Rigorous bounds for posterior inference in universal probabilistic programming

Raven Beutner[1] Luke Ong[2] **Fabian Zaiser**[2]

[1]CISPA Helmholtz Center for Information Security [2]University of Oxford

Languages for Inference @ POPL 2022

A random walk as a probabilistic program

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

A random walk as a probabilistic program

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

- continuous distributions
- unbounded loops
- unbounded number of `sample`s

Existing inference methods

1. Approximate: posterior $\approx X$

- Monte Carlo (particle filter, MCMC)
- or optimization-based (variational inference)

Existing inference methods

1. Approximate: posterior $\approx X$

- Monte Carlo (particle filter, MCMC)
- or optimization-based (variational inference)

PYRO

2. Exact: posterior $= X$

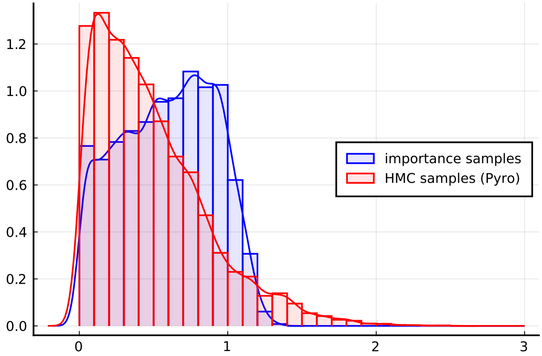
- symbolic expression

PSI SOLVER

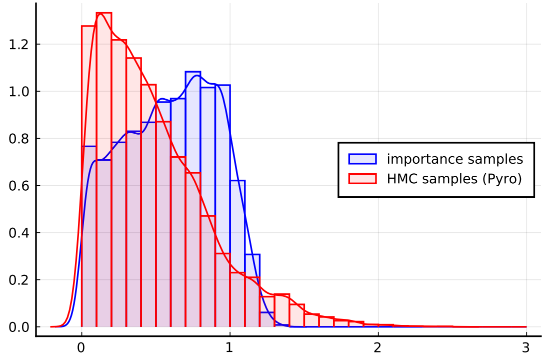
HAKARU

ΣΠPL

Issues with existing methods



Issues with existing methods



- **approximate methods:** convergence (e.g. for multimodal models)
- **exact methods:** restricted models (e.g. no recursion)

Rigorous Bounds on the Posterior:
$\mathsf{posterior}(E) \in [a, b]$

Rigorous Bounds on the Posterior: posterior$(E) \in [a, b]$

... for

- a universal PPL (including branching & recursion)
- with continuous distributions
- and conditioning (`observe`)

Rigorous Bounds on the Posterior: posterior$(E) \in [a, b]$

…for

- a universal PPL (including branching & recursion)
- with continuous distributions
- and conditioning (`observe`)

Why?

- construct ground truth for inference problems
- to debug approximate inference

Rigorous Bounds on the Posterior: $\text{posterior}(E) \in [a, b]$

…for

- a universal PPL (including branching & recursion)
- with continuous distributions
- and conditioning (`observe`)

Why?

- construct ground truth for inference problems
- to debug approximate inference

How?

1. interval traces & interval arithmetic (basic idea)
2. interval type system (overapproximation)
3. symbolic execution (optimization of special case)

Method 1: Interval traces

	standard semantics
traces	$\langle 0.2, 0.8 \rangle$ $\mathbb{T} := \bigcup_{n \in \mathbb{N}} \mathbb{R}^n$
value	$\mathsf{val}_P : \mathbb{T} \to \mathbb{R}$
weight	$\mathsf{wt}_P : \mathbb{T} \to [0, \infty)$
posterior	$[\![P]\!](E)$ integral over $\mathbb{T}$

