

# Analyzing Differentiable Fuzzy Logic Operators

Emile van Krieken<sup>a,\*</sup>, Erman Acar<sup>a</sup>, Frank van Harmelen<sup>a</sup>

<sup>a</sup> VU University Amsterdam

---

## Abstract

In recent years there has been a push to integrate symbolic AI and deep learning, as it is argued that the strengths and weaknesses of these approaches are complementary. One such trend in the literature are weakly supervised learning techniques that use operators from fuzzy logics. They employ prior background knowledge described in logic to benefit the training of a neural network from unlabeled and noisy data. By interpreting logical symbols using neural networks, this background knowledge can be added to regular loss functions used in deep learning to integrate reasoning and learning. In this paper, we analyze how a large collection of logical operators from the fuzzy logic literature behave in a differentiable setting. We find large differences between the formal properties of these operators that are of crucial importance in a differentiable learning setting. We show that many of these operators, including some of the best known, are highly unsuitable for use in a differentiable learning setting. A further finding concerns the treatment of implication in these fuzzy logics, with a strong imbalance between gradients driven by the antecedent and the consequent of the implication. Finally, we empirically show that it is possible to use Differentiable Fuzzy Logics for semi-supervised learning. However, to achieve the most significant performance improvement over a supervised baseline, we have to resort to non-standard combinations of logical operators which perform well in learning, but which no longer satisfy the usual logical laws. We end with a discussion on extensions to large-scale problems.

---

## 1. Introduction

In recent years much work has been published on the integration of symbolic and statistical approaches to Artificial Intelligence (AI) (Garcez et al., 2012; Besold et al., 2017). This development is partly inspired by critiques on the statistical method deep learning (Marcus, 2018; Pearl, 2018), which has been the focus of the AI community in the last decade. While deep learning has brought many important breakthroughs in computer vision (Brock et al., 2018), natural language processing (Devlin et al., 2018) and reinforcement learning (Silver et al., 2017), the concern is that progress will be halted if its shortcomings are not dealt with. Among these is the massive amounts of data that deep learning needs to effectively learn a concept. On the other hand, symbolic AI can reuse concepts and knowledge using only a small amount of data (e.g. a single logical statement). Additionally, it is easier to interpret the decisions of symbolic AI as the explicit symbols refer to concepts that have a clear meaning to humans, while deep learning uses complex mathematical models using millions or billions of numerical parameters. Finally, it is much easier to describe background knowledge using symbolic AI and to integrate it into such a system.

A major downside of symbolic AI is that it is unable to capture the noisiness and ambiguity of sensory data. It is difficult to precisely express how small changes in the input data should produce different outputs. This is related to the *symbol grounding problem* which Harnad (1990) defines as how “the semantic interpretation of a formal symbol system can be made intrinsic to the system, rather than just parasitic on the meanings in our heads”. Symbols refer to concepts that have an intrinsic meaning to us humans, but computers manipulating these symbols can not trivially understand this meaning. In contrast to symbolic AI, a properly trained deep learning model excels at modeling complex sensory data. These models could bridge the gap between symbolic systems and the real world. Therefore, several recent approaches (Diligenti et al., 2017b; Garnelo et al., 2016; Serafini and Garcez, 2016; Manhaeve et al., 2018; Evans and Grefenstette, 2018) aim to interpret symbols that are used in logic-based systems using deep learning models. These are some of the first systems to implement a proposition going back 20 years from Harnad (1990), namely “a hybrid nonsymbolic/symbolic system (...) in which the elementary symbols are grounded in (...) non-symbolic representations that pick out, from their proximal sensory projections, the distal object categories to which the elementary symbols refer.”

---

\*Corresponding author

Email addresses: e.van.krieken@vu.nl (Emile van Krieken), erman.acar@vu.nl (Erman Acar), Frank.van.Harmelen@vu.nl (Frank van Harmelen)

### 1.1. Reasoning and Learning using Gradient Descent

We introduce *Differentiable Fuzzy Logics* (DFL). DFL integrates reasoning and learning by using logical formulas which express background knowledge. The symbols in these formulas are interpreted using a deep learning model of which the parameters are to be learned. DFL constructs differentiable loss functions based on these formulas that can be minimized using gradient descent. This ensures that the deep learning model acts in a manner that is consistent with the background knowledge as we can backpropagate towards the deep learning model parameters.

In order to ensure loss functions are differentiable, DFL uses fuzzy logic semantics (Klir and Yuan, 1995). Predicate, function and constant symbols are interpreted using the deep learning model. By maximizing the degree of truth of the background knowledge using gradient descent, both learning and reasoning are performed in parallel.

By adding the loss function of DFL to other loss functions commonly used in deep learning, DFL can be used for more challenging machine learning tasks than purely supervised learning. These methods fall under the umbrella of weakly supervised learning (Zhou, 2017). For example, it becomes possible to detect noisy or inaccurate supervision by correcting inconsistencies between the labels, the model’s predictions and the background knowledge (Donadello et al., 2017). A promising application is semi-supervised learning in which only a limited fraction of the dataset is labeled, and a large part is unlabeled (Xu et al., 2018; Hu et al., 2016). This is done by correcting the predictions of the deep learning model when it is logically inconsistent.

In this paper, we present an analysis of the choice of operators used to compute the logical connectives in DFL. For example, functions called *t-norms* are used to connect two fuzzy propositions (Klir and Yuan, 1995). Because they return the degree of truth of the event that both propositions are true, such t-norms generalizes the Boolean conjunction. Similarly, a fuzzy implication generalizes the Boolean implication. Most of these operators are differentiable, allowing them to be used in DFL. Interestingly, the derivatives of these operators determine how DFL corrects the deep learning model when its predictions are inconsistent with the background knowledge. We will show that the qualitative properties of these derivatives are integral to both the theory and practice of DFL.

### 1.2. Contributions

The main question that we aim to answer in this work is: “*which fuzzy logic operators for aggregation, conjunction, disjunction and implication have convenient theoretical properties when using them in gradient descent?*”. We analyze both theoretically and empirically the effect of the choice of operators used to compute the logical connectives in Differentiable Fuzzy Logics on the learning behaviour of a DFL system. To this end,

- we introduce several known operators from fuzzy logic (Section 3) and the framework of Differentiable Fuzzy Logics (Section 4) that uses these operators;
- we analyze the theoretical properties of four types of operators (Section 5): Aggregation functions, which are used to compute the universal quantifier  $\forall$ , conjunction and disjunction operators, which are used to compute the connectives  $\wedge$  and  $\vee$ , and fuzzy implications which are used to compute the connective  $\rightarrow$ ;
- we perform experiments to compare these fuzzy logic operators in a semi-supervised experiment (Section 9).
- We conclude with several recommendations for choices of operators.

## 2. Differentiable Logics

Differentiable Logics (DL) are logics for which differentiable loss functions can be constructed that represent logical formulas. These logics use background knowledge to deduce the truth value of statements in unlabeled or poorly labeled data. This allows us to use such data during learning, possibly together with normal labeled data. This can be beneficial as unlabeled, poorly labeled and partially labeled data is cheaper and easier to come by. Importantly, this approach differs from Inductive Logic Programming (Muggleton and de Raedt, 1994) which derives rules from data. DL is the other way around: the logic informs what the truth values of the statements could have been.

We motivate the use of Differentiable Logics with the following scenario: Assume we have an agent  $A$  whose goal is to describe the scene on an image. It gets feedback from a supervisor  $S$ , who does not have an exact description of these images available. However,  $S$  does have a background knowledge base  $\mathcal{K}$ , encoded in some logical formalism, about the concepts contained on the images. The intuition behind Differentiable Logics is that  $S$  can correct  $A$ ’s descriptions of scenes when they are not consistent with its knowledge base  $\mathcal{K}$ .

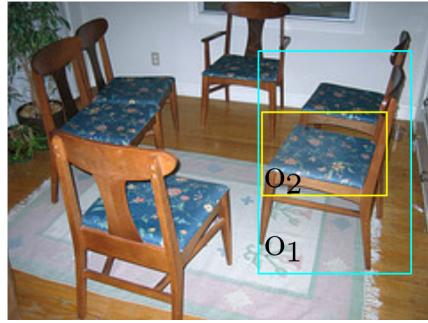


Figure 1: In this running example, we have an image with two objects on it,  $o_1$  and  $o_2$ .

**Example 1.** We illustrate this idea with the following example. Agent  $A$  has to describe the image  $I$  in Figure 1 containing two objects,  $o_1$  and  $o_2$ .  $A$  and the supervisor  $S$  only know of the unary class predicates  $\{\text{chair}, \text{cushion}, \text{armRest}\}$  and the binary predicate  $\{\text{partOf}\}$ . Since  $S$  does not have a description of  $I$ , it will have to correct  $A$  based on the knowledge base  $\mathcal{K}$ .  $A$  describes the image as follows, where the probability indicates the confidence in an observation:

$$\begin{array}{ll}
p(\text{chair}(o_1)|I, o_1) = 0.9 & p(\text{chair}(o_2)|I, o_2) = 0.4 \\
p(\text{cushion}(o_1)|I, o_1) = 0.05 & p(\text{cushion}(o_2)|I, o_2) = 0.5 \\
p(\text{armRest}(o_1)|I, o_1) = 0.05 & p(\text{armRest}(o_2)|I, o_2) = 0.1 \\
p(\text{partOf}(o_1, o_1)|I, o_1) = 0.001 & p(\text{partOf}(o_2, o_2)|I, o_2) = 0.001 \\
p(\text{partOf}(o_1, o_2)|I, o_1, o_2) = 0.01 & p(\text{partOf}(o_2, o_1)|I, o_2, o_1) = 0.95
\end{array}$$

Suppose that  $\mathcal{K}$  contains the following logic formula which says objects that are a part of a chair are either cushions or armrests:

$$\forall x, y \text{ chair}(x) \wedge \text{partOf}(y, x) \rightarrow \text{cushion}(y) \vee \text{armRest}(y).$$

$S$  might now reason that since  $A$  is relatively confident of  $\text{chair}(o_1)$  and  $\text{partOf}(o_2, o_1)$  that the antecedent of this formula is satisfied, and thus  $\text{cushion}(o_2)$  or  $\text{armRest}(o_2)$  has to hold. Since  $p(\text{cushion}(o_2)|I, o_2) > p(\text{armRest}(o_2)|I, o_2)$ , a possible correction would be to tell  $A$  to increase its degree of belief in  $\text{cushion}(o_2)$ .

We would like to automate the kind of supervision  $S$  performs in the previous example. To this end, we identify a significant family of Differentiable Logics in the literature that are based on fuzzy logic: We call it **Differentiable Fuzzy Logics** (DFL). Examples of logics in this family are Real Logic (Serafini and Garcez, 2016), the rather similarly named Deep Fuzzy Logic (Marra et al., 2019b), and the logics underlying Semantic Based Regularization (Diligenti et al., 2017b), LYRICS (Marra et al., 2018) and KALE (Guo et al., 2016). We compare these logics in Section 10.1. The objective of DFL is to maximize the satisfaction of the full grounding of a fuzzy knowledge base. To this end, truth values of ground atoms are not discrete but continuous, and logical connectives are interpreted using some function over these truth values.

### 3. Background

#### 3.1. Logic

We assume that the basic syntax and semantics of first-order logic is familiar. We will denote predicates using the sans serif font, for example  $\text{cushion}$ , variables by  $x, y, z, x_1, \dots$  and objects by  $o_1, o_2, \dots$ . For convenience, we will be limiting ourselves to function-free formulas in prenex normal form. Functions in **prenex normal form** start with quantifiers followed by a quantifier-free subformula. An example of a formula in prenex form is  $\forall x, y P(x, y) \wedge Q(x) \rightarrow R(y)$ . An **atom** is  $P(t_1, \dots, t_m)$  where  $t_1, \dots, t_m$  are terms. If  $t_1, \dots, t_m$  are all constants, we say it is a **ground atom**.

Fuzzy logic is a real-valued logic where truth values are real numbers in  $[0, 1]$  where 0 denotes completely false and 1 denotes completely true. Fuzzy logic models the concept of vagueness by arguing that the truth value of many propositions can be noisy to measure, or subjective. We will be looking at predicate fuzzy logics in particular. Predicate fuzzy logics extend propositional fuzzy logics with universal and existential quantification.

#### 3.2. Operators for Conjunction, Disjunction and Aggregation

In this section, we will introduce the semantics of the fuzzy operators  $\wedge$  (conjunction),  $\vee$  (disjunction) and  $\neg$  (negation) that are used to connect truth values of fuzzy predicates, and the semantics of the  $\forall$  quantifier. We follow (Jayaram and Baczyński, 2008) in this section and refer to it for proofs and additional results.

Name	T-norm	Properties
Gödel (minimum)	$T_G(a, b) = \min(a, b)$	idempotent, continuous
Product	$T_P(a, b) = a \cdot b$	strict
Lukasiewicz	$T_{LK}(a, b) = \max(a + b - 1, 0)$	continuous
Drastic product	$T_D(a, b) = \begin{cases} \min(a, b), & \text{if } a = 1 \text{ or } b = 1 \\ 0, & \text{otherwise} \end{cases}$	
Nilpotent minimum	$T_{nM}(a, b) = \begin{cases} 0, & \text{if } a + b \leq 1 \\ \min(a, b), & \text{otherwise} \end{cases}$	left-continuous
Yager	$T_Y(a, b) = \max(1 - ((1 - a)^p + (1 - b)^p)^{\frac{1}{p}}, 0), p \geq 1$	continuous

Table 1: The t-norms of interest.

**Definition 1.** A function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is called

- a) **continuous** if for all  $a \in D$ ,  $\lim_{x \rightarrow a} f(x) = f(a)$ ;
- b) **left-continuous** if for all arbitrarily small  $\epsilon > 0$  there exists another value  $\delta > 0$  such that for all  $a \in D$  it holds that  $|f(x) - f(a)| < \epsilon$  whenever  $a - \delta < x < a$ , and similarly for **right-continuous**;
- c) **increasing** if for all  $a, b \in D$ , if  $a \leq b$  then  $f(a) \leq f(b)$ , and similarly for **decreasing**;
- d) **strictly increasing** if for all  $a, b \in D$ , if  $a < b$  then  $f(a) < f(b)$ , and similarly for **strictly decreasing**.

Left-continuity informally means that when a point is approached from the left, no ‘jumps’ will occur.

Moreover, a function  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  is called

- a) **commutative** if for all  $a, b \in D$ ,  $f(a, b) = f(b, a)$ .
- b) **associative** if for all  $a, b, c \in D$ ,  $f(f(a, b), c) = f(a, f(b, c))$ .

### 3.2.1. Fuzzy Negation

The functions that are used to compute the negation of a truth value of a formula are called **fuzzy negations**.

**Definition 2.** A **fuzzy negation** is a function  $N : [0, 1] \rightarrow [0, 1]$  so that  $N(0) = 1$  and  $N(1) = 0$ .  $N$  is called **strict** if it is strictly decreasing and continuous, and **strong** if for all  $a \in [0, 1]$ ,  $N(N(a)) = a$ .

In this paper we will exclusively use the strict and strong classic negation  $N_C(a) = 1 - a$ .

### 3.2.2. Triangular Norms

The functions that are used to compute the conjunction of two truth values are called **t-norms**. For a rigorous overview, see Klement et al. (2013).

**Definition 3.** A **t-norm** (triangular norm) is a function  $T : [0, 1]^2 \rightarrow [0, 1]$  that is commutative and associative, and

1. **Monotonicity:** For all  $a \in [0, 1]$ ,  $T(a, \cdot)$  is increasing and
2. **Neutrality:** For all  $a \in [0, 1]$ ,  $T(1, a) = a$ .

The phrase ‘ $T(a, \cdot)$  is increasing’ means that whenever  $0 \leq b_1 \leq b_2 \leq 1$ , then  $T(a, b_1) \leq T(a, b_2)$ .

**Definition 4.** A t-norm  $T$  can have the following properties:

- a) **Continuity:** A continuous t-norm is continuous in both arguments.
- b) **Left-continuity:** A left-continuous t-norm is left-continuous in both arguments.
- c) **Idempotency:** An idempotent t-norm has the property that for all  $a \in [0, 1]$ ,  $T(a, a) = a$ .
- d) **Strict-monotony:** A strictly monotone t-norm has the property that for all  $a \in [0, 1]$ ,  $T(a, \cdot)$  is strictly increasing.
- e) **Strict:** A strict t-norm is continuous and strictly monotone.

Table 1 shows the four basic t-norms and two other t-norms of interest alongside their properties.

Name	T-conorm	Properties
Gödel (maximum)	$S_G(a, b) = \max(a, b)$	idempotent, continuous
Product (probabilistic sum)	$S_P(a, b) = a + b - a \cdot b$	strict
Lukasiewicz	$S_{LK}(a, b) = \min(a + b, 1)$	continuous
Drastic sum	$S_D(a, b) = \begin{cases} \max(a, b), & \text{if } a = 0 \text{ or } b = 0 \\ 1, & \text{otherwise} \end{cases}$	
Nilpotent maximum	$S_{nM}(a, b) = \begin{cases} 1, & \text{if } a + b \geq 1 \\ \max(a, b), & \text{otherwise} \end{cases}$	right-continuous
Yager	$S_Y(a, b) = \min((a^p + b^p)^{\frac{1}{p}}, 1), p \geq 1$	continuous

Table 2: The t-conorms of interest.

### 3.2.3. Triangular Conorms

The functions that are used to compute the disjunction of two truth values are called **t-conorms** or **s-norms**.

**Definition 5.** A **t-conorm** (triangular conorm, also known as s-norm) is a function  $S : [0, 1]^2 \rightarrow [0, 1]$  that is commutative and associative, and

1. **Monotonicity:** For all  $a \in [0, 1]$ ,  $S(a, \cdot)$  is increasing and
2. **Neutrality:** For all  $a \in [0, 1]$ ,  $S(0, a) = a$ .

T-conorms are obtained from t-norms using De Morgan's laws from classical logic, i.e.  $p \vee q = \neg(\neg p \wedge \neg q)$ . Therefore, if  $T$  is a t-norm and  $N_C$  the classical negation,  $T$ 's  $N_C$ -dual  $S$  is calculated using

$$S(a, b) = 1 - T(1 - a, 1 - b) \quad (1)$$

Table 2 shows several common t-conorms derived using Equation 1 and the t-norms from Table 1, alongside the same optional properties as those for t-norms in Definition 4.

### 3.2.4. Aggregation operators

The functions that are used to compute quantifiers like  $\forall$  and  $\exists$  are aggregation operators (Liu and Kerre, 1998).

**Definition 6.** An **aggregation operator** is a function  $A : [0, 1]^n \rightarrow [0, 1]$  that is symmetric and increasing with respect to each argument, and for which  $A(0, \dots, 0) = 0$   $A(1, \dots, 1) = 1$ . A **symmetric** function is one in which the output value is the same for every ordering of its arguments.

Aggregation operators are *variadic functions* which are functions that are defined for any finite set of arguments. For this reason we will often use the notation  $\mathbf{A}_{i=1}^n x_i := A(x_1, \dots, x_n)$ . Table 3 shows some common aggregation operators that we will talk about.

The  $\forall$  quantifier is interpreted as the conjunction over all arguments  $x$ . Therefore, we can extend a t-norm  $T$  from 2-dimensional inputs to  $n$ -dimensional inputs as they are commutative and associative (Klement et al., 2013):

$$\begin{aligned} A_T(x_1) &= x_1 \\ A_T(x_1, x_2, \dots, x_n) &= T(x_1, A_T(x_2, \dots, x_n)) \end{aligned} \quad (2)$$

These operators are a straightforward choice for modelling the  $\forall$  quantifier, as they can be seen as a series of conjunctions. We can do the same for a t-conorm  $S$  to model the  $\exists$  quantifier:

$$\begin{aligned} A_S(x_1) &= x_1 \\ A_S(x_1, x_2, \dots, x_n) &= S(x_1, A_S(x_2, \dots, x_n)) \end{aligned} \quad (3)$$

### 3.3. Fuzzy Implications

The functions that are used to compute the truth value of  $p \rightarrow q$  are called fuzzy implications.  $p$  is called the **antecedent** and  $q$  the **consequent** of the implication. We follow Jayaram and Baczynski (2008) and refer to it for details and proofs.

**Definition 7.** A **fuzzy implication** is a function  $I : [0, 1]^2 \rightarrow [0, 1]$  so that for all  $a, c \in [0, 1]$ ,  $I(\cdot, c)$  is decreasing,  $I(a, \cdot)$  is increasing and for which  $I(0, 0) = 1$ ,  $I(1, 1) = 1$  and  $I(1, 0) = 0$ .

Name	Generalizes	Aggregation operator
Minimum	$T_G$	$A_{T_G}(x_1, \dots, x_n) = \min(x_1, \dots, x_n)$
Product	$T_P$	$A_{T_P}(x_1, \dots, x_n) = \prod_{i=1}^n x_i$
Lukasiewicz	$T_{LK}$	$A_{T_{LK}}(x_1, \dots, x_n) = \max(\sum_{i=1}^n x_i - (n-1), 0)$
Maximum	$S_G$	$A_{S_G}(x_1, \dots, x_n) = \max(x_1, \dots, x_n)$
Probabilistic sum	$S_G$	$A_{SP}(x_1, \dots, x_n) = 1 - \prod_{i=1}^n (1 - x_i)$
Bounded sum	$S_{LK}$	$A_{S_{LK}}(x_1, \dots, x_n) = \min(\sum_{i=1}^n x_i, 1)$

Table 3: Some common aggregation operators.

Name	T-conorm	S-implication	Properties
Gödel (Kleene-Dienes)	$S_G$	$I_{KD}(a, c) = \max(1 - a, c)$	All but IP
Product (Reichenbach)	$S_P$	$I_{RC}(a, c) = 1 - a + a \cdot c$	All but IP
Lukasiewicz	$S_{LK}$	$I_{LK}(a, c) = \min(1 - a + c, 1)$	All
Dubouis-Prade	$S_D$	$I_{DP}(a, c) = \begin{cases} c, & \text{if } a = 1 \\ 1 - a, & \text{if } c = 0 \\ 1, & \text{otherwise} \end{cases}$	All
Nilpotent (Fodor)	$S_{Nm}$	$I_{FD}(a, c) = \begin{cases} 1, & \text{if } a \leq c \\ \max(1 - a, c), & \text{otherwise} \end{cases}$	All

Table 4: Some common S-implications formed from Equation 4 using  $N_C$  and the common t-conorms from Table 2.

From this definition follows that  $I(0, 1) = 1$ .

**Definition 8.** Let  $N$  be a fuzzy negation. A fuzzy implication  $I$  satisfies

- a) **left-neutrality (LN)** if for all  $c \in [0, 1]$ ,  $I(1, c) = c$ ;
- b) the **exchange principle (EP)** if for all  $a, b, c \in [0, 1]$ ,  $I(a, I(b, c)) = I(b, I(a, c))$ ;
- c) the **identity principle (IP)** if for all  $a \in [0, 1]$ ,  $I(a, a) = 1$ ;
- d)  **$N$ -contrapositive symmetry (CP)** if for all  $a, c \in [0, 1]$ ,  $I(a, c) = I(N(c), N(a))$ ;
- e)  **$N$ -left-contrapositive symmetry (L-CP)** if for all  $a, c \in [0, 1]$ ,  $I(N(a), c) = I(N(c), a)$ ;
- f)  **$N$ -right-contrapositive symmetry (R-CP)** if for all  $a, c \in [0, 1]$ ,  $I(a, N(c)) = I(c, N(a))$ .

All these statements generalize a law from classical logic. *Left neutrality* generalizes  $(1 \rightarrow p) \equiv p$ , the *exchange principle* generalizes  $p \rightarrow (q \rightarrow r) \equiv q \rightarrow (p \rightarrow r)$ , and the *identity principle* generalizes that  $p \rightarrow p$  is a tautology. Furthermore,  *$N$ -contrapositive symmetry* generalizes  $p \rightarrow q \equiv \neg q \rightarrow \neg p$ ,  *$N$ -left-contrapositive symmetry* generalizes  $\neg p \rightarrow q \equiv \neg q \rightarrow p$  and  *$N$ -right-contrapositive symmetry* generalizes  $p \rightarrow \neg q \equiv q \rightarrow \neg p$ .

### 3.3.1. S-Implications

In classical logic, the (material) implication is defined as follows:

$$p \rightarrow q = \neg p \vee q$$

Using this definition, we can use a t-conorm  $S$  and a fuzzy negation  $N$  to construct a fuzzy implication.

**Definition 9.** Let  $S$  be a t-conorm and  $N$  a fuzzy negation. The function  $I_{S,N} : [0, 1]^2 \rightarrow [0, 1]$  is called an **(S, N)-implication** and is defined for all  $a, c \in [0, 1]$  as

$$I_{S,N}(a, c) = S(N(a), c). \quad (4)$$

If  $N$  is a strong fuzzy negation, then  $I_{S,N}$  is called an **S-implication** (or strong implication).

As we only consider the classical negation  $N_C$ , we omit the  $N$  and use  $I_S$  to refer to  $I_{S,N_C}$ .

All S-implications  $I_S$  are fuzzy implications and satisfy LN, EP and R-CP. Additionally, if the negation  $N$  is strong, it satisfies CP and if, in addition, it is strict, it also satisfies L-CP. In Table 4 we show several S-implications that use the strong fuzzy negation  $N_C$  and the common t-conorms (Table 2). Note that S-implications are rotations of the t-conorms.

### 3.3.2. R-Implications

R-implications are another way of constructing implication operators. They are the standard choice in t-norm fuzzy logics.

Name	T-norm	R-implication	Properties
Gödel	$T_G$	$I_G(a, c) = \begin{cases} 1, & \text{if } a \leq c \\ c, & \text{otherwise} \end{cases}$	LN, EP, IP
product (Goguen)	$T_P$	$I_{GG}(a, c) = \begin{cases} 1, & \text{if } a \leq c \\ \frac{c}{a}, & \text{otherwise} \end{cases}$	LN, EP, IP
Lukasiewicz	$T_{LK}$	$I_{LK}(a, c) = \min(1 - a + c, 1)$	All
Weber	$T_D$	$I_{WB}(a, c) = \begin{cases} 1, & \text{if } a < 1 \\ c, & \text{otherwise} \end{cases}$	LN, EP, IP
nilpotent (Fodor)	$T_{Nm}$	$I_{FD}(a, c) = \begin{cases} 1, & \text{if } a \leq c \\ \max(1 - a, c), & \text{otherwise} \end{cases}$	All

Table 5: The R-implications constructed using the t-norms from Table 1.

**Definition 10.** Let  $T$  be a t-norm. The function  $I_T : [0, 1]^2 \rightarrow [0, 1]$  is called an **R-implication** and defined as

$$I_T(a, c) = \sup\{b \in [0, 1] | T(a, b) \leq c\} \quad (5)$$

The **supremum** of a set  $A$ , denoted  $\sup\{A\}$ , is the lowest upper bound of  $A$ . All R-implications are fuzzy implications, and all satisfy LN, IP and EP.  $T$  is a left-continuous t-norm if and only if the supremum can be replaced with the maximum function. Note that if  $a \leq c$  then  $I_T(a, c) = 1$ . We can see this by looking at Equation 5. The largest value for  $b$  possible is 1, since then, using the *neutrality* property of t-norms,  $T(a, 1) = a \leq c$ .

Table 5 shows the four R-implications created from the four common T-norms. Note that  $I_{LK}$  and  $I_{FD}$  appear in both tables: They are both S-implications and R-implications.

