Neural-Symbolic Reasoning Under Open-World and Closed-World Assumptions

Benedikt Wagner, Artur d'Avila Garcez

City, University of London, Northampton Square, London, EC1V0HB, United Kingdom

Abstract

Neural-Symbolic approaches are becoming increasingly prominent due to their ability to integrate knowledge and data. In this paper, we propose the iterative use of a neurosymbolic approach and evaluate its reasoning capability. We deploy the Logic Tensor Networks neurosymbolic approach iteratively and compare its reasoning capability with purely symbolic reasoning under closed-world and open-world assumptions. Reasoning capability is evaluated on two data sets, a family relationship task and a typical ontology reasoning data set. The use of an iterative neurosymbolic approach improves reasoning from an F1 score of 0.64 to 0.97 in one case, and from 0.60 to 0.88 in the other, which is higher than what was reported previously in the literature. Our results also show that an open-world neurosymbolic approach based on differentiable fuzzy logic can excel at recall, while a logical reasoner under a closed-world assumption can achieve high precision when the domain is under-specified.

Keywords

Neurosymbolic AI, Practical Reasoning, Open-World Assumption, Closed-World Assumption

1. Introduction

The integration of neural and symbolic AI approaches has become increasingly prominent [1, 2]. This has been motivated by the fact that neural and symbolic approaches are complementary: while deep learning has been widely successful across a large variety of data-driven applications, symbolic AI has been shown capable of overcoming limitations such as a need for large amounts of labelled data or lack of explainability.

Neural-Symbolic (NeSy) learning approaches can be weakly-supervised whereby only a limited number of data is required. Furthermore, NeSy systems can be in principle interpretable, allowing for mitigation of undesired system properties [3, 4]. Learning from data and knowledge using logical formulas that express background knowledge in the form of differentiable loss functions has been formalised as *Differentiable Fuzzy Logic* in [5]. Krieken et al. [5] provide a valuable contribution by analysing the effect of different specific fuzzy logic implications on reasoning and learning that indicate imbalances associated with different fuzzy operators. Bianchi and Hitzler [6] have approached the study of the reasoning capability of neural networks (NNs) in practical terms by measuring the F1 score of the deductive closure of a knowledge-base

CEUR Workshop Proceedings (CEUR-WS.org)

trained by the network. In this paper, we seek to expand both analyses by investigating how varying forms of background knowledge may impact the reasoning capability of a NeSy system. We investigate how the selection of background knowledge can affect reasoning in NNs. We further show that reasoning results may vary considerably depending on the assumption of a closed or open-world reasoning. In order to measure and compare results precisely, we also provide a definition for reasoning in NNs. By investigating such relevant examples of reasoning, we conclude that background knowledge as used traditionally by NeSy systems and symbolic machine learning may need to be formulated including the explicit use of negation for the purpose of improving reasoning capabilities.

In Section 2, we specify the relevant methods before defining practical reasoning and the querying method in Section 3. Section 4 provides practical examples and is followed by a conclusion in Section 5.

2. Differentiable Fuzzy Logic

Neural-Symbolic approaches that employ fuzzy logic with differentiable logical operators build differentiable loss functions from symbolic background knowledge specified in the language of first-order logic involving multiple logical formulas. Given data, these approaches use fuzzy logic to determine the truth-value of statements in first-order logic. The method works in a semi-supervised setting whereby the data can be unlabeled or sparsely labelled. Such method is exemplified by [7, 8, 9, 10, 11].

In the following, we provide a high-level overview of the Differentiable Fuzzy Logic (DFL) method before discussing in more detail the Logic Tensor Network (LTN) framework [8] as an instance of DFL. The semantics of DFL methods is derived from vector embeddings and functions whereby logic terms are interpreted in a real-valued vector space. Objects are d-dimensional vectors of real values in a domain. Predicates are mappings from these vectors to fuzzy truth-values in the interval [0,1], which can be represented by trainable NNs with learneable parameters θ. Varying the values of θ will derive different interpretations according to the embedding. To emphasize that symbols are interpreted based on their grounding onto real numbers, LTN refers to an interpretation as a *grounding*, denoted by $\mathcal{G}$. The truth-values of logical formulas containing quantifiers are determined by using an aggregation function $A : [0,1]^n \rightarrow [0,1]$, for example the harmonic mean. Fuzzy truth-values for atomic formulas are derived in the usual way, relying on fuzzy logical operators that preserve the properties of Boolean logic in the {0,1} extreme cases. These are aggregated into the satisfiability $\mathrm{Sat}(\mathcal{K})$ of the knowledge-base, i.e. the set of logical formulas. In LTN and other DFL methods, the parameters θ associated with the predicates' groundings are learned using a training signal such as the maximum satisfiability of the knowledge-base $\langle \mathcal{K}, \mathcal{G}_{\boldsymbol{\theta}} \rangle$. This subsequently reduces to an optimisation problem that can be solved using gradient-descent methods, such that:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\max} \ \mathrm{Sat}_A(\mathcal{G}_{\theta}(\mathcal{K})),$$

subject to an aggregation A of all logical formulas in $\mathcal{K}$; $\boldsymbol{\theta}^*$ is obtained by minimising a loss function that encompasses grounded knowledge $\langle \mathcal{K}, \mathcal{G}_{\boldsymbol{\theta}} \rangle$ and maximising satisfiability of $\mathcal{K}$.