Method 1: Interval traces

Idea: summarize traces using intervals

	standard semantics	interval semantics
traces	$\langle 0.2, 0.8 \rangle$	$\langle [0.2, 0.3], [0.7, 0.8] \rangle$
	$\mathbb{T} := \bigcup_{n \in \mathbb{N}} \mathbb{R}^n$	$\mathbb{T}_{\mathbb{I}} := \bigcup_{n \in \mathbb{N}} \mathbb{I}^n$
value	$\mathsf{val}_P : \mathbb{T} \to \mathbb{R}$	$\mathsf{val}_P^{\mathbb{I}} : \mathbb{T}_{\mathbb{I}} \to \mathbb{I}$
weight	$\mathsf{wt}_P : \mathbb{T} \to [0, \infty)$	$\mathsf{wt}_P^{\mathbb{I}} : \mathbb{T}_{\mathbb{I}} \to \mathbb{I}_{[0,\infty)}$
posterior	$[\![P]\!](E)$	$[\mathsf{lowerBd}_P^{\mathcal{T}}(E), \mathsf{upperBd}_P^{\mathcal{T}}(E)]$
	integral over $\mathbb{T}$	sum over partition $\mathcal{T} \subset \mathbb{T}_{\mathbb{I}}$

Interval trace semantics

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

Interval trace semantics

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

	standard	interval semantics
`start`	0.6	$[0.5, 0.6]$
`position`		
`distance`		
trace t	$\langle 0.6, 0.2, -0.8\rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8]\rangle$
weight wt(t)	1	$[1, 1]$
return value val(t)		

Interval trace semantics

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

	standard	interval semantics
`start`	0.6	$[0.5, 0.6]$
`position`	0.6	$[0.5, 0.6]$
`distance`	0.0	$[0.0, 0.0]$
trace t	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight wt(t)	1	$[1, 1]$
return value val(t)		

Interval trace semantics

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

	standard	interval semantics
`start`	0.6	$[0.5, 0.6]$
`position`	0.6	$[0.5, 0.6]$
`distance`	0.0	$[0.0, 0.0]$
trace t	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight wt(t)	1	$[1, 1]$
return value val(t)		

Interval trace semantics

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

	standard	interval semantics
`start`	0.6	$[0.5, 0.6]$
`position`	0.8	$[0.6, 0.8]$
`distance`	0.2	$[0.1, 0.2]$
trace t	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight $\mathsf{wt}(t)$	1	$[1, 1]$
return value $\mathsf{val}(t)$		

Interval trace semantics

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

	standard	interval semantics
`start`	0.6	$[0.5, 0.6]$
`position`	0.8	$[0.6, 0.8]$
`distance`	0.2	$[0.1, 0.2]$
trace t	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight wt(t)	1	$[1, 1]$
return value val(t)		

Interval trace semantics

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

	standard	interval semantics
`start`	0.6	$[0.5, 0.6]$
`position`	0.0	$[-0.3, 0.0]$
`distance`	1.0	$[0.9, 1.1]$
trace t	$\langle 0.6\,, 0.2\,, -0.8\rangle$	$\langle [0.5, 0.6]\,, [0.1, 0.2]\,, [-0.9, -0.8]\rangle$
weight $\mathsf{wt}(t)$	1	$[1, 1]$
return value $\mathsf{val}(t)$		

Interval trace semantics

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

	standard	interval semantics
start	0.6	$[0.5, 0.6]$
position	0.0	$[-0.3, 0.0]$
distance	1.0	$[0.9, 1.1]$
trace t	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight wt(t)	≈ 2.4	$[0.53, 3.99]$
return value val(t)		

