Differential Logical Relations

Ugo Dal Lago Francesco Gavazzo Akira Yoshimizu

University of Bologna and INRIA Sophia Antipolis

November 14, 2019

DLRs: What Are We Talking About?

A new approach to program distance for HO sequential languages

A foundation for approximate program transformations

First step towards a differential program semantics

Let us proceed step by step...

From Program Equivalence to Program Distance

Program Equivalence

What is program distance?

A natural refinement of program equivalence: do programs have the same behavior?

$$\left(\begin{array}{rl} \texttt{let} & \texttt{x}_1 = e_1 \\ & \texttt{x}_2 = e_2 \\ \texttt{in} & e \end{array} \right) \equiv \left(\begin{array}{rl} \texttt{let} & \texttt{x}_2 = e_2 \\ & \texttt{x}_1 = e_1 \\ \texttt{in} & e \end{array} \right)$$

Useful for many applications:

- Verification
- Security
- Optimization

Program Equivalence

Key points:

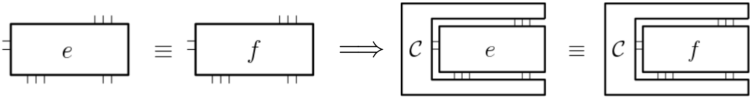
(1) Same observable behaviour:

$$e \equiv f \implies Obs(e) = Obs(f)$$

(2) Same interaction with the environment:

$$\forall C.\ e \equiv f \implies C[e] \equiv C[f]$$

Compositionality



From Equivalences to Distances

Program equivalence works fine for exact program analysis

Too strong for quantitative and approximate program analysis

- Probabilistic programming
- Differential privacy
- Real-valued computing
- **Approximate computing**

The Challenge of Approximate Computing

[Mittal 2016] Introducing soft errors in exact computations can provide disproportionate gains in efficiency and resource-consumption

Many applications:

- Computer vision
- Sensor-based computation
- Data analytics and machine learning

The Challenge of Approximate Computing

[Misailovic et al. 2010] A language-based approach to approximate computing via approximate program transformation

Program transformation

$$e \mapsto T(e)$$

Correct: e and $T(e)$ have same output ($e \equiv T(e)$)

(ε-)**Approximately correct**: e and $T(e)$ can have (ε-)different outputs

The Challenge of Approximate Computing: Examples

Loop perforation [Misailovic et al. 2010;2011]: skip iterations in expensive loops over large dataset

c skip factor:

$$\begin{array}{c} \texttt{for (i = 0; i < b; i++)}\ \{e\} \\ \Big\downarrow T \\ \texttt{for (i = 0; i < b; i += c)}\ \{e\} \end{array}$$

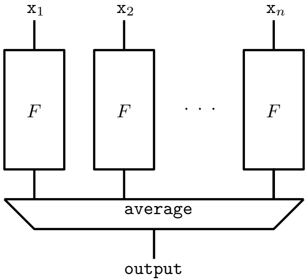
Example

$$\left(\begin{array}{l} \texttt{for (i = 0; i < N; i++)} \\ \qquad \texttt{sum += A[i]} \end{array} \right) \xrightarrow{T} \left(\begin{array}{l} \texttt{for (i = 0; i < N; i+=k)} \\ \qquad \texttt{sum += A[i]} \end{array} \right)$$

The Challenge of Approximate Computing: Examples

Numerical integrations [Zeyuan et al. 2012]: e integrates a function $F : \mathbb{R} \to \mathbb{R}$ over $[a, b]$

$$\begin{array}{ll} \texttt{let} & \Delta = \frac{b-a}{n} \\ & \texttt{x}_i = a + i \cdot \frac{\Delta}{2} \\ \texttt{in} & \\ & \frac{1}{n}\sum_{i=1}^{n}(b-a) \cdot F(\texttt{x}_i) \end{array}$$

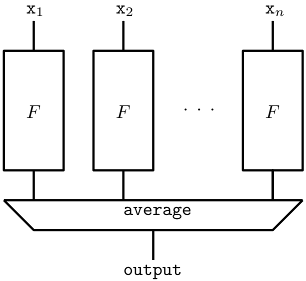


E.g. $F(x) = x \cdot \sin(\log(x))$

The Challenge of Approximate Computing: Examples

Substitution: Substitute F with F' computing a less accurate output in less time

Perforation: Instead of computing $F(\mathtt{x}_i)$ for all n-inputs, take only $s \leq n$ inputs

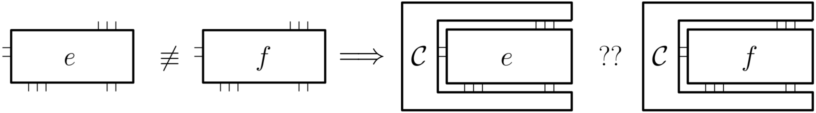


The Challenge of Approximate Computing

Approximate program transformations raise several questions

How to reason about APTs? When can we safely apply them?