Due to the fact that all fuzzy operators are differentiable, this optimisation can be achieved by gradient descent given knowledge and data.

Logic Tensor Networks [8, 12, 7] implement a many-valued first-order logic (FOL) language $\mathcal{L}$. The semantics for the connectives is defined according to fuzzy logic semantics: conjunctions are approximated by t-norms (e.g. $min(a, b)$), disjunctions by t-conorms (e.g. $max(a, b)$), negation by fuzzy negation (e.g. $1 - a$) and implication by fuzzy implications (e.g. $max(1 - a, b)$). Predicates are grounded as mappings onto the interval $[0, 1]$ representing the predicate's degree of truth given the data. Formulas in $\mathcal{L}$ facilitate the specification of relational knowledge using variables. For example, the atomic formula $\text{partOf}(a, X)$ with variable X and object a may indicate the objects that form the parts of object a. Additionally, since we are interested in learning and reasoning in real-world scenarios, exceptions to the rule may occur. Due to the fuzzy semantics adopted by the language, formulas may be partially true in case there are exceptions in the data.

In order to obtain a satisfiability level that serves as the optimisation criterion for training, it is necessary to aggregate all the formulas in the knowledge-base $\mathcal{K}$. Universal and existential quantifiers need to be expressed using the differentiable fuzzy logic $\mathcal{L}$. LTN therefore replaces X with all objects x_i ($x_i \in X$) in the data whenever a quantified free variable ($\forall x$) is part of a formula in $\mathcal{L}$. The truth-value of such formula is then obtained by aggregating the truth-values of each grounded object x_i. The universal quantifier and clause aggregation function adopted in the experiments reported in this paper is the generalised mean, also referred to as p-mean[1]. Existential quantification is not used in this paper. We adopt the product t-norm for conjunctions, its corresponding t-conorm for disjunctions, and the Reichenbach implication. As reported in [7], this setup is seen to be superior due to enhanced gradient propagation for learning.

3. Practical Reasoning

The term *reasoning* has been used frequently in the Machine Learning (ML) literature recently to denote various inference tasks. It seems desirable to specify the way that reasoning can be performed within ML. The most intuitive definition is that of a system capable of generating conclusions from a given knowledge-base. In a seminal paper about reasoning in ML [13], Leon Bottou argues that "instead of trying to bridge the gap between machine learning and sophisticated all-purpose inference mechanisms, we can algebraically enrich the set of manipulations applicable to training systems, and build reasoning capabilities from the ground up." More importantly, Bottou shows that there is continuity between algebraically rich inference systems such as logical or probabilistic systems and simple manipulations such as the mere concatenation of learning models.

By mapping symbolic knowledge into regular loss functions, DFL can make reasoning a part of learning as argued by Bottou. In this setting, one can analyse how knowledge and data follow specific rules of inference by observing how the loss term influences model learning while hoping to measure the reasoning capabilities that these optimisations enable, or one can extract

---

[1] $p\text{-}mean(x_1, \ldots, x_n) = \left(\frac{1}{n} \sum_{i=1}^{n} x_i^p \right)^{\frac{1}{p}}.$

knowledge explicitly from the trained model and evaluate by proxy the reasoning of the model by evaluating the reasoning and fidelity to the model of the extracted knowledge.

In the first option above, one evaluates both the network's forward (inference) and backward pass (learning). In the second option, the focus is exclusive on the forward pass. Furthermore, while the former is interested in specific examples (local reasoning), the latter takes the view of reasoning capabilities obtained through a set of examples (global reasoning). In this respect, while [5] analyses each fuzzy operator at a time w.r.t. their ability to translate rules of inference into the neural network (based on specific examples by analysing the partial derivatives of logical subformulas), in this paper, we take a broad perspective and evaluate reasoning w.r.t. a set of axioms being trained in a network through a set of examples. This motivates the following definition of practical reasoning.

