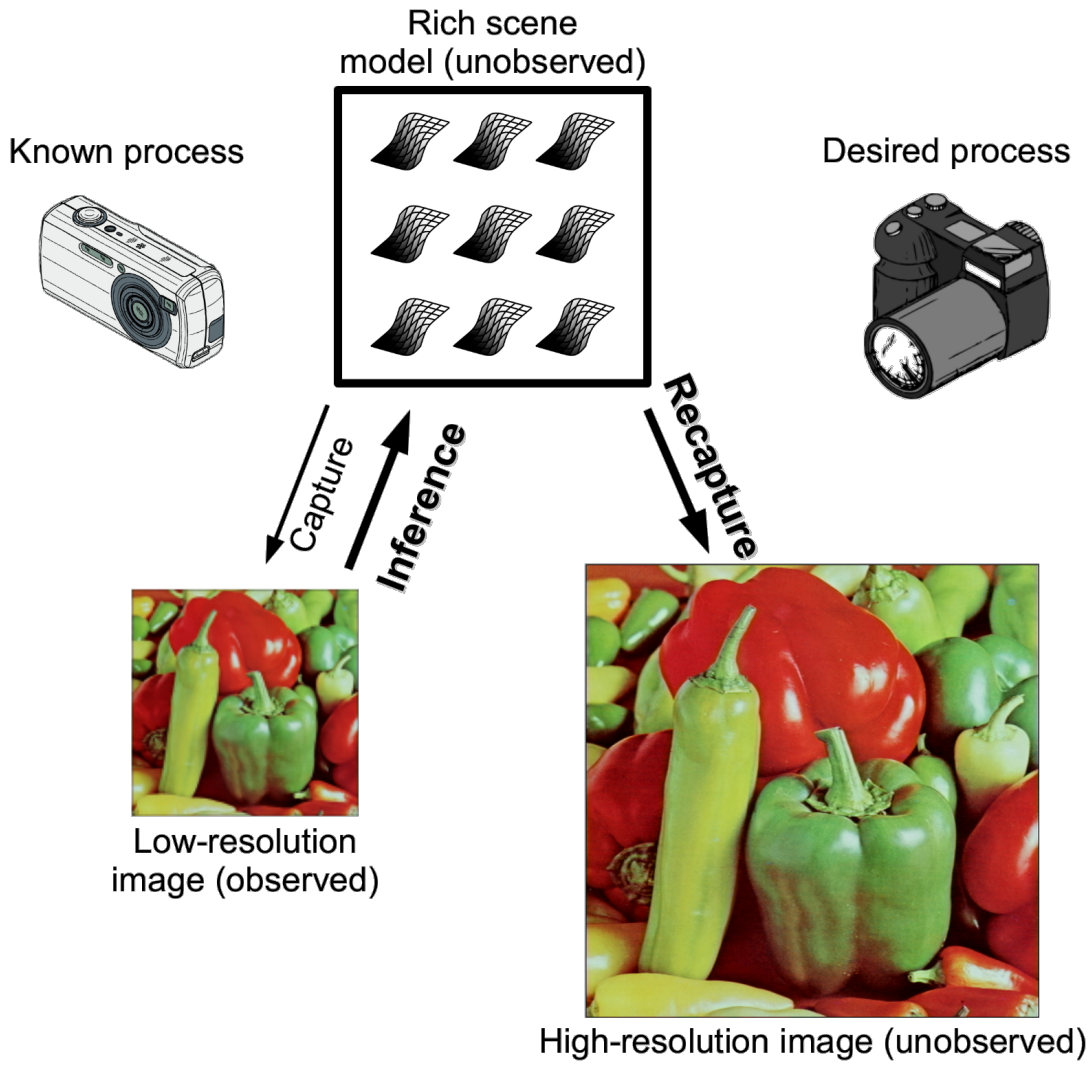
Running Probabilistic Programs Backward

Neil Toronto

PLT @ Brigham Young University

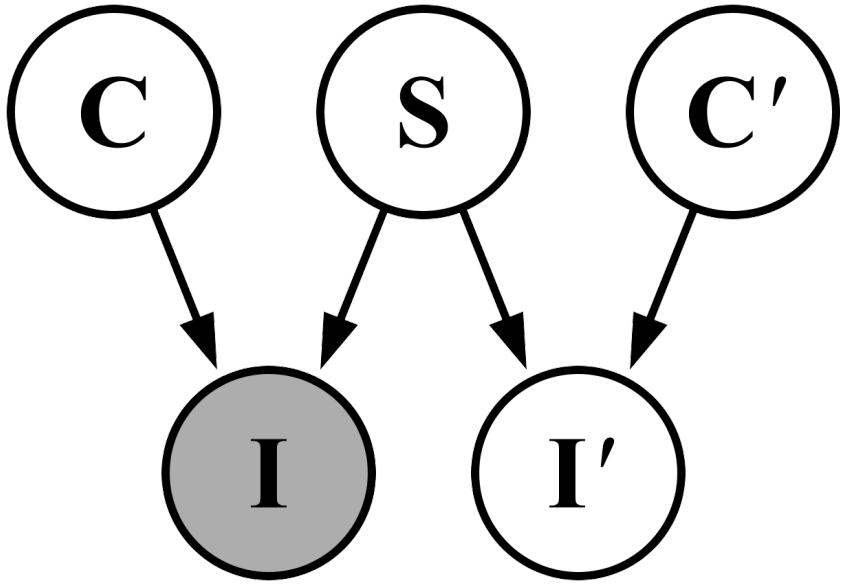
Master's Research: Super-Resolution



Toronto et al. Super-Resolution via Recapture and Bayesian Effect Modeling. CVPR 2009

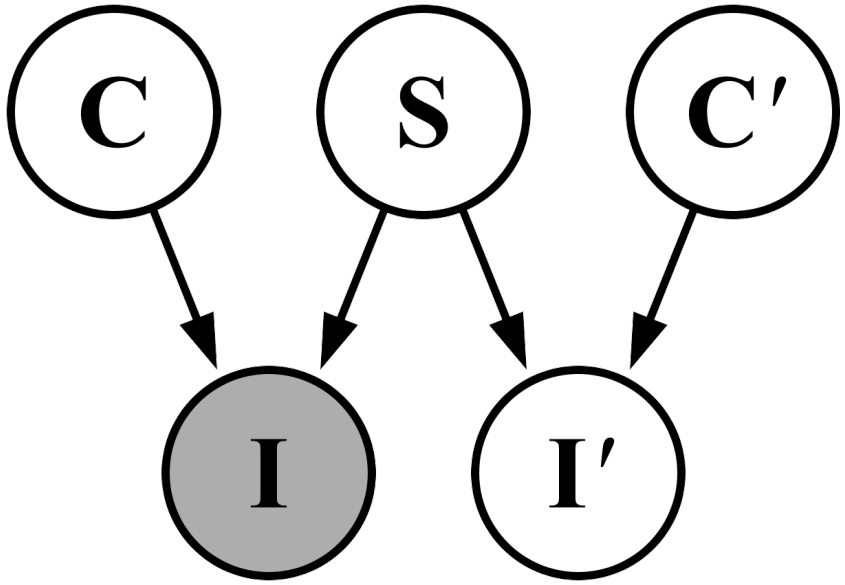
Master's Research: Super-Resolution

- Bayesian Edge Inference (BEI) graphical model:



Master's Research: Super-Resolution

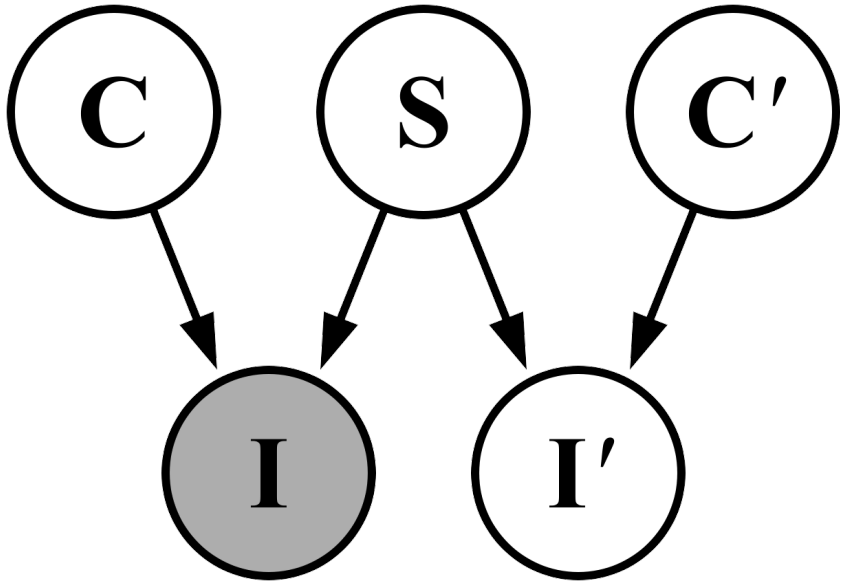
- Bayesian Edge Inference (BEI) graphical model:



- Objective: Obtain one sample from $I' \mid I$

Master's Research: Super-Resolution

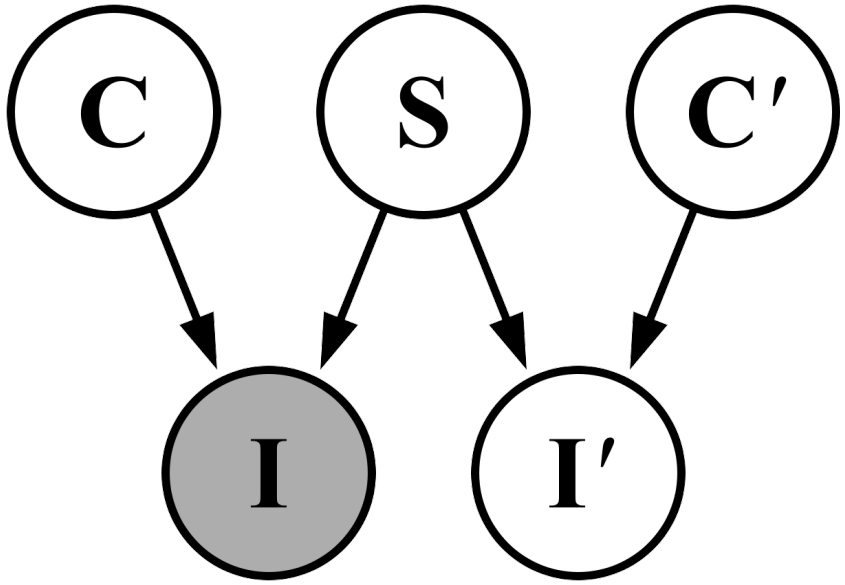
- Bayesian Edge Inference (BEI) graphical model:



- Objective: Obtain one sample from $I' \mid I$
- Model and query: Half a page of beautiful math

Master's Research: Super-Resolution

- Bayesian Edge Inference (BEI) graphical model:



- Objective: Obtain one sample from $I' \mid I$
- Model and query: Half a page of beautiful math
- Query implementation: 600 lines of evil Python

Main Results: Super-Resolution

Resolution Synthesis

Main Results: Super-Resolution

Resolution Synthesis

Local Correlation

Main Results: Super-Resolution

Resolution Synthesis

Local Correlation

Bayesian Edge Inference

Main Results: Super-Resolution

Resolution Synthesis

Local Correlation

Bayesian Edge Inference

- Also beat state-of-the-art on “objective” measures

Pleasantly Surprising Results: Inpainting

- Dr. Morse: “Hey, doesn’t Bayesian inference handle missing data easily?”

Pleasantly Surprising Results: Inpainting

- Dr. Morse: “Hey, doesn’t Bayesian inference handle missing data easily?”
- Me: Yep. Gimme 20 minutes...

Pleasantly Surprising Results: Inpainting

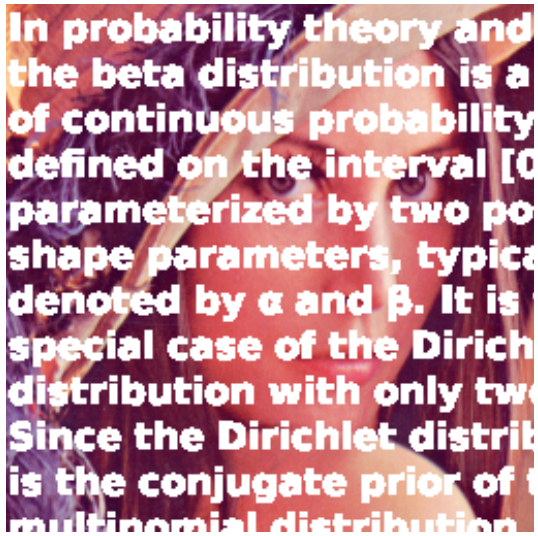
- Dr. Morse: “Hey, doesn’t Bayesian inference handle missing data easily?”
- Me: Yep. Gimme 20 minutes...

Original Image

Pleasantly Surprising Results: Inpainting

- Dr. Morse: “Hey, doesn’t Bayesian inference handle missing data easily?”
- Me: Yep. Gimme 20 minutes...

Original Image

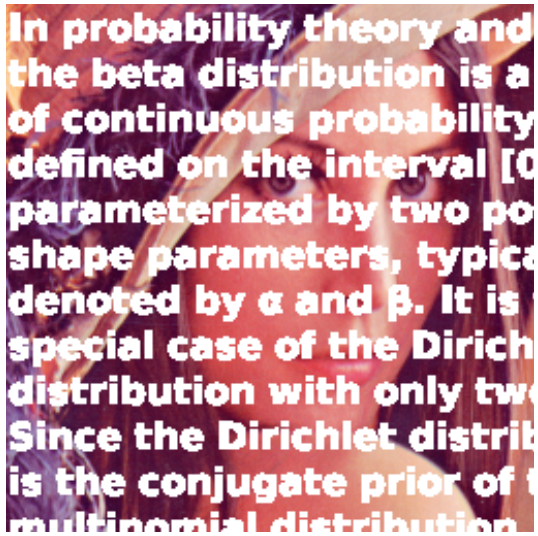


33% Defaced

Pleasantly Surprising Results: Inpainting

- Dr. Morse: “Hey, doesn’t Bayesian inference handle missing data easily?”
- Me: Yep. Gimme 20 minutes...

Original Image

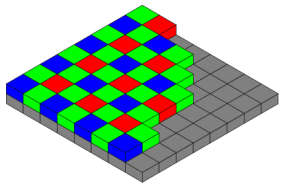


33% Defaced

Bayesian Edge Inference

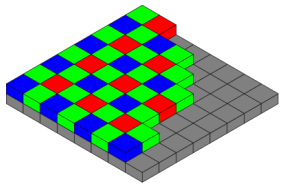
Pleasantly Surprising Results: CCD Demosaicing

- Dr. Morse: “You know, you can think of the result of Bayer filtering on a consumer camera as missing data.”

Bayer filter: 

Pleasantly Surprising Results: CCD Demosaicing

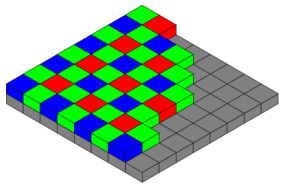
- Dr. Morse: “You know, you can think of the result of Bayer filtering on a consumer camera as missing data.”

Bayer filter: 

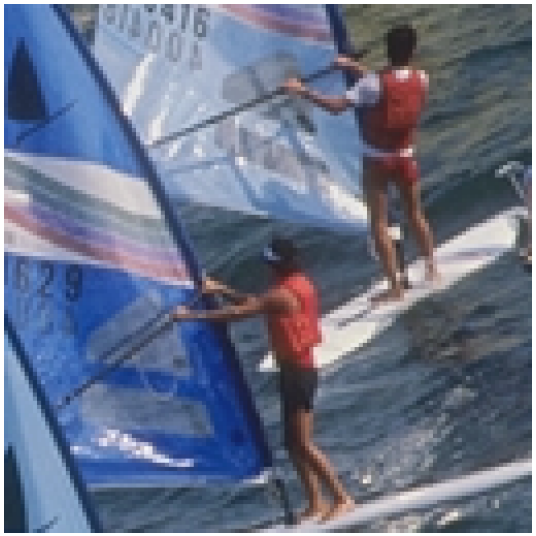
- Me: Huh. Gimme 20 minutes...

Pleasantly Surprising Results: CCD Demosaicing

- Dr. Morse: “You know, you can think of the result of Bayer filtering on a consumer camera as missing data.”

Bayer filter: 

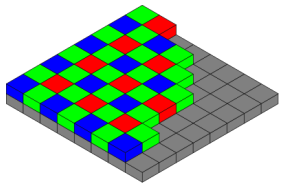
- Me: Huh. Gimme 20 minutes...



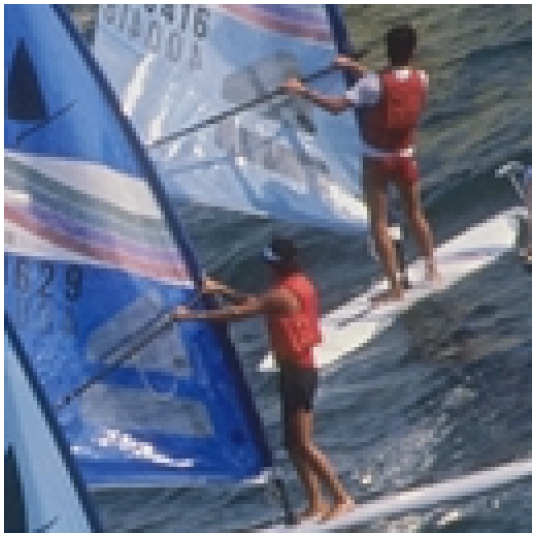
Original Image

Pleasantly Surprising Results: CCD Demosaicing

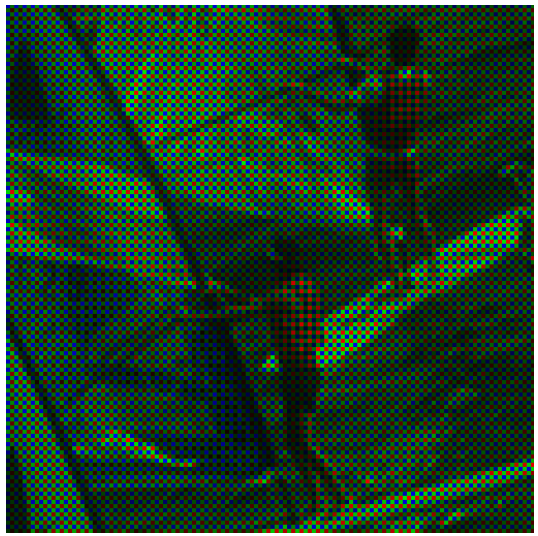
- Dr. Morse: “You know, you can think of the result of Bayer filtering on a consumer camera as missing data.”

Bayer filter: 

- Me: Huh. Gimme 20 minutes...



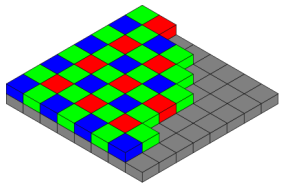
Original Image



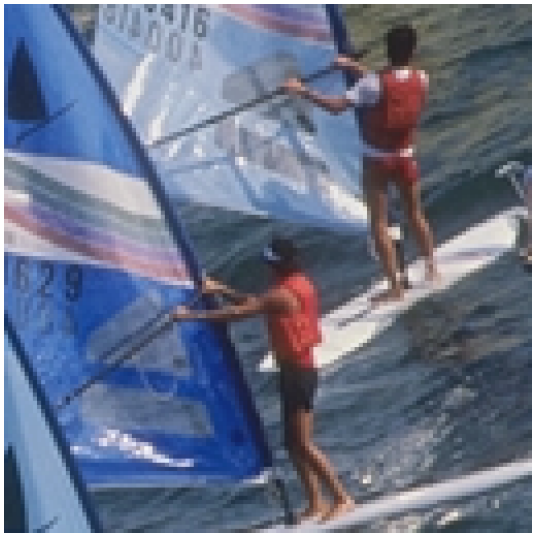
Bayer Filtered

Pleasantly Surprising Results: CCD Demosaicing

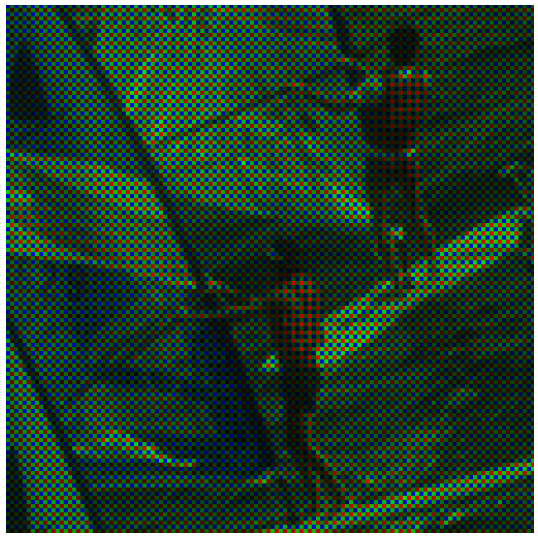
- Dr. Morse: “You know, you can think of the result of Bayer filtering on a consumer camera as missing data.”

Bayer filter: 

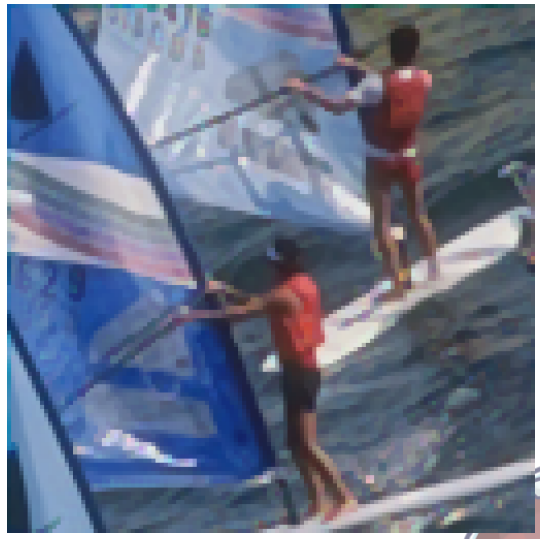
- Me: Huh. Gimme 20 minutes...



Original Image



Bayer Filtered



Bayesian Edge Inference

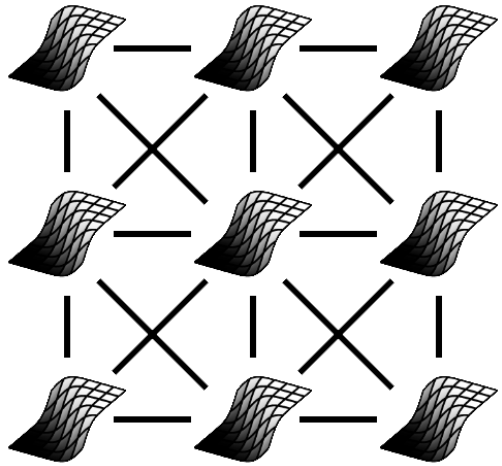
Only Mostly Satisfying

Reason 1: Still not sure the program is right

Only Mostly Satisfying

Reason 1: Still not sure the program is right

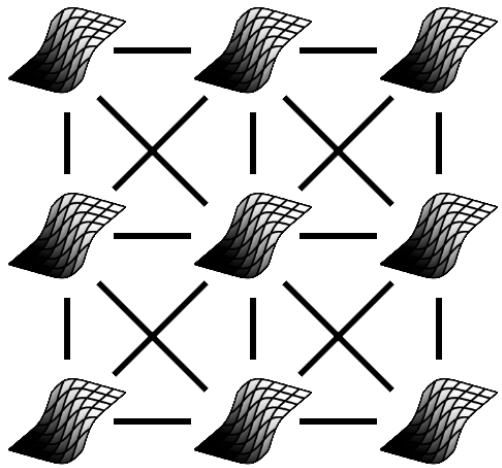
Scene: Markov random field of overlapping facets:



Only Mostly Satisfying

Reason 1: Still not sure the program is right

Scene: Markov random field of overlapping facets:



Reason 2: “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*.”

Simple Example Theory

Simple Example Theory

- Example theory: Normal-Normal

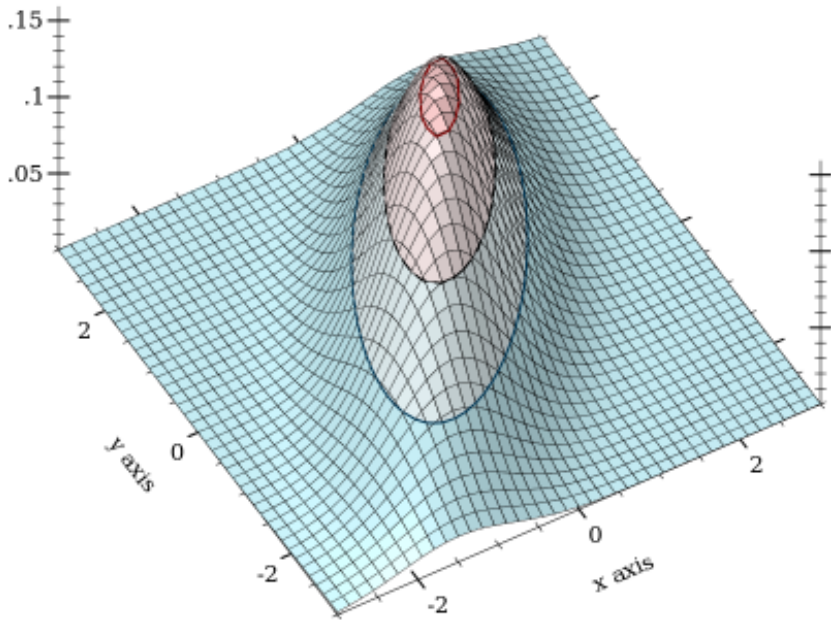
$$X \sim \text{Normal}(0, 1)$$
$$Y \sim \text{Normal}(X, 1)$$

Simple Example Theory

- Example theory: Normal-Normal

$$X \sim \text{Normal}(0, 1)$$
$$Y \sim \text{Normal}(X, 1)$$

- Density model $f : \mathbb{R} \times \mathbb{R} \to [0, \infty)$:



An Observation Problem

- Find the distribution of $X, Y \mid X + Y = 1$

An Observation Problem

- Find the distribution of $X, Y | X + Y = 1$

MIGHT BE TOO EASY

An Observation Problem

- Find the distribution of $X, Y \mid X + Y = 1$

MIGHT BE TOO EASY

- Find the distribution of $X, Y \mid \sqrt{X^2 + Y^2} = 1$

An Observation Problem

- Find the distribution of $X, Y \mid X^2 + Y^2 = 1$

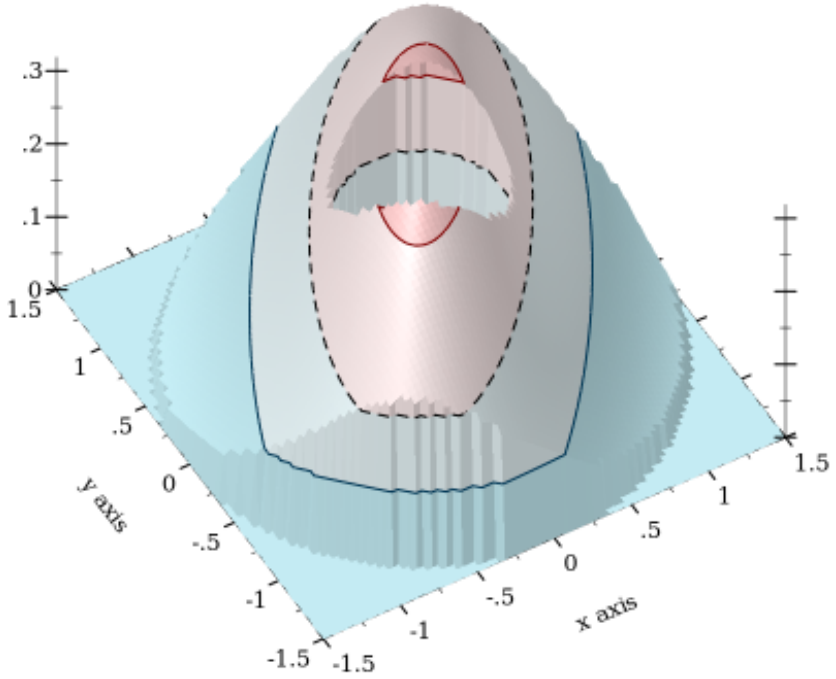
MIGHT BE TOO EASY

- Find the distribution of $X, Y \mid \sqrt{X^2 + Y^2} = 1$
- Maybe take a limit of $\left|\sqrt{X^2 + Y^2} - 1\right| < \epsilon$?

An Observation Problem

- ~~Find the distribution of $X, Y \mid X = 1$~~ MIGHT BE TOO EASY
- Find the distribution of $X, Y \mid \sqrt{X^2 + Y^2} = 1$
- Maybe take a limit of $\left|\sqrt{X^2 + Y^2} - 1\right| < \epsilon$?

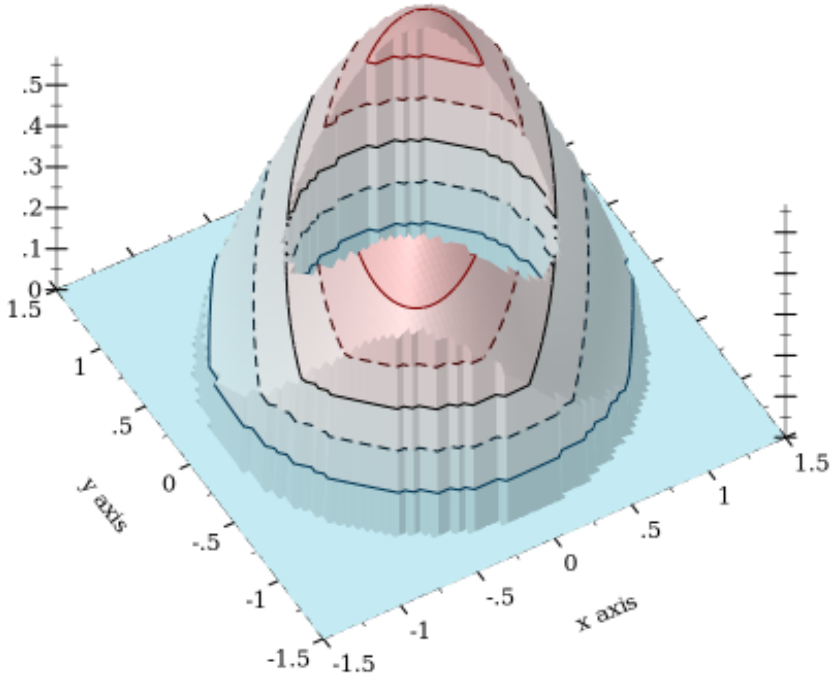
Conditional density with $\epsilon = 0.5$:



An Observation Problem

- ~~Find the distribution of $X, Y \mid X = 1$~~ MIGHT BE TOO EASY
- Find the distribution of $X, Y \mid \sqrt{X^2 + Y^2} = 1$
- Maybe take a limit of $\left|\sqrt{X^2 + Y^2} - 1\right| < \epsilon$?

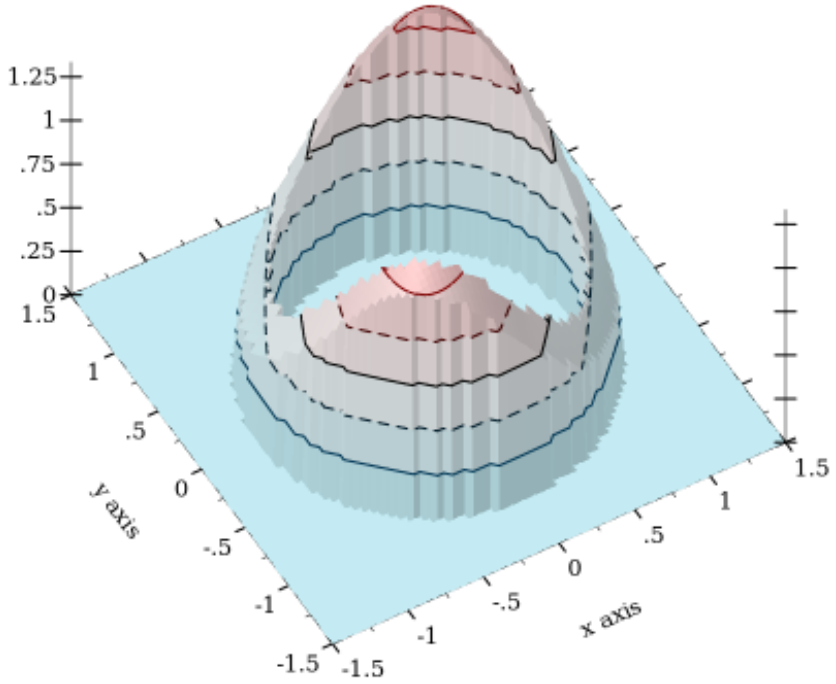
Conditional density with $\epsilon = 0.25$:



An Observation Problem

- ~~Find the distribution of $X, Y \mid X = 1$~~ MIGHT BE TOO EASY
- Find the distribution of $X, Y \mid \sqrt{X^2 + Y^2} = 1$
- Maybe take a limit of $\left|\sqrt{X^2 + Y^2} - 1\right| < \epsilon$?

