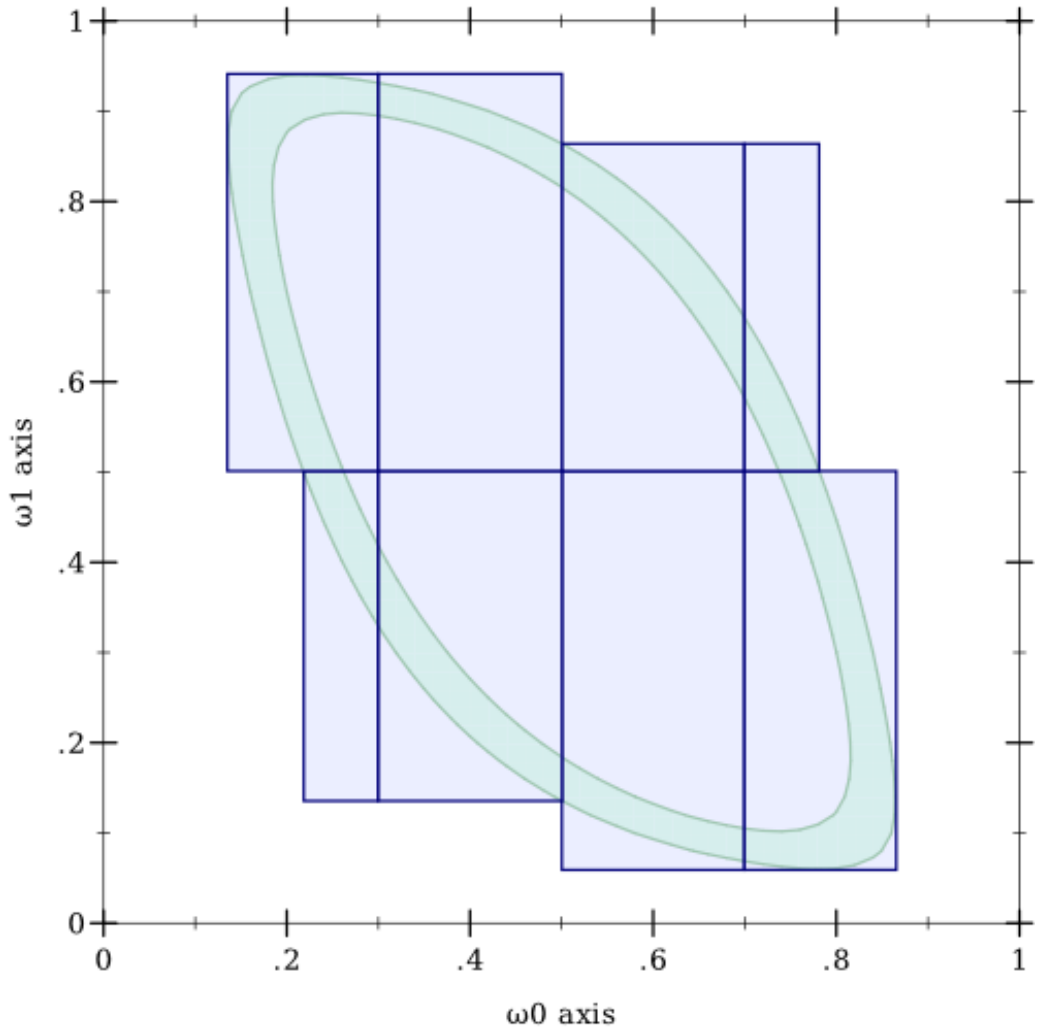
What About Approximating?

Restricting preimages to rectangular subdomains:



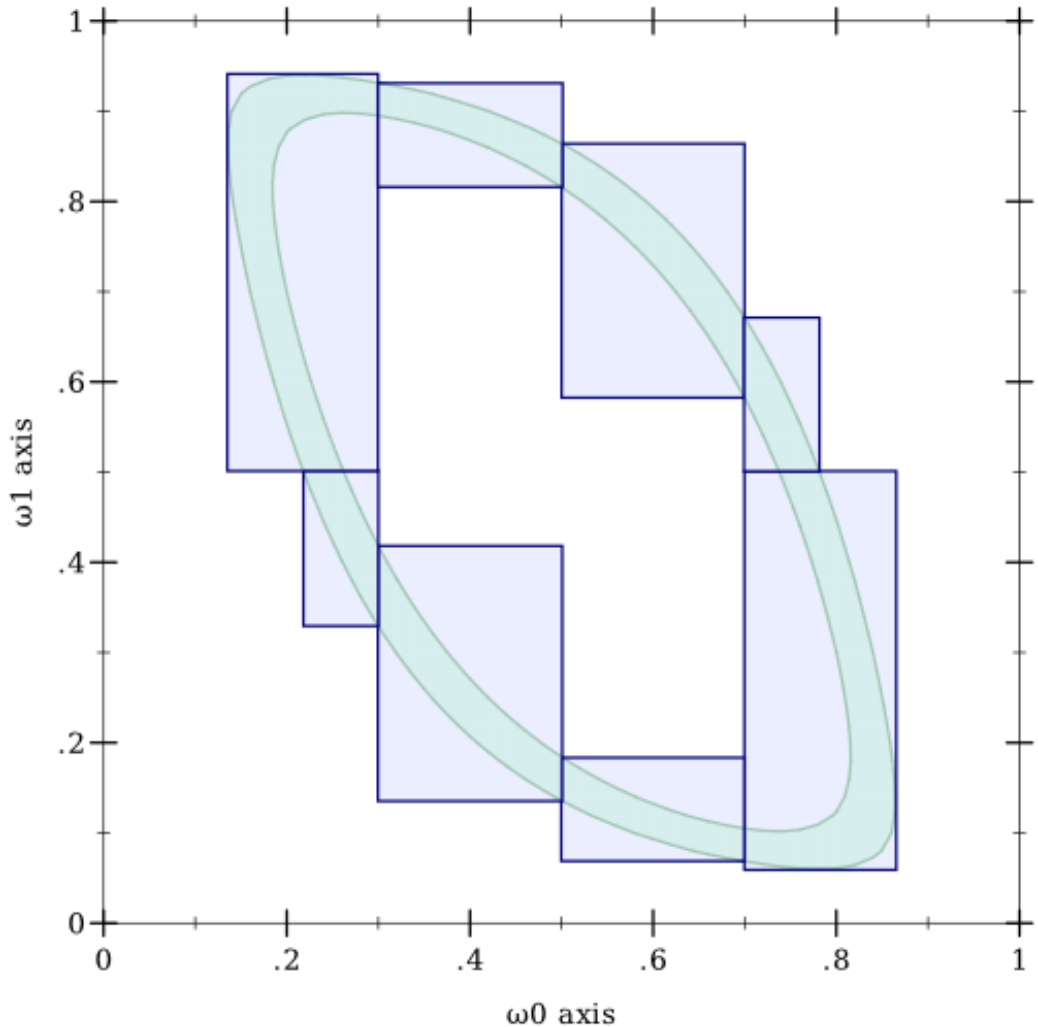
What About Approximating?

Restricting preimages to rectangular subdomains:



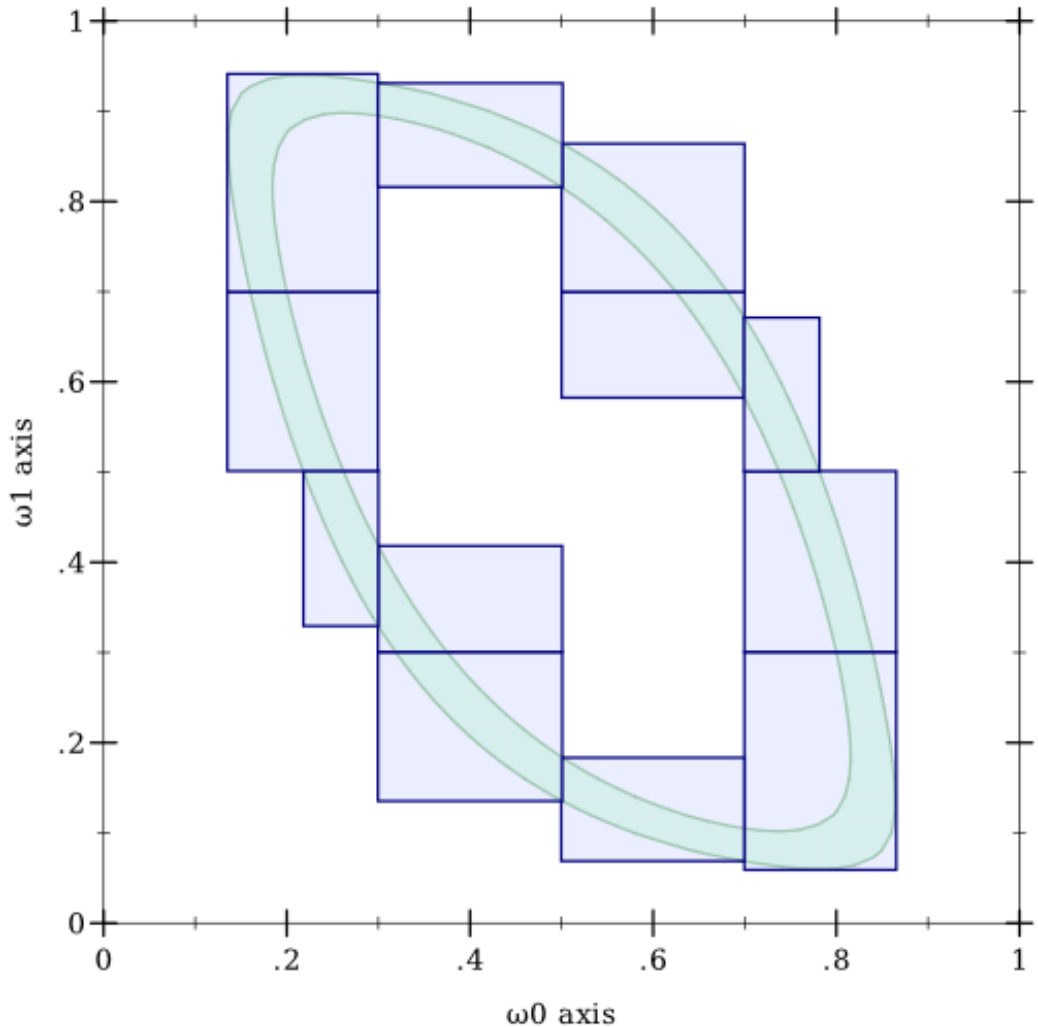
What About Approximating?

