Real Logic and Logic Tensor Networks

MSc Thesis *(Afstudeerscriptie)*

written by

Haukur Páll Jónsson
(born July 28th, 1989 in Reykjavík, Iceland)

under the supervision of **Frank van Harmelen** and **Jakub Szymanik**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
August 28, 2018	Ronald de Wolf
	Jaap Kamps
	Miguel Angel Ríos Gaona

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

Abstract

In recent years interest has risen in combining knowledge representation and machine learning and in this thesis we explore Real Logic (RL). RL offers a novel approach to this combination. RL uses first-order logic (FOL) syntax and has a many-valued semantics in which terms are interpreted as real-valued vectors. By making assumptions about the model space and the relation between terms and predicates, we have a well-defined search procedure to search for models to our logical theory in a framework implementing RL called Logic Tensor Networks (LTN).

We evaluate RL and LTN in an empirical setting using the PASCAL-Part dataset and describe the dataset using FOL and then search for a model which satisfies our logical description of the dataset. The task of Semantic Image Interpretation (SII) is used to evaluate RL and compare different instantiations of RL. The goal of SII is to produce a scene graph given an image and prior knowledge about entities in the image. Solutions to this task are expected to take into account the prior knowledge when making predictions based on low-level features. We will model the task using RL and demonstrate that logical constraints improve classification of entities, relation and make predictions more logically consistent. Along the way, we formulate hypotheses about the inner workings of the model and perform experiments to test those hypotheses. The most notable hypotheses are the following four. A model trained with logical constraints will have less variation in performance compared to an unconstrained model. Some instantiations of RL will not work, or work poorly, in a neural network setting. A model trained with logical constraints will produce more logically consistent predictions. The predecessor of the LTN will perform equivalently to the LTN in this setting. We will see from the theoretical results and experimental results that some these hypotheses are incorrect whilst we establish the correctness of the others. We conclude the thesis by summarizing our results and recognize the novel step RL takes in combining knowledge representation and machine learning.

Acknowledgements

I would like to acknowledge my partner Inga Rán and my son Arnar Flóki for their patience and support during the making of this thesis. I could not have done it without their support. Furthermore, I would like to thank my supervisors Frank van Harmelen and Jakub Szymanik for their work in the making of this thesis. I would also like to thank Miguel Angel Ríos Gaona and Jaap Kamps for taking part in the thesis committee and I hope that they enjoy the thesis. Lastly, I would like to acknowledge the help I received from Shuai Wang, Peter Bloem, Erman Acar, Emile van Krieken and Luciano Serafini. The discussions I had with all of you helped make this thesis.

Contents

Introduction

Knowledge Representation and Reasoning is one of the core areas of artificial intelligence. The goal of knowledge representation is to model domain knowledge using a well-defined language along with inference rules to deduce facts. On the other hand, the goal of machine learning is to make predictions based on previous experiences by making assumptions about the underlying process. In both fields, the goal is to infer new knowledge given some input, but the structure of the input and assumptions made to do the inference differ quite a lot. One fundamental difference between the two approaches is the way objects are represented. Symbolic approaches tend to ignore the representation of the objects to be modelled and define symbolic inference to be invariant of representation. Machine learning approaches rely solely on the object's representation when making predictions and one could say that a major factor in recent progress in machine learning is due to advances in learning better representations. Both approaches have enjoyed great success, but in order to solve tasks of increased complexity a combination of both approaches is in order. In particular for problems in which the problem statement is defined in terms of low-level representations and requires reasoning. Statistical Relational Learning (SRL) explores such tasks and tackles complex domains, which need to take into account uncertainty and complex relations between objects.

In this thesis, we will use one such task, the task of Semantic Image Interpretation (SII) as an example to gain a better understanding of this combination. The goal of SII is to output a scene graph, given an image and some background knowledge about objects in the images. We want a solution to this task which is able to process an image from the raw pixel values and produces a graph in which the nodes have labels representing the type of object and labelled edges which describes their relationship in the image. Furthermore, we expect the predicted relations and labels to be consistent with some background knowledge. The labelling and edge-prediction tasks can be performed using machine learning alone, but a problem arises when the predictions are not consistent with the background knowledge. When the solutions are not logically consistent an object labelled as "a cat" and another labelled as "a tail" might be predicted to be in the "part of" relation such that "a cat is a part of a tail", when it should much rather be "a tail is a part of a cat" which we know from common sense knowledge. We would like to reject such predictions and better yet, improve the classifier in these cases.

This brings us to Real Logic (RL) (Serafini and Garcez, 2016; Serafini and d'Avila Garcez, 2016). RL is defined using the syntax of first-order logic (FOL) and has a many-valued semantics in which terms are interpreted in an n-dimensional space and consequently function symbols and predicate symbols as well. The intended application of RL is to describe knowledge about some domain using FOL and then represent the objects of that domain in a real-valued n-dimensional space. By assuming that terms which are represented "close" to one another should have similar truth values, we can make predictions about otherwise unseen representations. By making this assumption, we can view the task of labelling and edge-prediction as knowledge base completion which is based on the representation of the terms. This further implies that all predictions are fundamentally limited by the representation and functions considered. Thus, when evaluating RL we want to make sure that the representation is sufficiently informative, and together with the functions they are able to satisfy our theory. The particular functions considered are made concrete when we introduce the Logic Tensor Networks (LTN), which implement predicates of RL as a neural network. By working under the assumption that the functions considered are differentiable, we will use a search procedure from the gradient descent family.

In this thesis, we will evaluate RL by using the task of SII in two ways. First, we want to know how well RL is able to increase the logical consistency of predictions as well as overall classification performance compared to a baseline model. This will give us an indication whether RL is able to combine learning based on low-level representation and logical reasoning. Secondly, we want to know whether all instantiations of RL perform equally when using a search procedure from the gradient descent family. This will allow us to study the inner workings of RL and offer practical considerations based on theoretical and experimental evidence. Along the way, we formulate a number of hypotheses about the behaviour of RL and perform experiments to test them. This thesis is based on the works of Serafini et al. (2017) and Donadello et al. (2017) who show that RL using LTN is able to increase classification performance and performs well when trained on noisy data in the task of SII. In this thesis, we replicate these experiments and put forward other hypotheses. We confirm the original results but observe that the model does not learn in the way initially anticipated, the performance is somewhat unstable and some adjustments needed to be made to the structure of the model to achieve the same results. We start by making some observations how different implementations of RL will propagate the model's error and quickly see that some implementations will never work while other's might. After viewing the experimental evidence and discussing the results we see conclusively that, out of the models we consider, a single class of models performs best. We experiment to see if the model is progressively able to satisfy the constraints and if more constraints make the predictions more logically consistent. We will see that the model does make the predictions more logically consistent and discuss how the model is able to make the predictions better. We hypothesize that adding more constraints will decrease the variation in the performance of the model, and when we observe the opposite, we try to understand the origin of the variation. We do not explain the variation conclusively and only give a plausible explanation. Lastly, we compare the expressivity of the LTN to its predecessor. We quickly see when we compare the models that the LTN will probably outperform its predecessor in this setting and verify it during experimentation.

We start by setting the stage by defining all relevant concepts in section 1. In section 1.1 we define first-order logic, the notion of satisfiability and knowledge bases. In section 1.2 we define t-norms which generalise classical conjunction, its dual the s-norm, aggregation functions which serve as semantics for the universal quantifier. We also explore the partial derivatives of these operators and consider implementation issues. In section 1.3 we define the RL framework, its semantics and the maximum satisfiability problem. In section 1.4 we define the LTN and the Neural Tensor Network (NTN) as possible implementations for the predicates of RL and derive the gradient of a sentence's generic grounding.

In section 2 we describe how RL is used in the task of SII and how it will be evaluated. In section 2.1 we describe the task and dataset on which all evaluations are based on. In section 2.2 we define FOL language $\mathcal{L}$ which defines using the dataset as terms in the task of SII and present different implementations of RL which will be considered. In section 2.3 we describe how we will evaluate the performance of the model as we are interested in the performances of the classification task, relation prediction as well as the logical consistency of predictions. In section 2.4 we state our experimental hypotheses and motivate them.

In section 3 we will go through the results of each experimental hypothesis and in section 4 we will contemplate the results of the experiments and try to explain them.

In section 5 we conclude the thesis and summarize the results.

We conclude this thesis by discussing some interesting future directions of research based on our observations and issues encountered and recognize the novel step RL takes in combining knowledge representation and machine learning allowing us to, intuitively, describe data which have complex relations and incorporate uncertainty.

Chapter 1

Theory

In this section, we define all relevant concepts required for different parts of the thesis. In section 1.1, we start with some fundamental concepts in logic, namely the language and satisfiability. In section1.2, we move to many-valued extensions of conjunction and universal quantifiers. In section 1.3 RL is defined through the concept of a grounding, which is essentially an interpretation in which satisfiability is defined over the real field, in the range [0,1] and subsets of closed terms of the language are considered when evaluating the truth of universal quantification. In section 1.4 we make assumptions about the space of possible groundings, introduce the LTN, demonstrate how the gradient of the network is computed, make observations based on the structure of the gradient and compare with the operators defined in section 1.2.

1.1 First Order Logic

Let us start by introducing FOL syntax.

Definition 1 (Vocabulary of FOL $\mathcal{L}$)**.** *The vocabulary of First-order logic language $\mathcal{L}$ consists of* variables $x, y, z, \dots,$ logical constants $\neg, \vee, \wedge, \Rightarrow, \forall, \exists$, non-logical constants *which consist of* individual constants $a, b, c, \dots,$ predicates $P, R, \dots$ *and* function symbols $f, \dots$ *and other* auxiliary symbols.

We let $\mathcal{P}$ be the set of all predicates, $\mathcal{F}$ be the set of all function symbols and $\mathcal{C}$ be the set of all constants for language $\mathcal{L}$. Each $P \in \mathcal{P}$ and each $f \in \mathcal{F}$ is assigned a natural number, the *arity* of the function α, $\alpha : \mathcal{P} \cup \mathcal{F} \to \mathbb{N}$. We write $\alpha(P) = n$ to denote that the arity of P is n. We note that function symbols of arity 0 can be used instead of the individual constants but we refer to the constants $\mathcal{C}$. We further assume that $\mathcal{P}$ and $\mathcal{F}$ are disjoint. The tuple $\langle \mathcal{F}, \mathcal{P}, \sigma \rangle$ is called the *signature* of $\mathcal{L}$.

Definition 2 (Terms of $\mathcal{L}$)**.** *The terms of FOL $\mathcal{L}$ are*

- *The individual variables.*
- *The individual constants.*
- *If f is a function symbol of arity n and $t_1, \dots, t_n$ are terms, then $f(t_1, \dots, t_n)$ is a term.*
- *Nothing else is a term.*

We refer to *closed terms* as terms which do not contain variables and denote them as $\overline{terms(\mathcal{L})}$.

Definition 3 (Formulas of $\mathcal{L}$)**.** *We define formulas for $\mathcal{L}$ inductively.*

- *If $P \in \mathcal{P}$ has arity n and $t_1, \dots, t_n$ are terms then $P(t_1, \dots, t_n)$ is a formula, also called an* atom *or* atomic formula*.*
- *If φ is a formula then $\neg\varphi$ is a formula.*
- *If φ and ψ are formulas then $\varphi \vee \psi$ and $\varphi \wedge \psi$ are formulas.*
- *If φ is a formula which has at least one occurrence of variable x and does not contain $\forall x$ or $\exists x$ already, then $\forall x\varphi$ and $\exists x\varphi$ are formulas.*
- *Nothing else is a formula.*

We let $\varphi \Rightarrow \psi$ be a short-hand for $\neg\varphi \vee \psi$ and use $\varphi(t_1, \dots, t_n)$ to denote a formula defined over terms $t_1, \dots, t_n$.

Definition 4 (Literal of $\mathcal{L}$)**.** *Let φ be an atom of $\mathcal{L}$ then φ is a* literal *and $\neg\varphi$ is a* literal*.*

Definition 5 (Clause of $\mathcal{L}$)**.** *Let $\varphi_1, \dots, \varphi_n$ be literals of $\mathcal{L}$ then $\varphi_1 \vee \cdots \vee \varphi_n$ is a clause of $\mathcal{L}$.*

Using this definition we can create formulas of the form $\forall x P(y) \wedge R(x, y)$ which leads to the definition of bound and free variables and the corresponding closed and open formulas of $\mathcal{L}$.

Definition 6 (Bound and free variables of $\mathcal{L}$)**.** *A variable x is* bound *iff it occurs in φ of the form $\forall x\varphi$. A variable x is* free *if it is not bound.*

Definition 7 (Closed and open formulas of $\mathcal{L}$)**.** *A formula in which no variables occur free is a* closed *formula, also called a* sentence*. Otherwise, the formula is* open*.*

These definitions offer quite a range of writing FOL formulas and in later sections, we want to limit our discussion to sentences which have a particular structure, therefore we introduce three normal forms.

Definition 8 (Conjunctive Normal Form (CNF))**.** *A formula of $\mathcal{L}$ is in CNF if it is a conjunction of clauses.*

Definition 9 (Prenex Normal Form (PNF))**.** *A formula of $\mathcal{L}$ is in PNF if all the quantifiers of the formula occur in a sequence at the beginning of the formula.*

Definition 10 (Skolem Normal Form (SNF))**.** *A sentence of $\mathcal{L}$ is in SNF if it is in PNF with only universal quantifiers.*

From now on, when we say a formula in its normal form, it is in CNF, PNF and SNF. By assuming that a sentence is in a normal form, we know the sentence's structure, which is important when we explore the gradient of a sentence. Thus, we want to be sure that when a sentence is expressed we can alter it to a normal form without affecting the satisfiability of the sentence, allowing us to look for a model for sentence in a normal form. First, we need to define what satisfiability is and it relies on the notion of an interpretation.

Definition 11 (Interpretation for FOL). *An* interpretation, *$\mathcal{I}$, for FOL is a pair $\langle \mathcal{D}, \mathcal{V} \rangle$, s.t. $\mathcal{D}$ is a non-empty set of objects called the* domain *and $\mathcal{V}$ a* valuation function *which assigns objects from $\mathcal{D}$ to individual constants and a set of ordered n-tuples of objects to predicates.*

An interpretation is only defined in terms of the signature of the $\mathcal{L}$ and if we were to define truth of a sentence only based on the signature we would not be able to talk about the truth of a sentence which includes variables. This is addressed using the β-variant of interpretation $\mathcal{I}$.

Definition 12 (β-variant of interpretation $\mathcal{I}$). *Let β be an individual constant and $\mathcal{I}$ and interpretation which assigns some element to β, then $\mathcal{I}$* is a β-variant of interpretation $\mathcal{I}$ iff $\mathcal{I}$* and $\mathcal{I}$ differ only (if at all) in the assignment of β.*

For formula φ, variable x, and constant c we denote $\varphi[c/x]$ as the the formula in which all occurrences of x have been replaced by c. We now have everything in place to define truth in an interpretation of FOL.

Definition 13 (Truth in an interpretation). *Let $\mathcal{I}$ be an interpretation for $\mathcal{L}$, we define truth in interpretation $\mathcal{I}$ as*

- *If P is a predicate of arity n and $c_1, \ldots, c_n$ are individual constants then $P(c_1, \ldots, c_n)$ is true in $\mathcal{I}$ iff $\langle \mathcal{V}(c_1), \ldots, \mathcal{V}(c_n) \rangle \in \mathcal{V}(P)$.*
- *φ is true in $\mathcal{I}$ iff $\neg\varphi$ is not true in $\mathcal{I}$.*
- *$\varphi \wedge \psi$ is true in $\mathcal{I}$ iff φ and ψ are true in $\mathcal{I}$.*
- *$\varphi \vee \psi$ is true in $\mathcal{I}$ iff φ or ψ are true in $\mathcal{I}$.*
- *$\exists x \varphi$ is true in $\mathcal{I}$ iff $\varphi[c/x]$ is true in some β-variant of of $\mathcal{I}$.*
- *$\forall x \varphi$ is true in $\mathcal{I}$ iff $\varphi[c/x]$ is true in every β-variant of of $\mathcal{I}$.*

Definition 14 (Logical consequence). *Let Γ be a set of sentences (possibly empty) and φ a sentence of $\mathcal{L}$ then φ is said to be a* logical consequence *of Γ iff whenever Γ is true then φ is true. We denote this with $\Gamma \vDash \varphi$.*

When talking about particular interpretations, the following two notions are helpful.

Definition 15 (A model). *Let Γ be a set of sentences of $\mathcal{L}$ and $\mathcal{I}$ be an interpretation s.t. for all $\varphi \in \Gamma$ φ is true in $\mathcal{I}$ then $\mathcal{I}$ is a* model *of Γ, denoted as $\mathcal{M}$.*

This leads to the definition of *satisfiability* of a set of sentences.

Definition 16 (Satisfiability). *Let Γ be a set of sentences of $\mathcal{L}$. If there exists a model for Γ then Γ is satisfiable.*

Now we come back to the normal forms.

Proposition 1 (Normal forms preserve satisfiability). *A formula φ of $\mathcal{L}$ in some theory T not in SNF or CNF can be converted to φ' in CNF and SNF s.t. φ' is in language $\mathcal{L}$' which has been extended with a new function symbol in the process of* skolemization *and if $\mathcal{M} \vDash \varphi$ then $\mathcal{M}' \vDash \varphi'$ where $\mathcal{M}'$ is a model of the conservative extension of T with model $\mathcal{M}$.*

This proposition is a bit out of the scope of this thesis but let us briefly describe why we need to extend the language and therefore consider these extra caveats. In the process of *skolemization* we replace existential quantifiers with new function symbols. Let us assume that we have a sentence $\forall x \forall y \exists z \varphi(x, y, z)$. When we read this sentence we read it such that for every x and given this x for all y there exists some z such that φ. The existence of z is dependent on the interpretation of x and y. So when we want to replace the existential quantifier and substitute it with a new function symbol, the function must depend on x and y.

Proof. We first transform φ to CNF. van Dalen (2004) (1.3.9) proves that such a transformation preserves satisfiability. We can then transform the CNF sentence to PNF. van Dalen (2004) (2.5.11) proves that such a transformation preserves satisfiability. Lastly, we can transform the PNF sentence to SNF through the process of *skolemization*. In this process the language $\mathcal{L}$ is extended with a new function symbol for each existential quantifier in the formula, resulting in φ'. In van Dalen (2004) (3.4.4) it is proved if $T \vdash \varphi$ then the (Skolem) extension of T is conservative and for every model of T there is a Skolem expansion $\mathcal{M}'$. $\square$

We could also have written $\varphi' \vDash \varphi$, which is roughly the same. This proposition implies that instead of searching for models of φ we can equally search for models of φ', which we will do.

But let us now transform a sentence to CNF, PNF and SNF.

Example 1 (A sentence converted to normal form)**.**

$$\forall x (P(x) \Rightarrow \exists y R(x, y))$$

$$\forall x (\neg P(x) \lor \exists y R(x, y)) \ \textit{(CNF)}$$

$$\forall x \exists y (\neg P(x) \lor R(x, y)) \ \textit{(PNF)}$$

$$\forall x (\neg P(x) \lor R(x, f(x))) \ \textit{(SNF)}$$

$$\neg P(x) \lor R(x, f(x)) \ \textit{(short-hand)}$$

Indeed, in the last step the $\forall$ quantifier can be omitted since all variables are bound and thus implicitly universally quantified. We will therefore consider all variables bound for the rest of the thesis.

We have now defined all the required FOL concepts. In later sections, we will consider a *knowledge base KB*, which is a consistent set of sentences in SNF and CNF which describes knowledge in FOL and we look to extend it consistently. We will extend KB semantically rather than syntactically. That is, we start with some model which should satisfy our knowledge base and ask whether this model makes φ true, i.e. we check if $KB \vDash \varphi$ If φ is true, we add it to our set, if not, the negation is added to KB. One of the problems we will encounter is that finding a model for KB in the first place is hard, which should be no surprise. To allow a systematic search procedure through the model space, the model space is reduced by making assumptions about the domains and valuation functions considered.

