Densities of Almost Surely Terminating Probabilistic Programs are Differentiable Almost Everywhere

Carol Mak Luke Ong Hugo Paquet

Dominik Wagner

UNIVERSITY OF OXFORD

ESOP 2021

Probabilistic Programming:

Probabilistic Programming:

Make Bayesian Machine Learning more accessible

Probabilistic Programming:

*Make Bayesian Machine Learning more accessible**

** to domain experts with basic programming skills*

Probabilistic Programming:

Make Bayesian Machine Learning more *accessible**

** to domain experts with basic programming skills*

- separate modelling from inference

Model

Inference

Model

PPL =

probabilistic programming language

Inference

e.g. conditionals, recursion,
higher-order functions

Model

PPL = "regular" language

+

+

probabilistic programming language

Inference

e.g. conditionals, recursion,
higher-order functions

Model

PPL = “regular” language

+ sample

+

probabilistic programming language

Inference

e.g. conditionals, recursion, higher-order functions

Model

PPL = "regular" language

+ sample

+ ***observations***

probabilistic programming language

Inference

e.g. conditionals, recursion, higher-order functions

Model

PPL = “regular” language

+ **sample**

+ ***observations***

probabilistic programming language

hard!

Inference

e.g. conditionals, recursion, higher-order functions

Model

PPL = "regular" language

+ sample

+ *observations*

probabilistic programming language

Inference

hard!

- MCMC
  - HMC
- Variational Inference
- ...

e.g. conditionals, recursion, higher-order functions

Model

PPL = "regular" language

+ sample

+ *observations*

probabilistic programming language

hard!

Inference

- MCMC
  - HMC
- Variational Inference
- ...

e.g. conditionals, recursion, higher-order functions

Model

PPL = "regular" language

+ sample

+ *observations*

probabilistic programming language

Inference

hard!

- MCMC
  - HMC
- Variational Inference
- ...

$\rightarrow$ ***exploit gradients***

e.g. conditionals, recursion, higher-order functions

Model

PPL = "regular" language

+ sample

+ *observations*

probabilistic programming language

correct?

Inference

hard!

- MCMC
  - HMC
- Variational Inference
- ...

→ ***exploit gradients***

[Yang, FSCD 2019]

"Can a probabilistic program denote a distribution with a density that is not differentiable at some non-measure-zero set?"

[Yang, FSCD 2019]

Contributions

Main Result

Contributions

Main Result

The value- and weight-functions are differentiable *almost everywhere*

(except for set of measure 0)

Contributions

Main Result

The value- and weight-functions are differentiable *almost everywhere* provided

(except for set of measure 0)

1. the program *terminates almost surely*

(probability non-termination is 0)

Contributions

Main Result

The value- and weight-functions are differentiable *almost everywhere* provided

(except for set of measure 0)

1. the program *terminates almost surely*
2. the primitive operations are well-behaved.

(probability non-termination is 0)

Contributions

Main Result

The value- and weight-functions are differentiable *almost everywhere* provided

(except for set of measure 0)

1. the program *terminates almost surely*
2. the primitive operations are well-behaved.

(probability non-termination is 0)

- special case: purely *deterministic* programs

Contributions

Main Result

The value- and weight-functions are differentiable *almost everywhere* provided

(except for set of measure 0)

1. the program *terminates almost surely*
2. the primitive operations are well-behaved.

(probability non-termination is 0)

- special case: purely *deterministic* programs
- proof technique: *symbolic* execution

Contributions

Main Result

The value- and weight-functions are differentiable *almost everywhere* provided

(except for set of measure 0)

1. the program *terminates almost surely*
2. the primitive operations are well-behaved.

(probability non-termination is 0)

- special case: purely *deterministic* programs
- proof technique: *symbolic* execution

This talk:

- focus on *weight*-function

Contributions

Main Result

The value- and weight-functions are differentiable *almost everywhere* provided

(except for set of measure 0)

1. the program *terminates almost surely*
2. the primitive operations are well-behaved.