Restricting preimages to rectangular subdomains:



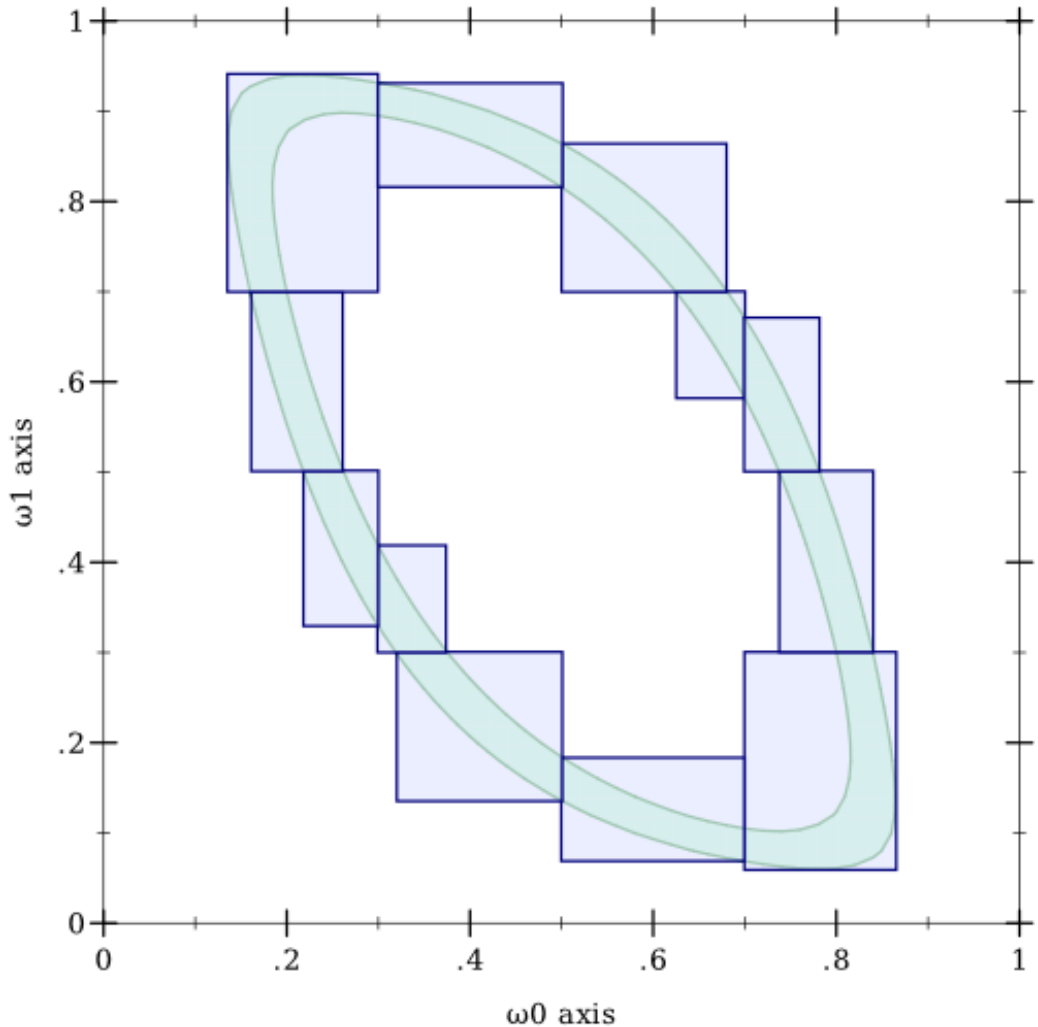
What About Approximating?

Restricting preimages to rectangular subdomains:



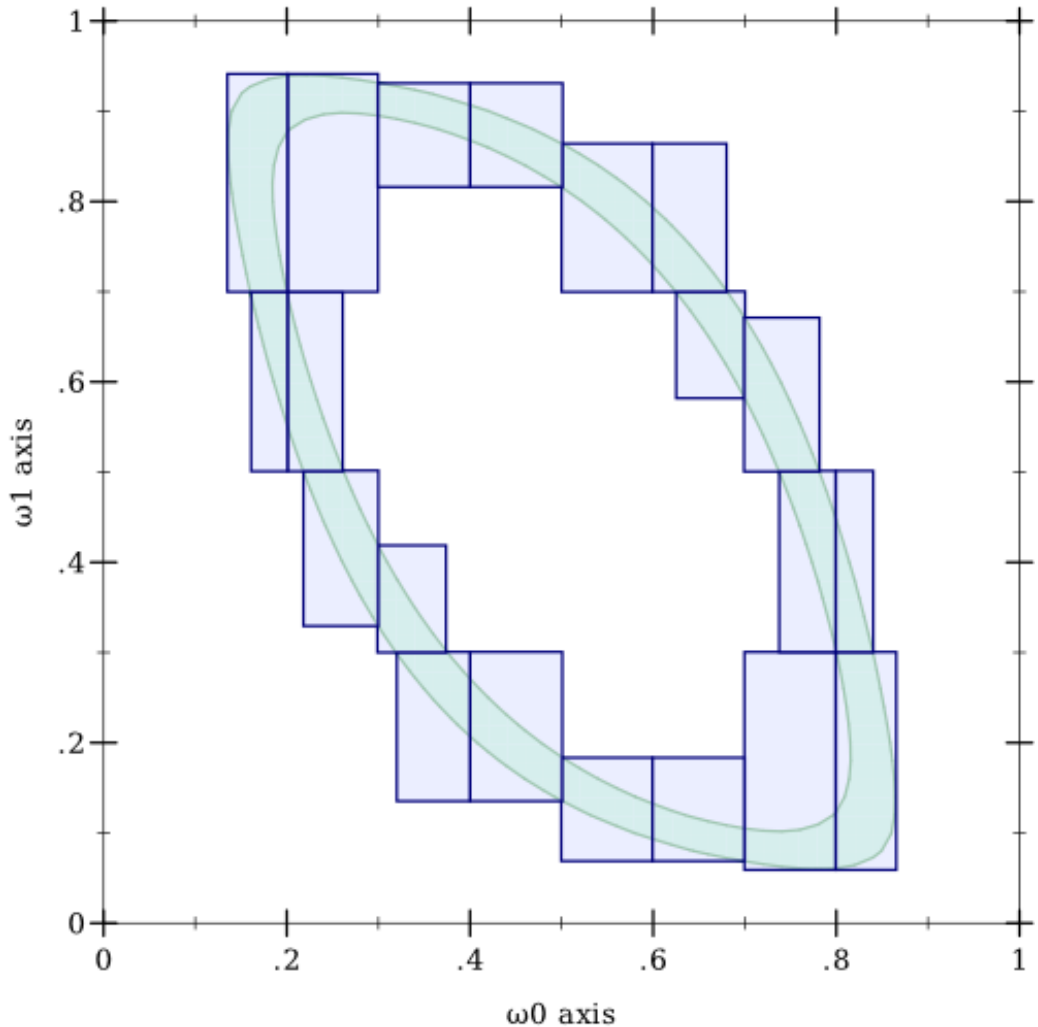
What About Approximating?

Restricting preimages to rectangular subdomains:



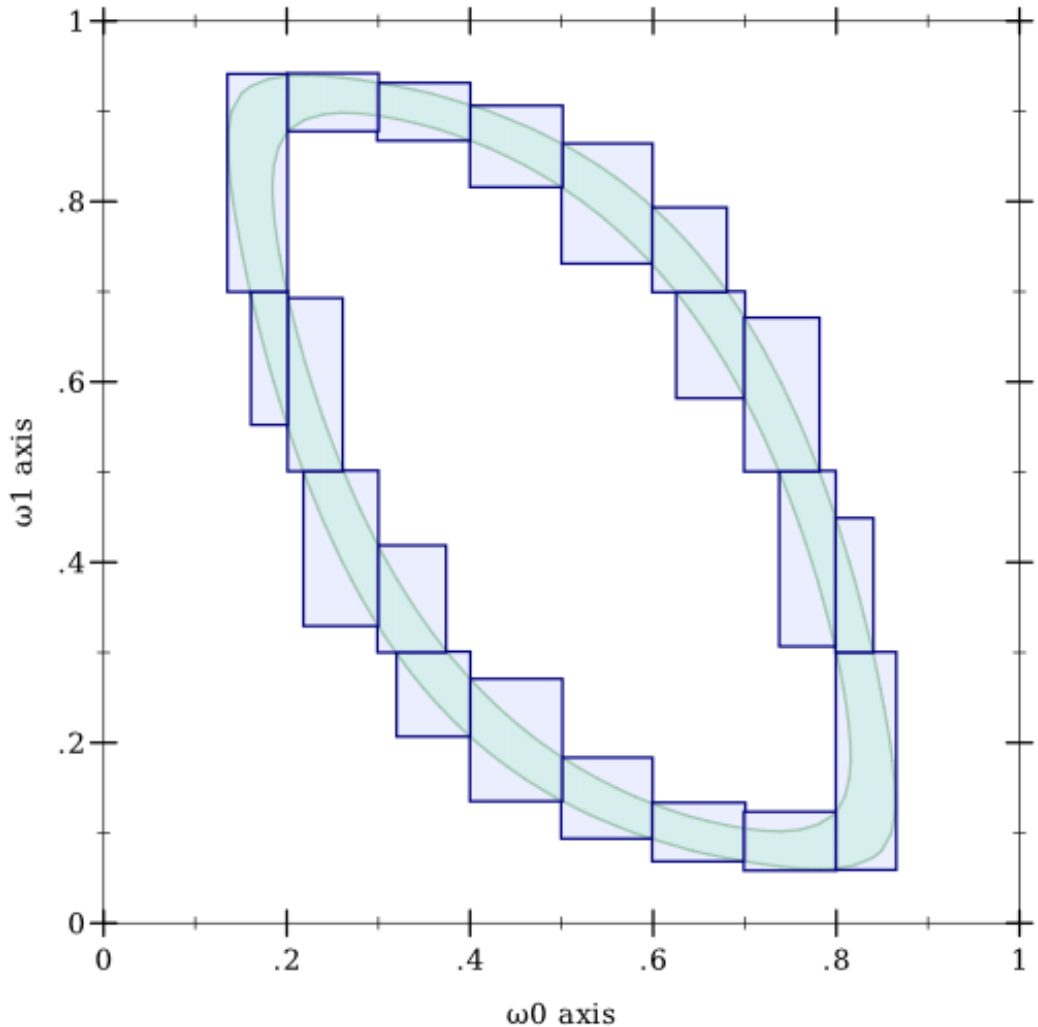
What About Approximating?

Restricting preimages to rectangular subdomains:



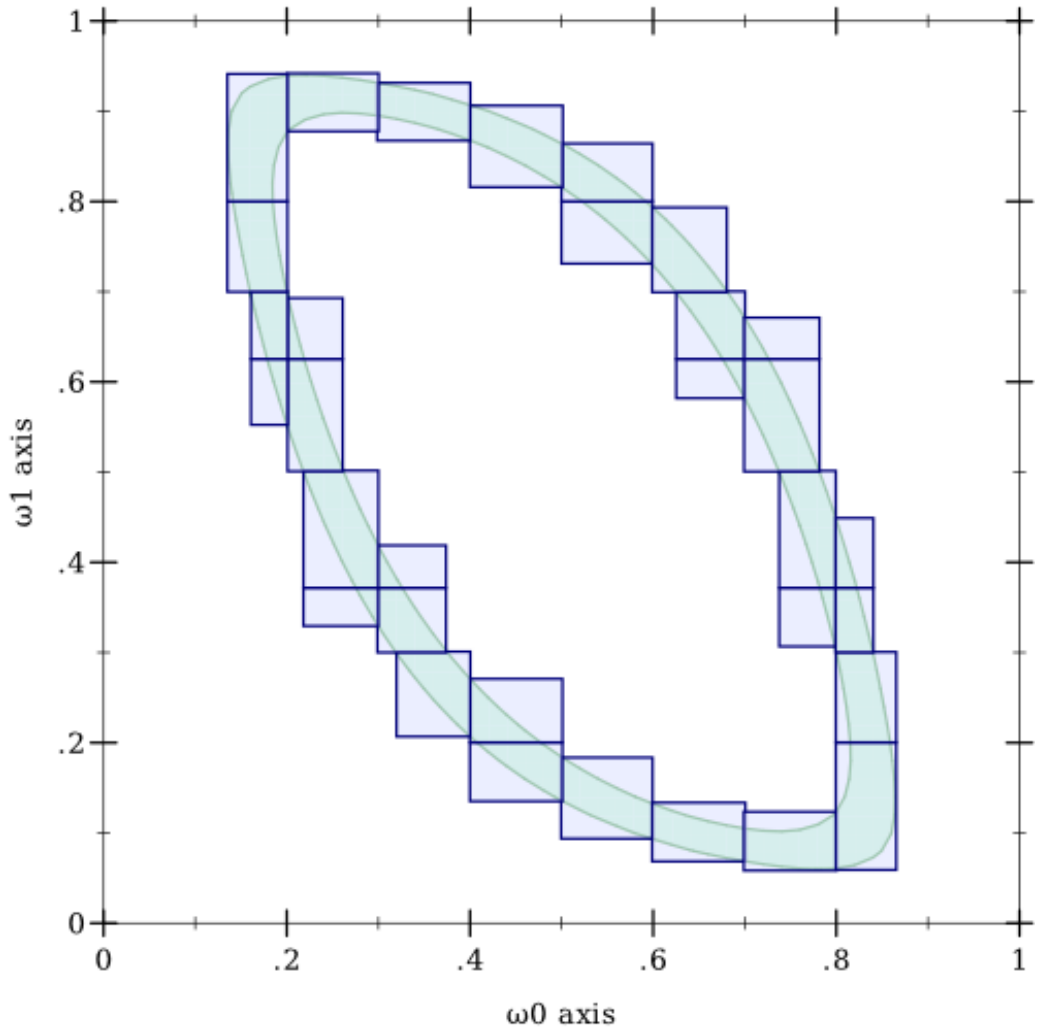
What About Approximating?