What about compositionality?



The classic theory of program equivalence is useless. . .

Program Distance

Program Distance

Obviously, $e \not\equiv T(e) \ldots$ but differences bounded by accuracy ε

Natural guess: refine equivalences into distances

[Lawvere 1973]: equivalences and pseudometrics are essentially the same

$$\begin{array}{c} e \equiv f \\ \Updownarrow \\ \delta(e, f) = 0 \\ \big\downarrow \\ \delta(e, f) = \varepsilon \end{array}$$

Program Distance

What about compositionality?

Program Equivalence: compositionality = congruence

$$\forall C.\ e \equiv f \implies C[e] \equiv C[f]$$

Program distance: compositionality = non-expansiveness

$$\delta(e, f) \geq \sup_C \delta(C[e], C[f])$$

Question: is this the end of the story? Not for higher-order languages...

A Toy Language and its Denotational Semantics

We fix a simply-typed λ-calculus with primitives for real numbers

$$\tau ::= \texttt{real} \mid \tau \to \tau \mid \tau \times \tau$$

Higher-order language

$$\frac{\Gamma, \mathtt{x} : \tau \vdash e : \sigma}{\Gamma \vdash \lambda \mathtt{x}.e : \tau \to \sigma} \qquad \frac{\Gamma \vdash f : \tau \to \sigma \quad \Gamma \vdash e : \tau}{\Gamma \vdash f\, e : \sigma}$$

Real-valued computations

$$\frac{}{\Gamma \vdash \underline{r} : \texttt{real}} \qquad \frac{\Gamma \vdash e_i : \texttt{real} \quad F : \mathbb{R}^n \to \mathbb{R}}{\Gamma \vdash \underline{F}(e_1, \ldots, e_n) : \texttt{real}}$$

Standard denotational semantics

$$[\![\texttt{Real}]\!] = \mathbb{R} \quad [\![\tau \to \sigma]\!] = [\![\tau]\!] \to [\![\sigma]\!] \quad [\![\tau \times \sigma]\!] = [\![\tau]\!] \times [\![\sigma]\!]$$

Distance Amplification

Theorem [Crubillé & Dal Lago 2017; Gavazzo 2019]

Suppose:

(1) $\delta : \Lambda \times \Lambda \to [0, \infty]$ pseudometric

(2) δ is non-expansive

(3) δ is adequate: $\forall e, f : \texttt{real}.\ \delta(e, f) \geq |[\![e]\!] - [\![f]\!]|$

Then, $\delta(e, f) > 0 \implies \delta(e, f) = \infty$

Distance Amplification: Solutions

Several solutions . . .

- Denotationally-based distances [de Amorim et al. 2017]
- Tuple and contextual distance [Crubillé & Dal Lago 2017]
- Applicative distances [Gavazzo 2018]
- Metric logical relations [Reed & Pierce 2010]

Metric Logical Relations

Design a type system for program sensitivity (a.k.a. program robustness [Chaudhuri et al. 2011])

A program e has sensitivity $c \in [0, \infty]$ if

$$\delta(v, v') \leq \varepsilon \implies \delta(e(v), e(v')) \leq c \cdot \varepsilon$$

Sensitivity tracked using a bounded linear type system

$\sigma \multimap \tau$	non-expansive functions
$!_c \sigma \multimap \tau$	c-Lipschitz continuous functions

Metric Logical Relations

Main Result

A metric logical relation inducing a pseudometric δ^L s.t.

