On Taming Differentiable Logics

Reynald Affeldt, Alessandro Bruni, Matthew Daggitt, Ekaterina Komendantskaya,
Natalia Ślusarz, **Kathrin Stark**, Robert Stewart

This Work

Reynald Affeldt

Alessandro Bruni

Matthew Daggitt

Katya Komendantskaya

Natalia Ślusarz

Kathrin Stark

Rob Stewart

Towards Ensuring that Neural Networks Satisfy Verification Properties

We want a neural network N to satisfy certain properties P; for example:

$\epsilon - \delta$ robustness
Given a vector v, N is said to be $\epsilon - \delta$ robust w.r.t v if:

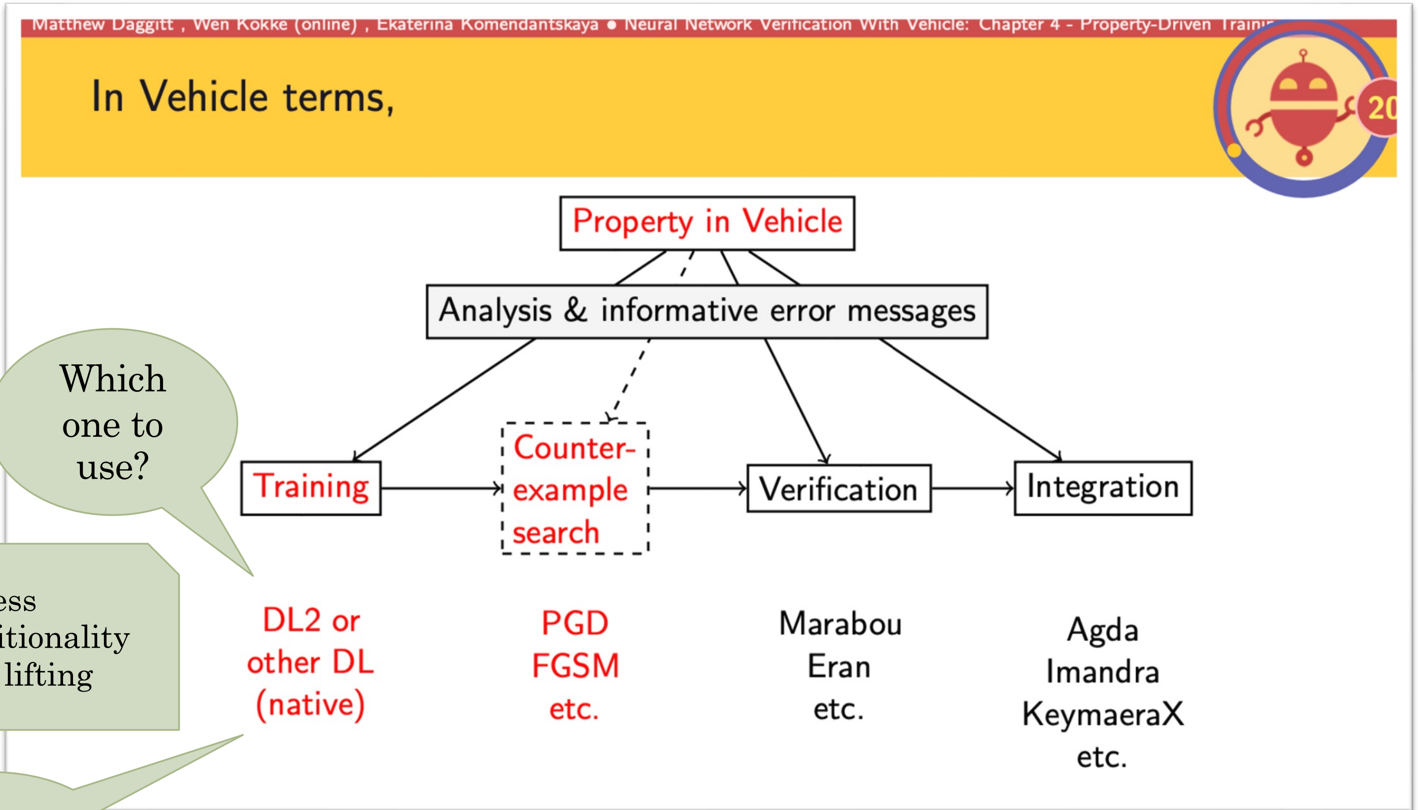
$$\forall x. |x - v|_{L_\infty} \leq \epsilon \Rightarrow |N(x) - N(v)|_{L_\infty} \leq \delta'$$

But: Even the most accurate neural networks fail even the most natural verification properties, such as robustness [Fischer et al., ‘19].

