# On the Versatility of Logical Relations

Joint work with Gilles Barthe, Ugo Dal Lago, Francesco Gavazzo


Raphaëlle Crubillé

IMDEA


January 30, 2020

# Outline

Containment Theorems by way of open logical relations

Correctness for Automatic Differentiation Algorithms

Soundness of a refinement type system for local continuity

Conclusion

# (First-Order) Containment in Principle

## A (terminating) programming language built from:

- real numbers as data type;
- a family $\mathcal{F}$ of primitive functions $\mathbb{R}^n \to \mathbb{R}^m$;
- programming constructs: variables assignments, `if`, `while`...

## Program interpretation:

real-valued functions $\quad [\![M]\!] : \mathbb{R}^n \to \mathbb{R}^m$

## Definition (Containment Property)

We suppose a (compositionnal) predicate $\mathcal{P}$ on functions such that $\forall f \in \mathcal{F}$, $\mathcal{P}(f)$.
$\mathcal{P}$ is **contained** when: $\quad \forall M$ a program $, \mathcal{P}([\![M]\!])$*holds*.

# A Simple Example: (Global) Continuity.

$$\mathcal{P} = \textbf{Cont} := \{ f : \mathbb{R}^n \to \mathbb{R}^m \mid f \text{ is continuous } \}.$$

**Fact**

**Cont** is contained for a restricted language:

- sequencing, variable assignment;
- no `if`, no `while`

**Proof.**

The predicate **Cont** is compositionnal. □

**Example**

$$M = x := x + y; x := 3 + x^2; y := y + 1$$
$$[\![M]\!] : (x, y) \in \mathbb{R}^2 \mapsto (3 + (x + y)^2, y + 1) \in \mathbb{R}^2$$

$[\![M]\!]$ is indeed a continuous function.

# Higher-Order Languages

## Higher-order Programming Languages:

functions are *first-class citizens*:

- they can be passed as argument;
- they can be returned as output.

## Motivations

- code reuse
- modularity
- conciseness

## Example

Higher-order languages:

- Haskell, ML, Java, Python, Scala . . .
- Model: $\lambda$-calculus (Church 1930s)

# Simply-typed $\lambda$-calculus with reals as base type

### The types

$$\tau ::= \mathtt{R} \mid \tau \times \tau \mid \tau \to \tau$$

### Example (An order 2 type)

$$(\mathtt{R} \to \mathtt{R}) \to (\mathtt{R} \times (\mathtt{R} \to \mathtt{R}))$$

### The programs

$$t \in \Lambda_{\mathbb{R}}^{\mathcal{F}} ::= x \mid \underline{r} \mid \underline{f}(t, \ldots, t) \qquad \text{with } f \in \mathcal{F}, r \in \mathbb{R}$$
$$\mid \lambda x.t \mid tt \mid (t, t) \mid t.1 \mid t.2 \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$$

### Remark

The type system ensures termination–even strong normalization–of all programs.

# A first-order program in $\Lambda_{\mathbb{R}}$





Example ($M : \mathtt{R} \to \mathtt{R}$ build using HO components)

We suppose $f_1, f_2$ two primitives functions.

$$M[f_1, f_2] := \lambda y. \left( \begin{array}{l} \lambda x.(x(y+1) + x(y-1)) \\ (\lambda z.\mathtt{if}\ z > 0\ \mathtt{then}\ \underline{f_1}(z)\ \mathtt{else}\ \underline{f_2}(z)) \end{array} \right)$$

$$[\![M]\!][f_1, f_2] : \mathbb{R} \to \mathbb{R}$$

$$y \mapsto \begin{cases} f_1(y+1) + f_1(y-1) \text{ when } y - 1 > 0 \\ f_1(y+1) + f_2(y-1) \text{ when } y - 1 \leq 0 < y + 1 \\ f_2(y+1) + f_2(y-1) \text{otherwise} \end{cases}$$

# The Question

How to extend containment theorems to this **higher-order** framework ?

