Guaranteed 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

PLDI 2022

Our work

- Probabilistic programming: Bayesian statistical models as programs
- Vision: Bayesian inference algorithms for any program

Our work

- Probabilistic programming: Bayesian statistical models as programs
- Vision: Bayesian inference algorithms for any program

Problem: existing inference algorithms

✗ have few guarantees on the result or

✗ only work on a restricted class of models

Our work

- Probabilistic programming: Bayesian statistical models as programs
- Vision: Bayesian inference algorithms for any program

Problem: existing inference algorithms

✗ have few guarantees on the result or

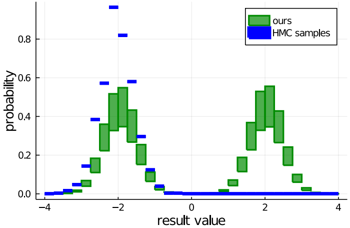
✗ only work on a restricted class of models

Our contribution:

guaranteed bounds on the posterior

✓ can find errors in inference results

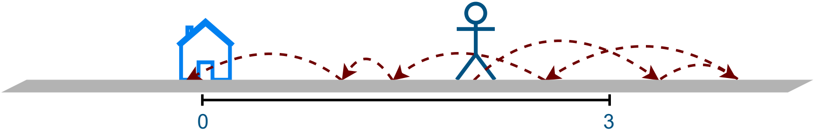
✓ applicable to a broad class of probabilistic programs



Example model

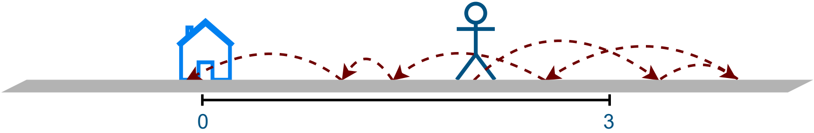

```
start = sample uniform(0,3)
```

Example model



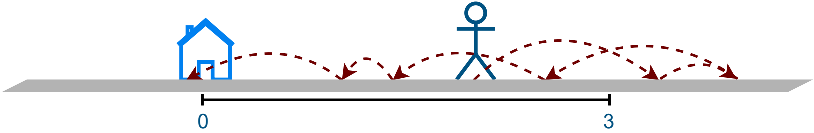

```
start = sample uniform(0,3)
position = start; distance = 0
while position > 0:
    step = sample uniform(-1, 1)
    position += step
    distance += abs(step)
```

Example model




```
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, 0.1)
```

Example model




```
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, 0.1)
return start
```

Posterior distribution $p(start \mid \text{observation})$?

Existing inference methods

Existing inference methods

1. Approximate: posterior $\approx X$

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

Stan *Pyro* *Anglican*

Existing inference methods

1. Approximate: posterior $\approx X$

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

Stan *Pyro* *Anglican*

2. Exact: posterior $= X$

- symbolic expression

PSI SOLVER HAKARU ΣΠPL

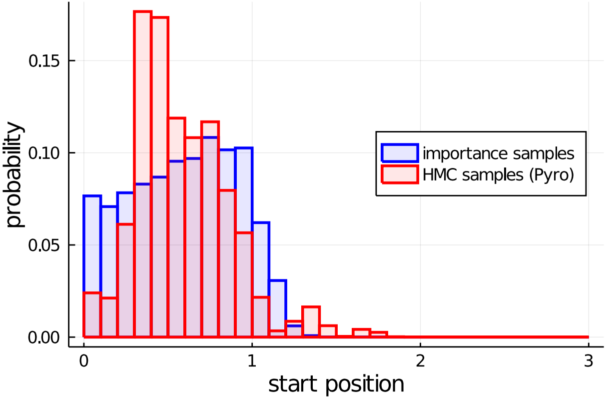
SPPL

Issues with existing methods

- **exact methods:** restricted models
- **approximate methods:** implicit assumptions, slow convergence

Issues with existing methods

- **exact methods:** restricted models
- **approximate methods:** implicit assumptions, slow convergence



Guaranteed Bounds on the Posterior:

$\text{posterior}(E) \in [a, b]$

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

...for a universal PPL with continuous distributions and **`observe`**.