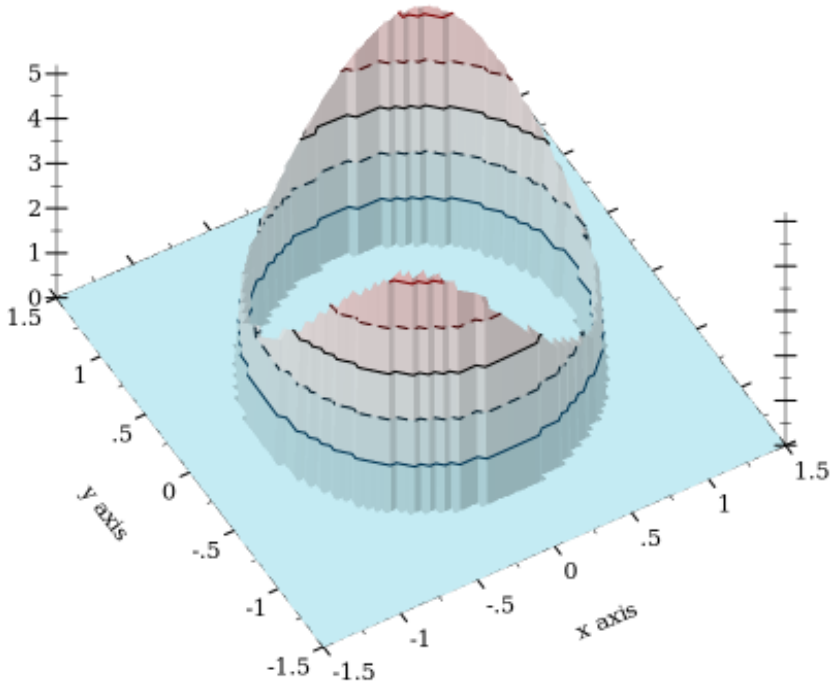
Conditional density with $\epsilon = 0.1$:



An Observation Problem

- ~~Find the distribution of $X, Y \mid X = 1$~~ MIGHT BE TOO EASY
- Find the distribution of $X, Y \mid \sqrt{X^2 + Y^2} = 1$
- Maybe take a limit of $\left|\sqrt{X^2 + Y^2} - 1\right| < \epsilon$?

Conditional density with $\epsilon = 0.025$:



What Can't Densities Model?

- Anything that puts nonzero probability on a zero-volume domain subset

What Can't Densities Model?

- Anything that puts nonzero probability on a zero-volume domain subset
  - Non-axis-aligned, zero-probability conditions (can't reduce dimension)

What Can't Densities Model?

- Anything that puts nonzero probability on a zero-volume domain subset
  - Non-axis-aligned, zero-probability conditions (can't reduce dimension)
  - CDFs with discontinuities

What Can't Densities Model?

- Anything that puts nonzero probability on a zero-volume domain subset
  - Non-axis-aligned, zero-probability conditions (can't reduce dimension)
  - CDFs with discontinuities
  - Discontinuous change of variable

What Can't Densities Model?

- Anything that puts nonzero probability on a zero-volume domain subset
  - Non-axis-aligned, zero-probability conditions (can't reduce dimension)
  - CDFs with discontinuities
  - Discontinuous change of variable
- Distributions with variable-dimension support

What Can't Densities Model?

- Anything that puts nonzero probability on a zero-volume domain subset
  - Non-axis-aligned, zero-probability conditions (can't reduce dimension)
  - CDFs with discontinuities
  - Discontinuous change of variable
- Distributions with variable-dimension support
- Nontrivial distributions on infinite products like $[0, 1]^{\mathbb{N}}$

What Can't Densities Model?

- Anything that puts nonzero probability on a zero-volume domain subset
  - Non-axis-aligned, zero-probability conditions (can't reduce dimension)
  - CDFs with discontinuities
  - Discontinuous change of variable
- Distributions with variable-dimension support
- Nontrivial distributions on infinite products like $[0, 1]^{\mathbb{N}}$

There are tricks to get around limitations, but none are generally applicable...

Measure-Theoretic Probability

- Main ideas:
  - Don’t assign numbers to *changes* in value, assign numbers to *sets* of values

Measure-Theoretic Probability

- Main ideas:
  - Don’t assign numbers to *changes* in value, assign numbers to *sets* of values
  - Confine assumed randomness to one place by making random variables *deterministic functions* that observe a random source

Measure-Theoretic Probability

- Main ideas:
  - Don't assign numbers to *changes* in value, assign numbers to *sets* of values
  - Confine assumed randomness to one place by making random variables *deterministic functions* that observe a random source
- Measure-theoretic model of example theory:

$$\Omega = \mathbb{R} \times \mathbb{R}$$

$$P : \mathrm{Set}(\Omega) \rightharpoonup [0,1],\ P(A) = \int_A f\ d\lambda$$

Measure-Theoretic Probability

- Main ideas:
  - Don't assign numbers to *changes* in value, assign numbers to *sets* of values
  - Confine assumed randomness to one place by making random variables *deterministic functions* that observe a random source
- Measure-theoretic model of example theory:

$$\Omega = \mathbb{R} \times \mathbb{R}$$
$$P : \mathrm{Set}(\Omega) \rightharpoonup [0,1],\ P(A) = \int_A f\ d\lambda$$
$$X : \Omega \to \mathbb{R},\ X(\omega) = \omega_0$$
$$Y : \Omega \to \mathbb{R},\ Y(\omega) = \omega_1$$

Measure-Theoretic Queries

- Specific query:

$$\Pr[X > 1] = P(\{\omega \in \Omega \mid X(\omega) > 1\})$$

Measure-Theoretic Queries

- Specific query:

$$\Pr[X > 1] = P(\{\omega \in \Omega \mid X(\omega) > 1\})$$

- Generalized: For $Z : \Omega \to C$ and $C' \subseteq C$,

$$\Pr[Z \in C'] = P(\{\omega \in \Omega \mid Z(\omega) \in C'\})$$

Measure-Theoretic Queries

- Specific query:

$$\Pr[X > 1] = P(\{\omega \in \Omega \mid X(\omega) > 1\})$$

- Generalized: For $Z : \Omega \to C$ and $C' \subseteq C$,

$$\Pr[Z \in C'] = P(\{\omega \in \Omega \mid Z(\omega) \in C'\})$$

- Conditional query:

$$\Pr[e_1 \mid e_2] = \Pr[e_1, e_2]/\Pr[e_2] \quad \text{if } \Pr[e_2] > 0$$

Measure-Theoretic Queries

- Specific query:

$$\Pr[X > 1] = P(\{\omega \in \Omega \mid X(\omega) > 1\})$$

- Generalized: For $Z : \Omega \to C$ and $C' \subseteq C$,

$$\Pr[Z \in C'] = P(\{\omega \in \Omega \mid Z(\omega) \in C'\})$$

- Conditional query:

$$\Pr[e_1 \mid e_2] = \Pr[e_1, e_2]/\Pr[e_2] \quad \text{if } \Pr[e_2] > 0$$

But $\Pr[\sqrt{X^2 + Y^2} = 1] = 0$...

Zero-Probability Conditions (Axial)

- Observation query:

$$\Pr[X > 1 \mid Y = 2] \;=\; \lim_{\epsilon \to 0} \Pr[X > 1 \mid |Y - 2| < \epsilon]$$

Zero-Probability Conditions (Axial)

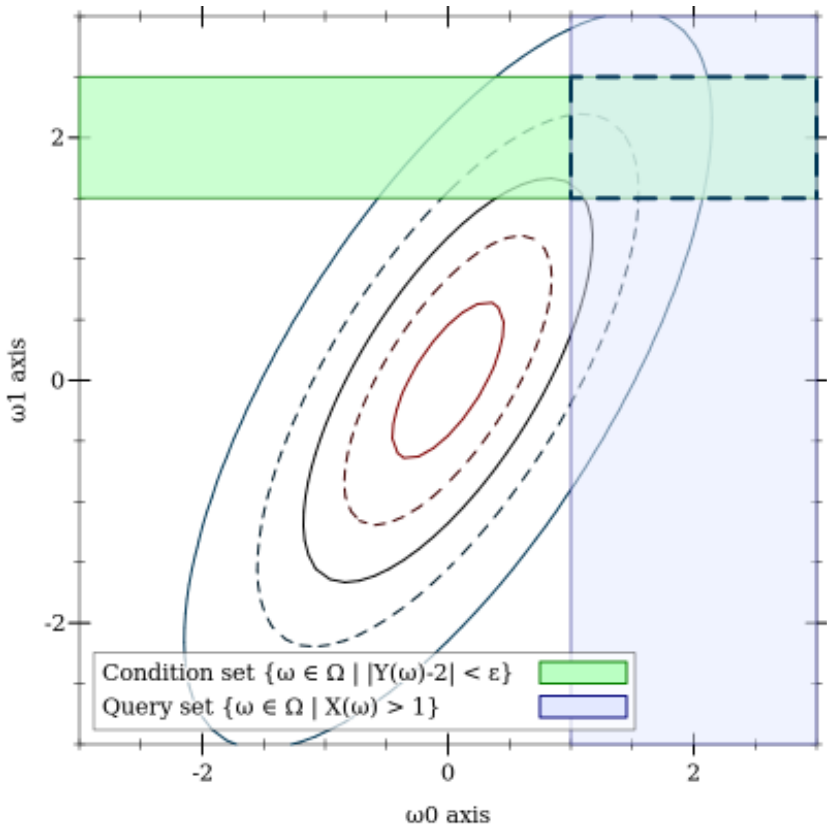
- Observation query:

$$\begin{aligned}\Pr[X > 1 \mid Y = 2] &= \lim_{\epsilon \to 0} \Pr[X > 1 \mid |Y - 2| < \epsilon] \\ &= \lim_{\epsilon \to 0} \frac{\Pr[X > 1, |Y - 2| < \epsilon]}{\Pr[|Y - 2| < \epsilon]}\end{aligned}$$

Zero-Probability Conditions (Axial)

- Observation query:

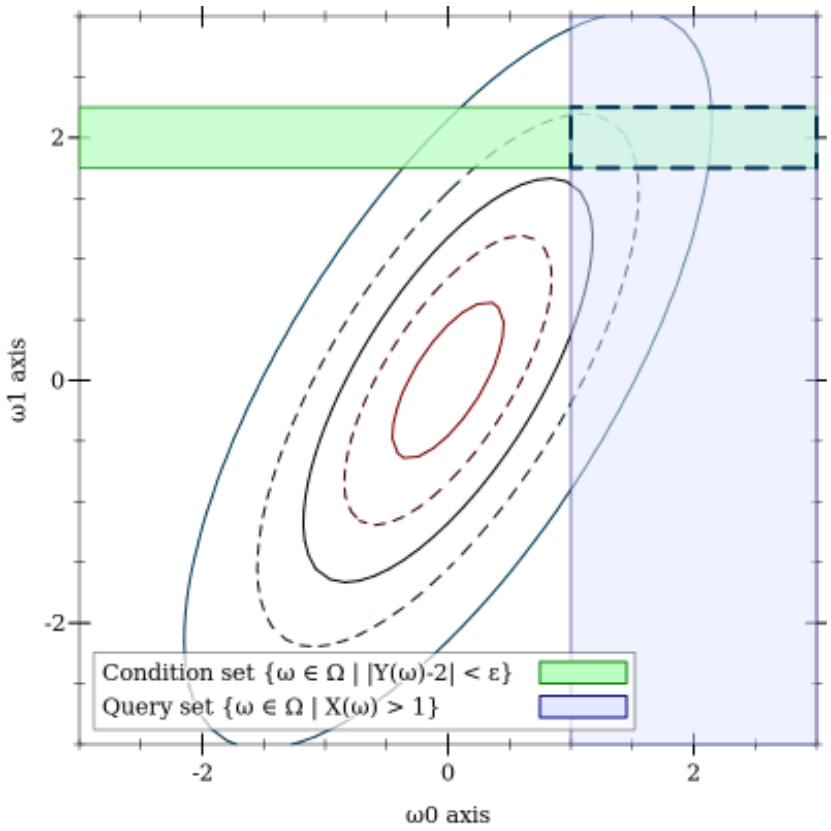
$$\begin{aligned}\Pr[X > 1 \mid Y = 2] &= \lim_{\epsilon \to 0} \Pr[X > 1 \mid |Y - 2| < \epsilon] \\ &= \lim_{\epsilon \to 0} \frac{\Pr[X > 1, |Y - 2| < \epsilon]}{\Pr[|Y - 2| < \epsilon]}\end{aligned}$$



Zero-Probability Conditions (Axial)

- Observation query:

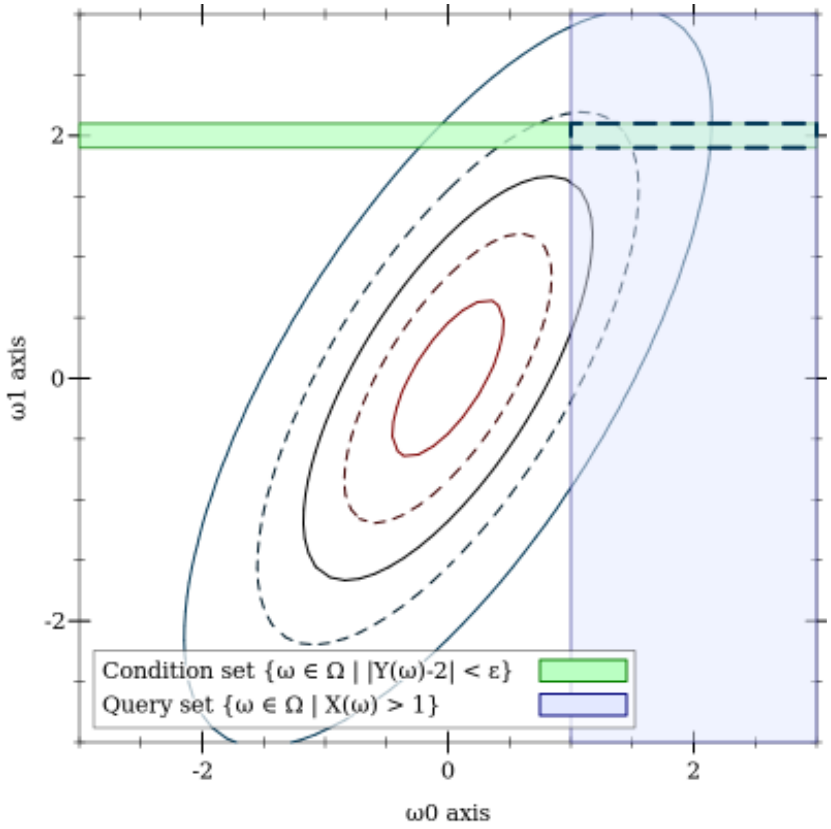
$$\begin{aligned}\Pr[X > 1 \mid Y = 2] &= \lim_{\epsilon \to 0} \Pr[X > 1 \mid |Y - 2| < \epsilon] \\ &= \lim_{\epsilon \to 0} \frac{\Pr[X > 1, |Y - 2| < \epsilon]}{\Pr[|Y - 2| < \epsilon]}\end{aligned}$$



Zero-Probability Conditions (Axial)

- Observation query:

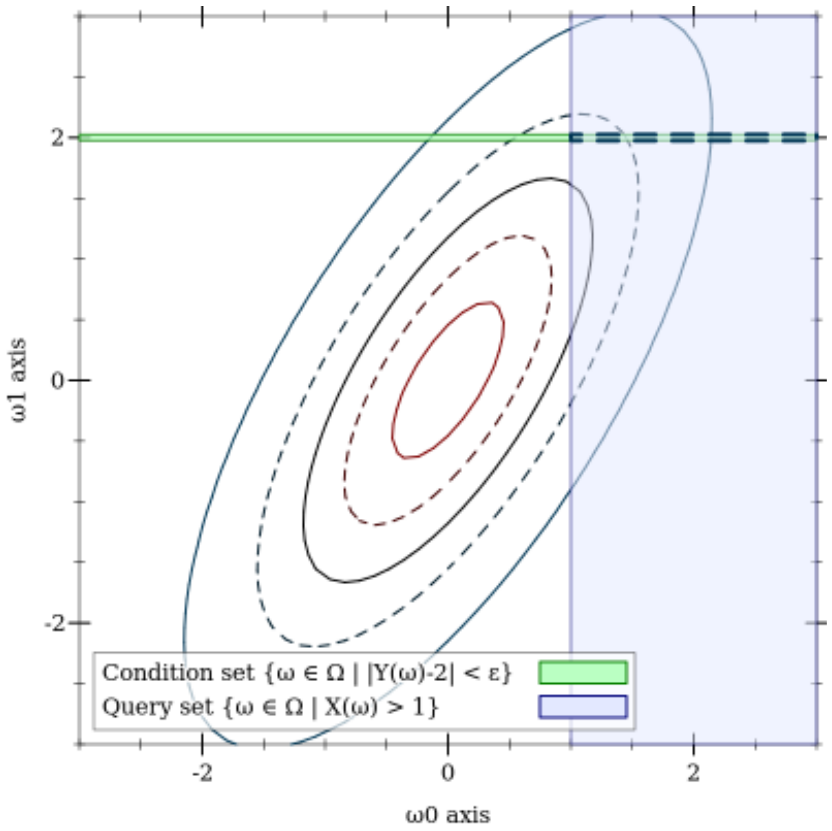
$$\begin{aligned}\Pr[X > 1 \mid Y = 2] &= \lim_{\epsilon \to 0} \Pr[X > 1 \mid |Y - 2| < \epsilon] \\ &= \lim_{\epsilon \to 0} \frac{\Pr[X > 1, |Y - 2| < \epsilon]}{\Pr[|Y - 2| < \epsilon]}\end{aligned}$$



Zero-Probability Conditions (Axial)

- Observation query:

$$\begin{aligned}\Pr[X > 1 \mid Y = 2] &= \lim_{\epsilon \to 0} \Pr[X > 1 \mid |Y - 2| < \epsilon] \\ &= \lim_{\epsilon \to 0} \frac{\Pr[X > 1, |Y - 2| < \epsilon]}{\Pr[|Y - 2| < \epsilon]}\end{aligned}$$



Zero-Probability Conditions (Circular)

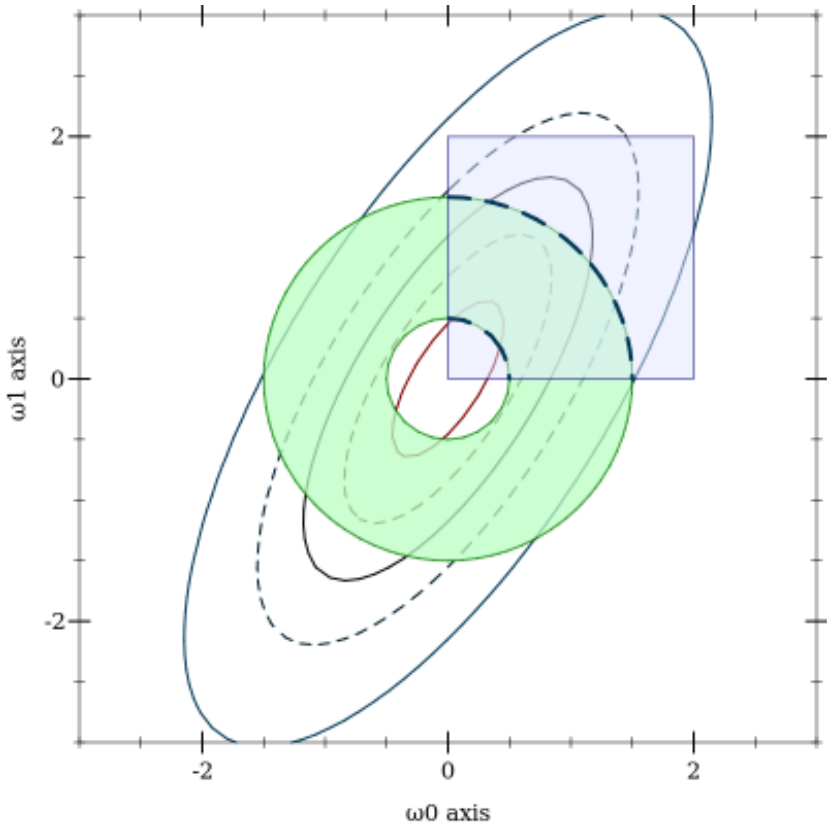
- Can differentiate w.r.t. any random variable:

$$\Pr[e \mid \sqrt{X^2 + Y^2} = 1] \;=\; \lim_{\epsilon \to 0} \Pr[e \mid |\sqrt{X^2 + Y^2} - 1| < \epsilon]$$

Zero-Probability Conditions (Circular)

- Can differentiate w.r.t. any random variable:

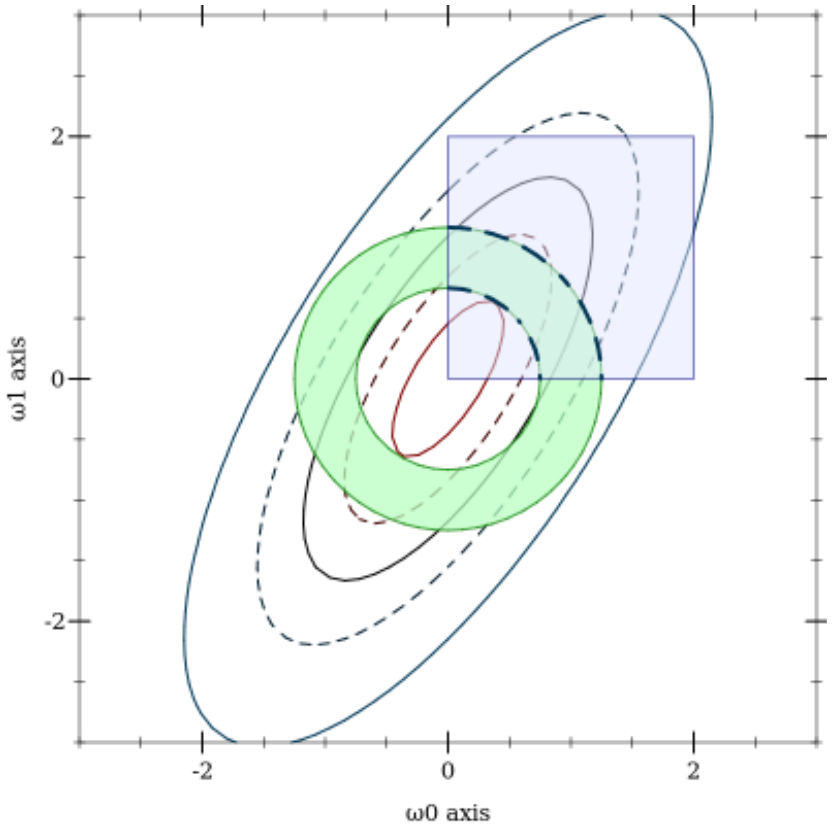
$$\Pr[e \mid \sqrt{X^2 + Y^2} = 1] = \lim_{\epsilon \to 0} \Pr[e \mid |\sqrt{X^2 + Y^2} - 1| < \epsilon]$$



Zero-Probability Conditions (Circular)

- Can differentiate w.r.t. any random variable:

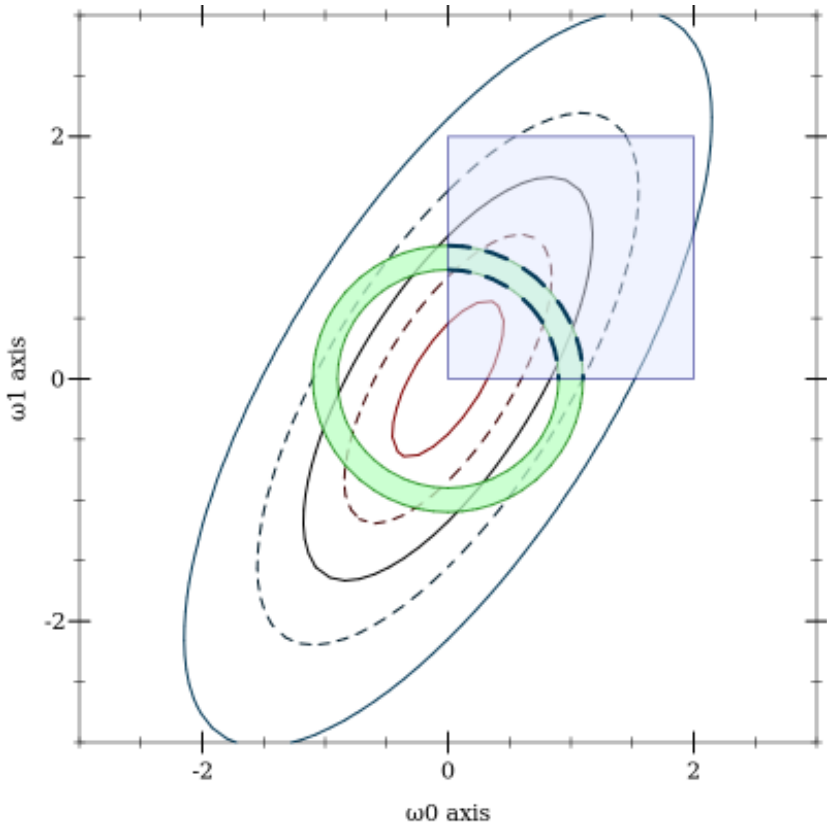
$$\Pr[e \mid \sqrt{X^2 + Y^2} = 1] \;=\; \lim_{\epsilon \to 0} \Pr[e \mid |\sqrt{X^2 + Y^2} - 1| < \epsilon]$$



Zero-Probability Conditions (Circular)

- Can differentiate w.r.t. any random variable:

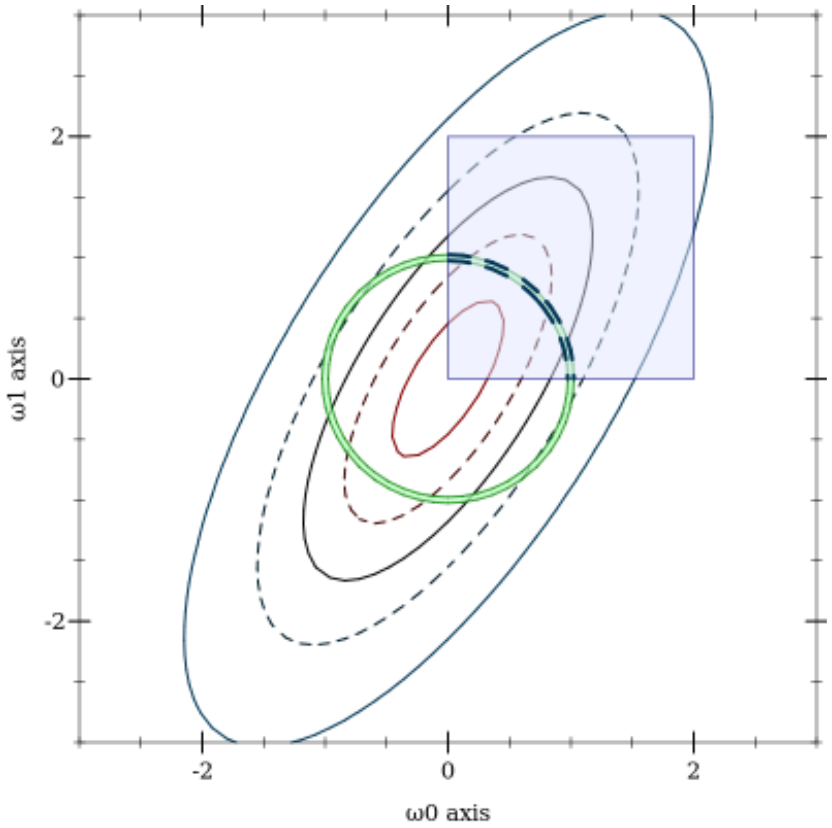
$$\Pr[e \mid \sqrt{X^2 + Y^2} = 1] \;=\; \lim_{\epsilon \to 0} \Pr[e \mid |\sqrt{X^2 + Y^2} - 1| < \epsilon]$$



Zero-Probability Conditions (Circular)

- Can differentiate w.r.t. any random variable:

$$\Pr[e \mid \sqrt{X^2 + Y^2} = 1] \;=\; \lim_{\epsilon \to 0} \Pr[e \mid |\sqrt{X^2 + Y^2} - 1| < \epsilon]$$



Measure-Theoretic Models

- Random variables and $\Pr[\cdot]$ are an abstraction boundary that hides Ω and P

Measure-Theoretic Models

- Random variables and $\Pr[\cdot]$ are an abstraction boundary that hides Ω and P
- Queries return the same answers for any model whose random variables' outputs have the correct joint distribution

Measure-Theoretic Models

- Random variables and $\Pr[\cdot]$ are an abstraction boundary that hides Ω and P
- Queries return the same answers for any model whose random variables' outputs have the correct joint distribution

Let $F : \mathbb{R} \to [0, 1]$ be the Normal CDF, and define a **uniform random source** model:

$$\Omega = [0, 1] \times [0, 1]$$
$$P : \mathrm{Set}(\Omega) \rightharpoonup [0, 1], \;\; P(A) = \lambda(A) \;\; (\text{i.e. } A\text{'s area})$$

Measure-Theoretic Models

- Random variables and $\Pr[\cdot]$ are an abstraction boundary that hides Ω and P
- Queries return the same answers for any model whose random variables' outputs have the correct joint distribution

Let $F : \mathbb{R} \to [0, 1]$ be the Normal CDF, and define a **uniform random source** model:

$$\Omega = [0, 1] \times [0, 1]$$
$$P : \mathrm{Set}(\Omega) \rightharpoonup [0, 1],\ P(A) = \lambda(A) \quad (\text{i.e. } A\text{'s area})$$
$$X : \Omega \to \mathbb{R},\ X(\omega) = F^{-1}(\omega_0)$$
$$Y : \Omega \to \mathbb{R},\ Y(\omega) = F^{-1}(\omega_1) + X(\omega)$$

Now We Finally Get to Running Things Backward

- Generalized query:

$$\Pr[Z \in C'] = P(\{\omega \in \Omega \mid Z(\omega) \in C'\})$$

Now We Finally Get to Running Things Backward

- Generalized query:

$$\begin{aligned}\Pr[Z \in C'] &= P(\{\omega \in \Omega \mid Z(\omega) \in C'\}) \\ &= P(Z^{-1}(C'))\end{aligned}$$

i.e. output distributions are defined by **preimages**

Now We Finally Get to Running Things Backward

- Generalized query:

$$\begin{aligned}\Pr[Z \in C'] &= P(\{\omega \in \Omega \mid Z(\omega) \in C'\}) \\ &= P(Z^{-1}(C'))\end{aligned}$$

i.e. output distributions are defined by **preimages**

- For a uniform random source model,
  - Compute output probabilities by computing **preimage** areas

Now We Finally Get to Running Things Backward

- Generalized query:

$$\begin{aligned}\Pr[Z \in C'] &= P(\{\omega \in \Omega \mid Z(\omega) \in C'\}) \\ &= P(Z^{-1}(C'))\end{aligned}$$

i.e. output distributions are defined by **preimages**

- For a uniform random source model,
  - Compute output probabilities by computing **preimage** areas
  - Compute conditional probabilities as quotients of **preimage** areas

Now We Finally Get to Running Things Backward

- Generalized query:

$$\begin{aligned}\Pr[Z \in C'] &= P(\{\omega \in \Omega \mid Z(\omega) \in C'\}) \\ &= P(Z^{-1}(C'))\end{aligned}$$

i.e. output distributions are defined by **preimages**

- For a uniform random source model,
  - Compute output probabilities by computing **preimage** areas
  - Compute conditional probabilities as quotients of **preimage** areas
  - Sample conditional distributions by sampling uniformly from **preimages** of conditions

Is Your Crazy Idea Even Feasible?

