

# Densities of Almost Surely Terminating Probabilistic Programs are Differentiable Almost Everywhere

Carol Mak Luke Ong Hugo Paquet

Dominik Wagner



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:

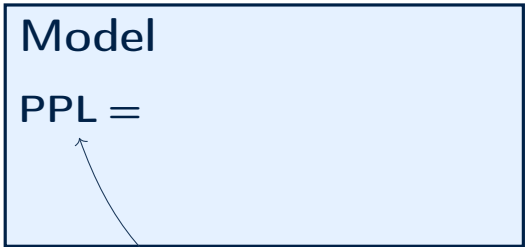
*Make **Bayesian** Machine Learning more accessible\**

*\* to domain experts with basic programming skills*

- ▶ separate modelling from inference

**Model**

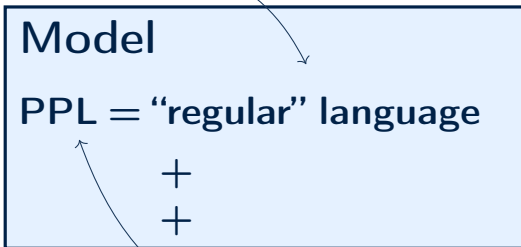
**Inference**



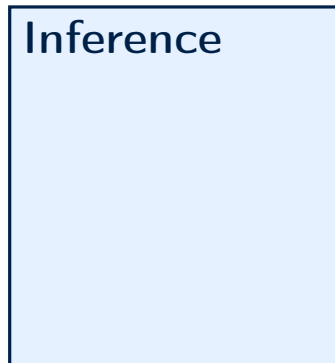
probabilistic programming language



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

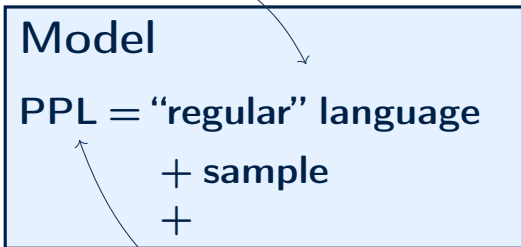


probabilistic programming language

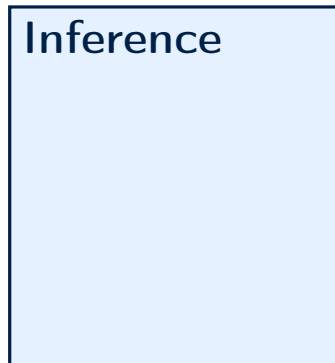




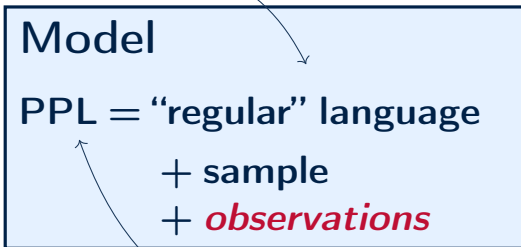
*e.g. conditionals, recursion,  
higher-order functions*



probabilistic programming language



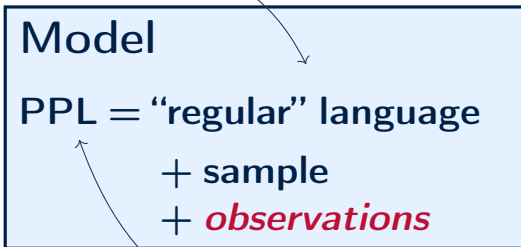
*e.g. conditionals, recursion,  
higher-order functions*