(probability non-termination is 0)

- special case: purely *deterministic* programs
- proof technique: *symbolic* execution

This talk:

- focus on *weight*-function
- conditions on primitive operations
- symbolic execution and differentiability

Part I:
Operational Semantics

Recap

Probabilistic Program:

Probabilistic Program:

deterministic **function from random** ***samples*** **to**
value **(or failure) and unnormalised** ***density*** **(or weight)**

[Kozen 1979, Borgström et al. 2016, ...]

$$\text{if sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1})$$

$$\text{if sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1})$$

$$\text{weight}([0.1]) = 0$$

$$\text{if sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1})$$

$$\text{weight}([0.1]) = 0$$
$$\text{weight}([0.7]) = 1$$

Operational Semantics

Configuration:

Operational Semantics

Configuration: $\langle \; M \; , \; w \; , \; \mathbf{s} \; \rangle$

Operational Semantics

Configuration: $\langle\ M\ ,\ w\ ,\ \mathbf{s}\ \rangle$

term   accumulated weight   trace of samples

Operational Semantics

Configuration: $\langle M , w , \mathbf{s} \rangle$

term accumulated weight trace of samples

$$\langle \text{if sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [] \rangle$$

Operational Semantics

Configuration: $\langle\, M \,,\, w \,,\, \mathbf{s} \,\rangle$

term — accumulated weight — trace of samples

$$\langle \text{if } \textcolor{red}{\text{sample}} < \underline{0.5} \text{ then } \text{score}(\underline{0}) \text{ else } \text{score}(\underline{1}), 1, [] \rangle$$

Operational Semantics

Configuration: $\langle M, w, \mathbf{s} \rangle$

term accumulated weight trace of samples

$$\to \langle \text{if } \underline{\mathbf{0.1}} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [\mathbf{0.1}] \rangle$$

Operational Semantics

Configuration: $\langle\, \underbrace{M}_{\text{term}}\,,\ \underbrace{w}_{\text{accumulated weight}}\,,\ \underbrace{\mathbf{s}}_{\text{trace of samples}}\,\rangle$

$$\rightarrow \langle \text{if } \underline{\mathbf{0.1}} < \underline{\mathbf{0.5}} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [0.1]\rangle$$

Operational Semantics

Configuration: $\langle M, w, \mathbf{s} \rangle$

term accumulated weight trace of samples

$$\to \langle \mathsf{score}(\underline{0}), 1, [0.1] \rangle$$

Operational Semantics

Configuration: $\langle \underbrace{M}_{\text{term}}, \underbrace{w}_{\text{accumulated weight}}, \underbrace{\mathbf{s}}_{\text{trace of samples}} \rangle$

$$\to \langle \mathsf{score}(\underline{\mathbf{0}}), 1, [0.1] \rangle$$

Operational Semantics

Configuration: $\langle\, M \,,\, w \,,\, \mathbf{s} \,\rangle$

term

accumulated weight

trace of samples

$$\to \langle \underline{0}, \mathbf{0}, [0.1] \rangle$$

Operational Semantics

Configuration: $\langle\, M \,,\, w \,,\, \mathbf{s} \,\rangle$

term

accumulated weight

trace of samples

$$\to \langle \underline{0}, \mathbf{0}, [0.1] \rangle$$

$$\text{weight}([0.1]) = 0$$

Operational Semantics

Configuration: $\langle \underbrace{M}_{\text{term}} , \underbrace{w}_{\text{accumulated weight}} , \underbrace{\mathbf{s}}_{\text{trace of samples}} \rangle$

$\mathrm{weight}(\mathbf{s}) :=$

Operational Semantics

Configuration: $\langle M , w , \mathbf{s} \rangle$

term   accumulated weight   trace of samples

$$\text{weight}(\mathbf{s}) := \begin{cases} \mathbf{w} & \text{if } \langle M, 1, [] \rangle \to^* \langle V, \mathbf{w}, \mathbf{s} \rangle \\ & \end{cases}$$

Operational Semantics

Configuration: $\langle M , w , \mathbf{s} \rangle$

term — accumulated weight — trace of samples

$$\mathrm{weight}(\mathbf{s}) := \begin{cases} \mathbf{w} & \text{if } \langle M, 1, [] \rangle \to^* \langle V, \mathbf{w}, \mathbf{s} \rangle \\ 0 & \text{otherwise} \end{cases}$$

Operational Semantics

Configuration: $\langle\, M \,,\, w \,,\, \mathbf{s} \,\rangle$

term — accumulated weight — trace of samples

differentiable almost everywhere?