Restricting preimages to rectangular subdomains:



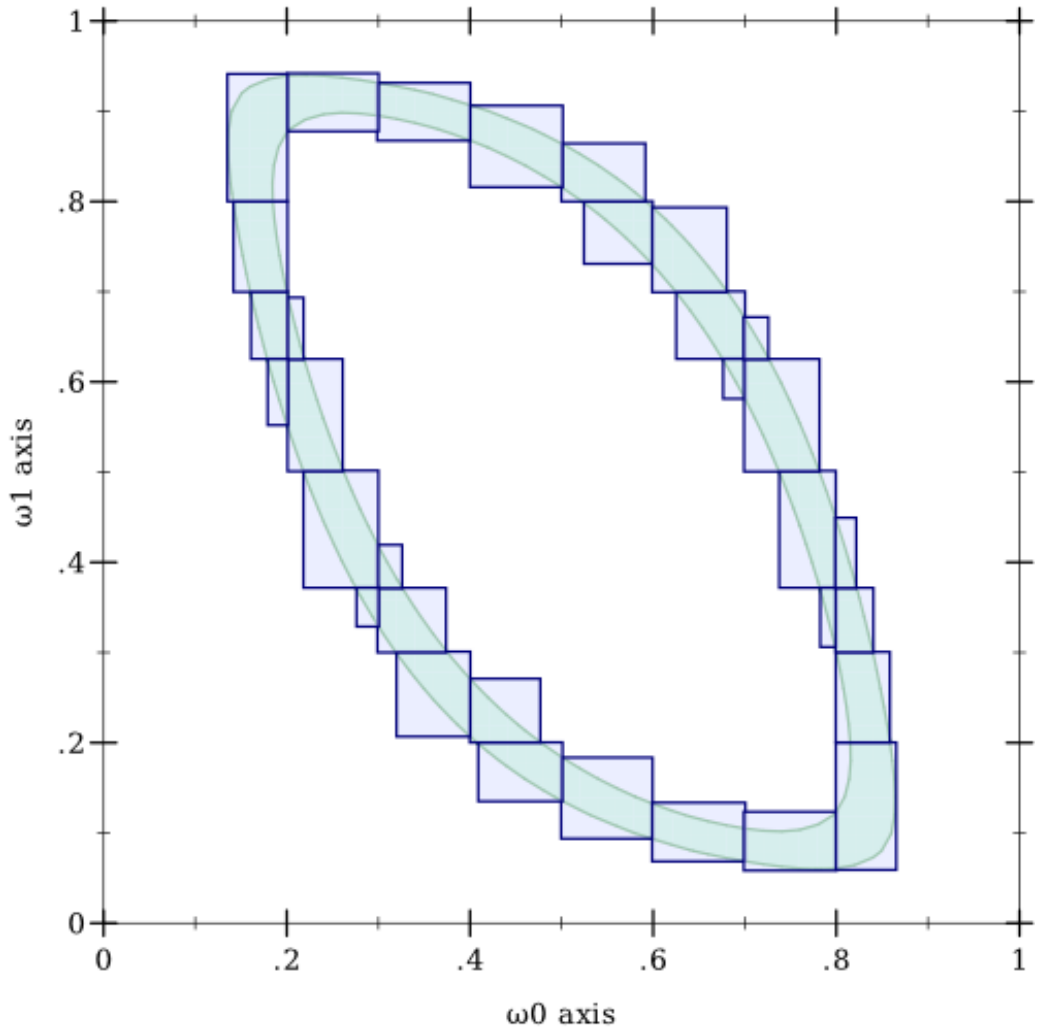
What About Approximating?

Restricting preimages to rectangular subdomains:



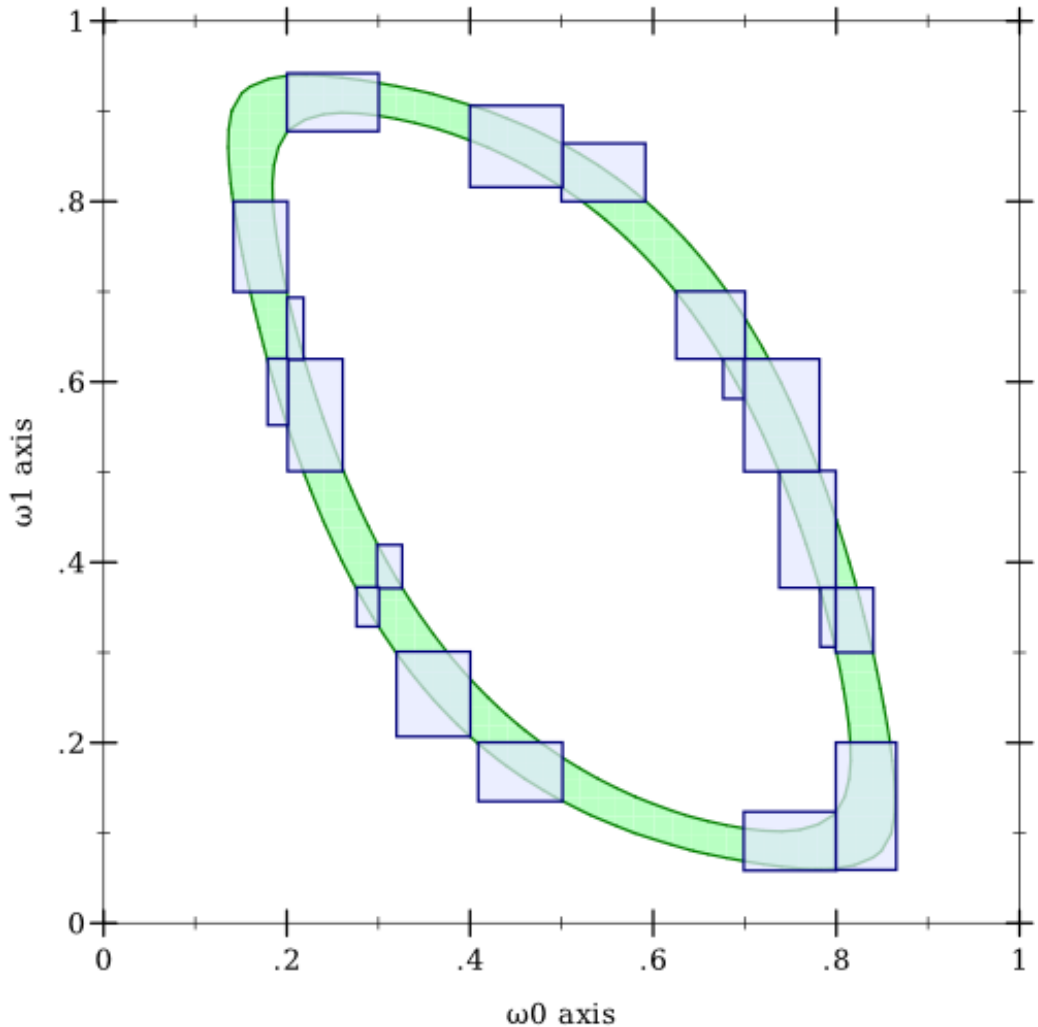
What About Approximating?

Restricting preimages to rectangular subdomains:



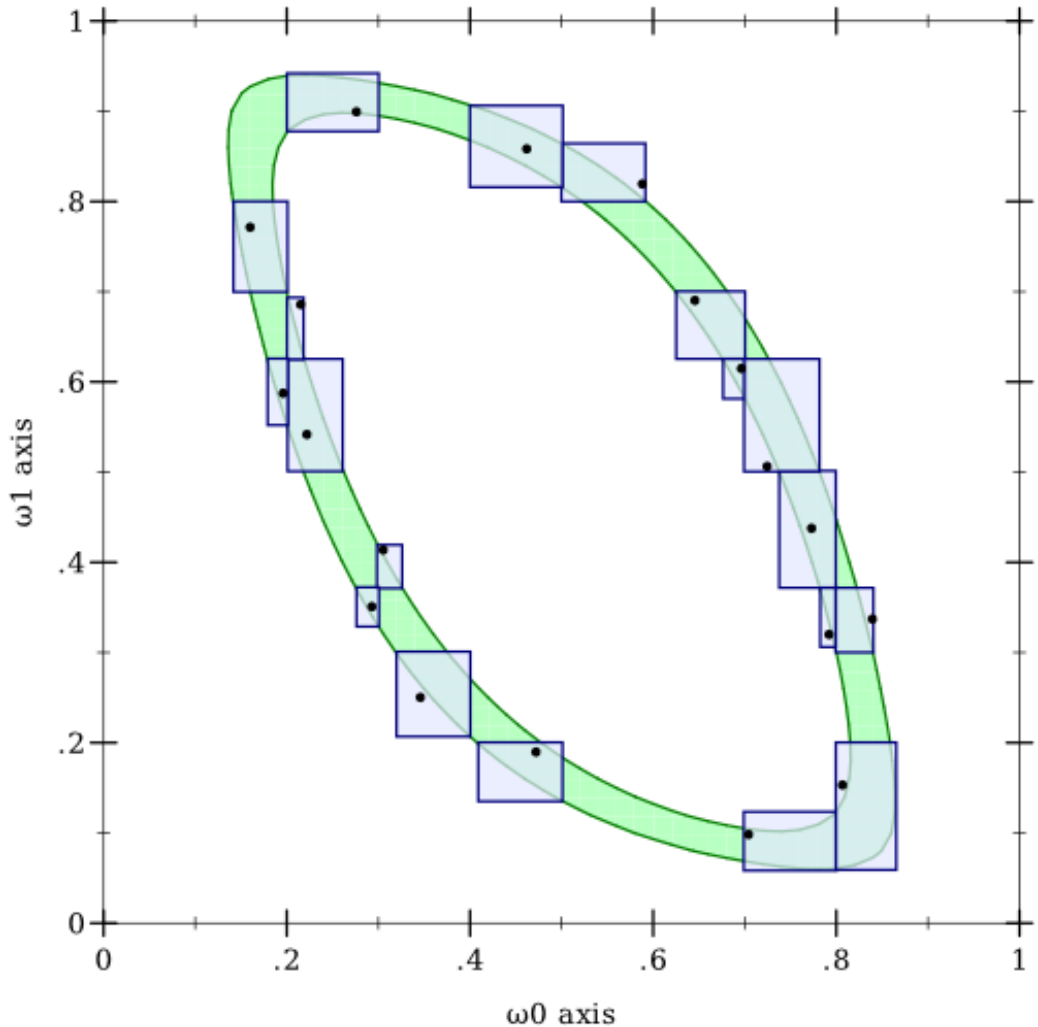
What About Approximating?