probabilistic programming language



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

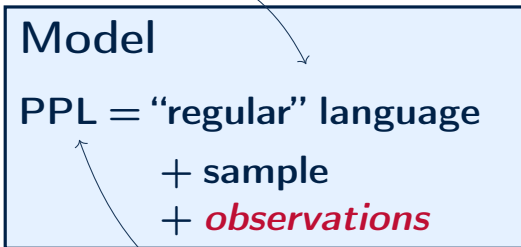


probabilistic programming language

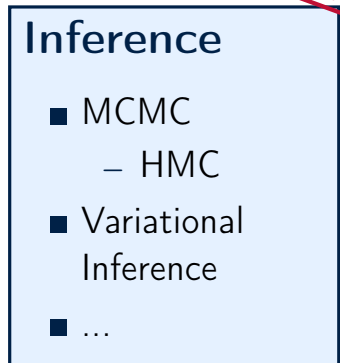
Inference

**hard!**

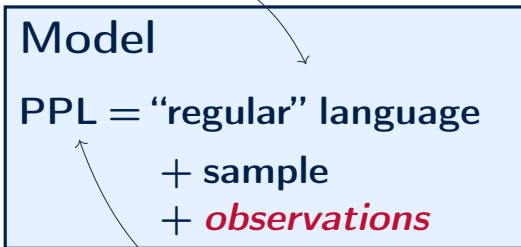
*e.g. conditionals, recursion,  
higher-order functions*



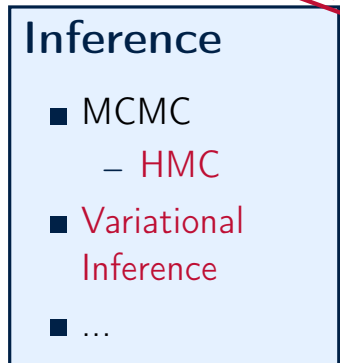
probabilistic programming language



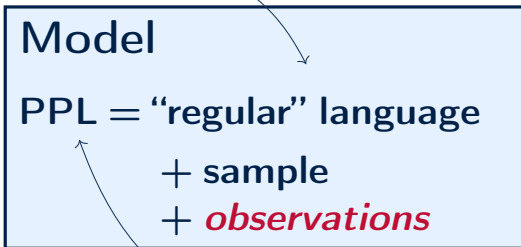
*e.g. conditionals, recursion,  
higher-order functions*



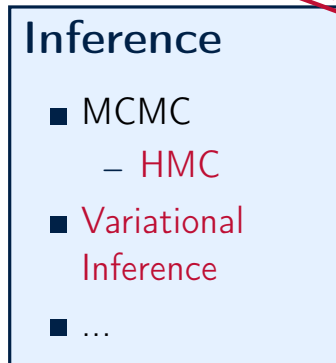
probabilistic programming language



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

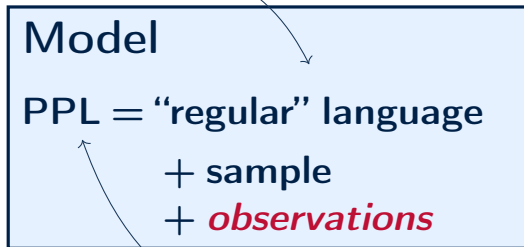


probabilistic programming language



→ *exploit gradients*

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



probabilistic programming language

*correct?*

A light blue rectangular box with a dark blue border. It contains the text 'Inference' at the top, followed by a list: '■ MCMC', '– HMC', '■ Variational Inference', and '■ ...'. A red dashed double-headed arrow labeled 'correct?' connects the Model box to this box. A red stamp with the word 'hard!' is in the top right corner. Below the box is the text '→ exploit gradients'.

→ *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* ← (probability non-termination is 0)
2. the primitive operations are well-behaved.

# 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)
2. the primitive operations are well-behaved.

- 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* ← (probability non-termination is 0)
2. the primitive operations are well-behaved.

- 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* ← (probability non-termination is 0)
2. the primitive operations are well-behaved.

- 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* ← (probability non-termination is 0)
2. the primitive operations are well-behaved.