$$\mathrm{weight}(\mathbf{s}) := \begin{cases} \mathbf{w} & \text{if } \langle M, 1, []\rangle \to^* \langle V, \mathbf{w}, \mathbf{s}\rangle \\ 0 & \text{otherwise} \end{cases}$$

Part II:
Failure of Differentiability

1. Primitive Operations

1. Primitive Operations

```
let x = sample
in  score(f(x))
```

1. Primitive Operations

```
let x = sample
in  score(f(x))
```

$$\text{weight}([s]) = f(s)$$

1. Primitive Operations

```
let x = sample
in  score(χ_ℚ(x))
```

$$\text{weight}([s]) = \chi_{\mathbb{Q}}(s) = \begin{cases} 1 & \text{if } s \in \mathbb{Q} \\ 0 & \text{otherwise} \end{cases}$$

1. Primitive Operations

```
let x = sample
in  score(χ_ℚ(x))
```

not differentiable anywhere

$$\mathrm{weight}([s]) = \chi_{\mathbb{Q}}(s) = \begin{cases} 1 & \text{if } s \in \mathbb{Q} \\ 0 & \text{otherwise} \end{cases}$$

1. Primitive Operations

```
let x = sample
in  score(χ_ℚ(x))
```

not differentiable anywhere

$$\text{weight}([s]) = \chi_{\mathbb{Q}}(s) = \begin{cases} 1 & \text{if } s \in \mathbb{Q} \\ 0 & \text{otherwise} \end{cases}$$

Assumption 1: Primitives are differentiable.

2. Conditionals

2. Conditionals

```
let  x = sample
in   if x < 0 then
       score(0)
     else
       score(1)
```

2. Conditionals

```
let  x = sample
in   if x < 0 then
        score(0)
     else
        score(1)
```

$$\mathrm{weight}([s]) = \begin{cases} 0 & \text{if } s < 0 \\ 1 & \text{otherwise} \end{cases}$$

2. Conditionals

```
let  x = sample
in   if x < 0 then
       score(0)
     else
       score(1)
```

not differentiable at 0

$$\mathrm{weight}([s]) = \begin{cases} 0 & \text{if } s < 0 \\ 1 & \text{otherwise} \end{cases}$$

✓

2. Conditionals

```
let  x = sample
in   if f(x) < 0 then
       score(0)
     else
       score(1)
```

2. Conditionals

```
let  x = sample
in   if f(x) < 0 then
        score(0)
     else
        score(1)
```

$$\text{weight}([s]) = \begin{cases} 0 & \text{if } \mathbf{f(s)} < 0 \\ 1 & \text{otherwise} \end{cases}$$

2. Conditionals

```
let x = sample
in  if f(x) < 0 then
      score(0)
    else
      score(1)
```

not differentiable at $\partial f^{-1}(-\infty, 0)$

$$\mathrm{weight}([s]) = \begin{cases} 0 & \text{if } \mathbf{f}(\mathbf{s}) < 0 \\ 1 & \text{otherwise} \end{cases}$$

2. Conditionals

```
let  x = sample
in   if f(x) < 0 then
        score(0)
     else
        score(1)
```

not differentiable at $\partial f^{-1}(-\infty, 0)$

$$\text{weight}([s]) = \begin{cases} 0 & \text{if } \mathbf{f(s)} < 0 \\ 1 & \text{otherwise} \end{cases}$$

Assumption 2: $\partial f^{-1}(-\infty, 0)$ has measure 0 for primitives f.

2. Conditionals

```
let  x = sample
in   if g(f(x)) < 0 then
       score(0)
     else
       score(1)
```

Assumption 2: $\partial f^{-1}(-\infty, 0)$ has measure 0 for primitives f.

2. Conditionals

```
let x = sample
in  if g(f(x)) < 0 then
      score(0)
    else
      score(1)
```

Assumption 2: $\partial f^{-1}(-\infty, 0)$ has measure 0 for primitives f.

Assumption 3: Primitives are closed under composition.