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

…for a universal PPL with continuous distributions and `observe`.

Why?

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

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

…for a universal PPL with continuous distributions and **`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)

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

... for a universal PPL with continuous distributions and `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)

Semantics of a probabilistic program

- trace s records **sample**d values, e.g. $\langle 0.23, 0.79 \rangle$
- result value $\mathsf{resval}(s)$ for trace s
- weight $\mathsf{weight}(s)$: product of likelihoods of observations

Semantics of a probabilistic program

- trace $\boldsymbol{s}$ records **sample**d values, e.g. $\langle 0.23, 0.79 \rangle$
- result value $\mathsf{resval}(\boldsymbol{s})$ for trace $\boldsymbol{s}$
- weight $\mathsf{weight}(\boldsymbol{s})$: product of likelihoods of observations

Unnormalized posterior of E (*joint probability*):

$$[\![P]\!](E) := \int_{\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in E\}} \mathsf{weight}(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s} = \text{“}\,\mathbb{P}(\texttt{start} \in E, \mathsf{obs})\,\text{”}$$

Semantics of a probabilistic program

- trace s records **sampled** values, e.g. $\langle 0.23, 0.79 \rangle$
- result value $\mathsf{resval}(s)$ for trace s
- weight $\mathsf{weight}(s)$: product of likelihoods of observations

Unnormalized posterior of E (*joint probability*):

$$[\![P]\!](E) := \int_{\{s \mid \mathsf{resval}(s) \in E\}} \mathsf{weight}(s)\,\mathrm{d}s = \text{“}\mathbb{P}(\mathtt{start} \in E, \mathsf{obs})\text{”}$$

Bayes' rule $\rightsquigarrow$ **normalized posterior** (*conditional probability*):

$$\mathbb{P}(\mathtt{start} \in E \mid \mathsf{obs}) = \frac{\mathbb{P}(\mathtt{start} \in E, \mathsf{obs})}{\mathbb{P}(\mathsf{obs})} = \frac{[\![P]\!](E)}{[\![P]\!](\mathbb{R})}$$

Semantics of a probabilistic program

- trace s records **sample**d values, e.g. $\langle 0.23, 0.79 \rangle$
- result value $\mathsf{resval}(s)$ for trace s
- weight $\mathsf{weight}(s)$: product of likelihoods of observations

Unnormalized posterior of E (*joint probability*):

$$[\![P]\!](E) := \int_{\{s \mid \mathsf{resval}(s) \in E\}} \mathsf{weight}(s)\, \mathrm{d}s = \text{“}\mathbb{P}(\mathtt{start} \in E, \mathsf{obs})\text{”}$$

Bayes' rule $\rightsquigarrow$ **normalized posterior** (*conditional probability*):

$$\mathbb{P}(\mathtt{start} \in E \mid \mathsf{obs}) = \frac{\mathbb{P}(\mathtt{start} \in E, \mathsf{obs})}{\mathbb{P}(\mathsf{obs})} = \frac{[\![P]\!](E)}{[\![P]\!](\mathbb{R})}$$

Interval traces

Want to bound $[\![P]\!](E) = \int_{\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in E\}} \mathsf{weight}(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{s}$.

Interval traces

Want to bound $[\![P]\!](E) = \int_{\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in E\}} \mathsf{weight}(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{s}$.

Idea: *Riemann sums*

Interval traces

Want to bound $[\![P]\!](E) = \int_{\{\boldsymbol{s} | \mathsf{resval}(\boldsymbol{s}) \in E\}} \mathsf{weight}(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{s}$.

Idea: *Riemann sums*

Cover $\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in E\}$ with interval traces $\mathcal{T}$

- e.g. $\langle [0.1, 0.3], [0.7, 1] \rangle$ contains $\langle 0.2, 0.9 \rangle$

Interval traces

Want to bound $[\![P]\!](E) = \int_{\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in E\}} \mathsf{weight}(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{s}$.