In the next subsection, we will define many-valued extensions of conjunction, disjunction and the universal quantifier as in RL we do not consider truth to be binary but rather in the interval $[0, 1]$.

1.2 Many-valued Operators

The truth of sentences in some natural language is not necessarily binary, in the sense that they are either true or false, but rather they might have varying degrees of truth. For example, assume that we have three people Bob, John and George. Bob is 180cm tall, George is 200cm tall and John is 160cm. We can attempt to model facts about these three people using FOL, one constant for each person and a single predicate for the property of being tall. Thus we might consider $Tall(George)$ to be true and $Tall(John)$ to be false. But where would be place Bob? In this example we might consider putting Bob between John and George, thus using more values for truth than just two.

Fuzzy logic considers these generalizations to the truth values, in particular, truth values in the range $[0, 1]$ which we will use when RL is defined(Bergmann, 2008; Novák, 1987). The continuity of the range is important, as later on it allows us to consider functions which are differentiable over this range. We will now define the generalizations for conjunction and disjunction which was developed along with many-valued logics but we will deviate from the conventional approach of fuzzy logic when it comes to universal quantification and rather use *aggregation functions*. The motivation here is that fuzzy logics are very strict when it comes to the truth value of universal quantification, considering the greatest lower bound over all β-variants(Bergmann, 2008). So if one β-variant has a truth value of 0, the whole sentence has a truth value of 0. The RL framework is not this strict, and we will see that it is beneficial when computing the gradient but the cost is that the existential quantifier will have the same meaning as the universal quantifier. We are not too concerned with that, as we have already assumed that all sentences are in SNF which does not contain existential quantification.

We will now start with the generalization for conjunction and disjunction for truth values in the range of $[0, 1]$. Triangular norms, or simply t-norms, are binary operations T on the interval $[0, 1]$ which satisfy the following conditions.

Definition 17 (T-norm)**.** *A* triangular norm *or* t-norm *is a binary operation.* $T : [0, 1] \times [0, 1] \to [0, 1]$ *which satisfies the following conditions.*

- $T(x, y) = T(y, x)$ *(commutativity)*
- $T(x, T(y, z)) = T(T(x, y), z)$ *(associativity)*
- $y \leq z \Rightarrow T(x, y) \leq T(x, z)$ *(monotonicity)*
- $T(x, 1) = x$ *(neutral element 1)*

T-norms serve to generalize the classical conjunction.

Example 2 (Examples of t-norms)**.** *See figure 1.1 for a contour plot of these functions.*

- *Gödel t-norm:* $T_G(x, y) = \min(x, y)$
- *Łukasiewicz t-norm:* $T_Ł(x, y) = \max(x + y - 1, 0)$
- *Product t-norm:* $T_P(x, y) = x \cdot y$

Due to associativity and commutativity T-norms can be extended to an n-ary operation Klement et al. (2004), $n \in \mathbb{N}$. We denote the extended t-norms with $T(x_1, \ldots, x_n)$.

Example 3 (n-ary t-norms)**.** *We extend the t-norms from our previous example.*

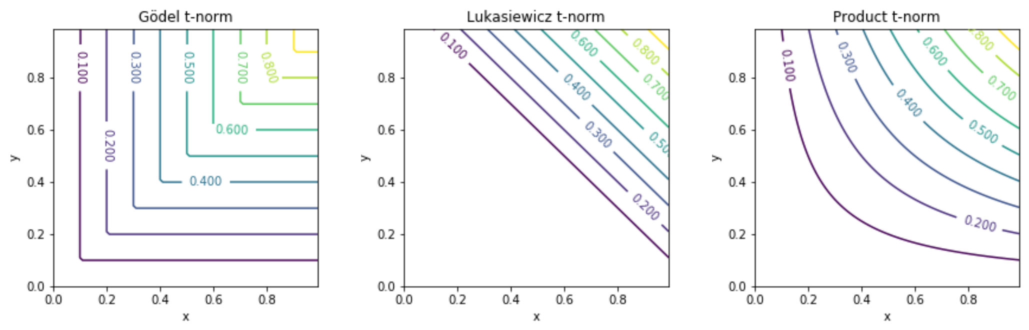


Figure 1.1: The three t-norms; Gödel, Łukasiewicz and Product t-norm and their values along $[0,1] \times [0,1]$. Below the diagonal of the Łukasiewicz it takes the value 0.

- *Gödel t-norm:* $T_G(x_1, \ldots, x_n) = \min(x_1, \ldots, x_n)$
- *Lukasiewicz t-norm:* $T_L(x_1, \ldots, x_n) = \max(\sum_{i=1}^n x_i - (n-1), 0)$
- *Product t-norm:* $T_P(x_1, \ldots, x_n) = \prod_{i=1}^n x_i$

The reader can verify that these norms are indeed generalizations of classical conjunction by checking that they result in the same values as the classical operators. At the same time, they all behave quite differently on the same inputs.

Example 4 (Examples of t-norm computations). *Let* $x_1 = 0.3, x_2 = 0.5, x_3 = 0.3, x_4 = 0.1.$

$$T_G(x_1, x_2, x_3, x_4) = \min(0.3, 0.5, 0.3, 0.1) = 0.1$$

$$T_L(x_1, x_2, x_3, x_4) = \max(0.3 + 0.5 + 0.3 + 0.1 - n + 1, 0) = 0$$

$$T_P(x_1, x_2, x_3, x_4) = 0.3 * 0.5 * 0.3 * 0.1 = 0.0045$$

We notice that the Łukasiewicz t-norm might often result in 0 when n is large. For example, with only two values, the average value needs to be greater than half to result in a positive number. We will make this observation concrete in section 1.4. We also observe that the product t-norm might cause issues in numerical computations, as computers do not handle multiplication of small numbers well. We will see that this is indeed an issue in hypothesis 1.1 but we will also present a solution to this in section 1.4 and follow up with hypothesis 7.

We now define the dual of a t-norm, a t-co-norm, also called s-norm[1]. Similarly, s-norms generalize the classical disjunction.

Definition 18 (S-norm). *If T is a t-norm then S is an s-norm.*

$$S(x, y) = 1 - T(1 - x, 1 - y)$$

Example 5 (Examples of s-norms). *We give the duals of the previous examples*

- *Gödel s-norm:* $S_G(x, y) = \max(x, y)$
- *Łukasiewicz s-norm:* $S_L(x, y) = \min(x + y, 1)$

[1]We will use s-norm instead of t-co-norm in the following discussion as it makes for an easier notation.

- *Product s-norm:* $S_P(x, y) = x + y - xy$

We also extend the s-norms to accept n inputs and denote it as $S(x_n, \dots, x_1)$.

Example 6 (N-ary s-norms). *We extend the s-norms from our previous example.*

- *Gödel s-norm:* $S_G(x_1, \dots, x_n) = \max(x_1, \dots, x_n)$
- *Łukasiewicz s-norm:* $S_L(x_1, \dots, x_n) = \min(\sum_{i=1}^{n} x_i, 1)$
- *Product s-norm:* $S_P(x_1, \dots, x_n) = 1 - \prod_{i=1}^{n}(1 - x_i)$

Let us now explore how these s-norms work in computation.

Example 7 (Examples of t-norm computations). *Let* $x_1 = 0.3, x_2 = 0.5, x_3 = 0.3, x_4 = 0.1$.

$$S_G(x_1, x_2, x_3, x_4) = \max(0.3, 0.5, 0.3, 0.1) = 0.5$$

$$S_L(x_1, x_2, x_3, x_4) = \min(0.3 + 0.5 + 0.3 + 0.1, 1) = 0.9$$

$$S_P(x_1, x_2, x_3, x_4) = 1 - (0.7 * 0.5 * 0.7 * 0.9) = 0.7795$$

Again, we see that the Łukasiewicz s-norm behaves quite strangely, we can almost trivially satisfy it just by adding enough $x_i > 0$. We will make this observation more concrete in section 1.4.

We have now introduced generalizations of conjunction and disjunction for truth values in the range $[0, 1]$ and now we will introduce aggregation functions, which are used for universal quantification in RL. The motivation to use the more generic aggregation functions instead of the standard fuzzy logic universal generalization is because they are simply too strict in the sense that a single β-variant can drag the truth value of the whole sentence down. We look for a softer approach than this. As an example, consider a sentence stating "all swans are white". We all know that this is sentence is false, as in fact "most swans are white". But stating it to be false seems to ignore a lot of the evidence that went into the statement. One could also consider the universal quantifier to be simply "most" and thus the sentence would be true, but then the question arises, where to draw the line? What is "most"? Instead, RL attempts to avoid this problem by considering generic aggregation functions[2] so that one can choose based on a modelling scenario.

Definition 19 (Aggregation function). *An* aggregation function A *is a function* $A : [0, 1]^n \to [0, 1]$ *s.t.*

- $A(\mathbf{0}) = 0$.
- $A(\mathbf{1}) = 1$.

If $T = \{x_1, \dots, x_n\}$ then we denote $A(T)$ to mean $A(x_1, \dots, x_n)$ where we have ordered the elements in some arbitrary order. One can think of aggregation functions which behave differently w.r.t. the order of inputs, and we consider those later on, but then we assume that the function itself orders the inputs.

[2] In fact, they are so generic in the original proposal that they consider all functions from $[0, 1]^n \to [0, 1]$. We will deviate a bit from the original proposal (Serafini and d'Avila Garcez, 2016) here and present aggregation functions which better capture the intended meaning.

Example 8 (Examples of aggregation functions)**.** *We give a few examples of aggregation functions which capture the intended definition.*

- *Arithmetic mean:* $A_{ari}(x_1, \dots, x_n) = \frac{\sum_{i=1}^n x_i}{n}$
- *Harmonic mean:* $A_{har}(x_1, \dots, x_n) = \left(\frac{\sum_{i=1}^n x_i^{-1}}{n}\right)^{-1} = \frac{n}{\sum_{i=1}^n x_i^{-1}}$
- *Minimum:* $A_{min}(x_1, \dots, x_n) = \min(x_1, \dots, x_n)$
- *Maximum:* $A_{max}(x_1, \dots, x_n) = \max(x_1, \dots, x_n)$

Out of the example aggregation functions the A_{min} is the only candidate which is considered as in fuzzy logic (Bergmann, 2008).

We will also consider a different class of aggregation operators known as the Ordered Weighted Average (OWA). The OWA operator is an important aggregation function commonly used in multi-criteria decision making, see Yager (1993).

Definition 20 (Ordered weighted average function (OWA))**.** *An* ordered weighted average function A *is an aggregate function s.t.*

$$A_{owa}(x_1, \dots, x_n) = \sum_{i=1}^{n} w_i x_{\pi(i)}$$

where π *is a permutation from* $[n]$ *to* $[n]$ *s.t.* $x_{\pi(i)}$ *is the i-th largest element of* $x_1, \dots, x_n$ *and* w_i *are the weights of the operator s.t.* $\sum_{i=1}^n w_i = 1$.

We use $[n]$ to denote to the set $\{1, 2, \dots, n\}$. Take note that for $w_1 = 1$ and $w_i = 0$ for all $i \neq 1$ then $A_{owa}(x_1, \dots, x_n) = A_{max}(x_1, \dots, x_n)$. Similarly, for $w_n = 1$ and $w_i = 0$ for all $i \neq n$ then $A_{owa}(x_1, \dots, x_n) = A_{min}(x_1, \dots, x_n)$ and for $w_i = 1/n$ then $A_{owa}(x_1, \dots, x_n) = A_{ari}(x_1, \dots, x_n)$.

We have now defined the many-valued operators we will consider for RL. The t-norms generalize the classical conjunction and s-norms generalize the classical disjunction. The treatment of many-valued universal quantification as aggregation functions is proposed as a more general approach than the standard fuzzy universal quantifiers, but we will show in the next section that this comes with a cost. Furthermore, in section 1.4, we will make concrete the limitations of the Łukasiewicz norms and analyse the partial derivative of the norms and aggregation functions. Now everything is in place to define RL.

1.3 Real Logic

RL is defined using the syntax of FOL and semantics in which terms are interpreted in an n-dimensional space and in turn function symbols and predicate symbols are defined over the same domain with predicates taking truth values in the range [0,1]. The intended use case for RL is to describe knowledge about the domain in which elements can be described by their numerical features. RL proposes a novel way to combine logical expressions and numerical features by leveraging ideas from fuzzy logic in addition to new semantics for universal quantification. RL was originally proposed by Serafini and Garcez (2016); Serafini and d'Avila Garcez (2016) and they simultaneously propose LTN to implement RL. Shortly after the original proposal, two other papers demonstrate RL in action Donadello et al. (2017); Serafini et al. (2017) and we plan to replicate their findings in this thesis.

RL assumes that there is an underlying domain of objects $O = \{o_1, o_2, \dots\}$, possibly infinite. The objects are meant to be real-world objects which can be represented by a vector of real values. The mapping from the set of objects to the domain of representation is called a grounding, denoted by $\mathcal{G}$. Thus, $\mathcal{G}(o_i)$ is the vector representation of object o_i. The assumption is made that the representation of these objects preserves some latent structure between the numerical properties and the relations defined on $O^{\alpha(R_i)}$, where R_i is some relation on O with arity $\alpha(R_i)$. The purpose of this formalism is to infer knowledge about the relational structure of O as well as to predict the numerical properties of $\mathcal{G}(o_i)$, based on the latent numerical properties and knowledge about O (Serafini and Garcez, 2016).

To explain the intuition behind a grounding let us assume that our objects are John, George and Bob, as we did before. We would like to know whether George is taller than Bob and if we were to represent Bob and George by their height, i.e. $\mathcal{G}(Bob) = (height_{Bob})$ and $\mathcal{G}(George) = (height_{George})$ and we want to know, $isTaller(George, Bob)$ where $isTaller$ is a binary predicate. Recall that Bob is 180cm tall, George is 200cm tall and John is 160cm. If our grounding for the $isTaller$ predicate accurately takes their height into account when computing the truth value of $isTaller(George, Bob)$ we should be able to decide this binary predicate based on the representation of Bob and George. On the other hand, if we were to represent Bob and George by something else than their height, for example, by how many friends they have and hair color we would not expect to be able to decide whether George is taller than Bob because we do not expect this grounding to preserve the latent structure between the numerical properties and the relation considered. Let us now assume that we do not know the height of John, but we do know that $Tall(John) = 0.2$, $Tall(Bob) = 0.6$ and $Tall(George) = 0.9$. Using the representation of "tallness" we should also be able to decide whether $isTaller(George, Bob)$ and $isTaller(Bob, John)$, if we further ensure that the $isTaller$ predicate is transitive we should also be able to infer that $isTaller(George, John)$ without ever needing to observe it. In the experimental setting, we consider these "logical representations" of the objects so it is good to keep this example in mind. Lastly, if we use this representation we could also infer something about John's actual height, based on $Tall(John) = 0.2$, George's and Bob's height, $Tall(Bob) = 0.6$ and $Tall(George) = 0.9$.

We will now define a grounding, which is essentially an interpretation which assumes that the domain is $\mathbb{R}^n$ and that predicates map to truth values in the range $[0, 1]$.

Definition 21 (Grounding)**.** *A grounding $\mathcal{G}$ of the signature of $\mathcal{L}$ is a function s.t.*

- $\mathcal{G}(c) \in \mathbb{R}^n$, *for every* $c \in \mathcal{C}$
- $\mathcal{G}(f) \in \mathbb{R}^{n \cdot \alpha(f)} \to \mathbb{R}^n$ *for every* $f \in \mathcal{F}$
- $\mathcal{G}(P) \in \mathbb{R}^{n \cdot \alpha(P)} \to [0, 1]$ *for every* $P \in \mathcal{P}$

We inductively extend the definition over sentences φ of $\mathcal{L}$ over $t_1, \dots, t_m \in \overline{terms(\mathcal{L})}$, the closed terms of $\mathcal{L}$.

- $\mathcal{G}(f(t_1, \dots, t_m)) = \mathcal{G}(f)(\mathcal{G}(t_1), \dots, \mathcal{G}(t_m))$
- $\mathcal{G}(P(t_1, \dots, t_m)) = \mathcal{G}(P)(\mathcal{G}(t_1), \dots, \mathcal{G}(t_m))$
- $\mathcal{G}(\neg P(t_1, \dots, t_m)) = 1 - \mathcal{G}(P(t_1, \dots, t_m))$

- $\mathcal{G}(\varphi_1 \wedge \cdots \wedge \varphi_n) = T(\mathcal{G}(\varphi_1), \ldots, \mathcal{G}(\varphi_n))$
- $\mathcal{G}(\varphi_1 \vee \cdots \vee \varphi_n) = S(\mathcal{G}(\varphi_1), \ldots, \mathcal{G}(\varphi_n))$

Where T is a t-norm and S the corresponding s-norm. As mentioned before, the semantics of universal quantification deviates from the normal fuzzy logic semantics and is given in terms of *aggregation semantics.* Aggregation semantics consider more generic functions and not all β-variants need to be considered when evaluating the universal quantification.

Definition 22 (Aggregation semantics)**.** *Let $\forall x_1, \ldots, x_n \varphi(x_1, \ldots, x_n)$ be a sentence of $\mathcal{L}$ with n variables*[3]*, $T_1, \ldots, T_n \subseteq \overline{terms(\mathcal{L})}$ and A is an aggregation operator from $[0,1]^{|T_1|\cdot\cdots\cdot|T_n|} \rightarrow [0,1]$ then the aggregated truth value of $\varphi(x_1, \ldots, x_n)$ over $T_1 \times \cdots \times T_n$ is*

- $\mathcal{G}(\forall x_1, \ldots, x_n \varphi(x_1, \ldots, x_n)) = A(\{\mathcal{G}(\varphi(t_{x_1}, \ldots, t_{x_n})) \mid (t_{x_1}, \ldots, t_{x_n}) \in T_1 \times \cdots \times T_n\})$

We now denote $\forall x_1, \ldots, x_n \varphi(x_1, \ldots, x_n)$ as $\varphi(\boldsymbol{x})$ where $\boldsymbol{x}$ is a vector of variables. We now refer to the choice of aggregation operator, t-norm and corresponding s-norm operator as an instantiation *of RL.*

Before continuing, let us carefully consider what is going on and start with an example.

Example 9 (Computing aggregation)**.** *Let us consider an aggregation using the arithmetic mean and $\overline{terms(\mathcal{L})} = \{a, b, c, d, e\}$ and some sentence φ with two variables.*

Let $\mathcal{G}(\forall x_1 x_2 \varphi(x_1, x_2)) = A_{ari}(\{\mathcal{G}(\varphi(x_1, x_2)) \mid (t_{x_1}, t_{x_2}) \in T_1 \times T_2\})$ with $T_1 = \{a, b, c\}$ and $T_2 = \{a, c, d, e\}$.

Thus, $A_{ari}(\{\mathcal{G}(\varphi(x_1, x_2)) \mid (t_{x_1}, t_{x_2}) \in T_1 \times T_2\}) = A_{ari}(\mathcal{G}(\varphi(a, a)), \mathcal{G}(\varphi(a, c)), \mathcal{G}(\varphi(a, d)), \mathcal{G}(\varphi(a, e)), \mathcal{G}(\varphi(b, a)), \mathcal{G}(\varphi(b, c)), \mathcal{G}(\varphi(b, d)), \mathcal{G}(\varphi(b, e)), \mathcal{G}(\varphi(c, a)), \mathcal{G}(\varphi(c, c)), \mathcal{G}(\varphi(c, d)), \mathcal{G}(\varphi(c, e)))$

Indeed it is a function $[0,1]^{12} \rightarrow [0,1]$

Let us further assume that $\mathcal{G}(\varphi(b, a)) = 0$ but for all the other pairs considered here the grounding is 1.

Thus $\mathcal{G}(\forall x_1 x_2 \varphi(x_1, x_2)) = \frac{11}{12}$.