- **Adequacy**

$$\delta^L_{\mathtt{real}}(e, f) \geq |[\![e]\!] - [\![f]\!]|$$

- **Respects function space**

$$\delta^L_{\tau \multimap \sigma}(\lambda \mathtt{x}.e, \lambda \mathtt{x}.f) \geq \sup_v \delta^L_\sigma(e[v/\mathtt{x}], f[v/\mathtt{x}])$$

- **Metric preservation** For any $\vdash e : !_c \sigma \multimap \tau$

$$\delta^L_\sigma(v, v') \leq \varepsilon \implies \delta^L_\tau(e(v), e(v')) \leq c \cdot \varepsilon$$

Metric Logical Relations

Metric logical relations and sensitivity analysis applicable to

- Differential privacy [Reed & Pierce 2010; de Amorim et al. 2017; 2019]
- Probabilistic computations [Gavazzo 2018]
- Approximate computing

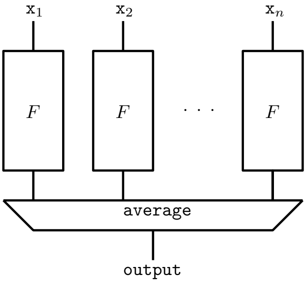
  Loop perforation [Chaudhuri et al. 2011]

Question: Is this really the end of the story? ... Not really ...

The Challenge of Approximate Computing, Reloaded

Recall our numerical integration example

$$\begin{array}{ll} \texttt{let} & \Delta = \frac{b-a}{n} \\ & \mathtt{x}_i = a + i \cdot \frac{\Delta}{2} \\ \texttt{in} & \\ & \frac{1}{n}\sum_{i=1}^{n}(b-a)\cdot F(\mathtt{x}_i) \end{array}$$



and the substitution technique: we substitute F with F' computing a less accurate output in less time

The Challenge of Approximate Computing, Reloaded

In approximate computing substitution is usually context-aware

Context aware-substitution: Substitute F with F' computing a less accurate output in less time **on inputs in a given set** A

If input $\in [0.5, 1]$

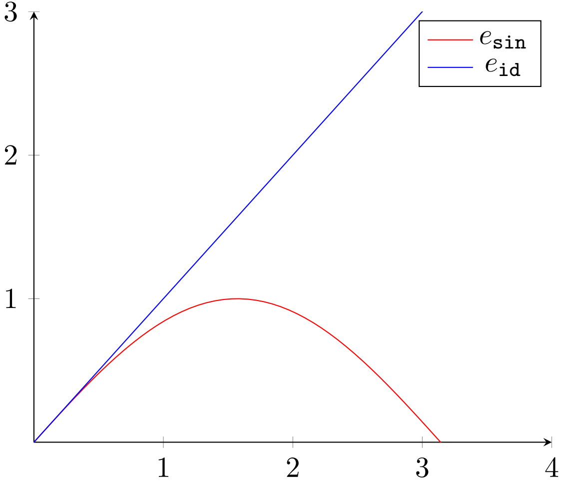
$$\lambda\texttt{x}.\texttt{1/x} \mapsto \lambda\texttt{x}.2.823 - \texttt{1.882*x}$$

If input ≈ 0

$$\lambda\texttt{x}.\texttt{sin x} \mapsto \lambda\texttt{x}.\texttt{x}$$

The Challenge of Approximate Computing, Reloaded

$e_{\texttt{id}} = \lambda \texttt{x}.\texttt{x}$ $\quad$ $e_{\texttt{sin}} = \lambda \texttt{x}.\texttt{sin x}$



The Challenge of Approximate Computing, Reloaded

Can we use δ^L to prove approximate correctness of $T(e_{\mathtt{sin}}) = e_{\mathtt{id}}$? No!

$$\delta^L(e_{\mathtt{id}}, e_{\mathtt{sin}}) = \infty$$

Even worst ...

Proposition

For any pseudometric $\delta : \Lambda \times \Lambda \to [0, \infty]$ which is adequate and respect function spaces

$$\delta_{\tau \multimap \sigma}(\lambda \mathtt{x}.e, \lambda \mathtt{x}.f) \geq \sup_v \delta_\sigma(e[v/\mathtt{x}], f[v/\mathtt{x}])$$

we have

$$\delta(e_{\mathtt{id}}, e_{\mathtt{sin}}) = \infty$$

Differential Logical Relations

Towards Differential Logical Relations

A striking intuition

A Semantics for Approximate Program Transformations

Edwin Westbrook and Swarat Chaudhuri
Department of Computer Science, Rice University
Houston, TX 77005
Email: {emw4,swarat}@rice.edu

Thinking to program differences as just numbers is too restrictive.