Is Your Crazy Idea Even Feasible?

- Maybe we can find out by looking at preimages

Is Your Crazy Idea Even Feasible?

- Maybe we can find out by looking at preimages
- Normal-Normal random variables in uniform source model:

$$X : \Omega \to \mathbb{R}, \;\; X(\omega) = F^{-1}(\omega_0)$$
$$Y : \Omega \to \mathbb{R}, \;\; Y(\omega) = F^{-1}(\omega_1) + X(\omega)$$

Is Your Crazy Idea Even Feasible?

- Maybe we can find out by looking at preimages
- Normal-Normal random variables in uniform source model:

$$X : \Omega \to \mathbb{R},\ X(\omega) = F^{-1}(\omega_0)$$
$$Y : \Omega \to \mathbb{R},\ Y(\omega) = F^{-1}(\omega_1) + X(\omega)$$

- Random variable for the unit circle condition:

$$Z : \Omega \to \mathbb{R},\ Z(\omega) = |\sqrt{X(\omega)^2 + Y(\omega)^2} - 1|$$

Is Your Crazy Idea Even Feasible?

- Maybe we can find out by looking at preimages
- Normal-Normal random variables in uniform source model:

$$X : \Omega \to \mathbb{R},\ X(\omega) = F^{-1}(\omega_0)$$
$$Y : \Omega \to \mathbb{R},\ Y(\omega) = F^{-1}(\omega_1) + X(\omega)$$

- Random variable for the unit circle condition:

$$Z : \Omega \to \mathbb{R},\ Z(\omega) = |\sqrt{X(\omega)^2 + Y(\omega)^2} - 1|$$

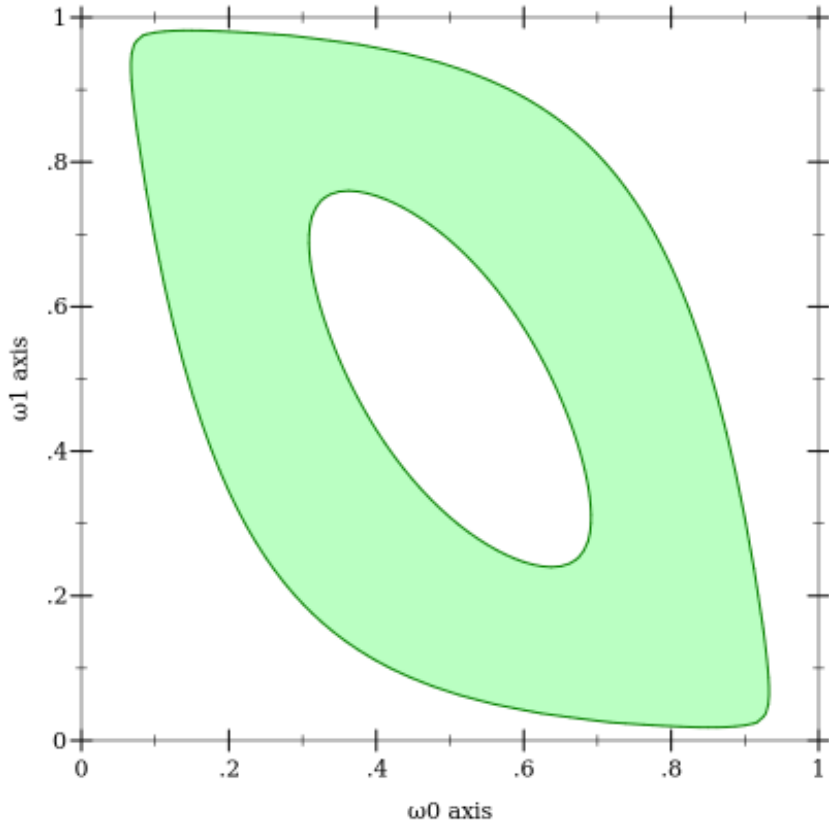
- What do preimages under Z look like? How hard would they be to represent and sample from uniformly?

Preimages Under the Unit Circle Condition

- Random variable for the unit circle condition:

$$Z(\omega) = |\sqrt{X(\omega)^2 + Y(\omega)^2} - 1|$$

$Z^{-1}([0, \epsilon)), \epsilon = 0.5$:

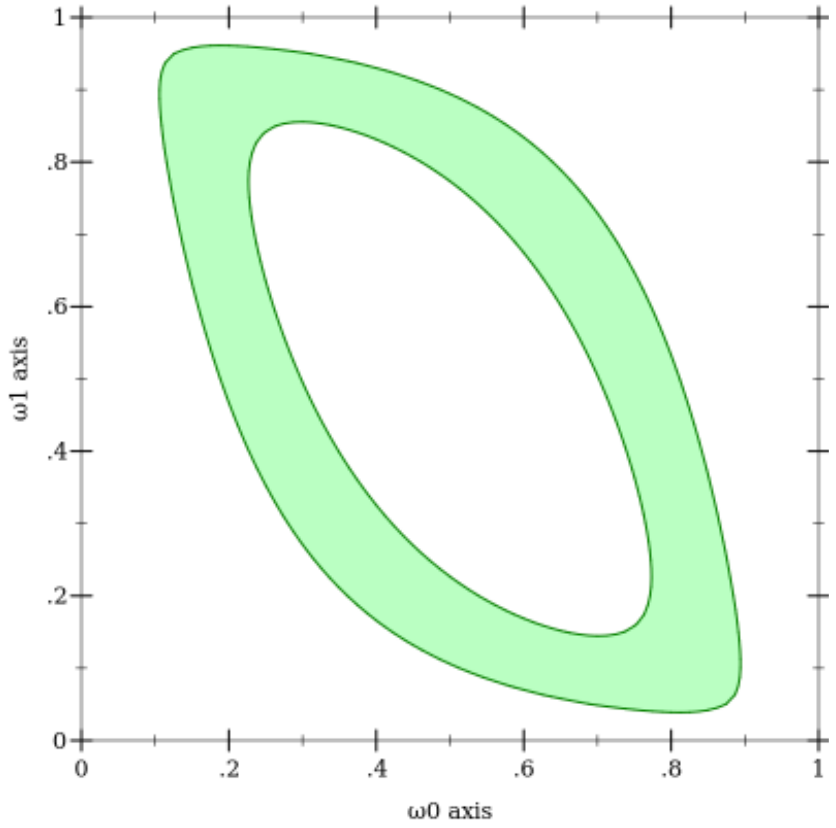


Preimages Under the Unit Circle Condition

- Random variable for the unit circle condition:

$$Z(\omega) = |\sqrt{X(\omega)^2 + Y(\omega)^2} - 1|$$

$Z^{-1}([0, \epsilon)), \epsilon = 0.25$:

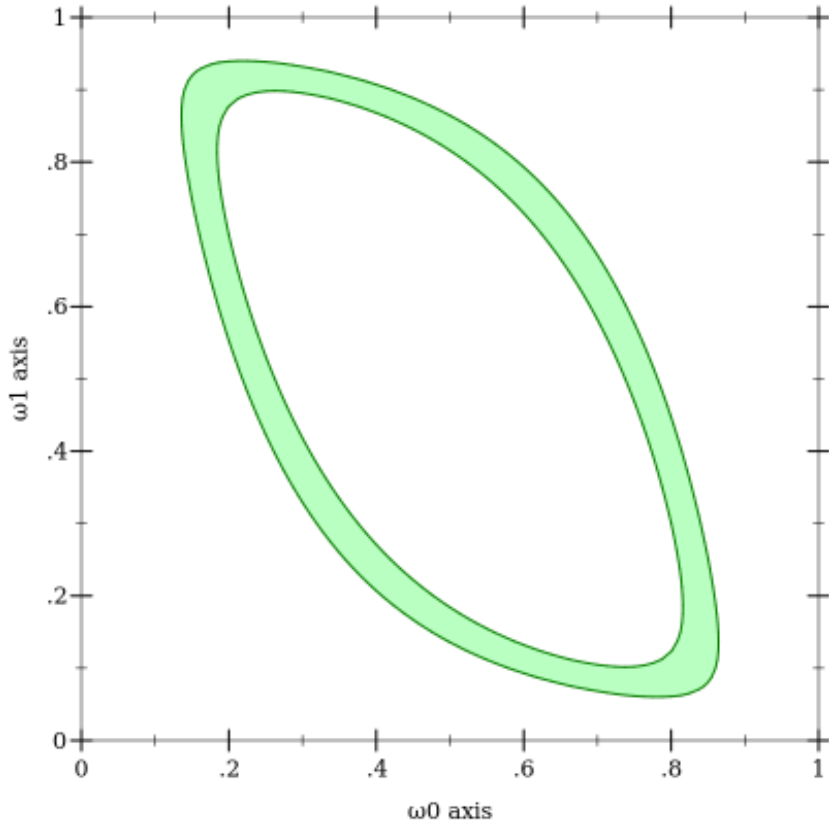


Preimages Under the Unit Circle Condition

- Random variable for the unit circle condition:

$$Z(\omega) = |\sqrt{X(\omega)^2 + Y(\omega)^2} - 1|$$

$Z^{-1}([0, \epsilon)), \epsilon = 0.1$:

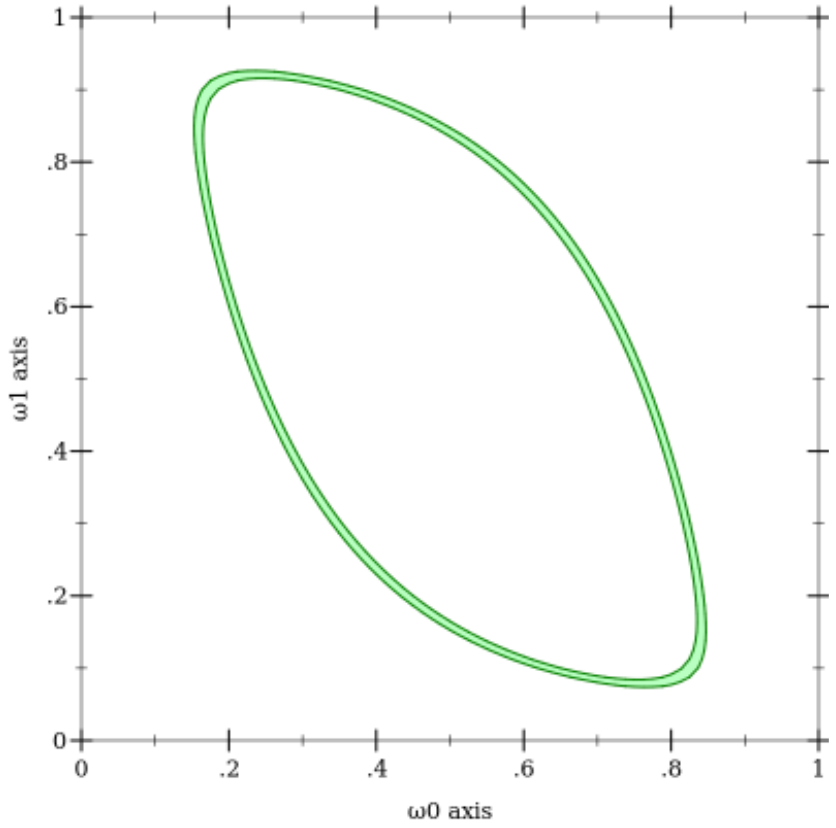


Preimages Under the Unit Circle Condition

- Random variable for the unit circle condition:

$$Z(\omega) = |\sqrt{X(\omega)^2 + Y(\omega)^2} - 1|$$

$Z^{-1}([0, \epsilon)), \epsilon = 0.025$:



Feasible If...

- Seems like we need:
  - Standard interpretation of programs as pure functions from a random source

Feasible If...

- Seems like we need:
  - Standard interpretation of programs as pure functions from a random source
  - Efficient way to compute preimage sets

Feasible If...

- Seems like we need:
  - Standard interpretation of programs as pure functions from a random source
  - Efficient way to compute preimage sets
  - Efficient representation of arbitrary sets

Feasible If...

- Seems like we need:
  - Standard interpretation of programs as pure functions from a random source
  - Efficient way to compute preimage sets
  - Efficient representation of arbitrary sets
  - Efficient way to sample uniformly in preimages

Feasible If...

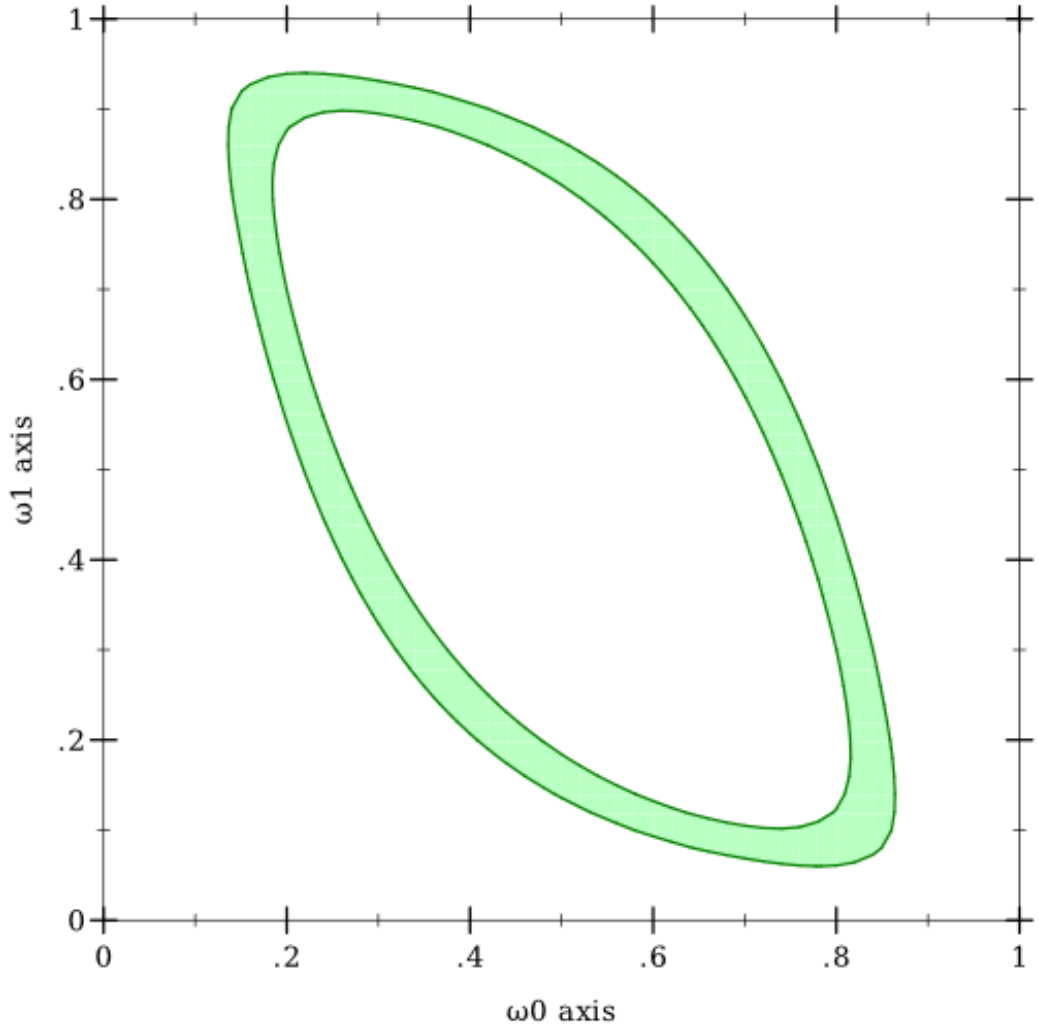
- Seems like we need:
  - Standard interpretation of programs as pure functions from a random source
  - Efficient way to compute preimage sets
  - Efficient representation of arbitrary sets
  - Efficient way to sample uniformly in preimages
  - Proof of correctness w.r.t. standard interpretation

Feasible If...

- Seems like we need:
  - Standard interpretation of programs as pure functions from a random source
  - Efficient way to compute preimage sets
  - Efficient representation of arbitrary sets
  - Efficient way to sample uniformly in preimages
  - Proof of correctness w.r.t. standard interpretation
- Exact implementation is out of reach, but...

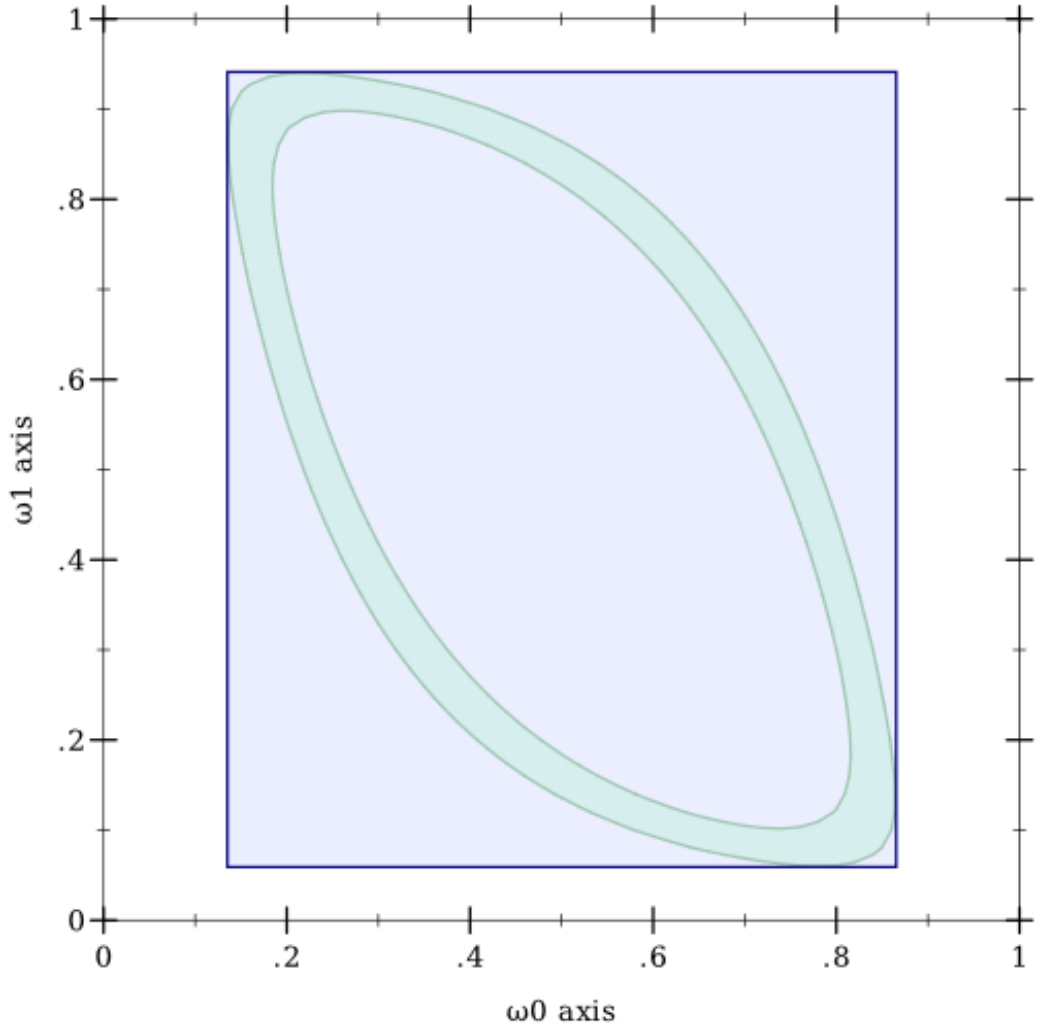
What About Approximating?

Conservative approximation with rectangles:



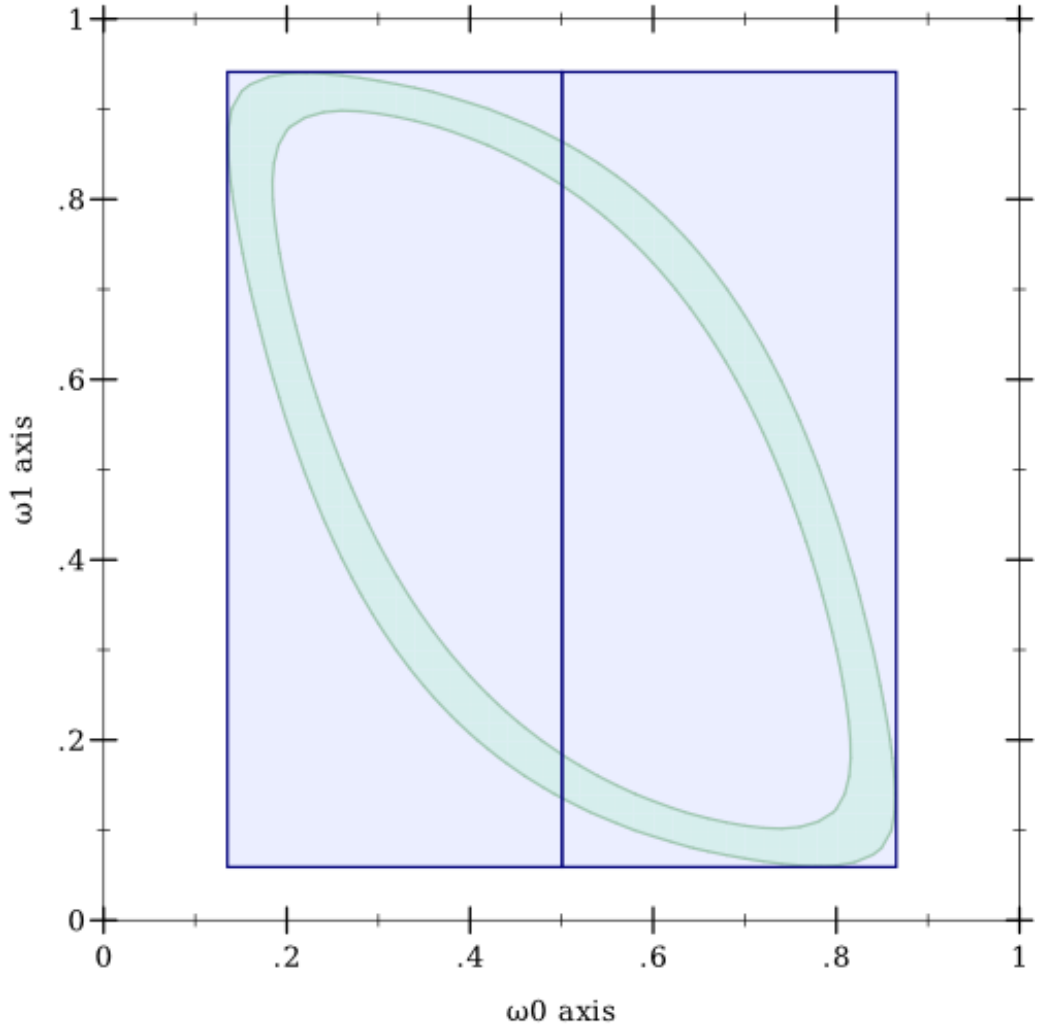
What About Approximating?

Conservative approximation with rectangles:



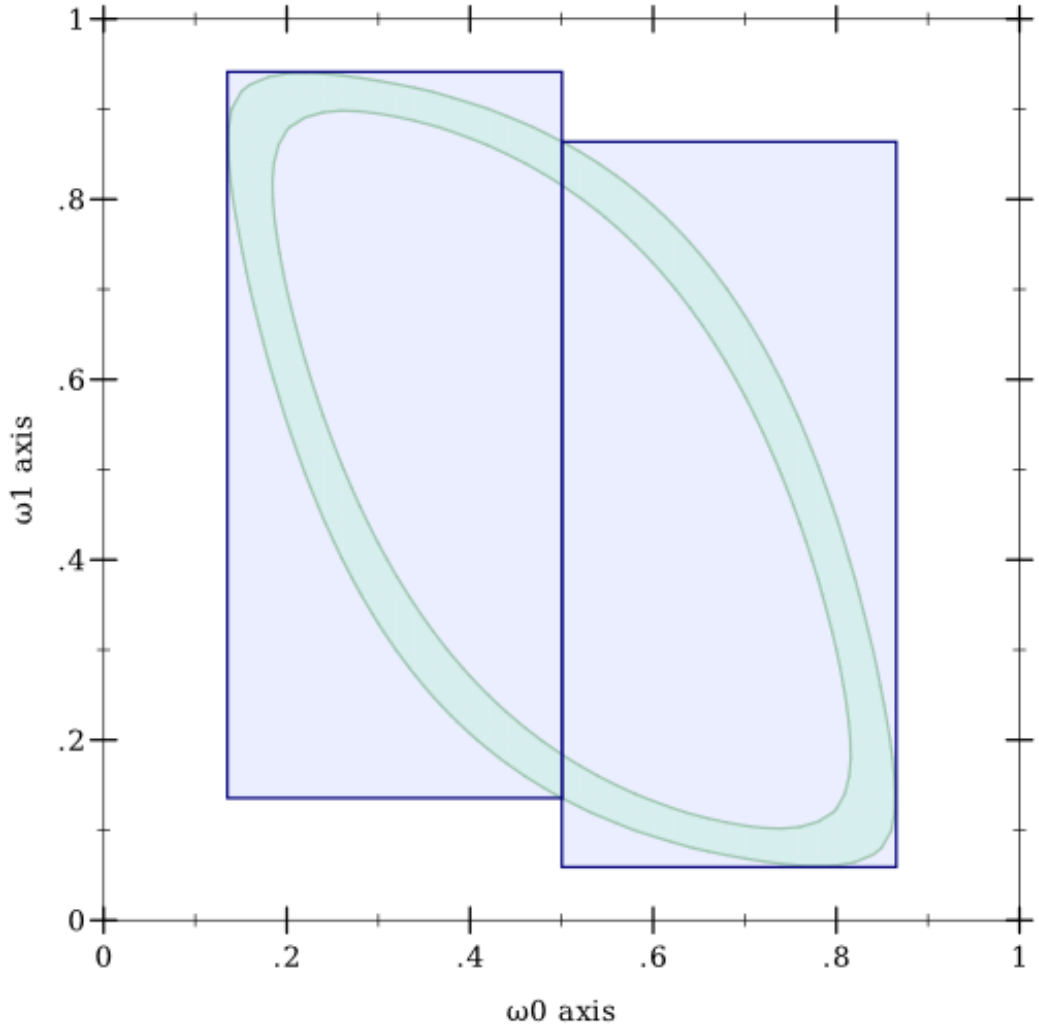
What About Approximating?

Restricting preimages to rectangular subdomains:



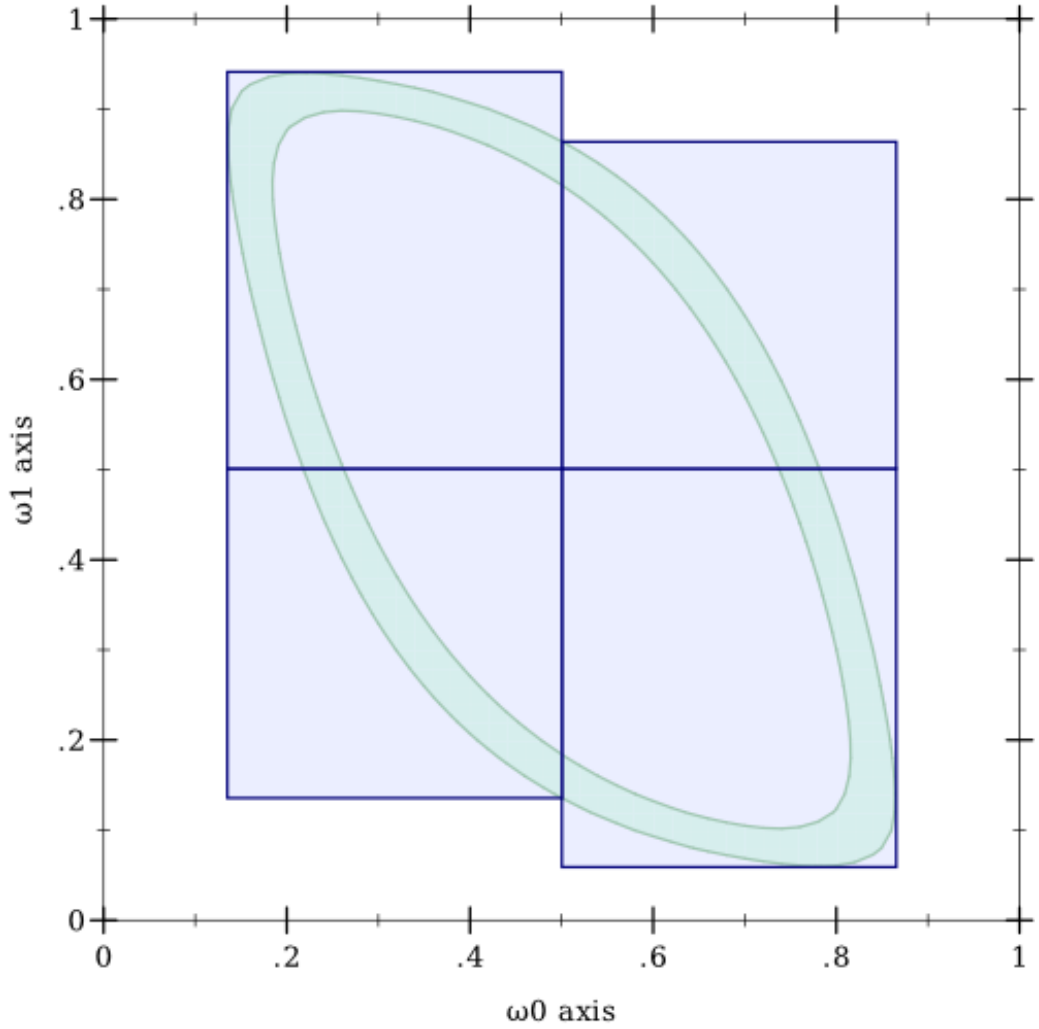
What About Approximating?

Restricting preimages to rectangular subdomains:



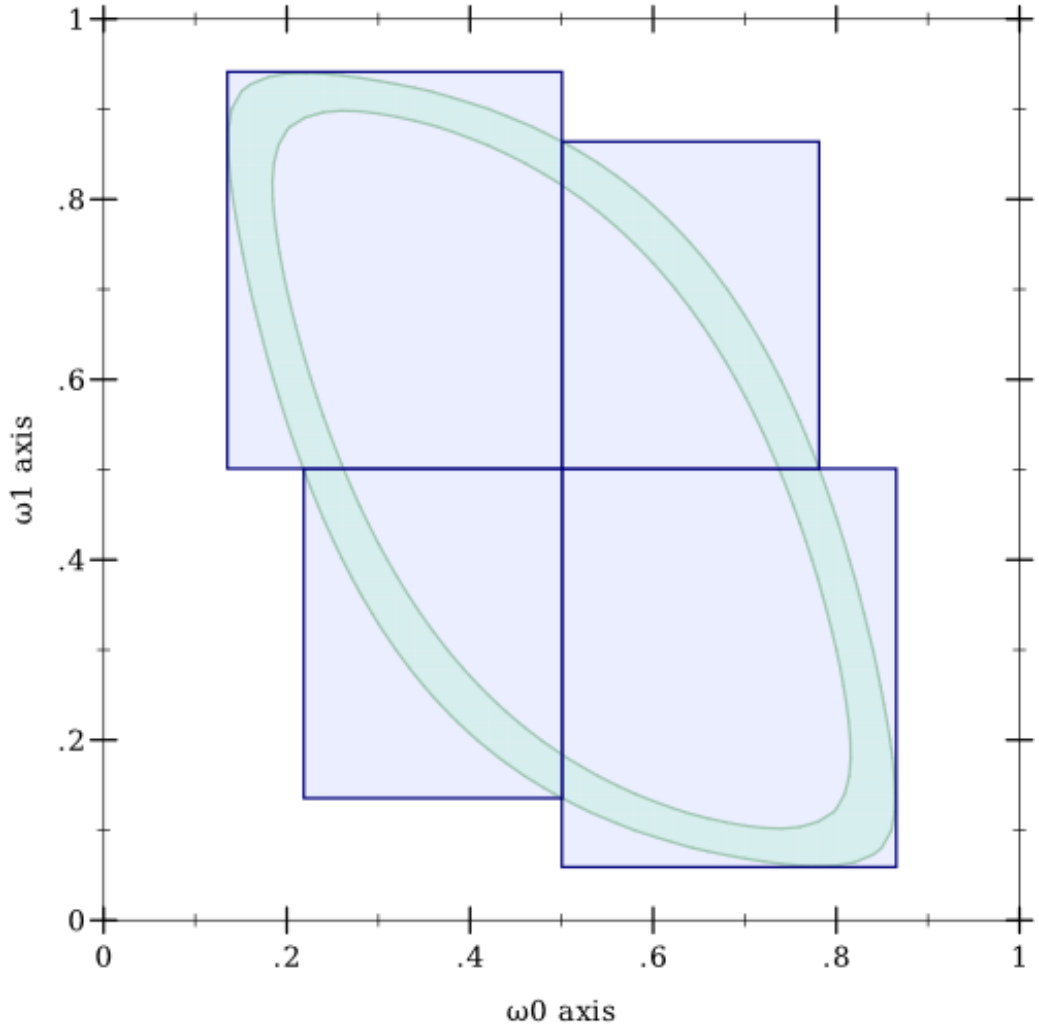
What About Approximating?