A possible solution: Translate the desired property P into a differentiable **loss function** that punishes not satisfying P (**Differentiable Logic/DL)** in the tradition of property-based training for neural networks [Giunchiglia et al., IJCAI ‘22]

For example:
$[\![e_1 \leq e_2]\!] = -\max([\![e_1]\!] - [\![e_2]\!], 0)$

Towards PL Tools for Neural Networks



Which one to use?

Wish List
- Soundness
- Compositionality
- Shadow lifting

It's a trade-off!

https://vehicle-lang.github.io/tutorial/#introduction

This Talk

A **common framework** for different differentiable logics that allows to give a uniform semantics: **LDL** (**L**ogic of **D**ifferentiable **L**ogic) [Ślusarz et al., LPAR ’23]

Goal: A generic framework in which logical/geometric properties of different DLs can be formalized/proven.

A common framework to **compare properties** of different DLs

A **mechanization*** in Coq/MathComp, allowing to easily test out new extensions/different approaches [Affeldt et al., ITP ’24]

* So far: Without quantifiers

This Talk

A **common framework** for different differentiable logics that allows to give a uniform semantics: **LDL** (**L**ogic of **D**ifferentiable **L**ogic) [Ślusarz et al., LPAR '23]

Goal: A generic framework in which logical/geometric properties of different DLs can be formalized/proven.

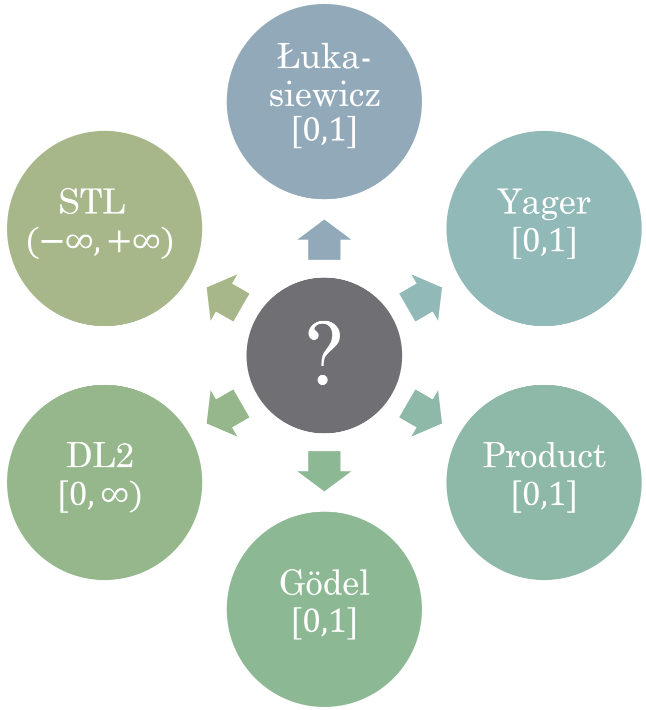
A common framework to **compare properties** of different DLs

A **mechanization*** in Coq/MathComp, allowing to easily test out new extensions/different approaches [Affeldt et al., ITP '24]

* So far: Without quantifiers

Possible Differentiable Logics

- DL2 [Fischer et al., ‘19]
- STL [Varnai, Dimarogonas ‘20]
- Fuzzy logics [Krieken et al. ‘21]
  - Gödel
  - Łukasiewicz
  - Yager
  - Product



But:
1. Only propositional fragment
2. Syntax/semantics/pragmatics partially not well-separated
3. Lack of unified general syntax and semantics

LDL: A Logic of Differentiable Logics

Dependently Typed Version from [Affeldt et al., ITP '24]

$$\text{type} \ni t ::= \text{Bool} \mid \text{Index } n \text{ for } n \in \mathbb{N} \mid \text{Real} \mid \text{Vec } n \mid \text{Fun } n\ m \text{ for } n, m \in \mathbb{N}$$