#### 4. Differentiable Fuzzy Logics

We next discuss **Differentiable Fuzzy Logics** (DFL), a family of Differentiable Logics based on fuzzy logic. Truth values of ground atoms are continuous, and logical connectives are interpreted using fuzzy operators. In principle, DFL can handle both predicates and functions. To ease the discussion, we will not analyze functions, constants and existential quantifiers and thus leave them out of the discussion.<sup>1</sup> We follow both Real Logic in Serafini and Garcez (2016) and *embeddings based semantics* in Guha (2015) as an introduction.

##### 4.1. Semantics

DFL defines a new semantics using vector embeddings and functions on such vectors in place of classical semantics. In classical logic, a **structure** consists of a domain of discourse and an interpretation function, and is used to give meaning to the predicates. Similarly, in DFL a structure consists of a probability distribution defined on an embedding space and an *embedded interpretation*.<sup>2</sup>

**Definition 11.** A **Differentiable Fuzzy Logics** structure is a tuple  $\langle p, \eta_\theta \rangle$ , where  $p$  is a **domain distribution** over  $d$ -dimensional<sup>3</sup> objects  $o \in \mathbb{R}^d$ . The **domain of discourse** is the **support** of  $p$ , that is  $O = \text{supp}(p) = \{o | p(o) > 0, o \in \mathbb{R}^d\}$ .  $\eta_\theta$  is an **(embedded) interpretation**, which is a function parameterized by  $\theta$  so that for all predicate symbols  $P \in \mathcal{P}$  with arity  $\alpha$ ,  $\eta_\theta(P) : O^\alpha \rightarrow [0, 1]$ .

To address the *symbol grounding problem* (Harnad, 1990), objects in DFL semantics are  $d$ -dimensional vectors of reals. Their semantics come from the underlying semantics of the vector space as terms are interpreted in a real (valued) world (Serafini and Garcez, 2016). Likewise, predicates are interpreted as functions mapping these vectors to a fuzzy truth value. The domain distribution is used to limit the size of the vector space. For example,  $p$  might be the distribution over images representing only the natural images. Embedded interpretations can

<sup>1</sup>Existential quantification can be modeled in a similar way to universal quantification, but by using operators generalizing t-conorms instead (similar to how t-norms are used to model universal quantification). Furthermore, functions and constants are modelled in Serafini and Garcez (2016) and Marra et al. (2018).

<sup>2</sup>Serafini and Garcez (2016) uses the term “(semantic) grounding” or “symbol grounding” (Mayo, 2003) instead of ‘embedded interpretation’, “to emphasize the fact that  $\mathcal{L}$  is interpreted in a ‘real world’” but we find this potentially confusing as this could also refer to groundings in Herbrand semantics. Furthermore, by using the word ‘interpretation’ we highlight the parallel with classical logical interpretations.

<sup>3</sup>Without loss of generality we fix the dimensionality of the vectors representing the objects. Extensions to a varying number of dimensions are straightforward by introducing types.

be implemented using any deep learning model<sup>4</sup>. Different values of the trainable parameters  $\theta$  will produce different interpretations  $\eta_\theta$  and so we include  $\theta$  in the notation.

Next, we define how to compute the truth value of sentences of DFL.

**Definition 12.** A **variable assignment**  $\mu$  maps variable symbols  $x$  to objects  $o \in O$ .  $\mu(x)$  retrieves the object  $o \in O$  assigned to  $x$  in  $\mu$ .

**Definition 13.** Let  $\langle p, \eta_\theta \rangle$  be a DFL structure,  $N$  a fuzzy negation,  $T$  a t-norm,  $S$  a t-conorm,  $I$  a fuzzy implication and  $A$  an aggregation operator. Then the **valuation function**  $e_{\eta_\theta, p, N, T, S, I, A}$  (or, for brevity,  $e_\theta$ ) computes the truth value of a formula  $\varphi$  in  $\mathcal{L}$  given a variable assignment  $\mu$ . It is defined inductively as follows:

$$e_\theta(\mathsf{P}(x_1, \dots, x_m)) = \eta_\theta(\mathsf{P})(\mu(x_1), \dots, \mu(x_m)) \quad (6)$$

$$e_\theta(\neg\phi) = N(e_\theta(\phi)) \quad (7)$$

$$e_\theta(\phi \wedge \psi) = T(e_\theta(\phi), e_\theta(\psi)) \quad (8)$$

$$e_\theta(\phi \vee \psi) = S(e_\theta(\phi), e_\theta(\psi)) \quad (9)$$

$$e_\theta(\phi \rightarrow \psi) = I(e_\theta(\phi), e_\theta(\psi)) \quad (10)$$

$$e_\theta(\forall x \phi) = \bigwedge_{o \in O} e_\theta(\phi), \quad \text{where } x \text{ is assigned to } o \text{ in } \mu. \quad (11)$$

Equation 6 defines the fuzzy truth value of an atomic formula.  $l$  finds the objects assigned to the terms  $x_1, \dots, x_m$  resulting in a list of  $d$ -dimensional vectors. These are the inputs to the interpretation of the predicate symbol  $\eta_\theta(\mathsf{P})$  to get a fuzzy truth value. Equations 7 - 10 define the truth values of the connectives using the operators  $N, T, S$  and  $I$ . Finally, Equation 11 defines the truth value of universally quantified formulas  $\forall x \phi$ . This is done by enumerating the domain of discourse  $o \in O$ , computing the truth value of  $\phi$  with  $o$  assigned to  $x$  in  $\mu$ , and combining the truth values using an aggregation operator  $A$ .

#### 4.2. Relaxing Quantifiers

For infinite domains, or for domains that are so large that we cannot compute the full semantics of the  $\forall$  quantifier, we can choose to sample a batch of  $b$  objects from  $O$  to approximate the computation of the valuation. This can be done by replacing Equation 11 with

$$e_\theta(\forall x \phi) = \bigwedge_{i=1}^b e_\theta(\phi), \quad o_1, \dots, o_b \text{ chosen from } O. \quad (12)$$

An obvious way would be to sample from the domain distribution  $p$ , if it is available. It is commonly assumed in Machine Learning Goodfellow et al. (2016)(p.109) that a dataset  $\mathcal{D}$  contains independent samples from the domain distribution  $p$  and thus using such samples approximates sampling from  $p$ . Unfortunately, by relaxing quantifiers in this way we lose soundness of the logic.

#### 4.3. Learning using Fuzzy Maximum Satisfiability

In DFL, the parameters  $\theta$  are learned using *fuzzy maximum satisfiability* (Donadello et al., 2017), which finds parameters that maximize the valuation of the knowledge base  $\mathcal{K}$ .

**Definition 14.** Let  $\mathcal{K}$  be a knowledge base of formulas,  $\langle p, \eta_\theta \rangle$  a DFL structure for the predicate symbols in  $\mathcal{K}$  and  $e_{\eta_\theta, p, N, T, S, I, A}$  a valuation function. Then the **Differentiable Fuzzy Logics loss**  $\mathcal{L}_{DFL}$  of a knowledge base of formulas  $\mathcal{K}$  is computed using

$$\mathcal{L}_{DFL}(\theta; O, \mathcal{K}) = - \sum_{\varphi \in \mathcal{K}} w_\varphi \cdot e_{\eta_\theta, p, N, T, S, I, A}(\varphi), \quad (13)$$

where  $w_\varphi$  is the weight for formula  $\varphi$  which denotes the importance of the formula  $\varphi$  in the loss function. The **fuzzy maximum satisfiability problem** is the problem of finding parameters  $\theta^*$  that minimize Equation 13:

$$\theta^* = \operatorname{argmin}_\theta \mathcal{L}_{DFL}(\theta; O, \mathcal{K}). \quad (14)$$

This optimization problem can be solved using a gradient descent-like method. If the operators  $N, T, S, I$  and  $A$  are all differentiable, we can repeatedly apply the chain rule, i.e. reverse-mode differentiation, on the DFL loss  $\mathcal{L}_{DFL}(\theta_n; O, \mathcal{K})$ ,  $n = 0, \dots, N$ . This procedure finds the derivative with respect to the truth values of

---

<sup>4</sup>We use ‘models’ to refer to deep learning models like neural networks, and not to models from model theory.

the ground atoms  $\frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\mathbf{P})(o_1, \dots, o_m)}$ . We can use these partial derivatives to update the parameters  $\boldsymbol{\theta}_n$ , resulting in a different embedded interpretation  $\eta_{\boldsymbol{\theta}_{n+1}}$ . This procedure is computed as follows:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \epsilon \cdot \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \boldsymbol{\theta}_n} = \boldsymbol{\theta}_n - \epsilon \cdot \sum_{\mathbf{P}(o_1, \dots, o_m)} \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\mathbf{P})(o_1, \dots, o_m)} \cdot \frac{\partial \eta_{\boldsymbol{\theta}_n}(\mathbf{P})(o_1, \dots, o_m)}{\partial \boldsymbol{\theta}_n}, \quad (15)$$

where  $\epsilon$  is the learning rate. We refer for implementation details to Appendix A.

**Example 2.** To illustrate the computation of the valuation function  $e_{\boldsymbol{\theta}}$ , we return to the problem in Example 1. The *domain of discourse* is the set of subimages of natural images. The *domain distribution* is a distribution over those subimages. We take  $\{o_1, o_2\}$  as the batch for the aggregation operator. The valuation of the formula  $\varphi = \forall x, y \text{ chair}(x) \wedge \text{partOf}(y, x) \rightarrow \text{cushion}(y) \vee \text{armRest}(y)$  is computed as:

$$\begin{aligned} e_{\boldsymbol{\theta}}(\varphi) = & A(A(I(T(\eta_{\boldsymbol{\theta}}(\text{chair})(o_1), \eta_{\boldsymbol{\theta}}(\text{partOf})(o_1, o_1)), S(\eta_{\boldsymbol{\theta}}(\text{cushion})(o_1), \eta_{\boldsymbol{\theta}}(\text{armRest})(o_1))), \\ & I(T(\eta_{\boldsymbol{\theta}}(\text{chair})(o_1), \eta_{\boldsymbol{\theta}}(\text{partOf})(o_2, o_1)), S(\eta_{\boldsymbol{\theta}}(\text{cushion})(o_2), \eta_{\boldsymbol{\theta}}(\text{armRest})(o_2))), \\ & A(I(T(\eta_{\boldsymbol{\theta}}(\text{chair})(o_2), \eta_{\boldsymbol{\theta}}(\text{partOf})(o_1, o_2)), S(\eta_{\boldsymbol{\theta}}(\text{cushion})(o_1), \eta_{\boldsymbol{\theta}}(\text{armRest})(o_1))), \\ & I(T(\eta_{\boldsymbol{\theta}}(\text{chair})(o_2), \eta_{\boldsymbol{\theta}}(\text{partOf})(o_2, o_2)), S(\eta_{\boldsymbol{\theta}}(\text{cushion})(o_2), \eta_{\boldsymbol{\theta}}(\text{armRest})(o_2)))) \end{aligned}$$

Next, we choose the operators as  $T = T_P$ ,  $S = S_P$ ,  $A = A_{T_P}$  and  $I = I_{RC}$ . The computation of the valuation function can then be written as

$$e_{\boldsymbol{\theta}}(\varphi) = \prod_{x, y \in C} 1 - \eta_{\boldsymbol{\theta}}(\text{chair})(x) \cdot \eta_{\boldsymbol{\theta}}(\text{partOf})(y, x) \cdot (1 - \eta_{\boldsymbol{\theta}}(\text{cushion})(y))(1 - \eta_{\boldsymbol{\theta}}(\text{armRest})(y))$$

If we interpret the predicate functions using a lookup in the table on the probabilities from Example 1 so that  $\eta_{\boldsymbol{\theta}_n}(\mathbf{P}(x)) = p(\mathbf{P}(x)|I, x)$ , we find that  $e_{\boldsymbol{\theta}_n}(\varphi) = 0.612$ . Taking  $\mathcal{K} = \{\varphi\}$ , we find using repeated applications of the chain rule that

$$\begin{aligned} \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{chair})(o_1)} &= -0.4261 & \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{chair})(o_2)} &= -0.0058 \\ \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{cushion})(o_1)} &= 0.0029 & \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{cushion})(o_2)} &= 0.7662 \\ \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{armRest})(o_1)} &= 0.0029 & \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{armRest})(o_2)} &= 0.4257 \\ \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{partOf})(o_1, o_1)} &= -0.4978 & \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{partOf})(o_2, o_2)} &= -0.1103 \\ \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{partOf})(o_1, o_2)} &= -0.2219 & \frac{\partial \mathcal{L}_{DFL}(\boldsymbol{\theta}_n; O, \mathcal{K})}{\partial \eta_{\boldsymbol{\theta}_n}(\text{partOf})(o_2, o_1)} &= -0.4031. \end{aligned}$$

We can now do a gradient update step to update the probabilities in the table of probabilities from Example 1, or find what the partial derivative of the parameters  $\boldsymbol{\theta}_n$  of some deep learning model  $p_{\boldsymbol{\theta}_n}$  should be using Equation 15.

One particularly interesting property of Differentiable Fuzzy Logics is that the partial derivatives of the subformulas with respect to the satisfaction of the knowledge base have a somewhat explainable meaning. For example, as hypothesized in Example 1, the computed gradients reflect that we should increase  $p(\text{cushion}(o_2)|I, o_2)$ , as it is indeed the (*absolute*) largest partial derivative.

## 5. Derivatives of Operators

We will now show that the choice of operators that are used for the logical connectives actually determines the inferences that are done when using DFL. If we used a different set of operators in Example 2, we would have gotten very different derivatives. These could in some cases make more sense, and in some other cases less. Furthermore, it is much easier to find a global minimum of the fuzzy maximum satisfiability problem (Equation 14) for some operators than for others. This is often because of the smoothness of the operators. In this section, we analyze a wide variety of functions that can be used for logical reasoning and present some of their properties that determine how useful they are in inferences such as those illustrated above.

We will not discuss any varieties of fuzzy negations since the classical negation  $N_C(a) = 1 - a$  is already continuous, intuitive and has simple derivatives.

**Definition 15.** A function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is said to be **nonvanishing** if  $f(a) \neq 0$  for all  $a \in \mathbb{R}$ , i.e. it is nonzero everywhere.

A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  has a **nonvanishing derivative** if for all  $a_1, \dots, a_n \in \mathbb{R}$  there is some  $1 \leq i \leq n$  such that  $\frac{\partial f(a_1, \dots, a_n)}{\partial a_i} \neq 0$ .

Whenever an operator vanishes, it loses its learning signal. Notice that derivatives of composites of nonvanishing functions can still be vanishing. For instance, using the product t-conorm for  $a \vee \neg a$ , we find that the derivative of  $S_P(a, 1 - a)$  is 0 at  $\frac{1}{2}$ . Furthermore, all the partial derivatives of the connectives used in the backward pass from the valuation function to the ground atoms have to be multiplied. If the partial derivatives are less than 1, their product will also approach 0.

The drastic product  $T_D$  and operators derived from it such as the drastic sum  $S_D$  and the Dubois-Prade and Weber implications ( $I_{DP}$  and  $I_{WB}$ ) have vanishing derivatives almost everywhere. In deep learning models, output probabilities are the result of transformations on real numbers using functions like the sigmoid or softmax that result in truth values in  $(0, 1)$ . The operators derived from  $T_D$  only have nonvanishing derivatives when their inputs are exactly 0 or 1, making them not very useful for this application.

**Definition 16.** A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is said to be **single-passing** if for all  $x_1, \dots, x_n \in [0, 1]$  it holds that  $\left| \left\{ x_i \mid \frac{\partial f(x_1, \dots, x_n)}{\partial x_i} \neq 0, i \in \{1, \dots, n\} \right\} \right| \leq 1$ .

A single-passing function has nonzero derivatives on at most one input argument. Using just single-passing Fuzzy Logic operators can be inefficient, since then at most one input will have a nonzero derivative (i.e. a learning signal), yet the complete forward pass still has to be computed to find this input.

**Proposition 1.** Any composition of single-passing functions is also single-passing.

For the proof, see Appendix B.1.

Concluding, for any logical operator to be usable in the learning task, it will need to have a nonvanishing derivative at the majority of the input signals (so that it can contribute to the learning signal at all), and ideally to not be single-passing (so that it can contribute efficiently to the learning signal).

## 6. Aggregation

After the global considerations from the previous section, we next analyze in detail each aggregation operator separately and outline their benefits and disadvantages. As explained above, we limit ourselves to aggregation functions for universal quantification only.

### 6.1. Minimum Aggregator

The minimum aggregator is given as  $A_{T_G}(x_1, \dots, x_n) = \min(x_1, \dots, x_n)$ . The partial derivatives are

$$\frac{\partial A_{T_G}(x_1, \dots, x_n)}{\partial x_i} = \begin{cases} 1 & \text{if } i = \operatorname{argmin}_{j \in \{1, \dots, n\}} x_j \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

It is single-passing with the only nonzero gradient being on the input with the lowest truth value. Many practical formulas have exceptions. An exception to a formula like  $\forall x \text{ Raven}(x) \rightarrow \text{Black}(x)$  would be a raven over which a bucket of red paint is thrown. The minimum aggregator would have a derivative on that exception when ‘red raven’ is correctly predicted. Additionally, it is inefficient, as we still have to compute the forward pass for inputs that do not get a feedback signal.

### 6.2. Lukasiewicz Aggregator

The Lukasiewicz aggregator is given as  $A_{T_{LU}}(x_1, \dots, x_n) = \max(\sum_{i=1}^n x_i - (n-1), 0)$ . The partial derivatives are given by

$$\frac{\partial A_{T_{LU}}(x_1, \dots, x_n)}{\partial x_i} = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i > n-1 \\ 0 & \text{otherwise.} \end{cases} \quad (17)$$

The gradient is nonvanishing only when  $\sum_{i=1}^n x_i > n-1$ , that is, when the average value of  $x_i$  is larger than  $\frac{n-1}{n}$  (Pál Jónsson, 2018). As  $\lim_{n \rightarrow \infty} \frac{n-1}{n} = 1$ , for larger values of  $n$ , all inputs have to be true for this condition to hold.

For the next proposition, we refer to the **fraction of inputs** for which some condition holds. The probability that the condition holds for a point uniformly sampled from  $[0, 1]^n$  is this fraction.

**Proposition 2.** The fraction of inputs  $x_1, \dots, x_n \in [0, 1]$  for which the derivative of  $A_{T_{LU}}$  is nonvanishing is  $\frac{1}{n!}$ .

For proof, see Appendix B.2.1. Clearly, for the majority of inputs there is a vanishing gradient, implying that this aggregator would not be useful in a DFL learning setting.

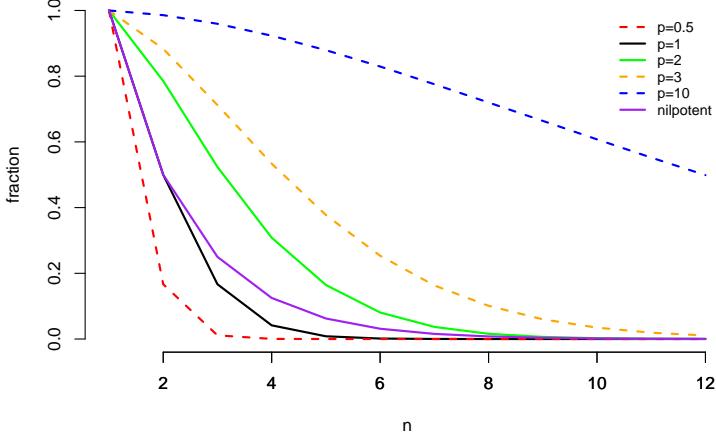


Figure 2: The fraction of inputs for which the Yager aggregator  $A_{TY}$  for several values of  $p$  and the nilpotent minimum aggregator  $A_{TM}$  have nonvanishing derivatives. The values for dotted lines are estimated using Monte Carlo simulation.

### 6.3. Yager Aggregator

The Yager aggregator is given by

$$A_{TY}(x_1, \dots, x_n) = \max \left( 1 - \left( \sum_{i=1}^n (1-x_i)^p \right)^{\frac{1}{p}}, 0 \right), \quad p \geq 1 \quad (18)$$

Where  $p = 1$  this corresponds to the Lukasiewicz aggregator,  $p = \infty$  corresponds to the minimum aggregator and  $p = 0$  corresponds to the aggregator formed from the drastic product  $A_{TD}$ . The derivative of the Yager aggregator is

$$\frac{\partial A_{TY}(x_1, \dots, x_n)}{\partial x_i} = \begin{cases} \left( \sum_{j=1}^n (1-x_j)^p \right)^{1-\frac{1}{p}} \cdot (1-x_i)^{p-1} & \text{if } \left( \sum_{j=1}^n (1-x_j)^p \right)^{\frac{1}{p}} < 1 \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

This derivative vanishes whenever  $\sum_{j=1}^n (1-x_j)^p \geq 1$ . As  $1-x_i \in [0, 1]$ ,  $(1-x_i)^p$  is a decreasing function with respect to  $p$ . Therefore,  $\sum_{i=1}^n (1-x_i)^p < 1$  holds for a larger fraction of inputs when  $p$  increases, with the fraction being 0 for  $p = 0$  as it corresponds to the drastic aggregator and 1 for  $p = \infty$ .

The exact fraction of inputs with a nonvanishing derivative is hard to express.<sup>5</sup> However, we can find a closed-form expression for the Euclidean case  $p = 2$ .

**Proposition 3.** The fraction of inputs  $x_1, \dots, x_n \in [0, 1]$  for which the derivative of  $A_{TY}$  with  $p = 2$  is nonvanishing is  $\frac{\pi^{\frac{n}{2}}}{2^n \cdot \Gamma(\frac{1}{2}n + \frac{1}{2})}$ , where  $\Gamma$  is the Gamma function.

See Appendix B.2.2 for proof. We plot the fraction of nonvanishing derivatives for several values of  $p$  in Figure 2. For fairly small  $p$ , the vast majority of the inputs will have a vanishing derivative, and similar for high  $n$ , showing that this aggregator is also of little use in a learning context.

### 6.4. Mean- $p$ Error Aggregator

If we are concerned only in maximizing the truth value of  $A_{TY}$ , we can simply remove the *max* constraint, resulting in an ‘unbounded Yager’ aggregator that has a nonvanishing derivative everywhere. However, then the co-domain of the function is no longer  $[0, 1]$ . We can do a linear transformation on this function to ensure this is the case (Appendix C.1).

<sup>5</sup>Assume that  $x_1, \dots, x_n \sim U(0, 1)$  are independently and standard uniformly distributed. Note that  $z_i = 1 - x_i$  is also standard uniformly distributed.  $z_i^p$  is distributed by the beta distribution  $Beta(1/p, 1)$  (Gupta and Nadarajah, 2004).  $Y = \sum_{i=1}^n z_i$  is the sum of  $n$  such beta-distributed variables. Unfortunately, there is no closed-form expression for the probability density function of sums of independent beta random variables (Pham and Turkkan, 1994). A suitable approximation would be to use the central limit theorem as  $z_1, \dots, z_n$  are identically and independently distributed.

**Definition 17.** For some  $p \geq 0$ , the Mean- $p$  Error aggregator  $A_{pME}$  is defined as

$$A_{pME}(x_1, \dots, x_n) = 1 - \left( \frac{1}{n} \sum_{i=1}^n (1 - x_i)^p \right)^{\frac{1}{p}}. \quad (20)$$

The ‘error’ here is difference between the predicted value  $x_i$  and the ‘ground truth’ value, 1. This function has the following derivative:

$$\frac{\partial A_{pME}(x_1, \dots, x_n)}{\partial x_i} = (1 - x_i)^{p-1} \frac{1}{n} \left( \sum_{j=1}^n (1 - x_j)^p \right)^{\frac{1}{p}-1}. \quad (21)$$

When  $p > 1$ , this derivative is largest for the inputs that are smallest, which can speed up the optimization by being sensitive to outliers. For  $p < 1$ , the opposite is true. A special case is  $p = 1$ :

$$A_{MAE}(x_1, \dots, x_n) = 1 - \frac{1}{n} \sum_{i=1}^n (1 - x_i) \quad (22)$$

having the simple derivative  $\frac{\partial A_{MAE}(x_1, \dots, x_n)}{\partial x_i} = \frac{1}{n}$ . This measure is equal to 1 minus the *mean absolute error* (*MAE*) and is associated with the Lukasiewicz t-norm. Another special case is  $p = 2$ :

$$A_{RMSE}(x_1, \dots, x_n) = 1 - \sqrt{\frac{1}{n} \sum_{i=1}^n (1 - x_i)^2}. \quad (23)$$

This function is equal to 1 minus the *root-mean-square error* (*RMSE*) which is commonly used for regression tasks and heavily weights outliers. We can do the same for the Yager t-conorm  $\min((a^p + b^p)^{1/p}, 1)$  (Appendix C.1):

**Definition 18.** For some  $p \geq 0$ , the  $p$ -Mean aggregator is defined as

$$A_{pM}(x_1, \dots, x_n) = \left( \frac{1}{n} \sum_{i=1}^n x_i^p \right)^{\frac{1}{p}}. \quad (24)$$

The case of  $p = 1$  corresponds to the arithmetic mean and  $p = 2$  to the geometric mean. In contrast to the Mean- $p$  Error, its derivative  $\frac{1}{n} \left( \frac{1}{n} \sum_{j=1}^n x_j^p \right)^{\frac{1}{p}-1} x_i^{p-1}$  has greater values for smaller inputs when  $p < 1$ , and lower values when  $p > 1$ . Note that the arithmetic mean  $A_{1M}$  has the same derivative as the mean absolute error  $A_{MAE}$ . Unlike  $A_{SY}$ , the only maximum of this aggregator is  $x_1, \dots, x_n = 1$ . Therefore, it is not a sensible choice for generalizing the  $\exists$  quantifier, and the  $p$ -Mean error outperforms it for generalizing  $\forall$ .

### 6.5. Product Aggregator

The product aggregator is given as  $A_{TP}(x_1, \dots, x_n) = \prod_{i=1}^n x_i$ . This is also the probability of the intersection of  $n$  independent events. It has the following partial derivatives:

$$\frac{\partial A_{TP}(x_1, \dots, x_n)}{\partial x_i} = \prod_{j=1, j \neq i}^n x_j. \quad (25)$$

The derivative is only nonvanishing if there are fewer than two  $x_i$  so that  $x_i = 0$ . Furthermore, the derivative for  $x_i$  will be decreased if some other input  $x_j$  is low, despite them being independent. Finally, we cannot compute this aggregator in practice due to numerical underflow when multiplying many small numbers. Noting that  $\text{argmax } f(x) = \text{argmax } \log(f(x))$ , we observe that the **log-product aggregator**

$$A_{\log T_P}(x_1, \dots, x_n) = (\log \circ A_{TP})(x_1, \dots, x_n) = \sum_{i=1}^n \log(x_i) \quad (26)$$

can be used for formulas in prenex normal form, as then the truth value of the universal quantifiers is not used for another connective. Unlike the other aggregators, its codomain is the non-positive numbers instead of  $[0, 1]$ . Furthermore, the log-product aggregator can be seen as the log-likelihood function where we take the correct label to be 1, and thus this is similar to cross-entropy minimization. The partial derivatives are

$$\frac{\partial A_{\log T_P}(x_1, \dots, x_n)}{\partial x_i} = \frac{1}{x_i}. \quad (27)$$

In contrast to Equation 25, the values of the other inputs are irrelevant, and derivatives with respect to lower-valued inputs will be far greater as there is a singularity at  $x = 0$  (i.e. the value becomes infinite). We can conclude therefore that the product aggregator is particularly promising as it is nonvanishing almost everywhere and can handle outliers.