Traditionally, reasoning in LTN is the process of searching for a set of parameters and grounding that satisfy a given logical proposition. Since, as discussed, in this paper we are concerned with evaluating the capabilities of a trained network, in what follows we define practical reasoning in LTN as akin to the task known as inference in Machine Learning. We therefore focus on reasoning given a single grounded theory $\langle \mathcal{K}, \mathcal{G}_{\boldsymbol{\theta}} \rangle$ under the usual assumption that some sound statistical evaluation, e.g. cross-validation or bootstrapping, has been applied to select the best available grounding (set of parameters).

Given a logical axiom Λ to be checked on a LTN, we generate symbolically a set of formulas $\{\phi\}_\Lambda$ such that $\Lambda \models \phi$, where $\models$ denotes logical consequence. We say that a LTN *proves* formula ϕ, written $\langle \mathcal{K}, \mathcal{G}_{\boldsymbol{\theta}} \rangle \vdash \phi$, if the satisfiability of ϕ given the best available grounding is greater than a predefined real number q ($0.5 \leq q < 1$), that is:

$$\langle \mathcal{K}, \mathcal{G}_{\boldsymbol{\theta}} \rangle \vdash \phi \text{ if } \mathcal{G}_\theta(\phi) > q.$$

We then check whether $\langle \mathcal{K}, \mathcal{G}_{\boldsymbol{\theta}} \rangle \vdash \phi$ for all $\phi \in \{\phi\}_\Lambda$, which gives us a measure of the neural network's reasoning capability. For example, to check whether a NN satisfies the axiom of Modus Ponens, we define $\Lambda = \{A, A \rightarrow B \models B\}$. Given atomic formulas $P(x)$ and $Q(x)$, suppose that $\langle \mathcal{K}, \mathcal{G}_{\boldsymbol{\theta}} \rangle \vdash P(a)$ and $\langle \mathcal{K}, \mathcal{G}_{\boldsymbol{\theta}} \rangle \vdash \forall x (P(x) \rightarrow Q(x))$. Thus, $Q(a)$ is added to the set of formulas to be checked for satisfiability in the NN, i.e. $Q(a) \in \{\phi\}_\Lambda$; $\{\phi\}_\Lambda$ may contain therefore all the instances that can be obtained given a set of axioms Λ. Examples of such axioms may include transitivity, monotonicity and symmetry of relations.

Given a set of examples $\boldsymbol{E} \subseteq \{\phi\}_\Lambda$, the relation $\vdash$ allows one to measure the reasoning capability of network $\langle \mathcal{K}, \mathcal{G}_{\boldsymbol{\theta}} \rangle$ w.r.t. Λ by measuring the level of satisfiability of all the cases in $\boldsymbol{E}$ in comparison with the satisfiability of all ϕ in $\{\phi\}_\Lambda$. Suppose for example that one wishes to evaluate in practice the reasoning capability of a given neural network that has been trained to recognise images with various bounding boxes in them. Suppose further that it is useful to check whether the network has learned that, whenever a bounding box is inside another bounding box, which in turn is inside another bounding box then the first bounding box must be inside that last bounding box. In other words, this property P of bounding boxes satisfies the transitivity relation, namely, $\phi = \forall X, Y, Z : (P(X,Y) \wedge P(Y,Z)) \rightarrow P(X,Z)$. A set of examples $\boldsymbol{E}$ may contain a number of bounding boxes $b_1, b_2, ... b_n$ satisfying instances of the above transitivity relation, such that given for example $P(b_1, b_2)$ and $P(b_2, b_3)$, one concludes that $P(b_1, b_3)$ holds. Checking the network's reasoning capability w.r.t. property P includes

checking whether $P(b_1, b_3)$ is satisfied by the network (soundness) but also quantifying the size of $\boldsymbol{E}$ as a proportion of $\{\phi\}_\Lambda$ (as a measure of completeness). $\{\phi\}_\Lambda$ should include all the cases of transitivity that can be derived symbolically by following a given proof procedure, not just the cases in the training set.

Notice how practical reasoning takes into account the ability of the system to interpolate (by measuring soundness) and extrapolate (by measuring completeness) since knowledge derived symbolically may be out of distribution. In what follows, the LTN network will be queried systematically to obtain a measure of its reasoning capability[2]. Querying using first-order logic (FOL) clauses in LTN is not only a post-hoc explanation in the traditional sense. We argue that querying should form an integral part of an iterative process allowing for incremental explanation and system improvement through the distillation of knowledge guided by reasoning from knowledge and data (c.f. Figure 1).

Given a trained network (a set of parameters θ), we compute the value of a grounding $\mathcal{G}_\theta(\phi_q)$ for a user-defined query ϕ_q [3]. Specifically, we save and reinstate the learned parameters stored in the LTN implementation resulting from $\theta^* = \arg\max_{\theta \in \Theta} \mathrm{Sat}_A \mathcal{G}_\theta(\mathcal{K})$. This also means that changes to the knowledge-base will instead start from saved state θ^*, which allows us to continually query and guide the learning process according to knowledge-base $\mathcal{K}_{new}$, an approach akin to that of continual learning.