Interval trace semantics

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
observe 1.1 from normal(distance, 0.1²)
return start
```

	standard	interval semantics
`start`	0.6	$[0.5, 0.6]$
`position`	0.0	$[-0.3, 0.0]$
`distance`	1.0	$[0.9, 1.1]$
trace t	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight $\mathsf{wt}(t)$	≈ 2.4	$[0.53, 3.99]$
return value $\mathsf{val}(t)$	0.6	$[0.5, 0.6]$

Lower bounds on the posterior

If $\mathcal{T} \subset \mathbb{T}_{\mathbb{I}}$ is "non-overlapping", then define for all intervals I:

$$\mathsf{lowerBd}_P^{\mathcal{T}}(I) := \sum_{\boldsymbol{s}_{\mathbb{I}} \in \mathcal{T}} \mathrm{vol}(\boldsymbol{s}_{\mathbb{I}}) \cdot (\min \mathsf{wt}_P^{\mathbb{I}}(\boldsymbol{s}_{\mathbb{I}})) \cdot [\mathsf{val}_P^{\mathbb{I}}(\boldsymbol{s}_{\mathbb{I}}) \subseteq I]$$

$$\mathrm{vol}(\langle [a_1, b_1], \ldots, [a_n, b_n] \rangle) := (b_1 - a_1) \times \cdots \times (b_n - a_n)$$

Lower bounds on the posterior

If $\mathcal{T} \subset \mathbb{T}_{\mathbb{I}}$ is "non-overlapping", then define for all intervals I:

$$\mathsf{lowerBd}_P^{\mathcal{T}}(I) := \sum_{\boldsymbol{s}_{\mathbb{I}} \in \mathcal{T}} \mathrm{vol}(\boldsymbol{s}_{\mathbb{I}}) \cdot (\min \mathsf{wt}_P^{\mathbb{I}}(\boldsymbol{s}_{\mathbb{I}})) \cdot [\mathsf{val}_P^{\mathbb{I}}(\boldsymbol{s}_{\mathbb{I}}) \subseteq I]$$

$$\mathrm{vol}(\langle [a_1, b_1], \ldots, [a_n, b_n] \rangle) := (b_1 - a_1) \times \cdots \times (b_n - a_n)$$

Upper bounds on the posterior

If $\mathcal{T} \subset \mathbb{T}_{\mathbb{I}}$ is "exhaustive" (covers every trace), then define for all intervals I:

$$\mathsf{upperBd}_P^{\mathcal{T}}(I) := \sum_{\boldsymbol{s}_{\mathbb{I}} \in \mathcal{T}} \mathrm{vol}(\boldsymbol{s}_{\mathbb{I}}) \cdot (\max \mathsf{wt}_P^{\mathbb{I}}(\boldsymbol{s}_{\mathbb{I}})) \cdot [\mathsf{val}_P^{\mathbb{I}}(\boldsymbol{s}_{\mathbb{I}}) \cap I \neq \emptyset]$$

Soundness

$$\mathsf{lowerBd}_P^{\mathcal{T}} \leq [\![P]\!] \leq \mathsf{upperBd}_P^{\mathcal{T}}.$$

Soundness

$$\text{lowerBd}_P^{\mathcal{T}} \leq [\![P]\!] \leq \text{upperBd}_P^{\mathcal{T}}.$$

Completeness

For all intervals I and $\epsilon > 0$, there is a countable set $\mathcal{T} \subseteq \mathbb{T}_{\mathbb{I}}$ s.t.

$$\text{upperBd}_P^{\mathcal{T}}(I) - \epsilon \leq [\![P]\!](I) \leq \text{lowerBd}_P^{\mathcal{T}}(I) + \epsilon$$

under the assumptions:

- the primitive functions are continuous*
- each **`sample`**d value is used at most once in each condition, **`observe`** statement, and in the return value.

Soundness

$$\mathsf{lowerBd}_P^{\mathcal{T}} \leq [\![P]\!] \leq \mathsf{upperBd}_P^{\mathcal{T}}.$$

Completeness

