ωPAP Spaces
reasoning denotationally about differentiable & probabilistic programs

Mathieu Huot*, Alexander Lew*, Vikash Mansinghka, Sam Staton

* Equal Contribution
Differentiable Programming

Torch  TensorFlow  Enzyme  JAX  Taichi  MXNet

Probabilistic Programming

Gen  SPPL  Birch  Stan  ProbTorch  Pyro
3D Scene Understanding (Gothoskar et al. 2021)

Online Goal Inference (Zhi-Xuan et al. 2020)

AI / Robotics

Data Deduplication and Cleaning (Lew et al. 2021)

Time Series Forecasting (Saad et al. 2023)

Data Analysis

LHC Particle Mass Inference (Baydin et al. 2019)

Statistical Phylogenetics (Ronquist et al. 2021)

Natural Sciences
3D Scene Understanding (Gothoskar et al. 2021)

Online Goal Inference (Zhi-Xuan et al. 2020)

AI / Robotics

Data Deduplication and Cleaning (Lew et al. 2021)

Time Series Forecasting (Saad et al. 2023)

Data Analysis

LHC Particle Mass Inference (Baydín et al. 2019)

Statistical Phylogenetics (Ronquist et al. 2021)

Natural Sciences

Modeling Languages
3D Scene Understanding (Gothoskar et al. 2021)

Data Deduplication and Cleaning (Lew et al. 2021)

LHC Particle Mass Inference (Baydin et al. 2019)

Online Goal Inference (Zhi-Xuan et al. 2020)

Time Series Forecasting (Saad et al. 2023)

Recursive Phylogeny Simulator

AI / Robotics

Data Analysis

Natural Sciences
3D Scene Understanding (Gothoskar et al. 2021)

Online Goal Inference (Zhi-Xuan et al. 2020)

Data Deduplication and Cleaning (Lew et al. 2021)

Time Series Forecasting (Saad et al. 2023)

LHC Particle Mass Inference (Baydìn et al. 2019)

Statistical Phylogenetics (Ronquist et al. 2021)

AI / Robotics

Data Analysis

Natural Sciences

Automation
3D Scene Understanding (Gothoskar et al. 2021)

Data Deduplication and Cleaning (Lew et al. 2021)

LHC Particle Mass Inference (Baydin et al. 2019)

Online Goal Inference (Zhi-Xuan et al. 2020)

Time Series Forecasting (Saad et al. 2023)

Statistical Phylogenetics (Ronquist et al. 2021)

AI / Robotics

Data Analysis

Natural Sciences
Outline

- **What do we need semantics for?**
- Key challenge
- Our approach
- Main results
Correctness of Automation

Input Program
\[ t : \mathbb{R} \rightarrow \mathbb{R} \]

Objective Function
\[ \mathcal{L}(\theta) = [t](\theta) \]

AD Program Transformation

Output Program
\[ s : \mathbb{R} \rightarrow \mathbb{R} \]

Objective Gradient
\[ \mathcal{L}'(\theta) = \frac{d}{d\theta} [t](\theta) = [s](\theta) \]
Reasoning about Coverage
Reasoning about Coverage

\[ \text{fair\_flip} = \text{do} \]
\[ \quad u \leftarrow \text{sample} \]
\[ \quad \text{if } u < 0.5 \text{ then} \]
\[ \quad \quad \text{return } 1 \]
\[ \quad \text{else} \]
\[ \quad \quad \text{return } 0 \]
std_normal = \( \textbf{do} \)
\( u_1 \leftarrow \text{sample} \)
\( u_2 \leftarrow \text{sample} \)
\( \text{let } r = \sqrt{-2 \ln(u_1)} \)
\( \text{let } t = 2\pi u_2 \)
\( \text{return } (r\cos(t), r\sin(t)) \)
Reasoning about Coverage

uniform_band = do
  x <- sample
  y <- sample
  return (x, x + y)
Reasoning about Coverage

mcdonalds = do
  x <- sample
  let t = 2*pi*x-pi
  return (t*cos(t), t*sin(t))
Reasoning about Coverage

bowtie_heart = do
  x <- std_normal
  let t = x * pi / 2
  r <- if |t| < 0.4 then
       std_normal
     else
       return t
  return (r*cos(t), r*sin(t))
Reasoning about Coverage

$S^c$
Complement of Sierpinski Triangle

$S$
Sierpinski Triangle
Reasoning about Coverage

Can my language express all models users care about?

Do my interfaces and automation algorithms support any program the user could throw at me?
Reasoning about Coverage

We can use semantics to reason about the **definable measures**

\[ \{\llbracket e \rrbracket \mid e : M \mathbb{R}^n\} \]
Outline

• What do we need semantics for?
• **Key challenge**
• Our approach
• Main results
Key challenge: the Goldilocks problem

- \(\omega_{QBS}\): Admits pathological examples
- \(\omega_{Diff}\): Can’t model non-smooth primitives
- \(\omega_{PAP}\): This Work: Just Right?
Outline

• What do we need semantics for?
• Key challenge
• **Our approach**
• Main results
Concrete categories and higher-order recursion

With applications including probability, differentiability, and full abstraction

Cristina Matache
University of Oxford
Department of Computer Science
Oxford, UK

Sean Moss
University of Oxford
Department of Computer Science
Oxford, UK

Sam Staton
University of Oxford
Department of Computer Science
Oxford, UK

\[ \text{Smooth } \mathbb{R} \to \mathbb{R} \text{ functions} \]

\[ \text{Measurable } \mathbb{R} \to \mathbb{R} \text{ functions} \]

\[ \text{PAP } \mathbb{R} \to \mathbb{R} \text{ functions} \]

\[ \omega \text{Diff} \]

\[ \omega \text{QBS} \]

\[ \omega \text{PAP} \]
PAP functions

Piecewise analytic under analytic partition (PAP)
Lee et al. (2020)