3. Recursion

3. Recursion

```
let rec enumQ y = ...
```

3. Recursion

```
let rec enumQ y = ...
```

return 1 if y is rational,
diverge otherwise

3. Recursion

```
let rec enumQ y = ...
```

return 1 if y is rational,
diverge otherwise

```
let x = sample
in  score(enumQ(x))
```

3. Recursion

```
let rec enumQ y = ...

let x = sample
in  score(enumQ(x))
```

return 1 if y is rational,
diverge otherwise

$$\text{weight}([s]) = \begin{cases} 1 & \text{if } s \in \mathbb{Q} \\ 0 & \text{otherwise} \end{cases}$$

3. Recursion

```
let rec enumQ y = ...
```

return 1 if y is rational,
diverge otherwise

```
let x = sample
in  score(enumQ(x))
```

not differentiable anywhere

$$\mathrm{weight}([s]) = \begin{cases} 1 & \text{if } s \in \mathbb{Q} \\ 0 & \text{otherwise} \end{cases}$$

Part III:
Symbolic Execution

$$\text{if sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1})$$

$$\text{if sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1})$$

$$\text{weight}([0.1]) = 0$$

$$\text{if sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1})$$

$$\text{weight}([0.1]) = 0$$

$$\text{weight}([0.11]) = ???$$

*Examine weight for **all** samples consistent with a **branch** at once.*

Symbolic Execution

Symbolic Execution

$$\langle \text{if sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [] \rangle$$

Symbolic Execution

$$\langle \text{if } \textcolor{red}{\text{sample}} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [] \rangle$$

Symbolic Execution

$$\rightarrow \langle \text{if } \underline{\mathbf{0.1}} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [\mathbf{0.1}] \rangle$$

Symbolic Execution

$$\rightarrow \langle \mathsf{if}\ \underline{\mathbf{0.1}} < \underline{0.5}\ \mathsf{then\ score}(\underline{0})\ \mathsf{else\ score}(\underline{1}), 1, [\mathbf{0.1}]\rangle$$

$$\Rightarrow \langle\!\langle \mathsf{if}\ \ \alpha < \underline{0.5}\ \mathsf{then\ score}(\underline{0})\ \mathsf{else\ score}(\underline{1}), 1, (\mathbf{0}, \mathbf{1})\rangle\!\rangle$$

sampling variable

Symbolic Execution

$$\to \langle \text{if } \underline{\mathbf{0.1}} < \underline{\mathbf{0.5}} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [0.1] \rangle$$

$$\Rightarrow \langle\!\langle \text{if } \alpha < \underline{\mathbf{0.5}} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, (0, 1) \rangle\!\rangle$$

sampling variable

Symbolic Execution

$$\to \langle \mathsf{score}(\underline{0}), 1, [0.1] \rangle$$

$$\Rightarrow \langle\!\langle \mathsf{score}(\underline{0}), 1, \mathbf{(0, 0.5)} \rangle\!\rangle$$

Symbolic Execution

$$\to \langle \mathsf{score}(\underline{\mathbf{0}}), 1, [0.1] \rangle$$

$$\Rightarrow \langle\!\langle \mathsf{score}(\underline{\mathbf{0}}), 1, (0, 0.5) \rangle\!\rangle$$

Symbolic Execution

$$\to \langle \underline{0}, \mathbf{0}, [0.1] \rangle$$

$$\Rightarrow \langle\!\langle \underline{0}, \mathbf{0}, (0, 0.5) \rangle\!\rangle$$

Symbolic Execution

$$\to \langle \underline{0}, \mathbf{0}, [0.1] \rangle$$

$$\Rightarrow \langle\!\langle \underline{0}, \mathbf{0}, (0, 0.5) \rangle\!\rangle$$

$$\text{weight}([s]) = 0 \qquad \text{whenever } s < 0.5$$

term accumulated weight trace of samples

Configuration: $\langle M, w, \mathbf{s} \rangle$

term accumulated weight trace of samples

Configuration: $\langle\ M\ ,\ w\ ,\ \mathbf{s}\ \rangle$

Symbolic Configuration: $\langle\!\langle\ \mathcal{M}\ ,\ w\ ,\ U\ \rangle\!\rangle$

term
accumulated weight
trace of samples

Configuration: $\langle \underset{\text{term}}{M}, \underset{\text{accumulated weight}}{w}, \underset{\text{trace of samples}}{\mathbf{s}} \rangle$