For all intervals I and $\epsilon > 0$, there is a countable set $\mathcal{T} \subseteq \mathbb{T}_{\mathbb{I}}$ s.t.

$$\mathsf{upperBd}_P^{\mathcal{T}}(I) - \epsilon \leq [\![P]\!](I) \leq \mathsf{lowerBd}_P^{\mathcal{T}}(I) + \epsilon$$

under the assumptions:

- the primitive functions are continuous*
- each **`sample`**d value is used at most once in each condition, **`observe`** statement, and in the return value.

Soundness

$$\text{lowerBd}_P^{\mathcal{T}} \leq [\![P]\!] \leq \text{upperBd}_P^{\mathcal{T}}.$$

Completeness

For all intervals I and $\epsilon > 0$, there is a finite set $\mathcal{T} \subseteq \mathbb{T}_{\mathbb{I}}$ s.t.

$$[\![P]\!](I) \leq \text{lowerBd}_P^{\mathcal{T}}(I) + \epsilon$$

under the assumptions:

- the primitive functions are continuous*
- each **`sample`**d value is used at most once in each condition, **`observe`** statement, and in the return value.

Method 2: Interval type system

$\rightarrow$ to overapproximate recursion and conditionals (not resolvable by intervals)

Method 2: Interval type system

$\rightarrow$ to overapproximate recursion and conditionals (not resolvable by intervals)

- types keep track of the value and weight interval
- $\vdash P : \begin{Bmatrix} [v, v'] \\ [w, w'] \end{Bmatrix}$ means $\mathsf{val}_P(\boldsymbol{s}) \in [v, v']$ and $\mathsf{wt}_P(\boldsymbol{s}) \in [w, w']$.
- efficient type inference
- uses interval arithmetic & widening to approximate fixpoints

Method 3: Symbolic execution

$\rightarrow$ optimization for a common special case

Method 3: Symbolic execution

$\rightarrow$ optimization for a common special case

For each program path,

- α_k: the k-th **`sample`**
- $\mathcal{V}$: result value, e.g. $\alpha_1 + 2\alpha_2$
- Δ: guards, e.g. $\{\alpha_1 \leq 0, \alpha_1 + \alpha_2 > 1\}$
- Ξ: weights, e.g. $\{\mathrm{pdf}_{\mathrm{Normal}(0,1)}(\alpha_1 - \alpha_2), \mathrm{pdf}_{\mathrm{Normal}(1,2)}(\alpha_3)\}$

Method 3: Symbolic execution

$\to$ optimization for a common special case

For each program path,

- α_k: the k-th **`sample`**
- $\mathcal{V}$: result value, e.g. $\alpha_1 + 2\alpha_2$
- Δ: guards, e.g. $\{\alpha_1 \leq 0, \alpha_1 + \alpha_2 > 1\}$
- Ξ: weights, e.g. $\{\mathrm{pdf}_{\mathrm{Normal}(0,1)}(\alpha_1 - \alpha_2), \mathrm{pdf}_{\mathrm{Normal}(1,2)}(\alpha_3)\}$

$$[\![P]\!](I) = \sum_{\text{paths}} \int_{\Delta \cup \{\mathcal{V} \in I\}} \left(\prod \Xi \right) \mathrm{d}\alpha$$

Method 3: Symbolic execution

$\rightarrow$ optimization for a common special case

For each program path,

- α_k: the k-th **`sample`**
- $\mathcal{V}$: result value, e.g. $\alpha_1 + 2\alpha_2$
- Δ: guards, e.g. $\{\alpha_1 \leq 0, \alpha_1 + \alpha_2 > 1\}$