Towards Differential Logical Relations

Replace $[0, \infty]$ in

$$\delta : \Lambda \times \Lambda \to [0, \infty]$$

with a more complex space of differences

$$\delta : \Lambda \times \Lambda \to \mathcal{D}$$

$\mathcal{D}$ reflects the interactive complexity of programs, hence type-dependent

Differential Logical Relations

Question: What is a difference between two programs?

- Ordinary logical relations: boolean values.
- Metric logical relations: (real) numbers.
- Differential logical relations: difference spaces $(\!|\tau|\!)$

Differential Logical Relations

What is a difference between two (real) numbers?

$$(\!|\texttt{real}|\!) = [0, \infty]$$

What is a difference between two functions $\vdash e, f : \sigma \to \tau$?

$$(\!|\sigma \to \tau|\!) = [\![\sigma]\!] \times (\!|\sigma|\!) \to (\!|\tau|\!)$$

Intuition

$$(\!|\texttt{Input} \to \texttt{Output}|\!) \approx \texttt{Input} \times \texttt{Error}(\texttt{Input}) \to \texttt{Error}(\texttt{Output}).$$

Differential Logical Relations

Putting things together. . .

$$
\begin{aligned}
(\!|\mathtt{real}|\!) &= [0, \infty] \\
(\!|\sigma \to \tau|\!) &= [\![\sigma]\!] \times (\!|\sigma|\!) \to (\!|\tau|\!) \\
(\!|\sigma \times \tau|\!) &= (\!|\sigma|\!) \times (\!|\tau|\!).
\end{aligned}
$$

Differential Logical Relations

A differential logical relations is a ternary relation $\mathcal{D}_\tau \subseteq \Lambda_\tau \times (\!|\tau|\!) \times \Lambda_\tau$

Informal reading

$$\mathcal{D}_\tau(e, \phi, f) \iff \text{"}\phi \text{ is a difference between } \vdash e, f : \tau\text{"}$$

What are the defining clauses of $\mathcal{D}$?

$$\begin{aligned}
\mathcal{D}_{\mathtt{real}}(e, r, f) &\iff |[\![e]\!] - [\![f]\!]| \leq r \\
\mathcal{D}_{\tau_1 \times \tau_2}(e, (\phi_1, \phi_2), f) &\iff \forall i \in \{1, 2\}.\ \mathcal{D}_{\tau_i}(e.i, \phi_i, f.i) \\
\mathcal{D}_{\sigma \to \tau}(e, \phi, f) &\iff ???
\end{aligned}$$

Differential Logical Relations

How to define $\mathcal{D}_{\sigma\to\tau}(e, \phi, f)$?

Intuition

$$(\!|\texttt{Input} \to \texttt{Output}|\!) \approx \texttt{Input} \times \texttt{Diff}(\texttt{Input}) \to \texttt{Diff}(\texttt{Output}).$$

Hence $\phi : [\![\sigma]\!] \times (\!|\sigma|\!) \to (\!|\tau|\!)$

$$\mathcal{D}_{\sigma\to\tau}(e, \phi, f) \iff \underbrace{\forall v, w \in \mathcal{V}_\sigma}_{\forall\ \text{inputs}}.$$

$$\underbrace{\forall \zeta \in (\!|\sigma|\!).\ \mathcal{D}_\sigma(v, \zeta, w)}_{\forall\ \text{input difference}\ \zeta} \implies \underbrace{\begin{cases} \mathcal{D}_\tau(ev, \phi([\![v]\!], \zeta), fw) \\ \mathcal{D}_\tau(ew, \phi([\![v]\!], \zeta), fv) \end{cases}}_{\phi([\![v]\!], \zeta)\ \text{output difference}}$$

Differential Logical Relations

How can we use DLRs to reason about context-aware approximate program transformation?

Compositional reasoning for ordinary and metric logical relations follows from fundamental lemma.

What is FL for differential logical relations?

Logical Relations	Fundamental Lemma
Ordinary	$R(e, e) = \text{true}$
Metric	$\delta(e, e) = 0$
Differential	???

Differential Logical Relations

Fundamental Lemma

For any $\vdash e : \tau$, there exists $\phi \in (\!|\tau|\!)$ such that $\mathcal{D}_\tau(e, \phi, e)$.

Why non-null differences?

- ϕ describes the sensitivity of e.
- E.g. $\vdash \lambda\mathtt{x}.\mathtt{x} : \mathtt{real} \to \mathtt{real}$ has self-difference

$$\lambda\!\!\lambda(r, \varepsilon).\varepsilon$$

- Thus only terms at ground types have null-difference.