### 6.6. Nilpotent Aggregator

The Nilpotent t-norm is given by  $T_{nM}(a, b) = \begin{cases} \min(a, b), & \text{if } a + b > 1 \\ 0, & \text{otherwise.} \end{cases}$

In Appendix C.3 we show that the Nilpotent aggregator  $A_{T_{nM}}$  is equal to

$$A_{T_{nM}}(x_1, \dots, x_n) = \begin{cases} \min(x_1, \dots, x_n), & \text{if } x_i + x_j > 1; x_i \text{ and } x_j \text{ are the two lowest values in } x_1, \dots, x_n \\ 0, & \text{otherwise} \end{cases}. \quad (28)$$

The derivative is found as follows:

$$\frac{\partial A_{T_{nM}}(x_1, \dots, x_n)}{\partial x_i} = \begin{cases} 1, & \text{if } i = \operatorname{argmin}_j x_j \text{ and } x_i + x_j > 1 \text{ where } x_j \text{ is the second lowest value in } x_1, \dots, x_n \\ 0, & \text{otherwise} \end{cases}. \quad (29)$$

Like the minimum aggregator it is single-passing, and like the Lukasiewicz aggregator it has a derivative that vanishes for the majority of the input space, as it vanishes when the sum of the two smallest values is lower than 1.

**Proposition 4.** The fraction of inputs  $x_1, \dots, x_n \in [0, 1]$  for which the derivative of  $A_{T_{nM}}$  is nonvanishing is  $\frac{1}{2^{n-1}}$ .

For proof, see Appendix B.2.3. The fraction of inputs for which there is a nonvanishing derivative is plotted in Figure 2. Again, this means that for larger numbers of inputs  $n$ , this aggregator will vanish on almost every input and is not a useful construction in a learning context.

### 6.7. Summary

The minimum aggregator is computationally inefficient and cannot handle exceptions well. Aggregation operators that vanish when receiving a large amount of inputs will not scale well, and these include operators based on the Yager family of t-norms and the nilpotent aggregator. Removing the bounds from the Yager aggregators introduces interesting connections to loss functions from the classical machine learning literature. This is also the case for the logarithmic version of the product aggregator, which corresponds to the cross-entropy loss function. They have natural means for dealing with outliers, and thus are promising for practical use.

## 7. Conjunction and Disjunction

Next, we analyze the partial derivatives of t-norms and t-conorms, which are used as conjunction and disjunction in Fuzzy Logics. In t-norm Fuzzy Logics, the weak disjunction  $\max(a, b)$ , or the Gödel t-conorm is used instead of the dual t-conorm.

Suppose that we have a t-norm  $T$  and a t-conorm  $S$ . We define the following two quantities, where the choice of taking the partial derivative to  $a$  is without loss of generality, since  $T$  and  $S$  are commutative by definition:

$$d_T(a, b) = \frac{\partial T(a, b)}{\partial a}, \quad d_S(a, b) = \frac{\partial S(a, b)}{\partial a} \quad (30)$$

It should be noted that by Definition 3,  $d_T(a, 1) = 1$  as  $T(a, 1) = a$  for any t-norm  $T$ , and by Definition 5,  $d_S(a, 0) = 1$  as  $S(a, 0) = a$  for any t-conorm  $S$ . Furthermore, we note that if  $S$  is a t-conorm and the  $N_C$ -dual of the t-norm  $T$ , then  $d_S(a, b) = \frac{\partial 1-T(1-a, 1-b)}{\partial a} = -\frac{\partial 1-T(1-a, 1-b)}{\partial 1-a} = \frac{\partial T(1-a, 1-b)}{\partial 1-a} = d_T(1-a, 1-b)$ .

The main difference in analyzing t-norms and t-conorms is that the maximum of  $T(a, b)$  (namely 1) is when both arguments  $a$  and  $b$  are 1. Consequently, in t-conorms, an infinite number of maxima exist. Some of these maxima might be more desirable than others. Referring back to the formula in Example 1, we showed that it is preferable to increase the truth value of  $\text{cushion}(y)$  and not of  $\text{armRest}(y)$ . Similarly, when a conjunct is negated, or when it appears in the antecedent of an implication (like in the aforementioned formula) we have to choose which of the two conjuncts to decrease. By noting that  $d_T(a, b) = d_S(1-a, 1-b)$ , we find that the t-norm “chooses” in the same way its dual t-conorm would “choose”. Similarly, if a disjunction is negated, it will minimize both its arguments in the way that its dual t-norm would maximize its arguments.

**Example 3.** We introduce a running example to analyze the behavior of different t-norms. Let us optimize  $(a \wedge b) \vee (c \wedge \neg a)$  using gradient descent. The truth value of this expression is computed using  $f(a, b, c) = S(T(a, b), T(1-a, c))$ . Using the boundary conditions from Definition 3 and 5, we find the global optima  $a = 1.0$ ,  $b = 1.0$  and  $a = 0.0$ ,  $c = 1.0$ . The derivative to this function can be computed using the multivariable chain rule:

$$\frac{\partial f(a, b, c)}{\partial a} = \frac{\partial S(T(a, b), T(1-a, c))}{\partial T(a, b)} \cdot \frac{\partial T(a, b)}{\partial a} + \frac{\partial S(T(a, b), T(1-a, c))}{\partial T(1-a, c)} \cdot \frac{\partial T(1-a, c)}{\partial a} \quad (31)$$

$$= d_S(T(a, b), T(1-a, c)) \cdot d_T(a, b) - d_S(T(1-a, c), T(a, b)) \cdot d_T(1-a, c) \quad (32)$$

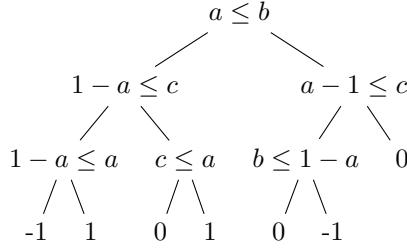


Figure 3: Decision tree for the derivative of  $S(T(a,b), T(1-a,c))$  with respect to  $a$  when using the Gödel t-norm and t-conorm.

### 7.1. Gödel T-Norm

The Gödel t-norm is  $T_G(a,b) = \min(a,b)$  and the Gödel t-conorm is  $S_G(a,b) = \max(a,b)$ . We find

$$d_{T_G}(a,b) = \begin{cases} 1, & \text{if } a < b \\ 0, & \text{if } a > b \end{cases}, \quad d_{S_G}(a,b) = \begin{cases} 1, & \text{if } a > b \\ 0, & \text{if } a < b \end{cases}. \quad (33)$$

Both  $T_G$  and  $S_G$  are single-passing, although they have nonvanishing derivatives only whenever  $a \neq b$ . A benefit of the magnitude of the derivative nearly always being 1 is that there will not be any exploding or vanishing gradients caused by multiple repeated applications of the chain rule.

**Example 4.** Filling in Equation 32 representing  $(a \wedge b) \vee (c \wedge \neg a)$  with  $T_G$  and  $S_G$  gives

$$\begin{aligned} \frac{\partial f(a,b,c)}{\partial a} &= \mathbf{1}_{\min(a,b) \geq \min(1-a,c)} \cdot \mathbf{1}_{a \leq b} - \mathbf{1}_{\min(1-a,c) \geq \min(a,b)} \cdot \mathbf{1}_{1-a \leq c} \\ &= \mathbf{1}_{a \geq \min(1-a,c) \wedge a \leq b} - \mathbf{1}_{1-a \geq \min(a,b) \wedge 1-a \leq c} \end{aligned}$$

where the indicator function  $\mathbf{1}_c$  returns 1 if the condition  $c$  is true, and 0 otherwise. This corresponds to the decision tree in Figure 3. The value of  $a$  can be modified to increase the truth of either one of the conjunctions. In order to choose which of the two should be true, it compares  $a$  with  $1 - a$ . If  $1 - a \leq a$ , increasing  $T(a,b)$  will also increase  $S(T(a,b), T(1-a,c))$ . Gradient ascent always finds a global optimum for this formula.

A small perturbation in the truth values of the inputs can flip the derivative around. For instance, if  $a \leq b$  and  $1 - a \leq c$ , then it will increase  $a$  if its value is 0.501 and decrease it if it is 0.499. Furthermore, it can cause gradient ascent to get stuck in local optima even for simple problems. For instance, if  $\varphi = (a \vee b) \wedge (\neg a \vee c)$  and  $a = 0.4, b = 0.2$  and  $c = 0.1$ , gradient ascent increases  $a$  until  $a > 0.5$ , at which point the gradient flips and it decreases  $a$  until  $a < 0.5$ . Experiments with a simple gradient ascent algorithm, have shown us that the algorithm can only find a global optimum in 88.8% of random initializations of  $a, b$  and  $c$ .

### 7.2. Lukasiewicz T-Norm

The Lukasiewicz t-norm is  $T_{LK}(a,b) = \max(a + b - 1, 0)$  and the Lukasiewicz t-conorm is  $S_{LK}(a,b) = \min(a + b, 1)$ . The partial derivatives are:

$$d_{T_{LK}}(a,b) = \begin{cases} 1, & \text{if } a + b \geq 1 \\ 0, & \text{otherwise} \end{cases}, \quad d_{S_{LK}}(a,b) = \begin{cases} 1, & \text{if } a + b \leq 1 \\ 0, & \text{otherwise} \end{cases} \quad (34)$$

These derivatives vanish on as much as half of their domain (Proposition 2). However, like the Gödel t-norm, when there is a gradient, it is large and it will not cause vanishing or exploding gradients.

**Example 5.** Using the Lukasiewicz t-norm and t-conorm in Equation 32 gives rise to the following computation

$$\frac{\partial f(a,b,c)}{\partial a} = \mathbf{1}_{\max(a+b-1,0)+\max(c-a,0) \leq 1} \cdot (\mathbf{1}_{a+b \geq 1} - \mathbf{1}_{c-a \geq 0}).$$

Choosing random values to initialize  $a, b$  and  $c$ , gradient descent is able to find a (global) optimum in about 83.5% of the initializations.

### 7.3. Yager T-Norm

The family of Yager t-norms (Yager, 1980) is  $T_Y(a,b) = \max(1 - ((1-a)^p + (1-b)^p)^{\frac{1}{p}}, 0)$  and the family of Yager t-conorms is  $S_Y(a,b) = \min((a^p + b^p)^{\frac{1}{p}}, 1)$  for  $p \geq 1$ . We plot these for  $p = 2$  in Figure 4. The derivatives

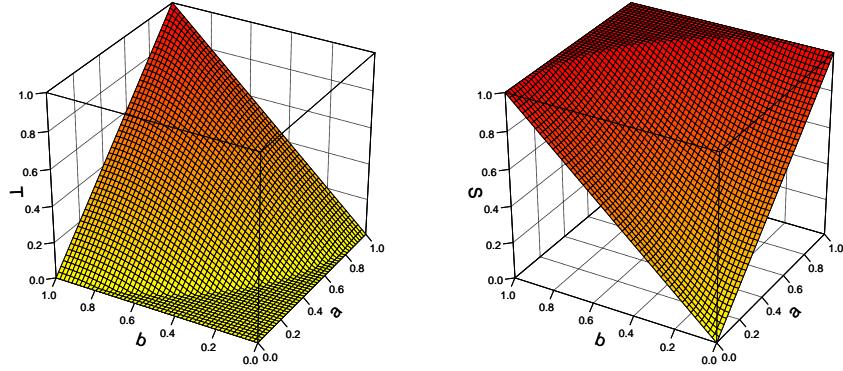


Figure 4: Left: The Yager t-norm. Right: The Yager t-conorm. For both,  $p = 2$ .

are given by

$$d_{T_Y}(a, b) = \begin{cases} ((1-a)^p + (1-b)^p)^{\frac{1}{p}-1} \cdot (1-a)^{p-1} & \text{if } (1-a)^p + (1-b)^p \leq 1, \\ 0 & \text{otherwise.} \end{cases} \quad (35)$$

$$d_{S_Y}(a, b) = \begin{cases} (a^p + b^p)^{\frac{1}{p}-1} \cdot a^{p-1} & \text{if } a^p + b^p \leq 1, \\ 0 & \text{otherwise.} \end{cases} \quad (36)$$

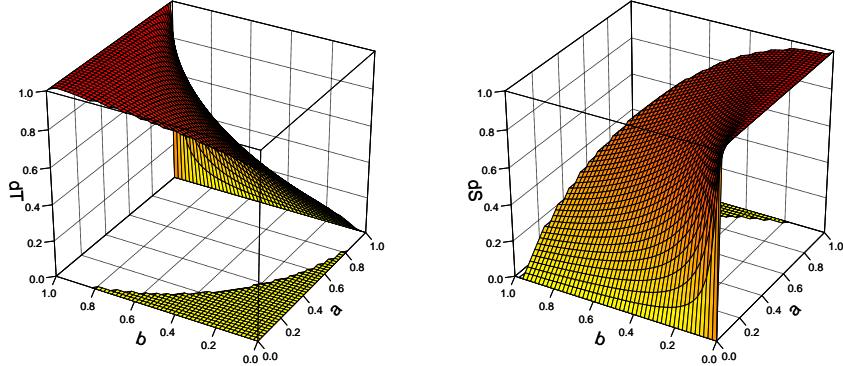


Figure 5: Left: The derivative of the Yager t-norm. Right: The derivative of the Yager s-norm. For both,  $p = 2$ .

We plot these derivatives in Figure 5, showing for each a vanishing derivative on a non-negligible section of the domain. Using the method described in footnote 5 (Section 6.3), Mathematica finds a closed form expression for the fraction of inputs for which the Yager t-norm is nonvanishing as  $\frac{\sqrt{\pi}4^{-1/p}\Gamma(\frac{1}{p})}{p\Gamma(\frac{1}{2}+\frac{1}{p})}$ . Observe that when  $p \neq 1$ , the derivative of  $T_Y$  is undefined at  $a = b = 1$  and the derivative of  $S_Y$  is undefined at  $a = b = 0$ . For  $p > 1$ , the lower of the two truth values has a higher derivative for the t-norm, while for the t-conorm, the higher of the two truth values has a higher derivative. As  $p$  increases,  $T_Y$  and  $S_Y$  will behave more like  $T_G$  and  $S_G$ . Note that when  $p < 1$ , the t-norm will have higher derivatives for higher inputs as the derivative has a singularity at  $\lim_{a \rightarrow 1} = \infty$  ( $b < 1$ ).

#### 7.4. Product T-Norm

The product t- and t-conorms, visualized in Figure 6, are  $T_P(a, b) = a \cdot b$  and  $S_P(a, b) = a + b - a \cdot b$ . Their derivatives are

$$d_{T_P}(a, b) = b, \quad d_{S_P}(a, b) = 1 - b. \quad (37)$$

The derivative of the t-norm only vanishes when  $a = b = 0$  and similarly the gradient of the t-conorm only vanishes when  $a = b = 1$ . The derivative of the t-norm can be interpreted as follows: ‘If we wish to increase  $a \wedge b$ ,  $a$  should be increased in proportion to  $b$ .’ This is not a sensible learning strategy: If both  $a$  and  $b$  are small, in which case the conjunction is most certainly not satisfied, the derivative will be low instead of high.

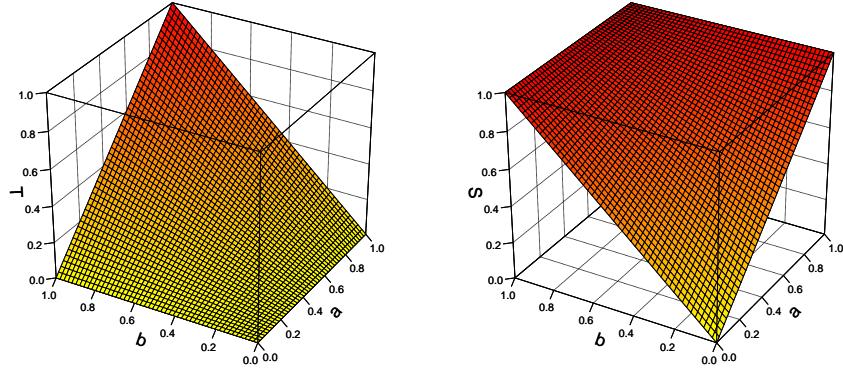


Figure 6: Left: The product t-norm. Right: The product t-conorm.

The derivative of the t-conorm is more intuitive, as it says ‘If we wish to increase  $a \vee b$ ,  $a$  should be increased in proportion to  $1 - b$ '. If  $b$  is not yet true, we definitely want at least  $a$  to be true.

**Example 6.** By using the product t-norm and t-conorm in Equation 32, we get

$$\frac{\partial f(a, b, c)}{\partial a} = (1 - (1 - a) \cdot c) \cdot b - (1 - a \cdot b) \cdot c$$

As explained, increase  $a$  in proportion to  $b$  if it is not true that  $c$  and  $\neg a$  are true, and decrease  $a$  in proportion to  $c$  if it is not true that  $a$  and  $b$  are true.

### 7.5. Summary

The Gödel t-norm and t-conorm are simple and effective, having strong derivatives almost everywhere. However, they can be quite brittle by making very binary choices. The Lukasiewicz t-norm and t-conorm also have strong derivatives, but vanish on half of the domain. The Yager family of t-norms and t-conorms also vanish on a significant part of its domain. The derivative of the t-norm is larger for lower values, which is a sensible learning strategy. This is not the case for the product t-norm, where the derivative is dependent on the other input value. However, the product t-conorm is intuitive, and corresponds to the intuition that if one input is not true, the other one should be.

## 8. Implication

Finally, we consider what functions are suitable for modelling the implication. We will start by discussing the particular challenges associated with the implication operator.

### 8.1. Challenges of the Material Implication

A significant proportion of background knowledge is written as universally quantified implications. Examples of such statements are ‘all humans are mortal’, ‘laptops consist of a screen, a processor and a keyboard’ and ‘only humans wear clothes’. These formulas are of the form  $\forall x \phi(x) \rightarrow \psi(x)$ , where we call  $\phi(x)$  the antecedent and  $\psi(x)$  the consequent.

The implication is used in two well known rules of inference from classical logic. **Modus ponens** inference says that if  $\forall x \phi(x) \rightarrow \psi(x)$  and we know that  $\phi(x)$  is true, then  $\psi(x)$  should also be true. **Modus tollens** inference says that if  $\forall x \phi(x) \rightarrow \psi(x)$  and we know that  $\psi(x)$  is false, then  $\phi(x)$  should also be false, as if  $\phi(x)$  were true,  $\psi(x)$  should also have been.

Unlike sequences of conjunctions where each of the formulas should simply be true, when the learning agent predicts a scene in which an implication is false, the supervisor has multiple choices to correct it. Consider the implication ‘all ravens are black’. There are 4 categories for this formula: *black ravens* (BR), *non-black non-ravens* (NBNR), *black non-ravens* (BNR) and *non-black ravens* (NBR). Assume our agent observes an NBR, which is inconsistent with the background knowledge. There are then four options to consider.

1. **Modus Ponens** (MP): The antecedent is true, so by modus ponens, the consequent is also true. That is, we trust the agent’s observation of a raven and believe it to be an BR.
2. **Modus Tollens** (MT): The consequent is false, so by modus tollens, the antecedent is also false. That is, we trust the agent’s observation of a non-black object and believe that it was not a raven (NBNR).

3. **Distrust:** We believe the agent is wrong (both about observing a raven and about observing a non-black object) and it is probably a black object which is not a raven (BNR).
4. **Exception:** We trust the agent and ignore the fact that its observation goes against the background knowledge that ravens are black.<sup>6</sup> Hence, it has to be a non-black raven (NBR).

The distrust option seems somewhat useless, and the exception option is often going to be correct, but we cannot know when this is just from the agent's observations alone. In such cases, DFL would not be very useful since it would not teach the agent anything new.

We can safely assume that there are far more non-black objects which are not ravens than there are ravens. We can argue that from a statistical perspective, it is most likely that the agent observed an NBNR. This shows the imbalance associated with the implication, which was first noted in van Krieken et al. (2019) for the Reichenbach implication. It is quite similar to the *class imbalance problem* in Machine Learning (Japkowicz and Stephen, 2002) in that the real world has far more 'negative' (or *contrapositive*) examples than positive examples of the background knowledge.

This problem is closely related to the Raven paradox (Hempel, 1945; Vranas, 2004; van Krieken et al., 2019) from the field of confirmation theory which ponders what evidence can confirm a statement like 'ravens are black'. It is usually stated as follows:

- Premise 1: Observing examples of a statement contributes positive evidence towards that statement.
- Premise 2: Evidence for some statement is also evidence for all logically equivalent statements.
- Conclusion: Observing examples of non-black non-ravens is evidence for 'all ravens are black'.

The conclusion follows from the fact that 'non-black objects are non-ravens' is logically equivalent to 'ravens are black'. Although we are considering logical validity instead of confirmation, we note that for DFL a similar thing happens. When we correct the observation of an NBR to a BR, the difference in truth value is equal to when we correct it to NBNR. More precisely, representing 'ravens are black' as  $I(a, b)$ , where, for example,  $I(1, 1)$  corresponds to BR:

$$A(x_1, \dots, I(1, 0), \dots, x_n) - A(x_1, \dots, I(1, 1), \dots, x_n) = A(x_1, \dots, I(1, 0), \dots, x_n) - A(x_1, \dots, I(0, 0), \dots, x_n)$$

as  $I(0, 0) = I(1, 1) = 1$ . Furthermore, when one agent observes a thousand BR's and a single NBR, and another agent observes a thousand NBNR's and a single NBR, their truth value for 'ravens are black' is equal. This seems strange, as the first agent has actually seen many ravens of which only a single exception was not black, while the second only observed many non ravens which were not black, among which a single raven that was not black either. Intuitively, the first agent's beliefs seem to be more in line with the background knowledge. We will now proceed to analyse a number of implication operators in the light of this discussion.

## 8.2. Analyzing the Implication Operators

We define two functions for a fuzzy implication  $I$ :

$$d_{Ic}(a, c) = \frac{\partial I(a, c)}{\partial c} \quad (38)$$

$$d_{I\neg a}(a, c) = -\frac{\partial I(a, c)}{\partial a} = \frac{\partial I(a, c)}{\partial \neg a}. \quad (39)$$

$d_{Ic}$  is the derivative with respect to the consequent and  $d_{I\neg a}$  is the derivative with respect to the *negated antecedent*. We choose to take the derivative with respect to the negated antecedent as it makes it slightly easier to compare the two: all fuzzy implications are monotonically decreasing with respect to the antecedent.

**Definition 19.** A fuzzy implication  $I$  is called **contrapositive differentiable symmetric** if  $d_{Ic}(a, c) = d_{I\neg a}(1 - c, 1 - a)$  for all  $a, c \in [0, 1]$ .

A consequence of contrapositive differentiable symmetry is that if  $c = 1 - a$ , then the derivatives are equal since  $d_{Ic}(a, c) = d_{I\neg a}(1 - c, 1 - a) = d_{I\neg a}(1 - (1 - a), c) = d_{I\neg a}(a, c)$ . This could be seen as the 'distrust' option in which it increases the consequent and negated antecedent equally.

**Proposition 5.** If a fuzzy implication  $I$  is  $N_C$ -contrapositive symmetric, where  $N_C$  is the classical negation, it is also contrapositive differentiable symmetric.

---

<sup>6</sup>This option is not completely ludicrous as white ravens do in fact exist. However, they are rare.

*Proof.* Say we have an implication  $I$  that is  $N_C$ -contrapositive symmetric. We find that  $d_{Ic}(a, c) = \frac{\partial I(a, c)}{\partial c}$  and  $d_{I\neg a}(1 - c, 1 - a) = -\frac{\partial I(1 - c, 1 - a)}{\partial 1 - c}$ . Because  $I$  is  $N_C$ -contrapositive symmetric,  $I(1 - c, 1 - a) = I(a, c)$ . Thus,  $d_{I\neg a}(1 - c, 1 - a) = -\frac{\partial I(a, c)}{\partial 1 - c} = \frac{\partial I(a, c)}{\partial c} = d_{Ic}(a, c)$ .  $\square$

In particular, by this proposition all S-implications<sup>7</sup> are contrapositive differentiable symmetric. This says that there is no difference in how the implication handles the derivatives with respect to the consequent and antecedent.

**Proposition 6.** If an implication  $I$  is left-neutral, then  $d_{Ic}(1, c) = 1$ . If, in addition,  $I$  is contrapositive differentiable symmetric, then  $d_{I\neg a}(a, 0) = 1$ .

*Proof.* First, assume  $I$  is left-neutral. Then for all  $c \in [0, 1]$ ,  $I(1, c) = c$ . Taking the derivative with respect to  $c$ , it turns out that  $d_{Ic}(1, c) = 1$ . Next, assume  $I$  is contrapositive differentiable symmetric. Then,  $d_{Ic}(1, c) = d_{I\neg a}(1 - c, 1 - 1) = d_{I\neg a}(1 - c, 0) = 1$ . As  $1 - c \in [0, 1]$ ,  $d_{I\neg a}(a, 0) = 1$ .  $\square$

All S-implications and R-implications are left-neutral, but only S-implications are also contrapositive differentiable symmetric. The derivatives of R-implications vanish when  $a \leq c$ , that is, on no less than half of the domain. Note that the plots in this section are rotated so that the smallest value is in the front to help understand the shape of the functions. In particular, plots of the derivatives of the implications are rotated 180 degrees compared to the implications themselves.

### 8.3. Gödel-based Implications

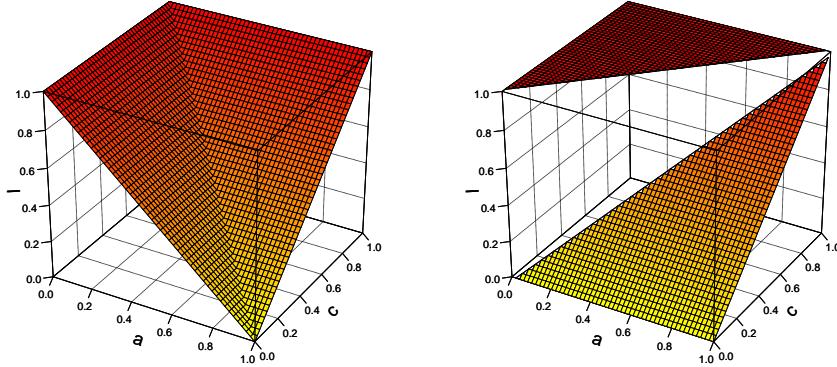


Figure 7: Left: The Kleene Dienes implication. Right: The Gödel implication.

Implications based on the Gödel t-norm make discrete choices and are single-passing. As  $I_{KD}(a, c) = \max(1 - a, c)$ , the derivatives are

$$d_{I_{KD}c}(a, c) = \begin{cases} 1, & \text{if } 1 - a < c \\ 0, & \text{if } 1 - a > c \end{cases}, \quad d_{I_{KD}\neg a}(a, b) = \begin{cases} 1, & \text{if } 1 - a > c \\ 0, & \text{if } 1 - a < c \end{cases}. \quad (40)$$

Or, simply put, if we are more confident in the truth of the consequent than in the truth of the negated antecedent, increase the truth of the consequent. Otherwise, decrease the truth of the antecedent. This decision can be somewhat arbitrary and does not take into account the imbalance of modus ponens and modus tollens.