Idea: *Riemann sums*

Cover $\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in E\}$ with interval traces $\mathcal{T}$

- e.g. $\langle [0.1, 0.3], [0.7, 1] \rangle$ contains $\langle 0.2, 0.9 \rangle$

$$[\![P]\!](E) \leq \sum_{\boldsymbol{t} \in \mathcal{T}} (\max \mathsf{weight}(\boldsymbol{t})) \, \mathrm{vol}(\boldsymbol{t})$$

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
```

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`		
`position`		
`distance`		
trace s	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight weight(s)	1	$[1, 1]$
return value resval(s)		

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 s	$\langle 0.6, 0.2, -0.8\rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8]\rangle$
weight weight(s)	1	$[1, 1]$
return value resval(s)		

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 s	$\langle 0.6\,, 0.2\,, -0.8\rangle$	$\langle [0.5, 0.6]\,, [0.1, 0.2]\,, [-0.9, -0.8]\rangle$
weight weight(s)	1	$[1, 1]$
return value resval(s)		

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 s	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight weight(s)	1	$[1, 1]$
return value resval(s)		

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 s	$\langle 0.6\,, 0.2\,, -0.8\rangle$	$\langle [0.5, 0.6]\,, [0.1, 0.2]\,, [-0.9, -0.8]\rangle$
weight weight(s)	1	$[1, 1]$
return value resval(s)		

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 s	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight weight(s)	1	$[1, 1]$
return value resval(s)		

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 s	$\langle 0.6\,, 0.2\,, -0.8 \rangle$	$\langle [0.5, 0.6]\,, [0.1, 0.2]\,, [-0.9, -0.8] \rangle$
weight $\text{weight}(s)$	1	$[1, 1]$
return value $\text{resval}(s)$		

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 s	$\langle 0.6\,, 0.2\,, -0.8\rangle$	$\langle [0.5, 0.6]\,, [0.1, 0.2]\,, [-0.9, -0.8]\rangle$
weight weight(s)	≈ 2.4	$[0.53, 3.99]$
return value resval(s)		

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 s	$\langle 0.6, 0.2, -0.8 \rangle$	$\langle [0.5, 0.6], [0.1, 0.2], [-0.9, -0.8] \rangle$
weight weight(s)	≈ 2.4	$[0.53, 3.99]$
return value resval(s)	0.6	$[0.5, 0.6]$

Theoretical results

Soundness

For a non-overlapping and exhaustive set of interval traces $\mathcal{T}$:

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

Theoretical results

Soundness

For a non-overlapping and exhaustive set of interval traces $\mathcal{T}$:

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

Completeness

For all intervals I and $\epsilon > 0$, there is a countable set $\mathcal{T}$ of interval traces (non-overlapping and exhaustive) s.t.

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

under mild assumptions about the program P.

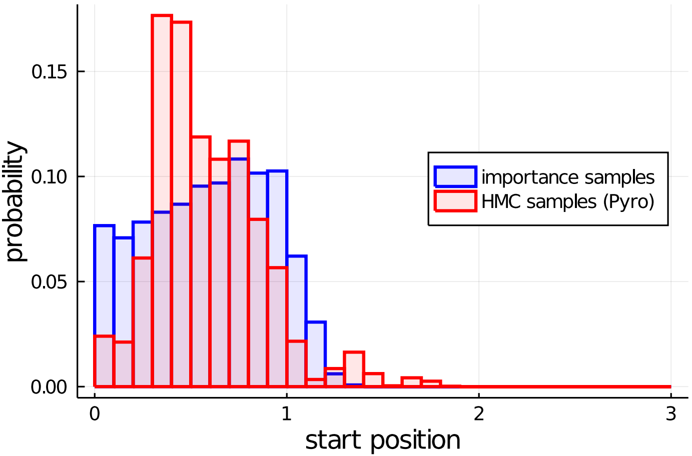
Empirical evaluation