A function is PAP if its domain can be partitioned into countably many analytic sets, restricted to each of which it is equal to some analytic function.

$$f(x) = \begin{cases} f_1(x) & \text{if } x \in A_1 \\ f_2(x) & \text{if } x \in A_2 \\ f_3(x) & \text{if } x \in A_3 \\ \vdots & \end{cases}$$

real-analytic functions defined on open $U_i \supseteq A_i$

analytic sets (finite intersections of sets of the form $\{x \in U \mid g(x) \leq 0\}$, for $U$ open and $g$ analytic) that partition $A$
Intuition: what functions are PAP?

Includes (for example):

- All primitives in `numpy`
  - arithmetic, trigonometry, comparison, floor, ceil, …
- All functions in `SpecialFunctions.jl`
- All functions in Haskell’s numeric typeclasses

Does not include (for example):

- Indicator function for Cantor set
- Measurable bijections betw. $\mathbb{R}$, $\mathbb{R}^2$
- Hilbert curve
A **PAP space** $X$ is:
- a set $|X|$, together with
- for each $n$ and each set $U \in \mathbb{R}^n = \bigcup_{i \in \mathbb{N}} A_i$, a set $\mathcal{P}_X^U \subseteq |X|^U$ of plots in $X$, satisfying several closure properties.

A function $f : |X| \to |Y|$ is a **PAP morphism** if, whenever $\phi \in \mathcal{P}_X^U$, the composition $f \circ \phi \in \mathcal{P}_Y^U$. 
An \( \omega \text{PAP} \) space \( X \) is:

- an \( \omega \text{cpo} \ |X| \),
- for each \( n \) and each set \( U \in \mathbb{R}^n = U_{i \in \mathbb{N}} A_i \), a set \( \mathcal{P}^U_X \subseteq |X|^U \) of Scott-continuous plots in \( X \), satisfying several closure properties

A Scott-continuous function \( f : |X| \to |Y| \) is an \( \omega \text{PAP} \) morphism if, whenever \( \phi \in \mathcal{P}^U_X \), the composition \( f \circ \phi \in \mathcal{P}^U_Y \).
Language (CBV PCF with $\mathbb{R}$ & probability)

$$\tau ::= 1 \mid \mathbb{R}^k \mid \mathbb{B} \mid \tau_1 \times \tau_2 \mid \tau_1 \rightarrow \tau_2 \mid M \tau$$

$$e ::= c \mid x \mid e_1 e_2 \mid \textbf{if} \ e_1 \ \textbf{then} \ e_2 \ \textbf{else} \ e_3 \mid (e_1, e_2) \mid \pi_1 e \mid \pi_2 e \mid \textbf{return} \ e$$
$$\mid \lambda x:\tau. \ e \mid \mu f:\tau_1 \rightarrow \tau_2. \lambda x:\tau_1. \ e \mid \textbf{sample} \mid \textbf{score} \ e \mid \textbf{do} \ \{m\}$$

$$m ::= e \mid x \leftarrow e; m$$
Can’t model non-smooth primitives

This Work: Just Right?
Outline

• What do we need semantics for?
• Key challenge
• Our approach
• Main results
... for deterministic programs
Deterministic programs are almost-everywhere differentiable

**Theorem.** If $\vdash e : \mathbb{R}^m \to \mathbb{R}^n$, then $\lceil e \rceil$ is differentiable at almost every input on which it is defined.
AD is almost everywhere correct

**Theorem.** If $\vdash e : \mathbb{R}^m \to \mathbb{R}^n$, then $\llbracket e \rrbracket$ is defined on the same inputs as $\llbracket AD\{e\} \rrbracket$, and for almost all such inputs $x$, $\llbracket AD\{e\} \rrbracket(x)$ is the derivative of $\llbracket e \rrbracket$ at $x$. 
Randomly initialized gradient descent converges, even when using AD

**Theorem.** Let $\vdash e : \mathbb{R}^m \to \mathbb{R}$, with $\|e\|$ total, $L$-smooth, and bounded below. Let $(\epsilon, x_0) \sim \mu$ for any $\mu$ supported on $(0, \frac{2}{L}) \times \mathbb{R}^m$. Then letting $x_t := x_{t-1} - \epsilon \cdot \|AD\{e\}\!(x_{t-1})$,

$$\lim_{t \to \infty} \nabla\|e\|_2(x_t) = 0 \text{ with probability 1.}$$
Gradient descent with pre-set learning rate $\epsilon$ may diverge when using AD

\[
[P] = \lambda x \frac{x^2}{2\epsilon}
\]

Gradient descent diverges to $-\infty$
... for probabilistic programs
Almost-surely terminating probabilistic programs have almost-everywhere differentiable weight functions

**Theorem.** If $\vdash e : M X$ and $\llbracket e \rrbracket$ almost surely halts, then $\text{wt} \llbracket e \rrbracket$ is differentiable almost everywhere.
All probabilistic programs on $\mathbb{R}^n$ are supported on a countable union of manifolds

**Theorem.** If $\vdash e : M \mathbb{R}^n$, then $[e]$ is supported on a countable union of smooth submanifolds of $\mathbb{R}^n$. 