The Gödel implication is a simple R-implication:  $I_G(a, c) = \begin{cases} 1, & \text{if } a \leq c \\ c, & \text{otherwise} \end{cases}$ .

Its derivatives are:

$$d_{I_Gc}(a, c) = \begin{cases} 1, & \text{if } a > c \\ 0, & \text{otherwise} \end{cases}, \quad d_{I_G\neg a}(a, b) = 0. \quad (41)$$

These two implications are shown in Figure 7. The Gödel implication increases the consequent whenever  $a > c$ , and the antecedent is never changed. This makes it a poorly performing implication in practice. For example, consider  $a = 0.1$  and  $c = 0$ . Then the Gödel implication increases the consequent, even if the agent is fairly certain that neither is true. Furthermore, as the derivative with respect to the negated antecedent is always 0, it can never choose the modus tollens correction, which, as we argued, is actually often the best choice.

<sup>7</sup>To be more precise, all (S, N)-implications formed from a strict and strong negation  $N$ , like  $N_C$ .

#### 8.4. Lukasiewicz and Yager-based Implications

The Lukasiewicz implication is both an S- and an R-implication. It is given by  $I_{LK}(a, c) = \min(1 - a + c, 1)$  and has the simple derivatives

$$d_{I_{LK}c}(a, c) = d_{I_{LK}\neg a}(a, c) = \begin{cases} 1, & \text{if } a > c \\ 0, & \text{otherwise.} \end{cases} \quad (42)$$

Whenever the implication is not satisfied because the antecedent is higher than the consequent, simply increase the negated antecedent and the consequent until it is lower. This could be seen as the ‘distrust’ choice as both observations of the agent are equally corrected, and so does not take into account the imbalance between modus ponens and modus tollens cases. The derivatives of the Gödel implication  $I_G$  are equal to those of  $I_{LK}$  except that  $I_G$  always has a zero derivative for the negated antecedent.

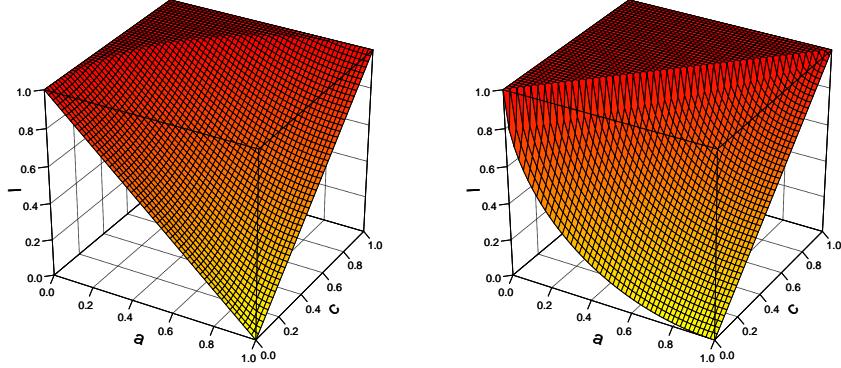


Figure 8: Left: The Yager S-implication. Right: The Yager R-implication. For both,  $p = 2$ .

The Yager S-implication is given as

$$I_Y(a, c) = \min\left(\left((1-a)^p + c^p\right)^{\frac{1}{p}}, 1\right), \quad p \geq 1. \quad (43)$$

We plot  $I_Y$  for  $p = 2$  in Figure 8. For  $p = 1$ ,  $I_Y$  reduces to  $I_{LK}$ , for  $p = 0$  to  $I_{DP}$ , and for  $p = \infty$  to  $I_{KD}$ . Since it is an S-implication, it is contrapositive symmetric with respect to  $N_C$ , left-neutral and it satisfies the exchange principle. For  $p \leq 1$ , it satisfies the identity principle. The derivatives are computed as

$$d_{I_Yc}(a, c) = \begin{cases} \left((1-a)^p + c^p\right)^{\frac{1}{p}-1} \cdot c^{p-1}, & \text{if } (1-a)^p + c^p \leq 1, \\ 0, & \text{otherwise} \end{cases} \quad (44)$$

$$d_{I_Y\neg a}(a, c) = \begin{cases} \left((1-a)^p + c^p\right)^{\frac{1}{p}-1} \cdot (1-a)^{p-1}, & \text{if } (1-a)^p + c^p \leq 1, \\ 0, & \text{otherwise.} \end{cases} \quad (45)$$

We plot these derivatives for  $p = 2$  in Figure 9. For all  $p$ ,  $\lim_{c \rightarrow 0} d_{I_Yc}(1, c) = 1$ . Furthermore, for  $p > 1$ ,  $\lim_{a \rightarrow 1} d_{I_Yc}(a, 0) = 0$  and for  $p < 1$ ,  $\lim_{a \rightarrow 1} d_{I_Yc}(a, 0) = \infty$ . For  $p > 1$ ,  $I_Y$  can be understood as an increasingly less smooth version of the Kleene-Dienes implication  $I_{KD}$ . Lastly, this derivative, like those for  $T_Y$  and  $S_Y$  (Section 7.3), is nonvanishing for only a fraction of  $\frac{\sqrt{\pi}4^{-1/p}\Gamma(\frac{1}{p})}{p\Gamma(\frac{1}{2}+\frac{1}{p})}$  of the input space.

The Yager R-implication is found (see Appendix C.4 for details) as

$$I_{TY}(a, c) = \begin{cases} 1, & \text{if } a \leq c \\ 1 - \left((1-c)^p - (1-a)^p\right)^{\frac{1}{p}}, & \text{otherwise.} \end{cases} \quad (46)$$

We plot  $I_{TY}$  for  $p = 2$  in Figure 8. As expected,  $p = 1$  reduces to  $I_{LK}$ ,  $p = 0$  reduces to  $I_{WB}$  and  $p = \infty$  reduces to  $I_G$ . It is contrapositive symmetric only for  $p = 1$ . The derivatives of this implication are

$$d_{I_{TY}c}(a, c) = \begin{cases} \left((1-c)^p - (1-a)^p\right)^{\frac{1}{p}-1} \cdot (1-c)^{p-1}, & \text{if } a > c \\ 0, & \text{otherwise,} \end{cases} \quad (47)$$

$$d_{I_{TY}\neg a}(a, c) = \begin{cases} \left((1-c)^p - (1-a)^p\right)^{\frac{1}{p}-1} \cdot (1-a)^{p-1}, & \text{if } a > c \\ 0, & \text{otherwise.} \end{cases} \quad (48)$$

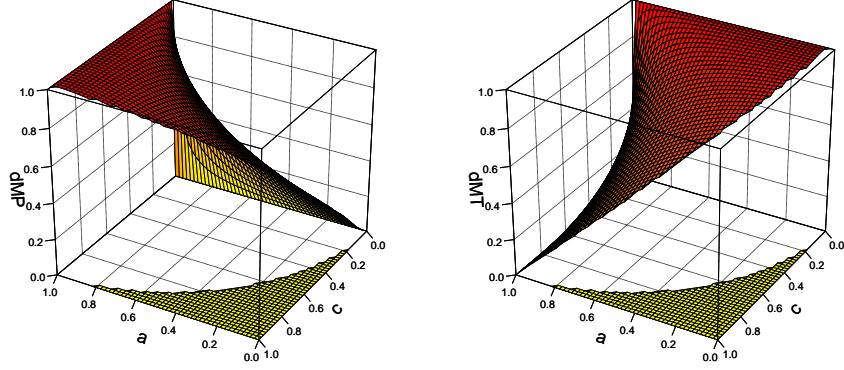


Figure 9: Plots of the derivatives of the Yager S-implication for  $p = 2$ .

We plot these in Figure 10. Note that if  $p > 1$ , for all  $c < 1$  it holds that  $\lim_{a \downarrow c} d_{I_{TY}c}(a, c) = \lim_{a \downarrow c} d_{I_{TY}\neg a}(a, c) = \infty$  as when  $a$  approaches  $c$  from above,  $(1 - c)^p - (1 - a)^p$  approaches 0, giving a singularity as  $0^{\frac{1}{p}-1}$  is undefined. This collection of singularities makes the training unstable in practice.

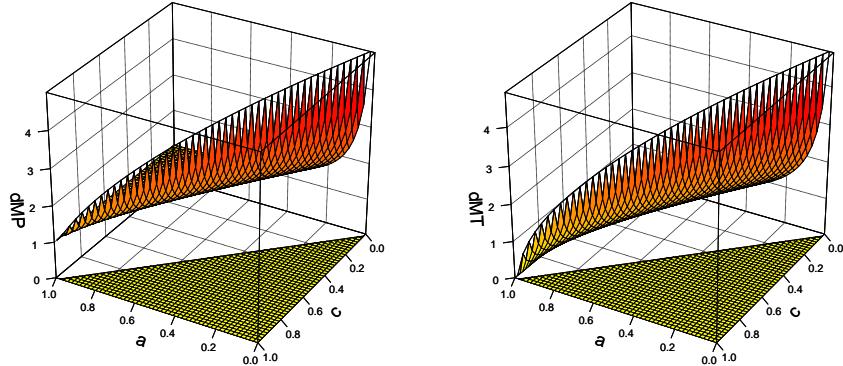


Figure 10: Plots of the derivatives of the Yager R-implication for  $p = 2$ .

### 8.5. Product-based Implications

The product S-implication, also known as the Reichenbach implication, is given by  $I_{RC}(a, c) = 1 - a + a \cdot c$ . We plot it in Figure 11. Its derivatives are given by:

$$d_{I_{RC}c}(a, c) = a, \quad d_{I_{RC}\neg a}(a, c) = 1 - c. \quad (49)$$

These derivatives closely follow the modus ponens and modus tollens rules. When the antecedent is high, increase the consequent, and when the consequent is low, decrease the antecedent. However, around  $(1 - a) = c$ , the derivative is equal and the ‘distrust’ option is chosen. This can result in counter-intuitive behaviour. For example, if the agent predicts 0.6 for **raven** and 0.5 for **black** and we use gradient descent until we find a maximum, we could end up at 0.3 for **raven** and 1 for **black**. We would end up increasing our confidence in **black** as **raven** was high. However, because of additional modus tollens reasoning, **raven** is barely true.

Furthermore, if the agent most of the time predicts values around  $a = 0$ ,  $c = 0$  as a result of the modus tollens case being the most common, then a majority of the gradient decreases the antecedent as  $d_{I_{RC}\neg a}(0, 0) = 1$ . We identify two methods that counteract this behavior. We introduce the second method in Section 8.5.1.

The first method for counteracting the ‘corner’ behavior notes that different aggregators change how the derivatives of the implications behave. In particular, we compare the log-product aggregator and the RMSE aggregator and how they combine with the Reichenbach implication. By using the chain rule and Equation 21,

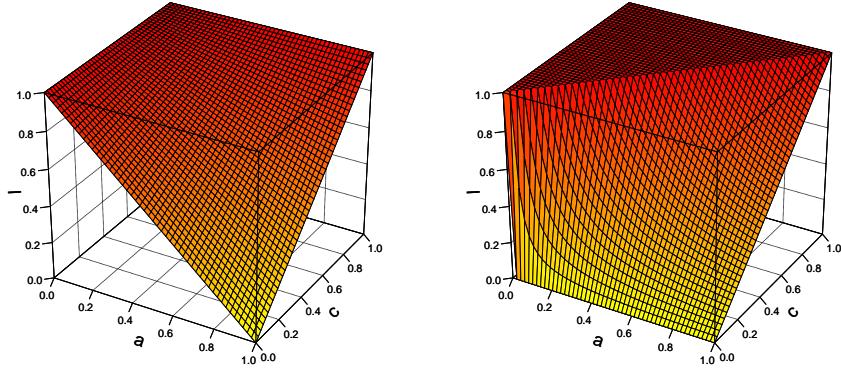


Figure 11: Left: The Reichenbach implication. Right: The Goguen implication.

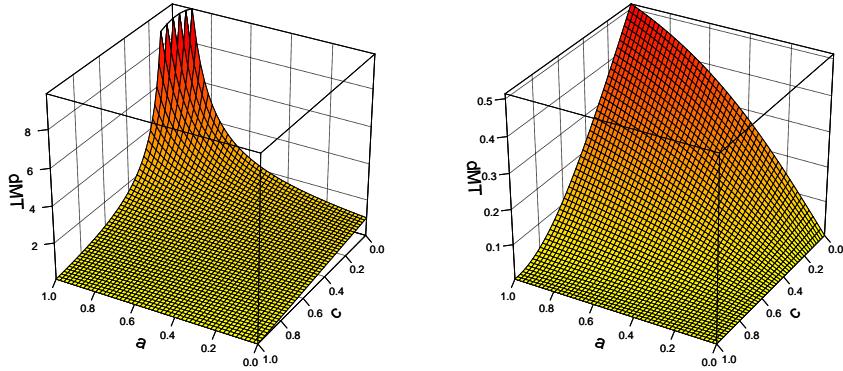


Figure 12: Left: The antecedent derivative of the Reichenbach implication with the log-product aggregator. Right: The antecedent derivative of the Reichenbach implication with the RMSE aggregator.

we find that the derivatives with respect to the negated antecedent using those aggregators are:

$$\frac{\partial \log \circ A_P(I_{RC}(a_1, c_1), \dots, I_{RC}(a_n, c_n))}{\partial 1 - a_i} = \frac{1 - c_i}{1 - a_i + a_i \cdot c_i} = \frac{-c}{a \rightarrow c}, \quad (50)$$

$$\frac{\partial A_{RMSE}(I_{RC}(a_1, c_1), \dots, I_{RC}(a_n, c_n))}{\partial 1 - a_i} = \frac{(1 - c_i)(a_i - a_i \cdot c_i)}{\sqrt{n \sum_{j=1}^n (a_j - a_j \cdot c_j)^2}} = \frac{-c_i(\neg(a_i \rightarrow c_i))}{\sqrt{n \sum_{j=1}^n (\neg(a_j \rightarrow c_j))^2}}. \quad (51)$$

We plot these functions with respect to  $a_i$  and  $c_i$  in Figure 12. Note that for the RMSE aggregator, the truth values of other inputs  $a_j, c_j, i \neq j$  can change the shape of the function. We arbitrarily choose  $n = 2$  and  $a_1, c_1$  so that  $(a_1 - a_1 \cdot c_1)^2 = 0.9$ . Note also that the derivative with respect to the negated antecedent using the RMSE aggregator is 0 in  $a_i = 0, c_i = 0$  as then  $a_i - a_i \cdot c_i = 0$ , and using the log-product aggregator, the derivative is 1. By differentiable contrapositive symmetry, the consequent derivative is 0 when using both aggregators. This shows that when using the RMSE aggregator, the derivatives will vanish at the two corners, i.e.  $a = 0, c = 0$  and  $a = 1, c = 1$ . On the other hand, when using the log-product aggregator, one of antecedent and consequent will have a gradient.

The R-implication of the product t-norm is called the Goguen implication and is given by

$$I_{GG}(a, c) = \begin{cases} 1, & \text{if } a \leq c \\ \frac{c}{a}, & \text{otherwise.} \end{cases} \quad (52)$$

We plot this implication in Figure 11. The derivatives of  $I_{GG}$  are

$$d_{I_{GG}c}(a, c) = \begin{cases} 0, & \text{if } a \leq c \\ \frac{1}{a}, & \text{otherwise} \end{cases}, \quad d_{I_{GG}\neg a}(a, c) = \begin{cases} 0, & \text{if } a \leq c \\ \frac{c}{a^2}, & \text{otherwise.} \end{cases}. \quad (53)$$

We plot these in Figure 13. This derivative is not very useful. First of all, both the modus ponens and modus tollens derivatives increase with  $\neg a$ . This is opposite of the modus ponens rule as when the antecedent

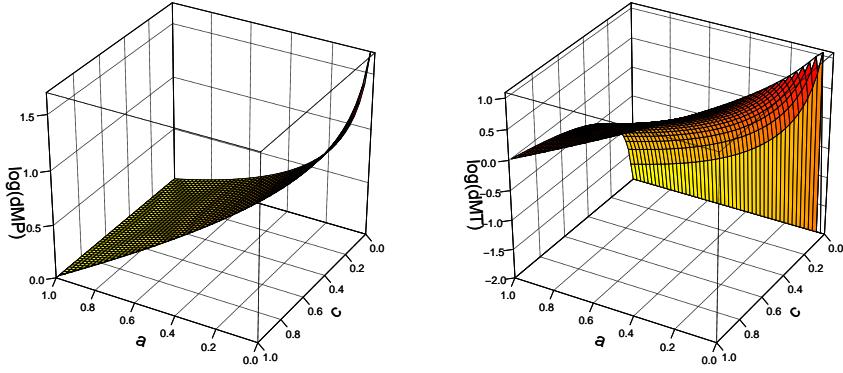


Figure 13: The derivatives of the Goguen implication. Note that we plot these in log scale.

is *low*, it increases the consequent most. For example, if `raven` is 0.1 and `black` is 0, then the derivative with respect to `black` is 10, because of the singularity when  $a$  approaches 0.

#### 8.5.1. Sigmoidal Implications

We introduce a new class of fuzzy implications formed by transforming other fuzzy implications using the sigmoid function and translating it so that the boundary conditions still hold. The derivation, along with several proofs of properties, can be found in Appendix C.2.

**Definition 20.** If  $I$  is a fuzzy implication, then the  $I$ -sigmoidal implication  $\sigma_I$  is given for some  $s > 0$  and  $b_0 \in \mathbb{R}$  as

$$\sigma_I(a, c) = \frac{1 + e^{-s(1+b_0)}}{e^{-b_0 s} - e^{-s(1+b_0)}} \cdot ((1 + e^{-b_0 s}) \cdot \sigma(s \cdot (I(a, c) + b_0)) - 1) \quad (54)$$

Here  $b_0$  is a parameter that controls the position of the sigmoidal curve and  $s$  controls the ‘spread’ of the curve.  $\sigma_I$  is the function  $\sigma(s \cdot (I(a, c) + b_0))$  linearly transformed so that its codomain is the closed interval  $[0, 1]$ . For the common value of  $b_0 = -\frac{1}{2}$ , a simpler form exists:

$$\sigma_I(a, c) = \frac{1}{e^{\frac{s}{2}} - 1} \cdot \left( (1 + e^{\frac{s}{2}}) \cdot \sigma \left( s \cdot \left( I(a, c) - \frac{1}{2} \right) \right) - 1 \right). \quad (55)$$

Next, we give the derivative of  $\sigma_I$ . Substituting  $d = \frac{1+e^{-s \cdot (1+b_0)}}{e^{-s \cdot b_0} - e^{-s \cdot (1+b_0)}}$  and  $h = (1 + e^{-s \cdot b_0})$ , we find

$$\frac{\partial \sigma_I(a, c)}{\partial I(a, c)} = d \cdot h \cdot s \cdot \sigma(s \cdot (I(a, c) + b_0)) \cdot (1 - \sigma(s \cdot (I(a, c) + b_0))). \quad (56)$$

The derivative keeps the properties of the original function but smoothes the gradient for higher values of  $s$ . As the derivative of the sigmoid function (that is,  $\sigma(x) \cdot (1 - \sigma(x))$ ) cannot be zero, this derivative vanishes only when  $\frac{\partial I(a, c)}{\partial a} = 0$  or  $\frac{\partial I(a, c)}{\partial c} = 0$ .

We plot the derivatives for the Reichenbach-sigmoidal implication  $\sigma_{I_{RC}}$  in Figure 15. As expected by Proposition 14, it is differentiable contrapositive symmetric. Compared to the derivatives of the Reichenbach implication it has a small gradient in all corners. When using the log-product aggregator, the derivative of the antecedent with respect to the total valuation is divided by the truth of the implication. In Figure 14 we compare the consequent derivative of the normal Reichenbach implication with the Reichenbach-sigmoidal implication when using the log function. Clearly, for both there is a singularity at  $a = 1, c = 0$ , as then the implication is 0 and so the derivative of the log function becomes infinite. A significant difference is that the sigmoidal variant is less ‘flat’ than the normal Reichenbach implication. This can be useful, as this means there is a larger gradient for values of  $c$  that make the implication less true. In particular, the gradient at the modus ponens case ( $a = 1, c = 1$ ) and the modus tollens case ( $a = 0, c = 0$ ) are far smaller, which could help balancing the effective total gradient by solving the ‘corner’ problem of the Reichenbach implication we brought up in Section 8.5. These derivatives are smaller for for higher values of  $s$ .

In Figure 16 we plot the Reichenbach-sigmoidal implication for different values of the hyperparameters  $b_0$  and  $s$ . Comparing 16a and 16b we see that larger values of  $b_0$  move the sigmoidal shape so that its center is at lower input values. Note that for  $s = 0.01$  in Figure 16c, the plotted function is indiscernible from the plot of the Reichenbach implication in Figure 11 as the interval on which the sigmoid acts is extremely small and

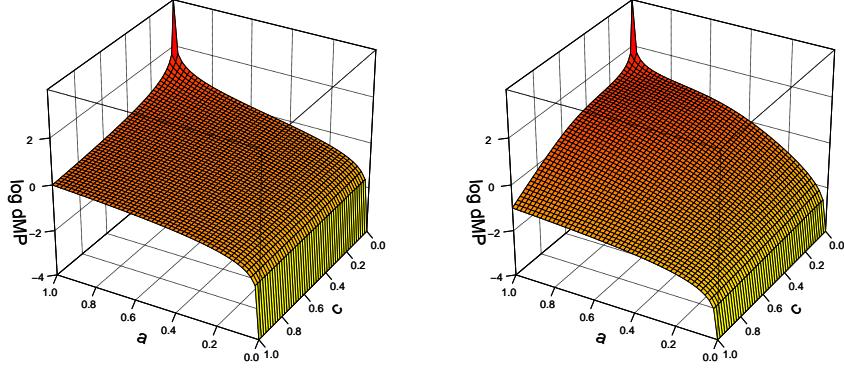


Figure 14: The modus ponens derivatives of the log-Reichenbach and log-Reichenbach-sigmoidal (with  $s=9$ ) implications. The figure is plotted in log scale.

the sigmoidal transformation is almost linear. For very high values of  $s$  like in 16d we see that the ‘S’ shape is much thinner, and a larger part of the domain has a low derivative.

In Figure 17 we show what part of the sigmoid function is utilized for different values of  $b_0$  and  $s$ : The restriction<sup>8</sup> between a pair of vertical lines correspond to some value of  $b_0$  and  $s$ . Furthermore, because of left-neutrality and contraposition,  $I_{RC}(x, 0)$  and  $I_{RC}(1, x)$  are both equal to  $x$ ,  $\sigma_{I_{RC}}(x, 0)$  and  $\sigma_{I_{RC}}(1, x)$  are linear transformations of this restriction. For example, the restriction of the sigmoid between the orange bars representing  $b_0 = -0.2$  and  $s = 9$  can be seen on the line  $\sigma_{RC}(a, 0)$  plotted in Figure 16b.

### 8.6. Summary

We analyzed several fuzzy implications from a theoretical perspective, while keeping the challenges caused by the material implication in mind. As a result of this analysis, we find that popular R-implications, in particular the Gödel implication, the Yager R-implication and the Goguen implication, will not work well in a differentiable setting. The other analyzed implications seem to have more intuitive derivatives, but may have other practical issues like non-smoothness.

## 9. Experiments

To get an idea of the practical behavior of these implications, and other operators, we now perform a series of simple experiments to analyze them in practice. In this section, we discuss experiments using the MNIST dataset of handwritten digits (LeCun and Cortes, 2010) to investigate the behavior of different fuzzy operators introduced in this paper.

### 9.1. Measures

To investigate the performance of the different configurations of DFL, we first introduce several useful metrics. These give us insight into how different operators behave.

**Definition 21.** The **consequent magnitude**  $|\text{cons}|_\varphi$  and the **antecedent magnitude**  $|\text{ant}|_\varphi$  for a formula  $\varphi = \forall x_1, \dots, x_m \phi \rightarrow \psi$  is defined as the sum of the partial derivatives of the consequent and antecedent with respect to the DFL loss:

$$|\text{cons}|_\varphi = \sum_{\mu \in M_\varphi} \frac{\partial e_\theta(\varphi, \{\})}{\partial e_\theta(\psi, \mu)}, \quad |\text{ant}|_\varphi = - \sum_{\mu \in M_\varphi} \frac{\partial e_\theta(\varphi, \{\})}{\partial e_\theta(\phi, \mu)} \quad (57)$$

where  $M_\varphi$  is the set of instances of the universally quantified formula  $\varphi$ .<sup>9</sup>

<sup>8</sup>By  $f|_A$  we refer to a domain restriction on a function  $f : D \rightarrow E$  i.e., the function  $f$  except only defined on the subdomain  $A \subset D$ .

<sup>9</sup>This is not the sum of consequent derivatives as defined by Equation 39 for all instances  $(\sum_{\mu \in M_\varphi} d_{Ic}(e_\theta(\phi, \mu), e_\theta(\psi, \mu)))$  as the choice of aggregator matters.

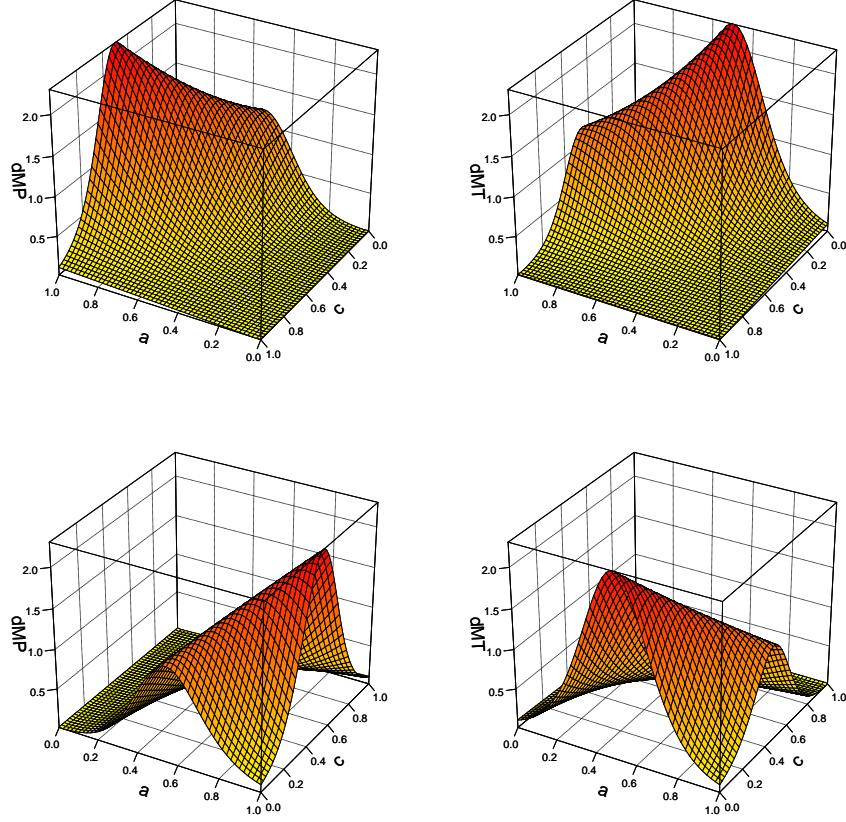


Figure 15: The derivatives of the Reichenbach-sigmoidal implication for  $s = 9$ , along with the same plots rotated 180 degrees to get additional insights.