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

PL **ML**

Ideals and Motivation: PL vs. ML

PL	ML
99.999% certainty isn't enough	

Ideals and Motivation: PL vs. ML

PL	ML
99.999% certainty isn't enough	Even 99% certainty isn't feasible

Ideals and Motivation: PL vs. ML

PL	ML
99.999% certainty isn't enough	Even 99% certainty isn't feasible
Only sound approximations are admissible: we must provide guarantees	

Ideals and Motivation: PL vs. ML

PL	ML
99.999% certainty isn't enough	Even 99% certainty isn't feasible
Only sound approximations are admissible: we must provide guarantees	Logical reasoning doesn't scale, so we can't provide guarantees

Ideals and Motivation: PL vs. ML

PL	ML
99.999% certainty isn't enough	Even 99% certainty isn't feasible
Only sound approximations are admissible: we must provide guarantees	Logical reasoning doesn't scale, so we can't provide guarantees
Correctness on all inputs is desirable and achievable	

Ideals and Motivation: PL vs. ML

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

Ideals and Motivation: PL vs. ML

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

All of the above statements are true

Ideals and Motivation: PL vs. ML

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

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 classifer that determines whether actions violate ACME's EULAs

ML Research Context Example

- Example: Creating a classifer 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 classifer 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 classifer 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 classifer 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

$$[\![\cdot]\!] : e \to (R \to B)$$

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

$[\![\cdot]\!] : e \to (R \to B)$ **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

$[\![\cdot]\!]_{\text{pre}} : e \rightarrow \text{preimage functions}$

$[\![\cdot]\!] : e \rightarrow (R \rightarrow B)$ **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

$[\![\cdot]\!]_{\text{pre}} : e \rightarrow \text{preimage functions}$	**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

$[\![\cdot]\!]_{\widehat{\mathrm{pre}}} : e \to$ approximating preimage functions

$[\![\cdot]\!]_{\mathrm{pre}} : e \to$ preimage functions **correct**

$[\![\cdot]\!] : e \to (R \to B)$ **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

$[\![\cdot]\!]_{\widehat{\mathrm{pre}}} : e \to \text{approximating preimage functions}$	**sound**
$[\![\cdot]\!]_{\mathrm{pre}} : e \to \text{preimage functions}$	**correct**
$[\![\cdot]\!] : e \to (R \to B)$	**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

Rectangular set library, inverse interval arithmetic

$[\![\cdot]\!]_{\widehat{\mathrm{pre}}} : e \to \text{approximating preimage functions}$ **sound**

$[\![\cdot]\!]_{\mathrm{pre}} : e \to \text{preimage functions}$ **correct**

$[\![\cdot]\!] : e \to (R \to B)$ **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

Rectangular set library, inverse interval arithmetic	**sound**
$[\![\cdot]\!]_{\widehat{\mathrm{pre}}} : e \to$ approximating preimage functions	**sound**
$[\![\cdot]\!]_{\mathrm{pre}} : e \to$ preimage functions	**correct**
$[\![\cdot]\!] : e \to (R \to B)$	**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

---

Rectangular set library, inverse interval arithmetic	**sound**
$[\![\cdot]\!]_{\widehat{\mathrm{pre}}} : e \to$ approximating preimage functions	**sound**
$[\![\cdot]\!]_{\mathrm{pre}} : e \to$ preimage functions	**correct**
$[\![\cdot]\!] : e \to (R \to B)$	**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

SOUNDNESS BARRIER DO NOT CROSS

Rectangular set library, inverse interval arithmetic	**sound**
$[\![\cdot]\!]_{\widehat{\text{pre}}} : e \to$ approximating preimage functions	**sound**
$[\![\cdot]\!]_{\text{pre}} : e \to$ preimage functions	**correct**
$[\![\cdot]\!] : e \to (R \to B)$	**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

Sampling algorithms	
SOUNDNESS BARRIER DO NOT CROSS	
Rectangular set library, inverse interval arithmetic	**sound**
$[\![\cdot]\!]_{\widehat{\text{pre}}} : e \to$ approximating preimage functions	**sound**
$[\![\cdot]\!]_{\text{pre}} : e \to$ preimage functions	**correct**
$[\![\cdot]\!] : e \to (R \to B)$	**assumed**

Closing the Gap in PPLs

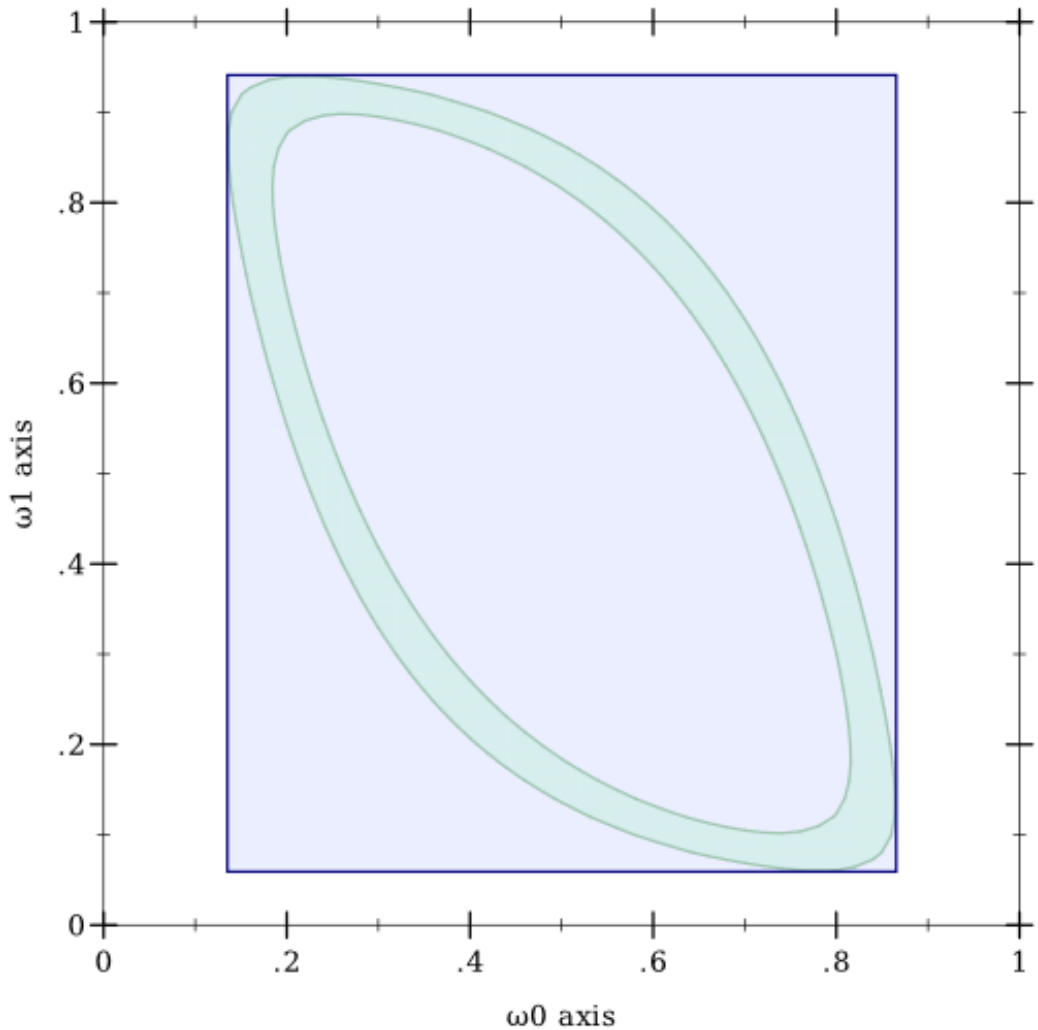
- PL ideals: Languages must be correct, provide guarantees
- ML ideals: Languages must provide approximations that scale

DrBayes satisfies (or dissatisfies) both sides by compartmentalizing

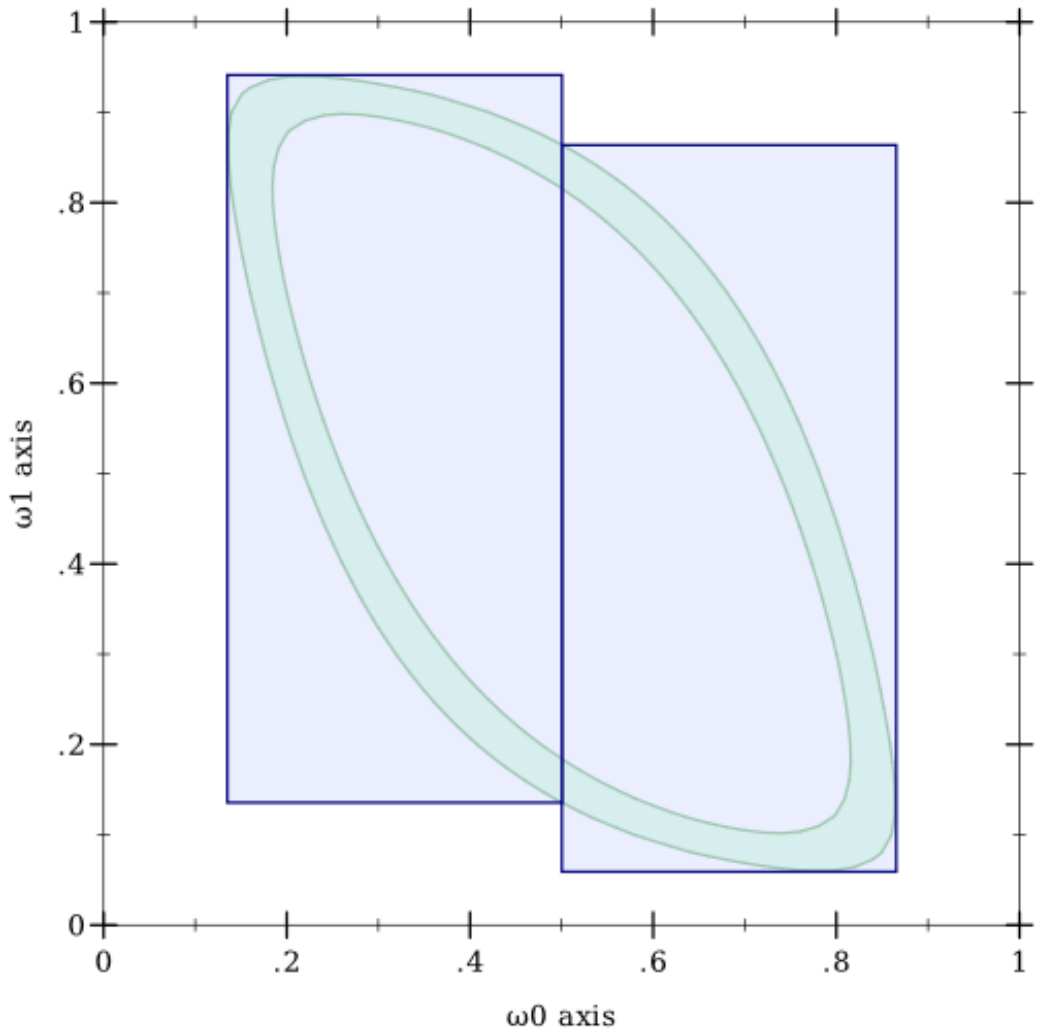
Sampling algorithms	**converge in probability**
SOUNDNESS BARRIER DO NOT CROSS	
Rectangular set library, inverse interval arithmetic	**sound**
$[\![\cdot]\!]_{\widehat{\text{pre}}} : e \to$ approximating preimage functions	**sound**
$[\![\cdot]\!]_{\text{pre}} : e \to$ preimage functions	**correct**
$[\![\cdot]\!] : e \to (R \to B)$	**assumed**