No implication

$$\begin{array}{llcl}
\text{exprInd} & \ni i & ::= & i \in \mathbb{N} \\
\text{exprR} & \ni r, r_1, r_2 & ::= & r \in \mathbb{R} \mid [v]_i \\
\text{exprFun} & \ni f & ::= & f \in \mathbb{R}^n \to \mathbb{R}^m \\
\text{exprVec} & \ni v & ::= & v \in R^n \mid f\ v
\end{array}$$

$$\begin{array}{llcl}
\text{exprB} & \ni p, p_0, \ldots, p_n & ::= & \mathit{True} \mid \mathit{False} \\
& & \mid & r_1 \leq r_2 \\
& & \mid & r_1 \neq r_2 \\
& & \mid & \bigwedge_M (p_0, \ldots, p_M) \\
& & \mid & \bigvee_M (p_0, \ldots, p_M) \\
& & \mid & \neg p \\
& & \mid & \exists\, (x : t).\, p \\
& & \mid & \forall (x : t).\, p
\end{array}$$

Required by (non-assoc.) STL

Partially not supported

$$\text{expr} \ni e \ ::= \ x \mid \lambda\, (x : t).\ e$$

These parts are not yet in the Coq formalisation

Example: For $\epsilon - \delta$ robustness, given concrete values for $\epsilon/\delta/\text{v}/N$ both the L_∞ norm and the right hand of the implication can be expressed via LDL.

One Logic – Several Interpretations

Generalization to n-ary

Strong negation

	$[\![\bigwedge_M s]\!]$	$[\![\bigvee_M s]\!]$	$[\![\neg e]\!]$
Gödel	$\min [\![s]\!]_{\mathrm{G}}$	$\max\ [\![s]\!]_{\mathrm{G}}$	$1 - [\![e]\!]_{\mathrm{G}}$
Łukasiewicz	$\max \left[\sum_{a \in [\![s]\!]_{\text{Ł}}} a - \lvert s \rvert + 1, 0\right]$	$\min \left[\sum_{a \in [\![s]\!]_{\text{Ł}}} a, 1\right]$	$1 - [\![e]\!]_{\text{Ł}}$
Yager	$\max \left[1 - \left(\sum_{a \in [\![s]\!]_{\mathrm{Y}}} (1-a)^p\right)^{1/p}, 0\right]$	$\min \left[\left(\sum_{a \in [\![s]\!]_{\mathrm{Y}}} a^p\right)^{1/p}, 1\right]$	$1 - [\![e]\!]_{\mathrm{Y}}$
product	$\prod_{a \in [\![s]\!]_{\mathrm{P}}} a$	fold $(\lambda x\ y\ .\ x + y - xy)\ 0\ [\![s]\!]_{\mathrm{P}}$	$1 - [\![e]\!]_{\mathrm{P}}$
DL2	$\sum_{a \in [\![s]\!]_{\mathrm{DL2}}} a$	$(-1)^{\lvert s \rvert + 1} \cdot \prod_{a \in [\![s]\!]_{\mathrm{DL2}}} a$	undefined[†]
STL	$and_S\ [\![s]\!]_{\mathrm{STL}}$		$-[\![e]\!]_{\mathrm{STL}}$
Bool	$\bigwedge_M [\![s]\!]_{\mathrm{B}}$		$\neg [\![e]\!]_{\mathrm{B}}$