In this example we consider T_1 and T_2 to be proper subsets of $\overline{terms(\mathcal{L})}$ and from now on when T is a proper subset of $\overline{terms(\mathcal{L})}$ we will denote the grounding $\mathcal{G}$ as a partial grounding $\widehat{\mathcal{G}}$. A *partial grounding* $\widehat{\mathcal{G}}$ is a grounding over a subset of the signature of $\mathcal{L}$ and a grounding $\mathcal{G}$ is said to be an extension of a partial grounding $\widehat{\mathcal{G}}$ if $\mathcal{G}$ and $\widehat{\mathcal{G}}$ coincide w.r.t. $\widehat{\mathcal{G}}$ and $\mathcal{G}$ is not a partial grounding[4]. But why do we need to consider partial groundings? The intuition behind the partial grounding is to allow us to make approximations when computing, by only considering a subset of the closed terms. In fact, just with a single 1-ary function symbol and one constant, we already need to deal with an infinite amount of closed terms[5] when computing the aggregation value. Thus, when some $T_i \neq \overline{terms(\mathcal{L})}$ is used in a grounding then the grounding is necessarily a partial grounding and we consider extensions of $\widehat{\mathcal{G}}$ to all $t_x \notin T_i$ and $t_x \in \overline{terms(\mathcal{L})}$. The assumption about the terms representation is made clear here, by assuming that the representation of terms and the function space of predicates and

[3]Here we could have also included open formulas of $\mathcal{L}$ with n free variables but since we focus on sentences and all free variables are implicitly bounded by a universal quantifier in our setting, making them sentences, it is not needed.

[4]We can of course also consider partial groundings w.r.t. either predicate or functions symbols but we will not consider these in this thesis.

[5]$f(a), f(f(a)), f(f(f(a))), \ldots$

function symbols generalizes from $\widehat{\mathcal{G}}$ to $\mathcal{G}$, by generalizing from some finite T to $\overline{terms(\mathcal{L})}$. In our experimental setting we assume that the space of functions is a particular class of differentiable functions and the groundings of the objects are constants allowing us to search through function parameters which best fit our expectations. Thus, in the experiments we let $T \subset \overline{terms(\mathcal{L})}$ be a finite set and treat T as an estimation for $\overline{terms(\mathcal{L})}$ and estimate how well $\widehat{\mathcal{G}}$ generalizes to $\mathcal{G}$ by testing it on previously unseen terms.

We pay a price when considering aggregation semantics. For some aggregation functions we lose *duality.* The universal quantifier and the existential quantifier can be defined in terms of one another as they are dual to each other. $\forall x \varphi(x) = \neg\exists x \neg\varphi(x)$. But when we consider A_{ari} then $\mathcal{G}(\forall x \varphi(x)) = \mathcal{G}(\exists x \varphi(x))$.

Proposition 2 (A_{ari} does not preserve duality)**.** *If A_{ari} is used as a grounding for the universal quantifier then $\mathcal{G}(\forall x \varphi(x)) = \mathcal{G}(\exists x \varphi(x))$.*

Proof.

$$\begin{aligned}\mathcal{G}(\exists x\varphi(x)) &= \mathcal{G}(\neg\forall x\neg\varphi(x)) = 1 - \mathcal{G}(\forall x \neg\varphi(x)) \\ &= 1 - A_{ari}(\{\mathcal{G}(\neg\varphi(a)) \mid a \in T\}) = 1 - \frac{\sum_{a\in T}(1-\mathcal{G}(\varphi(a)))}{|T|} \\ &= 1 - 1 + \frac{\sum_{a\in T}(\mathcal{G}(\varphi(a)))}{|T|} = \mathcal{G}(\forall x\varphi(x))\end{aligned}$$

□

We show this for demonstration purposes only and we will not show this for other aggregation functions. We will, like Donadello et al. (2017), not worry too much about this fact as we do not consider sentences which contain the existential quantifier.

Let us now make clear what we are optimizing when evaluating potential groundings, by first defining satisfiability of a sentence given a grounding.

Definition 23 (Satisfiability)**.** *Let $\varphi(\boldsymbol{x})$ be a sentence in $\mathcal{L}$, $\mathcal{G}$ a grounding of $\mathcal{L}$, $v \leq w \in [0,1]$ then we say that $\mathcal{G}$ satisfies $\varphi(\boldsymbol{x})$ in the interval $[v,w]$ when $\mathcal{G}(\varphi) \in [v,w]$. We use $\mathcal{G} \models_v^w \varphi(\boldsymbol{x})$ to denote the fact that $\mathcal{G}(\varphi) \in [v,w]$.*

Again, take notice that since $\mathcal{G}$ is not a partial grounding then the aggregation semantics are defined over the $\overline{terms(\mathcal{L})}$, not some subset of it. We continue as we want to be able to address satisfiability in terms of multiple sentences and $\widehat{\mathcal{G}}$.

Definition 24 (Ground theory)**.** *A* ground theory *is a pair $\langle \mathcal{K}, \widehat{\mathcal{G}} \rangle$ where $\mathcal{K}$ is a set of pairs $\langle [v,w], \varphi(\boldsymbol{x}) \rangle$ where $\varphi(\boldsymbol{x})$ is a sentence with variables $\boldsymbol{x}$ of $\mathcal{L}$ and $\widehat{\mathcal{G}}$ is a partial grounding.*

Definition 25 (Satisfiable ground theory)**.** *A ground theory $\langle \mathcal{K}, \widehat{\mathcal{G}} \rangle$ is satisfiable if there exists a grounding $\mathcal{G}$ which extends $\widehat{\mathcal{G}}$ s.t. for all $\langle [v,w], \varphi(\boldsymbol{x}) \rangle \in \mathcal{K}$, $\mathcal{G} \models_v^w \varphi(\boldsymbol{x})$.*

Finally, we can address what we want to optimize.

Definition 26 (Loss/Error of a sentence in ground theory)**.** *For a ground theory $\langle \mathcal{K}, \widehat{\mathcal{G}} \rangle$ with $\langle [v,w], \varphi(\boldsymbol{x}) \rangle \in \mathcal{K}$, the error of $\widehat{\mathcal{G}}$ with extension $\mathcal{G}$ w.r.t. $\varphi(\boldsymbol{x})$ is*

$$Loss(\mathcal{G}, \langle [v,w], \varphi(\boldsymbol{x}) \rangle) = \min_{v \leq k \leq w} |k - \mathcal{G}(\varphi(\boldsymbol{x}))|$$

Furthermore, we can see that $Loss(\mathcal{G}, \langle [v,w], \varphi(\boldsymbol{x})\rangle) = 0$ iff $\mathcal{G} \vDash_v^w \varphi(\boldsymbol{x})$. This loss is based on the extended grounding which is defined over $\overline{terms(\mathcal{L})}$ (the Cartesian product of $\overline{terms(\mathcal{L})}$) thus it might encompass an infinite number of terms. Instead we consider the *empirical loss* over some finite set of terms.

Definition 27 (Empirical loss/error of a sentence in ground theory w.r.t. T)**.** *For a ground theory $\langle \mathcal{K}, \widehat{\mathcal{G}}\rangle$ with $\langle [v,w], \varphi(\boldsymbol{x})\rangle \in \mathcal{K}$, the error of $\widehat{\mathcal{G}}$ w.r.t. $T = \{(t_{x_1}, \dots, t_{x_n}) \in T_1 \times \dots \times T_n)\}$ where $T_1, \dots, T_n \subseteq \overline{terms(\mathcal{L})}$ and $\varphi(\boldsymbol{x})$ is*

$$Loss(\widehat{\mathcal{G}}, \langle [v,w], \varphi(\boldsymbol{x})\rangle, T) = \min_{v \le k \le w} |k - \widehat{\mathcal{G}}(\varphi(\boldsymbol{x}))|$$

and $\widehat{\mathcal{G}}(c)$ is defined for all $c \in T$

Thus, we seek to minimize the empirical loss of the partial grounding $\widehat{\mathcal{G}}$ in our search for the extension $\mathcal{G}$. By minimizing the empirical loss, we are maximizing the satisfaction of the $\widehat{\mathcal{G}}$. In the experimental setting, we will only consider $v = w = 1$ for all sentences as we want all of the sentences to be fully satisfied[6]. Up to this point we have not explicitly considered any parameters for the model but essentially the parameters of the model are made concrete when we consider certain classes of groundings. For now, let Ω be the parameters of the model and denote $\widehat{\mathcal{G}}(\cdot \mid \Omega)$ as the grounding using parameters Ω. Thus, we can state the optimization problem which minimizes the empirical loss.

$$\Omega^* = \arg\min_{\Omega' \subseteq \Omega} (1 - \widehat{\mathcal{G}}(\varphi \mid \Omega')) = \arg\max_{\Omega' \subseteq \Omega} \widehat{\mathcal{G}}(\varphi \mid \Omega')$$

We add a regularizing term to this function to limit the size of the parameters to prevent overfitting where λ is a hyperparameter.

$$\Omega^* = \arg\max_{\Omega' \subseteq \Omega} \widehat{\mathcal{G}}(\varphi \mid \Omega') - \lambda ||\Omega||_2^2$$

We have now defined RL and a function which can be optimized to maximize satisfiability of FOL sentences. In the experimental sections, we refer to a ground theory as a knowledge base or as a set of constraints. In the next section, we will introduce a neural network which makes our function space assumptions concrete and in the process gives us a well-defined function parameter search procedure know as *backpropagation*. Since the search procedure relies on the partial derivatives, we will also explore the partial derivatives of the many-valued operators defined in section 1.2.

1.4 Realization

In this section we will define the LTN and the NTN as possible implementations for the predicates of RL[7]. This will make our assumptions about the space of functions concrete and ensures that the functions used to model the predicates are differentiable. After presenting

[6]One might consider other values, for example, $[0.9, 1]$, if one expects a sentence not to be fully satisfiable, but in this thesis, they are not considered.

[7]We will not present possible implementations for the function symbols as they are not explored in this thesis and the implementation in Serafini and Garcez (2016); Serafini and d'Avila Garcez (2016) is simply a linear transformation of the input and provides little insight to RL.

the predicates we will derive the gradient of a sentence's generic grounding and analyse the functions presented in section 1.2. We will end by deriving the logarithm of a generic grounding which will allow us to experiment on the product t-norm. Let us now define the implementations which we consider for the predicates.

Definition 28 (LTN grounding of predicate P)**.** *The* LTN grounding *of a predicate P of arity m with terms $t_1, \dots, t_m \in \overline{terms(\mathcal{L})}$, with the corresponding n dimensional grounding $\boldsymbol{t_1}, \dots, \boldsymbol{t_m}$ and concatenation of terms as $\boldsymbol{t} = (\boldsymbol{t_1}, \dots, \boldsymbol{t_m})$, is a function $\mathcal{G}(P)_{LTN} : \mathbb{R}^{mn} \to [0,1]$ with the following composition.*

$$\mathcal{G}(P)_{LTN} = \mathcal{G}(P)(\boldsymbol{t_1}, \dots, \boldsymbol{t_m}) = \sigma(\boldsymbol{u}_P^T \, tanh(\boldsymbol{t}^T \boldsymbol{W}_P^{[1:k]} \boldsymbol{t} + \boldsymbol{V}_P \boldsymbol{t} + \boldsymbol{b}_P))$$

The parameters of the model are the following, $\boldsymbol{W}_P^{[1:k]}$ a 3-D tensor in $\mathbb{R}^{mn \times mn \times k}$, $\boldsymbol{V}_P$ a matrix in $\mathbb{R}^{k \times mn}$, $\boldsymbol{b}_P$ a vector in $\mathbb{R}^k$ and $\boldsymbol{u}_P^T$ a vector in $\mathbb{R}^k$ and σ is the sigmoid function $\sigma(x) = \frac{e^x}{e^x+1}$.

The LTN grounding was originally introduced alongside the RL framework as a generalization of the NTN (Socher et al., 2013) which we will also consider as an implementation for the predicates. Before defining the NTN let us recall our previous example of Bob, George and John and let us consider how the LTN could decide whether $isTaller(Bob, George)$ based on their representation. Let us define the weights for the LTN so that we can compute $isTaller(George, John)$ based on the representation $Tall(John) = 0.2$ and $Tall(George) = 0.9$, that is, $n = 1$. We define $\mathcal{G}(John) = 0.2$, $\mathcal{G}(George) = 0.9$. We consider a single binary predicate ($m = 2$), $\mathcal{G}(isTaller(x, y))$ and we want $\mathcal{G}(isTaller(George, John))$ to be close to 1. We have no need for the added expressivity of k and set $k = 1$. Set $W = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $v = [100, -100]$, $b = 0$ and $u = 10$. Then we compute.

$$\begin{aligned}
\mathcal{G}(isTaller(George, John)) &= \mathcal{G}(isTaller)(\mathcal{G}(George), \mathcal{G}(John)) = \mathcal{G}(isTaller)(0.9, 0.2)^T \\
&= \sigma(10 \cdot tanh((0.9, 0.2)W(0.9, 0.2)^T + (100, -100)(0.9, 0.2)^T + 0)) \\
&= \sigma(10 \cdot tanh(70)) \approx \sigma(10) \approx 1
\end{aligned}$$

This little toy example shows how the LTN can be used to compute the truth value of a predicate. What this example does not show is how W works. If we were to make the problem a bit more complex and consider an example with $n = 2$ which is even more related to the experimental setting. We consider two terms, t_1 and t_2, a tail and a cat, respectively. We represent each term by its "tailness" and "catness", that is, $\mathcal{G}(t_1) = (tail(t_1), cat(t_1))$ and $\mathcal{G}(t_2) = (tail(t_2), cat(t_1))$. Lets assume that $\mathcal{G}(t_1) = (0.9, 0.2)$ and $\mathcal{G}(t_2) = (0.4, 0.9)$. We would then like to know whether $partOf(t_1, t_2)$ and we assume that this "part of" relation can be decided based on this representation. We would thus expect $partOf(t_1, t_2)$ to have a truth value close to 1 as t_1 is a tail and t_2 is a cat and it is quite possible that this tail is a part of this cat. Conversely, we would not expect $partOf(t_2, t_1)$ to be true, that is, we expect the "part of" relation to be asymmetric and furthermore expect it to be irreflexive. This example might look a bit convoluted at first sight but it reflects the experimental setting well.

Consider these weights and notice the pattern in W. Set $W = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & -100 & 0 & 0 \\ 0 & 0 & -100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$,

$v = [-5, -5, -5, -5]$, $b = 0$ and $u = 10$. We leave it to the reader to verify that, indeed the truth value of $partOf(t_1, t_2)$ is high and $partOf(t_2, t_1)$, $partOf(t_1, t_1)$ and $partOf(t_2, t_2)$ is low. This example shows two things, that W is important if we want to capture logical properties of the relation and that there is symmetry in the LTN. In the next paragraph, we will explore the symmetry in the LTN.

Training a neural network consists of finding values to the parameters of a function in order to minimize the loss defined w.r.t.the output of the network. To do this, the parameters of the function are updated after computing the loss w.r.t.each parameter. To keep the argument simple consider the case with the dimension of terms as n, $m = 2$ and $k = 1$, without loss of generality. That is, we are considering the case for a binary predicate like in the example above and let us now consider the computation $\boldsymbol{t^T} \boldsymbol{W}_P \boldsymbol{t}$ or $\boldsymbol{t_1} \cdot \boldsymbol{t_2^T} \boldsymbol{W}_P \boldsymbol{t_1} \cdot \boldsymbol{t_2}$ where $\cdot$ denotes concatenation. Let us denote the weights in $\boldsymbol{W}_P$ with $w_{t_1^i, t_2^j}$, where t_1^i refers to i-th dimension of term t_1 and t_2^j refers to j-th dimension of term t_2. Thus the single weight denoted by $w_{t_1^i, t_2^j}$ is the weight used in the computation for $\boldsymbol{t_1^i} \cdot \boldsymbol{t_2^j} \cdot w_{t_1^i, t_2^j}$ in which $\boldsymbol{t_1^i}$ refers to the value in the i-th dimension of $\boldsymbol{t_1}$. But then the weights $w_{t_1^i, t_2^j}$ and $w_{t_2^j, t_1^i}$ are used in different computations, $\boldsymbol{t_1^i} \cdot \boldsymbol{t_2^j} \cdot w_{t_1^i, t_2^j}$ and $\boldsymbol{t_2^j} \cdot \boldsymbol{t_1^i} \cdot w_{t_2^j, t_1^i}$ but we know that $\boldsymbol{t_1^i} \cdot \boldsymbol{t_2^j} = \boldsymbol{t_2^j} \cdot \boldsymbol{t_1^i}$. This implies that loss computed w.r.t. $w_{t_1^i, t_2^j}$ and $w_{t_2^j, t_1^i}$ on input $\boldsymbol{t_1^i}$ and $\boldsymbol{t_2^j}$ will be minimal for both weights at a single weight w^*, thus $w_{t_1^i, t_2^j}$ and $w_{t_2^j, t_1^i}$ will both be updated to move towards w^* in attempt to minimize the loss. After some iterations, we would expect the weights to converge to the same value. This argument demonstrates that there is redundancy in the LTN and the weights of the matrix $\boldsymbol{W}_P$ will contain symmetries. Thus these symmetries should be eliminated in implementation.

Let us now define the NTN grounding of a predicate and then compare these two models.

Definition 29 (NTN grounding of predicate P)**.** *The* NTN grounding *of a predicate P of arity* 2 *with terms $t_1, t_2 \in \overline{terms(\mathcal{L})}$, with the corresponding n dimensional grounding $\boldsymbol{t_1}, \boldsymbol{t_2}$ and concatenation of terms as $\boldsymbol{t} = (\boldsymbol{t_1}, \boldsymbol{t_2})$, is a function $\mathcal{G}(P)_{LTN} : \mathbb{R}^{2n} \to \mathbb{R}$ with the following composition.*

$$\mathcal{G}(P)_{NTN} = \mathcal{G}(P)(\boldsymbol{t}_1, \boldsymbol{t}_2) = \boldsymbol{u}_P^T \, tanh(\boldsymbol{t}_1^T \boldsymbol{W}_P^{[1:k]} \boldsymbol{t}_2 + \boldsymbol{V}_P \boldsymbol{t} + \boldsymbol{b}_P)$$

The parameters of the model are the following, $\boldsymbol{W}_P^{[1:k]}$ a 3-D tensor in $\mathbb{R}^{n \times n \times k}$, $\boldsymbol{V}_P$ a matrix in $\mathbb{R}^{k \times 2n}$, $\boldsymbol{b}_P$ a vector in $\mathbb{R}^k$ and $\boldsymbol{u}_P^T$ a vector in $\mathbb{R}^k$.

There are two differences between the functions which we are not too concerned with. First, the NTN does not accept more than two terms at a time, only allowing it to implement binary predicates when the LTN can accept an arbitrary number of terms allowing it to implement predicates of arity $m \in \mathbb{N}$. Secondly, the NTN is a scoring function outputting numbers in $\mathbb{R}$ which are then interpreted more generally than truth values, when the LTN has a specific $[0, 1]$ truth interpretation through the sigmoid function.

These two differences are not important in our experimental setting and for our purposes we adjust the NTN function with the sigmoid function, making it suitable for truth evaluations. To implement unary predicates we simply remove the 3-D tensor $\boldsymbol{W}_P^{[1:k]}$, resulting in the following function $\mathcal{G}(P_1)_{NTN} : \mathbb{R}^n \to \mathbb{R}$.

$$\mathcal{G}(P_1)_{NTN} = \mathcal{G}(P_1)(\boldsymbol{t}_1) = \boldsymbol{u}_P^T \, tanh(\boldsymbol{V}_P \boldsymbol{t} + \boldsymbol{b}_P)$$

No other structural changes are done, thus the parameters of this model are $\boldsymbol{V}_P$ a matrix in $\mathbb{R}^{k \times n}$, $\boldsymbol{b}_P$ a vector in $\mathbb{R}^k$ and $\boldsymbol{u}_P^T$ a vector in $\mathbb{R}^k$.

