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 $\mathbb{R}^n \rightarrow \mathbb{R}^m$;
- programming constructs: variables assignments, if, while...

Program interpretation:
real-valued functions $[M] : \mathbb{R}^n \rightarrow \mathbb{R}^m$

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: $\forall M$ a program, $\mathcal{P}([M])$ holds.
A Simple Example: (Global) Continuity.

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

**Fact**

\textbf{Cont} is contained for a restricted language:

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

**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.

**Proof.**

The predicate \textbf{Cont} is compositionnal. \[\square\]
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

$$\tau ::= \mathbb{R} \mid \tau \times \tau \mid \tau \rightarrow \tau$$

Example (An order 2 type)

$$(\mathbb{R} \rightarrow \mathbb{R}) \rightarrow (\mathbb{R} \times (\mathbb{R} \rightarrow \mathbb{R}))$$

The programs

$$t \in \Lambda_{\mathbb{R}}^F ::= x \mid r \mid f(t, \ldots, t) \quad \text{with } f \in F, r \in \mathbb{R}$$

$$\mid \lambda x.t \mid tt \mid (t, t) \mid t.1 \mid t.2 \mid \text{if } t \text{ then } t \text{ else } t$$

Remark

The type system ensures termination—even strong normalization—of all programs.
A first-order program in $\Lambda_R$

Example ($M : \mathbb{R} \rightarrow \mathbb{R}$ build using HO components)

We suppose $f_1, f_2$ two primitives functions.

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

$$\llbracket M \rrbracket[f_1, f_2] : \mathbb{R} \rightarrow \mathbb{R}$$

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

Defining Predicates on closed terms:

$$\text{Red}_R := \{ t \mid \vdash t : R \land t \text{ terminates} \}$$

$$\text{Red}_{\tau \to \sigma} := \{ t \mid \vdash t : \tau \to \sigma \land \forall s \in \text{Red}_\tau, ts \in \text{Red}_\sigma \} \ldots$$

Extending predicates to open terms via substitutions
For $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$:

$$\text{Red}_\Gamma = \{ \gamma : \{\text{variables}\} \to \{\text{programs}\} \mid \forall i, \gamma(x_i) \in \text{Red}_{\tau_i} \}$$

$$\text{Red}^\Gamma_\tau = \{ t \mid \Gamma \vdash t : \tau \text{ s.t.} \forall \gamma \in \text{Red}_\Gamma, t\gamma \in \text{Red}_\tau \}$$

To end the proof: $\Gamma \vdash t : \tau \iff t \in \text{Red}^\Gamma_\tau$. (By induction of the structure of the open term $t$: Base cases $t = x, t = r \ldots$)
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

\[ \Theta : x_1 : \mathbb{R}, \ldots, x_n : \mathbb{R}. \]

\[ t \in \mathcal{F}_R^\Theta \iff (\Theta \vdash t : \mathbb{R} \land \llbracket \Theta \vdash t : \mathbb{R} \rrbracket \in \mathfrak{T}) \]

\[ t \in \mathcal{F}_{\tau_1 \rightarrow \tau_2}^\Theta \iff (\Theta \vdash t : \tau_1 \rightarrow \tau_2 \land \forall s \in \mathcal{F}_{\tau_1}^\Theta. ts \in \mathcal{F}_{\tau_2}^\Theta) \]

Extending predicates to open terms via substitutions—for any context

For \( \Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n \):

\[ \mathcal{F}_\Gamma^\Theta = \{ \gamma : \{ \text{variables} \} \rightarrow \{ \text{programs} \} \mid \forall i, \gamma(x_i) \in \mathcal{F}_{\tau_i}^\Theta \} \]

\[ \mathcal{F}_{\tau}^\Theta,\Gamma = \{ t \mid \Theta, \Gamma \vdash t : \tau \text{ s.t.} \forall \gamma \in \mathcal{F}_\Gamma^\Theta, t\gamma \in \mathcal{F}_{\tau}^\Theta \} \]

To end the proof: \( \Gamma, \Theta \vdash t : \tau \iff t \in \mathcal{F}_{\tau}^\Theta,\Gamma \).
Theorem (Containment Theorem)

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

Example

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

Remark

It can also be deduced from a categorical theorem due to Lafont (1988).
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 across the syntax tree of the program.