Symbolic Configuration: $\langle\!\langle \mathcal{M}, w, U \rangle\!\rangle$

symbolic term

- $\mathcal{M}$: *symbolic* term
  - sampling variables $\alpha_1, \ldots, \alpha_n$
  - (delayed operations)

$$\text{Configuration:}\quad \langle \underset{\text{}}{M} , w , \mathbf{s} \rangle$$

term

accumulated weight

trace of samples

$$\text{Symbolic Configuration:}\quad \langle\!\langle \mathcal{M} , w , U \rangle\!\rangle$$

symbolic term

set of traces in branch

- $\mathcal{M}$: *symbolic* term
  - sampling variables $\alpha_1, \ldots, \alpha_n$
  - (delayed operations)
- $U \subseteq (0, 1)^n$

term accumulated weight trace of samples

Configuration: $\langle M , w , \mathbf{s} \rangle$

Symbolic Configuration: $\langle\!\langle \mathcal{M} , w , U \rangle\!\rangle$

symbolic term weight *function* *set* of traces in branch

- $\mathcal{M}$: *symbolic* term
  - sampling variables $\alpha_1, \ldots, \alpha_n$
  - (delayed operations)
- $U \subseteq (0,1)^n$
- $w : U \to \mathbb{R}_{\geq 0}$

Define Symbolic Execution to Closely Mirror Operational Semantics

Define S… or Operati…

Now, we introduce the following rules for ***symbolic redex contractions***:

$$
\begin{aligned}
\langle\!\langle (\lambda y.\, \mathcal{M})\, \mathcal{V}, w, U \rangle\!\rangle &\Rightarrow \langle\!\langle \mathcal{M}[\mathcal{V}/y], w, U \rangle\!\rangle \\
\langle\!\langle \underline{f}(\mathcal{V}_1, \ldots, \mathcal{V}_\ell), w, U \rangle\!\rangle &\Rightarrow \langle\!\langle \boxed{f}(\mathcal{V}_1, \ldots, \mathcal{V}_\ell), w, \mathsf{dom}\, \|\boxed{f}(\mathcal{V}_1, \ldots, \mathcal{V}_\ell)\| \cap U \rangle\!\rangle \\
\langle\!\langle \mathsf{Y}(\lambda y.\, \mathcal{M}), w, U \rangle\!\rangle &\Rightarrow \langle\!\langle \lambda z.\, \mathcal{M}\,[\mathsf{Y}(\lambda y.\, \mathcal{M})/y]\, z, w, U \rangle\!\rangle \\
\langle\!\langle \mathsf{if}(\mathcal{V} \leq 0, \mathcal{M}, \mathcal{N}), w, U \rangle\!\rangle &\Rightarrow \langle\!\langle \mathcal{M}, w, \|\mathcal{V}\|^{-1}(-\infty, 0] \cap U \rangle\!\rangle \\
\langle\!\langle \mathsf{if}(\mathcal{V} \leq 0, \mathcal{M}, \mathcal{N}), w, U \rangle\!\rangle &\Rightarrow \langle\!\langle \mathcal{N}, w, \|\mathcal{V}\|^{-1}(0, \infty) \cap U \rangle\!\rangle \\
\langle\!\langle \mathsf{sample}, w, U \rangle\!\rangle &\Rightarrow \langle\!\langle \alpha_{n+1}, w', U' \rangle\!\rangle \qquad (U \subseteq \mathbb{R}^m \times \mathbb{S}_n) \\
\langle\!\langle \mathsf{score}(\mathcal{V}), w, U \rangle\!\rangle &\Rightarrow \langle\!\langle \mathcal{V}, \|\mathcal{V}\| \cdot w, \|\mathcal{V}\|^{-1}[0, \infty) \cap U \rangle\!\rangle
\end{aligned}
$$

In the rule for sample, $U' := \{(\boldsymbol{r}, \boldsymbol{s} +\!\!+ [s']) \mid (\boldsymbol{r}, \boldsymbol{s}) \in U \wedge s' \in (0,1)\}$ and $w'(\boldsymbol{r}, \boldsymbol{s} +\!\!+ [s']) := w(\boldsymbol{r}, \boldsymbol{s})$; in the rule for $\mathsf{score}(\mathcal{V})$, $(\|\mathcal{V}\| \cdot w)(\boldsymbol{r}, \boldsymbol{s}) := \|\mathcal{V}\|\,(\boldsymbol{r}, \boldsymbol{s}) \cdot w(\boldsymbol{r}, \boldsymbol{s})$.