Not part of original definition

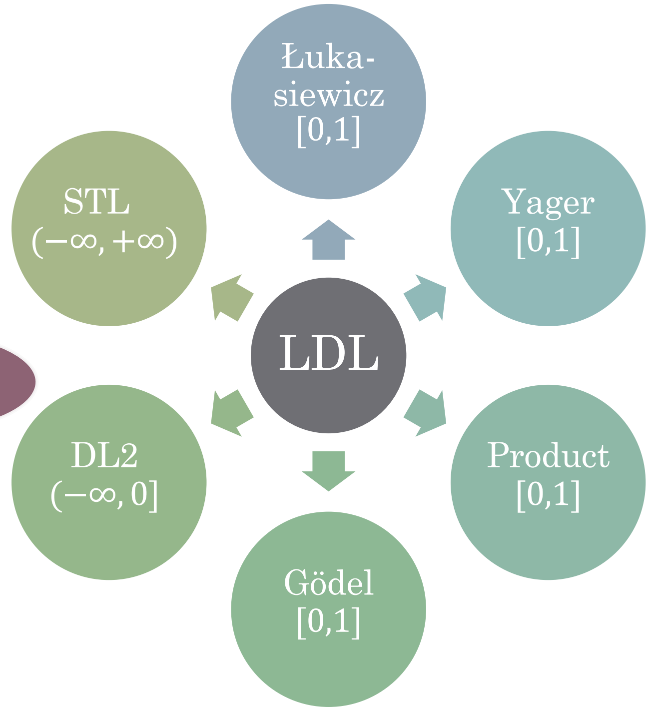
	$[\![e_1 = e_2]\!]$	$[\![e_1 \leq e_2]\!]$	$[\![True]\!]$	
f	$[\![e_1]\!] = -[\![e_2]\!]$ $[\![e_1]\!] = [\![e_2]\!]$ $\max \left[1 - \left\lvert \frac{[\![e_1]\!] - [\![e_2]\!]}{[\![e_1]\!] + [\![e_2]\!]} \right\rvert, 0\right]$	if $[\![e_1]\!] = -[\![e_2]\!]$ then $[\![e_1]\!] \leq [\![e_2]\!]$ else $\max \left[1 - \max \left[\frac{[\![e_1]\!] - [\![e_2]\!]}{[\![e_1]\!] + [\![e_2]\!]}, 0\right], 0\right]$	1	0
DL2	$-\lvert [\![e_2]\!]_{\mathrm{DL2}} - [\![e_1]\!]_{\mathrm{DL2}} \rvert$	$-\max\left[[\![e_1]\!]_{\mathrm{DL2}} - [\![e_2]\!]_{\mathrm{DL2}}, 0\right]$	0	$-\infty$
STL	$-\lvert [\![e_2]\!]_{\mathrm{STL}} - [\![e_1]\!]_{\mathrm{STL}} \rvert$	$[\![e_2]\!]_{\mathrm{STL}} - [\![e_1]\!]_{\mathrm{STL}}$	$+\infty$	$-\infty$
Bool	$[\![e_1]\!]_{\mathrm{B}} = [\![e_2]\!]_{\mathrm{B}}$	$[\![e_1]\!]_{\mathrm{B}} \leq [\![e_2]\!]_{\mathrm{B}}$	*True*	*False*

Originally: $[0, \infty)$

Not part of original definition

$$and_S\ [a_1, \ldots, a_M] = \begin{cases} \dfrac{\sum_i a_{\min} e^{\tilde{a}_i} e^{\nu \tilde{a}_i}}{\sum_i e^{\nu \tilde{a}_i}} & \text{if } a_{\min} < 0 \\ \dfrac{\sum_i a_i e^{-\nu \tilde{a}_i}}{\sum_i e^{-\nu \tilde{a}_i}} & \text{if } a_{\min} > 0 \\ 0 & \text{if } a_{\min} = 0 \end{cases} \quad \text{where} \quad \begin{aligned} \nu &\in \mathbb{R}^+ \text{ (constant)} \\ a_{\min} &= \min\ [a_1, \ldots, a_M] \\ \tilde{a}_i &= \frac{a_i - a_{\min}}{a_{\min}} \end{aligned}$$

or_S is analogous to and_S



This Talk

A **common framework** for different differentiable logics that allows to give a uniform semantics: **LDL** (**L**ogic of **D**ifferentiable **L**ogic) [Ślusarz et al., LPAR ’23]

Goal: A generic framework in which logical/geometric properties of different DLs can be formalized/proven.

A common framework to **compare properties** of different DLs

A **mechanization*** in Coq/MathComp, allowing to easily test out new extensions/different approaches [Affeldt et al., ITP ’24]

* So far: Without quantifiers

This Talk

A **common framework** for different differentiable logics that allows to give a uniform semantics: **LDL** (**L**ogic of **D**ifferentiable **L**ogic) [Ślusarz et al., LPAR '23]

Goal: A generic framework in which logical/geometric properties of different DLs can be formalized/proven.