Motivating Soundness To ML

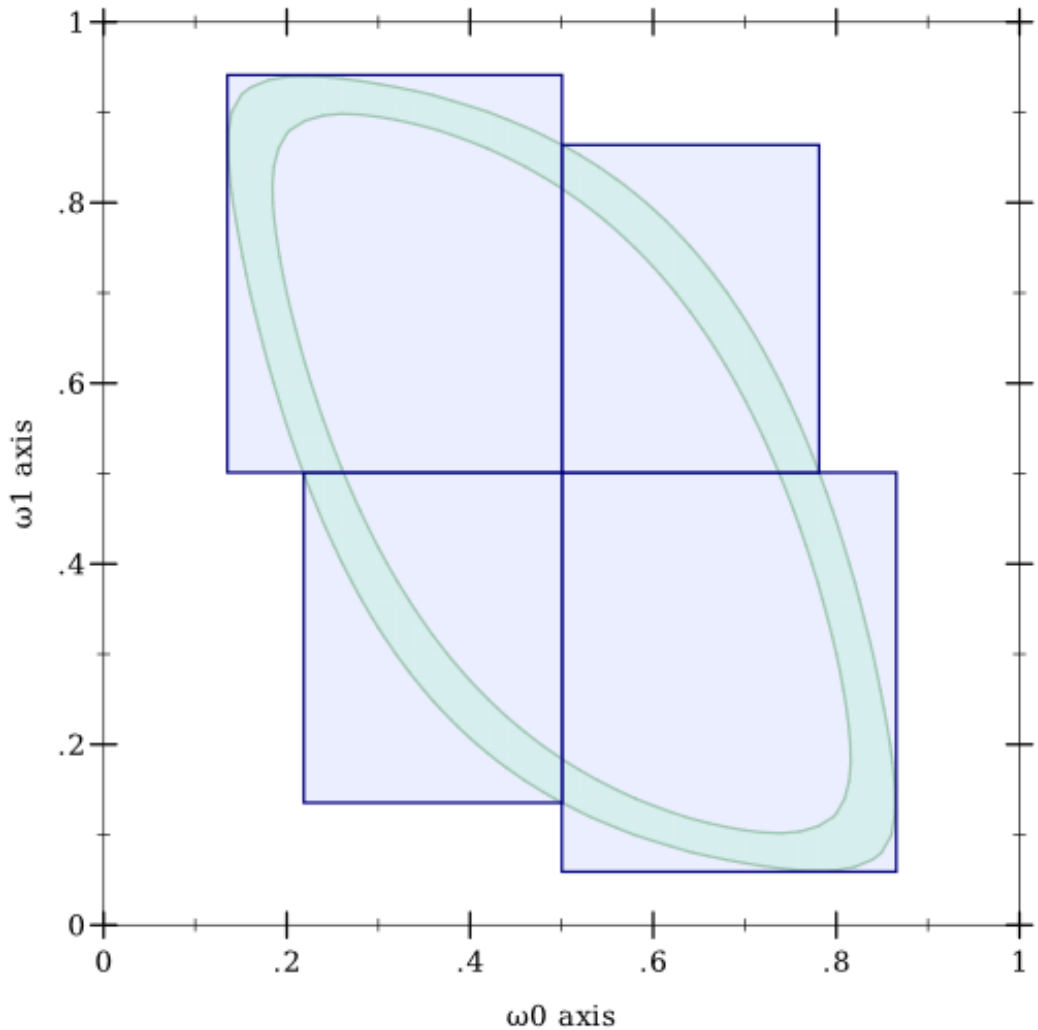
Motivating Soundness To ML



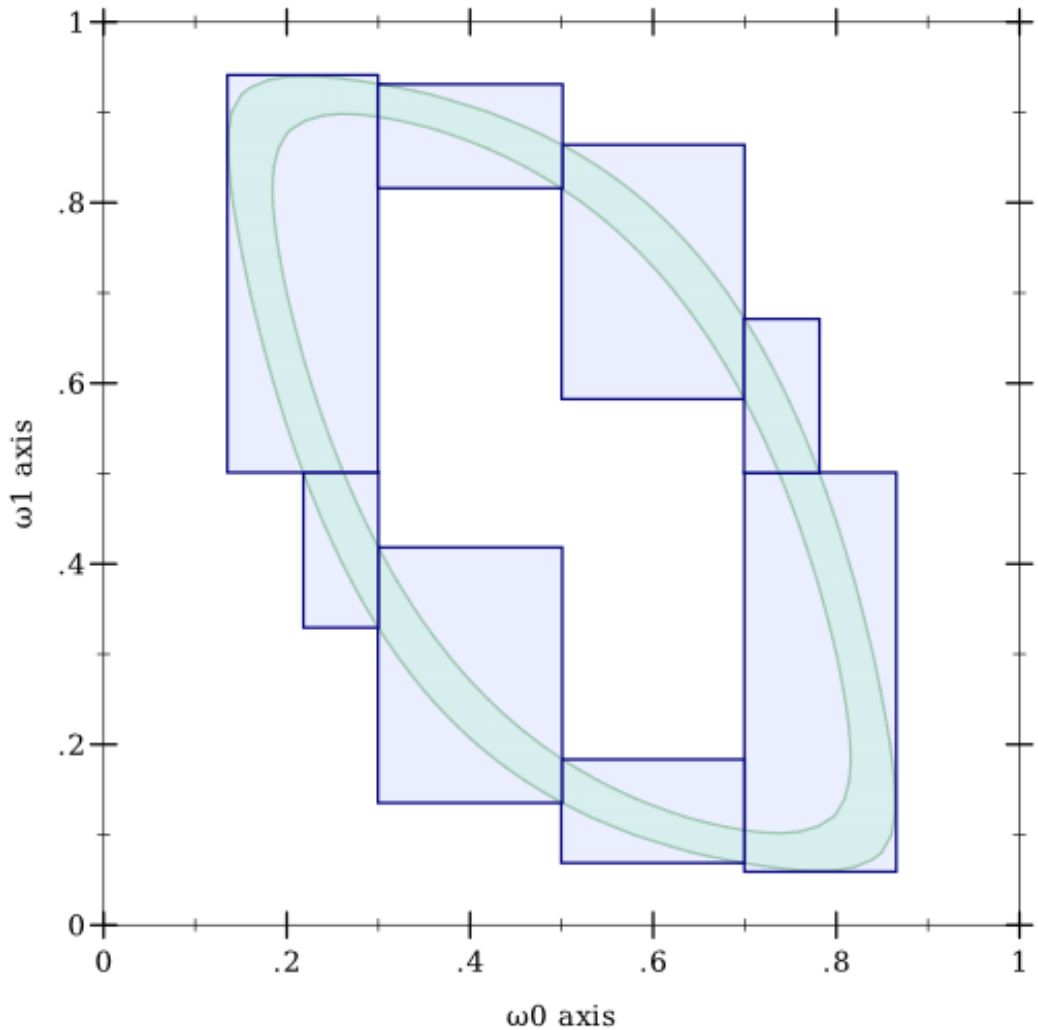
Motivating Soundness To ML



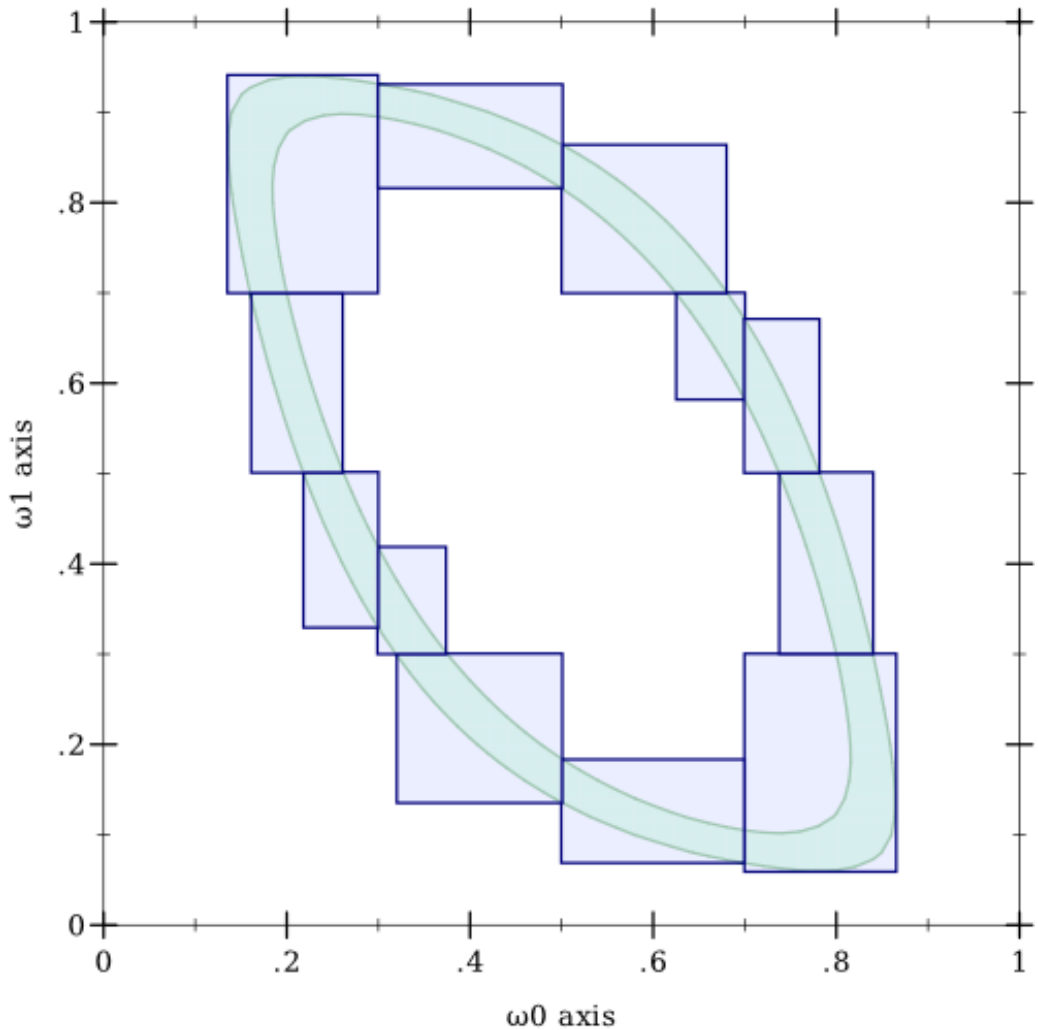
Motivating Soundness To ML



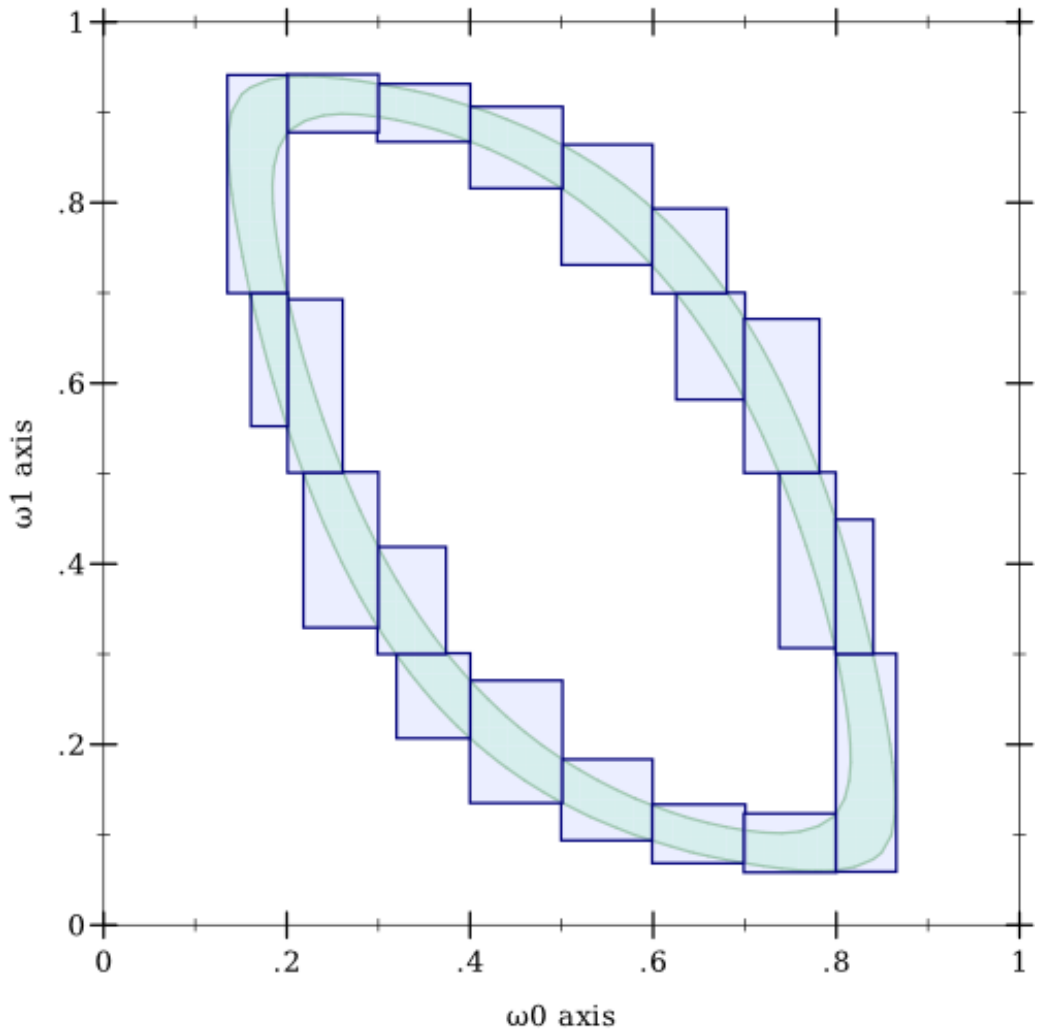
Motivating Soundness To ML



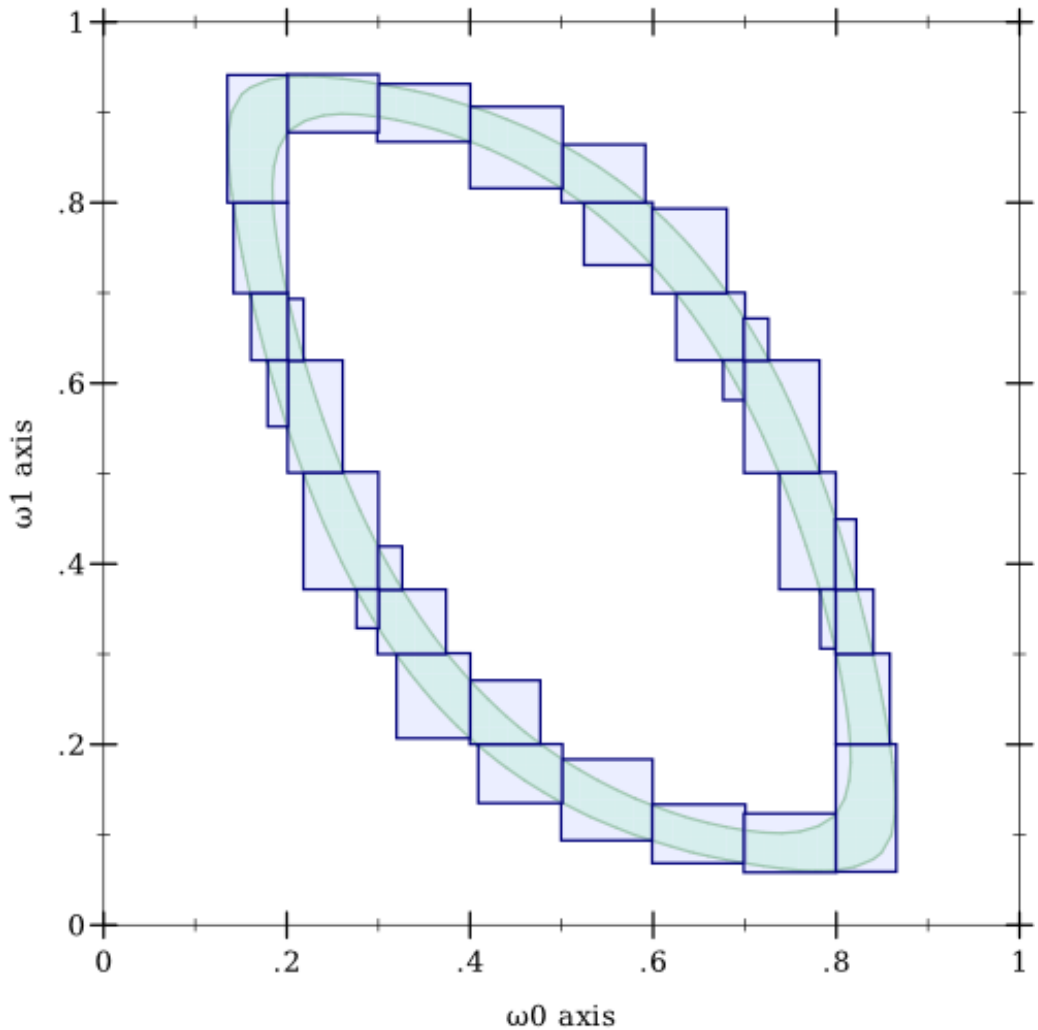
Motivating Soundness To ML



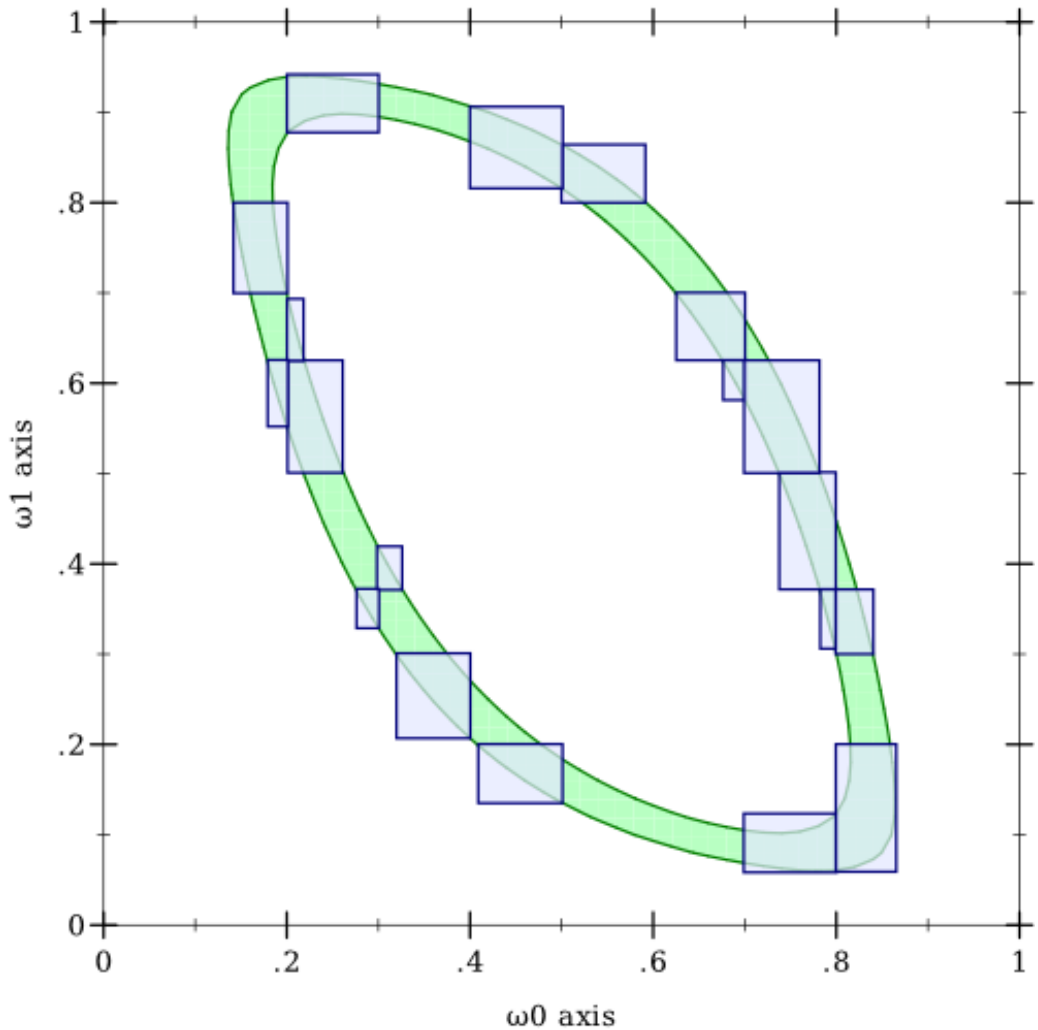
Motivating Soundness To ML



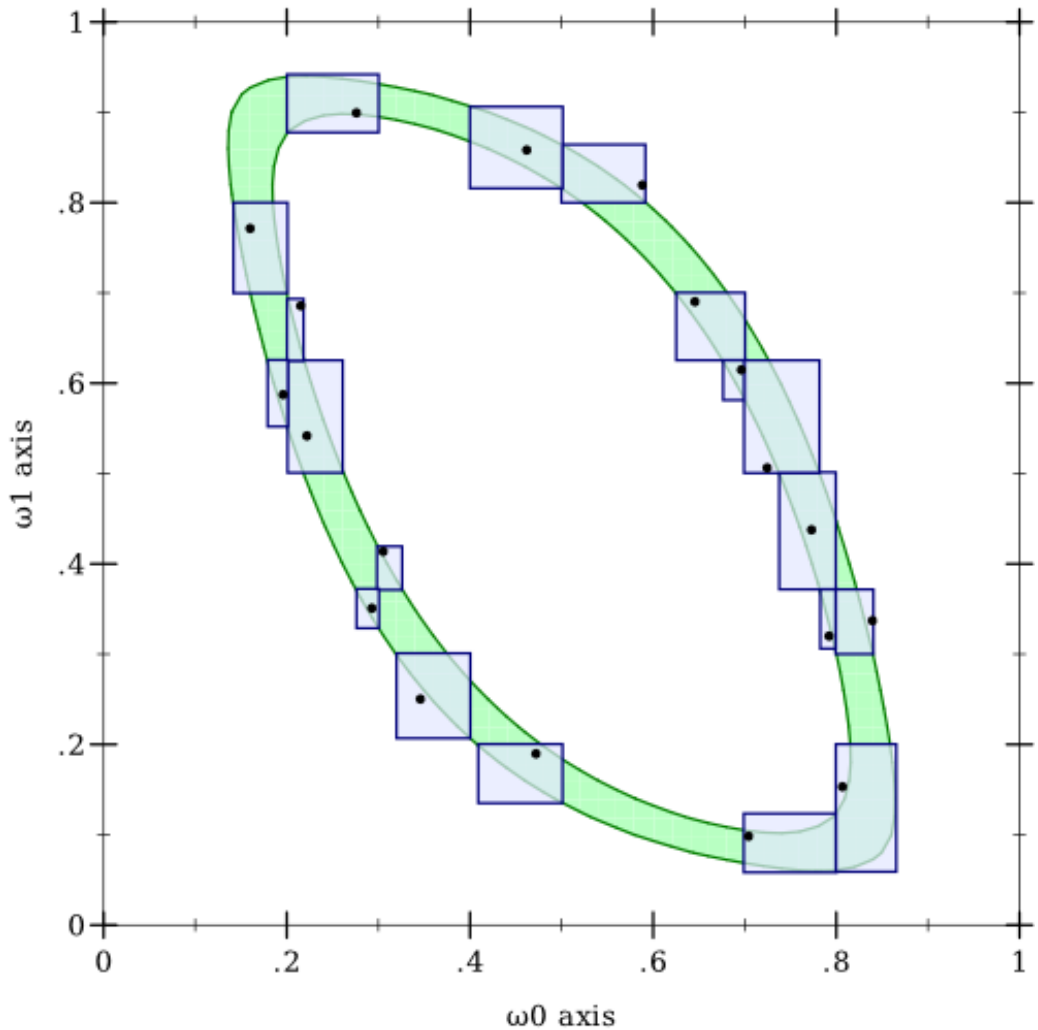
Motivating Soundness To ML



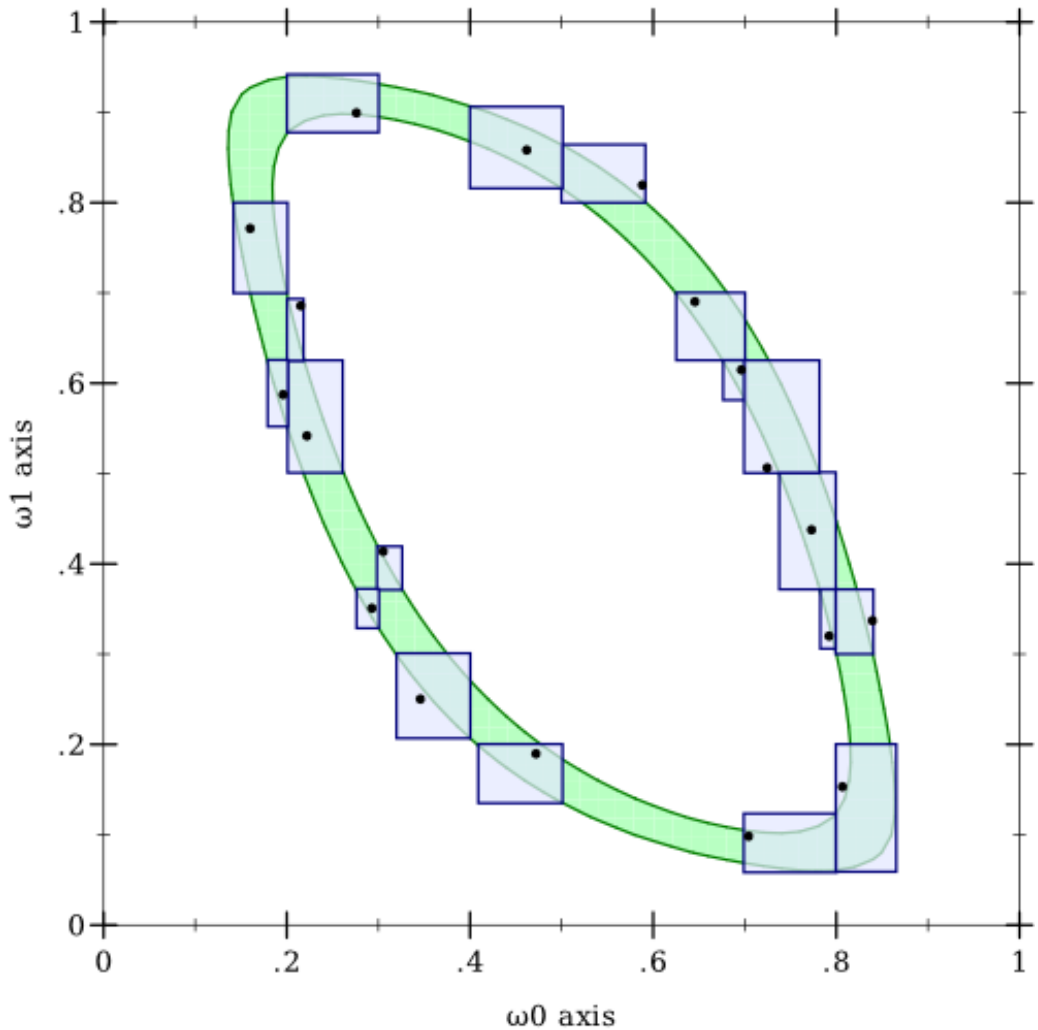
Motivating Soundness To ML



Motivating Soundness To ML



Motivating Soundness To ML



Convergence requires soundness of $[\![\cdot]\!]_{\widehat{\mathrm{pre}}}$