**Definition 22.** The **consequent ratio**  $\text{cons}\%$  is the sum of consequent magnitudes over all formulas divided by the sum of consequent and antecedent magnitudes over all formulas:

$$\text{cons}\% = \frac{\sum_{\varphi \in \mathcal{K}} |\text{cons}|_\varphi}{\sum_{\varphi \in \mathcal{K}} |\text{cons}|_\varphi + \sum_{\varphi \in \mathcal{K}} |\text{ant}|_\varphi}. \quad (58)$$

**Definition 23.** Given a **labeling function**  $l$  that returns the truth value of a formula given an instance  $\mu$  according to the data, the **consequent and antecedent correctly updated magnitudes** are the sum of partial derivatives for which the consequent or the negated antecedent is true:

$$cu_{\text{cons}\varphi} = \sum_{\mu \in M_\varphi} l(\psi, \mu) \cdot \frac{\partial e_\theta(\varphi, \{\})}{\partial e_\theta(\psi, \mu)}, \quad cu_{\text{ant}\varphi} = - \sum_{\mu \in M_\varphi} l(\neg\phi, \mu) \cdot \frac{\partial e_\theta(\varphi, \{\})}{\partial e_\theta(\phi, \mu)}. \quad (59)$$

That is, if the consequent is true in the data, we measure the magnitude of the derivative with respect to the consequent. To evaluate these quantities, we define ratios similar to a precision metric:

**Definition 24.** The **correctly updated ratio** for consequent and antecedent is defined as

$$cu_{\text{cons}}\% = \frac{\sum_{\varphi \in \mathcal{K}} cu_{\text{cons}\varphi}}{\sum_{\varphi \in \mathcal{K}} |\text{cons}|_\varphi}, \quad cu_{\text{ant}}\% = \frac{\sum_{\varphi \in \mathcal{K}} cu_{\text{ant}\varphi}}{\sum_{\varphi \in \mathcal{K}} |\text{ant}|_\varphi}. \quad (60)$$

These quantify what fraction of the updates are going in the right direction. When these ratios approach 1, DFL will increase the truth value of the consequent or negated antecedent correctly.<sup>10</sup> When it is less, we are increasing truth values of subformulas that are wrong, thus ideally, we want these measures to be high.

## 9.2. Formulas

We use a knowledge base  $\mathcal{K}$  of universally quantified logic formulas. There is a predicate for each digit, that is **zero**, **one**, ..., **eight** and **nine**. For example, **zero**( $x$ ) is true whenever  $x$  is a handwritten digit labeled with 0.

<sup>10</sup>It can still change the truth value of a ground atom wrongly if  $\phi$  or  $\psi$  are not atomic formulas.

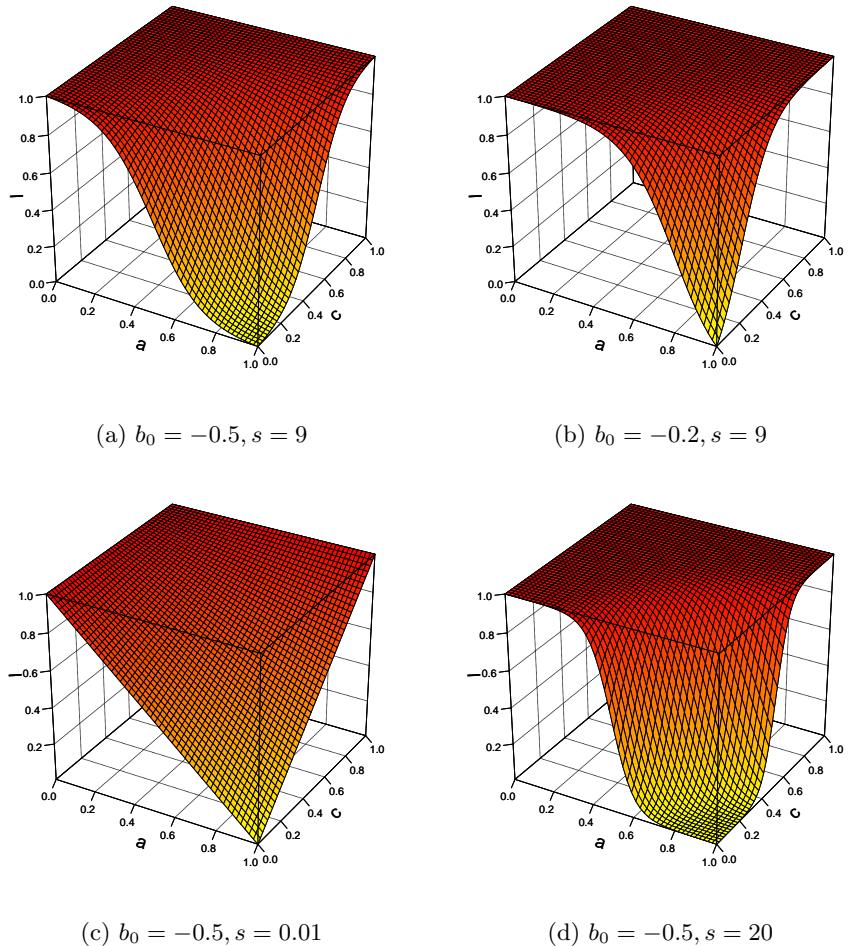


Figure 16: The Reichenbach-sigmoidal implication for different values of  $b_0$  and  $s$ .

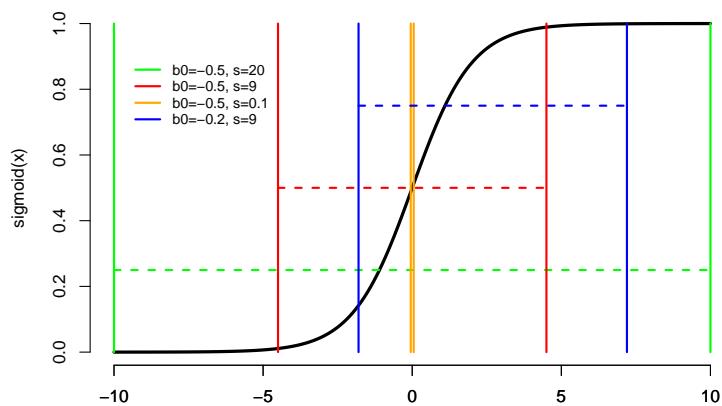


Figure 17: The sigmoid function  $\sigma(x)$  is plotted in black alongside indications of the domain of sigmoidal operators under various values of  $b_0$  and  $s$ . The values of  $x$  between the vertical bars with the same color (corresponding to some parameter values) is the domain used.

Secondly, there is the binary predicate `same` that is true whenever both its arguments are the same digit. We next describe the formulas we use. We also note what the values of  $cu_{\text{cons}}\%$  and  $cu_{\text{ant}}\%$  roughly are if we were to pick at random.

1.  $\forall x, y \text{ zero}(x) \wedge \text{zero}(y) \rightarrow \text{same}(x, y), \dots, \forall x, y \text{ nine}(x) \wedge \text{nine}(y) \rightarrow \text{same}(x, y)$ . If both  $x$  and  $y$  are handwritten zeros, for example, then they represent the same digit. For this formula,  $cu_{\text{cons}}\% \geq \frac{1}{10}$  as it is the distribution of  $\text{same}(x, y)$ <sup>11</sup> and  $cu_{\text{ant}}\% \leq \frac{99}{100}$  as it is 1 minus the probability that both  $x$  and  $y$  are zero. The modus ponens case is true in more than  $\frac{1}{100}$  cases, the modus tollens case in less than  $\frac{9}{10}$  cases and the ‘distrust’ option in more than  $\frac{9}{100}$  cases.
2.  $\forall x, y \text{ zero}(x) \wedge \text{same}(x, y) \rightarrow \text{zero}(y), \dots, \forall x, y \text{ nine}(x) \wedge \text{same}(x, y) \rightarrow \text{nine}(y)$ . If  $x$  and  $y$  represent the same digit and one of them represents zero, then the other one does as well. For this formula,  $cu_{\text{cons}}\% = \frac{1}{10}$  as it is the probability that a digit represents zero and  $cu_{\text{ant}}\% \leq \frac{99}{100}$ . The modus ponens cases is true in more than  $\frac{1}{100}$  cases, the modus tollens in  $\frac{9}{10}$  cases and the ‘distrust’ option in  $\frac{9}{100}$  cases.
3.  $\forall x, y \text{ same}(x, y) \rightarrow \text{same}(y, x)$ . This formula encodes the symmetry of the `same` predicate. As this is a bi-implication,  $cu_{\text{cons}}\% \geq \frac{1}{10}$  and  $cu_{\text{ant}}\% \leq \frac{9}{10}$ . The ‘distrust’ option is not possible in this formula.

From this, we can see that a set of operators is better than random guessing for the consequent updates if  $cu_{\text{cons}}\% > 0.1$ . It is more difficult to say what the value of  $cu_{\text{ant}}\%$  should be to be as good as random guessing, as the probabilities are upper bounded with the lowest bound at 0.9. We can only say that we know a set of operators to be better than random if  $cu_{\text{ant}}\% > 0.99$ .

### 9.3. Experimental Setup

We split the MNIST dataset so that 1% of it is labeled and 99% is unlabeled. We use two models.<sup>12</sup> Given a handwritten digit  $\mathbf{x}$  labeled with digit  $y$ , the first model  $p_{\theta}(y|\mathbf{x})$  computes the distribution over the 10 possible labels. We use 2 convolutional layers with max pooling, the first with 10 and the second with 20 filters. Then follows two fully connected hidden layers with 320 and 50 nodes and a softmax output layer. The probability that  $\text{same}(\mathbf{x}_1, \mathbf{x}_2)$  for two handwritten digits  $\mathbf{x}_1$  and  $\mathbf{x}_2$  holds is modeled by  $p_{\theta}(\text{same}|\mathbf{x}_1, \mathbf{x}_2)$ . This takes the 50-dimensional embeddings of  $\mathbf{x}_1$  and  $\mathbf{x}_2$  of the fully connected hidden layer  $e_{\mathbf{x}_1}$  and  $e_{\mathbf{x}_2}$ . These are used in a network architecture called a Neural Tensor Network (Socher et al., 2013):

$$p_{\theta}(\text{same}|\mathbf{x}_1, \mathbf{x}_2) = \sigma \left( u^T \tanh \left( e_{\mathbf{x}_1}^T W^{[1:k]} e_{\mathbf{x}_2} + V \begin{bmatrix} e_{\mathbf{x}_1} \\ e_{\mathbf{x}_2} \end{bmatrix} + b \right) \right). \quad (61)$$

$W^{[1:k]} \in \mathbb{R}^{d \times d \times k}$  is used for the bilinear tensor product,  $V \in \mathbb{R}^{k \times 2d}$  is used for the concatenated embeddings and  $b \in \mathbb{R}^k$  is used as a bias vector. We use  $k = 50$  for the size of the hidden layer.  $u \in \mathbb{R}^k$  is used to compute the output logit, which goes through the sigmoid function  $\sigma$  to get the probability.

The loss function we use is split up in three parts:

$$\mathcal{L}(\mathcal{D}_l, \mathcal{D}_u, \theta) = - \sum_{\mathbf{x}, y \in \mathcal{D}_l} \log p_{\theta}(y|\mathbf{x}) + w_{DFL} \cdot \mathcal{L}_{DFL}(\theta; \mathcal{D}_u, \mathcal{K}) \quad (62)$$

$$- \sum_{\substack{\mathbf{x}_1, y_1, \mathbf{x}_2, y_2 \\ \in \mathcal{D}_l \times \mathcal{D}_l}} \log p_{\theta}(\text{same} = I[y_1 = y_2]|\mathbf{x}_1, \mathbf{x}_2) \quad (63)$$

The first term is the supervised cross entropy loss with a batch size of 64. The second is the DFL loss which is weighted by the *DFL weight*  $w_{DFL}$ . The third is the supervised binary cross entropy loss used to learn recognize  $\text{same}(x, y)$ .<sup>13</sup> As there are far more negative examples than positive examples, we undersample the negative examples.

### 9.4. Results

We analyze the results for many different combinations of hyperparameters, in particular by choosing different operators for aggregation, conjunction and implication. It should be noted that the purely supervised baseline has a test accuracy of  $95.0\% \pm 0.001$  (3 runs). Semi-supervised methods that do not improve upon this baseline are useless.

We report the accuracy of recognizing digits in the test set. We do learning for at most 100.000 iterations (or until convergence). We also report the consequent ratio  $cons\%$  and the consequent and antecedent correctly updated ratios  $cu_{\text{cons}}\%$  and  $cu_{\text{ant}}\%$ . We can compute these values during the backpropagation of the DFL loss on the ‘unlabeled’ dataset. Because it is a split of a labeled dataset, we can access the labels for evaluation.

<sup>11</sup>It is slightly more than  $\frac{1}{10}$  because we are using a minibatch of examples. Therefore, the reflexive pairs (i.e.,  $\text{same}(x, x)$ ) are common.

<sup>12</sup>Code is available at <https://github.com/HEmile/differentiable-reasoning-thesis/tree/master/mnist%20experiments>.

<sup>13</sup>It is possible to not use this loss term and learn the `same` predicate using just the formulas, although this is more challenging and only works with a good set of fuzzy operators.

#### 9.4.1. Symmetric Configurations

First, we consider several **symmetric** configurations. A symmetric configuration is one for which the conjunction is a t-norm  $T$ , disjunction the dual t-conorm of  $T$ , aggregation the extended t-norm  $A_T$  (Equation 2) and the implication the S-implication based on the t-conorm. For example, for  $T_P$  we use  $T_P$  for conjunction,  $S_P$  for disjunction,  $A_{\log T_P} = \log \circ A_{T_P}$  for aggregation and  $I_{RC}$  for implication (i.e. DPFL from Section 7.4). Symmetric configurations have the benefit of retaining many equivalence relations in fuzzy logic compared to an arbitrary configuration of operators. All are run with  $w_{DFL} = 1$  except for  $T_P$  which is run using  $w_{DFL} = 10$ . The results can be found in Table 6.

The Gödel t-norm performs worse than the supervised baseline. This is probably because the min aggregator is not a very good choice as it only increases the truth value of a single instance, which might be just an exception as argued in Section 6.1 and evident from the low values of  $cu_{cons}\%$  and  $cu_{ant}\%$ .

The Lukasiewicz t-norm with  $A_{LK}$  performs much worse than the supervised baseline. What is likely happening is that because  $A_{LK}$  either has a derivative of 0 or 1 everywhere, the total gradient is very large when the condition is met. If instead we use the mean average error, the results stabilize and end slightly higher than the supervised baseline. By the definition of  $I_{LK}$ ,  $\text{cons}\% = \frac{1}{2}$  as the consequent and negated antecedent derivatives are equal (see Equation 42).  $cu_{cons}\%$  is very low with only 0.05, which as we argued is worse than random guessing. As halve of the gradient is MP reasoning, that halve is nearly always incorrect.

The performance of the Yager t-norm seems highly dependent on the choice of the parameter  $p$ . For  $p = 20$  the top performance is quite a bit higher than the baseline, although in the end it drops. However, for  $p = 2$ , the results are even worse than the Lukasiewicz t-norm, which corresponds to  $p = 1$ .

The product t-norm performs best and also has the highest values for  $cu_{cons}\%$  and  $cu_{ant}\%$ . To a large extend this is because the log-product aggregator is very effective for this problem as we will see in the next section.

#### 9.4.2. Varying the Aggregators

In this section, we analyze symmetric configurations, except that we use aggregators other than the one formed by extending the t-norm. In particular, we will consider the RMSE aggregator ( $A_{pME}$  with  $p = 2$ ) and the log-product aggregator  $A_{\log T_P}$ .

		Accuracy	cons%	$cu_{cons}\%$	$cu_{ant}\%$
$A_{RMSE}$ $w_{DFL} = 1$	$T_G$	96.3	0.10	<b>0.89</b>	0.97
	$T_{LK}$	96.6	0.5	0.33	0.67
	$T_P$	<b>96.7</b>	0.44	0.48	0.69
	$T_Y, p = 2$	96.4	0.40	0.54	0.74
	$T_Y, p = 20$	96.0	0.29	0.51	0.78
	$T_{Nm}$	95.4	0.29	0.44	0.84
$A_{\log T_P}$ $w_{DFL} = 10$	$T_G$	96.2	0.10	<b>0.89</b>	0.97
	$T_{LK}$	96.9	0.5	0.06	0.95
	$T_P$	<b>97.0</b>	0.08	0.81	0.98
	$T_Y, p = 2$	96.6	0.12	0.87	0.97
	$T_Y, p = 20$	95.9	0.18	0.82	0.98
	$T_{Nm}$	95.4	0.03	0.62	<b>0.99</b>

Table 7: Configurations using the RMSE aggregator with  $w_{DFL} = 1$  and the log product aggregator with  $w_{dfl} = 10$ .

Table 7 shows the results when using the RMSE aggregator and a DFL weight of 1 and the log product aggregator and a DFL weight of 10. Nearly all configurations perform significantly better using these aggregators than when using their ‘symmetric’ aggregator. In particular, the Gödel, Lukasiewicz and Yager t-norms all outperform the baseline with both aggregators as they are differentiable everywhere and can handle outliers.

The product t-norm seems to do slightly worse with the RMSE aggregator than with the log-product aggregator. Like we discussed in Section 8.5,  $\text{cons}\%$  is higher using this aggregator because the corners  $a_i = 0, c_i = 0$  and  $a_i = 1, c_i = 1$  will have no gradient when using the RMSE aggregator. However, the values of  $cu_{cons}\%$  and  $cu_{ant}\%$  are much lower than when using the log-product aggregator. This could have to do with the previously made point: As it no longer has a gradient of 1 at the corners  $a = 0, c = 0$  and  $a = 1, c = 1$ , the large gradients are only when the agent is not yet confident about some prediction. This case is inherently

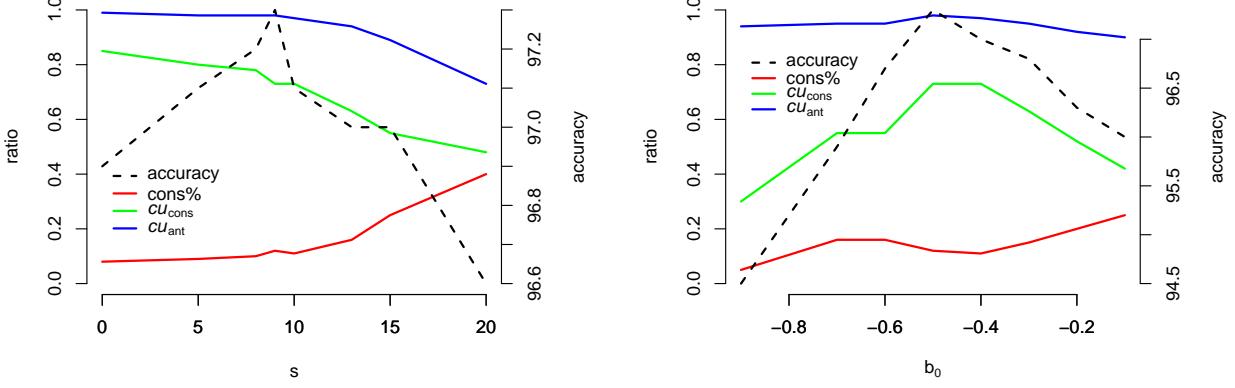


Figure 18: The results using the Reichenbach-sigmoidal implication  $\sigma_{I_{RC}}$ , the  $A_{\log T_P}$  aggregator,  $T_Y$  with  $p = 2$  and  $w_{DFL} = 10$ . Left shows the results for various values of  $s$ , keeping  $b_0$  fixed to  $-0.5$ , and right shows the results for various values of  $b_0$ , keeping  $s$  fixed to 9.

‘riskier’, but also contributes more information. It is not as informative to increase the confidence of  $a = 0$  if  $a$  is already very low.

The Lukasiewicz t-norm has a particularly high accuracy of 96.9% with the log product and is on the level of performance of the product t-norm. However, it has a very low value for  $cu_{\text{cons}}\%$  of 0.06 and a relatively low value for  $cu_{\text{ant}}\%$ . Interestingly, it is also the only configuration for which  $cu_{\text{cons}}\%$  is higher when using the RMSE aggregator than the log-product aggregator.

#### 9.4.3. Reichenbach-Sigmoidal Implication

The Reichenbach-sigmoidal implication  $\sigma_{I_{RC}}$  is a promising candidate for the choice of implication as we have argued in Section 8.5.1. We fix the aggregator to the log-product, the conjunction operator to the Yager t-norm with  $p = 2$ , and use a DFL weight of  $w_{DFL} = 10$ .

On the left plot of Figure 18 we find the results when we experiment with the parameter  $s$ , keeping  $b_0$  fixed to  $-\frac{1}{2}$ . Note that when  $s$  approaches 0 the Reichenbach-sigmoidal implication is  $I_{RC}$ . The value of 9 gives the best results, with 97.3% accuracy. Interestingly enough, there seem to be clear trends in the values of  $\text{cons}\%$ ,  $cu_{\text{cons}}\%$  and  $cu_{\text{ant}}\%$ . Increasing  $s$  seems to increase  $\text{cons}\%$ . This is because the antecedent derivative around the corner  $a = 0$ ,  $c = 0$  will be low, as argued in Section 8.5.1. When  $s$  increases, the corners will be more smoothed out. Furthermore, both  $cu_{\text{cons}}\%$  and  $cu_{\text{ant}}\%$  decrease when  $s$  increases. This could again be because around the corners the derivatives become small, and these are often the ‘safest’, as the model is already confident about those. For a higher value of  $s$ , most of the gradient magnitude is at less ‘safe’ instances. We note that the same happened when using the RMSE aggregator and the product t-norm. Regardless, the best parameter value clearly is not the one for which the values of  $cu_{\text{cons}}\%$  and  $cu_{\text{ant}}\%$  are highest, namely the Reichenbach implication itself.

On the right plot of Figure 18 we experiment with the value of  $b_0$ . Clearly,  $-\frac{1}{2}$  works best, having the highest accuracy and  $cu_{\text{cons}}\%$ .

#### 9.4.4. Conjunction Operators and Aggregators

Next, we compare the behavior of different t-norms in Table 8. As conjunctions are only used in the antecedents of implications, they get negative derivatives and thus have to choose which of the conjuncts to decrease. Therefore, they act like their dual t-conorm (see Section 7). The differences in accuracy are small and not significant except for the Nilpotent minimum.

Table 9 shows the results when varying the aggregator and DFL weight  $w_{DFL}$ .  $A_{2M}$  refers to the geometric mean by using  $p = 2$  in Equation 24. The log-product operator with  $w_{DFL} = 10$  does significantly better than the other aggregators. The geometric mean with  $w_{DFL} = 10$  has a very high value for  $cu_{\text{cons}}\%$  but still performs worst of the four. This is because it assigns the highest derivative to the most satisfied assignments, which are likely already correct.

	Accuracy	$\text{cons}\%$	$cu_{\text{cons}}\%$	$cu_{\text{ant}}\%$
$T_G$	<b>97.4</b>	0.10	0.76	0.98
$T_{LK}$	97.1	0.10	0.76	0.98
$T_P$	97.2	0.10	0.75	0.97
$T_Y$ , $p = 2$	97.3	0.12	0.73	0.98
$\sigma_{T_P}$	<b>97.4</b>	0.11	0.76	0.98
$T_{Nm}$	95.6	0.08	0.64	0.98

Table 8: The results using  $\sigma_{I_{RC}}$  for the implication,  $A_{\log T_P}$  for the aggregator with  $w_1 = 10$  and several different conjunction operators.

		Accuracy	cons%	$cu_{\text{cons}}\%$	$cu_{\text{ant}}\%$
$w_1 = 1$	$A_{RMSE}$	96.5	0.47	0.40	0.66
	$A_{\log T_P}$	96.2	0.15	0.64	0.95
	$A_{MAE}$	95.6	0.06	0.78	<b>0.99</b>
	$A_{2M}$	95.5	0.06	0.82	<b>0.99</b>
$w_1 = 10$	$A_{RMSE}$	96.6	0.46	0.40	0.75
	$A_{\log T_P}$	<b>97.3</b>	0.12	0.73	0.98
	$A_{MAE}$	96.5	0.06	0.82	<b>0.99</b>
	$A_{2M}$	96.2	0.05	<b>0.88</b>	<b>0.99</b>

Table 9: The results using  $\sigma_{I_{RC}}$  for the implication,  $T_Y$  for conjunction and some combinations of aggregators and different values of the DFL weight  $w_{DFL}$ .

	Accuracy	cons%	$cu_{\text{cons}}\%$	$cu_{\text{ant}}\%$
$I_{KD}$	96.1	0.10	<b>0.88</b>	0.97
$I_{LK}$	<b>97.0</b>	0.5	0.03	0.97
$I_{RC}$	96.9	0.08	0.85	<b>0.99</b>
$I_Y, p = 2$	96.6	0.12	0.87	0.97
$I_{FD}$	96.3	0.07	0.65	0.96
$I_G$	90.6	1	0.07	—
$I_{GG}$	94.0	0.86	0.01	0.97
$I_{T_Y}, p = 0.5$	96.3	0.18	0.65	0.37
$I_{T_Y}, p = 2, w_{DFL} = 0.1$	95.4	0.58	0.03	<b>0.99</b>

Table 10: The results using  $T_Y, p = 2$  for the conjunction,  $A_{\log T_P}$  for the aggregator with  $w_1 = 10$  and several different S-implications and R-implications.

#### 9.4.5. Implications

In Table 10, we compare different fuzzy implications, keeping the conjunction operator fixed to the Yager t-norm with  $p = 2$ , the aggregator to the log-product and the DFL weight to 10. Again, the Reichenbach implication and the Lukasiewicz implication work well, both having an accuracy around 97%. The Kleene Dienes and the Yager S- and R- implications surpass the baseline as well.

The Gödel implication and Goguen implication have worse performance than the supervised baseline. While the derivatives of  $I_{LK}$  and  $I_G$  only differ in that  $I_G$  disables the derivatives with respect to negated antecedent,  $I_{LK}$  performs among the best but  $I_G$  performs among the worst, suggesting that the derivatives with respect to the negated antecedent are required to successfully applying DFL. The Fodor implication is comparable in performance to the Kleene Dienes implication, which is not surprising as they are equal for all  $a > c$ .

#### 9.4.6. Influence of Individual Formulas

Finally, we compare what the influence of the different formulas are in Table 11. Removing the reflexivity formula (3) does not largely impact the performance. The biggest drop in performance is by removing formula (1) that defines the `same` predicate. Using only formula (1) gets slightly better performance than only using formula (2), despite the fact that no positive labeled examples can be found using formula (1) as the predicates `zero` to `nine` are not in its consequent. Since 95% of the derivatives are with respect to the negated antecedent, this formula contributes by finding additional counterexamples. Furthermore, improving the accuracy of the `same` predicate improves the accuracy on digit recognition: Just using the reflexivity formula (3) has the highest accuracy when used individually, even though it does not use the digit predicates.

Formulas used	Accuracy	cons%	$cu_{\text{cons}}\%$	$cu_{\text{ant}}\%$
(1) and (2)	<b>97.1</b>	0.05	0.54	<b>0.99</b>
(2) and (3)	95.9	0.12	<b>0.75</b>	0.95
(1) and (3)	96.3	0.15	0.52	0.98
(1)	95.6	0.05	0.59	<b>1.00</b>
(2)	95.2	0.03	<b>0.78</b>	0.99
(3)	<b>95.8</b>	0.19	0.64	0.95

Table 11: The results using  $\sigma_{I_{RC}}$  for the implication with  $s = 9$  and  $b_0 = -\frac{1}{2}$ ,  $T_Y, p = 2$  for the conjunction and  $A_{\log T_P}$  for the aggregator with  $w_{DFL} = 10$ , leaving some formulas out. The numbers indicated the formulas that are present during training.