# A Proof Scheme for Higher-Order Programs: Logical Relations

Used in the literature to study:

- lambda-definability;
- program termination (Gödel's system T (Tait 1967), System F (Girard 1972) ...)

# A toy example: termination for $\Lambda_{\mathbb{R}}$

## Defining Predicates on **closed** terms:

$$Red_{\mathtt{R}} := \{t \mid \vdash t : \mathtt{R} \wedge t \text{ terminates } \}$$
$$Red_{\tau\to\sigma} := \{t \mid \vdash t : \tau \to \sigma \wedge \forall s \in Red_\tau,\ ts \in Red_\sigma\} \ldots$$

## Extending predicates to **open** terms via substitutions

For $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$:

$$Red_\Gamma = \{\gamma : \{\text{variables}\} \to \{\text{programs}\} \mid \forall i,\ \gamma(x_i) \in Red_{\tau_i}\}$$
$$Red_\tau^\Gamma = \{t \mid \Gamma \vdash t : \tau \text{ s.t.} \forall \gamma \in Red_\Gamma,\ t\gamma \in Red_\tau\}$$

To end the proof: $\Gamma \vdash t : \tau \Leftrightarrow t \in Red_\tau^\Gamma$. (By induction of the structure of the **open** term $t$: Base cases $t = x, t = \underline{r} \ldots$)

# Proving Containment theorems by way of Logical Relations?

## Problem

- Logical relations are designed for 0-order properties: termination, equivalence between programs...
- We are interested in first-order properties, i.e. predicates on functions: continuity, polynomials, differentiability...

# Our Solution: Open Logical Relations

## Defining predicated on **open** terms–with real variables only context

. $\Theta : x_1 : \mathtt{R}, \dots, x_n : \mathtt{R}$.

$$t \in \mathcal{F}^{\Theta}_{\mathtt{R}} \iff (\Theta \vdash t : \mathtt{R} \wedge [\![\Theta \vdash t : \mathtt{R}]\!] \in \mathfrak{F})$$
$$t \in \mathcal{F}^{\Theta}_{\tau_1 \to \tau_2} \iff (\Theta \vdash t : \tau_1 \to \tau_2 \wedge \forall s \in \mathcal{F}^{\Theta}_{\tau_1}.\ ts \in \mathcal{F}^{\Theta}_{\tau_2})$$

## Extending predicates to **open** terms via substitutions–for any context

For $\Gamma = x_1 : \tau_1, \dots, x_n : \tau_n$:

$$\mathcal{F}^{\Theta}_{\Gamma} = \{\gamma : \{\text{variables}\} \to \{\text{programs}\} \mid \forall i,\ \gamma(x_i) \in \mathcal{F}^{\Theta}_{\tau_i}\}$$
$$\mathcal{F}^{\Theta,\Gamma}_{\tau} = \{t \mid \Theta, \Gamma \vdash t : \tau \text{ s.t.} \forall \gamma \in \mathcal{F}^{\Theta}_{\Gamma},\ t\gamma \in \mathcal{F}^{\Theta}_{\tau}\}$$

To end the proof: $\Gamma, \Theta \vdash t : \tau \Leftrightarrow t \in \mathcal{F}^{\Theta,\Gamma}_{\tau}$.

## Theorem (Containment Theorem)

*$\mathfrak{F}$: a collection of real-valued functions including projections and closed under function composition. Then, any $\Lambda_{\mathfrak{F}}^{\times,\to,\mathtt{R}}$ term $x_1 : \mathtt{R}, \ldots, x_n : \mathtt{R}_n \vdash t : \mathtt{R}$ denotes a function (from $\mathbb{R}^n$ to $\mathbb{R}$) in $\mathfrak{F}$.*

## Example

- $\mathfrak{F} = \{\text{continuous functions}\}$
- $\mathfrak{F} = \{\text{polynomial functions}\}$

## Remark

It can also be deduced from a categorical theorem due to Lafont (1988).

# Application of Open Logical Relations (1)

Correctness for Automatic Differentiation Algorithms

# Automatic Differentiation Algorithms

## Goal

Compute the derivative of a computer program representing a real-valued function.
By propagating the chain rule accross the syntax tree of the program.