Let us now compare what the LTN can express and the NTN cannot. Essentially, the LTN can express $\boldsymbol{t}^i_{\boldsymbol{k}} \cdot \boldsymbol{t}^j_{\boldsymbol{k}} \cdot w_{t^i_k, t^j_k}$, $k \in [m]$ and $i \in [n]$, which the NTN can never express. Originally, when comparing the expressivity of LTN and NTN we did not realize this and incorrectly interpreted the redundancy result and hypothesized that the NTN and LTN would perform equally. We will keep to this incorrect hypothesis (hypothesis 6) and report the results and the results will show that this expressivity is beneficial in our setting.

Let us now derive the gradient of a sentence's generic grounding. To derive the gradient of a sentence we need to compute the partial derivative of the sentence w.r.t. every input dimension. The partial derivative is the generalization of the derivative to many dimensions, i.e. the slope of the function w.r.t. to some dimension. By deriving the partial derivative w.r.t. to some arbitrary parameter we will see how the functions introduced in section 1.2 are present in a sentence's gradient. As mentioned in section 1.1 all sentences have the same structure as they are all in SNF and CNF. We now refer to definition 27 in which we defined the empirical loss of a sentence $\varphi(\boldsymbol{x})$ with variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ in a ground theory. Let us compute the partial derivative w.r.t. some parameter p and note that the only parameters of a grounding are parameters of either a predicate or a term. Thus we will compute the gradient up to some literal which is based on the parameter p. We will assume that $\varphi(\boldsymbol{x})$ is in SNF and CNF, that is $\varphi(\boldsymbol{x}) = \psi_1(\boldsymbol{x}) \wedge \cdots \wedge \psi_k(\boldsymbol{x}) = T(\psi_1(\boldsymbol{x}), \ldots, \psi_k(\boldsymbol{x}))$ and that each $\psi_i(\boldsymbol{x})$, $i \in [k]$, is in PNF, that is, $\varphi(\boldsymbol{x})$ is strictly speaking not in SNF as it is not in PNF but we do this because this aligns better with the experimental setting and allows us to optimize the universal quantification as not all $\psi_i(\boldsymbol{x})$ contain the same variables. Thus, $\psi_i(\boldsymbol{x}) = \forall \gamma(\boldsymbol{x})$ and $\gamma(\boldsymbol{x}) = l_1(\boldsymbol{x}) \vee \cdots \vee l_m(\boldsymbol{x}) = S(l_1(\boldsymbol{x}), \ldots, l_m(\boldsymbol{x}))$ where each l_j, $j \in [m]$, is a literal. We will assume that the universal quantification is over some set T of size o. Lastly, as mentioned before, we assume $v = w = 1$[8].

$$Loss(\widehat{\mathcal{G}}, \langle [v,w], \varphi(\boldsymbol{x}) \rangle, T) = \min_{v \le k \le w} |k - \widehat{\mathcal{G}}(\varphi(\boldsymbol{x}))| = 1 - \widehat{\mathcal{G}}(\varphi(\boldsymbol{x})) = \widehat{\mathcal{G}}(\neg\varphi(\boldsymbol{x}))$$

$$\begin{aligned} \frac{\partial(1 - \widehat{\mathcal{G}}(\varphi(\boldsymbol{x})))}{\partial p} &= \frac{\partial(1 - \widehat{\mathcal{G}}(\varphi(\boldsymbol{x})))}{\partial \widehat{\mathcal{G}}(\varphi(\boldsymbol{x}))} \frac{\partial \widehat{\mathcal{G}}(\varphi(\boldsymbol{x}))}{\partial p} = -\frac{\partial \widehat{\mathcal{G}}(\varphi(\boldsymbol{x}))}{\partial p} = -\frac{\partial \widehat{\mathcal{G}}(\psi_1(\boldsymbol{x}) \wedge \cdots \wedge \psi_k(\boldsymbol{x}))}{\partial p} \\ &= -\frac{\partial T(\widehat{\mathcal{G}}(\psi_1(\boldsymbol{x})), \ldots, \widehat{\mathcal{G}}(\psi_k(\boldsymbol{x})))}{\partial p} = \nabla T \cdot \begin{bmatrix} \frac{\partial \widehat{\mathcal{G}}(\psi_1(\boldsymbol{x}))}{\partial p} \\ \cdots \\ \frac{\partial \widehat{\mathcal{G}}(\psi_k(\boldsymbol{x}))}{\partial p} \end{bmatrix} \end{aligned}$$

Below we will explore ∇T[9]. We assume that some $\widehat{\mathcal{G}}(\psi_i(\boldsymbol{x}))$, $i \in [k]$, is a function of p and continue for $\widehat{\mathcal{G}}(\psi_i(\boldsymbol{x}))$.

$$\frac{\partial \widehat{\mathcal{G}}(\psi_i(\boldsymbol{x}))}{\partial p} = \frac{\partial \widehat{\mathcal{G}}(\forall \gamma(\boldsymbol{x}))}{\partial p} = \frac{\partial A(\{\widehat{\mathcal{G}}(\gamma(\boldsymbol{t})) \mid \boldsymbol{t} \in T\})}{\partial p} = \nabla A \cdot \begin{bmatrix} \frac{\partial \widehat{\mathcal{G}}(\gamma(\boldsymbol{t}_1))}{\partial p} \\ \cdots \\ \frac{\partial \widehat{\mathcal{G}}(\gamma(\boldsymbol{t}_o))}{\partial p} \end{bmatrix}$$

[8]Take note of the last equality sign in the first line. This special case can be considered as a refutation proof.

[9]Here, and in the following partial derivatives, we omit the input to ∇T instead of writing $\nabla T(\widehat{\mathcal{G}}(\psi_1(\boldsymbol{x})), \ldots, \widehat{\mathcal{G}}(\psi_k(\boldsymbol{x})))$ for readability.

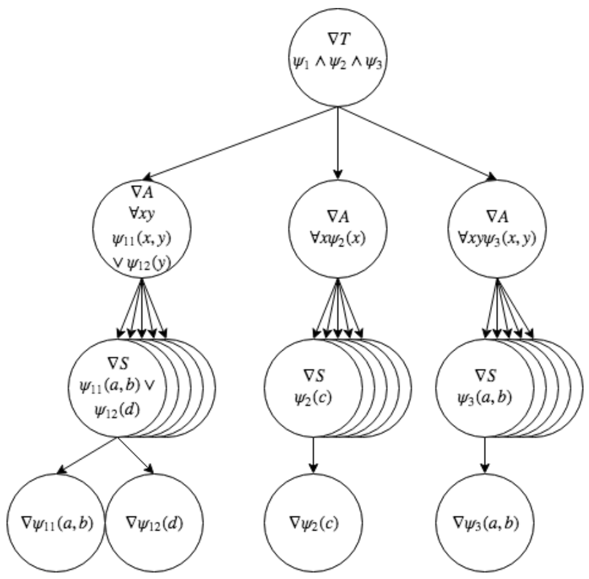


Figure 1.2: The flow of gradient through a sentence in an experimental setting. An arrow represents a partial derivative and shows how the gradient splits based on input dimensions. ∇T, ∇A and ∇S are dependent on the instantiation and this images shows the importance of $\nabla T \neq \mathbf{0}$. The leaf nodes are not replicated in the last layer to unclutter the image.

We assume that some $\widehat{\mathcal{G}}(\gamma(\boldsymbol{t}_i))$, $i \in [o]$ is a function of p and continue for $\widehat{\mathcal{G}}(\gamma(\boldsymbol{t}_i))$.

$$\frac{\partial \widehat{\mathcal{G}}(\gamma(\boldsymbol{t}_i))}{\partial p} = \frac{\partial \widehat{\mathcal{G}}(l_1(\boldsymbol{t}_i) \vee \cdots \vee l_m(\boldsymbol{t}_i))}{\partial p} = \frac{\partial S(\widehat{\mathcal{G}}(l_1(\boldsymbol{t}_i)), \ldots, \widehat{\mathcal{G}}(l_m(\boldsymbol{t}_i)))}{\partial p} = \nabla S \cdot \begin{bmatrix} \frac{\partial \widehat{\mathcal{G}}(l_1(\boldsymbol{t}_i))}{\partial p} \\ \cdots \\ \frac{\partial \widehat{\mathcal{G}}(l_m(\boldsymbol{t}_i))}{\partial p} \end{bmatrix}$$

At this point we will stop but one can see how the partial derivatives could be taken w.r.t. a literal, which might contain a negation, and then take the partial derivative of the predicate as defined by either the LTN or the NTN. This shows us how ∇T, ∇A and ∇S are chained in order as can be seen in figure 1.2. We can also see that if $\nabla T = \mathbf{0}$ then the gradient of the whole network will be $\mathbf{0}$. Thus, we want to see if some of our operators have $\mathbf{0}$ gradient w.r.t. some input dimension or for some parts of the input domain. To make it clear, ∇T, ∇A and ∇S are dependent on the instantiation of RL and we want to see how the choice of many-valued functions impacts the gradient of our network.

Let us now compute the partial derivatives of the t-norms and start with the Gödel t-norm.

$$\frac{\partial T_G(x_1, \ldots, x_n)}{\partial x_i} = \frac{\partial min(x_1, \ldots, x_n)}{\partial x_i} = \begin{Bmatrix} 1, & \text{if } x_i = \min(x_1, \ldots, x_n) \\ 0, & \text{otherwise} \end{Bmatrix}$$

We can see that $\frac{\partial T_G(x_1, \ldots, x_n)}{\partial x_i}$ is always 1 for a single x_i (the minimum value). Thus, T_G will never have a zero gradient, as there is always a 1 along some dimension for all values of the input domain, i.e. $\nabla T_G(x_1, \ldots, x_n) \neq \mathbf{0}$ for all $x_1, \ldots, x_n \in [0, 1]$[10]. Even though T_G has a

[10]Here it is assumed that the maximum/minimum operator always picks a single element even though all of them are equal. For our purposes, we do not care what element is picked. This is not stated in the definition of the aggregate function in order to make it more readable.

non-zero gradient, it will only update parameters to one dimension at a time and in section 4.1 we argue that this can cause issues. Let us now explore partial derivative of the Łukasiewicz t-norm.

$$\frac{\partial T_{\text{L}}(x_1,\ldots,x_n)}{\partial x_i} = \frac{\partial \max(\sum_{j=1}^n x_j - (n-1), 0)}{\partial x_i} = \begin{cases} 1, & \text{if } \sum_{i=1}^n x_i > n-1 \\ 0, & \text{otherwise} \end{cases}$$

We can see that $\frac{\partial T_{\text{L}}(x_1,\ldots,x_n)}{\partial x_i}$ is 1 along all dimensions when $\sum_{i=1}^n x_i > n-1$, otherwise 0. Essentially, the Łukasiewicz t-norm has $\mathbf{0}$ gradient when $\sum_{i=1}^n x_i < n-1$. This motivates the following proposition.

Proposition 3 (Łukasiewicz t-norm and input space)**.** *When the number of inputs tends to infinity and* $\frac{\partial T_L(x_1,\ldots,x_n)}{\partial x_i} \neq \mathbf{0}$ *then* $\frac{\sum_{i=1}^n x_i}{n} = 1$.

Proof. If $\frac{\partial T_{\text{L}}(x_1,\ldots,x_n)}{\partial x_i} \neq \mathbf{0}$ then $\sum_{i=1}^n x_i > n-1$, given $n > 0$ then $\frac{\sum_{i=1}^n x_i}{n} > \frac{n-1}{n}$ we want to know $\lim_{x\to\infty} \frac{\sum_{i=1}^n x_i}{n}$ thus we use L'Hôpital's rule on $\frac{n-1}{n}$ and when the number of input parameters tends to infinity then $\frac{\sum_{i=1}^n x_i}{n} = 1$, that is, the average value approaches 1. □

This tells us that when the number of inputs approaches infinity and we want the gradient to be non-zero then the average value for each input approaches 1. At first, this might not seem like a big issue for us in the experimental setting, but in fact this is an issue with $n = 2$. Consider the case for $n = 2$, with $x_1 = 0.4$ and $x_2 = 0.5$, then the gradient will be $\mathbf{0}$. In our experimental setting, all parameters of the network are initialized so that each x_i will have a value close to 0, thus the gradient will always be $\mathbf{0}$. We thus hypothesize that the Łukasiewicz t-norm will not be trainable in our experimental setting. We will revisit this hypothesis in section 2.4 in hypothesis 1.

Let us now compute the derivative of the product t-norm.

$$\frac{\partial T_P(x_1,\ldots,x_n)}{\partial x_i} = \frac{\partial \prod_{j=1}^n x_j}{\partial x_i} = \prod_{j=1 \wedge j\neq i}^{n} x_j$$

We can see that if some $x_j = 0$ then the partial derivative will be 0 to all variables but x_j. This does not cause much alarm for us, as in our experimental setting these values might be close to 0, but never actually 0. Again, we make note of the possible computational underflow issues.

We will not compute the partial derivatives of the s-norms, as they are very similar to the t-norms but we will address the issue of being "easily satisfiable" we raised when introducing the Łukasiewicz s-norm.

Proposition 4 (Łukasiewicz s-norm and input space)**.** *When the number of inputs tends to infinity and all* $x_i > 0$ *then* $\min(\sum_{i=1}^n x_i, 1) = 1$

Proof. If $x_i > 0$ then $\lim_{n\to\infty} \sum_{i=1}^n x_i$ tends to infinity thus, $\min(\lim_{n\to\infty} \sum_{i=1}^n x_i, 1) = 1$ □

This implies that the Łukasiewicz s-norm might become trivially satisfied when considering many inputs. This does not impact our experimental setting that much and we will see in the results of hypothesis 2 that the Łukasiewicz s-norm is indeed more easily satisfiable but at the same time outperforming the Gödel t-norm. This issue is not as serious as with the

Łukasiewicz t-norm, due to the structure of our sentences, but it would be if we considered Disjunctive Normal Form instead of CNF. We will discuss the norms better in section 4.1.

Now similarly as we did with the norms, we want to compute the partial derivatives of the aggregation functions. For the minimum and maximum, we refer to the partial derivatives derived previously from the norms. That is, the gradient for A_{min} and A_{max} w.r.t. x_i is always 1 for a single x_i, the minimum and maximum value, respectively. For all other input variables, it is 0. Let us now derive the partial derivative of A_{ari}.

$$\frac{\partial A_{ari}(x_1,\ldots,x_n)}{\partial x_i} = \frac{\partial \frac{\sum_{j=1}^{n} x_j}{n}}{\partial x_i} = \frac{1}{n}$$

We see that A_{ari} w.r.t. x_i has a constant gradient for every input over the whole domain, but we notice that if the number of input variables tends to infinity then the gradient tends to $\mathbf{0}$. Despite this drawback, we suspect that A_{ari} will be the most practical in high dimensional implementations due to the simplicity in computation, as it is a constant. Let us now derive the partial derivative of A_{har}.

$$\frac{\partial A_{har}(x_1,\ldots,x_n)}{\partial x_i} = \frac{\partial \frac{n}{\sum_{j=1}^{n} x_j^{-1}}}{\partial x_i} = \frac{n}{x_i^2(\sum_{j=1}^{n} x_j^{-1})^2}$$

Similarly, the gradient for A_{har} w.r.t. x_i is always positive for every input over the whole domain except when $x_i = 0$ or all other $x_j = 0$, $j \neq i$, then it is not defined. Again, despite these drawbacks, we are not too worried about these properties of the A_{har} and we will rely heavily on it during experimentation.

We can also see that the partial derivative of A_{owa} w.r.t. x_i is w_i. This fact motivates the hypothesis that in a high dimensional implementation the OWA operator might be useful by limiting computation over only the elements which are less satisfied. Thus, we hypothesize that an OWA operator which has 0 weights for the first elements and then distributes the remainder over the last elements will outperform the arithmetic mean in large-scale experiments and present this hypothesis in section 2.4 in hypothesis 8.

We will now address the issue of numerical underflow for the product norms. Instead of computing $\mathcal{G}(\varphi)$ we will compute $\log(\mathcal{G}(\varphi))$. Let us start by pointing to the fact that the log is a monotonically increasing function so the minimum of this function will also be the minimum of the log of this function. This implies that we can just as well search for optimal parameters of the log of the grounding, rather the grounding itself. Due to our sentence structure, we need to make sure that when we take the log of the conjunction, it is passed down, all the way to the disjunction. We start by considering the log-product t-norm.

Proposition 5 (The log-product t-norm). *The* log *of the product t-norm is*

$$\log(T_P(x_1,\ldots,x_n)) = \log(\prod_{i=1}^{n} x_i) = \sum_{i=1}^{n} \log(x_i)$$

We can retrieve the product t-norm value from the log-product norm, $T_P(x_1,\ldots,x_n) = e^{\log(T_P(x_1,\ldots,x_n))}$.

Next, we move to the aggregation functions and consider the logarithm of A_{har} as we will use it in the experimental setting to allow more numerically stable computation.

$$\begin{aligned}\log(A_{har}(x_1,\dots,x_n)) =& \log(\frac{n}{\sum_{i=1}^{n} x_i^{-1}}) = \log(n) - \log(\sum_{i=1}^{n} x_i^{-1}) \\ =& \log(n) - \log(\sum_{i=1}^{n} e^{\log(x_i^{-1})}) = \log(n) - \log(\sum_{i=1}^{n} e^{-\log(x_i)})\end{aligned}$$

We will then use a numerically stable estimation for $f(x_1,\dots,x_n) = \log(\sum_{i=1}^{n} e^{x_i})$ in the implementation.

Lastly, we consider the log of the product s-norm.

Definition 30 (The log-product s-norm)**.** *The* log *of the product s-norm is*

$$\log(S_P(x_1,\dots,x_n)) = \log(1 - \prod_{i=1}^{n}(1-x_i)) = log1p(-e^{\sum_{i=1}^{n}\log(1-x_i)})$$

Where $log1p(x) = \log(1+x)$ is an optimized function for small x provided with many numerical computation libraries.

These derivations allow us to use the product norms in computation and in hypothesis 7 we will see the benefits of this extra work.

We have now defined the LTN and the NTN as possible implementations for the predicates of RL, this makes our assumptions about the space of functions concrete and ensures that the functions used to model the predicates are differentiable. We also computed the gradient of a sentence's generic grounding and saw the importance of the role ∇T, ∇A and ∇S play in RL when using a gradient method to update the parameters.

Chapter 2

Experimental design

In this section, we describe how RL is used in the Semantic Image Interpretation (SII) task and how it will be evaluated. In section 2.1, we start by defining the task of SII and the dataset which is used to perform the task. In section 2.2, we define a knowledge base which describes the dataset along with logical properties of the dataset and define a grounding over this knowledge base allowing us to perform the SII task. In section 2.3, we define the measures used to evaluate classification performance and logical consistency of the model's predictions. In section 2.4, we state our experimental hypotheses and motivate them.

2.1 The Setting

On a high level, the task of SII is to produce a scene graph given an image. In the scene graph, the nodes represent some object in the image and edges between nodes imply a relationship between the objects (Donadello et al., 2017; Serafini et al., 2017). The task is then to label objects and potential relations in the image. In addition to the image, we also base the predictions on background knowledge in the form of a knowledge base. The knowledge base describes properties about objects and relations in the image. We view the knowledge base as additional constraints and expect the predictions of the model to be consistent with these constraints.

We perform this task on the PASCAL-Part dataset (Chen et al., 2014) for the three following reasons. First, it provides a good selection of images along with bounding boxes around objects of interest in each image, its labelling and pair labellings. Second, a simple ontology is provided with the dataset which is easy to model in FOL. Third, a lot of work has already been done for this task on this dataset by Donadello et al. (2017); Serafini et al. (2017) which provides a good place to start.