#### 9.4.7. Analysis

We plot the accuracy of the different configurations with respect to  $cu_{\text{cons}}$  and  $cu_{\text{ant}}$  in Figures 19a and 19b. Figure 19b seems to show a positive correlation. Furthermore, the best configurations using  $\sigma_{I_{RC}}$  are the ones with the highest value of  $cu_{\text{ant}}$ . Although there seems to be a slight positive correlation in Figure 19a, it is not as pronounced and the configurations with the highest accuracy are not quite the ones with the highest value

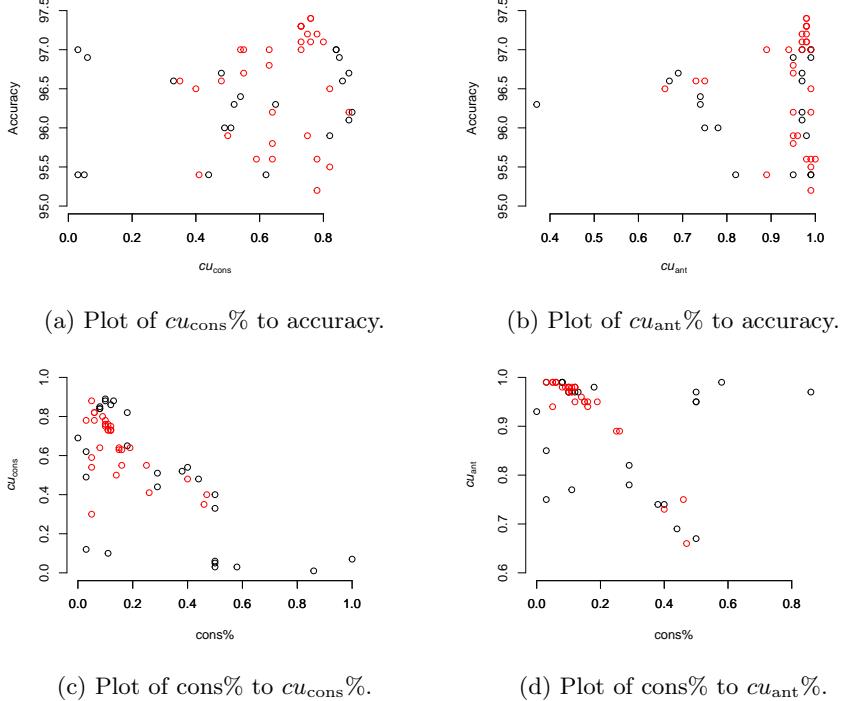


Figure 19: We plot several of the analytical measures to find their relations. Red dots represent configurations using  $\sigma_{I_{RC}}$ .

of  $cu_{cons}$ . Furthermore, there are decently performing methods, that use the Lukasiewicz implication, with low values for  $cu_{cons}$ .

We plot the values of  $cons\%$  to the values of  $cu_{cons}\%$  and  $cu_{ant}\%$  in Figures 19c and 19d. For both, there seems to be a negative correlation. Apparently, if the ratio of derivatives with respect to the consequent becomes larger, then this decreases the correctness of the updates. In Section 9.4.3 we argued, when experimenting with the value of  $s$ , that this could be because for lower values of  $cons\%$ , a smaller portion of the reasoning happens in the ‘safe’ corners around  $a = 0$ ,  $c = 0$  and  $a = 1$ ,  $c = 1$ , and more for cases that the agent is less certain about. As all S-implications have strong derivatives at both these corners (Proposition 6), this phenomenon is likely present in other S-implications.

#### 9.4.8. Conclusions

We have run experiments on many different configurations of hyperparameters to explore what works and what does not. The only well performing fully symmetric option is the product t-norm. If we are willing to forego symmetry, we find that using the Reichenbach-sigmoidal implication with the log-product aggregator and a DFL weight of 10 has somewhat better performance. The choice of conjunction does not seem to matter as much. The Lukasiewicz implication additionally does well with the log-product aggregator, suggesting that the choice of aggregator and DFL weight is quite vital.

Although Differentiable Fuzzy Logics significantly improves on the supervised baseline and is thus suited for semi-supervised learning, it is not competitive with state-of-the-art methods like Ladder Networks (Rasmus et al., 2015) which has an accuracy of 98.9% for 100 labeled pictures and 99.2% for 1000.<sup>14</sup>

## 10. Related Work

Differentiable Fuzzy Logics falls into the discipline of Statistical Relational Learning (Getoor and Taskar, 2007), which concerns models that can reason under uncertainty and learn relational structures like graphs.

### 10.1. Differentiable Fuzzy Logics

Special cases of DFL have been researched in several papers under different names. Real Logic (Serafini and Garcez, 2016) implements function symbols and uses a neural model called Logic Tensor Networks to interpret predicates. It uses t-norms with their dual t-conorm and their respective S-implications. Real Logic

<sup>14</sup>Evaluating and comparing semi-supervised methods is challenging because if the baseline supervised model is stronger, then the semi-supervised method is also going to have higher accuracy. However, the baseline used in that paper performs slightly worse than ours, legitimizing the comparison somewhat more.

is applied to weakly supervised learning on Scene Graph Parsing (Donadello et al., 2017) and transfer learning in Reinforcement Learning (Badreddine and Spranger, 2019).

Semantic-based regularization (SBR) (Diligenti et al., 2017a) applies DFL to kernel machines. They use R-implications and the mean aggregator. Sen et al. (2008) applies SBR to collective classification by predicting using a trained deep learning model, and then using gradient descent on the DFL loss to find new truth values. Using this method, predictions are consistent with the formulas during test-time.

Marra et al. (2019b) uses t-norm Fuzzy Logics, where the R-implication is used alongside weak disjunction. By using t-norms based on generator functions, the satisfiability computation can be simplified and generalizations of common loss functions can be found. Marra et al. (2018) applies DFL to image generation. It uses the product t-norm, the log-product aggregator and the Goguen implication. By using function symbols that represent generator neural networks, they create constraints that are used to create a semantic description of an image generation problem. Rocktäschel et al. (2015) uses the product t-norm and reichenbach implication for relation extraction by using an efficient matrix embedding of the rules. Guo et al. (2016) extends this to link prediction and triple classification by using a margin-based ranking loss for implications.

Demeester et al. (2016) uses a regularization technique equivalent to the Lukasiewicz implication. Instead of using existing data, it finds a loss function which does not iterate over objects, yet can guarantee that the rules hold. This is very scalable, but can only model simple implications. A promising approach is using adversarial sets (Minervini et al., 2017), which is a set of objects from the domain that do not satisfy the knowledge base. These are probably the most informative objects. It uses gradient descent to find objects that *minimize* the satisfiability. The parameters of the deep learning model are then updated so that it predicts consistent with the knowledge base on this adversarial set. A benefit of this approach is that it does not have to iterate over instances that already satisfy the constraints. Adversarial sets are applied to natural language interpretation in (Minervini and Riedel, 2018). Both papers use the Lukasiewicz implication and Gödel t- and t-conorm. They are not able to infer new labels on existing unlabeled data as they use artificial data, but these methods are not orthogonal and can be used jointly.

### 10.2. Methods using Differentiable Fuzzy Logic Operators

Posterior regularization (Ganchev and Gillenwater, 2010; Hu et al., 2016) is a framework for weakly-supervised learning on structured data. It projects the output of a deep learning model to a ‘rule-regularized subspace’ to make it consistent with the knowledge base. This output is used as a label for the deep learning model to imitate. Unlike this paper, it does not compute derivatives over the computation of the satisfaction of the knowledge base. Marra et al. (2019a) and Daniele and Serafini (2019) both instead use gradient descent for the projection. Therefore, unlike the other methods mentioned here, the derivatives with respect to the operators are relevant. They learn the formula weights jointly with the parameters of the deep learning model.

$\partial$ ILP (Evans and Grefenstette, 2018) is a differentiable inductive logic programming that uses the product t-norm and t-conorm to do differentiable inference. The Neural Theorem Prover (Rocktäschel and Riedel, 2017) does differentiable proving of queries and combines different proof paths using the Gödel t-norm and t-conorm. Šourek et al. (2015) also introduces a method for differentiable query proving, with learnable weights for rules. They use operators inspired by fuzzy logic and transformed by the sigmoid function.

There is a lot of literature on Fuzzy Neural Networks (Jang, 1993; Jang et al., 1997; Lin and Lee, 1991) that replace standard neural network neurons with neurons based on fuzzy logic. Some of the neurons use fuzzy logic operators which are differentiated through if the networks are trained using backpropagation.

### 10.3. Differentiable Probabilistic Logics

Some approaches use probabilistic logics instead of fuzzy logics and interpret predicates probabilistic. As deep learning classifiers model probability distributions, probabilistic logics could be a more natural choice than fuzzy logics. DeepProbLog (Manhaeve et al., 2018) is a probabilistic logic programming language with neural predicates that compute the probabilities of ground atoms. It supports automatic differentiation which can be used to back-propagate from the loss at a query predicate to the deep learning models that implement the neural predicates, similar to DFL. It supports probabilistic rules which can handle exceptions to rules. We compare another differentiable probabilistic logic called Semantic Loss (Xu et al., 2018) in Appendix D and show similarities between it and DFL using operators based on the product t-norm. This similarity suggests that many practical problems that DPFL has are also present in Semantic Loss. They apply Semantic Loss to MNIST semi-supervised learning with a different knowledge base than ours. As inference is exponential in the size of the grounding for probabilistic logics, both approaches use an advanced compilation technique (Darwiche, 2011) to make inference feasible for larger problems.

## 11. Discussion

This paper presented theoretical results of Differentiable Fuzzy Logics operators and then evaluated their behavior on semi-supervised learning on MNIST.

We now discuss some additional problems with deploying solutions using DFL.

DFL can be seen as a form of multi-objective optimization (Hwang and Masud, 2012). In the DFL loss (Equation 13) we sum up the valuations of different formulas, each of which is a separate objective. The loss landscape can significantly change when weights for individual formulas is changed. As we saw in Section 9.4.4, a lower value of the DFL weight has worse performance. Having so many different objectives requires significant hyperparameter tuning. A method capable of learning weights for the formulas jointly like (Marra et al., 2019a; Daniele and Serafini, 2019; Šourek et al., 2015), could solve this problem.

A second major challenge is related to the class imbalance problem (Japkowicz and Stephen, 2002; Buda et al., 2018). We argued in Section 8.1 that for a significant portion of common-sense background knowledge, the modus tollens case (i.e., the non-black non-raven) is by far the most common of the four cases. The simple and small MNIST problem indeed showed that most well-performing implications have a far larger derivative with respect to the negated antecedent than to the consequent. This imbalance will only increase for more complex problems. However, simply removing derivatives with respect to the antecedent does not seem to be the solution. A reason for this could be that those are usually correct, unlike derivatives with respect to the consequent. In fact, we found in Section 9.4.6 that the formula in which the digits are in the antecedent performs better on its own than the formula in which the digits are in the consequent, even though the model could not learn from any new positive examples.

Although we have focused on experimenting with the accuracy of the derivatives of the implication, it should be noted that the derivatives of the disjunction operator make a choice as well. For example, if the agent observes a walking object and the supervisor knows that only humans and animals can walk, how is the supervisor supposed to choose whether it is a human or an animal? Here, similar imbalances exist in the different possible classes: There might be more images of humans than of animals.

Lastly, we pose the question whether it is more important that we choose operators based on the performance on the task at hand, or based on its logical properties. The choice of operators that performed best in the MNIST problem uses the Gödel t-norm, the log-product aggregator and the Reichenbach-sigmoidal implication. None of these three operators are based on the same t-norm, that is, this choice of operators is not ‘symmetric’. Differentiable Product Fuzzy Logic is the only viable choice in which all of the operators can be chosen based on one t-norm, and it can only be used for formulas in prenex form. The largest benefit of a ‘symmetric’ choice of operators is that the truth value of formulas that are logically equivalent in classical logic will be equal. This makes it easier to analyze how the background knowledge will behave and does not require putting it in a particular form.

## 12. Conclusion

We analyzed Differentiable Fuzzy Logics in order to understand how reasoning using logical formulas behaves in a differentiable setting. We examined how the properties of a large amount of different operators affect DFL. We have found substantial differences between the properties of a large number of such Differentiable Fuzzy Logics operators, and we showed that many of them, including some of the most popular operators, are highly unsuitable for use in a differentiable learning setting. By analyzing aggregation functions, we found that the log-product aggregator and the RMSE aggregator have convenient connections to both fuzzy logic and machine learning and can deal with outliers. Next, we analyzed conjunction and disjunction operators and found several strong candidates. In particular, the Gödel t- and t-conorms are a strong yet simple choice, and the Yager t-norm and the product t-conorm have intuitive derivatives.

We noted an interesting imbalance between derivatives with respect to the negated antecedent and the consequent of the implication. Because the modus tollens case is much more common, we conclude that a large part of the useful inferences on the MNIST experiments are made by decreasing the antecedent, or by ‘modus tollens reasoning’. Furthermore, we found that derivatives with respect to the consequent often increase the truth value of something that is false as the consequent is false in the majority of times. Therefore, we argue that ‘modus tollens reasoning’ should be embraced in future research. As a possible solution to problems caused by this imbalance, we introduced a smoothed fuzzy implication called the Reichenbach-sigmoidal implication.

Experimentally, we found that the product t-norm is the only t-norm that can be used as a base for all choices of operators. The product t-conorm and the Reichenbach implication have derivatives that are intuitive and that correspond to inference rules from classical logic, and the log-product aggregator is among the most effective. The logic based on the product t-norm that we call Differentiable Product Fuzzy Logic (DPFL) has connections to probabilistic methods.

While the Lukasiewicz t-norm has about the same performance as DPFL on the MNIST problem when using the log-product aggregator, the Lukasiewicz aggregator vanishes on most of its domain and its relaxed version, the mean average error aggregator, is not able to distinguish outliers. However, the Lukasiewicz implication is the best R-implication in our experiments. Lastly, the Reichenbach-sigmoidal implication performs best on the MNIST experiments. The hyperparameters of sigmoidal implications can be tweaked to decrease the imbalance of the derivatives with respect to the negated antecedent and consequent. In order to gain the largest

improvements over a supervised baseline, we had to abandon the normal symmetric configurations of norms, where both t-norms, s-norms and the aggregation operators satisfy the usual algebraic relations. Instead, we had to resort to non-symmetric configurations where different norms are combined.

We believe a proper empirical comparison of different methods that introduce background knowledge through logic could be useful to properly understand the details, performance, possible applications and challenges of each method. Secondly, we believe more work is required in using background knowledge to help deep models train on real-world problems. One research direction would be to develop methods that can properly deal with exceptions. An approach in which the weights for the different formulas can be learned could be used to distinguish between relevant and irrelevant formulas in the background knowledge, and probabilistic instead of fuzzy logics could be a more natural fit. Lastly, additional research on the vast space of fuzzy logic operators might find more properties that are useful in DFL.

### 13. References

#### References

- Badreddine, S., Spranger, M., 2019. Injecting Prior Knowledge for Transfer Learning into Reinforcement Learning Algorithms using Logic Tensor Networks. arXiv preprint arXiv:1906.06576 .
- Ball, K., 1997. An elementary introduction to modern convex geometry. *Flavors of geometry* 31, 1–58.
- Besold, T.R., Garcez, A.d., Bader, S., Bowman, H., Domingos, P., Hitzler, P., Kuehnberger, K.U., Lamb, L.C., Lowd, D., Lima, P.M.V., de Penning, L., Pinkas, G., Poon, H., Zaverucha, G., 2017. Neural-Symbolic Learning and Reasoning: A Survey and Interpretation. arXiv preprint arXiv:1711.03902 URL: <http://arxiv.org/abs/1711.03902>.
- Bishop, C.M., 2006. Pattern Recognition and Machine Learning.
- Brock, A., Donahue, J., Simonyan, K., 2018. Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 .
- Buda, M., Maki, A., Mazurowski, M.A., 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks* 106, 249–259.
- Daniele, A., Serafini, L., 2019. Knowledge Enhanced Neural Networks, in: Nayak, A.C., Sharma, A. (Eds.), PRICAI 2019: Trends in Artificial Intelligence, Springer International Publishing, Cham. pp. 542–554.
- Darwiche, A., 2011. SDD: A new canonical representation of propositional knowledge bases. IJCAI International Joint Conference on Artificial Intelligence , 819–826doi:10.5591/978-1-57735-516-8/IJCAI11-143.
- David, H.A., Nagaraja, H.N., 2003. Order statistics, Third Edition. Encyclopedia of Statistical Sciences .
- Demeester, T., Rocktäschel, T., Riedel, S., 2016. Lifted Rule Injection for Relation Embeddings, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics. pp. 1389–1399. URL: <http://aclweb.org/anthology/D16-1146>, doi:10.18653/v1/D16-1146.
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 .
- Diligenti, M., Gori, M., Sacca, C., 2017a. Semantic-based regularization for learning and inference. *Artificial Intelligence* 244, 143–165.
- Diligenti, M., Roychowdhury, S., Gori, M., 2017b. Integrating Prior Knowledge into Deep Learning, in: Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on, IEEE. pp. 920–923.
- Donadello, I., Serafini, L., Garcez, A.d., 2017. Logic Tensor Networks for Semantic Image Interpretation, in: IJCAI International Joint Conference on Artificial Intelligence, pp. 1596–1602. URL: <http://arxiv.org/abs/1705.08968>.
- Evans, R., Grefenstette, E., 2018. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research* 61, 65–170. doi:10.1613/jair.5477.
- Ganchev, K., Gillenwater, J., 2010. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning Research* 11, 2001–2049. URL: <http://dl.acm.org/citation.cfm?id=1756006.1859918>.

- Garcez, A.S.d., Broda, K.B., Gabbay, D.M., 2012. Neural-symbolic learning systems: foundations and applications. Springer Science & Business Media.
- Garnelo, M., Arulkumaran, K., Shanahan, M., 2016. Towards deep symbolic reinforcement learning. arXiv preprint arXiv:1609.05518 .
- Getoor, L., Taskar, B., 2007. Introduction to statistical relational learning. volume 1. MIT press Cambridge.
- Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y., 2016. Deep learning. volume 1. MIT press Cambridge.
- Guha, R., 2015. Towards a model theory for distributed representations, in: 2015 AAAI Spring Symposium Series.
- Guo, S., Wang, Q., Wang, L., Wang, B., Guo, L., 2016. Jointly embedding knowledge graphs and logical rules, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 192–202.
- Gupta, A.K., Nadarajah, S., 2004. Handbook of beta distribution and its applications. CRC press.
- Hall, P., 1927. The Distribution of Means for Samples of Size N Drawn from a Population in which the Variate Takes Values Between 0 and 1, All Such Values Being Equally Probable. *Biometrika* 19, 240–245. URL: <http://www.jstor.org/stable/2331961>.
- Harnad, S., 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena* 42, 335–346.
- Hempel, C.G., 1945. Studies in the Logic of Confirmation (II.). *Mind* 54, 97–121. URL: <http://www.jstor.org/stable/2250948>, doi:10.2307/2250948.
- Hu, Z., Ma, X., Liu, Z., Hovy, E., Xing, E., 2016. Harnessing Deep Neural Networks with Logic Rules, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics. pp. 2410–2420. URL: <http://aclweb.org/anthology/P16-1228>, doi:10.18653/v1/P16-1228.
- Hwang, C.L., Masud, A.S.M., 2012. Multiple objective decision making—methods and applications: a state-of-the-art survey. volume 164. Springer Science & Business Media.
- Irwin, J.O., 1927. On the Frequency Distribution of the Means of Samples from a Population Having any Law of Frequency with Finite Moments, with Special Reference to Pearson's Type II. *Biometrika* , 225–239.
- Jang, J.S.R., 1993. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man and Cybernetics* doi:10.1109/21.256541.
- Jang, J.S.R., Sun, C.T., Mizutani, E., 1997. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence.
- Japkowicz, N., Stephen, S., 2002. The class imbalance problem: A systematic study. *Intelligent data analysis* 6, 429–449.
- Jayaram, B., Baczyński, M., 2008. Fuzzy Implications. volume 231. Springer, Berlin, Heidelberg. URL: <http://link.springer.com/10.1007/978-3-540-69082-5>, doi:10.1007/978-3-540-69082-5.
- Klement, E.P., Mesiar, R., Pap, E., 2013. Triangular norms. volume 8. Springer Science & Business Media.
- Klir, G., Yuan, B., 1995. Fuzzy sets and fuzzy logic. volume 4. Prentice hall New Jersey.
- van Krieken, E., Acar, E., van Harmelen, F., 2019. Semi-Supervised Learning using Differentiable Reasoning. *IFCoLog Journal of Logic and its Applications* 6, 633–653.
- LeCun, Y., Cortes, C., 2010. MNIST handwritten digit database URL: <http://yann.lecun.com/exdb/mnist/>.
- Lin, C.T., Lee, G.C., 1991. Neural-Network-Based Fuzzy Logic Control and Decision System. *IEEE Transactions on Computers* doi:10.1109/12.106218.
- Liu, Y., Kerre, E., 1998. An overview of fuzzy quantifiers.(I). Interpretations. *Fuzzy Sets and Systems* 95, 1–21.
- Manhaeve, R., Dumancić, S., Kimmig, A., Demeester, T., De Raedt, L., 2018. DeepProbLog: Neural Probabilistic Logic Programming, in: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. URL: <http://arxiv.org/abs/1805.10872><http://papers.nips.cc/book/advances-in-neural-information-processing-systems-31-2018>.

- Marcus, G., 2018. Deep learning: A critical appraisal. arXiv preprint arXiv:1801.00631 .
- Marra, G., Giannini, F., Diligenti, M., Gori, M., 2018. Constraint-Based Visual Generation. arXiv preprint arXiv:1807.09202 .
- Marra, G., Giannini, F., Diligenti, M., Gori, M., 2019a. Integrating Learning and Reasoning with Deep Logic Models. arXiv preprint arXiv:1901.04195 , 1–17URL: <https://arxiv.org/pdf/1901.04195v1.pdf>.
- Marra, G., Giannini, F., Diligenti, M., Maggini, M., Gori, M., 2019b. Learning and T-Norms Theory. arXiv preprint arXiv:1907.11468 .
- Mayo, M., 2003. Symbol Grounding and its Implications for Artificial Intelligence BT. Twenty-Sixth Australasian Computer Science Conference (ACSC2003) 16, 55–60. URL: <http://crpit.com/confpapers/CRPITV16Mayo.pdf>.
- Minervini, P., Demeester, T., Rocktäschel, T., Riedel, S., 2017. Adversarial sets for regularising neural link predictors, in: Uncertainty in Artificial Intelligence-Proceedings of the 33rd Conference, UAI 2017.
- Minervini, P., Riedel, S., 2018. Adversarially Regularising Neural NLI Models to Integrate Logical Background Knowledge, in: Proceedings of the 22nd Conference on Computational Natural Language Learning, pp. 65–74.
- Muggleton, S., de Raedt, L., 1994. Inductive Logic Programming: Theory and methods. The Journal of Logic Programming 19-20, 629–679. URL: <http://www.sciencedirect.com/science/article/pii/0743106694900353>, doi:[https://doi.org/10.1016/0743-1066\(94\)90035-3](https://doi.org/10.1016/0743-1066(94)90035-3).
- Murphy, K.P., Weiss, Y., Jordan, M.I., 2013. Loopy Belief Propagation for Approximate Inference: An Empirical Study, in: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc.. pp. 467–475. URL: <http://arxiv.org/abs/1301.6725>, doi:10.1.1.32.5538.
- Páll Jónsson, H., 2018. Real Logic and Logical Tensor Networks. University of Amsterdam, MSc thesis , 45.
- Pearl, J., 1988. Probabilistic reasoning in intelligent systems: networks of plausible inference. Elsevier.
- Pearl, J., 2018. Theoretical Impediments to Machine Learning With Seven Sparks from the Causal Revolution, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, ACM. p. 3.
- Pham, T.G., Turkkan, N., 1994. Reliability of a Standby System with Beta-Distributed Component Lives. IEEE Transactions on Reliability doi:10.1109/24.285114.
- Rasmus, A., Berglund, M., Honkala, M., Valpola, H., Raiko, T., 2015. Semi-supervised learning with ladder networks, in: Advances in Neural Information Processing Systems, pp. 3546–3554.
- Rocktäschel, T., Riedel, S., 2017. End-to-End Differentiable Proving URL: <https://arxiv.org/pdf/1705.11040.pdf>[http://arxiv.org/abs/1705.11040](https://arxiv.org/abs/1705.11040).
- Rocktäschel, T., Singh, S., Riedel, S., 2015. Injecting Logical Background Knowledge into Embeddings for Relation Extraction, in: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1119–1129. URL: <https://rockt.github.io/pdf/rocktaschel2015injecting.pdf><http://www.aclweb.org/anthology/N15-1118>, doi:10.3115/v1/N15-1118.
- Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T., 2008. Collective classification in network data. AI magazine 29, 93.
- Serafini, L., Garcez, A.D., 2016. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. CEUR Workshop Proceedings 1768.
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., others, 2017. Mastering the game of Go without human knowledge. Nature 550, 354.
- Socher, R., Chen, D., Manning, C.D., Ng, A.Y., 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. Proc.\ of NIPS'13 , 1–10doi:10.1023/A:1004554217069.
- Šourek, G., Aschenbrenner, V., Železný, F., Kuželka, O., 2015. Lifted relational neural networks, in: CEUR Workshop Proceedings. URL: <https://arxiv.org/pdf/1508.05128.pdf>.
- Šourek, G., Aschenbrenner, V., Železný, F., Schockaert, S., Kuželka, O., 2018. Lifted relational neural networks: Efficient learning of latent relational structures. Journal of Artificial Intelligence Research 62, 69–100. doi:10.1613/jair.1.11203.

Vranas, P.B., 2004. Hempel’s raven paradox: A lacuna in the standard Bayesian solution. British Journal for the Philosophy of Science 55, 545–560. doi:10.1093/bjps/55.3.545.

Weisberg, H., 1971. The Distribution of Linear Combinations of Order Statistics from the Uniform Distribution. Ann. Math. Statist. 42, 704–709. URL: <https://doi.org/10.1214/aoms/1177693419>, doi:10.1214/aoms/1177693419.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., den Broeck, G., 2018. A Semantic Loss Function for Deep Learning with Symbolic Knowledge, in: Dy, J., Krause, A. (Eds.), Proceedings of the 35th International Conference on Machine Learning, PMLR, Stockholmsmässan, Stockholm Sweden. pp. 5502–5511. URL: <http://proceedings.mlr.press/v80/xu18h.html>.

Yager, R.R., 1980. On a general class of fuzzy connectives. Fuzzy Sets and Systems 4, 235–242.

Zhou, Z.H., 2017. A brief introduction to weakly supervised learning. National Science Review 5, 44–53.