## Increasing interest in the community of programming languages

- Used for gradient descent $\Rightarrow$ applications in machine-learning, physical models...
- Automatic differentiation systems: Tensor Flow, Stan...
- Until recently, not much theoretical foundations, formal proofs techniques
  (this year: Pagani et al's POPL 2020, Staton et al's FOSSACS 2020) ...

# Forward AD in practice

## Our reference (Forward Mode)

Jones et al's: "Efficient differentiable programming in a functional array-processing language"
(only the functionnal core of their algorithm (no if, no iteration, no array...))

## The language

Simply typed $\Lambda_{\mathbb{R}}^{\mathfrak{F}}$ with $\mathfrak{F} \subseteq \{\text{differentiable functions}\}$

## A program transformation

$\mathtt{D} : \{\text{Programs}\} \to \{\text{Programs}\}$

- built by induction on the program structure.
- $\mathtt{D}t$ embedds the information of **both** the original program $t$ and its derivatives.

# The transformation $\mathtt{D}$ (1)

## Intuition

Type of dual numbers

$$\lambda x.t : \mathtt{R} \to \mathtt{R} \quad \Rightarrow \quad \lambda \mathtt{d}x.\mathtt{D}t : \overline{\mathtt{R} \times \mathtt{R}} \to \mathtt{R} \times \mathtt{R}$$

meaning: the original program $t$

meaning: the differential of $t$

## General Typing invariant

$$\lambda x.t : \tau_1 \to \tau_2 \quad \Rightarrow \quad \lambda \mathtt{d}x_1.\mathtt{D}t : \mathtt{D}\tau_1 \to \mathtt{D}\tau_2$$

## $\mathtt{D}$ on Types

$$\mathtt{DR} = \mathtt{R} \times \mathtt{R}$$

$$\mathtt{D}(\tau_1 \times \tau_2) = \mathtt{D}\tau_1 \times \mathtt{D}\tau_2 \qquad \mathtt{D}(\tau_1 \to \tau_2) = \mathtt{D}\tau_1 \to \mathtt{D}\tau_2$$

# The transformation $\mathtt{D}$ (2)

## $\mathtt{D}$ on Terms

$$\mathtt{D}\underline{r} = (\underline{r}, \underline{0}) \qquad \mathtt{D}x = \mathtt{d}x \qquad \mathtt{D}\lambda x.t = \lambda \mathtt{d}x.\mathtt{D}t$$

$$\mathtt{D}(\underline{f}(t_1, \ldots, t_n)) = (\underline{f}(\mathtt{D}t_1.1, \ldots, \mathtt{D}t_n.1), \sum_{i=1}^{n} \underline{\partial_{x_i} f}(\mathtt{D}t_1.1, \ldots, \mathtt{D}t_n.1) * \mathtt{D}t_i.2)$$

...

application of the chain rule

## Example $(t = (\lambda(x, y).\sin(x) + \cos(y)))$

$$\begin{aligned} \mathtt{D}t = \lambda(\mathtt{d}x, \mathtt{d}y).(&\sin(\mathtt{d}x.1) + \cos(\mathtt{d}y.1), \\ &\quad\cos(\mathtt{d}x.1) * \mathtt{d}x.2 - \sin(\mathtt{d}y.1) * \mathtt{d}y.2). \\ &: (\mathtt{R} \times \mathtt{R}) \times (\mathtt{R} \times \mathtt{R}) \to \mathtt{R} \times \mathtt{R} \end{aligned}$$

**Question:** How to recover the partial derivatives of $[\![t]\!] : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ ?