Differential Logical Relations

Fundamental Lemma $\implies$ compositional reasoning

- Given a context: $\mathcal{C} : \sigma \to \tau$
- FL $\implies \exists \zeta \in (\!|\sigma \to \tau|\!).\ (C, \zeta, C) \in \mathcal{D}_{\sigma \to \tau}$
- Given terms $\vdash e, f : \sigma$ s.t. $(e, \phi, f) \in \mathcal{D}_\sigma$
- Conclude:

$$(C[e], \zeta([\![e]\!], \phi), C[f]) \in \mathcal{D}_\tau$$

Moral: ζ captures context-awareness

The impact of replacing f with e in C is $\zeta([\![e]\!], \phi)$

The Challenge of Approximate Computing, Revolution

Look back at $e_{\mathtt{id}}$ vs $e_{\mathtt{sin}}$

Lemma

$$(e_{\mathtt{id}}, \lambda\!\!\lambda(r,\varepsilon).\varepsilon + |\sin r - r|, e_{\mathtt{sin}}) \in \mathcal{D}_{\mathtt{real}\to\mathtt{real}}$$

Consider the context $\vdash C = (\lambda \mathtt{x}..\mathtt{x}(\mathtt{x}\underline{c}))[-] : (\mathtt{Real} \to \mathtt{Real}) \to \mathtt{Real}$

Consider the self-difference ζ for C

$$\zeta \in [\![\mathtt{real} \to \mathtt{real}]\!] \times (\!|\mathtt{real} \to \mathtt{real}|\!) \to (\!|\mathtt{real}|\!)$$
$$\zeta = \lambda\!\!\lambda(\varphi, \psi).\psi(\varphi(c), \psi(c, 0)).$$

What is the impact of replacing $e_{\mathtt{sin}}$ with $e_{\mathtt{id}}$ in context C?

$$\zeta([\![e_{\mathtt{id}}]\!], \lambda\!\!\lambda(r,\varepsilon).\varepsilon + |\sin r - r|)$$

The Challenge of Approximate Computing, Revolution

Our analysis is compositional

We have taken the context C into account, but once and for all

The map ζ can be computed without any reference to $e_{\mathtt{id}}$ and $e_{\mathtt{sin}}$

Theoretical Results on Differential Logical Relations

DLRs and Ordinary LRs

Hereditary Null Differences

DLRs subsumes ordinary LRs

$$(\!|\texttt{real}|\!)^0 = \{0\} \qquad\qquad (\!|\sigma \times \tau|\!)^0 = (\!|\sigma|\!)^0 \times (\!|\tau|\!)^0$$

$$(\!|\sigma \to \tau|\!)^0 = \{\phi \in (\!|\sigma \to \tau|\!) \mid \forall x \in [\![\sigma]\!].\ \forall \zeta \in (\!|\sigma|\!)^0.\ \phi(x, \zeta) \in (\!|\tau|\!)^0\}$$

Proposition

Two programs $\vdash e, f : \tau$ are logically related iff $\exists \phi \in (\!|\tau|\!)^0.\ \mathcal{D}_\tau(e, \phi, f)$

DLRs and Metric LRs

DLRs subsumes metric LRs

Hereditary Real-valued Differences

We parametrize $(\!|\tau|\!)$ by a single real number r

$$(\!|\mathtt{real}|\!)^r = \{r\} \qquad\qquad (\!|\sigma \times \tau|\!)^r = (\!|\sigma|\!)^r \times (\!|\tau|\!)^r$$

$$(\!|\sigma \to \tau|\!)^r = \{\phi \in (\!|\sigma \to \tau|\!) \mid \forall x \in [\![\sigma]\!].\ \forall \zeta \in (\!|\sigma|\!)^s.\ \phi(x, \zeta) \in (\!|\tau|\!)^{r+s}\}$$

Proposition

For all programs $\vdash e, f : \tau$, $\delta^L(e, f) = r$ iff $\exists \phi \in (\!|\tau|\!)^r.\ \mathcal{D}_\tau(e, \phi, f)$

Hereditary Finite Differences

Since the calculus is strongly normalizing we expect differences to be hereditary finite