The PASCAL-Part dataset is a further annotated dataset of the PASCAL VOC 2010[1] dataset (Everingham et al., 2010). The original dataset contained 21,738 images (10,103 training, 9,637 testing) and 20 different classes for the classification task along with bounding boxes around objects of interests, but did not contain training data for relation prediction. This is added in the PASCAL-Part dataset, along with more fine-grained object classification. In fact, the object classification is so fine-grained that the parts are separated based on direction and alignment, f.ex. "left leg" and "right back upper leg". We follow Donadello

[1]PASCAL is an acronym for Pattern, Analysis, Statistical Modelling an Computational Learning and VOC is an acronym for Visual Object Classes.

et al. (2017); Serafini et al. (2017) by merging these finer partitions into a single partition, i.e. "left leg" and "right back upper leg" are taken as "leg". After making this adjustment there are 40 additional classes added to the dataset. These 40 classes are "parts" of 20 original "wholes". For these classes we consider a single binary relation, the "part of" relation for when "x is a part of y". The ontology provided states what "parts" each "whole" object consists of. As an example, in the dataset a "bicycle" is considered as a whole consisting of the parts "chain wheel","handlebar", "headlight", "saddle" and "wheel", thus "saddle is a part of bicycle" is a true statement according to the ontology. Thus, when we classify something as a "bicycle" and there is another object overlapping with "bicycle" and that object is one of "chain wheel","handlebar", "headlight", "saddle" and "wheel" we are inclined to infer that these two are related. This is not always the case and in our experiments we want to know whether a particular "saddle" is a part of a particular "bicycle", based on their representations. For practical reasons, these classes are then further separated into these three categories: indoor objects, vehicles and animals. The following experiments are only based on the PASCAL-Part training set (10,103 training examples) and we follow Donadello et al. (2017); Serafini et al. (2017) and eliminate images smaller than 6×6.

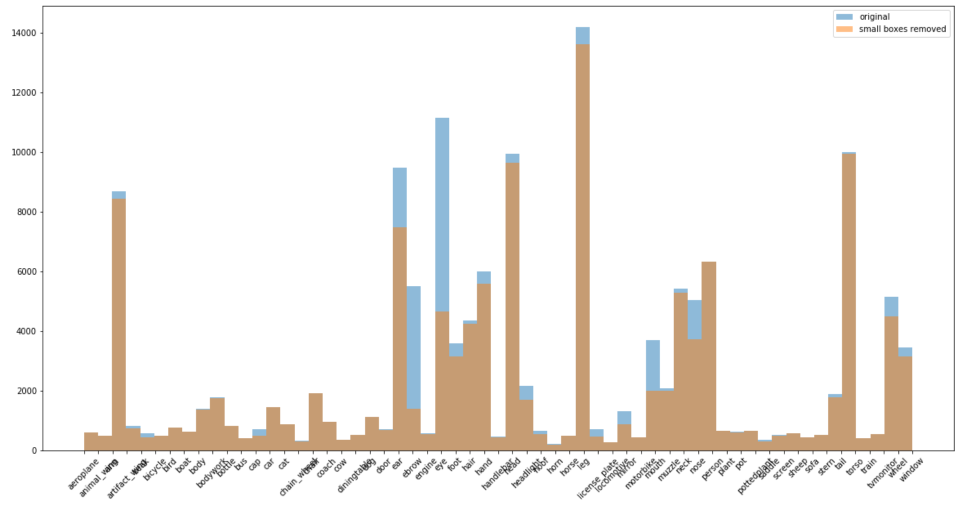
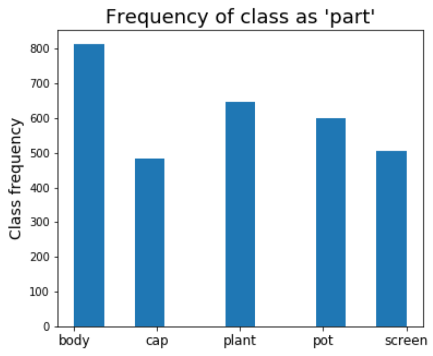
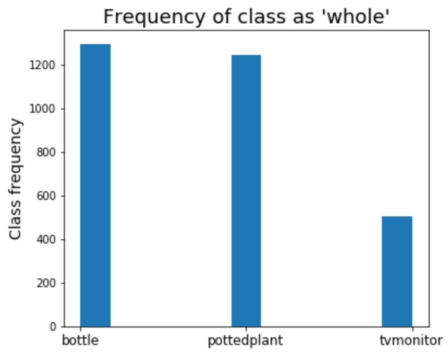


Figure 2.1: Class distribution of the training dataset. We can see that we have varying numbers of examples of each class. The image also shows the effect of removing images smaller than 6×6 from the dataset.

The dataset is split (80%/20%) to a training and testing set while maintaining the same proportion of classes in both sets. We then train on the training data and test how well the model generalizes on the test data. As can be seen in figure 2.1, there are unevenly distributed number of examples of each class. Measures which do not account for *class imbalance* can give a distorted view of the model's performance and we will discuss those in a later section. We can combat class imbalance in a few different ways, but ultimately they are all implemented through the loss function of the model. By adjusting the loss function to account for class imbalance we are essentially stating explicitly what we consider a "good" model.



(a) In this image we can see the different class frequencies of an object in the ”part of” relation as a ”part”. All parts take part in the ”part of” relation.



(b) Similar to figure 2.3a, in this image we can see the different class frequencies of an object in the ”part of” relation as a ”whole”. All the wholes are ”bottle”, ”pottedplant”, ”tvmonitor”, ”chair”, ”sofa”, ”diningtable” but since ”chair”, ”sofa” and ”diningtable” do not have any parts they are never in the ”part of” relation.

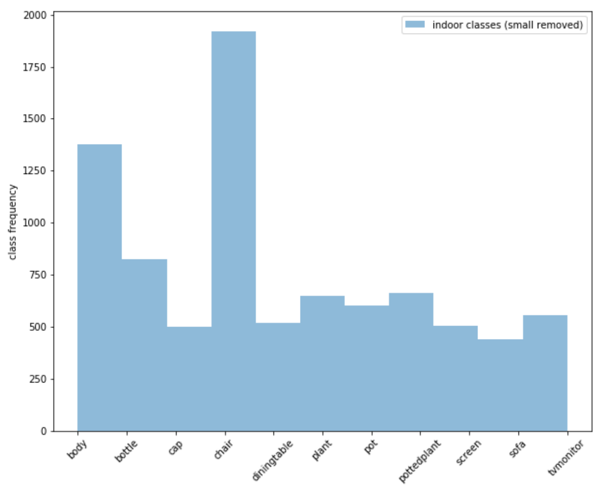


Figure 2.2: Class distribution over the training dataset with only indoor objects with small images removed. There are in total 11 classes.

Since the correct classification of classes impacts how well the ”part of” relation can be predicted, we are interested in learning more about this relation. For example, how sparse is the relation? That is, for all possible pairs of bounding boxes, how often are two bounding boxes related? Furthermore, what types of objects are most often the ”wholes” and ”parts” in the ”part of” relation? We will use the ”indoor” objects dataset to answer these questions. See figure 2.2 for the class distribution of the ”indoor” dataset. In the indoor dataset there are 2135 images broken into 8535 bounding boxes, thus roughly 4 bounding boxes per image. There are in total 76037 pairs of bounding boxes of which only 3049 are positive examples of the ”part of” relation or 4%.

As we can see in figures 2.3a and 2.3b the class imbalance is not that dramatic when it comes to the ”part of” relation so we can treat each class equally and during training we will sample each class equally. In some settings, this might lead to overfitting for the class prediction but we see as a result of hypothesis 4 that this is not an issue and the class prediction generalizes well.

The following experiments are all based on intermediary data. That is, we do not use the images themselves as data, but rather use precomputed representations by Donadello et al. (2017). The representations are based on the output of a Fast R-CNN as well as additional hand-crafted features (Girshick, 2015). By using this representation we side-step the task of predicting bounding boxes in the image as well as computing representations of those bounding boxes. Predicting bounding boxes dynamically, using reasoning, sounds like a challenging task which we will not explore now. We want to know if RL can use logical constraints to improve binary classification, relation prediction and make the predictions logically consistent. We demonstrate that it can, by applying it directly to this representation, which is effectively the predictions of the Fast R-CNN.

2.2 The Models

In the following section, we will define a FOL language $\mathcal{L}$ in which we will describe the elements of PASCAL-Part dataset and logical constraints which apply to it. These FOL sentences are then considered as a knowledge base and we train the model to minimize the *empirical loss* of these sentences. During training we search for a partial grounding $\widehat{\mathcal{G}}$ which is able to satisfy our knowledge base the most. The grounding is partial because the universal quantifiers are limited and 2147 constants withheld during training. Thus we optimize $\widehat{\mathcal{G}}$ w.r.t. to the training set and evaluate it on the withheld constants. During the experiments we observe that *no* $\widehat{\mathcal{G}}$ is able to satisfy the training dataset in the interval $[1, 1]$, or even $[0.95, 1]$, indicating that the model space is not expressive enough, that is, the representation, as well as the functions considered, cannot account for the data.

Let us now define a FOL $\mathcal{L}$ by defining a FOL signature which we use to describe a domain of bounding boxes. We let $B_{train} = \{b_1, \dots, b_{8525}\}$ be the training domain and $B_{test} = \{b_{8526}, \dots, b_{10672}\}$ be the testing domain, thus $B = B_{train} \cup B_{test}$ is the domain we want to model and we refer to each bounding box with a constant symbol b_i.

We model the object types as unary predicates $\mathcal{P}_1 = \{P_1, \dots, P_{60}\}$ and the "part of" relation as a binary predicate $\mathcal{P}_2 = \{R\}$, thus $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 = \{P_1, \dots, P_{60}, R\}$. We will not consider any function symbols for this task, $\mathcal{F} = \emptyset$. Thus, $\mathcal{L} = \langle \mathcal{C}, \mathcal{P}, \emptyset \rangle$.

Before continuing and defining a grounding $\mathcal{G}$ over this language we need to discuss groundings of pairs of bounding boxes. In the following definition we follow Donadello et al. (2017); Serafini et al. (2017) and define groundings w.r.t. pairs of bounding boxes. This is not well defined for RL as RL defines groundings over the signature of $\mathcal{L}$. If we were to use the current definition of groundings and treat pairs as constants we will need to adjust the RL definition quite a lot to make it work. We rather consider an extension of RL which also considers pairs of constants as separate entities, that is, we assume that $\mathcal{G}(a, b)$, where a and b are constants, is a function $\mathcal{C} \times \mathcal{C} \to \mathbb{R}^{2n+k}$ where $k \geq 0$ and n is the dimension of grounding of constants. We thus consider the possibility of adding extra dimensions to the representations when grounding pairs of constants. In the following paragraph we will define use this extension to define a pair grounding. Lastly, when we consider the pairs, $B_{pairs} \subseteq B \times B$ we do not consider all possible pairs of bounding boxes but rather only pairs derived from the same image, greatly reducing the number of possible pairs.

Now we will define the grounding of $\mathcal{L}$. We use the output of the Fast R-CNN to define the grounding of constants in addition to the bounding box's location information. The output of the Fast R-CNN given a bounding box b, dentoted as $p(b)$, is a probability distribution over all classes $P_1, \dots, P_{60}$ and the location information is the bottom-left location (x_1, y_1) of the box along with the top-right location (x_2, y_2). The grounding of a single bounding box $b_i \in B$ is

$$\mathcal{G}(b_i) = (p(P_1 \mid b_i), \ldots, p(P_{60} \mid b_i), x_1(b_i), y_1(b_i), x_2(b_i), y_2(b_i))$$

As mentioned above, we don't represent pairs just as the concetination of two representations of a bounding but we also add additional information in form of the inclusion ratio $ir(b_i, b_j)$ for the pair of bounding boxes $(b_i, b_j) \in B_{pairs}$.

$$ir(b_i, b_j) = \frac{area(b_i \cap b_j)}{area(b_i)}$$

$$\mathcal{G}(b_i, b_j) = (\mathcal{G}(b_i), \mathcal{G}(b_j), ir(b_i, b_j), ir(b_j, b_i))$$

We then define the baseline we want to compare to also in terms of groundings, for each $j \in [|\mathcal{P}_1|]$

$$\mathcal{G}_b(P_j)(\mathcal{G}(b_k)) = \begin{cases} 1 \text{ if } j = \arg\max_{i=1}^{|\mathcal{P}_1|} \mathcal{G}(b_k)_i \\ 0 \text{ o.w.} \end{cases}$$

Where $\mathcal{G}(b_k)_i$ is the i-th dimension of the grounding for bounding box b_k. Essentially, we choose the type with the highest predicted value from the Fast R-CNN for each bounding box. For the "part of" relation, we consider two baselines.

$$\mathcal{G}_b^1(R)(\mathcal{G}((b_i, b_j))) = \begin{cases} 1 & \text{if } ir(b_i, b_j) \geq th \\ & 0 \text{ o.w.} \end{cases}$$

$$\mathcal{G}_b^2(R)(\mathcal{G}((b_i, b_j))) = \begin{cases} 1 & \text{if } ir(b_i, b_j) \geq th \text{ and the resolved type of } b_i \text{ is "part of" } b_j \\ & 0 \text{ o.w.} \end{cases}$$

That is, $\mathcal{G}_b^2(R)$ takes into account type compatibility.

The unary predicate baseline can either predict a special type of class, which has no positive examples in neither the training nor testing datasets, the "background" class. This baseline will predict this class on a number of occasions, as this is what the Fast R-CNN will do. This will always lead to strictly worse performance. We will consider two baselines, one which predicts the "background" class and another which will never predict this class. We do this to offer an additional comparison to our model, but essentially, our model will never predict this class.

Furthermore, the framework needs to be instantiated with some t-norm, s-norm, aggregation function and predicate implementations and in the following experiments we consider some combinations of these instantiations and denote them in the subscript; $\mathcal{G}_{LTN,har,T_G}(\varphi)$ refers to the LTN implementation of predicates, the harmonic mean as an aggregation operator and the Gödel t-norm and s-norm. But since this notation is very verbose, we omit LTN, har and T_G from the notation, that is, $\mathcal{G}(\varphi) = \mathcal{G}_{LTN,har,T_G}(\varphi)$ and when some other t-norm is used instead, we indicate that using this notation. Thus, $\mathcal{G}_{LTN,har,T_{\text{L}}}(\varphi)$ is denoted as $\mathcal{G}_{T_{\text{L}}}(\varphi)$.

We have yet to define our knowledge base using $\mathcal{L}$ which we want to satisfy. Recall that the *empirical loss* is defined w.r.t. $\langle [v, w], \varphi(\boldsymbol{x}) \rangle \in \mathcal{K}$.

$$Loss(\widehat{\mathcal{G}}, \langle [v, w], \varphi(\boldsymbol{x}) \rangle, T)$$

For every sentence we need to stipulate the range of satisfaction and as mentioned before, we will only consider fully satisfied sentence and thus omit it when we define $\mathcal{K}$. For the training we define $\varphi_{P_i} = \forall x P_i(x)$ along with $T_i = \{b \mid b \in B \text{ and } b \text{ is of type } P_i\}$ and similarly $\varphi_{\neg P_i} = \forall x \neg P_i(x)$ with T_i's compliment $B \setminus T_i$. We furthermore limit the size for each T_i (and compliment) and sample with replacement. We do the same for the "part of" relation, namely $\varphi_R = \forall xy R(x, y)$ along with $T_R = \{(b, b') \mid b, b' \in B \text{ and } b \text{ is a part of } b'\}$ and similarly we have $\varphi_{\neg R}$. Thus,

$$\mathcal{K} = \{\varphi_{P_1}, \varphi_{\neg P_1}, \ldots, \varphi_{P_{60}}, \varphi_{\neg P_{60}}, \varphi_R, \varphi_{\neg R}\}$$

We then define $\mathcal{K}_f = \{\bigwedge_{\varphi \in \mathcal{K}} \varphi\}$, where the f stands for "facts". $\mathcal{K}$ can be considered as the "normal" classification setting in ML.

Now we define additional constraints in addition to our set of facts. In order to make the discussion about the constraints more intuitive, some more notation is helpful. Let $Wholes = \{P \mid P \in \mathcal{P}, P \text{ is a whole}\}$ and $Parts = \{P \mid P \in \mathcal{P}, P \text{ is a part}\}$. Note that in our dataset $Wholes \cap Parts = \emptyset$, which motivates sentences 2.5 and 2.6 below. Lastly, it is often helpful to talk about parts of a specific whole and the converse, we abuse the notation a bit and denote this with $Parts(P)$ where P is a whole and $Wholes(P)$ where P is a part. The following sentences in $\mathcal{L}$ describe logical constraints which the dataset should satisfy based on our knowledge[2]. We additionally tested the data w.r.t. these sentences and saw that this is indeed the case.

$$\forall xy(R(x, y) \implies \neg R(y, x)) \qquad \text{(Asymmetric)} \quad (2.1)$$

$$\forall x(\neg R(x, x)) \qquad \text{(Irreflexive)} \quad (2.2)$$

$$\forall x\Big(\bigvee_{P \in \mathcal{P}_1} P(x)\Big) \qquad \text{(Every } x \text{ is some type)} \quad (2.3)$$

$$P \in \mathcal{P}_1, \forall x\Big((P(x) \implies \neg\Big(\bigvee_{P' \in \mathcal{P}_1 \setminus \{P\}} P'(x)\Big)\Big)\Big) \qquad \text{(Disjoint types)} \quad (2.4)$$

$$P \in Wholes, \forall xy(P(x) \implies \neg(R(x, y))) \qquad \text{(Whole is not a part)} \quad (2.5)$$

$$P \in Parts, \forall xy(P(x) \implies \neg(R(y, x))) \qquad \text{(Part is not a whole)} \quad (2.6)$$

$$P \in Wholes, \forall xy(P(x) \wedge R(y, x) \implies \Big(\bigvee_{P' \in Parts(P)} P'(y)\Big)) \qquad \text{(Whole's parts)} \quad (2.7)$$

$$P \in Parts, \forall xy(P(x) \wedge R(x, y) \implies \Big(\bigvee_{P' \in Wholes(P)} P'(y)\Big)) \qquad \text{(Part's wholes)} \quad (2.8)$$

Take note that sentences 2.4-2.8 are schemas and are therefore instantiated as multiple sentences.

Notice that the universal quantifiers for all the above sentences either quantify over x or both x and y. One would initially expect quantification over x to only quantify over the bounding boxes and $\forall xy$ to quantify over pairs, but it is not that simple, as sentence 2.2 is referring to pairs of bounding boxes but still only quantifying over x. Thus, during implementation we treat these universal quantifiers somewhat equally, that is, they are all quantifying over some sampled subset $T \subset B_{pairs}$. For sentences 2.5-2.8, we need to take care of segmenting the pairs correctly, passing the x part of the pair to the unary predicate. For sentences 2.3 and 2.4, it can be argued that quantifying over B is sufficient, but we use the same sample

[2]The following constraints are not written in CNF but are of course implemented in CNF.

and only consider the first object in the pair. We use a similar trick for sentence 1 and mirror the representation. For sentence 2.2 we use the same trick again, we segment the pairs and then recombine the first object in the pair with itself and set both inclusion ratios to 2.1.