A common framework to **compare properties** of different DLs

A **mechanization*** in Coq/MathComp, allowing to easily test out new extensions/different approaches [Affeldt et al., ITP '24]

* So far: Without quantifiers

Properties of Interest

- **Soundness**: If a property interprets as true/false in the DL; it is true in Boolean logic
- **Compositionality**: Composition of negation with conjunction/disjunction; idempotence, commutativity, and associativity of conjunction/disjunction
- **Shadow-lifting:** Gradual improvement in training

The bad news: None of the existing DLs satisfies all of these requirements [Varnai, Dimarogonas '20]
Conclusion: We might need to provide support for incorporating a range of DLs for different scenarios

Properties of Interest
Compositionality

▶ **Definition 5** (Commutativity, idempotence, and associativity of $\bigwedge_M$). *Given a DL, the interpretation function of conjunction is commutative if for any permutation π of the integers $i \in \{1, \ldots, M\}$ we have*

$$\left[\!\!\left[\bigwedge\nolimits_M (p_0, \ldots, p_M) \right]\!\!\right]_{DL} = \left[\!\!\left[\bigwedge\nolimits_M \left(p_{\pi(0)}, \ldots, p_{\pi(M)}\right) \right]\!\!\right]_{DL}.$$

It is idempotent and associative if we have

$$\begin{aligned} \left[\!\!\left[\bigwedge\nolimits_M (p, \ldots, p) \right]\!\!\right]_{DL} &= [\![p]\!]_{DL}, \\ \left[\!\!\left[\bigwedge\nolimits_M \left(\bigwedge\nolimits_M (p_0, p_1), p_2 \right) \right]\!\!\right]_{DL} &= \left[\!\!\left[\bigwedge\nolimits_M \left(p_0, \bigwedge\nolimits_M (p_1, p_2) \right) \right]\!\!\right]_{DL}. \end{aligned}$$

Properties of Interest
Soundness

Soundness. Given a DL, an expression e, and a Boolean value b, the DL is sound if:

$$[\![e]\!]_{DL} = [\![b]\!]_{DL} \implies [\![e]\!]_B = b$$

	Soundness
Gödel	Yes
Łukasiewicz	No
Yager	No
Product	Yes
DL2	Yes*
STL	?

In [Ślusarz et al., LPAR ’23]

*negation-free fragment

On Soundness

- Clear if it's on closed intervals – see [Ślusarz et al., LPAR '23]
- How to even state soundness for open intervals – i.e., for DL2/STL?

Attempt 1: Add $-\infty$ and $+\infty$ as constants to the domain; keep the previous soundness statement.

- Vacuous proof – no formula evaluates to $-\infty$ or ∞

Not a problem in fuzzy logics – e.g., $[\![3 = 3]\!]_P = 1$

Attempt 2: Keep the open interval intact; re-define soundness in terms of intervals: If the interpretation of the formula e is greater or equal to 0, then $[\![e]\!]_B = True$, else $[\![e]\!]_B = False$.

- But: Negation is no longer sound. If $[\![3 = 3]\!]_{STL} = 0$, then also $[\![\neg\,(3 = 3)]\!]_P = 0$

Exclude 0? => Complicates interpretations/ not differentiable

Attempt 3: Use intervals to define truth/ remove negation.

Properties of Interest
Shadow Lifting [Varnai, Dimarogonas '20]

▶ **Definition 6** (Shadow-lifting property [27]). *The DL satisfies the shadow-lifting property if, for any* $[\![p]\!]_{DL} \neq 0$*:*

$$\left.\frac{\partial \left[\!\!\left[\bigwedge_M (p_0, \ldots, p_i, \ldots, p_M) \right]\!\!\right]_{DL}}{\partial [\![p_i]\!]_{DL}}\right|_{p_j = p \text{ where } i \neq j} > 0$$