- Ξ: weights, e.g. $\{\mathrm{pdf}_{\mathrm{Normal}(0,1)}(\alpha_1 - \alpha_2), \mathrm{pdf}_{\mathrm{Normal}(1,2)}(\alpha_3)\}$

$$[\![P]\!](I) = \sum_{\text{paths}} \int_{\Delta \cup \{\mathcal{V} \in I\}} \left(\prod \Xi\right) \mathrm{d}\alpha \leq \sum_{\text{paths}} \mathrm{vol}(\Delta \cup \{\mathcal{V} \in I\}) \prod_{\mathcal{W} \in \Xi} \max_\alpha \mathcal{W}$$

Method 3: Symbolic execution

→ optimization for a common special case

For each program path,

- α_k: the k-th **`sample`**
- $\mathcal{V}$: result value, e.g. $\alpha_1 + 2\alpha_2$
- Δ: guards, e.g. $\{\alpha_1 \leq 0, \alpha_1 + \alpha_2 > 1\}$
- Ξ: weights, e.g. $\{\mathrm{pdf}_{\mathrm{Normal}(0,1)}(\alpha_1 - \alpha_2), \mathrm{pdf}_{\mathrm{Normal}(1,2)}(\alpha_3)\}$

$$[\![P]\!](I) = \sum_{\text{paths}} \int_{\Delta \cup \{\mathcal{V} \in I\}} \left(\prod \Xi\right) \mathrm{d}\alpha \leq \sum_{\text{paths}} \mathrm{vol}(\Delta \cup \{\mathcal{V} \in I\}) \prod_{\mathcal{W} \in \Xi} \max_\alpha \mathcal{W}$$

If Δ and $\mathcal{V}$ are affine then use

- polytope volume computation (→ Vinci tool)

Method 3: Symbolic execution

$\rightarrow$ optimization for a common special case

For each program path,

- α_k: the k-th **`sample`**
- $\mathcal{V}$: result value, e.g. $\alpha_1 + 2\alpha_2$
- Δ: guards, e.g. $\{\alpha_1 \leq 0, \alpha_1 + \alpha_2 > 1\}$
- Ξ: weights, e.g. $\{\mathrm{pdf}_{\mathrm{Normal}(0,1)}(\alpha_1 - \alpha_2), \mathrm{pdf}_{\mathrm{Normal}(1,2)}(\alpha_3)\}$

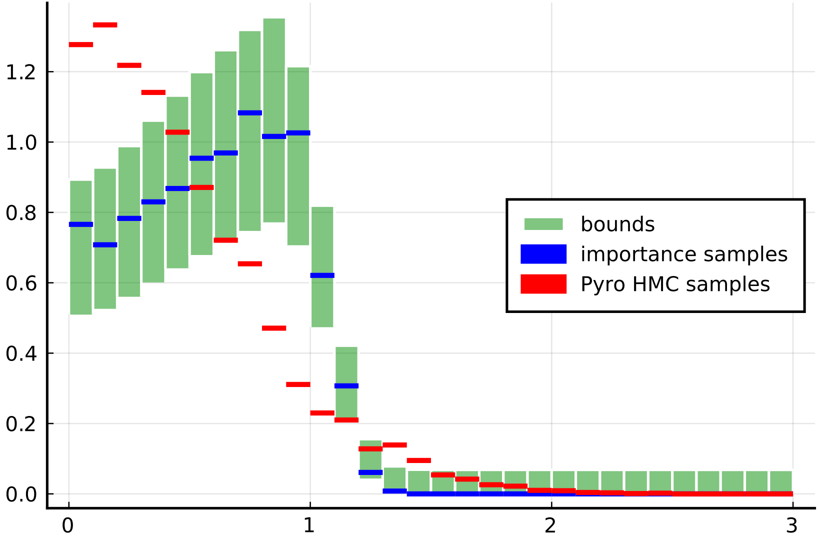
$$[\![P]\!](I) = \sum_{\text{paths}} \int_{\Delta \cup \{\mathcal{V} \in I\}} \left(\prod \Xi\right) \mathrm{d}\alpha \leq \sum_{\text{paths}} \mathrm{vol}(\Delta \cup \{\mathcal{V} \in I\}) \prod_{\mathcal{W} \in \Xi} \max_\alpha \mathcal{W}$$

If Δ and $\mathcal{V}$ are affine then use

- polytope volume computation ($\rightarrow$ Vinci tool)
- linear optimization & interval arithmetic

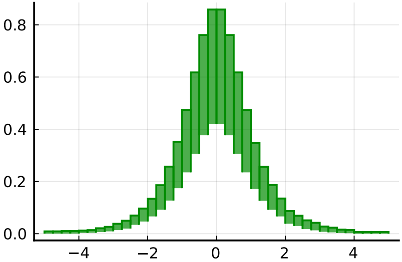
Empirical evaluation

Empirical evaluation

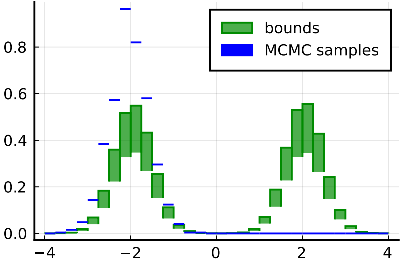


(time: 50 minutes)

Examples that are hard for MCMC



(a) Neal's funnel (5 seconds)



(b) Binary Gaussian mixture model (90 seconds)

Comparison with previous work

Sankaranarayanan et al. (PLDI13)

- bounding probabilities (but no `observe`)
- ours is usually slower, but often better bounds

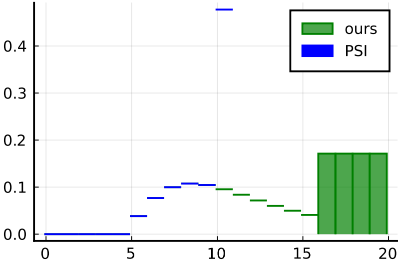
Comparison with previous work

Sankaranarayanan et al. (PLDI13)

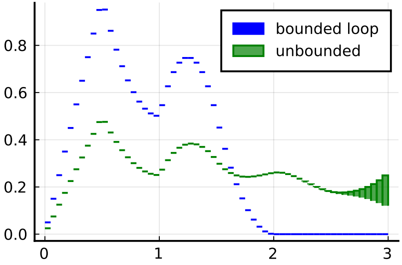
- bounding probabilities (but no `observe`)
- ours is usually slower, but often better bounds

PSI solver

- consistency check: benchmarks from the PSI repository