Example 10 (Treatment of quantifiers)**.** *Let* $T = \{(b_1, b_1), (b_1, b_3), (b_2, b_3)\}$ *be the sample and we want to evaluate sentence 2.1.*

$$\mathcal{G}(\forall xy R(x, y) \implies \neg R(y, x)) =$$
$$A(\mathcal{G}(\neg R(b_1, b_1) \vee \neg R(b_1, b_1)), \mathcal{G}(\neg R(b_1, b_3) \vee \neg R(b_3, b_1)), \mathcal{G}(\neg R(b_2, b_3) \vee \neg R(b_3, b_2)))$$

Similarly, sentence 2.2 will be evaluated as

$$\mathcal{G}(\forall x \neg R(x, x)) =$$
$$A(\mathcal{G}(\neg R(b_1, b_1)), \mathcal{G}(\neg R(b_1, b_1)), \mathcal{G}(\neg R(b_2, b_2)))$$

During the experiments, we consider different combinations of these sentences as constraints. We denote the union of $\mathcal{K}_f$ and the set of all constraints as $\mathcal{K}_{all}$. We then denote the combination of $\mathcal{K}_f$ and other axioms in the subscript, f.ex. the combination of the facts, asymmetry and irreflexivity as $\mathcal{K}_{asym,irr}$. We have now everything in place to describe different instantiations of the RL w.r.t. operators and different constraints. For example, consider $\mathcal{G}$ and $\mathcal{K}_{disj,one,parts,wholes}$. $\mathcal{G}$ refers to LTN implementation of the predicates, the Gödel t-norm and s-norm and the harmonic mean. $\mathcal{K}_{disj,one,parts,wholes}$ refers to all the facts and the following constraints: disjoint types, every x is some type, whole's parts and part's wholes, respectively.

Lastly, we define a single training iteration as a single forward pass on the network along with a backward pass to compute and update the parameters. There are quite a few hyper-parameters in this model which affect the performance of the model, especially when considering different instantiations, but we chose a set of hyper-parameters which most instantiations performed well on. In the following experiments the number of positive examples per class, $|T_i|$, is set to 200 and the number of negative examples per class, the complement of $|T_i|$ is set to 2000 during training. We use the same sampling parameters for positive and negative examples of the "part of" relation. For the universal quantifiers, we use a sample size of 2000. We then run 50 iterations before resampling new constants. We set $\lambda = 10^{-10}$ We use the RMS prop algorithm with a learning rate of 0.01 and decay of 0.9 (Tieleman and Hinton, 2012). The dimension k of the predicates (the slices) is set to 6. We then report results after 300 iterations. When evaluating the performance of the model we define the universal quantifiers over the whole test set to evaluate the logical constraints.

We have now defined a FOL $\mathcal{L}$ in which we describe the elements of PASCAL-Part dataset and logical constraints which apply to it. These sentences are our knowledge base. We defined a grounding over the knowledge base and we search for the partial grounding which is able to maximize satisfiability of our knowledge base. We will experiment with different instantiations of RL and different logical constraints. In the next section, we will discuss how we will measure the performance of these combinations.

2.3 Measures

In this section, we will introduce two types of measures used to evaluate the performance of the model. We will use $F1$ scores to measure the performance of the model w.r.t. correct classification. Since there are multiple class types, we aggregate the $F1$ scores of the types to report a single number, the macro $F1$ score. Similarly, we report the $F1$ score of the model for the "part of" relation. In order to measure logically consistent predictions, we report two other measures, the *grounding satisfaction* and *modus ponens satisfaction*, both of which are measured w.r.t. all of the logical constraints (sentences 1–8 above). The $F1$ score indicates whether the model is making correct predictions but we also want to know whether the model is making logically consistent predictions and we will see that an increase in logical consistency of predictions does not necessarily imply an increase in $F1$ score. The measures are computed using the complete test dataset.

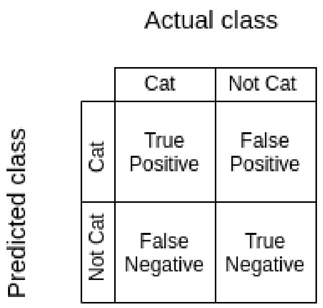


Figure 2.3: Confusion matrix for binary classification.

Let us start with an underlying concept used to compute the $F1$ score, the confusion matrix, see figure 2.3 for a binary confusion matrix. The confusion matrix is used to compute measures such as *accuracy*, *precision* and *recall* in classification tasks, which are then used to compute other measures like the $F1$ *score* and *Precision-Recall Area-Under-Curve* (PR AUC). The confusion matrix is a two-dimensional matrix in which each element c_{ij} represents the count how often the model predicted class i when the actual class according to the correct label was j. In our setting, we assign each bounding box a to a single class, not multiple classes. We assign a to the class $\arg\max_{P \in \mathcal{P}_1} \mathcal{G}(P)(\mathcal{G}(a))$. We refer to predictions which correctly assign a class to an example as a *true positives*, similarly a *true negative* is when the classifier correctly does not assign the class to a particular example. A *false positive* is an incorrect prediction in which the classifier assigned the class incorrectly to the example, similarly, a *false negative* is the incorrect prediction in which the classifier did not assign the class to the example, when it should have. The confusion matrix gives rise to many different measures. For example, *accuracy* is intuitively defined as

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{c_{11} + c_{22}}{c_{11} + c_{22} + c_{12} + c_{12}}$$

We will not use accuracy as a measure here as there are two problems in using accuracy as a measure in our setting. The first is that it leads to the problem that a small increase in accuracy can be a great improvement in unbalanced datasets. The second problem is that when we see an improvement in the measure it tells us little about the improvement. A common example used to describe this problem has to do with cancer classification. Let us assume that we are making a classifier to detect cancer cells on images and that most pictures do not contain any cancer cells. Thus, the number of true negatives is very high but we do not want to overlook images which potentially contain cancer cells, so the cost of false negative is high. The cost of a false positive is also high, but not as high as an untreated cancer patient. Now, a classifier might have a high accuracy if it classified everyone as not having cancer and when the accuracy increases, we want to know if it is making progress in predicting true positives and how it is reflected w.r.t. false positives and false negatives. That is, we want to be able to characterise different improvements of the classifier. Take for example a classifier with the following counts, $TP = 500$, $FP = 100$, $FN = 400$, $TN = 9000$. The accuracy is

$\frac{TP+TN}{TP+TN+FP+FN} = \frac{500+9000}{500+9000+100+400} = \frac{9500}{10000} = 0.95$. Let us now consider a different classifier with the following counts, $TP = 600$, $FP = 100$, $FN = 300$, $TN = 8900$. This classifier has the same accuracy as the previous classifier but we would much rather choose this classifier as it has less false negatives. The following definitions give us the tools to characterise the differences between these two classifiers.

Definition 31 (Precision, Recall and $F1$ score)**.** *Given a confusion matrix for k classes, the classification* precision *of class $i \in [k]$ is*

$$precision_i = \frac{c_{ii}}{\sum_{j=1}^{k} c_{ij}}$$

The recall *of a class i is*

$$recall_i = \frac{c_{ii}}{\sum_{j=1}^{k} c_{ji}}$$

The F1 score of a class is then defined as

$$F1_i = \frac{2}{\frac{1}{recall_i} + \frac{1}{precision_i}} = A_{har}(recall_i, precision_i)$$

If a model predicts "cancer" for everything in the dataset, then the precision is the fraction "cancers" in the whole dataset. In this case the recall will be 1, as the model was able to predict all of the cancers.

The first classifier has precision and recall

$$precision_1 = \frac{c_{11}}{\sum_{j=1}^{2} c_{1j}} = \frac{500}{500+100} = 0.83$$

$$recall_1 = \frac{c_{11}}{\sum_{j=1}^{k} c_{j1}} = \frac{500}{500+400} = 0.56$$

while the second classifier has precision $\frac{600}{600+100} = 0.85$ and recall $\frac{600}{600+300} = 0.67$. Of course, these measures decreased w.r.t. the complement class, but precision and recall further allow us to characterise that change for the compliment class if we so want. Essentially, accuracy is a measure over all classes whilst recall and precision is a measure over individual classes and thus generalizes well to settings which have multiple classes. As the indoor dataset has 11 classes and we want to have the possibility of reporting a single measure for each model we define two aggregates over $F1$ scores.

Definition 32 (Micro and macro $F1$ score)**.** *The* macro average F1 score *of a classifier over k classes is*

$$F1_{macro} = A_{har}(A_{ari}(recall_1, \ldots, recall_k), A_{ari}(precision_1, \ldots, precision_k)$$

The micro average F1 score *is*

$$F1_{micro} = A_{har}\left(\frac{\sum_{i=1}^{k} c_{ii}}{\sum_{l=1}^{k}\sum_{j=1}^{k} c_{lk}}, \frac{\sum_{i=1}^{k} c_{ii}}{\sum_{l=1}^{k}\sum_{j=1}^{k} c_{kl}}\right) = \frac{\sum_{i=1}^{k} c_{ii}}{\sum_{l=1}^{k}\sum_{j=1}^{k} c_{lk}} = accuracy$$

$F1_{micro} = accuracy$ does not hold in a multi-label setting, in which each element can be assigned multiple labels. In the following section we report the $F1_{macro}$ score aggregate over the object types and report the $F1$ score for the "part of" relation.

The $F1$ score gives a good idea of how the model is performing w.r.t. the binary classification task and relation prediction task but we are also interested in knowing how well the model satisfies the logical constraints.

Definition 33 (Grounding satisfaction). *The grounding satisfaction of sentence φ is $\mathcal{G}(\varphi)$ over the whole testing set.*

Essentially, the grounding satisfaction for some sentence φ is just 1 minus the loss of $\mathcal{G}(\varphi)$ and therefore tells us something about the loss of the model. We evaluate the grounding satisfaction for each logical constraint and compute the mean of the logical constraints based on the grouping of the 8 sentences above. We report the grounding satisfaction of a model as the mean grounding satisfaction over all groups. That is, the sum of the grounding satisfaction per group divided by the number of groups. Note that this treatment places high numerical salience on sentences which are not schemas, f.ex. the irreflexivity axiom

Example 11 (Computing the grounding satisfaction). *Let $\Gamma = \{\varphi, \psi_1, \psi_2\}$ and $\mathcal{G}(\varphi) = 0.5$, $\mathcal{G}(\psi_1) = 0.0$ and $\mathcal{G}(\psi_2) = 1.0$, where ψ_1 and ψ_2 are sentences based on the same schema. The grounding satisfaction for Γ is $\frac{0.5+\frac{0.0+1.0}{2}}{2} = 0.5$.*

Since the $F1$ score make our predictions binary, in the sense that we assign each bounding box a single label, we would also like to know how well the sentences are satisfied, given those predictions. Thus, we want a measure which is based on the class assignments using truth values $\{1, 0\}$, not the fuzzy predicate values. Furthermore, most of the constraints are implications in which the premise is false for most elements of the dataset as it addresses very particular elements. We want to be able to see the satisfaction given that the premise is true. This motivates the definition of *modus ponens satisfaction* in which we only evaluate the constraints *after class assignment* and when the premise is true, to compute satisfaction. The name comes from the fact that we are trying to measure how often given $\varphi(x)$ and $\varphi(x) \implies \psi(x)$ is $\psi(x)$ true.

Definition 34 (Modus ponens satisfaction). *For a sentence $\varphi(x) \implies \psi(x)$, the* modus ponens satisfaction *of this sentence over T is the fraction*

$$mp(\varphi(x) \implies \psi(x)) = \frac{|\{a \mid a \in T \textit{ and } \varphi(a) \wedge \psi(a) \textit{ is true}\}|}{|\{a \mid a \in T \textit{ and } \varphi(a) \textit{ is true}\}|}$$

For constraints which are not implications we consider the premise always true.

We have now introduced two types of measures to evaluate the performance of the model. First, the $F1$ score to measure correct classification and we will report the $F1$ score for the "part of" relation and macro $F1$ score for the types classification. We then report two measures the *grounding satisfaction* and the *modus ponens satisfaction* which measure how well the logical constraints are satisfied. We have now a clear idea of how different instantiations and logical theories can be expressed in this framework and how the performance of these combinations can be measured. In the next section we state a number of hypotheses about the framework and how we can test them.

2.4 Hypotheses

Serafini et al. (2017) find through experimentation that Real Logic implemented with the LTN performs better than a baseline model in terms of binary classification and relation prediction and predictions are robust when the data are noisy. Donadello et al. (2017) only makes the first claim but we plan to replicate both experiments. In addition to replicating these original experiments, we will do more experiments on different instantiations of the framework. We will explore different t-norms, different predicate implementations and explore the variation of $F1$ scores of the models.

Recall that we refer to $\mathcal{G}_{LTN,T_G,har}$ as $\mathcal{G}$ or $\mathcal{G}_{T_G}$. In this section, we will go through each hypothesis, motivate it and explain how we will test and measure it. We will start with two fundamental experiments which have an effect on how the following experiments are done.

Hypothesis 1. *The Gödel t-norm will outperform the Łukasiewicz and plain product t-norm.*

As mentioned in section 1.2, if we do not use the log product t-norm (resp. s-norm) we expect large conjunction (resp. disjunction) of product t-norms (resp. s-norms) to result in numerical underflow, making both unusable with gradient descent. We also mentioned that the Łukasiewicz will have gradient $\mathbf{0}$ over most of the input space, also making it unusable with gradient descent. We will see that this is indeed the case. To test this hypothesis we will train the following instantiations $\mathcal{G}_{T_G}$, $\mathcal{G}_{T_Ł}$ and $\mathcal{G}_{T_P}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$ as we want to see whether either conjunction length will work for Łukasiewicz and product t-norm. Originally, before implementing the log product t-norm (and s-norm), the conclusion of this experiment forced us to do the following experiments based on the Gödel t-norm. Thus, most of the following experiments are based on the Gödel t-norm. We will compare the Gödel t-norm (and s-norm) to the log product t-norm (and s-norm) in hypothesis 7. In this experiment, we will see that adding logical constraints can degrade the performance of the Gödel t-norm, which we will explore closer in hypothesis 3. We accept this hypothesis if the $F1$ score of the $\mathcal{G}_{T_G}$ is consistently higher than the others.

Hypothesis 2. *Using the harmonic mean instead of the Gödel and Łukasiewicz t-norms over the set of clauses will result in increased binary classification and relation prediction performance.*

That is, instead of evaluating the loss based on $T(\{\varphi \mid \varphi \in \mathcal{K}\})$ then instead we evaluate $A_{har}(\{\varphi \mid \varphi \in \mathcal{K}\})$. We do this experiment for two reasons. First, so that we can replicate the experiments reported by Serafini et al. (2017); Donadello et al. (2017) as their findings are based on this evaluation. Second, for the Gödel s-norm we suspect that evaluating all clauses at the same will lead to better performance, so we use the harmonic mean as a tool to that goal. By using this method we achieve considerably better performance for the Gödel and Łukasiewicz s-norms and we are able to replicate their findings, with some hesitation. We denote the grounding in which a t-norm T is substituted by an aggregation function A as $\mathcal{G}_{T/A}$, f.ex. $\mathcal{G}_{T_G/har}$. To test this hypothesis we will replace the Łukasiewicz and the Gödel t-norm with the harmonic mean, both in fully constrained and unconstrained setting. That is, we will train $\mathcal{G}_{T_G/har}$ and $\mathcal{G}_{T_Ł/har}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$. We will also see that $\mathcal{G}_{T_Ł/har}$ has high grounding satisfiability but low modus ponens satisfiability, which is not what we want for this setting, yet it still outperforms $\mathcal{G}_{T_G/har}$. We will evaluate performance using $F1$ scores and compare each s-norm separately. We will further discuss why this substitution improves predictions in section 4.

Hypothesis 3. *Applying logical constraints in addition to positive and negative examples will increase binary classification and relation prediction performance.*

Resolving the classification task and relation prediction task simultaneously or dependently will lead to increased performance since these two are related. To test this we will train the same model instantiation using four different logical theories. The instantiation is $\mathcal{G}$ and the four sets of constraints are $\mathcal{K}_f$, $\mathcal{K}_{disj,one,parts,wholes}$, $\mathcal{K}_{irr,asym}$ and $\mathcal{K}_{all}$. We will start by comparing our results to the results of Serafini et al. (2017). We will compare the performance of these experiments to the baselines introduced before and measure performance w.r.t. the $F1$ score in the classification task and relation prediction. We also compare the baselines to the results of hypothesis 2. As mentioned in hypothesis 1 this experiment only uses the Gödel t-norm but this limitation turns out to be quite educative as we will discuss in section 4.1. We will see that, in general, the constraints lead to better performance and further explore when they might not in section 4.2 and discuss their impact on the task in section 4.4. We accept this hypothesis if constraining the model consistently leads to better performance.

Hypothesis 4. *Applying logical constraints in addition to positive and negative examples offers less variation in performance.*

That is, we expect the logical constraints to be less disrupted by initial settings of the model and converge to similar predictions regardless of the initial conditions. Here we train 20 $\mathcal{G}$ instantiations using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$, 40 models in total. With the results of hypothesis 2 in mind, we also train 10 $\mathcal{G}_{T_G/har}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$, 20 models in total. We will see that constraints have the adverse effect for the "part of" relation, standard deviation increases with constraints. Similarly, we will see that $\mathcal{G}$ has higher standard deviation than $\mathcal{G}_{T_G/har}$ and in sections 4.2 and 4.1 we will explain these results. We evaluate all models using the $F1$ score and report the mean and standard deviation.

Hypothesis 5. *Applying logical constraints in addition to positive and negative examples offers robustness in performance when trained on noisy data.*

Here, robustness is interpreted as maintaining higher performance in light of noisy data compared to an unconstrained baseline. To evaluate the model under noise, incorrect labelling is introduced in the training data by sampling k-%, $k \in \{10, 20, 30, 40\}$ per cent of examples of each class and replace those labels with an incorrect label, sampled uniformly from the other examples. This ensures that each class is affected equally by the noise and the distribution of classes remains roughly the same. Initially, this experiment was done with $\mathcal{G}$ but the performance showed no particular pattern and quickly dropped to random guessing. We will thus do this experiment in the same way as presented by Donadello et al. (2017) using $\mathcal{G}_{T_G/har}$ and both $\mathcal{K}_f$ and $\mathcal{K}_{all}$ on the noisy data and measure the $F1$ score. We expect performance to drop w.r.t. $F1$ scores for both models but we will accept this hypothesis if we see the constrained model consistently outperforming the unconstrained model with increased noise. We will see that the performance of the models do not drop as much as one would expect and discuss this in section 4.2.

Hypothesis 6. *The NTN will perform equally well as the LTN.*

As previously discussed in section 1.4, a large proportion of the parameters of the LTN are redundant and initially we did not expect the added expressivity of the LTN to result in better performance. We train two instantiations, $\mathcal{G}_{NTN,T_G}$ and $\mathcal{G}_{NTN,T_G/har}$ using $\mathcal{K}_{all}$ and compare with $\mathcal{G}$ and $\mathcal{G}_{T_G/har}$ from the results of hypothesis 4. In order to get more conclusive results, we train 10 of each instantiation. We will see that in this setting the LTN does indeed perform better than the NTN and discuss why this is the case in section 4.3.

Hypothesis 7. *The log product norms will outperform the Gödel norms.*

As previously discussed in section 1.2 the partial derivative of the Gödel norms is 0 for all but one dimension while the log product norms have a non-zero partial derivative for all dimensions. We expect this property to be beneficial during training and allows the model to train all clauses at the same time. To test this hypothesis we will train 10 $\mathcal{G}_{\log(T_P)}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$ and compare with results from hypotheses 3 and 4. We will see that this property is even more important than we initially expected and discuss why this is the case in section 4.1.

Hypothesis 8. *The OWA operator which takes into account the worst 50% performing elements will perform better than the arithmetic mean.*

We initially expected the knowledge base to be satisfiable and that the OWA operator would allow us to get even more precise results by increasing the loss w.r.t. poorly satisfiable examples. When observing that the knowledge base was far from being satisfiable we decided not to perform this experiment. We will keep this hypothesis here but in section 4.1 we will discuss how this approach might lead to similar issues as we observed with the Gödel norms.

Chapter 3

Experimental results

In this section, we will present results based on the hypothesis introduced in the previous section and begin each discussion by stating the hypothesis again.

Hypothesis 1. *The Gödel t-norm will outperform the Łukasiewicz and plain product t-norm.*

As previously stated, we trained $\mathcal{G}_{T_G}$, $\mathcal{G}_{T_Ł}$ and $\mathcal{G}_{T_P}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$. In figure 3.1 we can see that, indeed, only the Gödel t-norm produces predictions better than random guessing, thus we accept our hypothesis. This was also visible when training the model as the loss never changed and the predictions are based on random weight initializations. Additionally, we see that $\mathcal{G}_{T_G}$ tends to perform better given fewer constraints, and we will explore this result closer in hypothesis 3 when we measure the effectiveness of constraints and find out that this is indeed a result of the Gödel t-norm.