holds for all $0 \leq i \leq M$*, where* ∂ *denotes partial differentiation.*

An Overview

	DL2	Gödel	Łuka-siewicz	Yager	Product	STL
Weak Smoothness	Yes*	No	No	No	Yes*	Yes
Shadow-Lifting	Yes	No	No	No	Yes	Yes [Varnai et al. ‘20]
Scale Invariance	Yes	Yes	No	No	No	Yes [Varnai et al. ‘20]
Negation	No	Yes	Yes	Yes	Yes	Yes
Idempotence	No	Yes [Krieken et al. ‘21]	No [Cintula et al. ‘11]	No [Klement et al. ‘04]	No [Cintula et al. ‘11]	Yes [Varnai et al. ‘20]
Commutativity	Yes	Yes [Krieken et al. ‘21]	Yes [Krieken et al. ‘21]	Yes [Krieken et al. ‘21]	Yes [Krieken et al. ‘21]	Yes [Varnai et al. ‘20]
Associativity	Yes	Yes [Krieken et al. ‘21]	Yes [Krieken et al. ‘21]	Yes [Krieken et al. ‘21]	Yes [Krieken et al. ‘21]	No [Varnai et al. ‘20]
Soundness	Yes†	Yes	No	No	Yes	Yes†

* Smoothness in propositional case † Negation-free fragment

This Talk

A **common framework** for different differentiable logics that allows to give a uniform semantics: **LDL** (**L**ogic of **D**ifferentiable **L**ogic) [Ślusarz et al., LPAR '23]

Goal: A generic framework in which logical/geometric properties of different DLs can be formalized/proven.

A common framework to **compare properties** of different DLs

A **mechanization*** in Coq/ MathComp, allowing to easily test out new extensions/ different approaches [Affeldt et al., ITP '24]

* So far: Without quantifiers

This Talk

A **common framework** for different differentiable logics that allows to give a uniform semantics: **LDL** (**L**ogic of **D**ifferentiable **L**ogic) [Ślusarz et al., LPAR '23]

Goal: A generic framework in which logical/geometric properties of different DLs can be formalized/proven.

A common framework to **compare properties** of different DLs

A **mechanization*** in Coq/MathComp, allowing to easily test out new extensions/different approaches [Affeldt et al., ITP '24]

* So far: Without quantifiers

Syntax

```
Inductive flag := def | undef.

Inductive ldl_type :=
  Bool_T of flag | Index_T of nat | Real_T | Vector_T of nat | Fun_T of nat & nat.
```

```
Definition Bool_T_undef := Bool_T undef.
Definition Bool_T_def := Bool_T def.
Inductive comparison := cmp_le | cmp_eq.

Inductive expr : ldl_type -> Type :=
 | ldl_real   : R -> expr Real_T
 | ldl_bool   : forall p, bool -> expr (Bool_T p)
 | ldl_idx    : forall n, 'I_n -> expr (Index_T n)
 | ldl_vec    : forall n, n.-tuple R -> expr (Vector_T n)
 | ldl_and    : forall x, seq (expr (Bool_T x)) -> expr (Bool_T x)
 | ldl_or     : forall x, seq (expr (Bool_T x)) -> expr (Bool_T x)
 | ldl_not    : expr Bool_T def -> expr Bool_T def
 | ldl_cmp    : forall x, comparison -> expr Real_T -> expr Real_T -> expr (Bool_T x)
 | ldl_fun    : forall n m, (n.-tuple R -> m.-tuple R) -> expr (Fun_T n m)
 | ldl_app    : forall n m, expr (Fun_T n m) -> expr (Vector_T n) -> expr (Vector_T m)
 | ldl_lookup : forall n, expr (Vector_T n) -> expr (Index_T n) -> expr Real_T.
```

Translation

```
Definition type_translation (t : ldl_type) : Type :=
match t with
| Bool_T x => R
| Real_T => R
| Vector_T n => n.-tuple R
| Index_T n => 'I_n
| Fun_T n m => n.-tuple R -> m.-tuple R
end.
```

```
Fixpoint stl_translation {t} (e : expr t) : type_translation t :=
  match e in expr t return type_translation t with
  | ldl_and _ (e0 :: s) => let A     := map stl_translation s in
                           let a0    := stl_translation e0 in
                           let a_min := \big[minr/a0]_(i <- A) i in
                           if a_min < 0 then stl_and_lt0 (a0 :: A) else
                           if a_min > 0 then stl_and_gt0 (a0 :: A) else
                           0
  | `~ E1                => - {[ E1 ]}
  | E1 `<= E2            => {[ E2 ]} - {[ E1 ]}
  ... (* see [29] for omitted connectives *)
  end where "{[ e ]}" := (stl_translation e).
```