## Appendix A. Implementation of Differentiable Fuzzy Logics

---

**Algorithm 1** Computation of the Differentiable Fuzzy Logics loss. First it computes the fuzzy Herbrand interpretation  $g$  given the current embedded interpretation  $\eta_\theta$ . This performs a forward pass through the neural networks that are used to interpret the predicates. Then it computes the valuation of each formula  $\varphi$  in the knowledge base  $\mathcal{K}$ , implementing Equations 6-11.

---

```

1: function  $e_{N,T,S,I,A}(\varphi, g, C, \mu)$                                 ▷ The valuation function computes the Fuzzy truth value of  $\varphi$ .
2:   if  $\varphi = P(x_1, \dots, x_m)$  then
3:     return  $g[P, (\mu(x_1), \dots, \mu(x_m))]$                                 ▷ Find the truth value of a ground atom using the dictionary  $g$ .
4:   else if  $\varphi = \neg\phi$  then
5:     return  $N(e_{N,T,S,I,A}(\phi, g, C, \mu))$ 
6:   else if  $\varphi = \phi \wedge \psi$  then
7:     return  $T(e_{N,T,S,I,A}(\phi, g, C, \mu), e_{N,T,S,I,A}(\psi, g, C, \mu))$ 
8:   else if  $\varphi = \phi \vee \psi$  then
9:     return  $S(e_{N,T,S,I,A}(\phi, g, C, \mu), e_{N,T,S,I,A}(\psi, g, C, \mu))$ 
10:  else if  $\varphi = \phi \rightarrow \psi$  then
11:    return  $I(e_{N,T,S,I,A}(\phi, g, C, \mu), e_{N,T,S,I,A}(\psi, g, C, \mu))$ 
12:  else if  $\varphi = \forall x \phi$  then                                         ▷ Apply the aggregation operator as a quantifier.
13:    return  $A_{o \in C} e_{N,T,S,I,A}(\phi, g, C, \mu \cup \{x/o\})$       ▷ Each assignment can be seen as an instance of  $\varphi$ .
14:  end if
15: end function
16:
17: procedure DFL( $\eta_\theta, \mathcal{P}, \mathcal{K}, O, N, T, S, I, A$ )                                ▷ Computes the Differentiable Fuzzy Logics loss.
18:    $C \leftarrow o_1, \dots, o_b$  sampled from  $O$                                          ▷ Sample  $b$  constants to use this pass.
19:    $g \leftarrow dict()$                                                        ▷ Collects truth values for ground atoms.
20:   for  $P \in \mathcal{P}$  do
21:     for  $o_1, \dots, o_{\alpha(P)} \in C$  do
22:        $g[P, (o_1, \dots, o_{\alpha(P)})] \leftarrow \eta_\theta(P)(o_1, \dots, o_{\alpha(P)})$       ▷ Calculate the truth values of the ground atoms.
23:     end for
24:   end for
25:   return  $A_{\varphi \in \mathcal{K}} w_\varphi \cdot e_{N,T,S,I,A}(\varphi, g, C, \emptyset)$  ▷ Calculate valuation of the formulas  $\varphi$ . Start with an empty variable assignment. This implements Equation 13.
26: end procedure

```

---

The computation of the satisfaction is shown in pseudocode form in Algorithm 1. By first computing the dictionary  $g$  that contains truth values for all ground atoms,<sup>15</sup> we can reduce the amount of forward passes through the computations of the truth values of the ground atoms that are required to compute the satisfaction.

This algorithm can fairly easily be parallelized for efficient computation on a GPU by noting that the individual terms that are aggregated over in line 12 (the different *instances* of the universal quantifier) are not dependent on each other. By noting that formulas are in prenex normal form, we can set up the dictionary  $g$  using tensor operations so that the recursion has to be done only once for each formula. This can be done by

---

<sup>15</sup>The dictionary  $g$  could be seen as a ‘fuzzy Herbrand interpretation’, in that it assigns a truth value to all ground atoms.

applying the fuzzy operators elementwise over vectors of truth values instead of a single truth value, where each element of the vector represents a variable assignment.

The complexity of this computation then is  $O(|\mathcal{K}| \cdot P \cdot b^d)$ , where  $\mathcal{K}$  is the set of formulas,  $P$  is the amount of predicates used in each formula and  $d$  is the maximum depth of nesting of universal quantifiers in the formulas in  $\mathcal{K}$  (known as the **quantifier rank**). This is exponential in the amount of quantifiers, as every object from the constants  $C$  has to be iterated over in line 12, although as mentioned earlier this can be mitigated somewhat using efficient parallelization. Still, computing the valuation for transitive rules (such as  $\forall x y z Q(x, z) \wedge R(z, y) \rightarrow P(x, y)$ ) will for example be far more demanding than for antisymmetry formulas (such as  $\forall x y P(x, y) \rightarrow \neg P(y, x)$ ).

## Appendix B. Proofs

### Appendix B.1. Single-Passing

**Proposition 7.** Any composition of single-passing functions is also single-passing.

*Proof.* We will proof this by induction. Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be a single-passing function and let  $x_1, \dots, x_n \in \mathbb{R}$ . Then clearly  $f(x_1, \dots, x_n)$  is single-passing.

Next, let  $g : \mathbb{R}^n \rightarrow \mathbb{R}$  be a single-passing function and each  $y_1, \dots, y_n$  be a composition of single-passing functions. Let  $X_i$  be the set of inputs to  $y_i$ . For any  $x \in X_i$  holds that

$$\frac{\partial g(y_1(X_1), \dots, y_n(X_n))}{\partial x} = \frac{\partial g(y_1(X_1), \dots, y_n(X_n))}{\partial y_i(X_i)} \frac{\partial y_i(X_i)}{\partial x}. \quad (\text{B.1})$$

As  $g$  is single-passing, there is at most 1  $y_i$  so that  $\frac{\partial g(y_1(X_1), \dots, y_n(X_n))}{\partial y_i(X_i)} \neq 0$ . If there is 0, then there can also be no  $x \in \cap_j X_j$  such that  $\frac{\partial g(y_1(X_1), \dots, y_n(X_n))}{\partial x} \neq 0$ . If there is 1, then by the inductive hypothesis,  $y_i(X_i)$  is single-passing. Therefore, there is at most 1 value  $x \in X_i$  so that  $\frac{\partial y_i(X_i)}{\partial x} \neq 0$  and by Equation B.1 there is at most 1 value  $x \in \cap_j X_j$  such that  $\frac{\partial g(y_1(X_1), \dots, y_n(X_n))}{\partial x} \neq 0$ . We conclude that  $g(y_1(X_1), \dots, y_n(X_n))$  is single-passing.  $\square$

### Appendix B.2. Nonvanishing Fractions

#### Appendix B.2.1. Lukasiewicz Aggregator

**Proposition 8.** The fraction of inputs  $x_1, \dots, x_n \in [0, 1]$  for which the derivative of  $A_{T_{LU}}$  is nonvanishing is  $\frac{1}{n!}$ .

*Proof.* Consider standard uniformly distributed random variables  $x_1, \dots, x_n \sim U(0, 1)$ . The sum  $Y = \sum_{i=1}^n x_i$  is Irwin-Hall distributed (Irwin, 1927; Hall, 1927). The cumulative density function of this distribution is

$$F_Y(y) = \frac{1}{n!} \sum_{k=0}^{\lfloor y \rfloor} (-1)^k \binom{n}{k} (y - k)^{n-1}. \quad (\text{B.2})$$

The derivative of  $A_{T_{LU}}$  is nonvanishing when  $Y > n - 1$ , or equivalently as the Irwin-Hall distribution is symmetric, when  $Y < 1$ . Using  $F_Y$  gives  $F_Y(1) = \frac{1}{n!} ((-1)^0 \binom{n}{0} (1 - 0)^n + (-1)^1 \binom{n}{1} (1 - 1)^n) = \frac{1}{n!}$ .  $\square$

#### Appendix B.2.2. Yager Aggregator

**Proposition 9.** The fraction of inputs  $x_1, \dots, x_n \in [0, 1]$  for which the derivative of  $A_{T_Y}$  with  $p = 2$  is nonvanishing is  $\frac{\pi^{\frac{n}{2}}}{2^n \cdot \Gamma(\frac{1}{2}n + \frac{1}{2})}$ , where  $\Gamma$  is the Gamma function.

*Proof.* The points  $x_1, \dots, x_n \in \mathbb{R}$  for which  $\sum_{i=1}^n (1 - x_i)^2 < 1$  holds describes the volume of an  $n$ -ball<sup>16</sup> with radius 1. This volume is found by (Ball, 1997)(p.5):

$$V(n) = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{1}{2}n + 1)}. \quad (\text{B.3})$$

We are interested in the part of this volume where  $x_1, \dots, x_n \in [0, 1]$ , that is, those in a single orthant<sup>17</sup> of the  $n$ -ball. The amount of orthants in which an  $n$ -ball lies is  $2^n$ .<sup>18</sup> Thus, the volume of the part of the  $n$ -ball where  $x_1, \dots, x_n \in [0, 1]$  is  $\frac{V(n)}{2^n} = \frac{\pi^{\frac{n}{2}}}{2^n \cdot \Gamma(\frac{1}{2}n + \frac{1}{2})}$ . As the total volume of points lying in  $[0, 1]^n$  is 1, this is also the fraction of points for which the derivative of  $A_{T_Y}$  with  $p = 2$  is nonvanishing.  $\square$

<sup>16</sup>An  $n$ -ball is the generalization of the concept of a ball to any dimension and is the region enclosed by a  $n - 1$  hypersphere. For example, the 3-ball (or ball) is surrounded by a sphere (or 2-sphere). Similarly, the 2-ball (or disk) is surrounded by a circle (or 1-sphere). A hypersphere with radius 1 is the set of points which are at a distance of 1 from its center.

<sup>17</sup>An orthant in  $n$  dimensions is a generalization of the quadrant in two dimensions and the octant in three dimensions.

<sup>18</sup>To help understand this, consider  $n = 2$ . The 1-ball is the circle with center  $(0, 0)$ . The area of this circle is evenly distributed over the four quadrants.

### Appendix B.2.3. Nilpotent Aggregator

**Proposition 10.** The fraction of inputs  $x_1, \dots, x_n \in [0, 1]$  for which the derivative of  $A_{T_{nM}}$  is nonvanishing is  $\frac{1}{2^{n-1}}$ .

*Proof.* Consider  $n$  standard uniformly distributed random variables  $x_1, \dots, x_n \sim U(0, 1)$ . We are interested in the probability that  $x_{(1)} + x_{(2)} > 1$ , where  $x_{(k)}$  is the  $k$ -th smallest sample (known as the  $k$ -th order statistic (David and Nagaraja, 2003)). Weisberg (1971) derives the cumulative density function for linear combinations of standard uniform order statistics. Let  $k_0 = 0 < k_1 < \dots < k_S \leq k_n$  be  $S$  integers indicating the coefficients  $d_i > 0$ . We aim to calculate the probability that  $\sum_{i=1}^S d_i x_{(i)} > v$ . Let  $r_s = k_s - k_{s-1}$  for all  $1, \dots, S$  and let  $r_{S+1} = n - k_S$ . Finally, let  $c_{S+1} = 0$  and  $c_{(s)} = c_{(s+1)} + d_i$ .  $m$  is the largest integer so that  $v \leq c_{(m)}$ . Then, Weisberg (1971) finds that

$$P \left\{ \sum_{s=1}^S d_s x_{(k_s)} > v \right\} = \sum_{s=1}^m \frac{g_s^{(r_s-1)}(c_{(s)})}{(r_s-1)!} \quad (\text{B.4})$$

where  $g_s^{(i)}$  is the  $i$ -th order derivative of

$$g_s(c) = \frac{(c-v)^n}{c \prod_{i=1, i \neq s}^{S+1} (c - c_{(i)})^{r_i}} \quad (\text{B.5})$$

Filling this in for our case, we find that  $S = 2$ , where  $k_1 = 1$ ,  $k_2 = 2$  as  $d_1 = d_2 = 1$ . Therefore,  $r_1 = r_2 = 1$  and  $r_3 = n - 2$  and  $c_{(1)} = 2$ ,  $c_{(2)} = 1$  and  $c_{(3)} = 0$ . The largest integer  $m$  so that  $1 \leq c_{(m)}$  is 2. Filling this in, we find that

$$P \{x_{(1)} + x_{(2)} > 1\} = \frac{g_1^{(1-1)}(2)}{(1-1)!} + \frac{g_2^{(1-1)}(1)}{(1-1)!} \quad (\text{B.6})$$

$$= \frac{(2-1)^n}{2(2-1)^1(2-0)^{n-2}} + \frac{(1-1)^n}{1(1-2)(1-0)^{n-2}} = \frac{1}{2^{n-1}} \quad (\text{B.7})$$

□

## Appendix C. Derivations of Used Functions

### Appendix C.1. $p$ -Error Aggregators

The unbounded Yager aggregator is

$$A_{UY}(x_1, \dots, x_n) = 1 - \left( \sum_{i=1}^n (1-x_i)^p \right)^{\frac{1}{p}}, \quad p \geq 0. \quad (\text{C.1})$$

We can do a linear transformation  $w \cdot A_{UY}(x_1, \dots, x_n) - h$  on this function to ensure the boundary conditions in Definition 6 hold.

$$w \cdot A_{UY}(0, \dots, 0) - h = 0 \quad (\text{C.2})$$

$$w \cdot A_{UY}(1, \dots, 1) - h = 0 \quad (\text{C.3})$$

Solving Equation C.2 and C.3 for  $h$ , we find

$$\begin{aligned} w \cdot \left( 1 - \left( \sum_{i=1}^n (1-0)^p \right)^{\frac{1}{p}} \right) - h &= 0 \\ w \cdot \left( 1 - n^{\frac{1}{p}} \right) - h &= 0 \\ h &= w - w \cdot \sqrt[p]{n}, \end{aligned} \quad (\text{C.4})$$

$$\begin{aligned} w \cdot \left( 1 - \left( \sum_{i=1}^n (1-1)^p \right)^{\frac{1}{p}} \right) - h &= 1 \\ w \cdot (1-0) - h &= 1 \\ h &= w - 1. \end{aligned} \quad (\text{C.5})$$

Equating C.4 and C.5 and solving for  $h$ , we find

$$\begin{aligned} w - w \cdot \sqrt[p]{n} &= w - 1 \\ w &= \frac{1}{\sqrt[p]{n}}. \end{aligned} \quad (\text{C.6})$$

And so  $h = \frac{1}{\sqrt[p]{n}} - 1$ . Filling in and simplifying we find

$$\begin{aligned} A_{pE}(x_1, \dots, x_n) &= \frac{1}{\sqrt[p]{n}} \cdot \left( 1 - \left( \sum_{i=1}^n (1 - x_i)^p \right)^{\frac{1}{p}} \right) - \left( \frac{1}{\sqrt[p]{n}} - 1 \right) \\ &= 1 - \frac{1}{\sqrt[p]{n}} \cdot \left( \sum_{i=1}^n (1 - x_i)^p \right)^{\frac{1}{p}} \\ &= 1 - \left( \frac{1}{n} \sum_{i=1}^n (1 - x_i)^p \right)^{\frac{1}{p}} \end{aligned} \quad (\text{C.7})$$

Similarly for the t-conorm  $A_{UYS}(x_1, \dots, x_n) = (\sum_{i=1}^n x_i^p)^{\frac{1}{p}}$ ,  $p \geq 0$ .

$$\begin{aligned} w \cdot \left( \sum_{i=1}^n 0^p \right)^{\frac{1}{p}} - h &= 0 \\ w \cdot 0 - h &= 0 \\ h &= 0 \end{aligned} \quad (\text{C.8})$$

$$\begin{aligned} w \cdot \left( \sum_{i=1}^n 1^p \right)^{\frac{1}{p}} - h &= 1 \\ w \cdot n^{\frac{1}{p}} - h &= 1 \\ w &= \frac{1}{\sqrt[p]{n}} \end{aligned} \quad (\text{C.9})$$

$$A_{p-MEAN}(x_1, \dots, x_n) = \left( \frac{1}{n} \sum_{i=1}^n x_i^p \right)^{\frac{1}{p}}, p \geq 0. \quad (\text{C.10})$$

### Appendix C.2. Sigmoidal Functions

In Machine Learning, the logistic function or sigmoid function  $\sigma(x) = \frac{1}{1+e^{-x}}$  is a common activation function (Goodfellow et al., 2016)(p.65-66). This inspired (šourek et al., 2018) to introduce parameterized families of aggregation functions they call Max-Sigmoid activation functions:

$$A'_{\sigma+\wedge}(x_1, \dots, x_n) = \sigma \left( s \cdot \left( \sum_{i=1}^n x_i - n + 1 + b_0 \right) \right), \quad A'_{\sigma+\vee}(x_1, \dots, x_n) = \sigma \left( s \cdot \left( \sum_{i=1}^n x_i + b_0 \right) \right) \quad (\text{C.11})$$

We generalize this transformation for any function  $f : [0, 1]^n \rightarrow \mathbb{R}$  that is symmetric and increasing:

$$A'_{\sigma f}(x_1, \dots, x_n) = \sigma(s \cdot (f(x_1, \dots, x_n) + b_0)) \quad (\text{C.12})$$

This cannot be an aggregation function according to Definition 6 as  $\sigma \in (0, 1)$  and so the boundary conditions  $A'_\sigma(1, \dots, 1) = 1$  and  $A'_\sigma(0, \dots, 0) = 0$  do not hold. We can solve this by adding two linear parameters  $w$  and  $h$ , redefining  $\sigma_f$  as

$$\sigma_f(x_1, \dots, x_n) = w \cdot \sigma(s \cdot (f(x_1, \dots, x_n) + b_0)) - h \quad (\text{C.13})$$

For this, we need to make sure the lowest value of  $f$  on the domain  $[0, 1]^n$  maps to 0 and the highest to 1. For this, we define  $\inf_f = \inf\{f(x_1, \dots, x_n) | x_1, \dots, x_n \in [0, 1]\}$  and  $\sup_f = \sup\{f(x_1, \dots, x_n) | x_1, \dots, x_n \in [0, 1]\}$ . This gives the following system of equations

$$A_\sigma(0, \dots, 0) = w \cdot \sigma(s \cdot (\inf_f + b_0)) - h = 0 \quad (\text{C.14})$$

$$A_\sigma(1, \dots, 1) = w \cdot \sigma(s \cdot (\sup_f + b_0)) - h = 1 \quad (\text{C.15})$$

First solve both equations for  $w$ , starting with Equation C.14:

$$\begin{aligned} w \cdot \sigma(s \cdot (\inf_f + b_0)) - h &= 0 \\ \frac{1}{1 + e^{-s \cdot (\inf_f + b_0)}} &= \frac{h}{w} \\ w &= h \cdot (1 + e^{-s \cdot (\inf_f + b_0)}) \end{aligned} \quad (\text{C.16})$$

Likewise for Equation C.15:

$$\begin{aligned} w \cdot \sigma(s \cdot (\sup_f + b_0)) - h &= 1 \\ \frac{1}{1 + e^{-s \cdot (\sup_f + b_0)}} &= \frac{1 + h}{w} \\ w &= (1 + h) \cdot (1 + e^{-s \cdot (\sup_f + b_0)}) \end{aligned} \quad (\text{C.17})$$

Now we can solve for  $h$  by equating Equations C.17 and C.16. Substitute  $y_1 = 1 + e^{-s \cdot (\sup_f + b_0)}$  and  $y_2 = 1 + e^{-s \cdot (\inf_f + b_0)}$ . Then

$$\begin{aligned} (1 + h) \cdot 1 + e^{-s \cdot (\sup_f + b_0)} &= h \cdot 1 + e^{-s \cdot (\inf_f + b_0)} \\ h &= \frac{1 + e^{-s \cdot (\sup_f + b_0)}}{1 + e^{-s \cdot (\inf_f + b_0)} - 1 + e^{-s \cdot (\sup_f + b_0)}} \end{aligned}$$

We thus get the following formula:

$$A_\sigma(x_1, \dots, x_n) = \frac{1 + e^{-s \cdot (\sup_f + b_0)}}{e^{-s \cdot (\inf_f + b_0)} - e^{-s \cdot (\sup_f + b_0)}} \cdot \left( \left( 1 + e^{-s \cdot (\inf_f + b_0)} \right) \cdot \sigma(s \cdot (f(x_1, \dots, x_n) + b_0)) - 1 \right) \quad (\text{C.18})$$

If  $f$  is a fuzzy logic operator of which the outputs are all in  $[0, 1]$ , the most straightforward choice of  $b_0$  is  $-\frac{1}{2}$ . This translates the outputs of  $f$  to  $[-\frac{1}{2}, \frac{1}{2}]$  and so it uses a symmetric part of the sigmoid function. For some function  $f \in [0, 1]^n \rightarrow [0, 1]$ , we find the following simplification, noting that the supremum of  $f$  is 1 and the infimum is 0:

$$\begin{aligned} \sigma_f(x_1, \dots, x_n) &= \frac{1 + e^{-s(1 - \frac{1}{2})}}{e^{s(0 - \frac{1}{2})} - e^{-s(1 - \frac{1}{2})}} \cdot \\ &\quad \left( \left( 1 + e^{-s(0 - \frac{1}{2})} \right) \cdot \sigma \left( s \cdot \left( f(x_1, \dots, x_n) - \frac{1}{2} \right) \right) - 1 \right) \end{aligned} \quad (\text{C.19})$$

$$= \frac{1 + e^{-\frac{s}{2}}}{e^{\frac{s}{2}} - e^{-\frac{s}{2}}} \cdot \frac{e^{\frac{s}{2}} - 1}{e^{\frac{s}{2}} - 1} \cdot \left( \left( 1 + e^{\frac{s}{2}} \right) \cdot \sigma \left( s \cdot \left( f(x_1, \dots, x_n) - \frac{1}{2} \right) \right) - 1 \right) \quad (\text{C.20})$$

$$= \frac{e^{\frac{s}{2}} - e^{-\frac{s}{2}}}{(e^{\frac{s}{2}} - e^{-\frac{s}{2}})(e^{\frac{s}{2}} - 1)} \cdot \left( \left( 1 + e^{\frac{s}{2}} \right) \cdot \sigma \left( s \cdot \left( f(x_1, \dots, x_n) - \frac{1}{2} \right) \right) - 1 \right) \quad (\text{C.21})$$

$$= \frac{1}{e^{\frac{s}{2}} - 1} \cdot \left( \left( 1 + e^{\frac{s}{2}} \right) \cdot \sigma \left( s \cdot \left( f(x_1, \dots, x_n) - \frac{1}{2} \right) \right) - 1 \right) \quad (\text{C.22})$$

(C.23)

Next, we proof several properties of the sigmoidal implication.

**Proposition 11.** For all  $a_1, c_1, a_2, c_2 \in [0, 1]$ ,

1. if  $I(a_1, c_1) < I(a_2, c_2)$ , then also  $\sigma_I(a_1, c_1) < \sigma_I(a_2, c_2)$ ;
2. if  $I(a_1, c_1) = I(a_2, c_2)$ , then also  $\sigma_I(a_1, c_1) = \sigma_I(a_2, c_2)$ .

*Proof.* 1. We note that  $\sigma_I$  can be written as  $\sigma_I(a, c) = w \cdot \sigma(s \cdot (I(a, c) + b_0)) - h$  for constants  $w = \frac{(1 + e^{-s \cdot (1 + b_0)})^2}{e^{-s \cdot b_0} - e^{-s \cdot (1 + b_0)}}$  and  $h = \frac{1 + e^{-s \cdot (1 + b_0)}}{e^{-s \cdot b_0} - e^{-s \cdot (1 + b_0)}}$ . As  $s > 0$ ,  $-s \cdot b_0 > -s \cdot (1 + b_0)$ . Therefore,  $e^{-s \cdot b_0} - e^{-s \cdot (1 + b_0)} > 0$ . Furthermore, as  $e^{-\frac{s}{2}} > 0$  then certainly  $(1 + e^{-\frac{s}{2}})^2 > 0$ . As both  $e^{-s \cdot b_0} - e^{-s \cdot (1 + b_0)} > 0$  and  $(1 + e^{-\frac{s}{2}})^2 > 0$ , then also  $w > 0$ . As  $s > 0$ ,  $s \cdot (I(a_1, c_1) + b_0) < s \cdot (I(a_2, c_2) + b_0)$  as  $I(a_1, c_1) < I(a_2, c_2)$ . Next, note that the sigmoid function  $\sigma$  is a monotonically increasing function. Using  $w > 0$  we find that  $\sigma_I(a_1, c_1) = w \cdot \sigma(s \cdot (I(a_1, c_1) + b_0)) < w \cdot \sigma(s \cdot (I(a_2, c_2) + b_0)) = \sigma_I(a_2, c_2)$ .

2.

$$\sigma_I(a_1, c_1) = w \cdot \sigma(s \cdot (I(a_1, c_1) + b_0)) - h = w \cdot \sigma(s \cdot (I(a_2, c_2) + b_0)) - h = \sigma_I(a_2, c_2)$$

□

**Proposition 12.**  $\sigma_I(a, c)$  is 1 if and only if  $I(a, c) = 1$ . Similarly,  $\sigma_I(a, c)$  is 0 if and only if  $I(a, c) = 0$ .

*Proof.* Assume there is some  $a, c \in [0, 1]$  so that  $I(a, c) = 1$ . By construction,  $\sigma_I(a, c)$  is 1 (see Appendix C.2).

Now assume there is some  $a_1, c_1 \in [0, 1]$  so that  $\sigma_I(a_1, c_1) = 1$ . Now consider some  $a_2, c_2$  so that  $I(a_2, c_2) = 1$ . By the construction of  $\sigma_I$ ,  $\sigma_I(a_2, c_2) = 1$ . For the sake of contradiction assume  $I(a_1, c_1) < 1$ . However, by Proposition 11 as  $I(a_1, c_1) < I(a_2, c_2)$  then  $\sigma_I(a_1, c_1) < \sigma_I(a_2, c_2)$  has to hold. This is in contradiction with  $\sigma_I(a_1, c_1) = \sigma_I(a_2, c_2) = 1$  so the assumption that  $I(a_1, c_1) < 1$  has to be wrong and  $I(a_1, c_1) = 1$ .

The proof for  $I(a, c) = 0$  is analogous.  $\square$

**Proposition 13.** For all fuzzy implications  $I$ ,  $\sigma_I$  is also a fuzzy implication.

*Proof.* By Definition 7  $I(\cdot, c)$  is decreasing and  $I(a, \cdot)$  is increasing. Therefore, by Proposition 11.1,  $\sigma_I(\cdot, c)$  is also decreasing and  $\sigma_I(a, \cdot)$  is also increasing. Furthermore,  $I(0, 0) = 1$ ,  $I(1, 1) = 1$  and  $I(1, 0) = 0$ . We find by Proposition 12 that then also  $\sigma_I(0, 0) = 1$ ,  $\sigma_I(1, 1) = 1$  and  $\sigma_I(1, 0) = 0$ .  $\square$

$I$ -sigmoidal implications only satisfy left-neutrality if  $I$  is left-neutral and  $s$  approaches 0.

**Proposition 14.** If a fuzzy implication  $I$  is contrapositive symmetric with respect to  $N$ , then  $\sigma_I$  also is.

*Proof.* Assume we have an implication  $I$  that is contrapositive symmetric and so for all  $a, c \in [0, 1]$ ,  $I(a, c) = I(N(c), N(a))$ . By Proposition 11.2,  $\sigma_I(a, c) = \sigma_I(N(c), N(a))$ . Thus,  $\sigma_I$  is also contrapositive symmetric with respect to  $N$ .  $\square$

By this proposition, if  $I$  is an S-implication,  $\sigma_I$  is contrapositive symmetric and thus also contrapositive differentiable symmetric.