Increasing interest in the community of programming languages

▶ Used for gradient descent ⇒ 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 $\Lambda_{\mathbb{R}}$ with $\mathfrak{F} \subseteq \{\text{differentiable functions}\}$

A program transformation
$D : \{\text{Programs}\} \rightarrow \{\text{Programs}\}$

- built by induction on the program structure.
- $Dt$ embeds the information of both the original program $t$ and its derivatives.
The transformation $D (1)$

Intuition

$\lambda x. t : R \rightarrow R \quad \Rightarrow \quad \lambda dx. Dt : R \times R \rightarrow R \times R$

- Type of dual numbers
- meaning: the original program $t$
- meaning: the differential of $t$

General Typing invariant

$\lambda x. t : \tau_1 \rightarrow \tau_2 \quad \Rightarrow \quad \lambda dx_1. Dt : D\tau_1 \rightarrow D\tau_2$

$D$ on Types

- $DR = R \times R$
- $D(\tau_1 \times \tau_2) = D\tau_1 \times D\tau_2$
- $D(\tau_1 \rightarrow \tau_2) = D\tau_1 \rightarrow D\tau_2$
The transformation $D (2)$

**D on Terms**

\[
D_r = (r, 0) \quad D x = dx \quad D \lambda x . t = \lambda dx . Dt \\
D(f(t_1, \ldots, t_n)) = (f(Dt_1.1, \ldots, Dt_n.1), \sum_{i=1}^{n} \partial_{x_i} f(Dt_1.1, \ldots, Dt_n.1) \cdot Dt_i.2)
\]

\ldots

\begin{align*}
\text{application} \quad \text{of the chain} \\
\text{rule}
\end{align*}
Example \((t = (\lambda(x, y). \sin(x) + \cos(y))))\)

\[
Dt = \lambda(dx, dy). (\sin(dx.1) + \cos(dy.1), \cos(dx.1) \cdot dx.2 - \sin(dy.1) \cdot dy.2).
\]

\[: (\mathbb{R} \times \mathbb{R}) \times (\mathbb{R} \times \mathbb{R}) \to \mathbb{R} \times \mathbb{R}\]

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

**Dual Expressions**

\[
dual_x(y) = \begin{cases} 
(y, 1) & \text{if } x = y \\
(y, 0) & \text{otherwise.}
\end{cases} \quad : \mathbb{R} \times \mathbb{R}.
\]

**Example**

\[
\lambda(x, y). (Dt(dual_x(x))(dual_x(y)).2) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}
\]

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

Theorem

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

$$\frac{\partial [t]}{\partial x_k} = \left[ \lambda(x_1, \ldots, x_n). (D_t(dual_{x_k}(x_1)), \ldots, (dual_{x_k}(x_n))). \right]_2$$
Logical Relations for Automatic Differentiation

(1)

A binary relation:

\[ R^\Theta_R \subseteq \{ \text{programs} \} \times \{ \text{programs} \} \]

Reminder: Base case for continuity

\( \Theta : x_1 : R, \ldots, x_n : R \)
\( t \in F^\Theta_R \iff (\Theta \vdash t : R \land [\Theta \vdash t : R] : \mathbb{R}^n \to \mathbb{R} \in \mathcal{F}) \)

Base Case for AD

\( \Theta : x_1 : R, \ldots, x_n : R; \quad D\Theta : dx_1 : R \times R, \ldots, dx_n : R \times R. \)

\[ t R^\Theta_R s \iff \begin{cases} \Theta \vdash t : R \land D\Theta \vdash s : R \times R \\ \forall y : R. [\Theta \vdash s[dual_y(x_1)/dx_1, \ldots, dual_y(x_n)/dx_n].1 : R] \\ = [\Theta \vdash t : R] \\ \forall y : R. [\Theta \vdash s[dual_y(x_1)/dx_1, \ldots, dual_y(x_n)/dx_n].2 : R] \\ = \partial_y [\Theta \vdash t : R] \end{cases} \]
Logical Relations for Automatic Differentiation

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

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

→ construct for AD

\[
t R^\Theta_{\tau_1 \to \tau_2} s \iff \begin{cases} 
\Theta \vdash t : \tau_1 \to \tau_2 \land D\Theta \vdash s : D\tau_1 \to D\tau_2 \\
\forall p, q. \, p R^\Theta_{\tau_1} q \implies tp R^\Theta_{\tau_2} sq
\end{cases}
\]
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 \( t \mathcal{R}_{\tau, \Theta}^\Gamma, \Theta \mathcal{D} t \).