## Dual Expressions

$$\mathtt{dual}_x(y) = \begin{cases} (y, \underline{1}) & \text{if } x = y \\ (y, \underline{0}) & \text{otherwise.} \end{cases} \qquad : \mathtt{R} \times \mathtt{R}.$$

## Example

$$\lambda(x, y).(\mathtt{D}t(\mathtt{dual}_x(x))(\mathtt{dual}_x(y)).2) \qquad : \mathtt{R} \times \mathtt{R} \to \mathtt{R}$$

$$\equiv^{ctx} \cos(x) * 1 - \sin(y) * 0 \equiv^{ctx} \frac{\partial [\![t]\!]}{\partial x}$$

# Correctness

## Theorem

*For any term* $t : \mathtt{R}^n \to \mathtt{R}$ *the term* $\mathtt{D}t : \mathtt{DR}^n \to \mathtt{DR}$ *computes the partial derivatives of* $t$*, in the sense that for any* $k \in \{1, \ldots, n\}$ *we have*

$$\frac{\partial [\![t]\!]}{\partial x_k} = [\![\lambda(x_1, \ldots, x_n).(\mathtt{D}t(\mathtt{dual}_{x_k}(x_1)), \ldots, (\mathtt{dual}_{x_k}(x_n))).2]\!]$$

# Logical Relations for Automatic Differentiation (1)

A **binary** relation:

$$\mathcal{R}_{\mathrm{R}}^{\Theta} \subseteq \{programs\} \times \{programs\}$$

## Reminder: Base case for continuity

$\Theta : x_1 : \mathrm{R}, \ldots, x_n : \mathrm{R}$

$t \in \mathcal{F}_{\mathrm{R}}^{\Theta} \iff (\Theta \vdash t : \mathrm{R} \wedge [\![\Theta \vdash t : \mathrm{R}]\!] : \mathbb{R}^n \to \mathbb{R} \in \mathfrak{F})$

## Base Case for AD

$\Theta : x_1 : \mathrm{R}, \ldots, x_n : \mathrm{R}; \qquad \mathtt{D}\Theta : \mathtt{d}x_1 : \mathrm{R} \times \mathrm{R}, \ldots, \mathtt{d}x_n : \mathrm{R} \times \mathrm{R}.$

$$t \mathcal{R}_{\mathrm{R}}^{\Theta} s \iff \begin{cases} \Theta \vdash t : \mathrm{R} \wedge \mathtt{D}\Theta \vdash s : \mathrm{R} \times \mathrm{R} \\ \forall y : \mathrm{R}. [\![\Theta \vdash s[\mathtt{dual}_y(x_1)/\mathtt{d}x_1, \ldots, \mathtt{dual}_y(x_n)/\mathtt{d}x_n].1 : \mathrm{R}]\!] \\ \qquad = [\![\Theta \vdash t : \mathrm{R}]\!] \\ \forall y : \mathrm{R}. [\![\Theta \vdash s[\mathtt{dual}_y(x_1)/\mathtt{d}x_1, \ldots, \mathtt{dual}_y(x_n)/\mathtt{d}x_n].2 : \mathrm{R}]\!] \\ \qquad = \partial_y [\![\Theta \vdash t : \mathrm{R}]\!] \end{cases}$$

# Logical Relations for Automatic Differentiation (2)

Reminder: HO construction of $\mathcal{F}^\Theta$ for continuity

$$t \in \mathcal{F}^\Theta_{\tau_1 \to \tau_2} \iff (\Theta \vdash t : \tau_1 \to \tau_2 \wedge \forall s \in \mathcal{F}^\Theta_{\tau_1}.\ ts \in \mathcal{F}^\Theta_{\tau_2})$$

$\to$ construct for AD

$$t\,\mathcal{R}^\Theta_{\tau_1 \to \tau_2}\,s \iff \begin{cases} \Theta \vdash t : \tau_1 \to \tau_2 \wedge \mathtt{D}\Theta \vdash s : \mathtt{D}\tau_1 \to \mathtt{D}\tau_2 \\ \forall p, q.\ p\,\mathcal{R}^\Theta_{\tau_1}\,q \implies tp\,\mathcal{R}^\Theta_{\tau_2}\,sq \end{cases}$$

# Proof of the Correctness Theorem by way of Logical Relations

### Lemma (Fundamental Lemma)

*For all environments* $\Gamma, \Theta$ *and for any expression* $\Gamma, \Theta \vdash t : \tau$, *we have* $t \, \mathcal{R}_{\tau}^{\Gamma,\Theta} \, \mathtt{D}t$.