- Implementation: *GuBPI* (gubpi-tool.github.io)

Empirical evaluation

- Implementation: *GuBPI* (gubpi-tool.github.io)

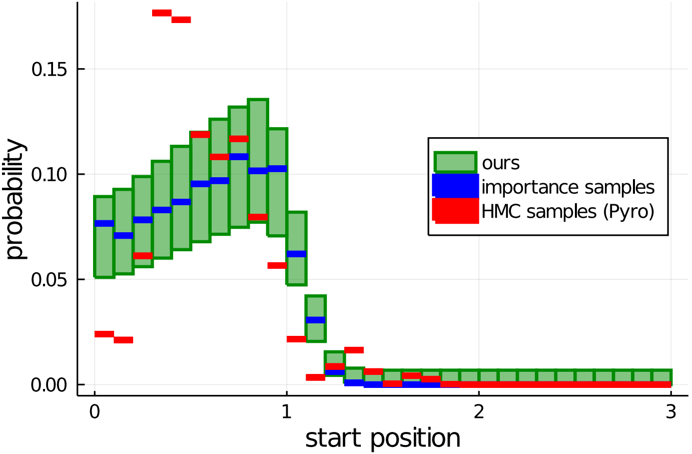
Pedestrian example:



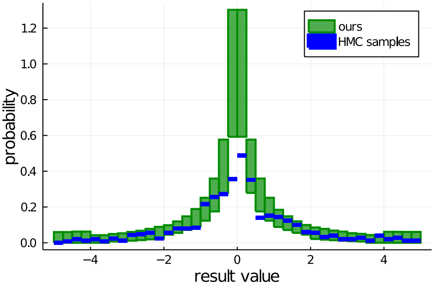
Empirical evaluation

- Implementation: *GuBPI* (gubpi-tool.github.io)

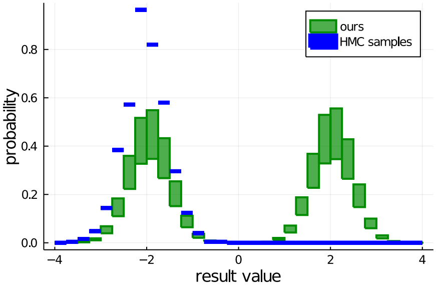
Pedestrian example:



Examples that are hard for MCMC



Neal’s funnel



Mixture model

Comparison with previous work

Sankaranarayanan et al. (PLDI2013)

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

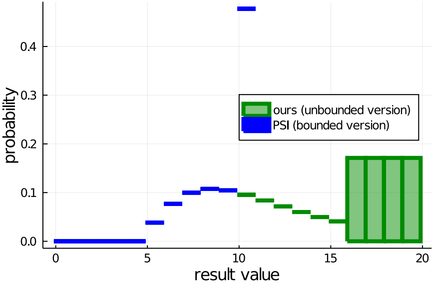
Comparison with previous work

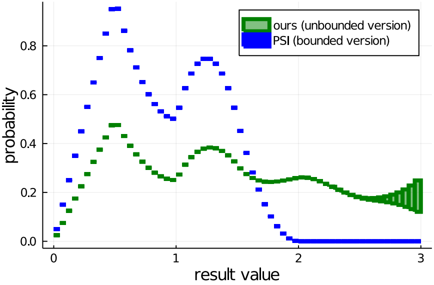
Sankaranarayanan et al. (PLDI2013)

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

PSI solver (CAV2016)

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





Also in the paper

- **Interval type system:** approximates unbounded loops and recursion *soundly*
- **Symbolic execution & linear programming:** optimization for linear guards
- **Comparison with statistical validation methods:** simulation-based calibration

Also in the paper

- **Interval type system:** approximates unbounded loops and recursion *soundly*
- **Symbolic execution & linear programming:** optimization for linear guards
- **Comparison with statistical validation methods:** simulation-based calibration

Limitations

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

Guaranteed Bounds for Posterior Inference in Universal Probabilistic Programming

Raven Beutner Luke Ong **Fabian Zaiser**

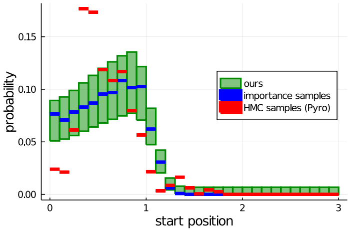
…are a **middle ground** between *approximate* and *exact*:

- guaranteed correct (vs. approximate inference)
- supports many language features (vs. exact inference)

Theory: soundness & completeness

Practice:

- detect issues with inference results
- competitive on existing benchmarks
- guaranteed results for programs that other tools cannot handle



Backup slides

Trace partitioning heuristics

Option 1: split equidistantly in each dimension

Option 2:

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

Method 1: Interval traces

	standard semantics
traces	$\boldsymbol{s} = \langle 0.2, 0.8 \rangle$
value	$\mathsf{resval}(\boldsymbol{s}) \in \mathbb{R}$
weight	$\mathsf{weight}(\boldsymbol{s}) \in [0, \infty)$
posterior	$[\![P]\!](E)$
	integral over traces $\boldsymbol{s}$

Method 1: Interval traces

Idea: summarize traces using intervals

	standard semantics	interval semantics
traces	$\boldsymbol{s} = \langle 0.2, 0.8 \rangle$	$\boldsymbol{t} = \langle [0.2, 0.3], [0.7, 0.8] \rangle$
value	$\mathsf{resval}(\boldsymbol{s}) \in \mathbb{R}$	$\mathsf{resval}^{\mathbb{I}}(\boldsymbol{t}) \in \mathbb{I}$
weight	$\mathsf{weight}(\boldsymbol{s}) \in [0, \infty)$	$\mathsf{weight}^{\mathbb{I}}(\boldsymbol{t}) \in \mathbb{I}_{[0,\infty)}$