Restricting preimages to rectangular subdomains:



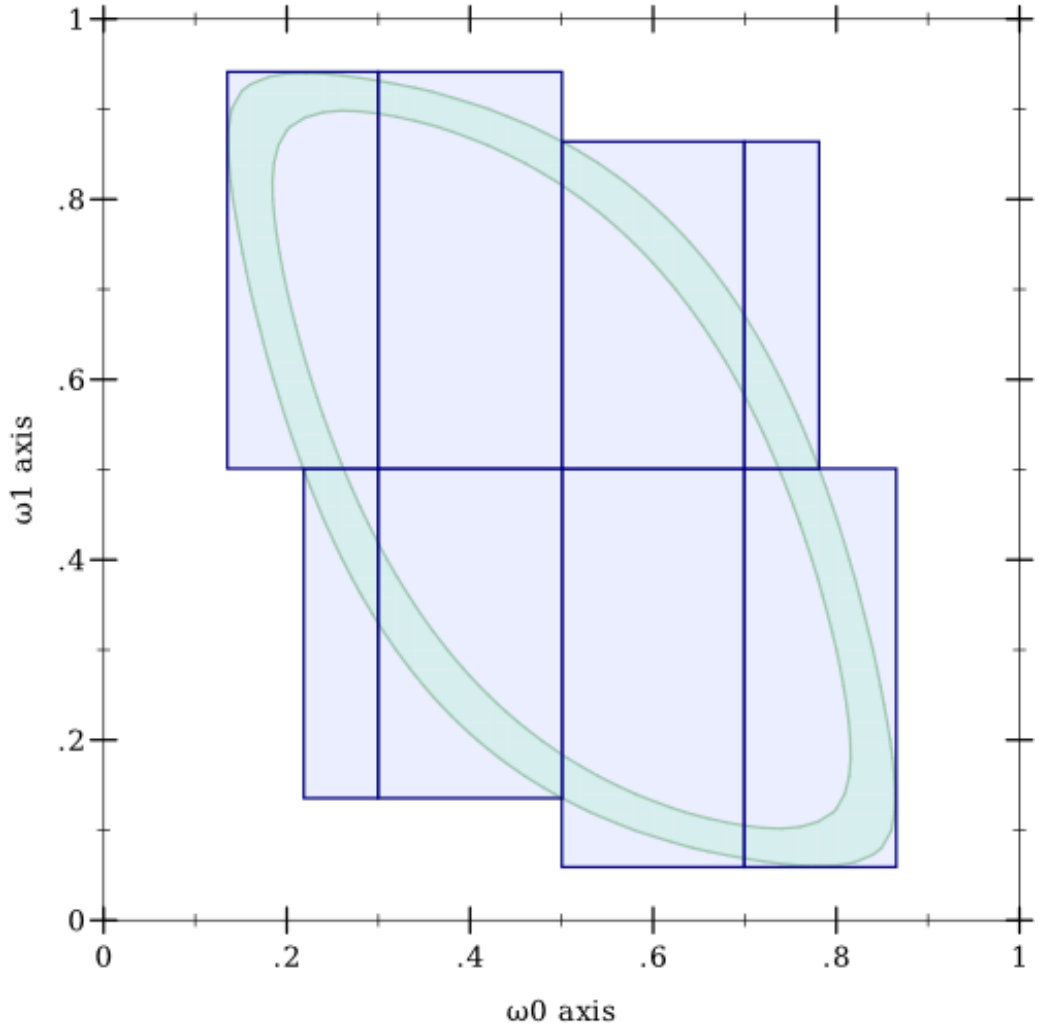
What About Approximating?

Restricting preimages to rectangular subdomains:



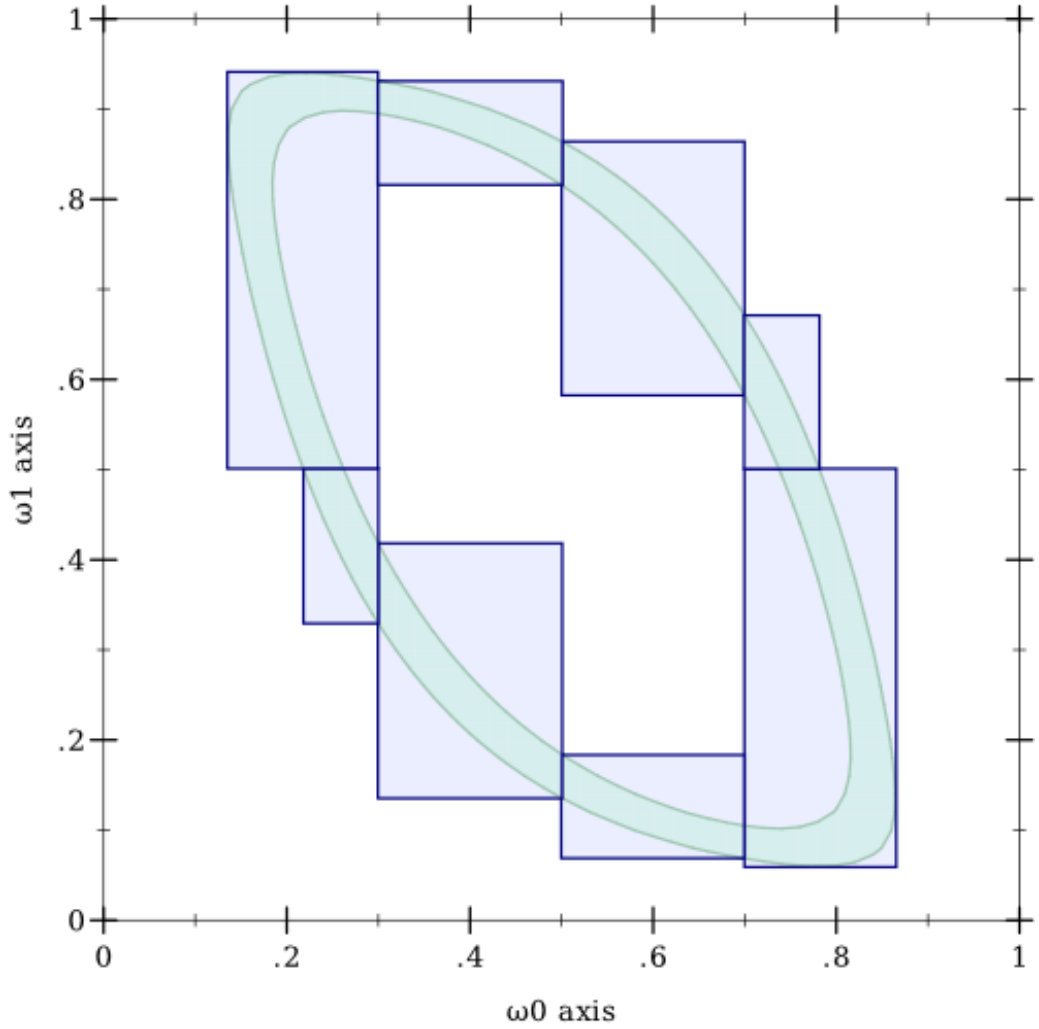
What About Approximating?

Restricting preimages to rectangular subdomains:



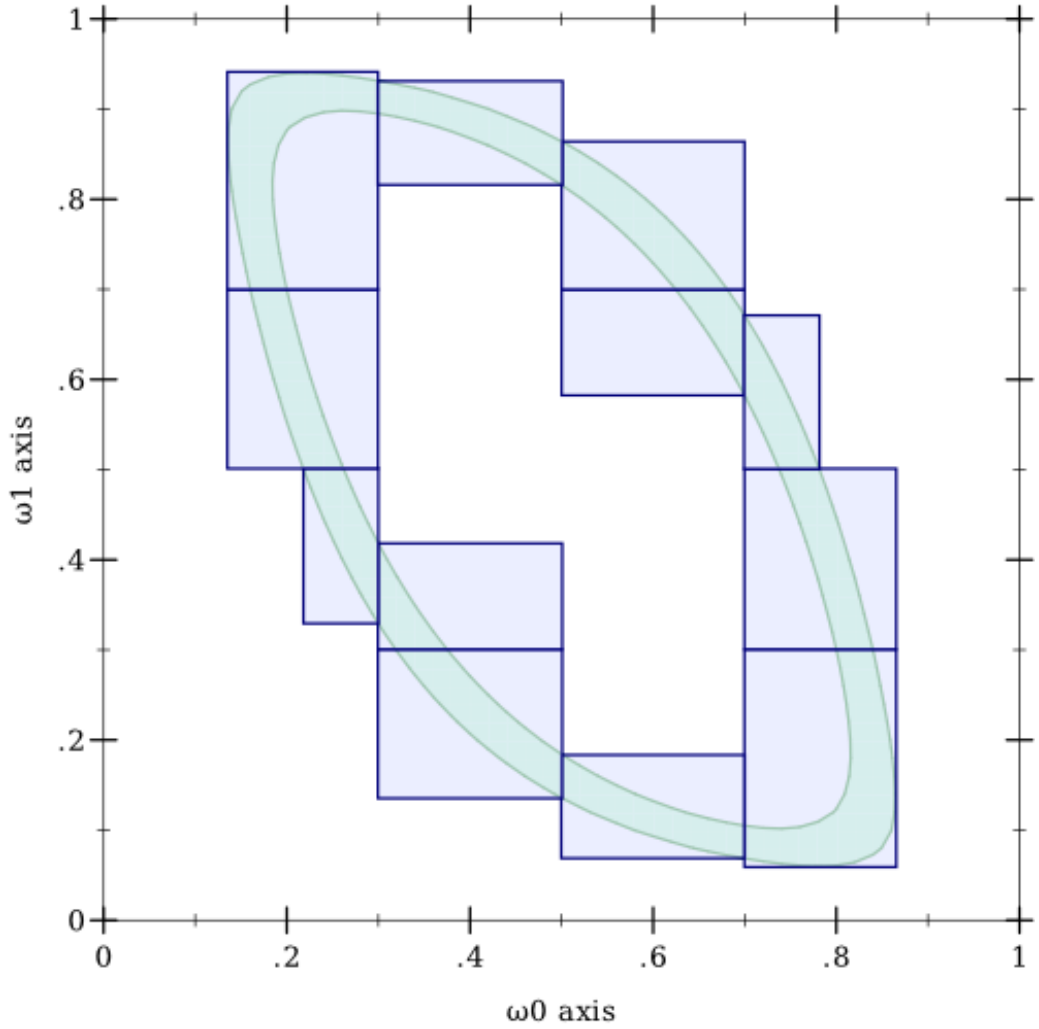
What About Approximating?

Restricting preimages to rectangular subdomains:



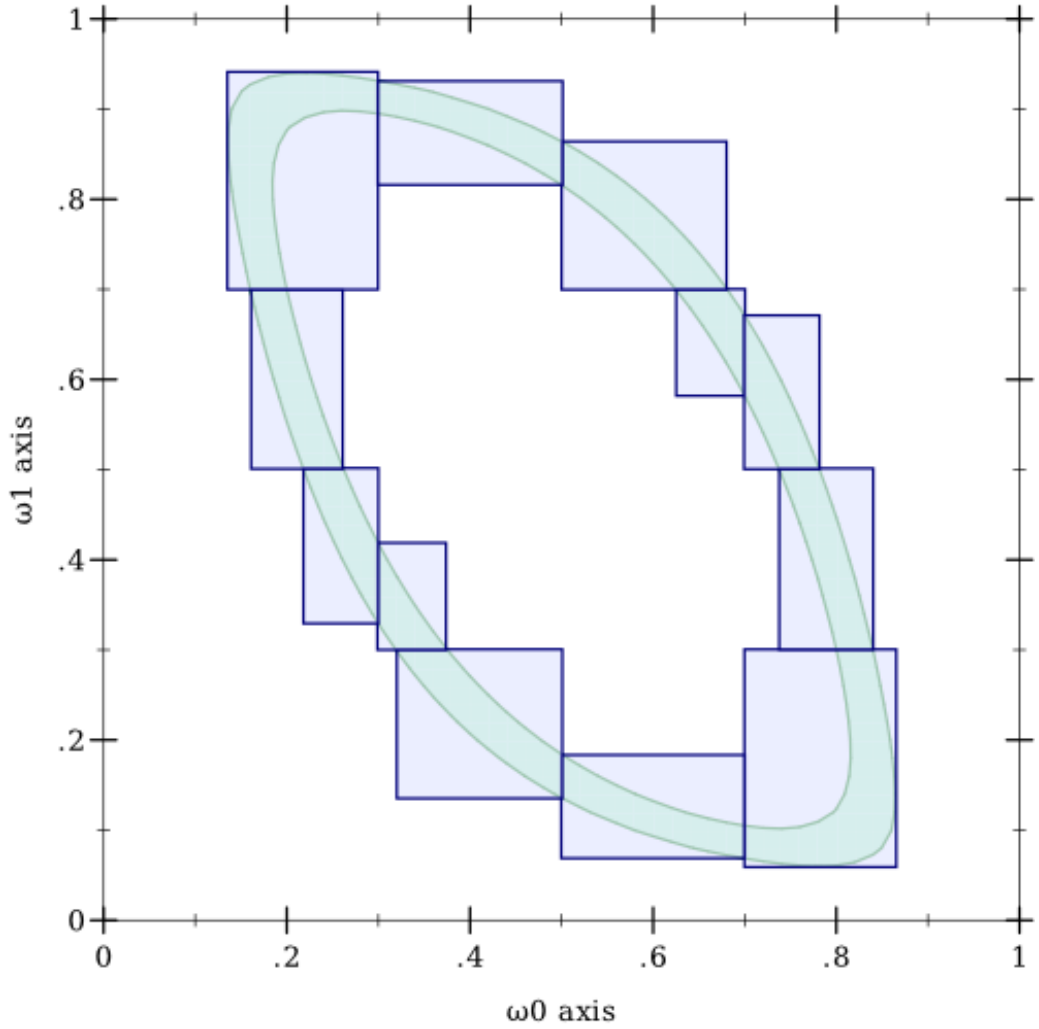
What About Approximating?

Restricting preimages to rectangular subdomains:



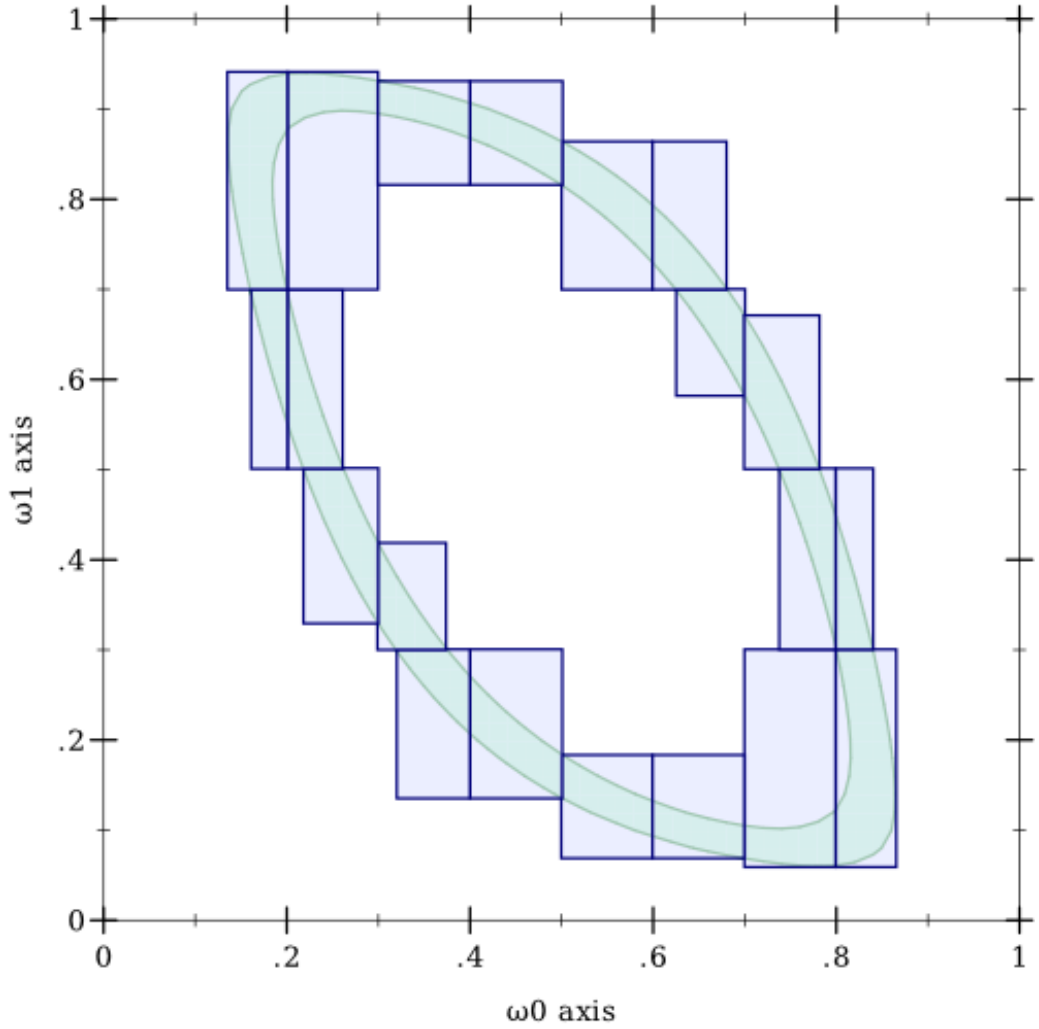
What About Approximating?

Restricting preimages to rectangular subdomains:



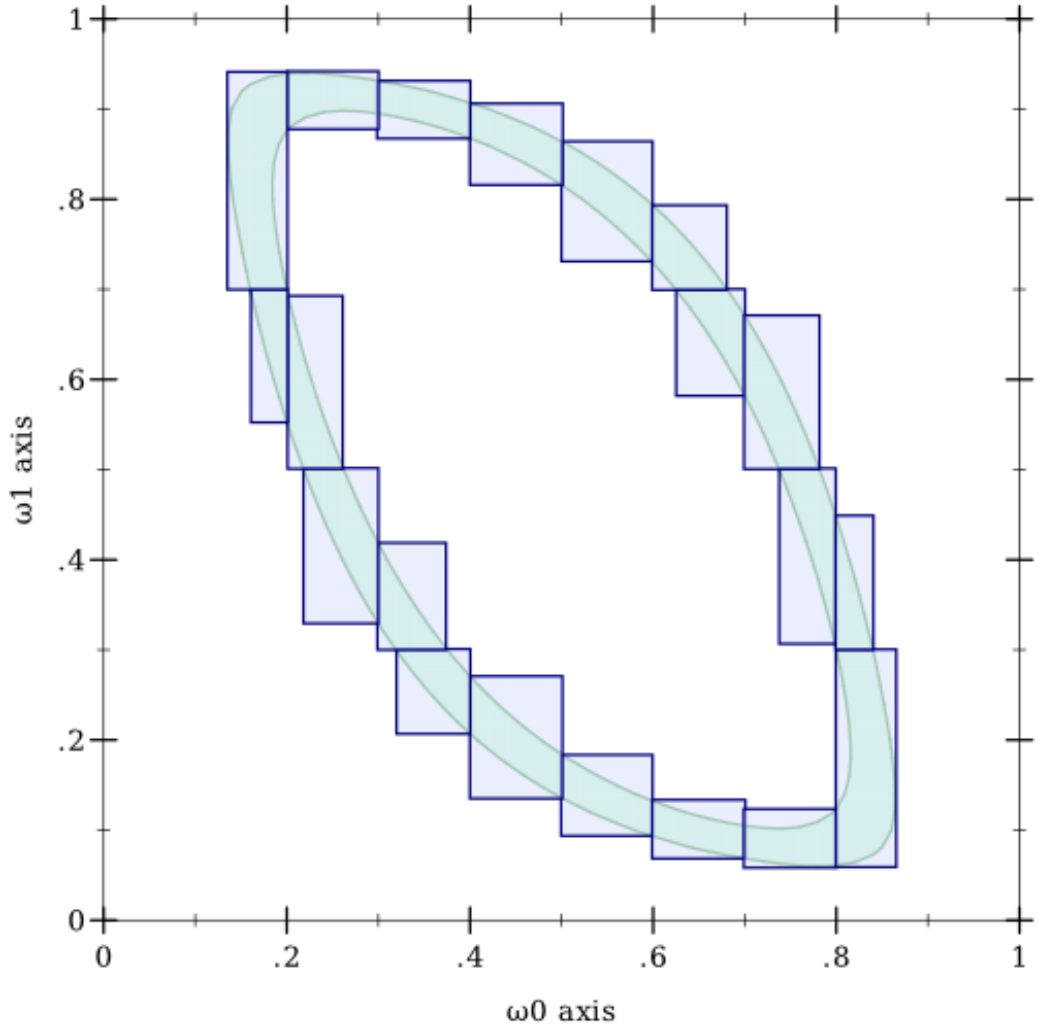
What About Approximating?

Restricting preimages to rectangular subdomains:



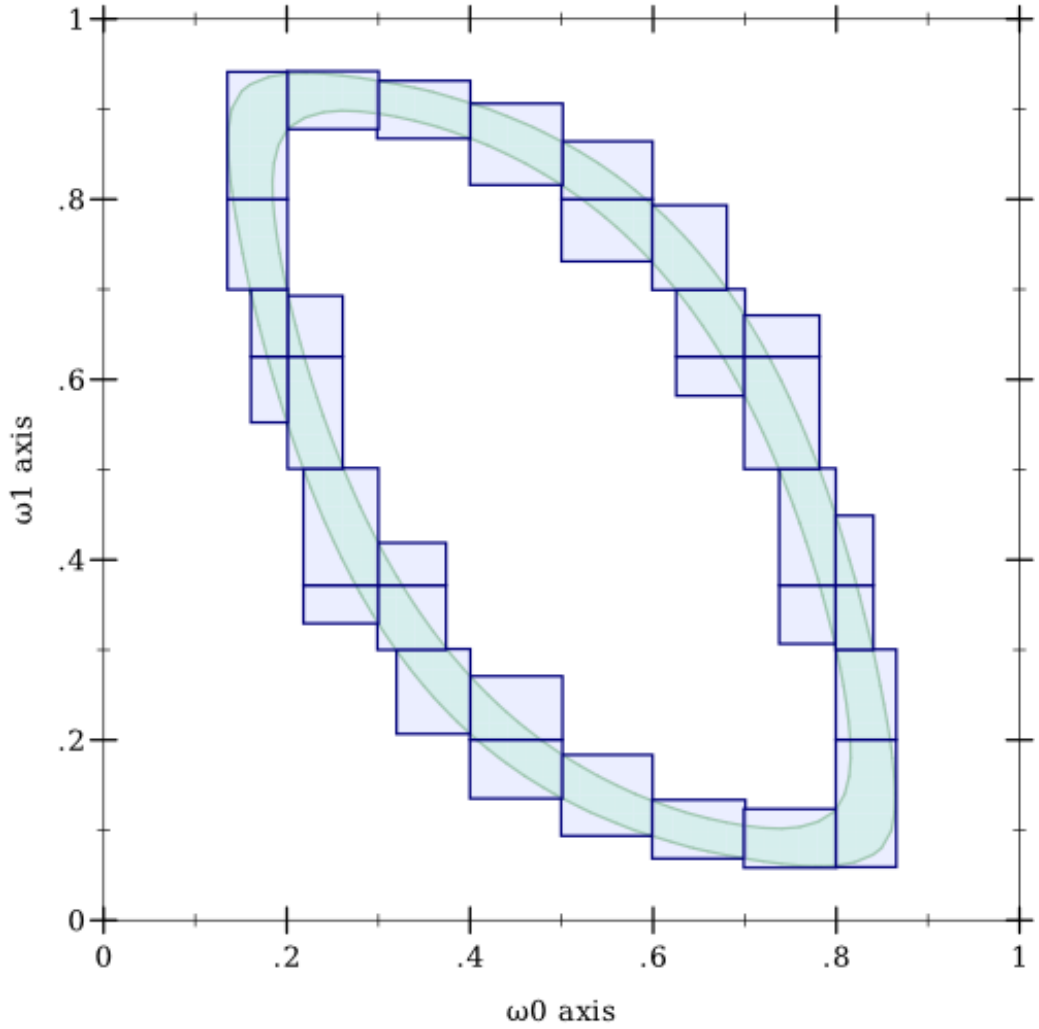
What About Approximating?

Restricting preimages to rectangular subdomains:



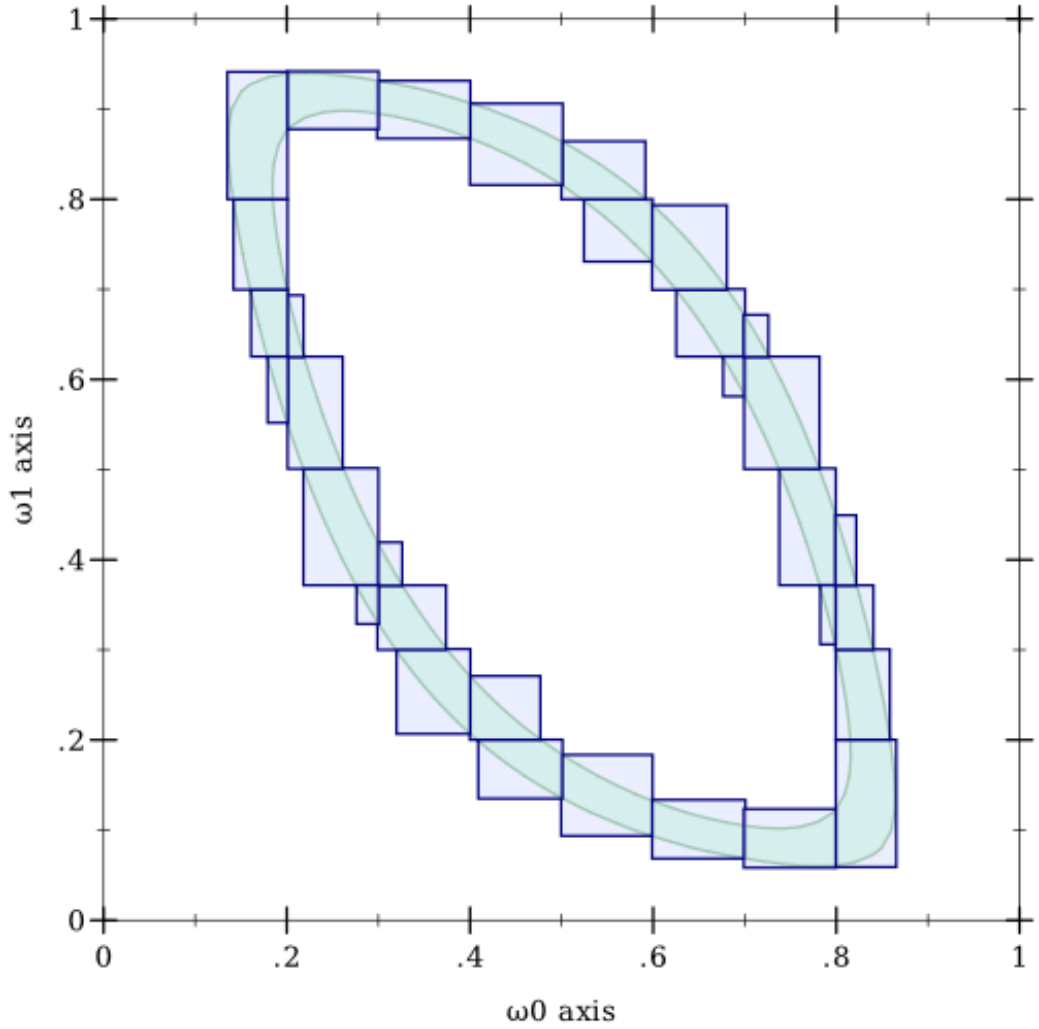
What About Approximating?

Restricting preimages to rectangular subdomains:



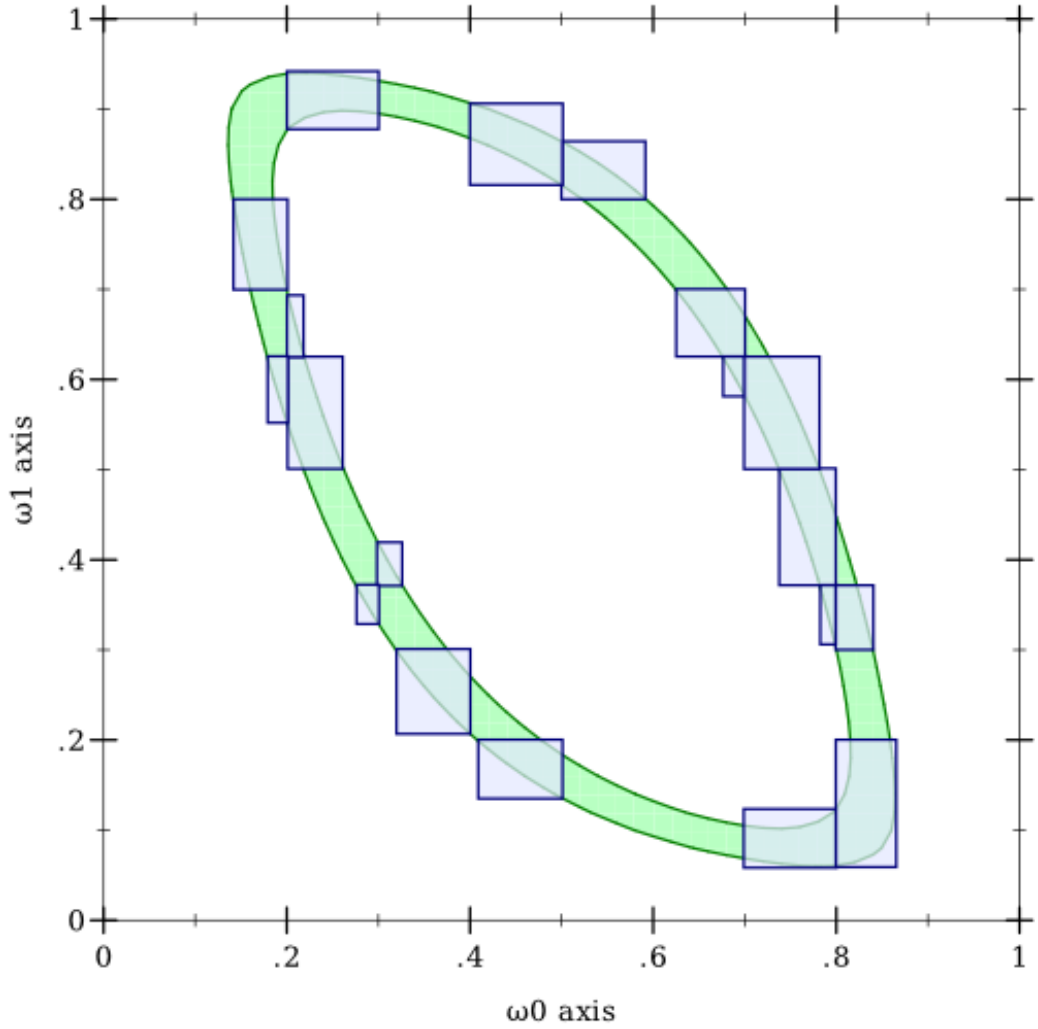
What About Approximating?