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-jə-dəs/ *n.* preconceived opinion that is not based on reason or actual experience

Culture Gap: Prejudice

pre·ju·dice /ˈpre-jə-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-jə-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·ju·dice /ˈpre-jə-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:

```
(let* ([x   (normal 0 1)]
       [y1  (normal x 1)]
       [y2  (normal x 1)])
  (condition x (and (<= (-  2 ε1) y1 (+  2 ε1))
                    (<= (- -1 ε2) y2 (+ -1 ε2)))))
```

New: Interval Paradox

- DrBayes doesn't allow zero-probability conditions: must use intervals instead of equality
- Standard normal-normal model with two observations:

```
(let* ([x   (normal 0 1)]
       [y1  (normal x 1)]
       [y2  (normal x 1)])
  (condition x (and (<= (-  2 ε1) y1 (+  2 ε1))
                    (<= (- -1 ε2) y2 (+ -1 ε2)))))
```

- `ε1` and `ε2` control widths of intervals

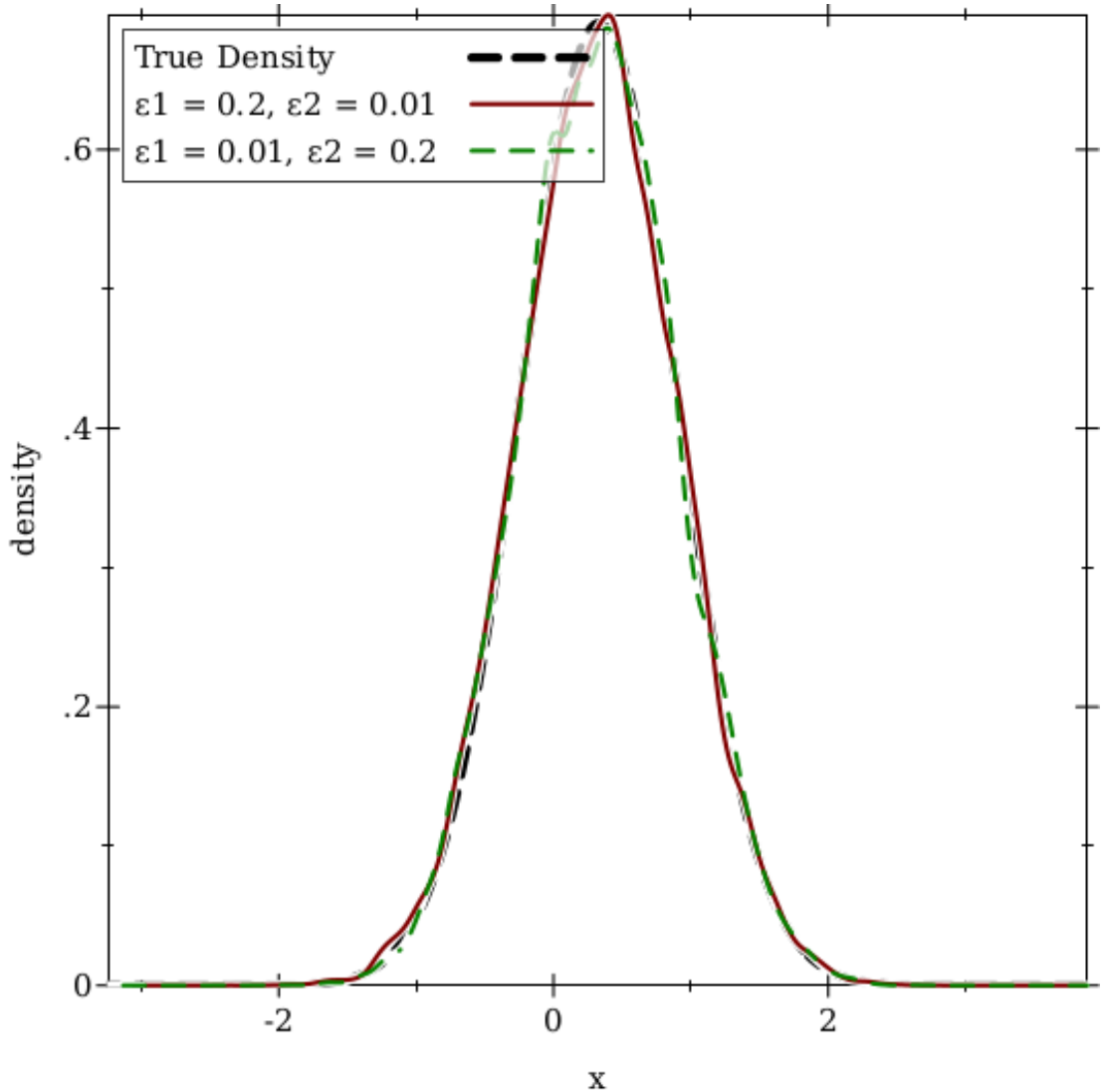
New: Interval Paradox

- DrBayes doesn't allow zero-probability conditions: must use intervals instead of equality
- Standard normal-normal model with two observations:

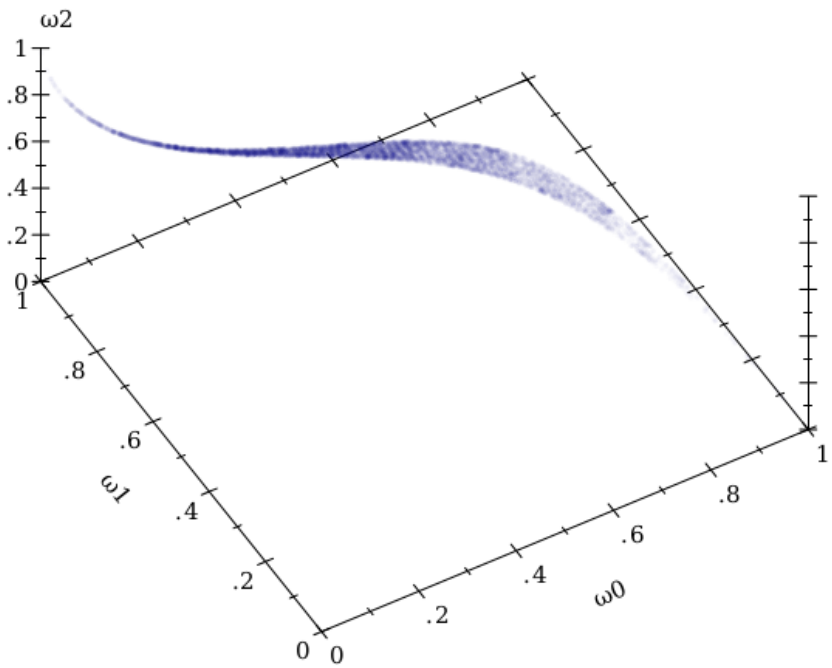
```
(let* ([x   (normal 0 1)]
       [y1  (normal x 1)]
       [y2  (normal x 1)])
  (condition x (and (<= (-  2 ε1) y1 (+  2 ε1))
                    (<= (- -1 ε2) y2 (+ -1 ε2)))))
```

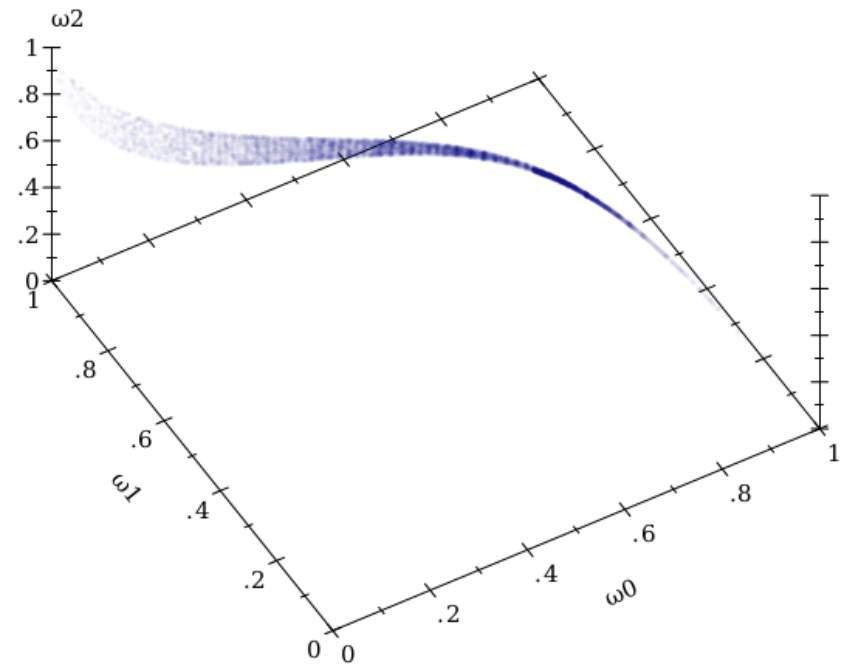
- `ε1` and `ε2` control widths of intervals
- Seems like width or at least proportionality should matter...