**Proposition 15.** If  $I$  satisfies the identity principle, then  $\sigma_I$  also satisfies the identity principle.

*Proof.* Assume we have a fuzzy implication  $I$  that satisfies the identity principle. Then  $I(a, a) = 1$  for all  $a$ . By Proposition 12 it holds that  $\sigma_I(a, a)$  is also 1.  $\square$

### Appendix C.3. Nilpotent Aggregator

**Proposition 16.** Equation 2 is equal for the Nilpotent t-norm to

$$A_{T_{nM}}(x_1, \dots, x_n) = \begin{cases} \min(x_1, \dots, x_n), & \text{if } x_i + x_j > 1; x_i \text{ and } x_j \text{ are the two lowest values in } x_1, \dots, x_n \\ 0, & \text{otherwise.} \end{cases} \quad (\text{C.24})$$

*Proof.* We will proof this by induction. Base case: Assume  $n = 2$ . Then  $A_{T_{nM}}(x_1, x_2) = T_{nM}(x_1, x_2)$ .  $x_1$  and  $x_2$  are the two lowest values of  $x_1, x_2$ , so the condition in Equation C.24 would change to  $x_1 + x_2 > 1$ .

Inductive step: We assume Equation C.24 holds for some  $n \geq 2$ . Then by Equation 2

$A_{T_{nM}}(x_1, \dots, x_{n+1}) = T_{nM}(A_{T_{nM}}(x_1, \dots, x_n), x_{n+1})$ . Note that if  $A_{T_{nM}}(x_1, \dots, x_n) = 0$  then  $A_{T_{nM}}(x_1, \dots, x_{n+1})$  is also 0 as  $x_{n+1} \in [0, 1]$  and so  $0 + x_{n+1} > 1$  can never hold. We identify three cases:

1. If  $x_{n+1}$  is the lowest value in  $x_1, \dots, x_{n+1}$ , then  $A_{T_{nM}}(x_1, \dots, x_n)$  is either the second lowest value in  $x_1, \dots, x_{n+1}$  or 0. If it is 0, the sum of the second and third lowest values is not greater than 1, and so the sum of the two lowest values can neither be. If it is not, then  $T_{nM}(A_{T_{nM}}(x_1, \dots, x_n), x_{n+1})$  first compares if  $A_{T_{nM}}(x_1, \dots, x_n) + x_{n+1} > 1$ , that is, if the sum of the two lowest values in  $x_1, \dots, x_{n+1}$  is higher than 1, and returns  $x_{n+1}$  if this holds and 0 otherwise.
2. If  $x_{n+1}$  is the second lowest value in  $x_1, \dots, x_{n+1}$ , then  $A_{T_{nM}}(x_1, \dots, x_n)$  is either the lowest value in  $x_1, \dots, x_{n+1}$  or 0. If it is 0, the sum of the first and third lowest values is not greater than 1, and so the sum of the two lowest values can neither be. If it is not, then  $T_{nM}(A_{T_{nM}}(x_1, \dots, x_n), x_{n+1})$  first compares if  $A_{T_{nM}}(x_1, \dots, x_n) + x_{n+1} > 1$ , that is, if sum of the two lowest values in  $x_1, \dots, x_{n+1}$  is higher than 1, and returns  $A_{T_{nM}}(x_1, \dots, x_n)$  if this holds and 0 otherwise.
3. If  $x_{n+1}$  is neither the lowest nor second lowest value in  $x_1, \dots, x_{n+1}$ , then the sum  $s$  of the two lowest values in  $x_1, \dots, x_n$  is also the sum of the two lowest values in  $x_1, \dots, x_{n+1}$ . If  $A_{T_{nM}}(x_1, \dots, x_n) = 0$ , then  $s$  can not have been greater than 1 and so  $A_{T_{nM}}(x_1, \dots, x_{n+1})$  is also 0. If it is not, then  $s > 1$  and  $A_{T_{nM}}(x_1, \dots, x_n)$  is the lowest value and surely  $A_{T_{nM}}(x_1, \dots, x_n) + x_{n+1} > 1$  as  $x_{n+1}$  is at least as large as the second lowest value.

$\square$

#### Appendix C.4. Yager R-Implication

The Yager t-norm is defined as  $T_Y(a, b) = 1 - ((1-a)^p + (1-b)^p)^{\frac{1}{p}}$ . The Yager R-implication then is defined (see Definition 10) as

$$I_{T_Y}(a, c) = \sup\{b \in [0, 1] | T_Y(a, b) \leq c\} \quad (\text{C.25})$$

When  $a \leq c$ ,  $I_{T_Y} = 1$  as  $T_Y(a, 1) = a \leq c$ . Assuming  $a > c$ , we find by filling in

$$I_{T_Y}(a, c) = \sup\{b \in [0, 1] | 1 - ((1-a)^p + (1-b)^p)^{\frac{1}{p}} \leq c\}, \quad a > c \quad (\text{C.26})$$

To get a closed-form solution of  $I_{T_Y}$  we have to find the largest  $b$  for which  $1 - ((1-a)^p + (1-b)^p)^{\frac{1}{p}} \leq c$ . Solving this inequality for  $b$ , we find

$$\begin{aligned} c &\geq 1 - ((1-a)^p + (1-b)^p)^{\frac{1}{p}} \\ (1-c)^p &\leq (1-a)^p + (1-b)^p \\ 1-b &\geq ((1-c)^p - (1-a)^p)^{\frac{1}{p}} \\ b &\leq 1 - ((1-c)^p - (1-a)^p)^{\frac{1}{p}}. \end{aligned} \quad (\text{C.27})$$

If  $a > c$ , then  $(1-c)^p > (1-a)^p$  and thus  $(1-c)^p - (1-a)^p > 0$ . Furthermore, as  $a, c \in [0, 1]$ ,  $(1-c)^p - (1-a)^p \leq 1$ . Therefore, it has to be true that  $1 - ((1-c)^p - (1-a)^p)^{\frac{1}{p}} \in [0, 1]$ . The largest value  $b \in [0, 1]$  for which the condition holds is then equal to  $1 - ((1-c)^p - (1-a)^p)^{\frac{1}{p}}$  as it is in  $[0, 1]$  and satisfies the inequality.

Combining this with the earlier observation that when  $a \leq c$ ,  $I_{T_Y} = 1$ , we find the following R-implication:

$$I_{T_Y}(a, c) = \begin{cases} 1, & \text{if } a \leq c \\ 1 - ((1-c)^p - (1-a)^p)^{\frac{1}{p}}, & \text{otherwise.} \end{cases} \quad (\text{C.28})$$

We plot  $I_{T_Y}$  for  $p = 2$  in Figure 8. As expected,  $p = 1$  reduces to the Lukasiewicz implication. The derivatives of this implication are

$$d_{I_{T_Y}c}(a, c) = \begin{cases} ((1-c)^p - (1-a)^p)^{\frac{1}{p}-1} \cdot (1-c), & \text{if } a > c \\ 0, & \text{otherwise,} \end{cases} \quad (\text{C.29})$$

$$d_{I_{T_Y}\neg a}(a, c) = \begin{cases} ((1-c)^p - (1-a)^p)^{\frac{1}{p}-1} \cdot (1-a), & \text{if } a > c \\ 0, & \text{otherwise.} \end{cases} \quad (\text{C.30})$$

#### Appendix D. Differentiable Product Fuzzy Logic

We compare Differentiable Product Fuzzy Logic (DPFL), which uses the product t- and t-conorm  $T_P$ ,  $S_P$ , the Reichenbach implication  $I_{RC}$  and the log-product aggregator  $A_{\log T_P}$ , and a probabilistic logic method called Semantic Loss (Xu et al., 2018).

**Definition 25.** Let  $\mathcal{P}$  be a set of predicates,  $O$  the domain of discourse,  $\eta_\theta$  an embedded interpretation of  $\mathcal{L}$  and  $\mathcal{K}$  a knowledge base of background knowledge. The **Semantic Loss** is defined as

$$\mathcal{L}_S(\theta; \mathcal{K}) = -\log \sum_{\mathbf{w} \models \mathcal{K}} \prod_{\mathbf{P}(o_1, \dots, o_m)} \eta_\theta(\mathbf{P})(o_1, \dots, o_m) \prod_{\mathbf{w} \models \neg \mathbf{P}(o_1, \dots, o_m)} (1 - \eta_\theta(\mathbf{P})(o_1, \dots, o_m)), \quad (\text{D.1})$$

where  $\mathbf{w}$  is a **world** (also known as a *Herbrand interpretation*) that assigns a binary truth value to every ground atom and where  $\eta_\theta(\mathbf{P})(o_1, \dots, o_m)$  is the probability of a ground atom.

This computes the logarithm of the sum of probabilities of all worlds for which  $\mathcal{K}$  holds, or the probability of sampling a world that is consistent with  $\mathcal{K}$ . The probability that a ground atom  $\mathbf{P}(o_1, \dots, o_m)$  is true is  $\eta_\theta(\mathbf{P})(o_1, \dots, o_m)$ . The different ground atoms are assumed to be independent. By marginalizing out the world  $\mathbf{w}$ , it can be used for injecting background knowledge in unsupervised or semi-supervised learning. Compared to DPFL, Semantic Loss is exponential in the size of the amount of ground atoms as the sum over valid worlds has to be computed. Equivalent formulas have equal Semantic Loss, and a knowledge base consisting of a conjunction of facts is equal to the cross-entropy loss function.

DPFL is connected in an interesting way to Semantic Loss. It corresponds to a single iteration of the *loopy belief propagation* algorithm (Murphy et al., 2013).

**Proposition 17.** Let  $\varphi$  be a closed formula so that if  $\mathbf{P}_1(o_{11}, \dots, o_{1m})$  and  $\mathbf{P}_2(o_{21}, \dots, o_{2m})$  are both ground atoms appearing in  $\varphi$ , then  $\mathbf{P}_1(o_{11}, \dots, o_{1m}) = \mathbf{P}_2(o_{21}, \dots, o_{2m})$ . Then it holds that  $\mathcal{L}_S(\theta; \varphi) = -e_\theta(\varphi; \emptyset)$  when using  $T = T_P$ ,  $S = S_P$ ,  $I = I_{RC}$  and  $A = A_{\log T_P}$ .

As there are no loops when each ground atom appears uniquely in  $\varphi$ , the factor graph over which loopy belief propagation is done is a tree. As  $e_{\theta}(\varphi; \emptyset)$  corresponds to a single iteration of loopy belief propagation, this is equal to regular belief propagation which is an exact method for computing queries on probabilistic models (Pearl, 1988). Clearly, this condition on  $\varphi$  is very strong. Although loopy belief propagation is known to often be a good approximation empirically (Murphy et al., 2013), the degree to which DPFL approximates Semantic Loss requires further research as this is not a guarantee. However, if DPFL approximates Semantic Loss well, it can be a strong alternative as it is not an exponential computation. However, it also means that most problems of DPFL will also be present in Semantic Loss. For example, if we just have the formula  $\forall \text{raven}(x) \rightarrow \text{black}(x)$ , the grounding of the knowledge base will not contain repeated ground atoms, and thus Semantic Loss and DPFL are equivalent and share difficulties related to the imbalance of modus ponens and modus tollens.

#### Appendix D.1. Proof

In this section, we proof Proposition 17. Without loss of generality we assume that  $\mathcal{K} = \varphi = \forall x_1, \dots, x_n \phi$ . Slightly rewriting Equation D.1, we find that the probability distribution of Semantic Loss is

$$p(\varphi | \eta_{\theta}) = \sum_{\mathbf{w}} p(\varphi | \mathbf{w}) p(\mathbf{w} | \eta_{\theta}) \quad (\text{D.2})$$

where we define the valuation probability  $p(\varphi | \mathbf{w}) = I[\mathbf{w} \models \varphi]$  and the world probability

$$p(\mathbf{w} | \eta_{\theta}) = \prod_{\mathbf{w} \models \mathsf{P}(o_1, \dots, o_k)} \eta_{\theta}(\mathsf{P})(o_1, \dots, o_k) \prod_{\mathbf{w} \models \neg \mathsf{P}(o_1, \dots, o_k)} (1 - \eta_{\theta}(\mathsf{P})(o_1, \dots, o_k)).$$

A Bayesian network is a joint distribution factorized as  $p(x) = \prod_{i=1}^n p(x_i | x_{\{pa(i)\}})$  where  $x_{\{pa(i)\}}$  is the set of random variables that are parents of  $x_i$ . In particular, we are interested in the joint distribution  $p(\varphi, \mathbf{w} | \eta_{\theta})$ . We use the compositional structure of  $\varphi$  to expand  $p(\varphi | \mathbf{w})$ .

Let  $p(\phi | ch(\phi))$  be the probability that  $\phi$ , a subformula of  $\varphi$ , is true according to the binary truth table conditioned on the truth value of the direct subformulas  $ch(\phi)$  of  $\phi$ . For example, if  $\phi = \alpha \wedge \beta$ , then  $p(\phi | \alpha, \beta) = 1$  if  $\alpha$  and  $\beta$  are 1, and otherwise  $p(\phi | \alpha, \beta) = 0$ . For an atomic formula  $\mathsf{P}(o_1, \dots, o_k)$ , we do a lookup in the world  $\mathbf{w}$ ,  $p(\mathsf{P}(o_1, \dots, o_k) | \mathbf{w}_{\mathsf{P}(o_1, \dots, o_k)}) = \mathbf{w}_{\mathsf{P}(o_1, \dots, o_k)}$ . Let  $\Phi$  be the set of all subformulas of  $\varphi$ . We express the joint distribution as

$$p(\Phi, \mathbf{w} | \eta_{\theta}) = p(\mathbf{w} | \eta_{\theta}) \prod_{\phi \in \Phi} p(\phi | ch(\phi)). \quad (\text{D.3})$$

A specific world  $\mathbf{w}$  uniquely determines a single  $\Phi$  so that  $\prod_{\phi \in \Phi} p(\phi | ch(\phi)) = 1$ . Note that the distribution  $p(\varphi | \mathbf{w}) = \prod_{\phi \in \Phi} p(\phi | ch(\phi))$  forms a polytree (or directed tree), as a logical expression is formed as a tree. From this Bayesian network, we define the factor graph over which we do the belief propagation. For brevity, we denote a specific ground atom  $\mathsf{P}(o_1, \dots, o_k)$  as  $\mathsf{P}_O$ .

- There is a variable node  $w_{\mathsf{P}_O}$  for every ground atom  $\mathsf{P}_O$  appearing in the grounding of  $\varphi$ . Additionally, there is a factor node  $f_{w_{\mathsf{P}(o_1, \dots, o_k)}}(w_{\mathsf{P}_O}) = \eta_{\theta}(\mathsf{P})(o_1, \dots, o_k)^{w_{\mathsf{P}_O}} \cdot (1 - \eta_{\theta}(\mathsf{P})(o_1, \dots, o_k))^{1-w_{\mathsf{P}_O}}$ .
- There is a variable node  $\phi$  and a factor node  $f_{\phi}$  for every subformula  $\phi \in \Phi$ .
- For every  $\phi = \mathsf{P}_O$ ,  $\phi \in \Phi$ ,  $f_{\phi}(\phi, w_{\mathsf{P}_O}) = I[\phi = w_{\mathsf{P}_O}]$ .
- For every  $\phi = \neg \alpha$ ,  $\phi \in \Phi$ ,  $f_{\phi}(\phi, \alpha) = I[\phi = 1 - \alpha]$ .
- For every  $\phi = \alpha \wedge \beta$ ,  $\phi \in \Phi$ ,  $f_{\phi}(\phi, \alpha, \beta) = I[\phi = \alpha \cdot \beta]$ .
- Let  $\varphi = \forall x_1, \dots, x_n \phi$  be the top node. Denote the set of all instances of  $\varphi$  is  $M$ . Then  $f_{\varphi}(\varphi, m_1, \dots, m_{|M|}) = I[\varphi = \prod_{m \in M} \alpha_m]$  where  $e_{\mu}$  is the random variable corresponding to the instantiation of  $m$  in  $\phi$ .

We ignore the other connectives as they can be formed from  $\neg$  and  $\wedge$ , both in classical logic as in DPFL. Next, we compute the messages in belief propagation. We start from the world variable nodes  $w_{\mathsf{P}_O}$  and move up through the computation tree to  $\varphi$ . The messages for factors to variables are given as (Bishop, 2006)  $\mu_{f_s \rightarrow x}(x) = \sum_X f_s(x, X) \prod_{y \in ne(f_s) \setminus x} \mu_{y \rightarrow f_s}(y)$  where  $ne(x)$  is the set of neighbours of node  $x$ . The messages for variables to factors are given as  $\mu_{x \rightarrow f_s}(x) = \prod_{l \in ne(x) \setminus f_s} \mu_{f_l \rightarrow x}(x)$ .

- a.  $\mu_{f_{w_{\mathsf{P}_O}} \rightarrow w_{\mathsf{P}_O}}(w_{\mathsf{P}_O}) = \eta_{\theta}(\mathsf{P})(o_1, \dots, o_k)^{w_{\mathsf{P}_O}} \cdot (1 - \eta_{\theta}(\mathsf{P})(o_1, \dots, o_k))^{1-w_{\mathsf{P}_O}}$ , factor to variable for ground atom.

- b.  $\mu_{\mathbf{w}_{P_O} \rightarrow f_\phi}(\mathbf{w}_{P_O}) = \mu_{f_{\mathbf{w}_{P_O}} \rightarrow \mathbf{w}_{P_O}}(\mathbf{w}_{P_O}) \prod_{\psi: \psi = P_O, \phi \neq \psi} \mu_{f_\psi \rightarrow \mathbf{w}_{P_O}} = \mu_{f_{\mathbf{w}_{P_O}} \rightarrow \mathbf{w}_{P_O}}(\mathbf{w}_{P_O})$ , ground atom variable to atomic formula factor  $\phi = P(o_1, \dots, o_k)$ . We here assume that the incoming messages  $\mu_{f_\psi \rightarrow \mathbf{w}_{P_O}}$  from other atomic formulas using ground atom  $P_O$  are initialized with 1. We have not yet, and are not able to, compute these as the graph might contain loops.
- c.  $\mu_{f_\phi \rightarrow \phi}(\phi) = I[\phi = 1]\mu_{\mathbf{w}_{P_O} \rightarrow f_\phi}(1) + I[\phi = 0]\mu_{\mathbf{w}_{P_O} \rightarrow f_\phi}(0) = \eta_{\theta}(P)(o_1, \dots, o_k)^\phi \cdot (1 - \eta_{\theta}(P)(o_1, \dots, o_k))^{1-\phi}$ , factor to variable for atomic formulas.
- d.  $\mu_{\phi \rightarrow f_\alpha}(\alpha) = \mu_{f_\alpha \rightarrow \alpha}(\alpha)$  for subformula variables to factors of other subformulas  $\alpha$ . As  $\phi$  is only used in two factors, this simply passes the downstream message through.
- e.  $\mu_{f_\phi \rightarrow \phi}(\phi) = \mu_{f_\alpha \rightarrow \alpha}(1)^{1-\phi} \cdot \mu_{f_\alpha \rightarrow \alpha}(0)^\phi$  factor to variable for negated subformulas  $\phi = \neg \alpha$ .
- f.  $\mu_{f_\phi \rightarrow \phi}(\phi) = (\mu_{f_\alpha \rightarrow \alpha}(1) \cdot \mu_{f_\beta \rightarrow \beta}(1))^\phi \cdot (\mu_{f_\alpha \rightarrow \alpha}(0) \cdot \mu_{f_\beta \rightarrow \beta}(0) + \mu_{f_\alpha \rightarrow \alpha}(1) \cdot \mu_{f_\beta \rightarrow \beta}(0) + \mu_{f_\alpha \rightarrow \alpha}(0) \cdot \mu_{f_\beta \rightarrow \beta}(1))^{1-\phi}$  factor to variable for conjunctions  $\phi = \alpha \wedge \beta$ .
- g.  $\mu_{f_\varphi \rightarrow \varphi}(\varphi) = (\prod_{m \in M} \mu_{f_{\alpha_m} \rightarrow \alpha_m}(1))^\varphi \cdot \left( \sum_{\alpha_1, \dots, \alpha_{|M|}: \exists i: \alpha_i = 0} \prod_{m \in M} \mu_{f_{\alpha_m} \rightarrow \alpha_m}(\alpha_m) \right)^{1-\varphi}$  for the factor of the universally quantified formula  $\varphi = \forall x_1, \dots, x_n \phi$  where  $\alpha_m$  is the subformula corresponding to the  $m$ 'th instantiation of  $\phi$ . The second term is the sum over all cases so that there is a subformula with truth value 0.

We wish to know what the marginal probability  $p(\varphi = 1 | \eta_{\theta})$  is. A marginal of a variable  $\phi$  in a factor graph is found as  $p(\phi) = \prod_{s \in ne(\phi)} \mu_{f_s \rightarrow x}(\phi)$ . The variable node  $\varphi$  only has the factor node  $f_\varphi$  as a neighbor, so using (g.) we find<sup>19</sup>

$$p(\varphi = 1 | \eta_{\theta}) \approx \prod_{m \in M} \mu_{\alpha_m \rightarrow f_\varphi}(1). \quad (\text{D.4})$$

Next, we use induction to proof that the computation of  $\mu_{f_{\alpha_m} \rightarrow \alpha_m}(1)$  is equal to Differentiable Product Fuzzy Logic.

- Let  $\phi$  be any subformula of  $\alpha_m$  for some instantiation  $m$  of  $\varphi$ . Let  $e_{\theta}$  be the valuation function (Definition 13) with  $T = T_P$  and  $N = N_C$ . We proof that  $\mu_{f_\phi \rightarrow \phi}(1) = e_{\theta}(\phi, m)$  and  $\mu_{f_\phi \rightarrow \phi}(0) = 1 - e_{\theta}(\phi, m)$ .
- Base case  $\phi = P_O$ . By (c.),  $\mu_{f_\phi \rightarrow \phi}(1) = \eta_{\theta}(P)(o_1, \dots, o_k)$  and  $\mu_{f_\phi \rightarrow \phi}(0) = 1 - \eta_{\theta}(P)(o_1, \dots, o_k)$ . By Equation 6<sup>20</sup>,  $e_{\theta}(\phi, m) = \eta_{\theta}(P)(o_1, \dots, o_k)$ .
- Inductive step  $\phi = \neg \alpha$ . By (e.) and using the inductive hypothesis,  $\mu_{f_\phi \rightarrow \phi}(1) = \mu_{f_\alpha \rightarrow \alpha}(0) = 1 - e_{\theta}(\alpha, m)$  and  $\mu_{f_\phi \rightarrow \phi}(0) = \mu_{f_\alpha \rightarrow \alpha}(1) = e_{\theta}(\alpha, m)$ . By Equation 7,  $e_{\theta}(\phi, m) = 1 - e_{\theta}(\alpha, m)$ .
- Inductive step  $\phi = \alpha \wedge \beta$ . By (f.) and using the inductive hypothesis,  $\mu_{f_\phi \rightarrow \phi}(1) = \mu_{f_\alpha \rightarrow \alpha}(1) \cdot \mu_{f_\beta \rightarrow \beta}(1) = e_{\theta}(\alpha, m) \cdot e_{\theta}(\beta, m)$  and  $\mu_{f_\phi \rightarrow \phi}(0) = \mu_{f_\alpha \rightarrow \alpha}(0) \cdot \mu_{f_\beta \rightarrow \beta}(0) + \mu_{f_\alpha \rightarrow \alpha}(1) \cdot \mu_{f_\beta \rightarrow \beta}(0) + \mu_{f_\alpha \rightarrow \alpha}(0) \cdot \mu_{f_\beta \rightarrow \beta}(1) = (1 - e_{\theta}(\alpha, m))(1 - e_{\theta}(\beta, m)) + e_{\theta}(\alpha, m)(1 - e_{\theta}(\beta, m)) + (1 - e_{\theta}(\alpha, m))e_{\theta}(\beta, m) = 1 - e_{\theta}(\alpha, m) \cdot e_{\theta}(\beta, m)$ . By Equation 8 and  $T_P(a, c) = a \cdot c$ ,  $e_{\theta}(\phi, m) = e_{\theta}(\alpha, m) \cdot e_{\theta}(\beta, m)$ .

Using Equation D.4 we then find that  $p(\varphi = 1 | \eta_{\theta}) \approx \prod_{m \in M} e_{\theta}(\phi_m, m)$ , which is equal to the Differentiable Product Fuzzy Logic computation of the universal quantifier in Equation 11.

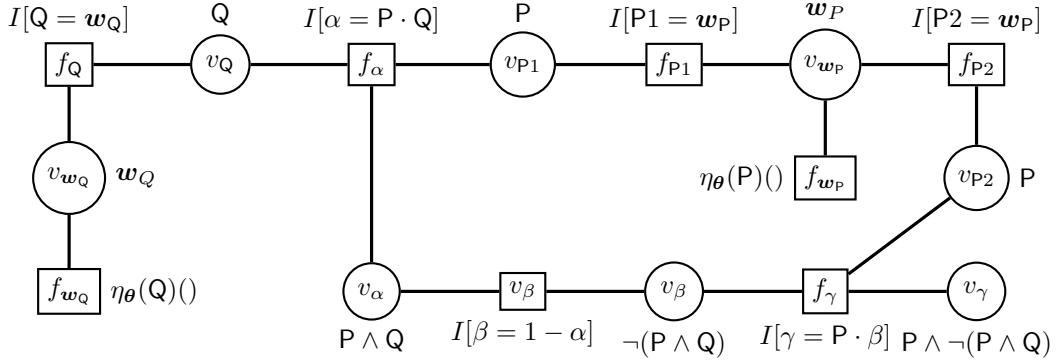
Importantly, as the computation of the logic is itself a tree, the only loops are caused through ground atoms appearing in multiple subformulas. Therefore, when each ground atom only appears in a single formula, Differentiable Product Fuzzy Logic computes the same probability as Semantic Loss.

## Appendix D.2. Example

For example, a formula  $P$  corresponds to a variable node  $P$  with two possible values 1 and 0, along with the factor node with factor  $\eta_{\theta}(P)()$  if  $P$  is *true* and  $1 - \eta_{\theta}(P)()$  otherwise. If we consider the formula  $\gamma = P \wedge \neg(P \wedge Q)$  we find the following factor graph:

<sup>19</sup>We use approximation equality for simplification. Note that this is not necessarily true in loopy belief propagation.

<sup>20</sup>Equation 6 refers to the ungrounded case, but here the full grounding is already done so the lookup function  $l$  is not required.



Here, box nodes correspond to factor nodes and circle nodes correspond to variable nodes. As  $\gamma$  is the top formula, this is where the messages get passed to. Note that there is a single loop, which is present because the atom  $P$  is used twice in the formula. This causes two incorrect messages:  $\mu_{v_{w_P} \rightarrow f_{P1}}$  and  $\mu_{v_{w_P} \rightarrow f_{P2}}$ . The first is incorrect as it does not have access to the incoming message  $\mu_{f_{P2} \rightarrow v_{w_P}}$  and puts it to 1.

With Differentiable Product Fuzzy Logic, we find the expression  $\eta_\theta(P)() \cdot (1 - \eta_\theta(P)() \cdot \eta_\theta(Q)())$ , while the correct probability is where  $P$  is 1 and  $Q$  is 0, that is  $\eta_\theta(P)() \cdot (1 - \eta_\theta(Q)())$ .