From there, we can deduce the correctness theorem.
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: \( [M] : \mathbb{R}^n \rightarrow \mathbb{R} \) 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 $b_i \vdash \text{Cont}(M_i, X)$;

Problem
Build a boolean condition $c$ such that:

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

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: $b_2 \land \neg b_1 \Rightarrow M_2 \equiv_{\text{obs}} M_3$.

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

The Language

\[ \Lambda_{f} + \text{if-construct} \]

with \( f \) any set of functions \( \mathbb{R}^n \rightarrow \mathbb{R} \)
(not necessarily continuous)

Our system

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

\[
\{ \alpha_1 \in \mathbb{R} \}, \ldots \{ \alpha_n \in \mathbb{R} \} \xrightarrow{\psi \sim \phi} \{ \alpha \in \mathbb{R} \}
\]

- in the spirit of Chaudhuri et al's for the FO fragment;

- designed to show: the program \( t : \mathbb{R}^n \rightarrow \mathbb{R} \) is continuous on \( \{ x \in \mathbb{R}^n \mid x \models \phi \} \)
Local Continuity on an Example

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

We suppose $f_1, f_2$ two primitives functions.

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

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

In our system, we can show:

- $M[f_1, f_2]$ is continuous on $\{x \mid x \neq 1 \land x \neq -1\}$;
- $M[f_1, f_2]$ 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:

$$x_1 : \{ \alpha_1 \in \mathbb{R} \}, \ldots, x_n : \{ \alpha_n \in \mathbb{R} \} \vdash_r t : \{ \beta \in \mathbb{R} \}.$$

Then it holds that:

- $\llbracket t \rrbracket (\text{Dom}(\theta))^{\alpha_1, \ldots, \alpha_n} \subseteq \text{Dom}(\theta')^\beta$;
- $\llbracket t \rrbracket$ is sequentially continuous on $\text{Dom}(\theta))^{\alpha_1, \ldots, \alpha_n}$.

Proof

By way of open logical relations.
Conclusion

Contributions

▶ flexibility of Open Logical Relations to show containment 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).
On the Versatility of Logical Relations

Raphaëlle Crubillé

Containment Theorems by way of open logical relations

Correctness for Automatic Differentiation Algorithms

Soundness of a refinement type system for local continuity

Conclusion

\[
\begin{align*}
\Gamma, x : \tau & \vdash x : \tau & \Gamma \vdash r : R & \quad \Gamma \vdash f(t_1, \ldots, t_n) : R \\
\Gamma, x : \tau_1 & \vdash t : \tau_2 & \Gamma \vdash \lambda x. t : \tau_1 \rightarrow \tau_2 \\
\Gamma \vdash s : \tau_1 \rightarrow \tau_2 & \quad \Gamma \vdash t : \tau_1 & \Gamma \vdash t_1 : \tau & \Gamma \vdash t_2 : \sigma & \Gamma \vdash (t_1, t_2) : \tau \times \sigma \\
\Gamma \vdash t : \tau_1 \times \tau_2 & \quad \Gamma \vdash t. i : \tau_i & \quad (i \in \{1, 2\})
\end{align*}
\]
Our Rule for the if-then-else

\[ \theta_t \leadsto (\beta = 0 \lor \beta = 1) \]
\[ \Gamma \vdash_r t : \{ \beta \in R \} \quad \theta_s \]
\[ \theta_{(t,0)} \leadsto (\beta = 0) \quad \Gamma \vdash_r s : T \quad c \quad 1 + 2 \]
\[ \theta_p \]
\[ \Gamma \vdash_r p : T \]
\[ \Gamma \vdash_r (t,1) \leadsto (\beta = 1) \]
\[ \Gamma \vdash_r t : \{ \beta \in R \} \]

\[ \Gamma \vdash_r \text{if } t \text{ then } s \text{ else } p : T \]

The side-conditions are given as:

1. \[ \models \theta \Rightarrow \]
   \[
   ((\theta^s \lor \theta^p) \land (\theta^{(t,1)} \lor \theta^p) \land (\theta^{(t,0)} \lor \theta^s) \land (\theta_t \lor (\theta_s \land \theta_p))).
   \]

2. \[ \forall \text{ logical assignment } \sigma \text{ compatible with } G \Gamma, \sigma \models \theta \land \neg \theta_t \text{ implies } H \Gamma \vdash s\sigma^{G\Gamma} \equiv^{ctx} p\sigma^{G\Gamma}. \]