Interval Paradox

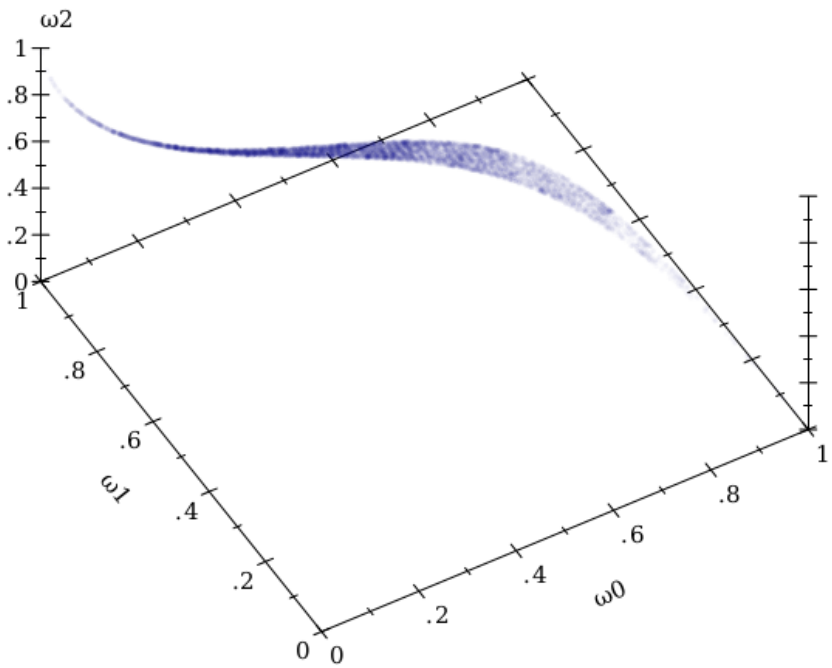


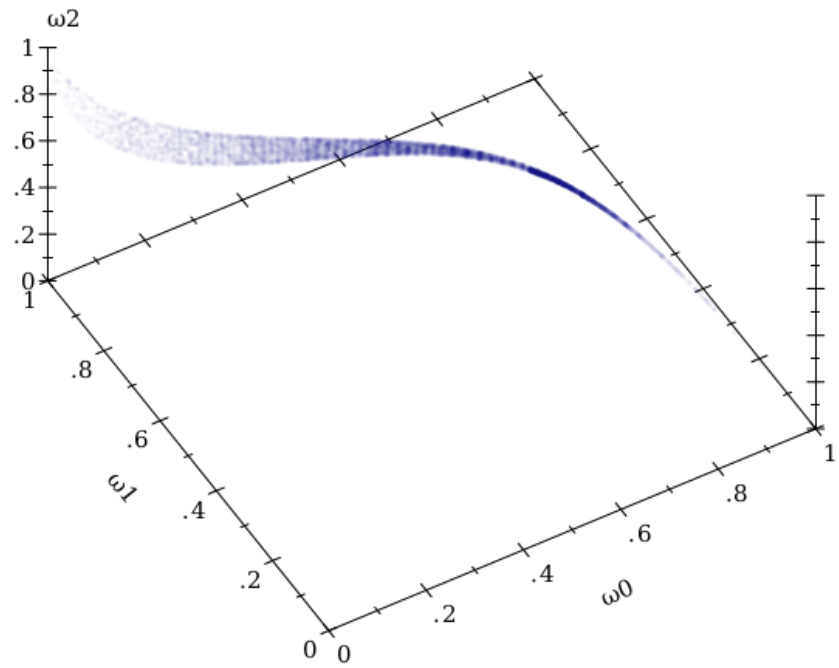
Interval Paradox





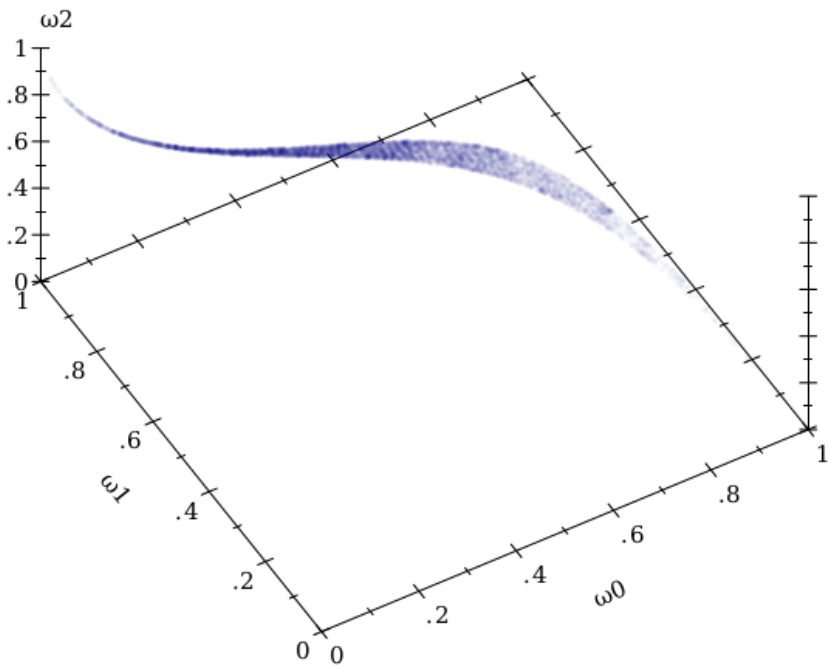
Interval Paradox

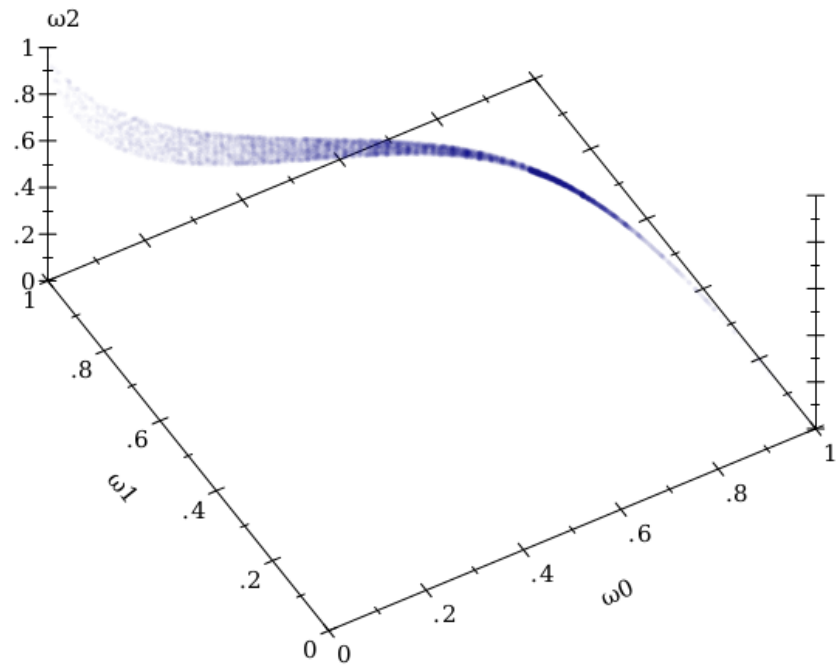




- Width matters a little, but proportionality doesn't

Interval Paradox





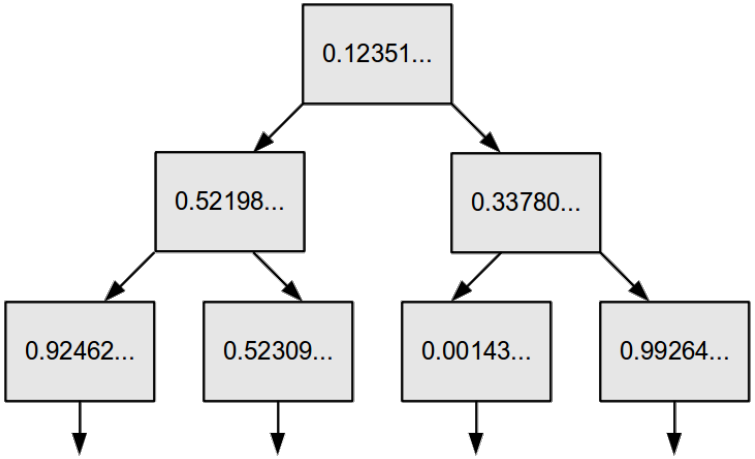
- Width matters a little, but proportionality doesn't
- General property due to Lebesgue differentiation theorem

Technical Gap: Algorithms

- Probabilistic program distributions are weird

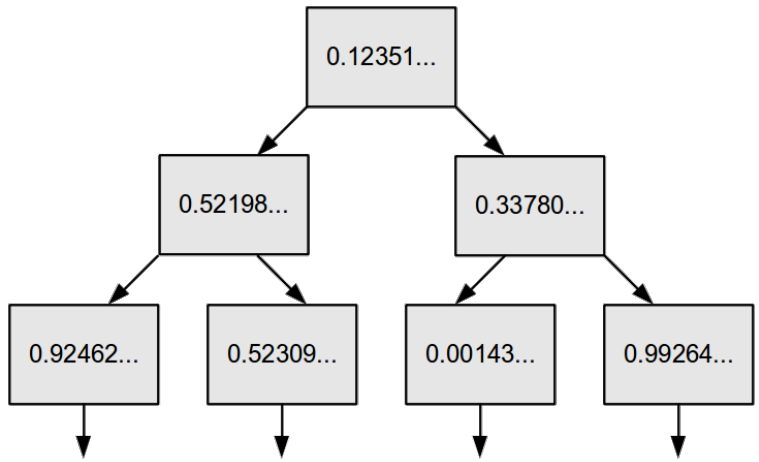
Technical Gap: Algorithms

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



Technical Gap: Algorithms

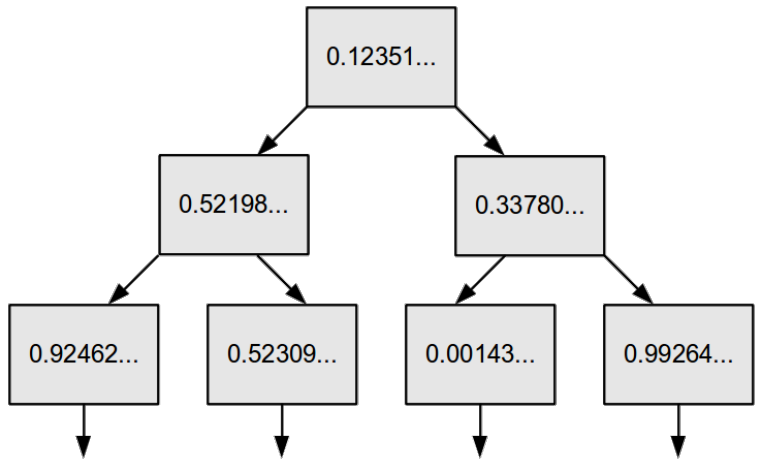
- Probabilistic program distributions are weird
  - One point on support of program domain distribution:



- 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 opportunites for foundational research