Restricting preimages to rectangular subdomains:



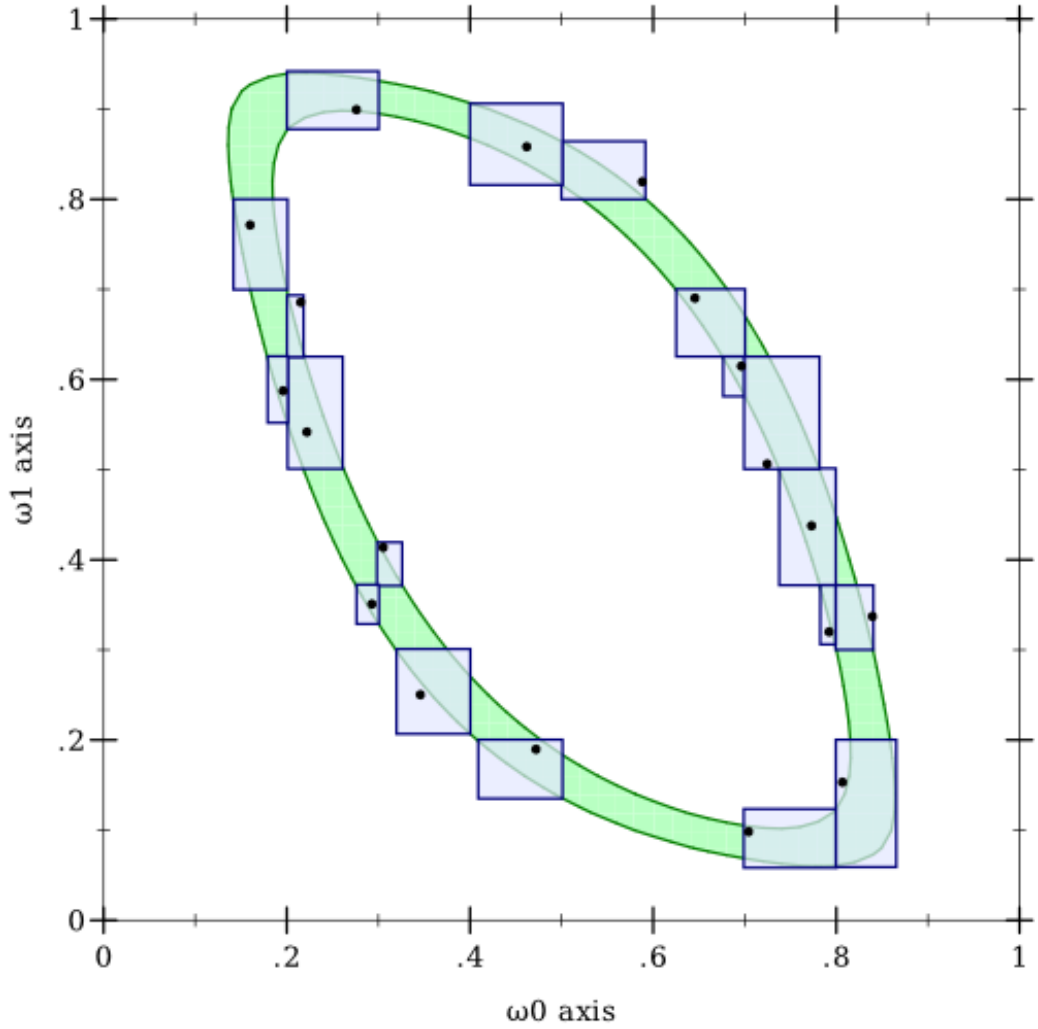
What About Approximating?

Sampling to overcome exponential explosion:



What About Approximating?

Sampling to overcome exponential explosion:



For Conditioned Sampling, We Need...

- Standard interpretation of programs as pure functions from a random source
- Efficient way to compute preimage sets
- Efficient representation of arbitrary sets
- Efficient way to sample uniformly in preimages

- Proof of correctness w.r.t. standard interpretation

For Conditioned Sampling, We Need...

- Standard interpretation of programs as pure functions from a random source
- Efficient way to compute approximate preimage subsets
- Efficient representation of arbitrary sets
- Efficient way to sample uniformly in preimages

- Proof of correctness w.r.t. standard interpretation

For Conditioned Sampling, We Need...

- Standard interpretation of programs as pure functions from a random source
- Efficient way to compute approximate preimage subsets
- Efficient representation of approximating sets
- Efficient way to sample uniformly in preimages

- Proof of correctness w.r.t. standard interpretation

For Conditioned Sampling, We Need...

- Standard interpretation of programs as pure functions from a random source
- Efficient way to compute approximate preimage subsets
- Efficient representation of approximating sets
- Efficient way to sample uniformly in preimages
  - Efficient domain partition sampling
- Proof of correctness w.r.t. standard interpretation

For Conditioned Sampling, We Need...

- Standard interpretation of programs as pure functions from a random source
- Efficient way to compute approximate preimage subsets
- Efficient representation of approximating sets
- Efficient way to sample uniformly in preimages
  - Efficient domain partition sampling
  - Efficient way to determine whether a domain sample is actually in the preimage (just use standard interpretation)
- Proof of correctness w.r.t. standard interpretation

Moar Precision: Standard Interpretation

- Grammar:

$$
\begin{array}{rcl}
p & ::= & x := e;\ \cdots\ ;\ x := e;\ e \\
e & ::= & x\ e \,|\, \text{if } e\ e\ e \,|\, \text{let } e\ e \,|\, \text{env } n \,|\, \langle e, e \rangle \,|\, \delta\ e \,|\, v \\
x & ::= & [\text{first-order function names}] \\
\delta & ::= & [\text{primitive function names}] \\
v & ::= & [\text{first-order values}]
\end{array}
$$

Moar Precision: Standard Interpretation

- Grammar:

$$
\begin{aligned}
p &::= x := e;\ \cdots\ ;\ x := e;\ e \\
e &::= x\ e \mid \text{if } e\ e\ e \mid \text{let } e\ e \mid \text{env } n \mid \langle e, e \rangle \mid \delta\ e \mid v \\
x &::= [\text{first-order function names}] \\
\delta &::= [\text{primitive function names}] \\
v &::= [\text{first-order values}]
\end{aligned}
$$

- Semantic function $[\![\cdot]\!] : p \to (\Omega \to B)$

Moar Precision: Standard Interpretation

- Grammar:

$$
\begin{aligned}
p &::= x := e;\ \cdots\ ;\ x := e;\ e \\
e &::= x\ e \mid \text{if } e\ e\ e \mid \text{let } e\ e \mid \text{env } n \mid \langle e, e \rangle \mid \delta\ e \mid v \\
x &::= [\text{first-order function names}] \\
\delta &::= [\text{primitive function names}] \\
v &::= [\text{first-order values}]
\end{aligned}
$$

- Semantic function $[\![\cdot]\!] : p \to (\Omega \to B)$
- Math has no general recursion, so $[\![p]\!]$ (i.e. interpretation of program p) is a *λ-calculus* term

Moar Precision: Standard Interpretation

- Grammar:

$$
\begin{array}{rcl}
p & ::= & x := e;\ \cdots\ ;\ x := e;\ e \\
e & ::= & x\ e \,|\, \text{if } e\ e\ e \,|\, \text{let } e\ e \,|\, \text{env } n \,|\, \langle e, e \rangle \,|\, \delta\ e \,|\, v \\
x & ::= & [\text{first-order function names}] \\
\delta & ::= & [\text{primitive function names}] \\
v & ::= & [\text{first-order values}]
\end{array}
$$

- Semantic function $[\![\cdot]\!] : p \to (\Omega \to B)$
- Math has no general recursion, so $[\![p]\!]$ (i.e. interpretation of program p) is a *λ-calculus* term
- Easy implementation: $[\![\cdot]\!]$ becomes a Racket macro

Why a First-Order Language?

Why a First-Order Language?

- Short answer: Measure theory is essentially first-order

Why a First-Order Language?

- Short answer: Measure theory is essentially first-order
- Longer answer: It is difficult to assign probabilities to sets of functions

Why a First-Order Language?

- Short answer: Measure theory is essentially first-order
- Longer answer: It is difficult to assign probabilities to sets of functions
- Precise answer: For uncountable Borel spaces A and B, there is no σ-algebra for the set of measurable functions in $A \to B$ for which $\mathrm{app} : (A \to B) \times A \to B$ is measurable

Why a First-Order Language?

- Short answer: Measure theory is essentially first-order
- Longer answer: It is difficult to assign probabilities to sets of functions
- Precise answer: For uncountable Borel spaces A and B, there is no σ-algebra for the set of measurable functions in $A \to B$ for which $\mathrm{app} : (A \to B) \times A \to B$ is measurable
- Short version: If you stick to standard measurable sets, the results of applying a random function to a random value can't have a sensible distribution

Why a First-Order Language?

- Short answer: Measure theory is essentially first-order
- Longer answer: It is difficult to assign probabilities to sets of functions
- Precise answer: For uncountable Borel spaces A and B, there is no σ-algebra for the set of measurable functions in $A \to B$ for which $\mathrm{app} : (A \to B) \times A \to B$ is measurable
- Short version: If you stick to standard measurable sets, the results of applying a random function to a random value can't have a sensible distribution
- It's okay, we can defunctionalize lambdas away (closures are just products)

Moar Precision: Nonstandard Interpretation

- Main idea: Compute preimages compositionally

Moar Precision: Nonstandard Interpretation

- Main idea: Compute preimages compositionally
- Preimage computation type constructor:

$$A \underset{\mathbf{pre}}{\rightsquigarrow} B ::= \mathrm{Set}(A) \to \langle \mathrm{Set}(B), \mathrm{Set}(B) \to \mathrm{Set}(A) \rangle$$

Moar Precision: Nonstandard Interpretation

- Main idea: Compute preimages compositionally
- Preimage computation type constructor:

$$A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} B ::= \mathrm{Set}(A) \rightarrow \langle \mathrm{Set}(B), \mathrm{Set}(B) \rightarrow \mathrm{Set}(A) \rangle$$

- Pair preimage combinator type:

$$\mathrm{pair}_{\mathbf{pre}} : (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} B_1) \rightarrow (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} B_2) \rightarrow (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} \langle B_1, B_2 \rangle)$$

Moar Precision: Nonstandard Interpretation

- Main idea: Compute preimages compositionally
- Preimage computation type constructor:

$$A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B ::= \mathrm{Set}(A) \to \langle \mathrm{Set}(B), \mathrm{Set}(B) \to \mathrm{Set}(A) \rangle$$

- Pair preimage combinator type:

$$\mathrm{pair}_{\mathbf{pre}} : (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B_1) \to (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B_2) \to (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} \langle B_1, B_2 \rangle)$$

Theorem. If

- $h_1 : A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B_1$ computes preimages under $f_1 : A \to B_1$

Moar Precision: Nonstandard Interpretation

- Main idea: Compute preimages compositionally
- Preimage computation type constructor:

$$A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B ::= \mathrm{Set}(A) \to \langle \mathrm{Set}(B), \mathrm{Set}(B) \to \mathrm{Set}(A) \rangle$$

- Pair preimage combinator type:

$$\mathrm{pair}_{\mathbf{pre}} : (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B_1) \to (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B_2) \to (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} \langle B_1, B_2 \rangle)$$

Theorem. If

- $h_1 : A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B_1$ computes preimages under $f_1 : A \to B_1$
- $h_2 : A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B_2$ computes preimages under $f_2 : A \to B_2$

Moar Precision: Nonstandard Interpretation

- Main idea: Compute preimages compositionally
- Preimage computation type constructor:

$$A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} B ::= \mathrm{Set}(A) \to \langle \mathrm{Set}(B), \mathrm{Set}(B) \to \mathrm{Set}(A) \rangle$$

- Pair preimage combinator type:

$$\mathrm{pair}_{\mathbf{pre}} : (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} B_1) \to (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} B_2) \to (A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} \langle B_1, B_2 \rangle)$$

Theorem. If

- $h_1 : A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} B_1$ computes preimages under $f_1 : A \to B_1$
- $h_2 : A \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\rightarrow}} B_2$ computes preimages under $f_2 : A \to B_2$

then $\mathrm{pair}_{\mathbf{pre}}\ h_1\ h_2$ computes preimages under $a \mapsto \langle f_1(a), f_2(a) \rangle$.

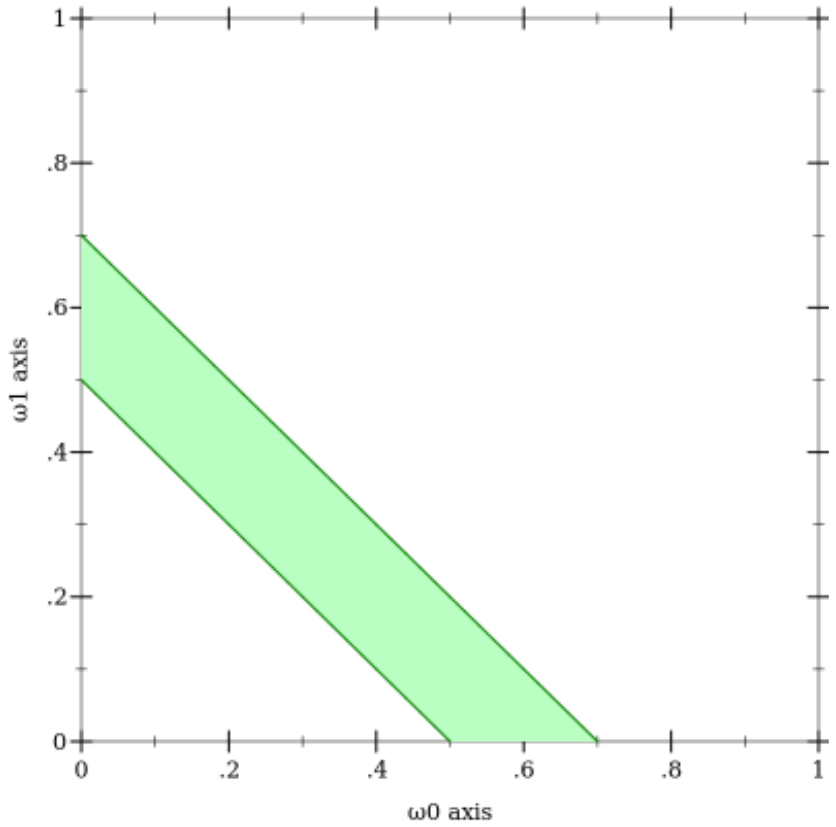
Pair Preimages

$$f_1(\omega) = \omega_0 + \omega_1 \qquad f_2(\omega) = \omega_0 \cdot \omega_1$$

Pair Preimages

$$f_1(\omega) = \omega_0 + \omega_1 \qquad f_2(\omega) = \omega_0 \cdot \omega_1$$

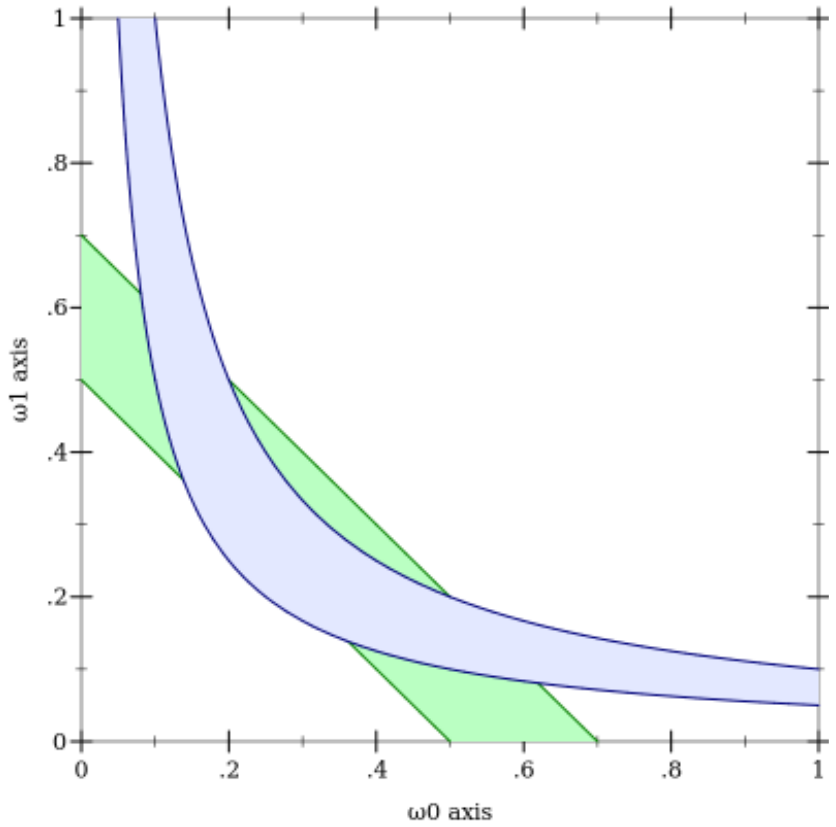
$f_1^{-1}([0.5, 0.7])$:



Pair Preimages

$$f_1(\omega) = \omega_0 + \omega_1 \qquad f_2(\omega) = \omega_0 \cdot \omega_1$$

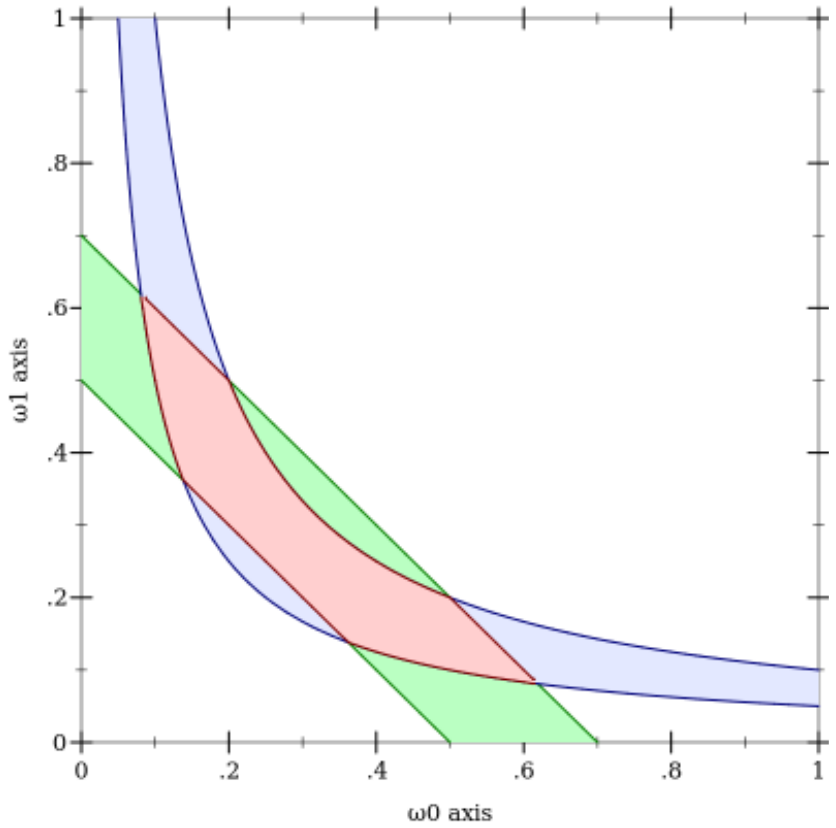
$f_1^{-1}([0.5, 0.7])$ and $f_2^{-1}([0.05, 0.1])$:



Pair Preimages

$$f_1(\omega) = \omega_0 + \omega_1 \qquad f_2(\omega) = \omega_0 \cdot \omega_1$$

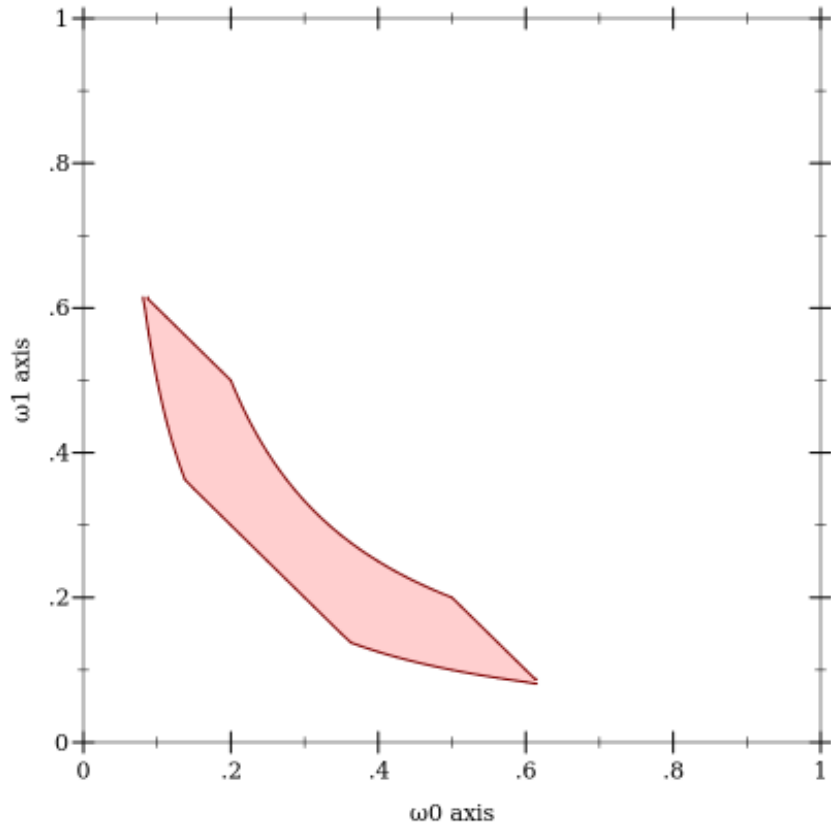
$f^{-1}([0.5, 0.7] \times [0.05, 0.1])$ where $f(\omega) = \langle f_1(\omega), f_2(\omega) \rangle$:



Pair Preimages

$$f_1(\omega) = \omega_0 + \omega_1 \qquad f_2(\omega) = \omega_0 \cdot \omega_1$$

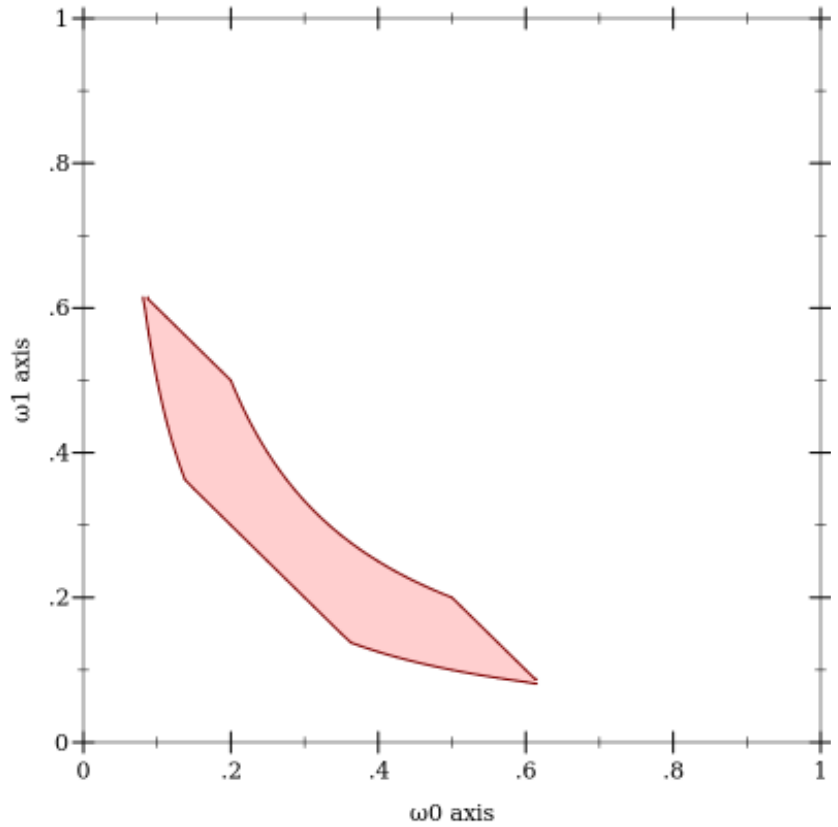
$f^{-1}([0.5, 0.7] \times [0.05, 0.1])$ where $f(\omega) = \langle f_1(\omega), f_2(\omega) \rangle$:



Pair Preimages

$$f_1(\omega) = \omega_0 + \omega_1 \qquad f_2(\omega) = \omega_0 \cdot \omega_1$$

$f^{-1}([0.5, 0.7] \times [0.05, 0.1])$ where $f(\omega) = \langle f_1(\omega), f_2(\omega) \rangle$:



- pair_{pre} does this, but for arbitrary output sets

Compositional Semantics

- Semantic function $[\![\cdot]\!]_{\mathbf{pre}} : p \to (\Omega \overset{\sim\!\!\rightarrow}{\mathbf{pre}} B)$

Compositional Semantics

- Semantic function $[\![\cdot]\!]_{\mathbf{pre}} : p \to (\Omega \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{\to}} B)$
  - Compare $[\![\cdot]\!] : p \to (\Omega \to B)$

Compositional Semantics

- Semantic function $[\![\cdot]\!]_{\mathbf{pre}} : p \to (\Omega \overset{\sim}{\underset{\mathbf{pre}}{\to}} B)$
  - Compare $[\![\cdot]\!] : p \to (\Omega \to B)$
- Compositional definition; e.g.

$$[\![\langle e_1, e_2 \rangle]\!]_{\mathbf{pre}} = \mathrm{pair}_{\mathbf{pre}}\ [\![e_1]\!]_{\mathbf{pre}}\ [\![e_2]\!]_{\mathbf{pre}}$$

Compositional Semantics

- Semantic function $[\![\cdot]\!]_{\mathrm{pre}} : p \to (\Omega \overset{\sim}{\underset{\mathrm{pre}}{\to}} B)$
  - Compare $[\![\cdot]\!] : p \to (\Omega \to B)$
- Compositional definition; e.g.

$$[\![\langle e_1, e_2 \rangle]\!]_{\mathrm{pre}} = \mathrm{pair}_{\mathrm{pre}}\ [\![e_1]\!]_{\mathrm{pre}}\ [\![e_2]\!]_{\mathrm{pre}}$$

Corollary. If

- $[\![e_1]\!]_{\mathrm{pre}}$ computes preimages under $[\![e_1]\!]$
- $[\![e_2]\!]_{\mathrm{pre}}$ computes preimages under $[\![e_2]\!]$

then $[\![\langle e_1, e_2 \rangle]\!]_{\mathrm{pre}}$ computes preimages under $[\![\langle e_1, e_2 \rangle]\!]$.

Nonstandard Interpretation Correctness

Theorem. For all programs p, $[\![p]\!]_{\mathbf{pre}}$ computes preimages under $[\![p]\!]$.

Proof. By structural induction on program terms.

Wait a Minute

- Q. Don’t the interpretations of $[\![\cdot]\!]_{\mathbf{pre}}$ do uncomputable things?

Wait a Minute

- Q. Don’t the interpretations of $[\![\cdot]\!]_{\mathbf{pre}}$ do uncomputable things?
  - A. Yes. Yes, they do.

Wait a Minute

- Q. Don’t the interpretations of $[\![\cdot]\!]_{\mathbf{pre}}$ do uncomputable things?
  - A. Yes. Yes, they do.
- Q. Where do I get a computer that runs them?

Wait a Minute

- Q. Don't the interpretations of $[\![\cdot]\!]_{\mathbf{pre}}$ do uncomputable things?
  - A. Yes. Yes, they do.
- Q. Where do I get a computer that runs them?
  - A. Nowhere, but we can approximate them.

Wait a Minute

- Q. Don't the interpretations of $[\![\cdot]\!]_{\mathbf{pre}}$ do uncomputable things?
  - A. Yes. Yes, they do.
- Q. Where do I get a computer that runs them?
  - A. Nowhere, but we can approximate them.
- Q. Where did you get a λ-calculus that could operate on arbitrary, possibly infinite sets, anyway?
  - A. Well...

Lambda-ZFC

N. Toronto and J. McCarthy. Computing in Cantor's Paradise With λ_{ZFC}. FLOPS 2012

Lambda-ZFC

N. Toronto and J. McCarthy. Computing in Cantor's Paradise With λ_{ZFC}. FLOPS 2012

λ calculus

Lambda-ZFC

N. Toronto and J. McCarthy. Computing in Cantor's Paradise With λ_{ZFC}. FLOPS 2012

λ calculus

+

Set theory

Lambda-ZFC

N. Toronto and J. McCarthy. Computing in Cantor's Paradise With λ_{ZFC}. FLOPS 2012

λ calculus

+

Set theory

=

λ_{ZFC}

Lambda-ZFC

N. Toronto and J. McCarthy. Computing in Cantor's Paradise With λ_{ZFC}. FLOPS 2012

λ calculus

Set theory

λ_{ZFC}

- Contemporary math, but with lambdas and general recursion; or functional programming, but with infinite sets

Lambda-ZFC

N. Toronto and J. McCarthy. Computing in Cantor's Paradise With λ_{ZFC}. FLOPS 2012

λ calculus

Set theory

λ_{ZFC}

- Contemporary math, but with lambdas and general recursion; or functional programming, but with infinite sets
- Can use essentially all contemporary mathematical theorems

Lambda-ZFC

N. Toronto and J. McCarthy. Computing in Cantor's Paradise With λ_{ZFC}. FLOPS 2012

λ calculus

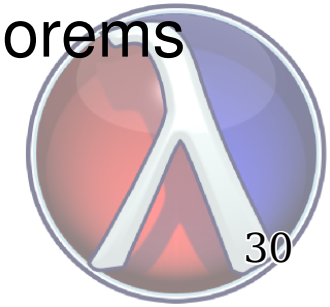
+

Set theory

=

λ_{ZFC}

- Contemporary math, but with lambdas and general recursion; or functional programming, but with infinite sets
- Can use essentially all contemporary mathematical theorems
- New theorems apply in contemporary mathematics*



Lambda-ZFC

N. Toronto and J. McCarthy. Computing in Cantor's Paradise With λ_{ZFC}. FLOPS 2012

λ calculus

Set theory

λ_{ZFC}

- Contemporary math, but with lambdas and general recursion; or functional programming, but with infinite sets
- Can use essentially all contemporary mathematical theorems
- New theorems apply in contemporary mathematics*

* assuming the existence of an inaccessible cardinal

Everything Is Measurable

- “Measurable” means “can have a sensible distribution”

Everything Is Measurable

- “Measurable” means “can have a sensible distribution”

Theorem. If every primitive function (i.e. those with names in δ) is measurable w.r.t. standard σ-algebras, then $[\![p]\!]$ is measurable w.r.t. standard σ-algebras, regardless of nontermination.

Everything Is Measurable

- "Measurable" means "can have a sensible distribution"

Theorem. If every primitive function (i.e. those with names in δ) is measurable w.r.t. standard σ-algebras, then $[\![p]\!]$ is measurable w.r.t. standard σ-algebras, regardless of nontermination.

Corollary. The halting set of every program is measurable.

Everything Is Measurable

- "Measurable" means "can have a sensible distribution"

Theorem. If every primitive function (i.e. those with names in δ) is measurable w.r.t. standard σ-algebras, then $[\![p]\!]$ is measurable w.r.t. standard σ-algebras, regardless of nontermination.

Corollary. The halting set of every program is measurable.

- Requires only that the standard σ-algebra of Bool is $\mathcal{P}(\mathrm{Bool})$ (i.e. true and false are distinguishable)

Everything Is Measurable

- "Measurable" means "can have a sensible distribution"

Theorem. If every primitive function (i.e. those with names in δ) is measurable w.r.t. standard σ-algebras, then $[\![p]\!]$ is measurable w.r.t. standard σ-algebras, regardless of nontermination.

Corollary. The halting set of every program is measurable.

- Requires only that the standard σ-algebra of Bool is $\mathcal{P}(\mathrm{Bool})$ (i.e. true and false are distinguishable)
- Includes uncomputable primitives like real equality and limits

Everything Is Measurable

- "Measurable" means "can have a sensible distribution"

Theorem. If every primitive function (i.e. those with names in δ) is measurable w.r.t. standard σ-algebras, then $[\![p]\!]$ is measurable w.r.t. standard σ-algebras, regardless of nontermination.

Corollary. The halting set of every program is measurable.