The rules are designed to closely mirror their concrete counterparts. Crucially, the rule for sample introduces a "fresh" sampling variable, and the two rules for conditionals split the last component $U \subseteq \mathbb{R}^m \times \mathbb{S}_n$ according to whether $\|\mathcal{V}\|\,(\boldsymbol{r}, \boldsymbol{s}) \leq 0$ or $\|\mathcal{V}\|\,(\boldsymbol{r}, \boldsymbol{s}) > 0$. The "delay" contraction (second rule) is introduced for a technical reason: ultimately, to enable item 1 (Soundness). Otherwise it is, for example, unclear whether $\lambda y.\, \alpha_1 + \underline{1}$ should correspond to $\lambda y.\, \underline{0.5} + \underline{1}$ or $\lambda y.\, \underline{1.5}$ for $s_1 = 0.5$.

Finally we lift this to arbitrary symbolic terms using the eli…
symbolic evaluation contexts:

(Soundness) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle\ \mathcal{V}\ , w, U \rangle\!\rangle$

(Soundness) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$

symbolic value

(Soundness) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle\ \mathcal{V}\ , w, U \rangle\!\rangle$ then

symbolic value

- $w_{|\mathbf{U}} = \mathrm{weight}_{|\mathbf{U}}$

(Soundness) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$ then

symbolic value

- $w_{|\mathbf{U}} = \text{weight}_{|\mathbf{U}}$

(Completeness) If M terminates on s

(Soundness) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$ then

symbolic value

- $w_{|\mathbf{U}} = \text{weight}_{|\mathbf{U}}$

(Completeness) If M terminates on s then $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$ s.t. $\mathbf{s} \in \mathbf{U}$.

(Soundness) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$ then

symbolic value

- $w_{|\mathbf{U}} = \mathrm{weight}_{|\mathbf{U}}$

(Completeness) If M terminates on s then $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$ s.t. $\mathbf{s} \in \mathbf{U}$.

(Invariance) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{N}, w, U \rangle\!\rangle$

(Soundness) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$ then

symbolic value

- $w_{|\mathbf{U}} = \text{weight}_{|\mathbf{U}}$

(Completeness) If M terminates on s then $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$ s.t. $\mathbf{s} \in \mathbf{U}$.

(Invariance) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{N}, w, U \rangle\!\rangle$ then

- w is *differentiable*
- boundary of U has measure 0

assumptions on primitives

(Soundness) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$ then

symbolic value

- $w_{|\mathbf{U}} = \text{weight}_{|\mathbf{U}}$

(Completeness) If M terminates on s then $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{V}, w, U \rangle\!\rangle$ s.t. $\mathbf{s} \in \mathbf{U}$.

(Invariance) If $\langle\!\langle M, 1, [] \rangle\!\rangle \Rightarrow^* \langle\!\langle \mathcal{N}, w, U \rangle\!\rangle$ then

- w is *differentiable*
- boundary of U has measure 0

assumptions on primitives

Theorem

The weight-function is differentiable for almost all terminating traces.

Conclusion

This talk:

- conditions on primitive operations
- *symbolic* execution as a proof technique

Conclusion

This talk:

- conditions on primitive operations
- *symbolic* execution as a proof technique
- differentiability of *weight*-function on almost all *terminating* traces

Conclusion

This talk:

- conditions on primitive operations
- *symbolic* execution as a proof technique
- differentiability of *weight*-function on almost all *terminating* traces

Also in the paper:

- extension to *almost all* traces assuming *almost-sure termination*
- *value*-function

Conclusion

This talk:

- conditions on primitive operations
- *symbolic* execution as a proof technique
- differentiability of *weight*-function on almost all *terminating* traces

Also in the paper:

- extension to *almost all* traces assuming *almost-sure termination*
- *value*-function

Future directions:

- applications in *inference* algorithms

Conclusion

Densities of almost surely terminating probabilistic programs are differentiable almost everywhere

Carol Mak   Luke Ong   Hugo Paquet

Dominik Wagner

dominik.wagner@cs.ox.ac.uk

backup slides