- 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, ...]

if sample < 0.5 then score(0) else score(1)

if sample < 0.5 then score(0) else score(1)

---

weight([0.1]) = 0

if sample < 0.5 then score(0) else score(1)

---

weight([0.1]) = 0

weight([0.7]) = 1

# Operational Semantics

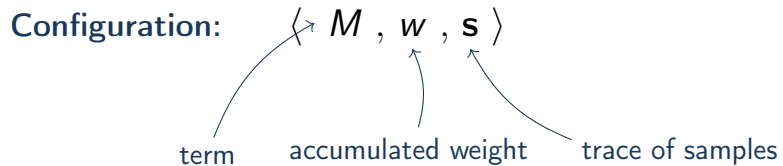
Configuration:



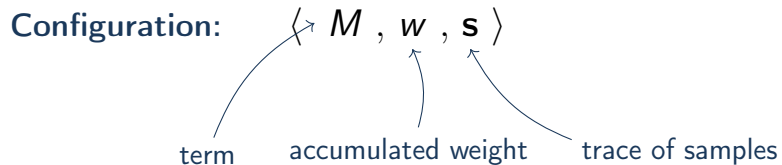
# Operational Semantics

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

# Operational Semantics

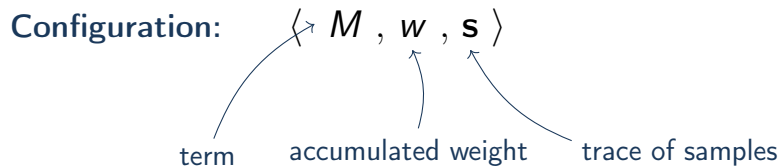


# Operational Semantics



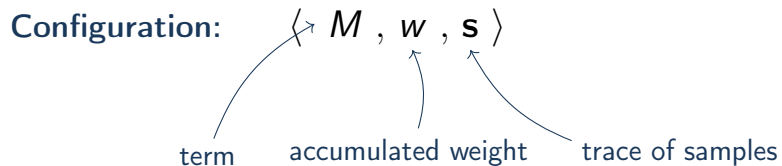
$\langle \text{if sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [] \rangle$

# Operational Semantics



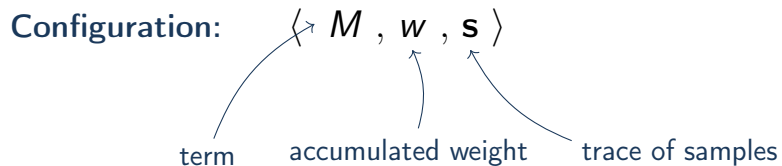
$\langle \text{if } \text{sample} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [] \rangle$

# Operational Semantics



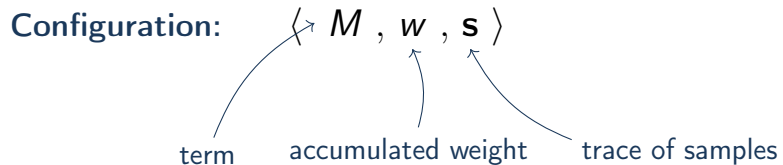
$\rightarrow \langle \text{if } \underline{0.1} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [\underline{0.1}] \rangle$

# Operational Semantics



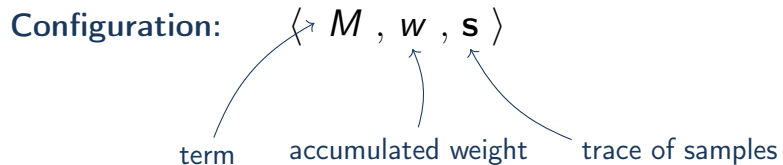
→  $\langle \text{if } \underline{0.1} < \underline{0.5} \text{ then score}(\underline{0}) \text{ else score}(\underline{1}), 1, [0.1] \rangle$