From there, we can deduce the correctness theorem.

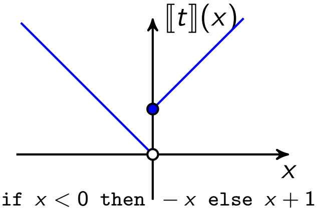
# Application of Open Logical Relations (2)

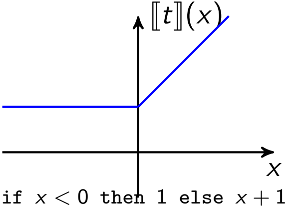
Local Continuity Properties in a language with an `if` construct

# Continuity and the if-construct

## Observation

The `if`-construct breaks global continuity




## Objective

Build a logical system to obtain continuity (local) guarantees on programs.

# Containing Local Continuity Properties: Chaudhuri et al's logical system

## Formal analysis of first-order programs

Judgments of the form:

$$b \vdash Cont(M, X)$$

- $b$: a boolean condition;
- $X$ a set of variables

designed to guarantee: $[\![M]\!] : \mathbb{R}^n \to \mathbb{R}$ is continuous along the variable in $X$ on all points that validates the condition $b$.

# Dealing with the if construct

We suppose three programs $M_1, M_2, M_3$ with $b_i \vdash Cont(M_i, X)$;

## Problem

Build a boolean condition $c$ such that:

$$c \vdash Cont(\texttt{if } M_1 \texttt{ then } M_2 \texttt{ else } M_3, X)$$

## Chaudhuri et al's Principle

- Ask $b_2 = b_3$: the domain of continuity of the branch is the same;
- In every discontinuity points $x$ of $M_1$, $M_2(x)$ and $M_3(x)$ must coincide: $b_2 \land \neg b_1 \Rightarrow M_2 \equiv_{obs} M_3$.

Then we can conclude: $b_2 \vdash Cont(\texttt{if } M_1 \texttt{ then } M_2 \texttt{ else } M_3, X)$.

# Our Contribution

## The Language

$$\Lambda_{\mathbb{R}}^{\mathfrak{F}} + \text{if-construct}$$

with $\mathfrak{F}$ any set of functions $\mathbb{R}^n \to \mathbb{R}$
(not necessarily continuous)

## Our system

- A **refinment type system** (add to types logical formulas $\phi$... to specify domains of $\mathbb{R}^n$);
  An instance of refined type:

$$\{\alpha_1 \in \mathtt{R}\}, \ldots \{\alpha_n \in \mathtt{R}\} \overset{\psi \rightsquigarrow \phi}{\to} \{\alpha \in \mathtt{R}\}$$

- in the spirit of Chaudhuri et al's for the FO fragment;
- designed to show: the program $t : \mathbb{R}^n \to \mathbb{R}$ is continuous on $\{x \in \mathbb{R}^n \mid x \models \phi\}$

# Local Continuity on an Example

**Example ($M : \mathtt{R} \to \mathtt{R}$ build using HO components)**

We suppose $f_1, f_2$ two primitives functions.

$$M[f_1, f_2] := \lambda y. \left( \begin{array}{l} \lambda x.(x(y+1) + x(y-1)) \\ (\lambda z.\mathtt{if}\ z > 0\ \mathtt{then}\ \underline{f_1}(z)\ \mathtt{else}\ \underline{f_2}(z)) \end{array} \right)$$

$$[\![M]\!] : y \in \mathbb{R} \mapsto \begin{cases} f_1(y+1) + f_1(y-1) \text{ when } y-1 > 0 \\ f_1(y+1) + f_2(y-1) \text{ when } y-1 \leq 0 < y+1 \\ f_2(y+1) + f_2(y-1) \text{otherwise} \end{cases}$$

In our system, we can show:

- $M[f_1, f_2]$ is continuous on $\{x \mid x \neq 1 \wedge x \neq -1\}$;
- $M[f_1, f_2]$ is continuous everywhere as soon as $f_1(1) = f_2(1)$ and $f_1(-1) = f_2(-1)$.