- Requires only that the standard σ-algebra of Bool is $\mathcal{P}(\mathrm{Bool})$ (i.e. true and false are distinguishable)
- Includes uncomputable primitives like real equality and limits
- Applies to all probabilistic language work to date and in the forseeable future

Back To Approximation

- $[\![\cdot]\!]_{\mathbf{pre}} : p \to (\Omega \overset{\rightsquigarrow}{\underset{\mathbf{pre}}{}} B)$ returns uncomputable functions

Back To Approximation

- $[\![\cdot]\!]_{\mathbf{pre}} : p \to (\Omega \underset{\mathbf{pre}}{\rightsquigarrow} B)$ returns uncomputable functions
- Recall:

$$A \underset{\mathbf{pre}}{\rightsquigarrow} B ::= \mathrm{Set}(A) \to \langle \mathrm{Set}(B), \mathrm{Set}(B) \to \mathrm{Set}(A) \rangle$$

Back To Approximation

- $[\![\cdot]\!]_{\mathbf{pre}} : p \to (\Omega \overset{\sim}{\underset{\mathbf{pre}}{\to}} B)$ returns uncomputable functions
- Recall:

$$A \overset{\sim}{\underset{\mathbf{pre}}{\to}} B ::= \mathrm{Set}(A) \to \langle \mathrm{Set}(B), \mathrm{Set}(B) \to \mathrm{Set}(A) \rangle$$

- Define:

$$A \overset{\sim}{\underset{\mathbf{pre}}{\to}}{}' B ::= \mathrm{Rect}(A) \to \langle \mathrm{Rect}(B), \mathrm{Rect}(B) \to \mathrm{Rect}(A) \rangle$$

Back To Approximation

- $[\![\cdot]\!]_{\mathbf{pre}} : p \to (\Omega \underset{\mathbf{pre}}{\rightsquigarrow} B)$ returns uncomputable functions
- Recall:

$$A \underset{\mathbf{pre}}{\rightsquigarrow} B ::= \mathrm{Set}(A) \to \langle \mathrm{Set}(B), \mathrm{Set}(B) \to \mathrm{Set}(A) \rangle$$

- Define:

$$A \underset{\mathbf{pre}}{\rightsquigarrow}' B ::= \mathrm{Rect}(A) \to \langle \mathrm{Rect}(B), \mathrm{Rect}(B) \to \mathrm{Rect}(A) \rangle$$

as well as Rect intersection, join (union-like), empty test, products, etc.

Back To Approximation

- $[\![\cdot]\!]_{\mathbf{pre}} : p \to (\Omega \underset{\mathbf{pre}}{\rightsquigarrow} B)$ returns uncomputable functions
- Recall:

$$A \underset{\mathbf{pre}}{\rightsquigarrow} B ::= \mathrm{Set}(A) \to \langle \mathrm{Set}(B), \mathrm{Set}(B) \to \mathrm{Set}(A) \rangle$$

- Define:

$$A \underset{\mathbf{pre}}{\rightsquigarrow}' B ::= \mathrm{Rect}(A) \to \langle \mathrm{Rect}(B), \mathrm{Rect}(B) \to \mathrm{Rect}(A) \rangle$$

as well as Rect intersection, join (union-like), empty test, products, etc.

- Derive $[\![\cdot]\!]'_{\mathbf{pre}} : p \to (\Omega \underset{\mathbf{pre}}{\rightsquigarrow}' B)$

In Theory...

Theorem (sound). $[\![\cdot]\!]'_{\mathrm{pre}}$ computes overapproximations of the preimages computed by $[\![\cdot]\!]_{\mathrm{pre}}$.

- Consequence: Sampling within preimages doesn't leave anything out

In Theory...

Theorem (sound). $[\![\cdot]\!]'_{\mathrm{pre}}$ computes overapproximations of the preimages computed by $[\![\cdot]\!]_{\mathrm{pre}}$.

- Consequence: Sampling within preimages doesn't leave anything out

Theorem (monotone). $[\![\cdot]\!]'_{\mathrm{pre}}$ is monotone.

- Consequence: Partitioning the domain never increases approximate preimages

In Theory...

Theorem (sound). $[\![\cdot]\!]'_{\mathrm{pre}}$ computes overapproximations of the preimages computed by $[\![\cdot]\!]_{\mathrm{pre}}$.

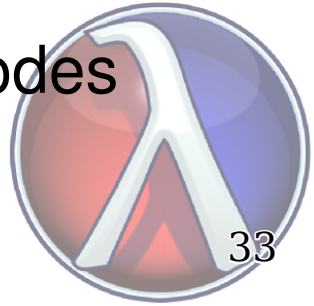
- Consequence: Sampling within preimages doesn't leave anything out

Theorem (monotone). $[\![\cdot]\!]'_{\mathrm{pre}}$ is monotone.

- Consequence: Partitioning the domain never increases approximate preimages

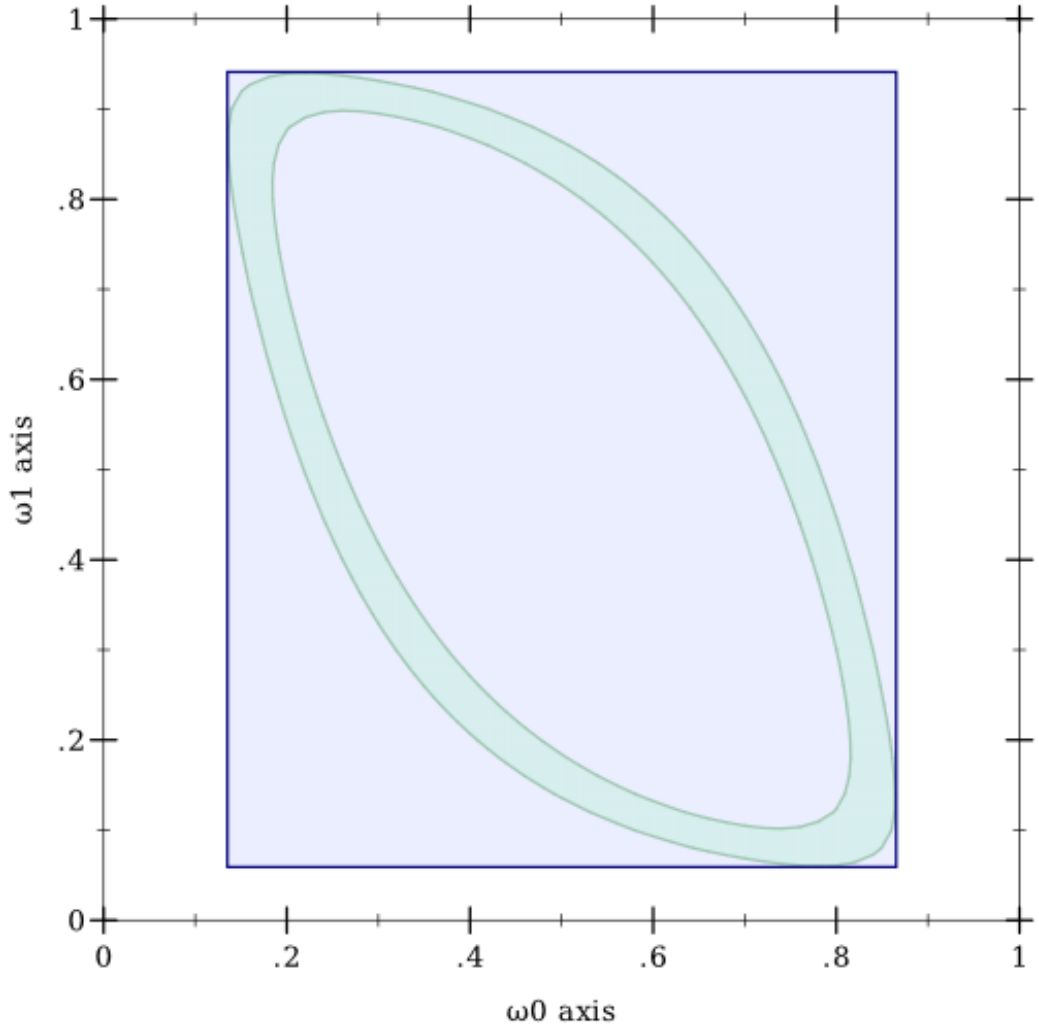
Theorem (decreasing). $[\![\cdot]\!]'_{\mathrm{pre}}$ never returns preimages larger than the given subdomain.

- Consequence: Refining preimage partitions never explodes



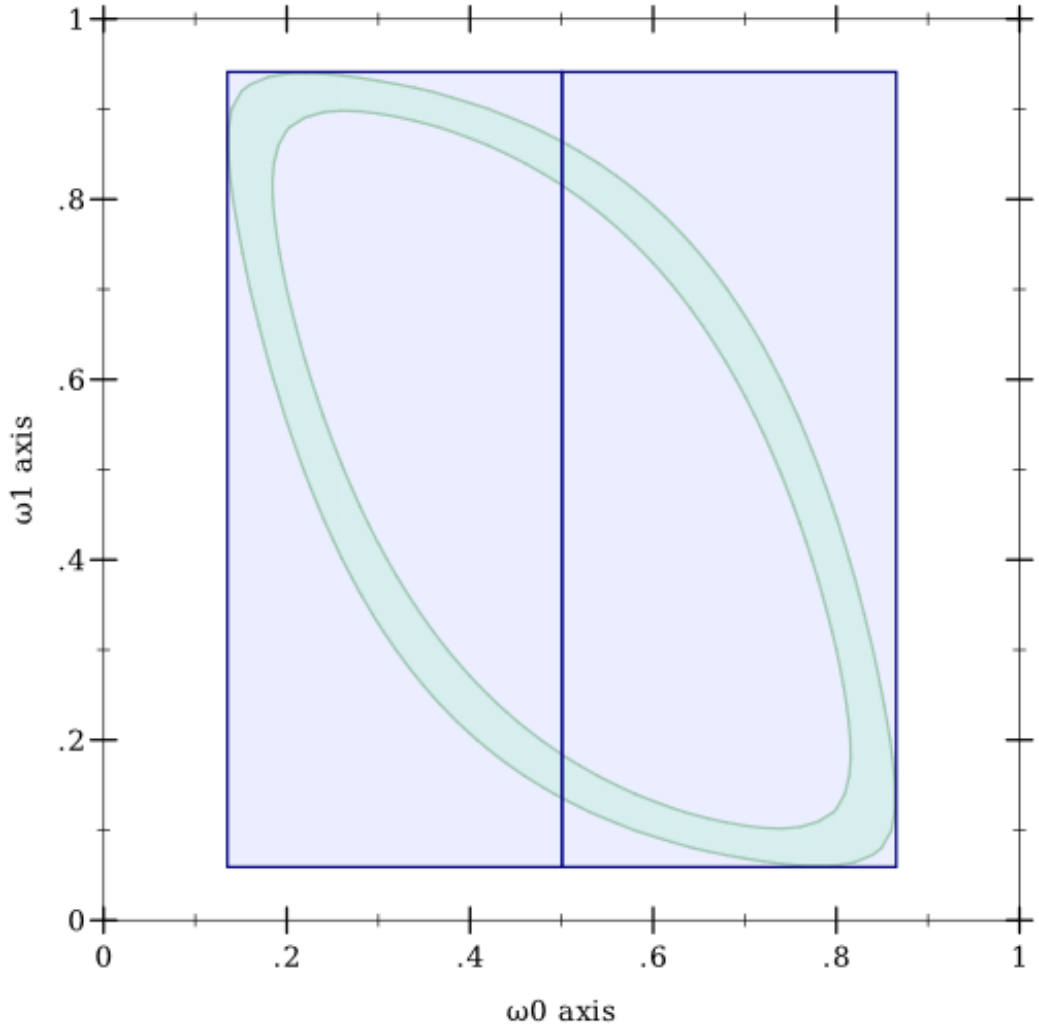
In Practice...

Theorems prove this always works:



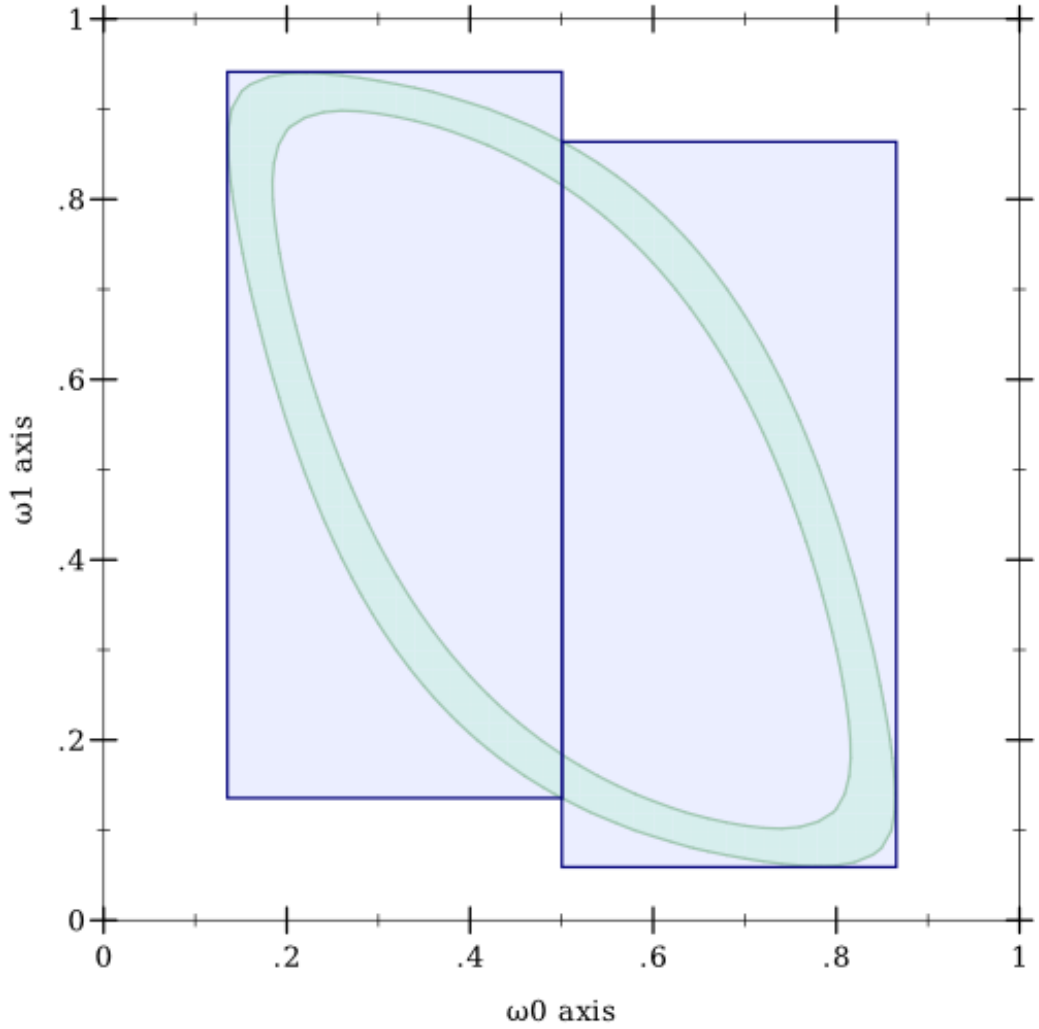
In Practice...

Theorems prove this always works:



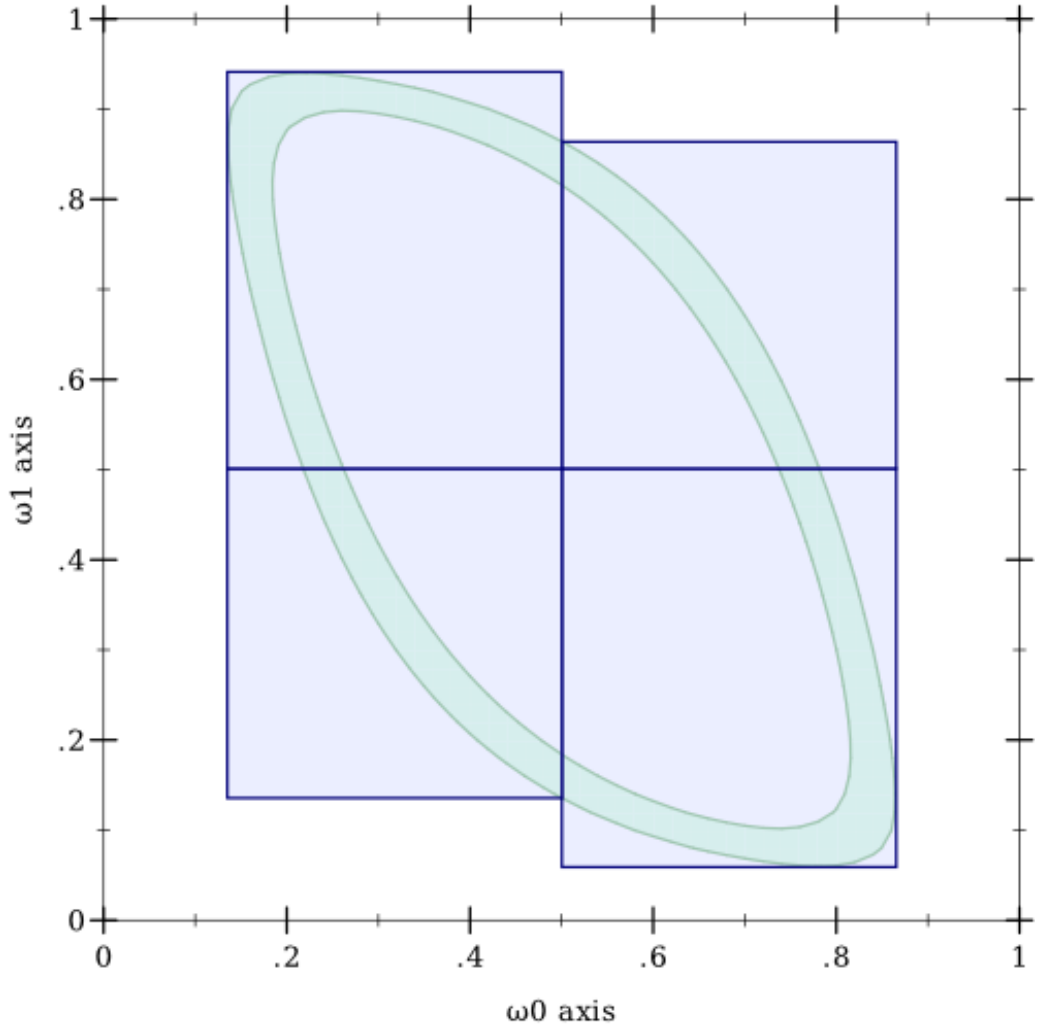
In Practice...

Theorems prove this always works:



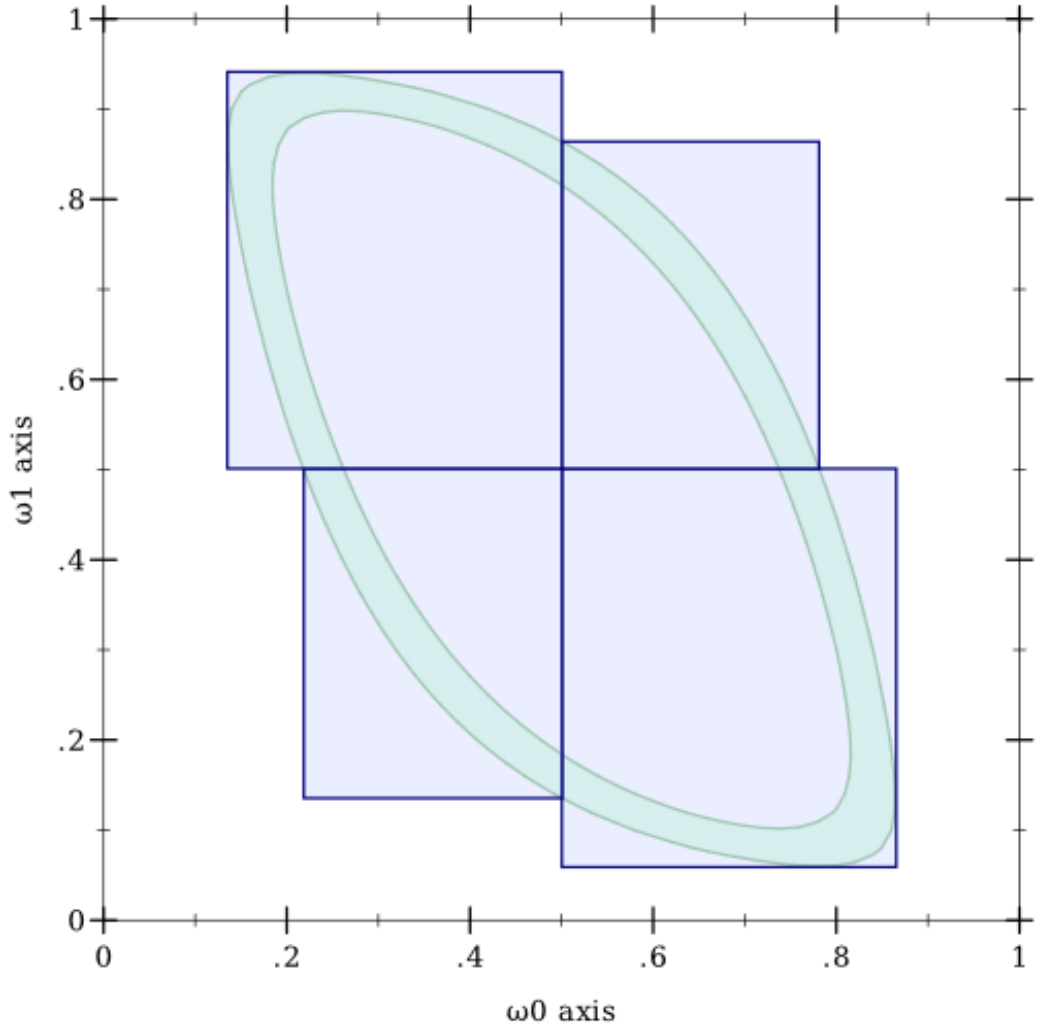
In Practice...

Theorems prove this always works:



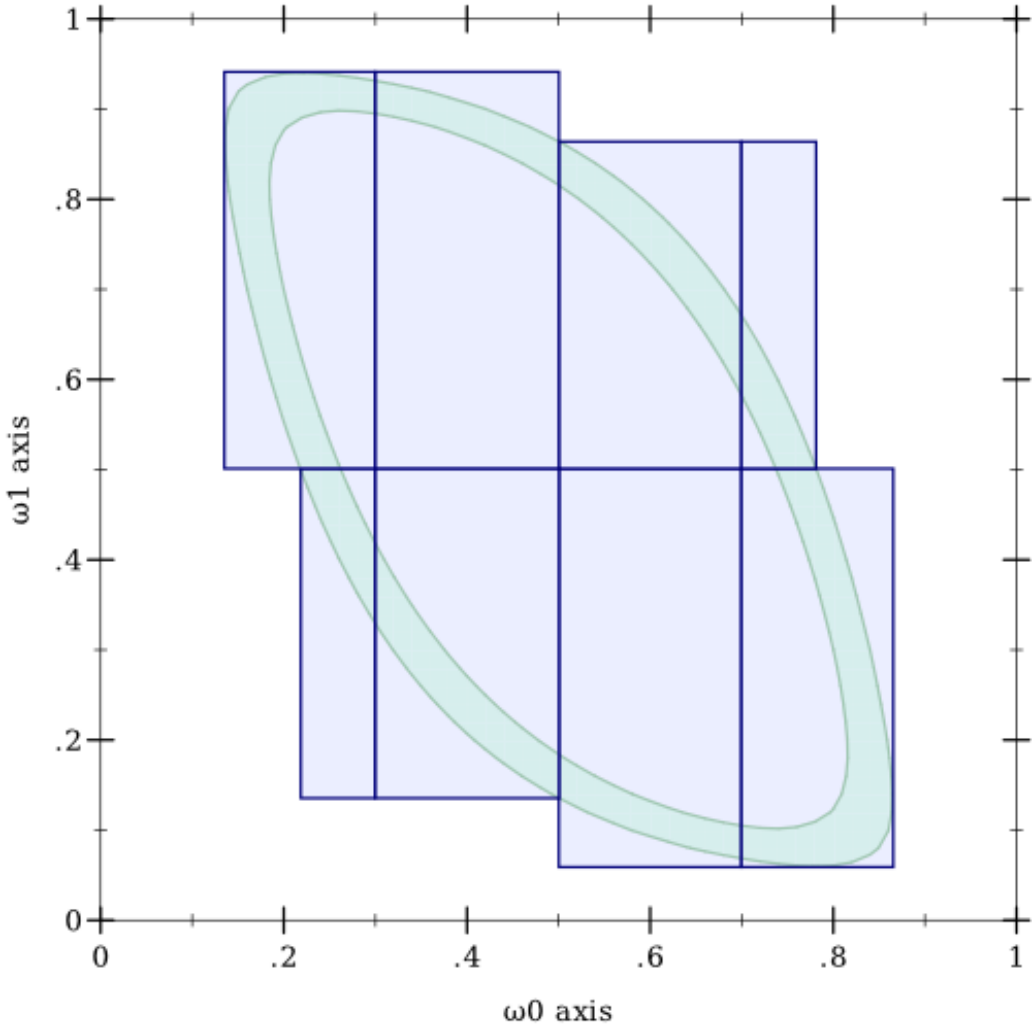
In Practice...

Theorems prove this always works:



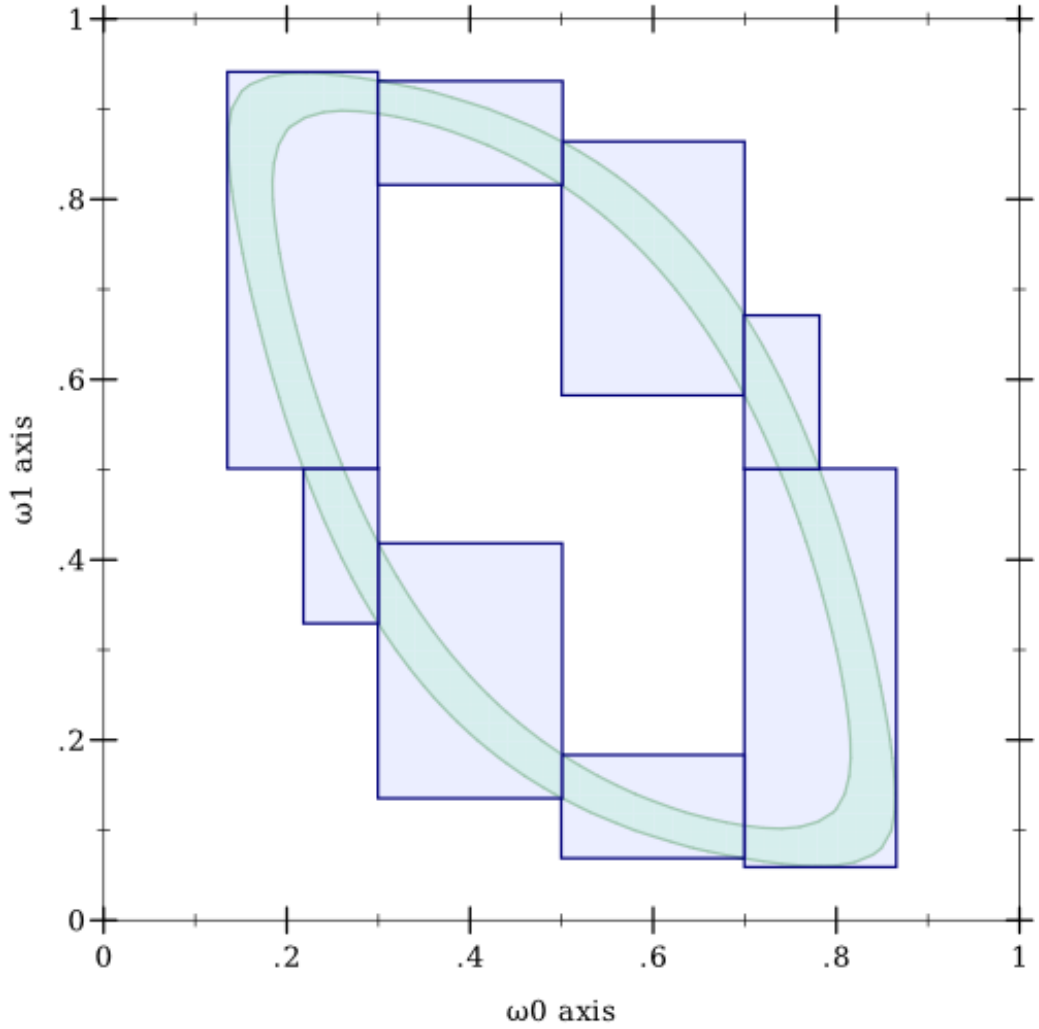
In Practice...

Theorems prove this always works:



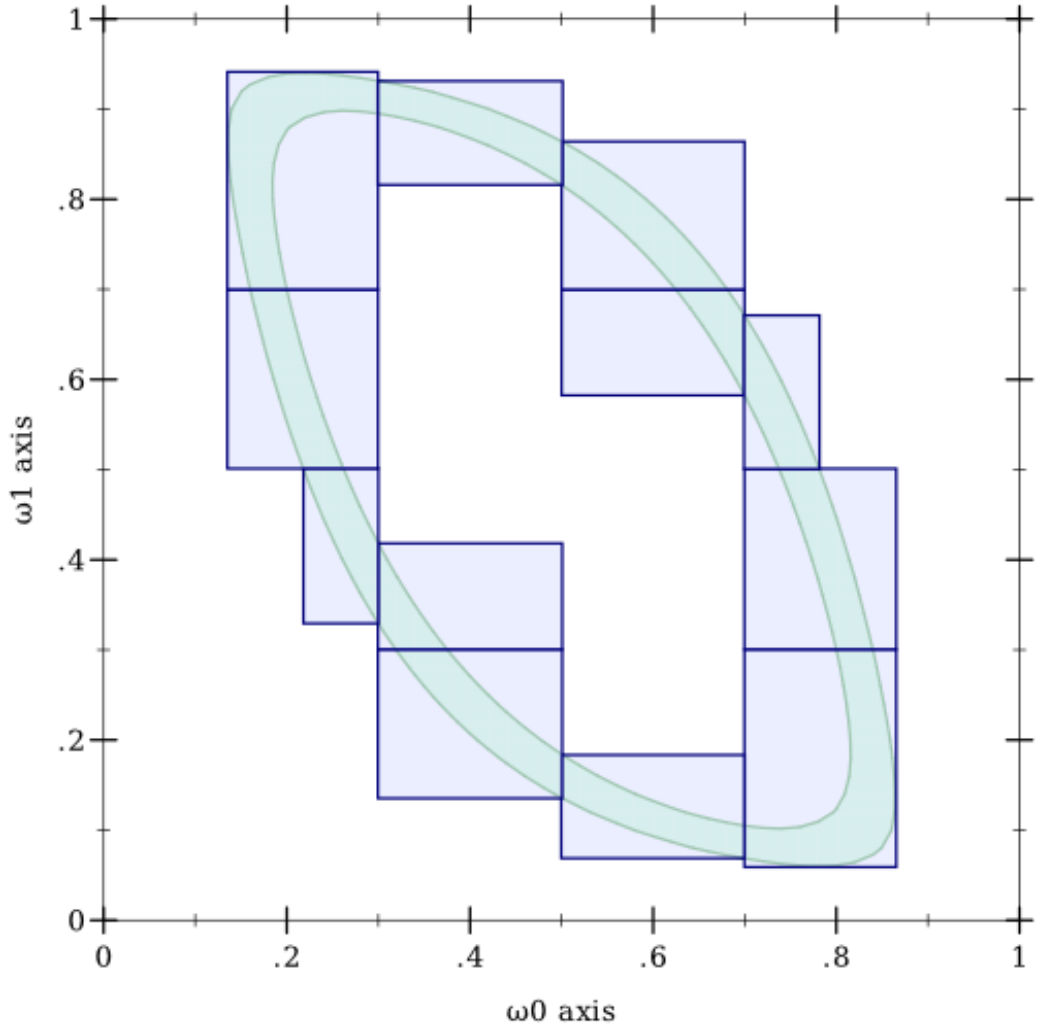
In Practice...