# Operational Semantics



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

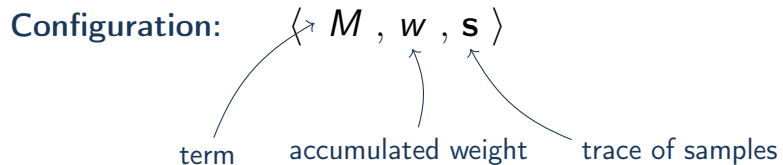
# Operational Semantics



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

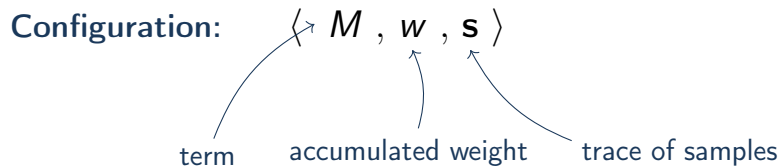


# Operational Semantics



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

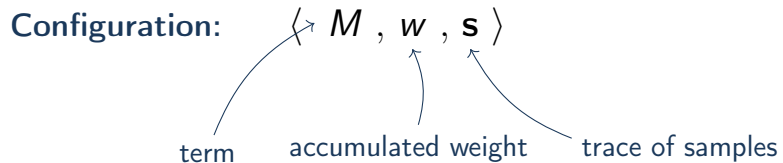
# Operational Semantics



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

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

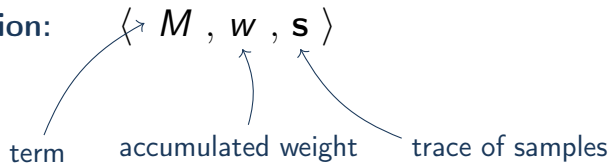
# Operational Semantics



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

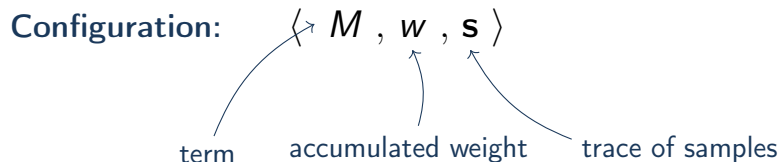
# Operational Semantics

Configuration:



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

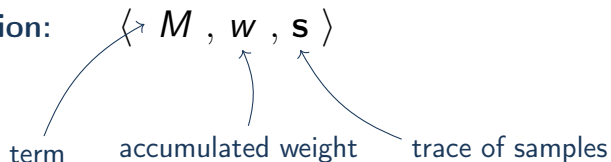
# Operational Semantics



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

# Operational Semantics

Configuration:



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

*differentiable  
almost everywhere?*

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( $\chi_{\mathbb{Q}}$ (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( $\chi_{\mathbb{Q}}$ (x))
```

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

not differentiable  
anywhere

# 1. Primitive Operations

```
let x = sample  
in score( $\chi_{\mathbb{Q}}$ (x))
```

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

not differentiable  
anywhere

**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)
```

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



## 2. Conditionals

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

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

not differentiable  
at 0



## 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)
```

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

## 2. Conditionals

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

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

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

## 2. Conditionals

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

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

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

**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



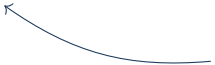
### 3. Recursion

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

### 3. Recursion

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

return 1 if y is rational,  
diverge otherwise




### 3. Recursion

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

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

return 1 if y is rational,  
diverge otherwise




### 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 = ...
```

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

return 1 if y is rational,  
diverge otherwise



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

not differentiable  
anywhere

# Part III: Symbolic Execution

if sample < 0.5 then score(0) else score(1)

if sample < 0.5 then score(0) else score(1)

---

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



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

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

# Symbolic Execution

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

# Symbolic Execution

→ ⟨if **0.1** < 0.5 then score(0) else score(1), 1, [**0.1**]⟩

---

⇒ ⟨⟨if  **$\alpha$**  < 0.5 then score(0) else score(1), 1, (**0, 1**)⟩⟩

← sampling variable

# Symbolic Execution

→ ⟨if 0.1 < 0.5 then score(0) else score(1), 1, [0.1]⟩

---

⇒ ⟨⟨if  $\alpha$  < 0.5 then score(0) else score(1), 1, (0, 1)⟩⟩

← sampling variable



# Symbolic Execution

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

---

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

# Symbolic Execution

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

---

$\Rightarrow \langle\langle \text{score}(\underline{0}), 1, (0, 0.5) \rangle\rangle$

# Symbolic Execution

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

---

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

# Symbolic Execution

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

---

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

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

whenever  $s < 0.5$

term      accumulated weight      trace of samples

Configuration:  $\langle M, w, s \rangle$

The diagram shows three labels at the top: 'term', 'accumulated weight', and 'trace of samples'. Below them is the text 'Configuration:  $\langle M, w, s \rangle$ '. Three curved arrows point from the labels to the elements of the tuple: 'term' points to  $M$ , 'accumulated weight' points to  $w$ , and 'trace of samples' points to  $s$ .