Hereditary Finite Differences

$$(\!|\texttt{real}|\!)^{<\infty} = \mathbb{R}_{\geq 0} \qquad (\!|\sigma \times \tau|\!)^{<\infty} = (\!|\sigma|\!)^{<\infty} \times (\!|\tau|\!)^{<\infty}$$

$$(\!|\sigma \to \tau|\!)^{<\infty} = \{\phi \in (\!|\sigma \to \tau|\!) \mid \forall x \in [\![\sigma]\!].\ \forall \zeta \in (\!|\sigma|\!)^{<\infty}.\ \phi(x, \zeta) \in (\!|\tau|\!)^{<\infty}\}$$

Fundamental Lemma, II

Assume all real-valued operators $\underline{F}$ to denote a *weakly bounded* function $F : \mathbb{R}^n \to \mathbb{R}$ (i.e. $F(B)$ bounded whenever $B \subseteq \mathbb{R}^n$ is). Then, for any program $\vdash e : \tau$ there exists $\phi \in (\!|\tau|\!)^{<\infty}$. $\mathcal{D}_\tau(e, \phi, e)$

Categorical Foundation

Metric LRs have a categorical foundation in the symmetric monoidal closed category of pseudometric spaces and Lipschitz-continuous maps

Also DLRs have a categorical foundation in the category of generalized metric domains

Definition

Given a quantale $(V, \leq, \oplus, 0)$, a generalised metric domain on V is a pair $(X, \mathcal{D}_X)$, where X is a set and $\mathcal{D}_X \subseteq X \times V \times X$ satisfies:

$$\begin{aligned}
\mathcal{D}_X(x, 0, y) &\implies x = y \\
\mathcal{D}_X(x, \varphi, y) &\implies \mathcal{D}(y, \varphi, x) \\
\mathcal{D}_X(x, \varphi, y) \wedge \mathcal{D}_X(y, \chi, y) \wedge \mathcal{D}(y, \xi, z) &\implies \mathcal{D}_X(x, \varphi \oplus \chi \oplus \xi, z)
\end{aligned}$$

Categorical Foundation

We have non-null self distance $\mathcal{D}_X(x, \varphi, x) \not\Longrightarrow \varphi = 0$

Theorem

GMDs form a cartesian closed category

Arrows are defined following the defining of DLRs

Conclusion

Conclusion

We have introduced DLRs and their basic properties.

We show the strengths of DLRs studying context-aware approximate program transformations

But DLRs are interesting also from a purely theoretical perspective

Differential Program Semantics

DLRs are the first step towards differential program semantics

ERC Consolidator Grant **DIAPASoN**

Focus on program difference rather than program identity

- Ugo Dal Lago
- Simone Martini
- Davide Sangiorgi
- Aurore Alcolei
- Guillaume Geoffroy
- Paolo Pistone
- Melissa Antonelli
- Gabriele Vannoni

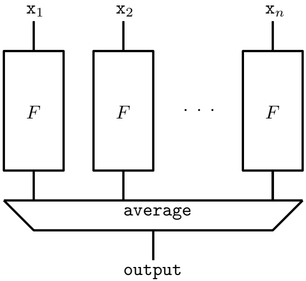
Future Work

Several future works

- Full recursion (step-indexing?)
- Better mathematical foundations (Partial metric spaces [Bukatin et al. 2009. Kopperman et al.2005])
- Connections with incremental computing?
- Differential bisimulation? Differential game semantics?
- Effects

Future Work: Probabilistic Approximate Program Transformation

Probabilistic loop perforation: compute $F(\mathtt{x}_i)$ only for $s \leq n$ **randomly sampled** inputs



Probabilistic substitution

$$\lambda \mathtt{x}.\mathtt{sin}\ \mathtt{x} \xrightarrow{T} (\lambda \mathtt{x}.\mathtt{sin}\ \mathtt{x}) \oplus (\lambda \mathtt{x}.\mathtt{x})$$

Questions

The Category of Generalized Metric Domains

Objects Triples $(X, \mathcal{D}_X, V)$

Arrows $(f, \Gamma) : (X, \mathcal{D}_X, V_X) \to (Y, \mathcal{D}_Y, V_Y)$

$$f : X \to Y \qquad \Gamma : X \times V_X \to V_Y$$

such that

$$\mathcal{D}_X(x, \varphi, x') \implies \begin{cases} \mathcal{D}_Y(f(x), \Gamma(x, \varphi), f(x')) \\ \mathcal{D}_Y(f(x), \Gamma(x', \varphi), f(x')) \end{cases}$$