Hypothesis 2. *Using the harmonic mean instead of a t-norm over the set of clauses will result in increased classification and relation prediction performance.*

As mentioned in the previous section, in these experiments we use the harmonic mean instead of the t-norm when evaluating the clauses. In figure 3.2 we can see using the harmonic mean instead of both t-norms greatly improves performance in both tasks. If we compare these results to the results of hypothesis 1, an $F1$ score of 0.538 in classification and 0.311 in relation prediction we can see that all models here outperform these results.

Similarly, we can see in figure 3.3 that this substitution also greatly improves both grounding satisfaction and modus ponens satisfaction. We can also see the discrepancy between the grounding satisfaction and the modus ponens satisfaction caused by the easily satisfiable Łukasiewicz s-norm, most salient in $\mathcal{K}_f$. This further implies that the poor performance seen in result 2 is somewhat caused by the Gödel t-norm and Łukasiewicz t-norms. We can thus accept that this hypothesis is true and we will discuss the reasons for this increased performance in section 4.1.

Hypothesis 3. *Applying logical constraints in addition to positive and negative examples will increase classification and relation prediction performance.*

This is the core claim presented by Serafini et al. (2017) and Donadello et al. (2017)[1]. Serafini et al. (2017) report results for $\mathcal{G}_{LTN,T_G/har,har}$, using the substitution from the previous

[1]This claim is not explicitily stated, but it

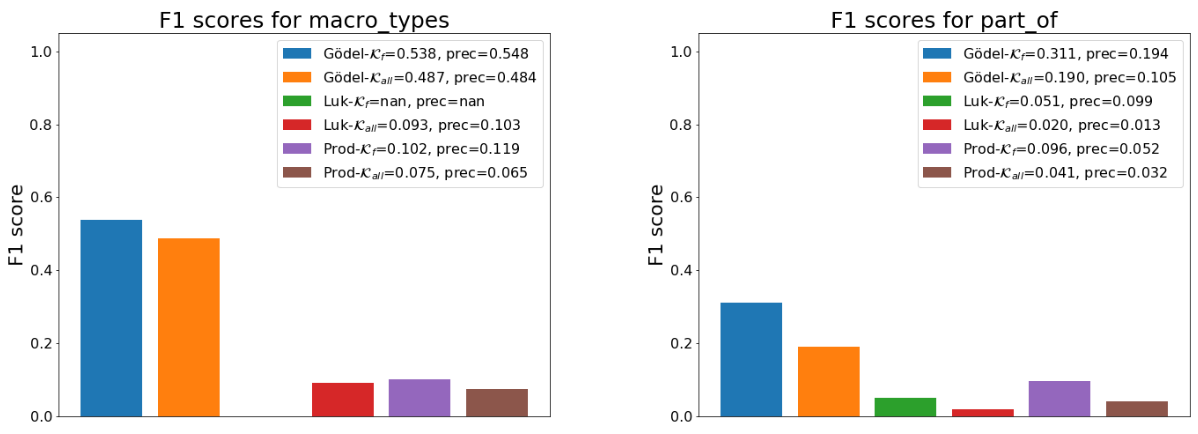


Figure 3.1: Left: The macro $F1$ scores in the classification task for $\mathcal{G}_{T_G}$, and $\mathcal{G}_{T_P}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$. Right: Results for the "part of" relation in the same experiment. We can see that only the Gödel t-norm produces any results better than random guessing and that more constraints give worse results. The "nan" is because the model never predicted the "part of" relation, causing division by 0.

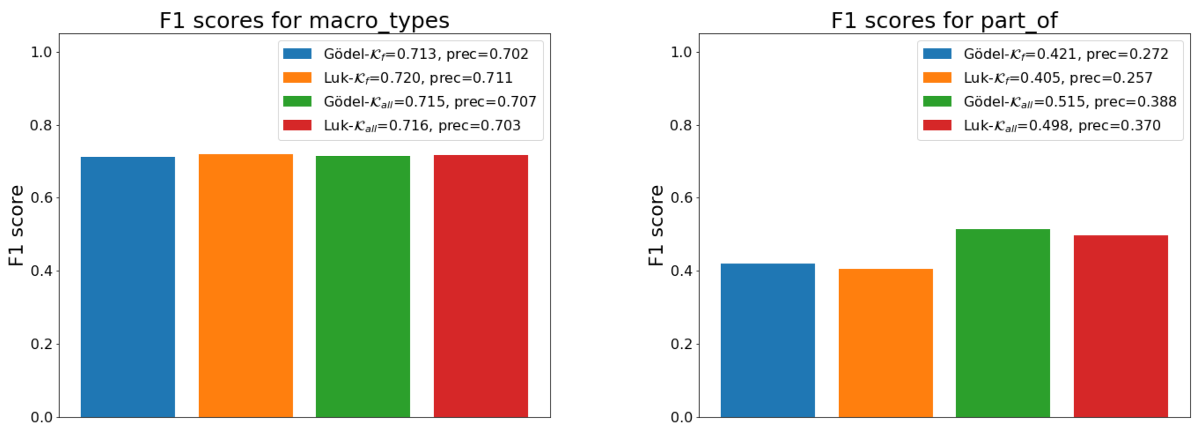


Figure 3.2: $F1$ scores for $\mathcal{G}_{T_G/har}$ and $\mathcal{G}_{T_L/har}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$. We can see that using the harmonic mean instead of both t-norms greatly improves performance in both tasks, compared to figure 3.1 and that the constraints do improve performance.

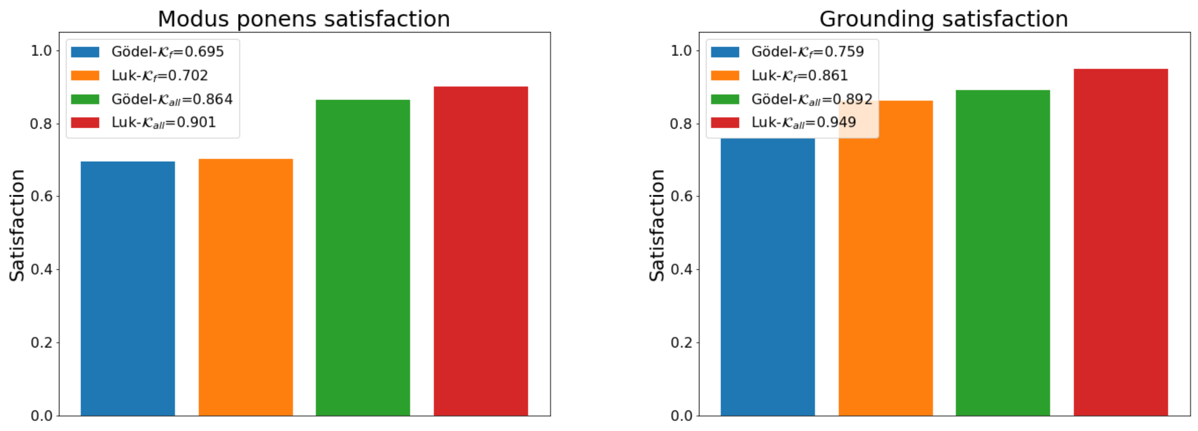


Figure 3.3: Modus ponens and grounding satisfaction for $\mathcal{G}_{T_G/har}$ and $\mathcal{G}_{T_{\text{Ł}}/har}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$. We can see that using the harmonic mean instead of both t-norms greatly improves satisfaction and the $\mathcal{G}_{T_{\text{Ł}}/har}$ outperforms $\mathcal{G}_{T_G/har}$. We can also see the difference between the modus ponens and grounding satisfaction of $\mathcal{G}_{T_{\text{Ł}}/har}$ using $\mathcal{K}_f$.

experiment, and $\mathcal{K}_{all}$ along with $\mathcal{K}_f$. They report the (simple) average $F1$ score of the classes, an $F1$ score of 0.706 for the constrained model and 0.700 for the unconstrained model. For the "part of" relation, they report an $F1$ score of 0.750 for the constrained model and 0.759 for the unconstrained model. In both cases, they beat the baseline, which has an $F1$ score of 0.658 for the classes and 0.663 for the "part of" relation. In their experiments, they do not limit themselves to only indoor objects, but they use the whole dataset, thus we cannot expect to get the exact same results.

In figure 3.4 we can see four different baselines. Two of which are constrained with the type compatibility which only affects the "part of" relation prediction. The constrained models are denoted with "Con" and unconstrained with "Uncon". The constraint leads to worse performance w.r.t. the $F1$ score since the recall drops by a large margin but precision increases. Furthermore, we report both baselines, one which predicts the background, denoted with "bg" and another which does not. The baselines were evaluated with a few different inclusion thresholds for the "part of" relation. An inclusion threshold of 0.6 was ultimately selected based on best performance.

Serafini et al. (2017) report the constrained baseline which will predict the background, which in our setting of only indoor objects results in an $F1$ score of 0.671 for the classification and 0.434 for the "part of" relation. From these results, we can see that none of the models presented here beat any of the baselines. If we compare with the previous result in hypothesis 2, we achieved an $F1$ score of 0.716 in the classification task and 0.515 in the relation prediction task using $\mathcal{G}_{T_G/har}$ and $\mathcal{K}_{all}$, we beat the baseline reported by Serafini et al. (2017) but we do not beat the best performing baseline for the "part of" relation. We are thus able to reproduce the same results as reported by Serafini et al. (2017).

Let us now try to decide whether the hypothesis is correct or not. Are the logical constraints able to improve predictions? If we look closer at $\mathcal{G}$ trained using $\mathcal{K}_{irr,asym}$ in figure 3.4, we can see that this model seems to be performing worse than the others in terms of macro $F1$ score. Though if we look at figure 3.5 we can see that these predictions are becoming more log-

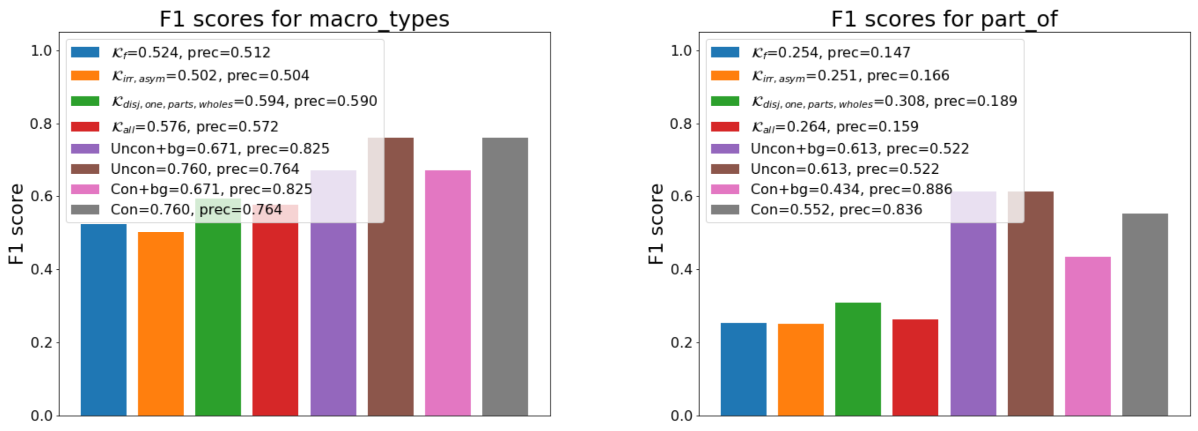


Figure 3.4: $F1$ scores for $\mathcal{G}$ using $\mathcal{K}_f$, $\mathcal{K}_{irr,asym}$, $\mathcal{K}_{disj,one,parts,wholes}$, and $\mathcal{K}_{all}$ and all baselines. We can see that no model is able to beat the baselines. If these results are compared to 2 we indeed to beat the Con+bg baseline as reported in the original experiments.

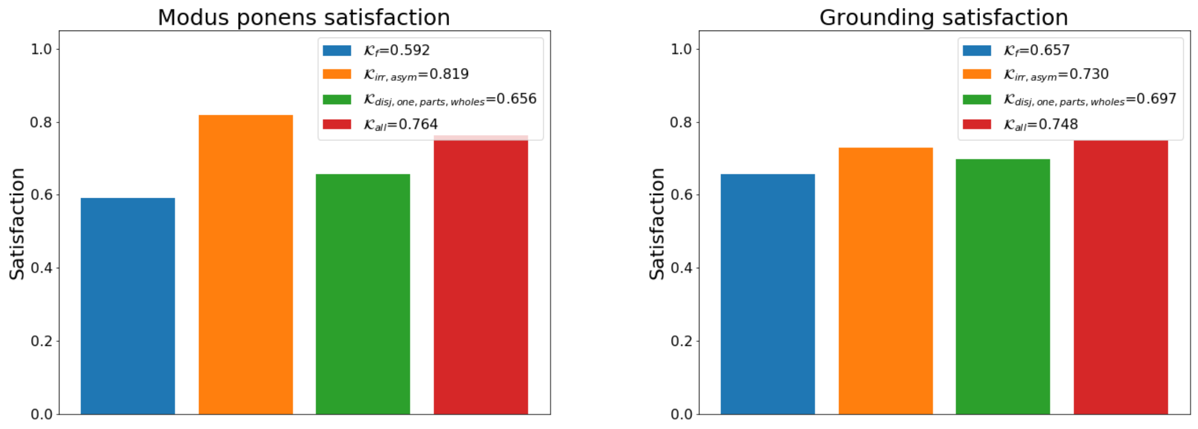
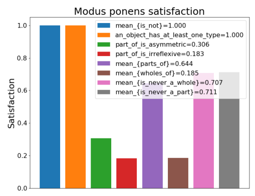
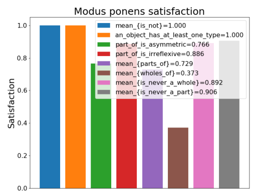


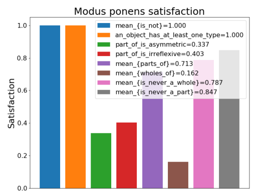
Figure 3.5: Left: Modus ponens satisfaction of $\mathcal{G}_{LTN,T_G,har}$ along trained with $\mathcal{K}_f$, $\mathcal{K}_{irr,asym}$, $\mathcal{K}_{disj,one,parts,wholes}$, and $\mathcal{K}_{all}$. Right: Grounding satisfaction for the same experiment.



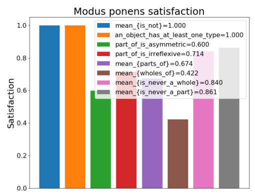
(a) $\mathcal{G}$ using $\mathcal{K}_f$



(b) $\mathcal{K}_{irr,asym}$



(c) $\mathcal{K}_{disj,one,parts,wholes}$



(d) $\mathcal{K}_{all}$

Figure 3.6: Detailed modus ponens satisfaction of $\mathcal{G}$ with different constraints.

ically consistent. The satisfaction difference between $\mathcal{K}_{irr,asym}$ and $\mathcal{K}_{disj,one,parts,wholes}$ seems to indicate that the latter model is doing considerably worse, but we refer to section 2.3 where we mention that the numerical salience of the irreflexivity and asymmetry constraint is more pronounced in this measure. In figures 3.6a,3.6b,3.6c and 3.6d we can see the breakdown of the modus ponens satisfaction. In general (satisfaction for the whole model), we see that the constraints do indeed increase the satisfaction, implying that the predictions are more logically consistent. But then why does $\mathcal{K}_{irr,asym}$ perform this poorly w.r.t. the $F1$ score? In section 4.1 we argue that when computing the gradient for some predicate, it needs to be computed w.r.t. all atoms of the sample which refer to that predicate and the Gödel norms do not do this and only update a single literal in one clause. If this is not done, then the training of that predicate is unstable which leads to the model not being able to train the other predicates sufficiently. This effect is particularly noticeable when we add the irreflexivity and asymmetry clause as they both refer to the same predicate. When adding the other clauses, we refer more often to the unary predicates. In section 4.4 we will discuss the effects of applying logical constraints and consider a different setting in which the effects of constraints might be even more effective.

Hypothesis 4. *Applying logical constraints in addition to positive and negative examples offers less variation in performance.*

$\mathcal{G}$	$\mathcal{K}_{all}$	$\mathcal{K}_f$
"part of"	0.216±0.045	0.282±0.042
macro $F1$	0.509±0.048	0.572±0.034

Table 3.1: Variation results for $\mathcal{G}$. We can see that the standard deviation for the "part of" relation is similar for the constrained and the unconstrained model. The standard deviation of the macro $F1$ score seems to be less for the unconstrained model.

$\mathcal{G}_{T_G/har}$	$\mathcal{K}_{all}$	$\mathcal{K}_f$
"part of"	0.495±0.046	0.433±0.024
macro $F1$	0.719±0.004	0.722±0.004

Table 3.2: Variation results for $\mathcal{G}_{T_G/har}$. We can see that the standard deviation for the "part of" relation is considerably lower for the unconstrained model and the macro $F1$ score has very low standard deviation for in both settings.

Here we trained 20 $\mathcal{G}$ instantiations and 10 $\mathcal{G}_{T_G/har}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$ and we hoped to see more consistent performance w.r.t. the variation of $F1$ score for the constrained model, we see the opposite in table 3.1. We see that for $\mathcal{G}$ that the standard deviation for the "part of" relation is similar for $\mathcal{K}_f$ and $\mathcal{K}_{all}$ and the standard deviation of the macro $F1$ score seems to be less for $\mathcal{K}_f$. We see similar results in table 3.2 for $\mathcal{G}_{T_G/har}$. $\mathcal{K}_f$ has lower standard deviation than $\mathcal{K}_{all}$ for the "part of" relation while both models have very low standard deviation in the $F1$ macro score. We can safely say that this hypothesis is not correct and we will attempt to explain this result in section 4.2.

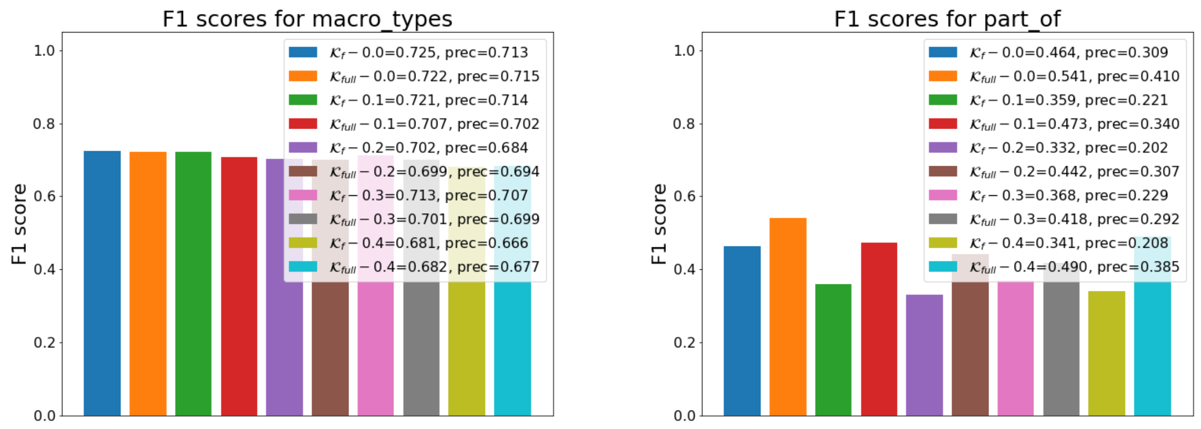


Figure 3.7: $F1$ scores for $\mathcal{G}_{T_G/har}$ using $\mathcal{K}_f$ and $\mathcal{K}_{all}$ in trained on dataset with increasing noise. We can see that the performance decreases less than expected in both tasks. For the relation prediction task we can see that $\mathcal{K}_{all}$ consistently outperforms $\mathcal{K}_f$.