# Soundness of our Refined Type System

## Theorem

*Let $t$ be any program such that:*

$$x_1 : \{\alpha_1 \in \mathtt{R}\}, \ldots, x_n : \{\alpha_n \in \mathtt{R}\} \overset{\theta \rightsquigarrow \theta'}{\vdash_{\mathtt{r}}} t : \{\beta \in \mathtt{R}\}.$$

*Then it holds that:*

- $[\![t]\!](Dom(\theta))^{\alpha_1,\ldots,\alpha_n} \subseteq Dom(\theta')^{\beta}$;
- $[\![t]\!]$ *is sequentially continuous on* $Dom(\theta))^{\alpha_1,\ldots,\alpha_n}$ .

## Proof

By way of open logical relations.

# Conclusion

## Contributions

- flexibility of Open Logical Relations to show containement of first-order predicate or properties to an higher-order language;
- A proof-of-concept for proving correctness of AD algorithms in a functionnal setting
- A logical system to guarantee local continuity for higher-order programs

# Conclusion

## Future works

- Extension of our correctness proof for AD to **backward** differentiation algorithm;
- Adapting our refinement type system to deal with the if construct in the context of AD (checking differentiability in critical points)
- Implement our refinement type system using standard SMT-based approach (as done for standard refinement types).

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \qquad \frac{}{\Gamma \vdash \underline{r} : \mathtt{R}} \qquad \frac{\Gamma \vdash t_1 : \mathtt{R} \quad \cdots \quad \Gamma \vdash t_n : \mathtt{R}}{\Gamma \vdash \underline{f}(t_1, \ldots, t_n) : \mathtt{R}}$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x.t : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash s : \tau_1 \to \tau_2 \quad \Gamma \vdash t : \tau_1}{\Gamma \vdash st : \tau_2} \qquad \frac{\Gamma \vdash t_1 : \tau \quad \Gamma \vdash t_2 : \sigma}{\Gamma \vdash (t_1, t_2) : \tau \times \sigma}$$

$$\frac{\Gamma \vdash t : \tau_1 \times \tau_2}{\Gamma \vdash t.i : \tau_i} \ (i \in \{1, 2\})$$

# Our Rule for the if-then-else

$$\dfrac{\begin{array}{l} \Gamma \overset{\theta_t \rightsquigarrow (\beta=0 \vee \beta=1)}{\vdash_{\mathtt{r}}} t : \{\beta \in \mathtt{R}\} \\ \Gamma \overset{\theta_{(t,0)} \rightsquigarrow (\beta=0)}{\vdash_{\mathtt{r}}} t : \{\beta \in \mathtt{R}\} \\ \Gamma \overset{\theta_{(t,1)} \rightsquigarrow (\beta=1)}{\vdash_{\mathtt{r}}} t : \{\beta \in \mathtt{R}\} \end{array} \qquad \begin{array}{l} \Gamma \overset{\theta_s}{\vdash_{\mathtt{r}}} s : T \\ \Gamma \overset{\theta_p}{\vdash_{\mathtt{r}}} p : T \end{array} \quad c \qquad 1+2}{\Gamma \overset{\theta}{\vdash_{\mathtt{r}}} \mathtt{if}\ t\ \mathtt{then}\ s\ \mathtt{else}\ p : T}$$

The side-conditions are given as:

1. $\models \theta \Rightarrow$ $\big((\theta^s \vee \theta^p) \wedge (\theta^{(t,1)} \vee \theta^p) \wedge (\theta^{(t,0)} \vee \theta^s) \wedge (\theta_t \vee (\theta_s \wedge \theta_p))\big)$.
2. $\forall$ logical assignment $\sigma$ compatible with $G\Gamma, \sigma \models \theta \wedge \neg\theta_t$ implies $H\Gamma \vdash s\sigma^{G\Gamma} \equiv^{ctx} p\sigma^{G\Gamma}$.