posterior	$[\![P]\!](E)$	$[\mathsf{lowerBd}_P^{\mathcal{T}}(E), \mathsf{upperBd}_P^{\mathcal{T}}(E)]$
	integral over traces $\boldsymbol{s}$	sum over interval traces $\boldsymbol{t}$

Bounding the posterior

$$[\![P]\!](I) := \int_{\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in I\}} \mathsf{weight}(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{s}$$

Bounding the posterior

$$[\![P]\!](I) := \int_{\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in I\}} \mathsf{weight}(\boldsymbol{s}) \, \mathrm{d}\boldsymbol{s}$$

...if $\mathcal{T}$ is a set of interval traces that is "exhaustive" (covers every trace)

Bounding the posterior

$$[\![P]\!](I) := \int_{\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in I\}} \mathsf{weight}(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}$$

$$\leq \sum_{\substack{\boldsymbol{t} \in \mathcal{T} \\ \mathsf{resval}^{\mathbb{I}}(\boldsymbol{t}) \cap I \neq \emptyset}} \mathrm{vol}(\boldsymbol{t}) \cdot (\max \mathsf{weight}^{\mathbb{I}}(\boldsymbol{t}))$$

... if $\mathcal{T}$ is a set of interval traces that is "exhaustive" (covers every trace) and where

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

Bounding the posterior

$$
\begin{aligned}
[\![P]\!](I) &:= \int_{\{\boldsymbol{s} \mid \mathsf{resval}(\boldsymbol{s}) \in I\}} \mathsf{weight}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s} \\
\leq & \sum_{\substack{\boldsymbol{t} \in \mathcal{T} \\ \mathsf{resval}^{\mathbb{I}}(\boldsymbol{t}) \cap I \neq \emptyset}} \mathrm{vol}(\boldsymbol{t}) \cdot (\max \mathsf{weight}^{\mathbb{I}}(\boldsymbol{t})) \\
=: & \ \mathsf{upperBd}_P^{\mathcal{T}}(I)
\end{aligned}
$$

. . . if $\mathcal{T}$ is a set of interval traces that is "exhaustive" (covers every trace) and where

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

Bounding the posterior

$$\sum_{\substack{\boldsymbol{t}\in\mathcal{T}\\ \mathsf{resval}^{\mathbb{I}}(\boldsymbol{t})\subseteq I}} \mathrm{vol}(\boldsymbol{t})\cdot(\min \mathsf{weight}^{\mathbb{I}}(\boldsymbol{t}))$$

$$\leq [\![P]\!](I) := \int_{\{\boldsymbol{s}\mid\mathsf{resval}(\boldsymbol{s})\in I\}} \mathsf{weight}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}$$

$$\leq \sum_{\substack{\boldsymbol{t}\in\mathcal{T}\\ \mathsf{resval}^{\mathbb{I}}(\boldsymbol{t})\cap I\neq\emptyset}} \mathrm{vol}(\boldsymbol{t})\cdot(\max \mathsf{weight}^{\mathbb{I}}(\boldsymbol{t}))$$

$$=: \mathsf{upperBd}_P^{\mathcal{T}}(I)$$

. . . if $\mathcal{T}$ is a set of interval traces that is "non-overlapping" and "exhaustive" (covers every trace) and where

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

Bounding the posterior

$$
\begin{aligned}
&\mathsf{lowerBd}_P^{\mathcal{T}}(I) \\
&:= \sum_{\substack{\boldsymbol{t}\in\mathcal{T} \\ \mathsf{resval}^{\mathbb{I}}(\boldsymbol{t})\subseteq I}} \mathrm{vol}(\boldsymbol{t}) \cdot (\min \mathsf{weight}^{\mathbb{I}}(\boldsymbol{t})) \\