Hypothesis 5. *Applying logical constraints in addition to positive and negative examples offers robustness in performance when trained on noisy data.*

Here we trained $\mathcal{G}_{T_G/har}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$ with increasing amount of noise. We refer to figure 3.7 for the $F1$ scores and we can see that indeed the fully-constrained models perform consistently better. Furthermore, we can see that performance decreases less than expected in both the classification task and relation prediction task. We can thus infer that the noise is not having much effect on the inner workings of the model. This is most likely due to the fact that the features the model uses to solve the tasks remain the best features even in light of more noise. We will discuss these results in section 4.2.

Hypothesis 6. *The NTN will perform equally well as the LTN.*

$\mathcal{G}_{NTN,T_G}$	$\mathcal{K}_{all}$	$\mathcal{K}_f$
"part of"	0.168±0.055	0.178±0.024
macro $F1$	0.215±0.078	0.290±0.058

Table 3.3: Variation results for $\mathcal{G}_{NTN,T_G}$. We can see that $\mathcal{G}_{NTN,T_G}$ performs worse than $\mathcal{G}$ (results in figure 3.1) and has higher standard deviation.

$\mathcal{G}_{NTN,T_G/har}$	$\mathcal{K}_{all}$	$\mathcal{K}_f$
"part of"	0.434±0.047	0.404±0.019
macro $F1$	0.720±0.007	0.720±0.005

Table 3.4: Variation results for $\mathcal{G}_{NTN,T_G/har}$. We can see that $\mathcal{G}_{NTN,T_G/har}$ performs worse than $\mathcal{G}_{T_G/har}$ (results in figure 3.2).

Here we compare $\mathcal{G}_{NTN,T_G}$, $\mathcal{G}_{NTN,T_G/har}$, $\mathcal{G}$ and $\mathcal{G}_{T_G/har}$ using $\mathcal{K}_{all}$. We can see the results in tables 3.3 and 3.4 which clearly show that the NTN does not perform as well as the LTN. We will argue in section 4.3 that this is because of bias in the data towards the extra expressivity the LTN offers in this task.

Hypothesis 7. *The log product norms will outperform the Gödel norms.*

To test this hypothesis we trained 10 $\mathcal{G}_{\log(T_P)}$ using both $\mathcal{K}_f$ and $\mathcal{K}_{all}$ want to compare with results from hypotheses 3 and 4. Let us first note that these are by far the best results.

$\mathcal{G}_{T_P}$	$\mathcal{K}_{all}$	$\mathcal{K}_f$
"part of"	0.586±0.029	0.452±0.020
macro $F1$	0.729±0.004	0.702±0.018

Table 3.5: Variation results for $\mathcal{G}_{T_P}$.

Compared to $\mathcal{G}_{T_G/har}$ which achieves 0.495 ± 0.046 for the "part of" relation and 0.719 ± 0.004 for the macro $F1$ score, thus we can safely say that this hypothesis is correct. If we compare these results to the baselines, we are still not able to beat the best baseline which achieves 0.613 and 0.760 $F1$ scores for the "part of" relation and types, respectively. We will try to explain why the product norms perform so well in section 4.1. We will also try to explain the high standard deviation in the constrained model in section 4.2.

Hypothesis 8. *The OWA operator which takes into accounts the worst 50% performing elements will perform better than the arithmetic mean.*

As previously mentioned this experiment was not performed and we will discuss why this hypothesis is likely to be false in section 4.1.

We have now gathered all the results and will discuss the theoretical results and the experimental results in the next section and try to understand better how the model works.

Chapter 4

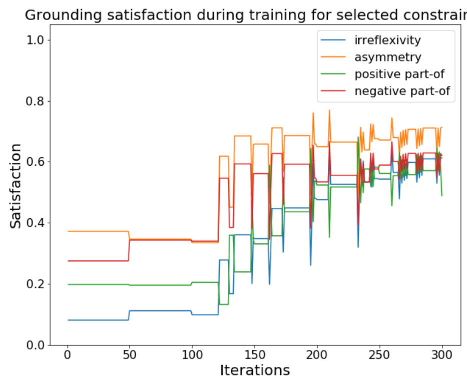
Discussion

In the following section, we discuss the theoretical and experimental results and try to answer the original question posed in this thesis. First, do all instantiations of RL perform equally when using a search procedure from the gradient descent family? We saw immediately in section 1.4 that this is not the case and verified this in result 1. We also noticed once in result 3 that when adding constraints, the $F1$ score degraded while satisfaction improved. We argue that this is because the Gödel t-norm does not consider all examples which refer to the same predicate when computing the gradient, also mentioned in result 2. This will lead us to the conclusion that the log-product norm is the best candidate for RL, as we also saw in result 7 and that the OWA hypothesis 8 is likely to be incorrect. Second, we will attempt to explain the variation seen in result 4 and noise performance results in 5. We believe both results are due to the biased features in addition to the universal quantification sampling. Third, is the additional expressivity of the LTN better than the NTN? Again, we saw immediately in section 1.4 that the LTN is able to express more complex function than the NTN, but contains redundancy, and in result 6 we saw that the LTN performs better than the NTN. As a consequence of this exploration, we also consider a different generalization of the LTN and the NTN which captures n-ary predicates in a more "multiplicative" way. Lastly, is RL able to improve logical consistency in predictions and improve classification? In result 3 we saw somewhat inconclusive results, but we refer to the above to explain the exception. We argue that RL is able to improve predictions but different instantiations handle contradictory examples in the data very differently.

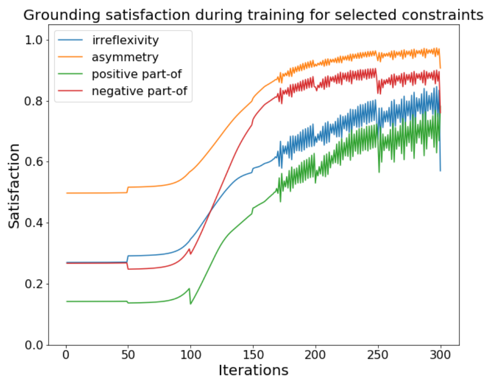
4.1 T-norms and aggregations

As mentioned in section 1.2, we want the gradient of our knowledge base to be non-zero over the whole input domain. The way our knowledge base is set up, using conjunction to join all the sentences together, it is paramount that $\nabla T \neq \mathbf{0}$. This is not the case for the Łukasiewicz t-norm and result 1 reflected this. We still saw the Łukasiewicz s-norm perform well in result 2 outperforming the Gödel s-norm. The Gödel s-norm only has a positive gradient over a single input, thus it will only direct gradient to a single literal, one for each clause, in each iteration. This indicates that it is better to compute the gradient w.r.t.all elements of the disjunction. Why is this?

Let us now look closely at the performance of the Gödel t-norm, in particular in result 3. In figure 4.1a we can see grounding satisfiability of a few selected clauses during training for



(a) Grounding satisfaction for $\forall x(\neg R(x,x))$, $\forall x(R(x,y) \implies \neg R(y,x))$, φ_R and $\varphi_{\neg R}$ using the Gödel t-norm during training on $\mathcal{K}_{irr,asym}$. We can see that in some iterations one clause goes up when another goes down.



(b) Grounding satisfaction for the same sentences as 4.1a but rather using the harmonic mean instead of the Gödel t-norm. In this figure we can see that the grounding satisfaction of the clauses move together. We can also see the drop when a new sample for the universal quantifier is taken (every 50 iterations).

$\mathcal{G}$ for $\mathcal{K}_{irr,asym}$. Note that when the model is updating other clauses, the satisfaction of the clauses we track remains unchanged. It seems that in almost every iteration one of the four clauses of positive examples of "part of", negative examples of "part of", irreflexivity and asymmetry is the least satisfied clause and the Gödel t-norm updates it. This will affect the satisfiability of the other clauses since they are all based on the same predicate, sometimes making them jump over one another. If we were to compute the gradient for all literals which refer to the "part of" predicate, we can guarantee that in the next iteration that the total loss w.r.t.that predicate will decrease or stay the same. If we do not do this, we cannot guarantee that the total loss w.r.t.predicate will go down. See figure 4.1b for grounding satisfiability of a few selected clauses during training for $\mathcal{G}_{T_G/har}$ for $\mathcal{K}_{irr,asym}$. Consider we define two clauses which contradict one another, $P(a) \wedge \neg P(a)$ and try to satisfy them using the Gödel t-norm (P contains some parameters). When either goes over the 0.5 mark, the Gödel t-norm will start updating the other one. If we consider both at the same time then we add the gradients together, partially cancelling each other out. Thus in these cases, we would expect the model to converge to 0.5. We suspect that the data is somewhat contradictory as we are never able to find a grounding which satisfies our data.

This explains why we saw better performance using the harmonic mean instead of the Gödel t-norm. The harmonic mean computes the gradient for all the clauses, which implies that we compute the gradient w.r.t.all predicates (in our case), but not necessarily all atoms which refer to those predicates. This leads to better performance for all predicates. This also explains why the Łukasiewicz s-norm performed better than the Gödel s-norm. Because then we compute the gradient for every atom which refers to every single predicate, by computing the gradient for every atom. This also points to the fact that the OWA experiment is fundamentally flawed as the setup we consider is susceptible to the same scenario.

Since the product norms both have a non-zero gradient for every input it is the best choice and result 7 shows that. Furthermore, the product norm guarantees that atleast a single element in a disjunction is true if the whole disjunction is true. The Łukasiewicz s-norm does not guarantee that.

We do not report any results for experimentation on aggregation functions as they were not that relevant in our experiments, but one can think of a setting in which they will begin to have more effect. If our data were satisfiable, that is, we would be able to find a grounding which satisfies our data very close to 1 and each sample of the data were large, i.e. the universal quantifier is computed over many elements then we might easily overlook examples which are not satisfied. We might want to make those examples more salient in the universal quantification and at the same time keep the computation somewhat reasonable.

4.2 Biased features and variation

Let us now try to explain the high standard deviation of the "part of" relation, which only seems to increase when we add logical constraints, opposite to what we initially expected. We argue that the high standard deviation of the "part of" relation is due to biased features in addition to the sampling of the universal quantifier. We also suspect that the noise robustness is due to the biased features as the biased features continue to be the best indicator of the objects actual class, even in light of the noise.

Recall that the grounding for pairs of objects are concatenations of the unary features along with the inclusion ratio and we know from the baseline that the inclusion ratio is highly correlated with the "part of" relation. Thus when we sample data for the universal quantifiers we can expect each sample to rely differently on the inclusion ratio, thus resulting in different predictions for the "part of" relation during testing. Adding more logical constraints, in particular, irreflexivity and asymmetry make the model even more sensitive to these biased features by adding more seemingly contradictory examples (very high inclusion ratio but not a "part of") thus placing higher loss on the inclusion ratio. Thus, we expect that the model will be biased towards the last sample of the universal quantification before stopping training and the biased features only make this bias more salient in our setting. That is, we see the model overfit on the universal quantification sampling. This is not surprising when viewed in this way.

Let us now consider the unary predicates and recall that the features of the unary predicates are also biased towards the object's class, as the "catness" feature is very indicative of the object being "a cat". We saw that the standard deviation was high when using the Gödel t-norm and we suspect that was because the model never got around to training the unary predicates properly as it was constantly fighting the "part of" relation as we discussed in section 4.1. When using the harmonic mean we saw the standard deviation drop a lot. We suspect that this is simply due to the biased features and the fact that we do not train the unary predicates over seemingly contradictory examples like we did for the "part of" relation.

Let us now consider the noise experiment. We believe that the biased features remain the best indicator of an object actual class even when trained on noisy data. So when the model is evaluated on the testing data, it still expects "catness" to be a good indicator for "a cat" and will not stop considering that until it sees something else becomes a better predictor for "cat", and we would need a lot of noise or biased noise to do that. The noise experiment shows that the model, mostly, continued predicting based on the same features as it had before, as the biased features were more often right than wrong.

4.3 Expressivity of LTN

As mentioned in section 1.4, we saw that there is some redundancy in the LTN but at the same time it is able to express more complex functions than the NTN. We originally expected the LTN and the NTN to perform similarly but as we saw during experimentation the LTN performed consistently better than the NTN. The possibility of deciding a predicate based on a single term is clearly beneficial in this setting. In this setting objects considered to be ”wholes” (resp. ”parts”) are more commonly found as the first term and ”parts” (resp. ”wholes”) as the latter term in a positive (resp. negative) example of the ”part of” relation. Thus, we must reject our hypothesis and accept that the LTN does perform better than the NTN on this dataset and note that this is probably in all cases where a binary predicate is biased in this way.

Let us now consider how the LTN represents a ternary relation. For ternary relations, there is no direct interaction between features of all three terms in the form of multiplication, which Socher et al. (2013) argues is what makes the bilinear model (Jenatton et al., 2012; Sutskever et al., 2009), a precursor to the NTN, better than its predecessors. The LTN only defines multiplication based on two features. If one wants to add interaction between all three terms, an extra dimension needs to be added. If the LTN were to be generalized so that another dimension would be added then for $m = 3$, W would be a 3-D tensor of dimensions $3n \times 3n \times 3n$, assuming n dimensions for every term. The number of parameters of this implementation is roughly $O(n\alpha(P)^{\alpha(P)})$, where $\alpha(P)$ is the arity of P, compared to $O(n\alpha(P)^2)$ of the LTN. This generalization further increases redundancy and it would be interesting to compare this ternary generalization to the ternary generalization of the LTN and see if it is worth the exponential growth.

4.4 Logical constraints

In this section, we will discuss the effects of the logical constraints. We saw during the experiments that logical constraints made the predictions more logically consistent and we consider this a great success for RL, as this was not shown in either Serafini et al. (2017) nor Donadello et al. (2017). In addition to making the predictions more logically consistent we also saw that they improve classification, most notable in the $F1$ score of the ”part of” relation as can be seen in result 2, 3, 4 and 7. We saw in result 3 without the logical constraints the model made illogical predictions, most notably in the case of irreflexivity, and needed to be trained not to, by training on a particular class of negative examples. It also makes clear the requirement for the logical properties of the relation to be present as logical constraints during training, especially when dealing with biased features as we saw in the discussion above. That is, the relation needs to be trained w.r.t.the logical properties of the relation to counter-act biased features. That is, the most conclusive results are the ones based on the logical properties of the relation being modelled.

Let us consider also the logical constraints which describe rules among the entities. That is, we consider constraints like ”when we have a tail and we know that it is a part of something, that something is a cat”. How do these constraints improve classification? These constraints have the effect that the ”cat classifier” is provided with additional samples to train on. In addition to providing more examples to train on, these examples *might be labelled differently or not at all in the training data.* This is not the case in our dataset as all our data are correctly and completely labelled for our classifiers. We thus suspect this to be more beneficial when training on possibly incorrect data or incomplete data. This effect could be beneficial in the noise experiment, but the experiment would need to be done differently. At some point, we would need to stop training on the positive and negative examples which might contradict the logical rules, as the noisy data are incorrect and/or incomplete. That is, we would need to start by using the supervised training examples and then switch to an unsupervised setting by removing $\mathcal{K}_f$ from the set of constraints, otherwise, we will continue training on contradicting examples. This is called semi-supervised learning and we believe that the logical constraints could be even more beneficial in that setting.

Chapter 5

Conclusion

We have now seen how RL allows us to phrase problems in FOL. Allowing us to express problems of classification, regression, clustering, relation prediction in a uniform framework where logic and logical constraints are defined w.r.t.the data. We challenge the reader to consider how regression can be performed using RL[1].

We saw that different instantiations of RL are not all equivalent when using the gradient to approximate maximum satisfiability. In particular, the Łukasiewicz t-norm is ill-behaved in this setting and we cannot think of any reason to use it. The Łukasiewicz s-norm is better but it cannot guarantee that when it is evaluated as true, that a single element is true, which is a useful property in many modelling scenarios. We also saw the importance of using some method which computes the gradient over all atoms of a particular predicate, presenting an argument against the use of the Gödel t-norm and s-norm. The log-product norms have none of these weaknesses and perform best. When viewing ML problems in light of RL we can also see the log-product in many forms.

We compared the LTN and the NTN and found that indeed the LTN can express more functions than the NTN but argue that the LTN's generalization of the NTN does not capture n-ary predicates in a multiplicative way and consider a different generalization of the LTN or the NTN which can capture that. We also tested the LTN and NTN in an experimental setting and find that the LTN does indeed perform better, demonstrating that on this dataset the increased expressivity is beneficial.

Lastly, we saw how RL using LTN is able to improve the logical consistency of predictions as well as overall classification performance in SII. We noticed that enforcing logical properties of the relation is important to avoid inconsistent predictions but does not necessarily imply better classification performance. We saw that constraints which describe logical rules improve classification, but in this setting, the reward was limited and the model very unstable due to biased features.

There are many interesting questions still unanswered and we will now mention a few of them for future research. First, what implications does it have for the framework that we define grounding over pairs? If we view the framework from a machine learning stand-point the implications do not seem to be so serious but it is hard to justify this treatment from a logical perspective. There are also some philosophical implications, implying that the whole is more than just its parts. We, however, believe that this is the right way to go and should

[1]Hint: consider a binary predicate representing "equal", where equality is defined in terms of some distance, transform the distance to a value between 0 and 1 and consider the log-product t-norm.

be explored further. Second, RL needs to be evaluated in a setting which requires longer chains of reasoning and an analysis needs to be done on what the effects of different values of satisfaction have on these longer chains. The intuition is that the longer the reasoning chain is, the less can be guaranteed. Lastly, it would be interesting to see how well the model performs in a semi-supervised setting as well, as that setting much rather allows changes in logical constraints during training.

RL does indeed offer a novel approach in combining knowledge representation and machine learning allowing us to, intuitively, describe domains which have complex relations and incorporate uncertainty, all through the language of FOL. One then wonders how the same concept could be extended for different logics, for example, temporal modal logics and agent-based logics. We should put these logics to the test and combine them with data, leveraging the power of deduction and induction at the same time.

Bibliography

Bergmann, M. (2008). *An introduction to many-valued and fuzzy logic: semantics, algebras, and derivation systems*. Cambridge University Press.

Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., and Yuille, A. (2014). Detect what you can: Detecting and representing objects using holistic models and body parts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Donadello, I., Serafini, L., and Garcez, A. d. (2017). Logic tensor networks for semantic image interpretation. *arXiv preprint arXiv:1705.08968*.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Jenatton, R., Roux, N. L., Bordes, A., and Obozinski, G. R. (2012). A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, pages 3167–3175.

Klement, E. P., Mesiar, R., and Pap, E. (2004). Triangular norms. position paper i: basic analytical and algebraic properties. *Fuzzy Sets and Systems*, 143(1):5 – 26. Advances in Fuzzy Logic.

Novák, V. (1987). First-order fuzzy logic. *Studia Logica*, 46(1):87–109.

Serafini, L. and d'Avila Garcez, A. S. (2016). Learning and reasoning with logic tensor networks. In Adorni, G., Cagnoni, S., Gori, M., and Maratea, M., editors, *AI*IA 2016 Advances in Artificial Intelligence*, pages 334–348, Cham. Springer International Publishing.

Serafini, L., Donadello, I., and Garcez, A. d. (2017). Learning and reasoning in logic tensor networks: theory and application to semantic image interpretation. In *Proceedings of the Symposium on Applied Computing*, pages 125–130. ACM.

Serafini, L. and Garcez, A. d. (2016). Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*.

Socher, R., Chen, D., Manning, C. D., and Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Sutskever, I., Tenenbaum, J. B., and Salakhutdinov, R. R. (2009). Modelling relational data using bayesian clustered tensor factorization. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1821–1828. Curran Associates, Inc.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

van Dalen, D. (2004). *Logic and structure*. Springer.

Yager, R. R. (1993). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. In *Readings in Fuzzy Sets for Intelligent Systems*, pages 80–87. Elsevier.