term      accumulated weight      trace of samples

Configuration:  $\langle M, w, s \rangle$

---

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

term      accumulated weight      trace of samples

Configuration:  $\langle M, w, s \rangle$

---

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

*symbolic* term

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

term      accumulated weight      trace of samples

Configuration:  $\langle M, w, s \rangle$

---

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

*symbolic* term      *set* of traces in branch

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



term      accumulated weight      trace of samples

Configuration:  $\langle M, w, s \rangle$

---

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  $\alpha_1, \dots, \alpha_n$
  - (delayed operations)
- $U \subseteq (0, 1)^n$
- $w : U \rightarrow \mathbb{R}_{\geq 0}$

*Define Symbolic Execution to Closely Mirror  
Operational Semantics*

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

$$\begin{aligned} \langle\langle \lambda y. \mathcal{M} \mathcal{V}, w, U \rangle\rangle &\Rightarrow \langle\langle \mathcal{M}[\mathcal{V}/y], w, U \rangle\rangle \\ \langle\langle \underline{f}(\mathcal{V}_1, \dots, \mathcal{V}_\ell), w, U \rangle\rangle &\Rightarrow \langle\langle \underline{f}(\mathcal{V}_1, \dots, \mathcal{V}_\ell), w, \text{dom } \|\underline{f}\|(\mathcal{V}_1, \dots, \mathcal{V}_\ell) \cap U \rangle\rangle \\ \langle\langle \Upsilon(\lambda y. \mathcal{M}), w, U \rangle\rangle &\Rightarrow \langle\langle \lambda z. \mathcal{M} [\Upsilon(\lambda y. \mathcal{M})/y] z, w, U \rangle\rangle \\ \langle\langle \text{if}(\mathcal{V} \leq 0, \mathcal{M}, \mathcal{N}), w, U \rangle\rangle &\Rightarrow \langle\langle \mathcal{M}, w, \|\mathcal{V}\|^{-1}(-\infty, 0] \cap U \rangle\rangle \\ \langle\langle \text{if}(\mathcal{V} \leq 0, \mathcal{M}, \mathcal{N}), w, U \rangle\rangle &\Rightarrow \langle\langle \mathcal{N}, w, \|\mathcal{V}\|^{-1}(0, \infty) \cap U \rangle\rangle \\ \langle\langle \text{sample}, w, U \rangle\rangle &\Rightarrow \langle\langle \alpha_{n+1}, w', U' \rangle\rangle \\ \langle\langle \text{score}(\mathcal{V}), w, U \rangle\rangle &\Rightarrow \langle\langle \mathcal{V}, \|\mathcal{V}\| \cdot w, \|\mathcal{V}\|^{-1}[0, \infty) \cap U \rangle\rangle \end{aligned}$$

$$(U \subseteq \mathbb{R}^m \times \mathcal{S}_n)$$

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


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  $U \subseteq \mathbb{R}^m \times \mathcal{S}_n$  according to whether  $\|\mathcal{V}\|(\mathbf{r}, \mathbf{s}) \leq 0$  or  $\|\mathcal{V}\|(\mathbf{r}, \mathbf{s}) > 0$ . 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 following symbolic evaluation contexts:

Define S  
Operati

**(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

■  $w|_{\mathbf{U}} = \text{weight}|_{\mathbf{U}}$

← symbolic value

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

■  $w|_{\mathbf{U}} = \text{weight}|_{\mathbf{U}}$

← symbolic value

**(Completeness)** If  $M$  terminates on  $s$

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

■  $w|_{\mathbf{U}} = \text{weight}|_{\mathbf{U}}$

 symbolic value

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



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

■  $w|_{\mathbf{U}} = \text{weight}|_{\mathbf{U}}$

 symbolic value

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

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

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

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

← symbolic value

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

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

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

← symbolic value

(Completeness) If  $M$  terminates on  $\mathbf{s}$  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  $\langle\langle M, 1, [] \rangle\rangle \Rightarrow^* \langle\langle \mathcal{X}, w, U \rangle\rangle$  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