On the Coq Formalization
Overview

File	Contents	L.o.c.
Additions to MathComp libraries		
`mathcomp_extra.v`	Lemmas iterated min/max, etc.	504
`analysis_extra.v`	L'Hôpital's rule, Cauchy's MVT (§ 5.3), etc.	820
Generic logic and generic definitions of properties		
`ldl.v`	LDL syntax and semantics(§ 3), shadow-lifting (Sect. 5.1)	417
Soundness, logical and geometric properties of concrete logics		
`dl2.v`	DL2: logical (§ 4), geometric (§ 5.2)	259
`fuzzy.v`	Gödel, Łukasiewicz, Yager, product: logical (§ 4), geometric (§ 5.2)	731
`stl.v`	STL; logical (§ 4), geometric (§ 5.4)	977
Alternative formalisations of logical properties/ soundness using extended reals		
`dl2_ereal.v`	DL2: logical (§ 4.2)	211
`stl_ereal.v`	STL: logical (§ 4.2)	362
	Total	4281

Easily fixable gaps

Translate to corresponding properties of underlying definitions

Helped in closing previous gaps

Helped in finding the right definitions

https://github.com/ndslusarz/formal_LDL

This Talk
Future Work

A **common framework** for different differentiable logics that allows to give a uniform semantics: **LDL** (**L**ogic of **D**ifferentiable **L**ogic) [Ślusarz et al., LPAR '23]

Connection of logics with the logics of Lawvere quantale [Bacci et al., '23]

Goal: A generic framework in which logical/geometric properties of different DLs can be formalized/proven.

A common framework to **compare properties** of different DLs

Revised negation

A **mechanization*** in Coq/ MathComp, allowing to easily test out new extensions/ different approaches [Affeldt et al., ITP '24]

* So far: Without quantifiers

In progress

The Question of Quantifiers

- **Motivation**: More expressive/required to internally interpret different forms of robustness

- On **finite domains** such as Bool/ Index n (e.g., [Krieken et al. '21]):
  - Use finitely composed conjunction and disjunction:
    Given $[\![\tau]\!] = [d_1, \dots, d_n\}$, have $[\![\forall x : \tau . e]\!] = [\![e[x \,/\, d_1] \dots e[x \,/\, d_n]]\!]$
  - Analogously for $\exists x : \tau . e$ and disjunction.

- **Infinite domains**
  - [Fischer et al. '19]: Interpret universally quantified formulae via expectation maximization methods
  - In [Ślusarz et al., LPAR '23]: Suggestion of a language-independent quantifier

$$[\![\forall x : \tau . e]\!]_L^{N,Q,\Gamma} = \mathbb{E}_{\min} \left[\left(\lambda y . [\![e]\!]^{N,Q,\Gamma[x \to y]}\right)(Q[x]) \right]$$

    where $\mathbb{E}_{\min} [g(\mathbf{X})] = \lim_{\gamma \to 0} \int_{x \in \mathbb{B}^{\gamma}_{x_{min}}} p_X(x) g(x) dx$
  - … and existential quantifier as maximum.

Practical algorithm for computing it?

- … to be continued

Any Questions?

Published Papers

Logic of differentiable logics: Towards a uniform semantics of DL
https://arxiv.org/pdf/2303.10650
Natalia Ślusarz, Ekaterina Komendantskaya, Matthew L Daggitt, Robert Stewart, KS (LPAR '23)

Taming Differentiable Logics with Coq Formalisation Reynald Affeldt, Alessandro Bruni, Ekaterina Komendantskaya, Natalia Ślusarz, KS (ITP '24)

Coq Development
https://github.com/ndslusarz/formal_LDL

September 9, 15:00, ITP

Parts of the project are looking for a PhD student:

Vacancy: PhD in Computer Science

Title: Formal Verification of AI Interfaces

Advisors: Ekaterina Komendantskaya (Southampton University, UK), Alessandro Bruni (IT University of Copenhagen, Denmark), Reynald Affeldt (AIST, Japan)

Start Date: As soon as the right candidate is found

Location: Southampton University, UK; with collaborative visits involving researchers at AIST, Japan and IT University of Copenhagen, Denmark.