- we can handle unbounded loops, contrary to PSI



(a) $\approx$ 2 minutes



(b) $\approx$ 20 seconds

Limitations

- lots of branching
- high-dimensional models (many `sample`s)

Limitations

- lots of branching
- high-dimensional models (many `sample`s)

Future work

- better heuristics for finding a "good" $\mathcal{T}$
- can this be refined into an approximate inference algorithm?

Rigorous Bounds on the Posterior: posterior$(E) \in [a, b]$

…for a universal PPL with continuous distributions and conditioning (`observe`)

Why?

- construct ground truth
- debug approximate inference

How?

1. interval trace semantics
2. interval type system
3. symbolic execution

Backup slides

Semantics of a probabilistic program

Let $P : \mathbb{R}$ a probabilistic program.

- trace space: $\mathbb{T} := \bigcup_{n\in\mathbb{N}} \mathbb{R}^n$
- value function: $\mathsf{val}_P : \mathbb{T} \to \mathbb{R}$
- weight function: $\mathsf{wt}_P : \mathbb{T} \to [0, \infty)$

Unnormalized posterior:

$$[\![P]\!](E) := \int_{\mathsf{val}_P^{-1}(E)} \mathsf{wt}_P(t)\mu_{\mathbb{T}}(\mathrm{d}t) \qquad \text{for event } E \subseteq \mathbb{R}.$$

Normalized posterior (probability distribution):

$$\mathsf{posterior}_P(E) := \frac{1}{Z}[\![P]\!](E) \qquad \text{where } Z := [\![P]\!](\mathbb{R}).$$

Trace partitioning heuristics

Option 1: split equidistantly in each dimension

Option 2:

- start with the full interval trace $\langle[-\infty, \infty], \ldots\rangle$
- pick the next interval $\boldsymbol{s}_{\mathbb{I}}$ trace or, depending on the input program, select it with a mix of the following criteria
  - high weight $\mathsf{wt}^{\mathbb{I}}_{P}(\boldsymbol{s}_{\mathbb{I}})$
  - wide value interval $\mathsf{val}^{\mathbb{I}}_{P}(\boldsymbol{s}_{\mathbb{I}})$
  - large volume $\mathrm{vol}(\boldsymbol{s}_{\mathbb{I}})$
- split that box in half along the dimension that reduces the width of the interval of the posterior expected value the most
- repeat.

Interval type system

Types:

- unweighted: $\sigma ::= [v, v'] \mid \sigma \to \mathcal{A}$
- weighted: $\mathcal{A} ::= \begin{Bmatrix} \sigma \\ [w, w'] \end{Bmatrix}$

Selected typing rules:

$$\frac{\Gamma; \varphi : \sigma \to \mathcal{A}; x : \sigma \vdash M : \mathcal{A}}{\Gamma \vdash \mu_x^{\varphi}.M : \begin{Bmatrix} \sigma \to \mathcal{A} \\ [1,1] \end{Bmatrix}}$$

$$\frac{\Gamma \vdash M : \begin{Bmatrix} \sigma_1 \to \begin{Bmatrix} \sigma_2 \\ [e, f] \end{Bmatrix} \\ [a, b] \end{Bmatrix} \quad \Gamma \vdash N : \begin{Bmatrix} \sigma_1 \\ [c, d] \end{Bmatrix}}{\Gamma \vdash MN : \begin{Bmatrix} \sigma_2 \\ [a, b] \times^{\mathbb{I}} [c, d] \times^{\mathbb{I}} [e, f] \end{Bmatrix}}$$