Theorems prove this always works:



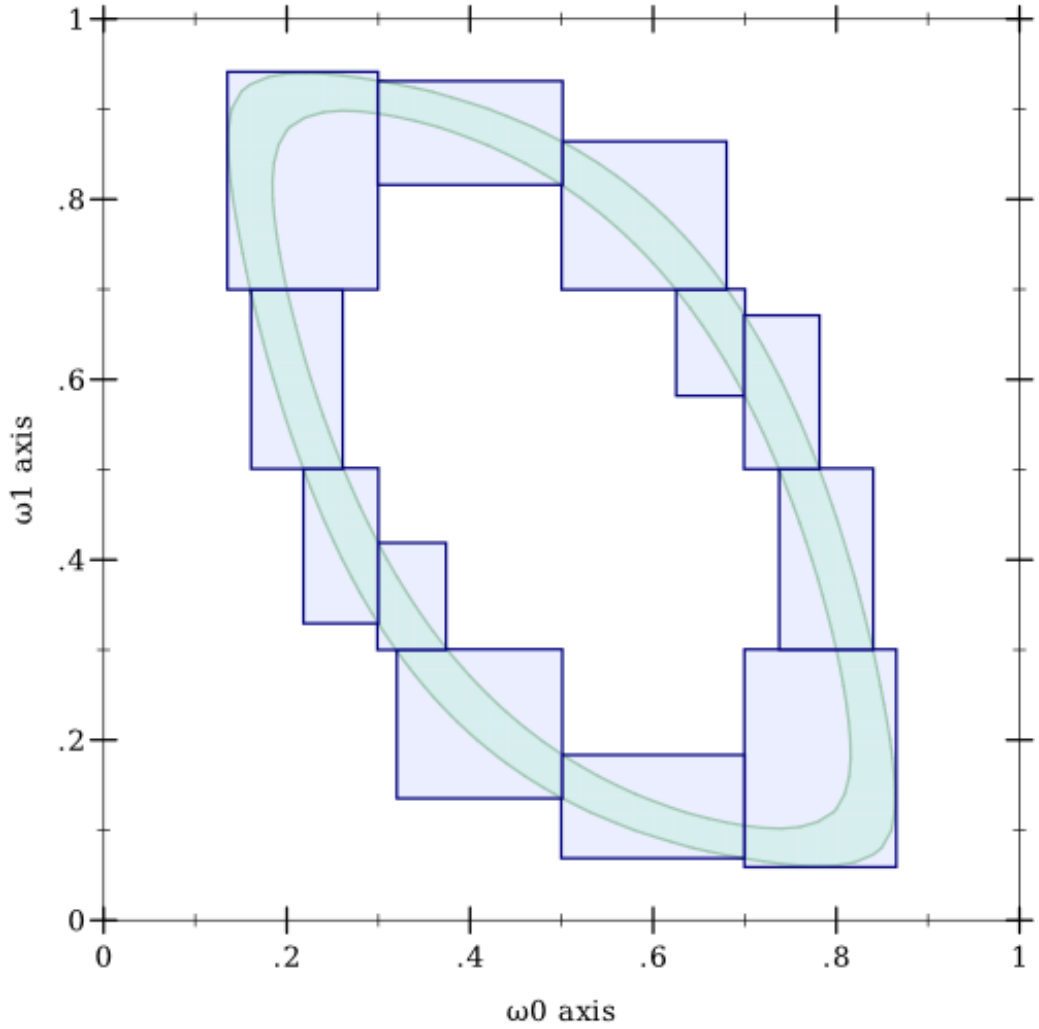
In Practice...

Theorems prove this always works:



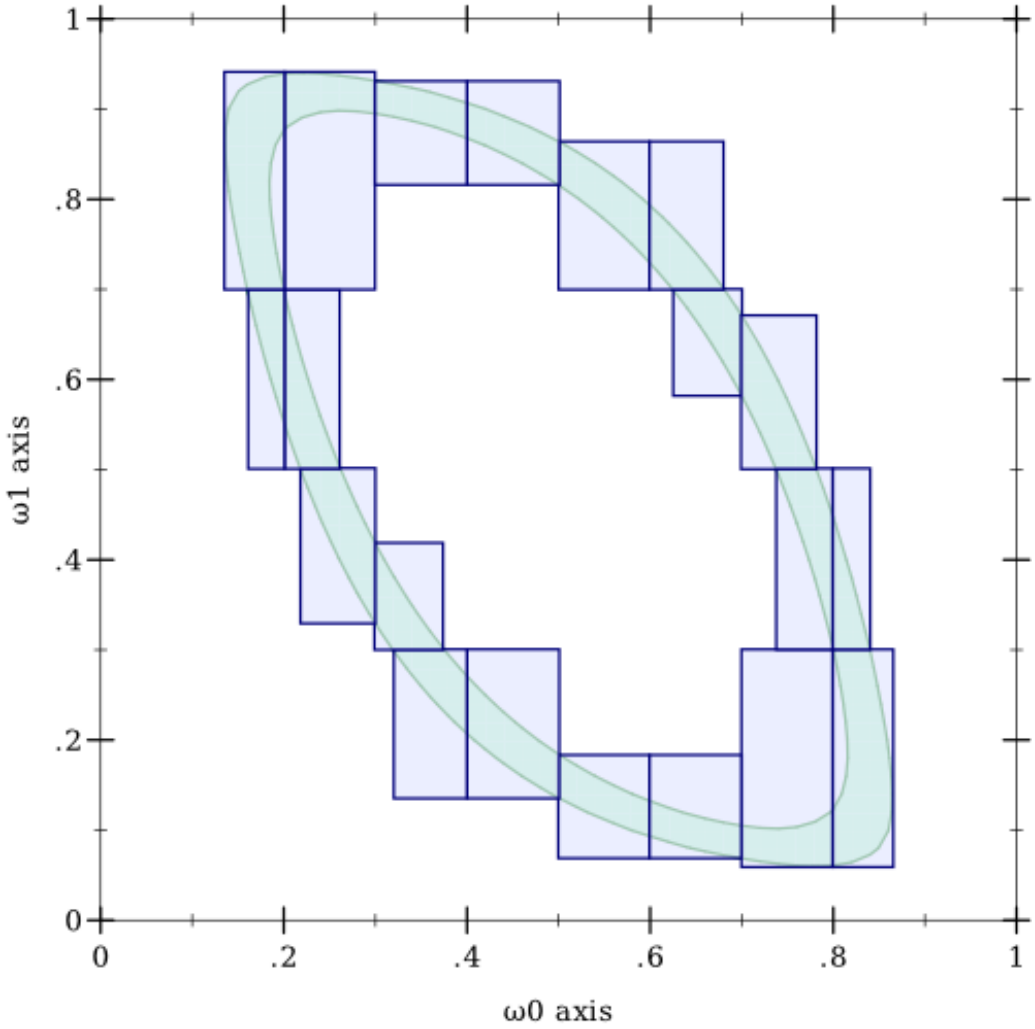
In Practice...

Theorems prove this always works:



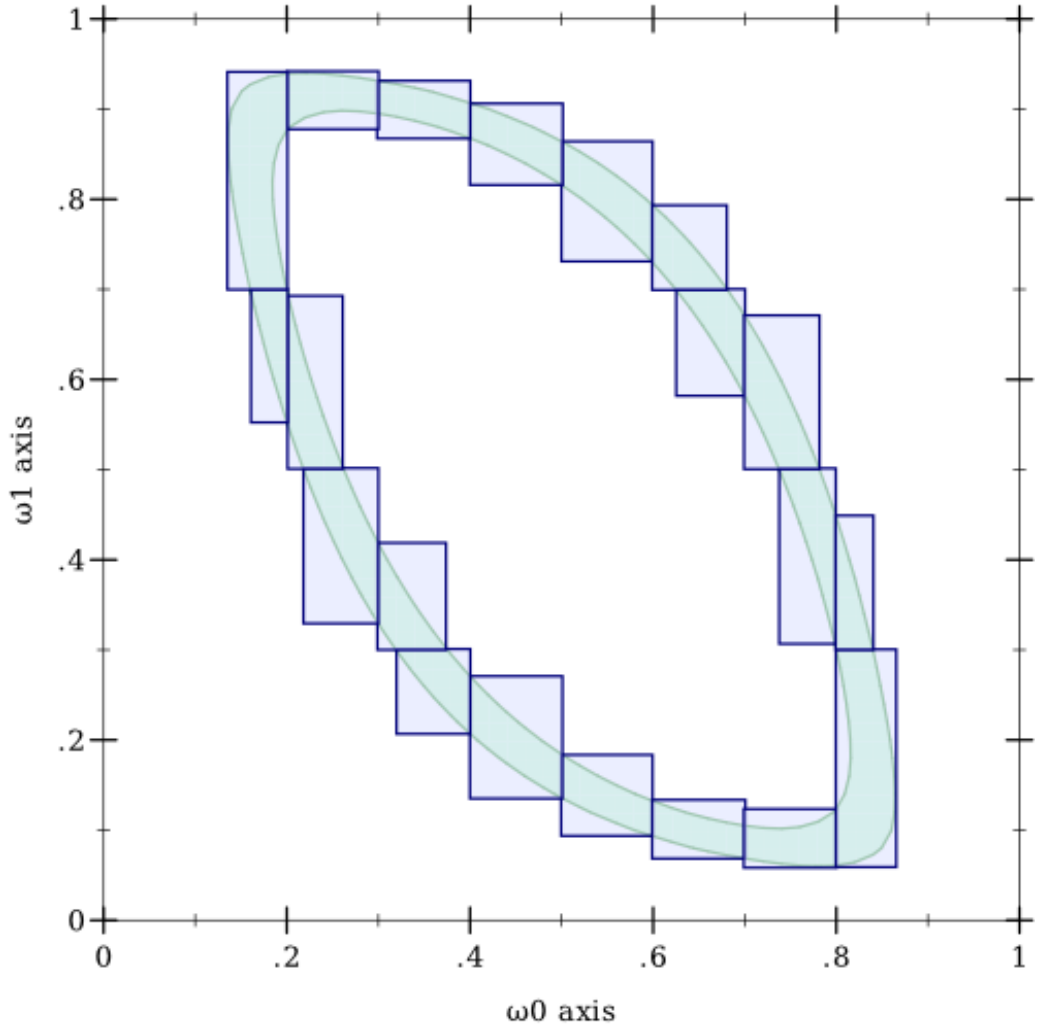
In Practice...

Theorems prove this always works:



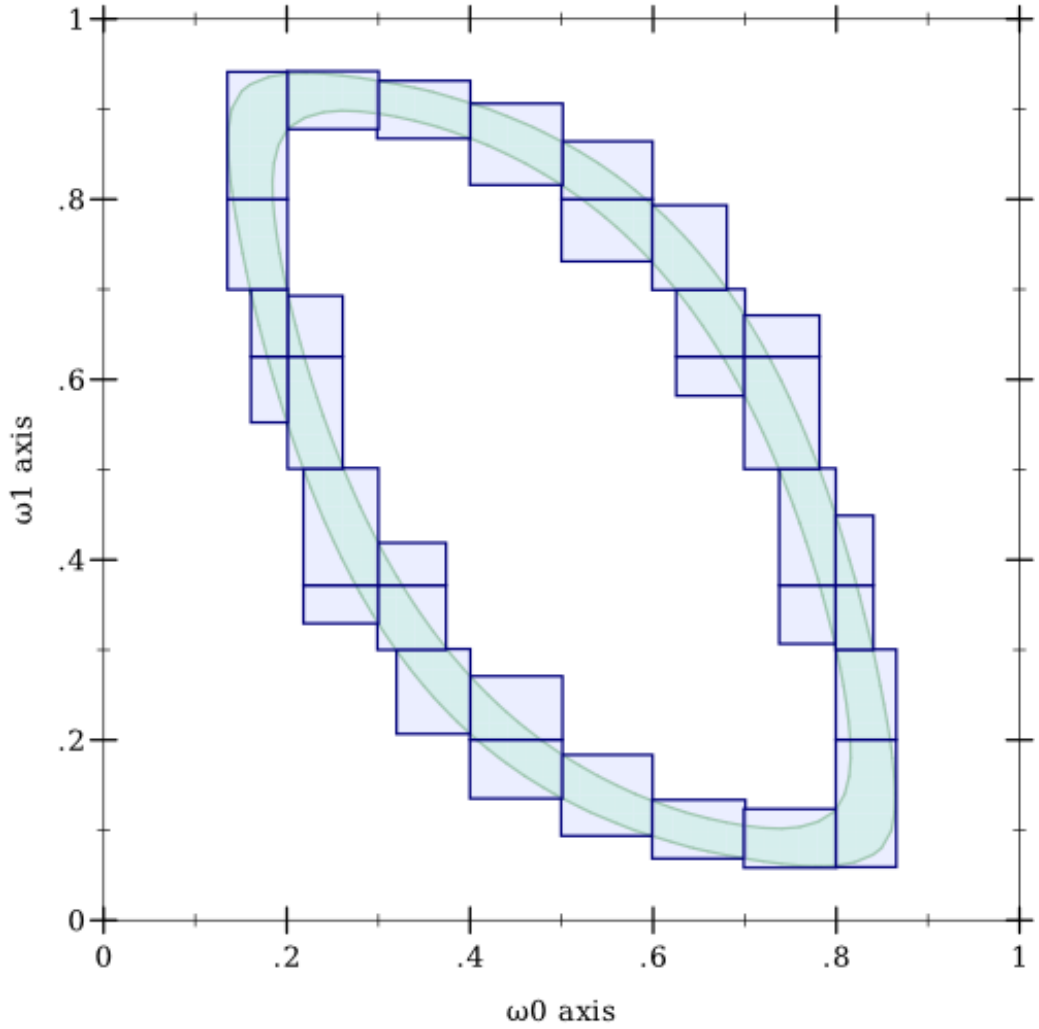
In Practice...

Theorems prove this always works:



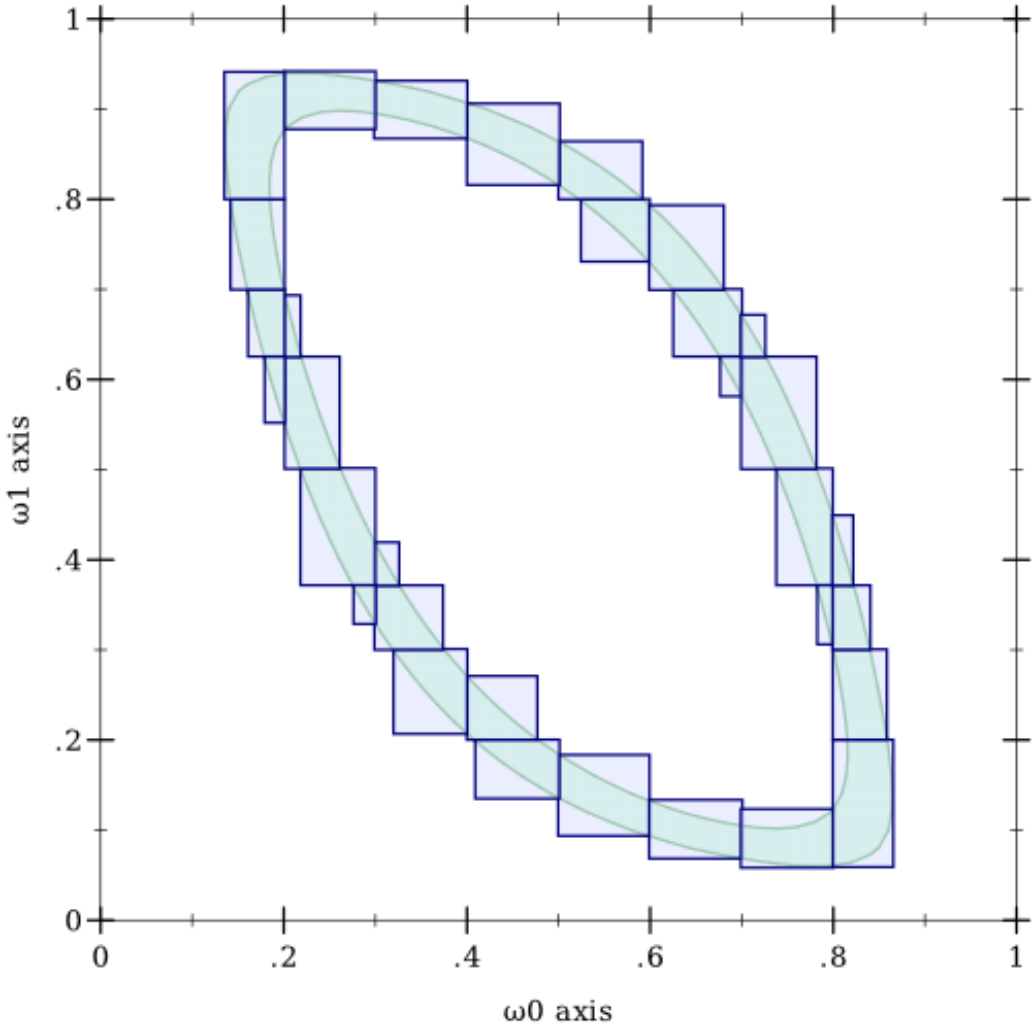
In Practice...

Theorems prove this always works:



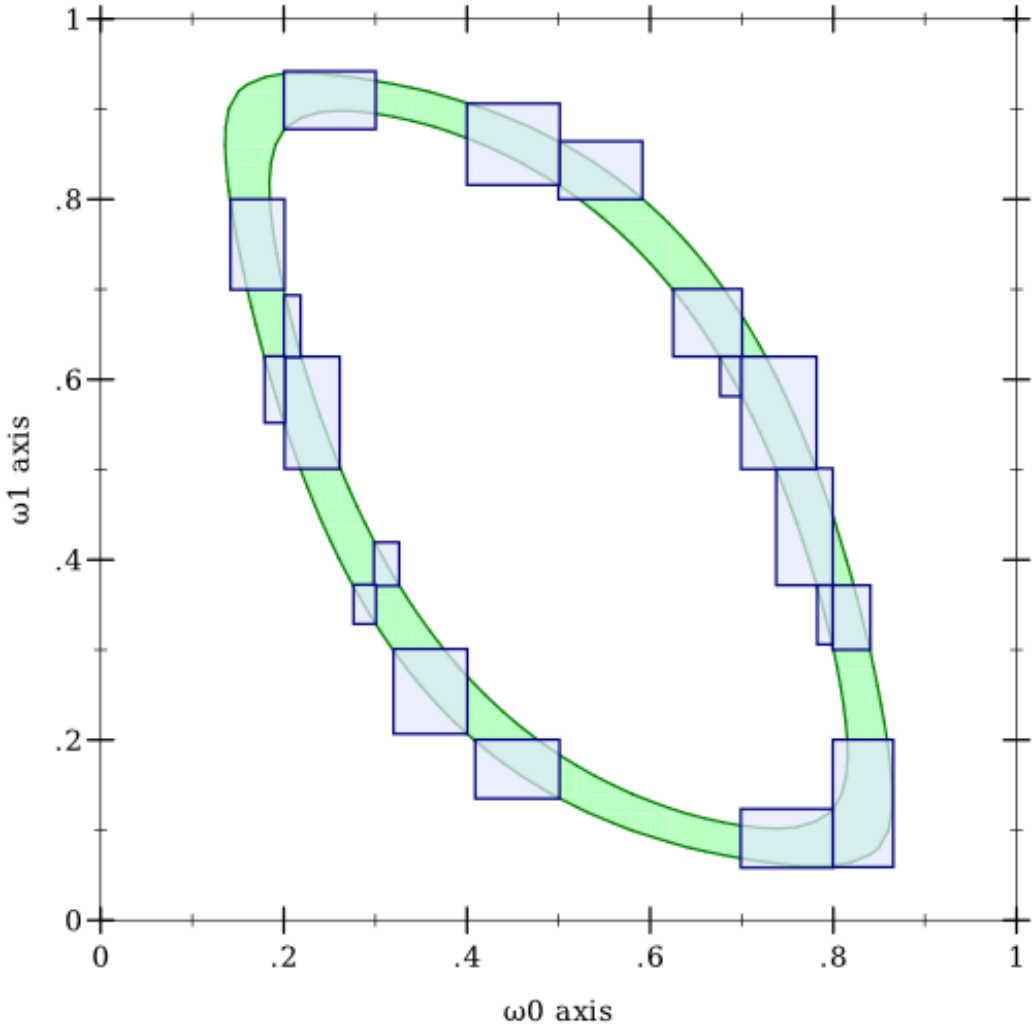
In Practice...

Theorems prove this always works:



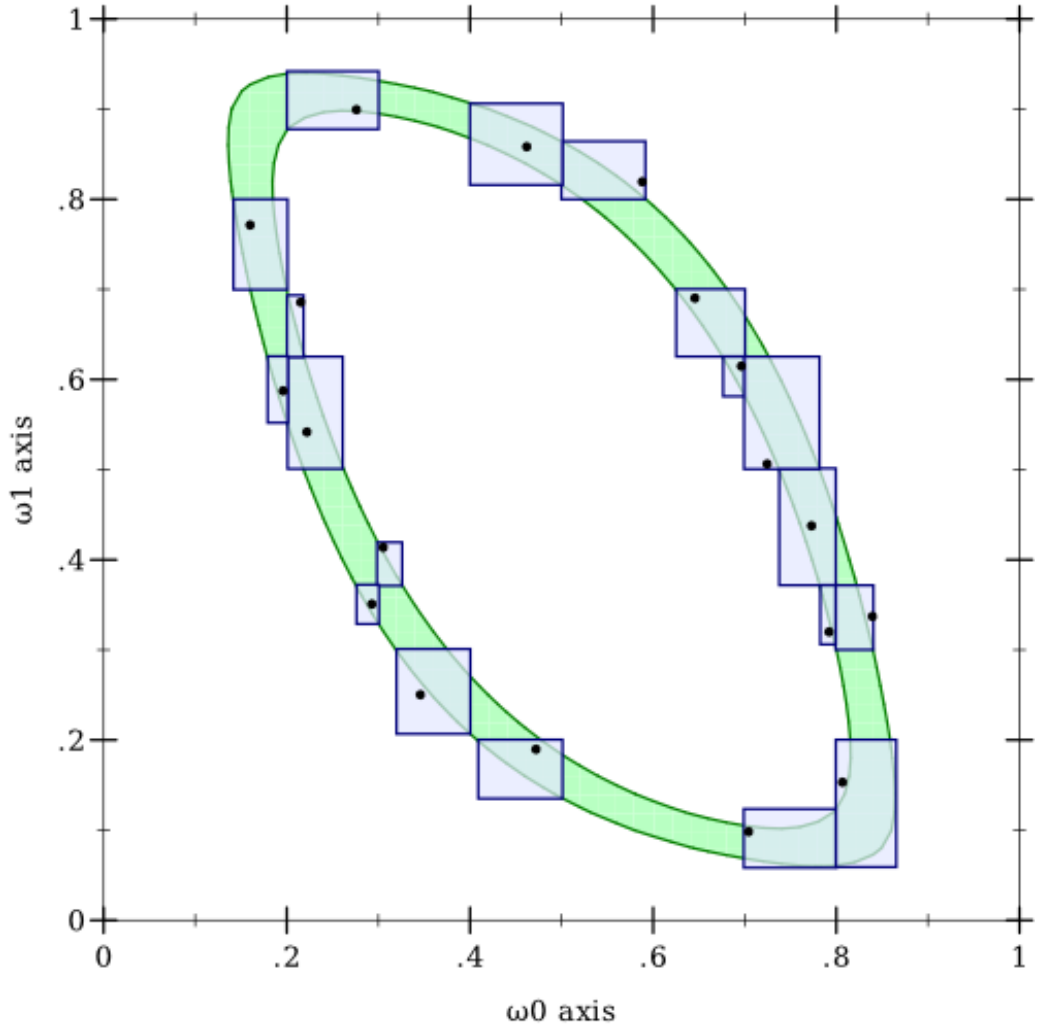
In Practice...

Theorems prove this always works:



In Practice...

Theorems prove this always works:



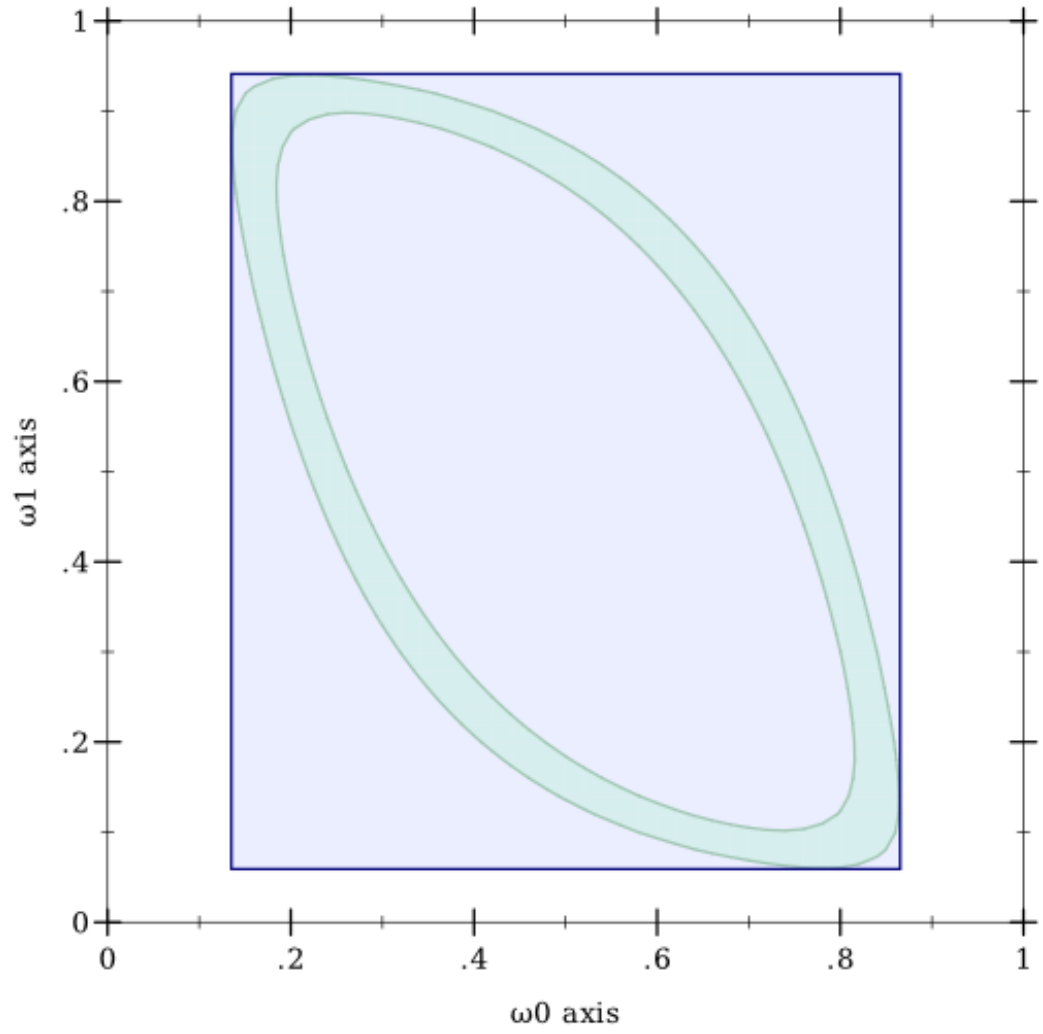
Algorithm 2

- Alternative to arbitrarily low-rate rejection sampling:

Algorithm 2

- Alternative to arbitrarily low-rate rejection sampling:

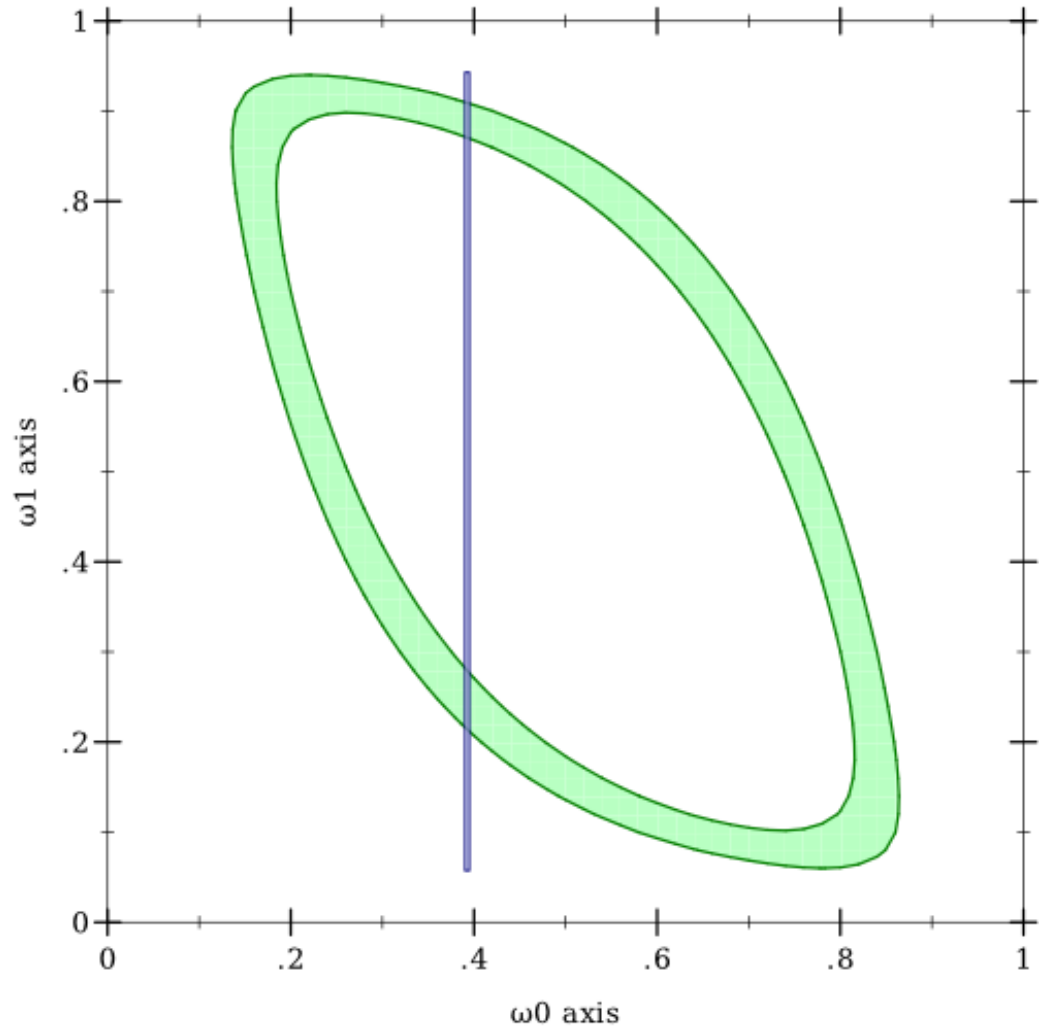
First, refine using preimage computation:



Algorithm 2

- Alternative to arbitrarily low-rate rejection sampling:

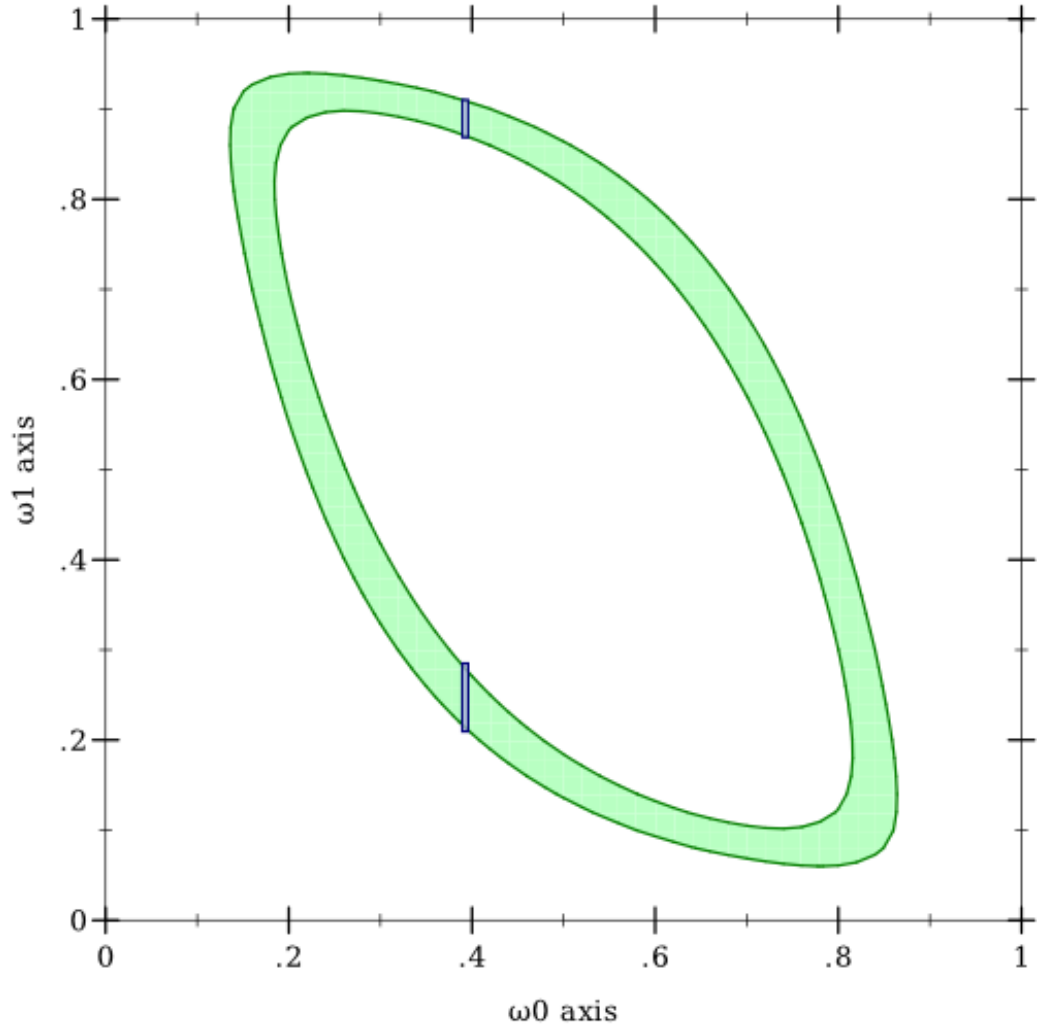
Second, randomly choose from arbitrarily fine partition:



Algorithm 2

- Alternative to arbitrarily low-rate rejection sampling:

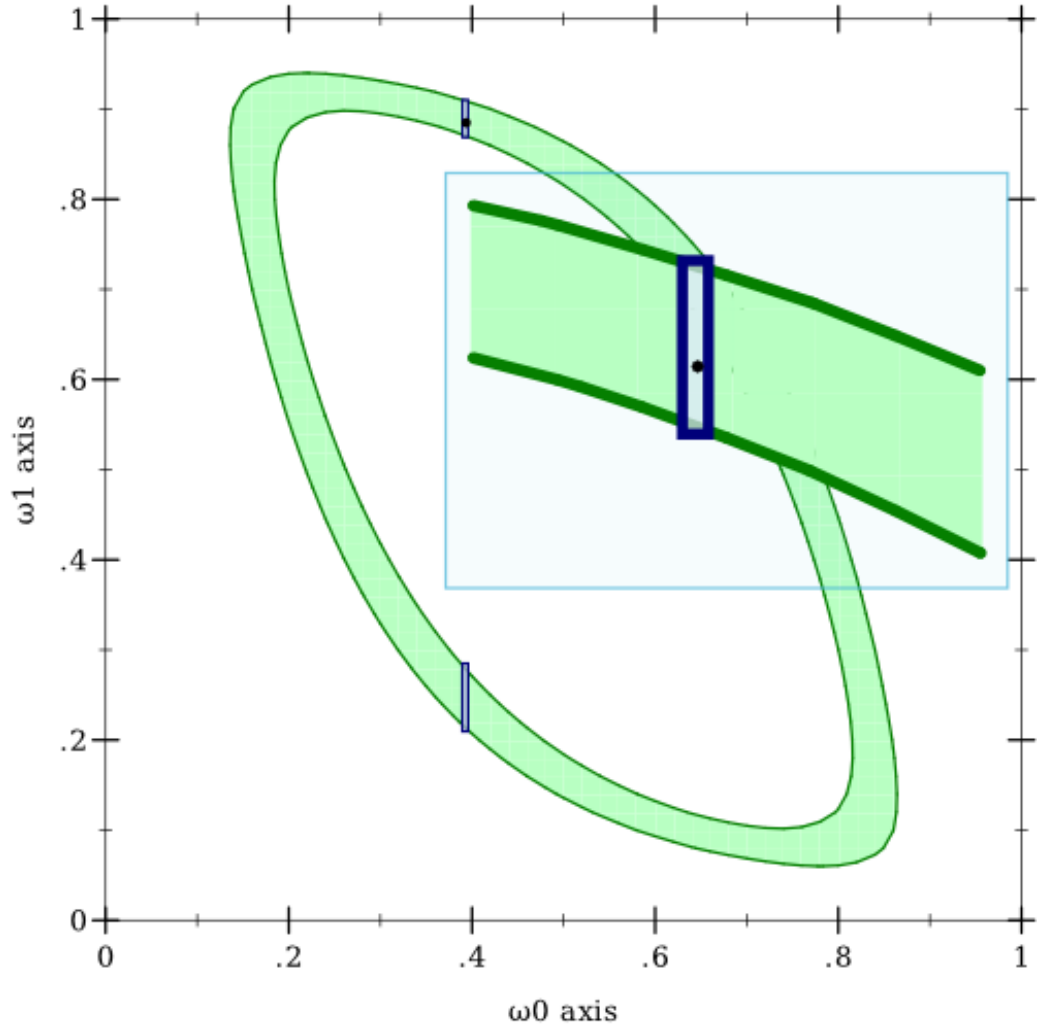
Third, refine again:



Algorithm 2

- Alternative to arbitrarily low-rate rejection sampling:

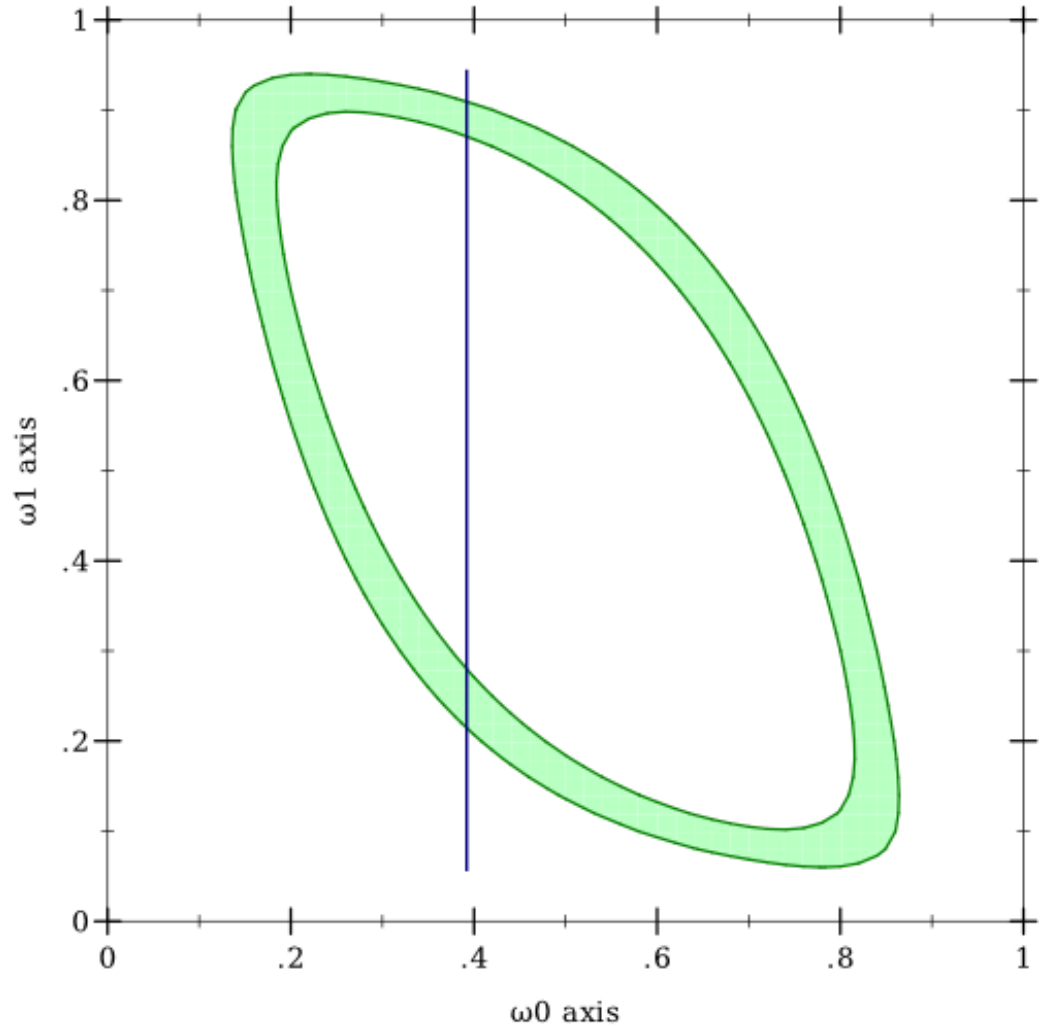
Fourth, sample uniformly:



Algorithm 2

- Alternative to arbitrarily low-rate rejection sampling:

Do process “in the limit”; i.e. choose $[\omega_0, \omega_0] \times \Omega_1$:



Likelihood Weighting?

- Each sample must be weighted by the inverse of probability it was chosen (i.e. sampling algorithm is an importance distribution)

Likelihood Weighting?

- Each sample must be weighted by the inverse of probability it was chosen (i.e. sampling algorithm is an importance distribution)
- Conjecture: Algorithm 2 approaches likelihood weighting (with densities for absolutely continuous distributions) as $\epsilon \to 0$

Likelihood Weighting?

- Each sample must be weighted by the inverse of probability it was chosen (i.e. sampling algorithm is an importance distribution)
- Conjecture: Algorithm 2 approaches likelihood weighting (with densities for absolutely continuous distributions) as $\epsilon \to 0$
- Do better than LW by combining Algorithm 1 and Algorithm 2

Floating-Point Sensitivity and Accuracy

- Algorithm 2 is *extremely* sensitive to floating-point error

Floating-Point Sensitivity and Accuracy

- Algorithm 2 is *extremely* sensitive to floating-point error
- Requires either perfectly outwardly rounded interval arithmetic for monotone functions:

```
> (ivl (exp- 1.1) (exp+ 1.1))
(ivl 3.004166023946433 3.004166023946434)
> (flonums-between 3.004166023946433 3.004166023946434)
1
```

Floating-Point Sensitivity and Accuracy

- Algorithm 2 is *extremely* sensitive to floating-point error
- Requires either perfectly outwardly rounded interval arithmetic for monotone functions:

```
> (ivl (exp- 1.1) (exp+ 1.1))
(ivl 3.004166023946433 3.004166023946434)
> (flonums-between 3.004166023946433 3.004166023946434)
1
```

- Or a bound on its implementations' error to overapproximate upward- and downward-rounded implementations:

```
> (ivl (norm-inv-cdf- 0.8) (norm-inv-cdf+ 0.8))
(ivl 0.8416212335729142 0.8416212335729149)
> (flonums-between 0.8416212335729142 0.8416212335729149)
8
```

What About Variable-Dimension Models?

- General recursion, programs that halt with probability 1; e.g.

```
(define/drbayes (geometric p)
  (if (bernoulli p) 0 (+ 1 (geometric p))))
```

What About Variable-Dimension Models?

- General recursion, programs that halt with probability 1; e.g.

```
(define/drbayes (geometric p)
  (if (bernoulli p) 0 (+ 1 (geometric p))))
```

- I lied about program domain: it's $\Omega \times T$

What About Variable-Dimension Models?

- General recursion, programs that halt with probability 1; e.g.

```
(define/drbayes (geometric p)
  (if (bernoulli p) 0 (+ 1 (geometric p))))
```

- I lied about program domain: it's $\Omega \times T$
- Every **branch trace** $t \in T$ determines which branch each `if` expression takes in an *infinite, fully inlined* program

```
(if (bernoulli p)
    0
    (+ 1 (if (bernoulli p)
             0
             (+ 1 ...)))))
```

What About Variable-Dimension Models?

- General recursion, programs that halt with probability 1; e.g.

```
(define/drbayes (geometric p)
  (if (bernoulli p) 0 (+ 1 (geometric p))))
```

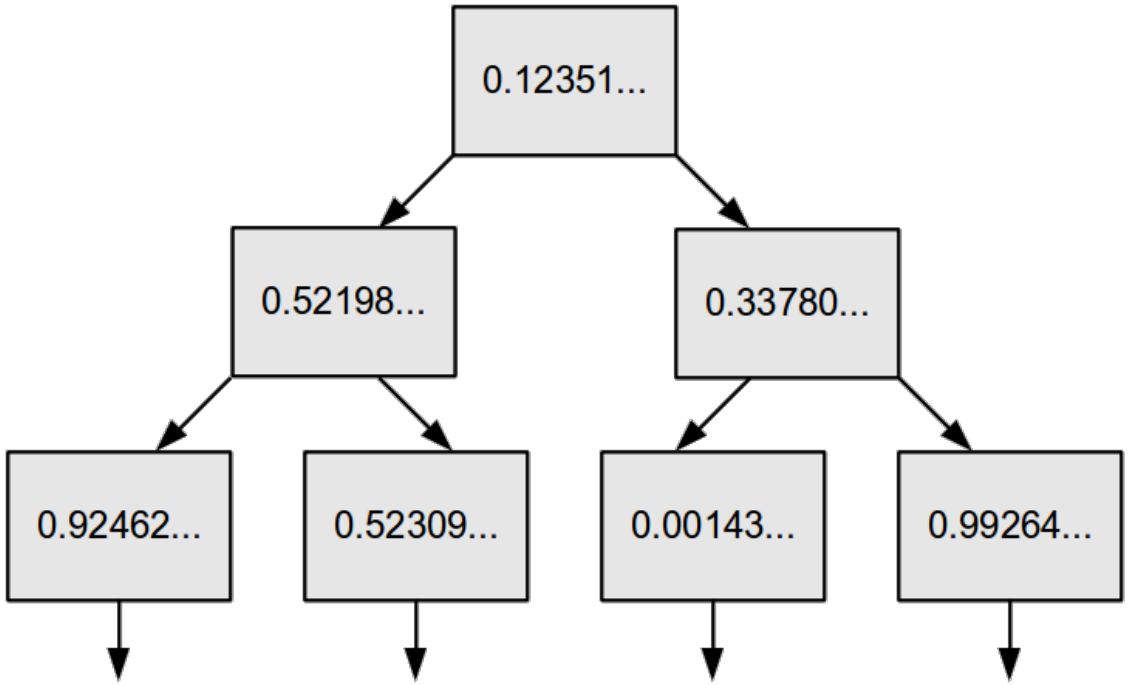
- I lied about program domain: it's $\Omega \times T$
- Every **branch trace** $t \in T$ determines which branch each `if` expression takes in an *infinite, fully inlined* program

```
(if (bernoulli p)
    0
    (+ 1 (if (bernoulli p)
             0
             (+ 1 ...)))))
```

- Computing preimages often decides branching; it's randomly forced otherwise

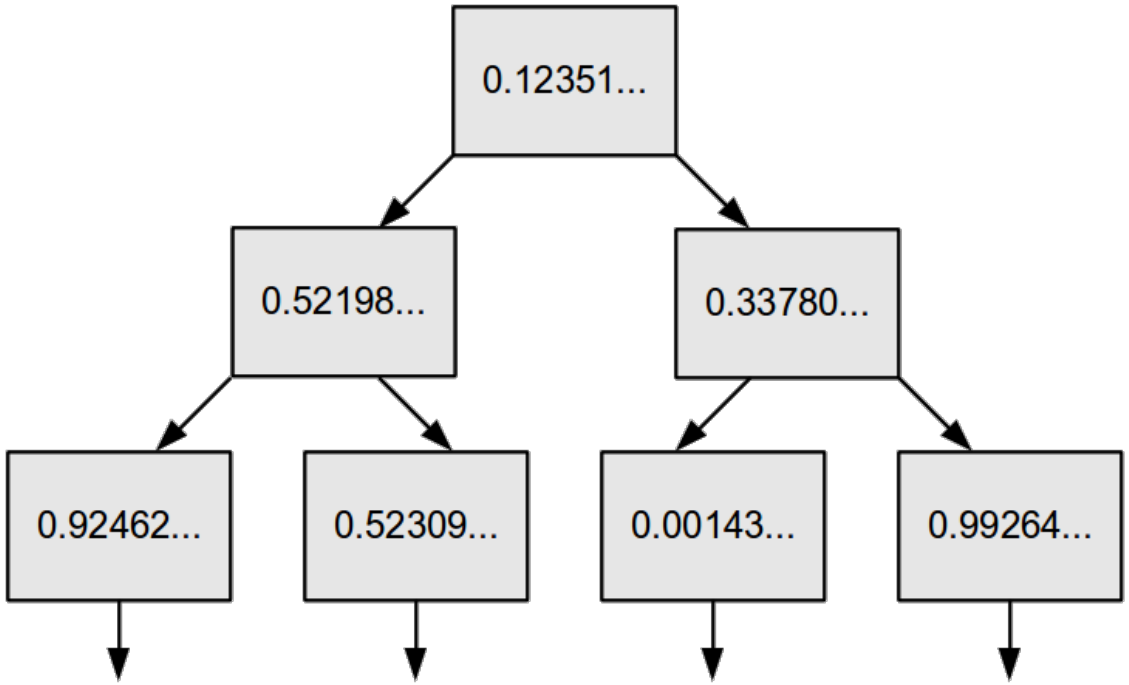
Domain Values

- Values $\omega \in \Omega$ are infinite binary trees:



Domain Values

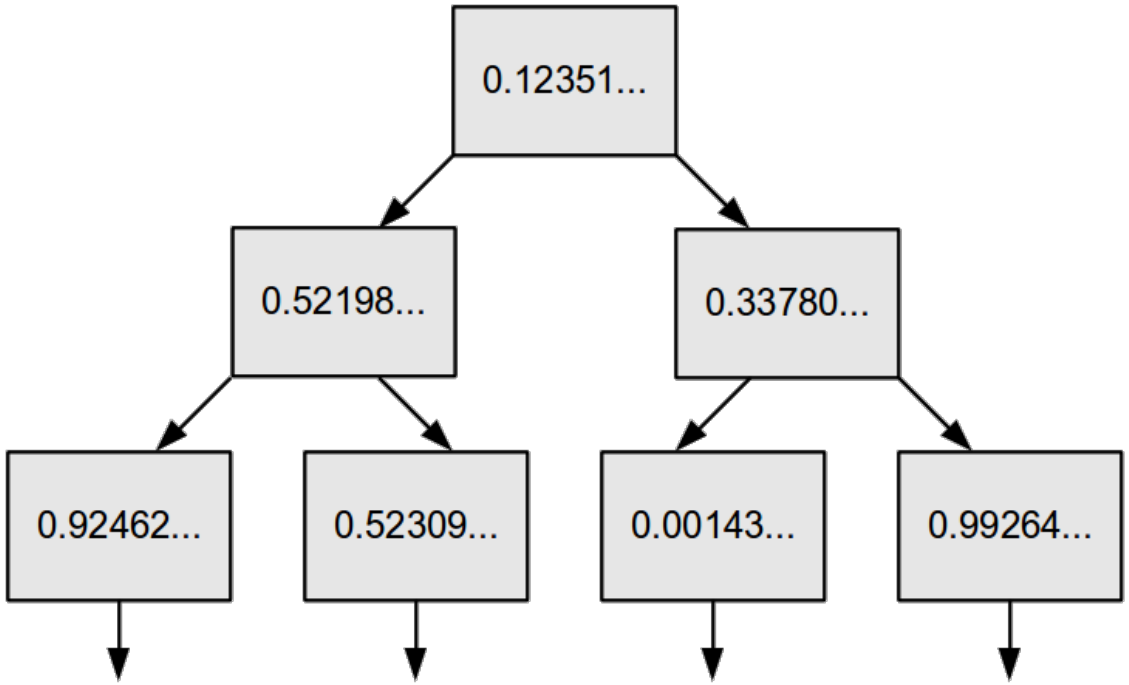
- Values $\omega \in \Omega$ are infinite binary trees:



- Implemented using lazy trees of random values

Domain Values

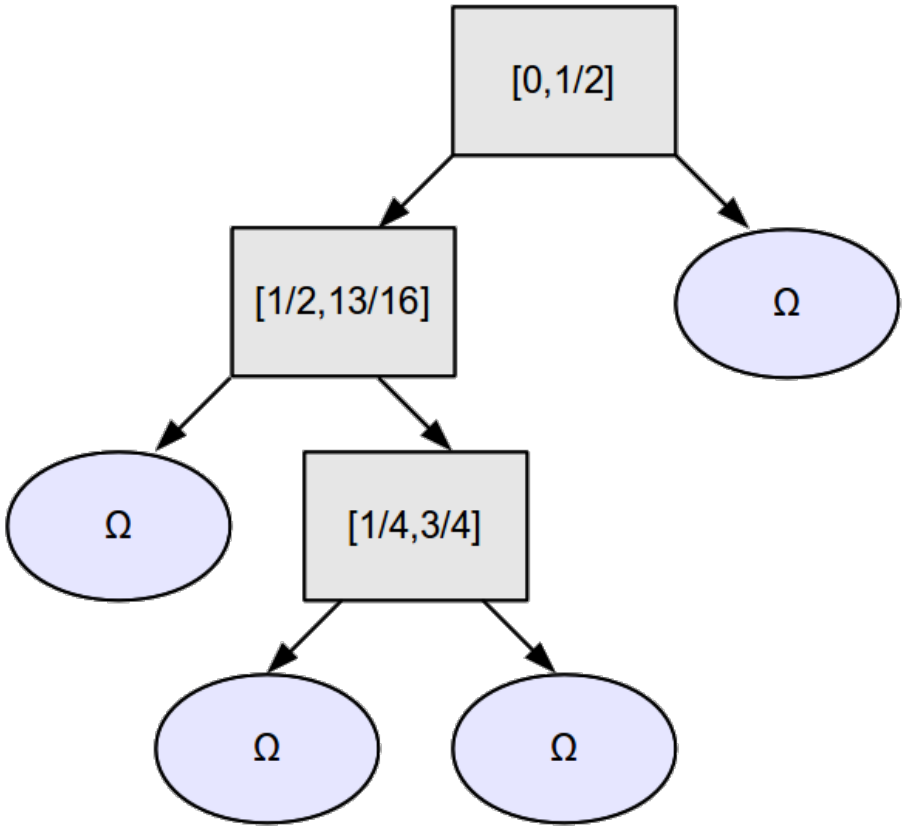
- Values $\omega \in \Omega$ are infinite binary trees:



- Implemented using lazy trees of random values
- Each $t \in T$ is an infinite tree of branch choices

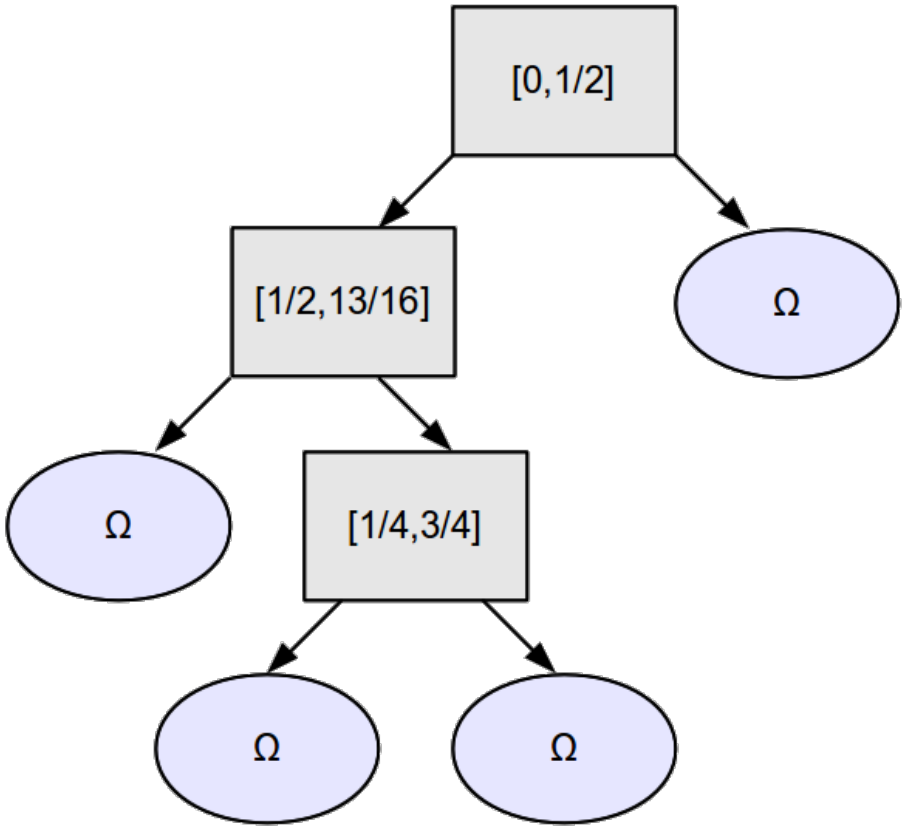
Domain Rectangles

- Rectangles $\Omega' \subseteq \Omega$ are products of tree axes:



Domain Rectangles

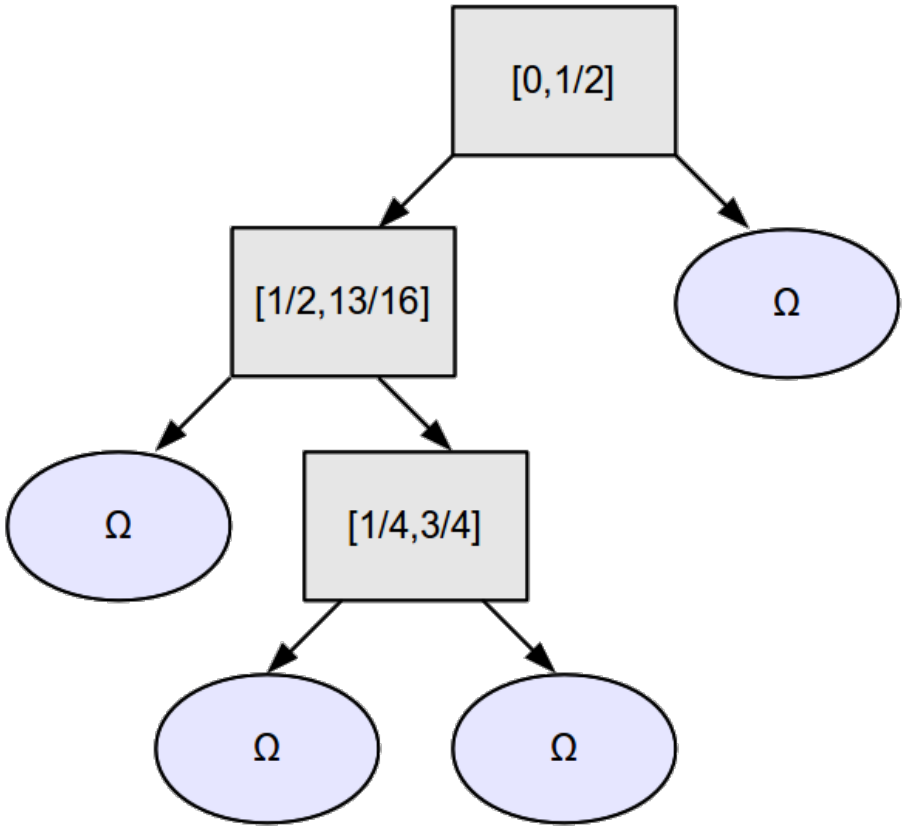
- Rectangles $\Omega' \subseteq \Omega$ are products of tree axes:



- Implemented using finite trees of real sets

Domain Rectangles

- Rectangles $\Omega' \subseteq \Omega$ are products of tree axes:



- Implemented using finite trees of real sets
- Rectangles $T' \subseteq T$ are similar

Solution Summary (1)

- Standard interpretation $[\![\cdot]\!]$
  - Directly implementable (with floating-point approximation)

Solution Summary (1)

- Standard interpretation $[\![\cdot]\!]$
  - Directly implementable (with floating-point approximation)
- Nonstandard interpretation $[\![\cdot]\!]_{\mathbf{pre}}$ for computing preimages
  - Unimplementable (except in λ_{ZFC})

Solution Summary (1)

- Standard interpretation $[\![\cdot]\!]$
  - Directly implementable (with floating-point approximation)
- Nonstandard interpretation $[\![\cdot]\!]_{\mathbf{pre}}$ for computing preimages
  - Unimplementable (except in λ_{ZFC})
  - Proof of correctness w.r.t. standard interpretation

Solution Summary (1)

- Standard interpretation $[\![\cdot]\!]$
  - Directly implementable (with floating-point approximation)
- Nonstandard interpretation $[\![\cdot]\!]_{\mathbf{pre}}$ for computing preimages
  - Unimplementable (except in λ_{ZFC})
  - Proof of correctness w.r.t. standard interpretation
- Approximating interpretation $[\![\cdot]\!]'_{\mathbf{pre}}$
  - Proofs of soundness w.r.t. nonstandard interpretation, and other nice properties

Solution Summary (1)

- Standard interpretation $[\![\cdot]\!]$
  - Directly implementable (with floating-point approximation)
- Nonstandard interpretation $[\![\cdot]\!]_{\mathbf{pre}}$ for computing preimages
  - Unimplementable (except in λ_{ZFC})
  - Proof of correctness w.r.t. standard interpretation
- Approximating interpretation $[\![\cdot]\!]'_{\mathbf{pre}}$
  - Proofs of soundness w.r.t. nonstandard interpretation, and other nice properties
  - Directly implementable, given a rectangular set library

Solution Summary (2)

- Rectangular set library
  - Sets required by semantics: rectangular subsets of Ω, T, Bool, finite cartesian products

Solution Summary (2)

- Rectangular set library
  - Sets required by semantics: rectangular subsets of Ω, T, Bool, finite cartesian products
  - Implementation-specific rectangles: $\{\mathrm{null}\}$, interval unions, tagged structures, symbol sets (planned)

Solution Summary (2)

- Rectangular set library
  - Sets required by semantics: rectangular subsets of Ω, T, Bool, finite cartesian products
  - Implementation-specific rectangles: $\{\mathrm{null}\}$, interval unions, tagged structures, symbol sets (planned)
  - Values for each rectangular set type

Solution Summary (2)

- Rectangular set library
  - Sets required by semantics: rectangular subsets of Ω, T, Bool, finite cartesian products
  - Implementation-specific rectangles: $\{\mathrm{null}\}$, interval unions, tagged structures, symbol sets (planned)
  - Values for each rectangular set type
- Two sampling algorithms (so far!)
  - Use interpretations to sample uniformly from preimages

Solution Summary (2)

- Rectangular set library
  - Sets required by semantics: rectangular subsets of Ω, T, Bool, finite cartesian products
  - Implementation-specific rectangles: $\{\mathrm{null}\}$, interval unions, tagged structures, symbol sets (planned)
  - Values for each rectangular set type
- Two sampling algorithms (so far!)
  - Use interpretations to sample uniformly from preimages
  - Correct for any program, any positive-probability condition (up to floating-point error)

Solution Summary (2)

- Rectangular set library
  - Sets required by semantics: rectangular subsets of Ω, T, Bool, finite cartesian products
  - Implementation-specific rectangles: $\{\mathrm{null}\}$, interval unions, tagged structures, symbol sets (planned)
  - Values for each rectangular set type
- Two sampling algorithms (so far!)
  - Use interpretations to sample uniformly from preimages
  - Correct for any program, any positive-probability condition (up to floating-point error)
  - Degrade to rejection sampling

Demos

- Normal-Normal with observation
- Normal-Normal with circular condition
- Thermometer
- Normal-Normals with varying ε
- Boolean PCFG
- Ray tracing

Future Work

- Optimizations
  - O(1) random variable lookup (with proof)

Future Work

- Optimizations
  - O(1) random variable lookup (with proof)
  - Supercombinators

Future Work

- Optimizations
  - O(1) random variable lookup (with proof)
  - Supercombinators
  - Incremental computation

Future Work

- Optimizations
  - O(1) random variable lookup (with proof)
  - Supercombinators
  - Incremental computation
  - Static analysis

Future Work

- Optimizations
  - O(1) random variable lookup (with proof)
  - Supercombinators
  - Incremental computation
  - Static analysis
- Other samplers
  - MCMC on partitioned program domain (countable)

Future Work

- Optimizations
  - O(1) random variable lookup (with proof)
  - Supercombinators
  - Incremental computation
  - Static analysis
- Other samplers
  - MCMC on partitioned program domain (countable)
  - In general, guide sampling using previous samples

Future Work

- Optimizations
  - O(1) random variable lookup (with proof)
  - Supercombinators
  - Incremental computation
  - Static analysis
- Other samplers
  - MCMC on partitioned program domain (countable)
  - In general, guide sampling using previous samples
- Discover, characterize and overcome current limitations