&\leq [\![P]\!](I) := \int_{\{\boldsymbol{s}\mid\mathsf{resval}(\boldsymbol{s})\in I\}} \mathsf{weight}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s} \\
&\leq \sum_{\substack{\boldsymbol{t}\in\mathcal{T} \\ \mathsf{resval}^{\mathbb{I}}(\boldsymbol{t})\cap I\neq\emptyset}} \mathrm{vol}(\boldsymbol{t}) \cdot (\max \mathsf{weight}^{\mathbb{I}}(\boldsymbol{t})) \\
&=: \mathsf{upperBd}_P^{\mathcal{T}}(I)
\end{aligned}
$$

…if $\mathcal{T}$ is a set of interval traces that is “non-overlapping” and “exhaustive” (covers every trace) and where

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

Theoretical results

Soundness

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

Theoretical results

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 $\mathcal{T}$ of interval traces (non-overlapping and exhaustive) 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.

Theoretical results

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 $\mathcal{T}$ of interval traces (non-overlapping and exhaustive) 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.

Theoretical results

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 finite set $\mathcal{T}$ of interval traces (non-overlapping and exhaustive) s.t.

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

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.

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
- $\vdash P : \left\{ \begin{matrix} [v, v'] \\ [w, w'] \end{matrix} \right\}$ means $\mathsf{resval}(\boldsymbol{s}) \in [v, v']$ and $\mathsf{weight}(\boldsymbol{s}) \in [w, w']$.
- efficient type inference
- uses interval arithmetic & widening to approximate fixpoints

Interval type system

Types:

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

Selected typing rules:

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

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

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

$\rightarrow$ 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. $\{\mathrm{pdf}_{\mathrm{Normal}(0,1)}(\alpha_1 - \alpha_2), \mathrm{pdf}_{\mathrm{Normal}(1,2)}(\alpha_3)\}$

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

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

$\rightarrow$ 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. $\{\mathrm{pdf}_{\mathrm{Normal}(0,1)}(\alpha_1 - \alpha_2), \mathrm{pdf}_{\mathrm{Normal}(1,2)}(\alpha_3)\}$

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

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

- polytope volume computation ($\rightarrow$ 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

Future work

- better heuristics for finding a "good" set of interval traces $\mathcal{T}$
- integration into an approximate inference algorithm?