A query is any logical formula expressed in FOL. Queries are evaluated by calculating the grounding $\mathcal{G}$ of any formula whose predicates are already grounded in the NN or by defining a new predicate in terms of existing predicates.

4. Quantitative Comparisons on Reasoning Tasks

We now evaluate comparatively the reasoning capabilities of NNs in the LTN framework on the experiments used by Bianchi and Hitzler [6] and Ebrahimi et al. [14]. In their experiments, redundant rules (formulas) are added to the knowledge-base. These are called redundant because they can be inferred symbolically from the existing knowledge-base. They conclude that redundancy produces an increase in the reasoning capability of NNs because the networks will see more data due to the additional rules. We demonstrate that only specific additional information is helpful in what concerns such improvement in reasoning.

We use the iterative continual framework of Figure 1 whereby a closed world or open world assumption can be implemented by adding specific propositions or quantified formulas to the knowledge-base. In a closed-world, it is assumed that a statement that is true is also known to be true. The absence of information is therefore treated as negative information, or a license to *jump to conclusions* until further information to the contrary become available. By contrast, in an open-world, completeness of information is not required. By making an open or closed-world assumption with varying amounts of incomplete knowledge, reasoning capability results will vary. This may not seem surprising in the presence of the above definition of practical reasoning, although to the best of our knowledge, this paper is the first to address the issue systematically,

---

[2]Our definition applies directly to the reasoning tasks evaluated in [6] where the reasoning capability of LTN is studied in the context of a set of experiments. Here, we allow such experiments to be generated systematically based not only on Prolog semantics but on any proof theory, in particular also using an open-world semantics.
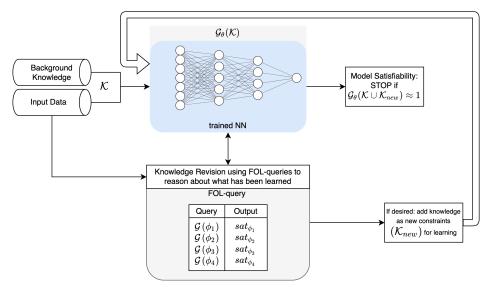
Figure 1: Illustration of the interactive continual learning framework introduced in [3]. Knowledge learned is revised in an iterative way using FOL queries. The revision process allows for the integration of symbolic reasoners to build up knowledge where NNs alone may struggle, e.g. in the case of extrapolation.

which is important with all the recent interest in NN reasoning.

In LTN, a quantified formula either expands the set of examples in an existing domain (a form of semi-supervised learning) or it helps with transfer learning from a data-rich domain (defined by a set of predicates) to a domain with insufficient data (e.g. a new predicate). In the first experiment below, we illustrate the first case and in the second experiment information is transferred from a domain with high availability of data (parent) to one without (ancestor).

4.1. Ontology Reasoning

The first example illustrates the manner in which the reasoning capabilities of a system that presumes an open-world (NN) may be affected and assessed when it is applied in a closed-world setting. An ontology's direct *subclass* relations are given and quantified formulas are utilised to derive and complement data in order to infer multiple hop inferences on the ontology's knowledge graph. Thus, the task is to learn *subclass* relations of any depth using quantifiable axioms in addition to 22 single-step *subclass* relations. The ontology is provided in Bianchi and Hitzler [6]. In this paper, we follow the task formalisation introduced by Badreddine et al. [7].

Domain: classes, to denote the taxonomy entries.

Constants: $C_1, .., C_{24}$ denoting the 24 classes for which an embedding is to be learned.

Predicates: $sub(x, y)$ for the subclass relation. $sub \in$ classes.

Variables: x, y, z are variable ranging over the domain (classes).

Axioms: Let $\mathcal{D} = \{(x, y)\}$ denote the 22 direct subclass relations, e.g. (Dog, Mammal), dog is a subclass of mammal. $\mathcal{D}$ is the training data for learning the groundings.

$$\forall x \in \mathcal{D}: \qquad \neg sub(x, x) \tag{1}$$

$$\forall x, y \in \mathcal{D}: \qquad sub(x, y) \rightarrow \neg sub(y, x) \tag{2}$$

$$\forall x, y, z \in \mathcal{D}: \qquad sub(x, y) \wedge sub(y, z) \rightarrow sub(x, z) \tag{3}$$

Let $\mathcal{E} = \{(x, y)\}$ denote the negative complement of the subclass relation. As Knowledge is assumed to be complete, it can be deduced by creating a set of all possible relations without the known subclass relation.

Grounding: $\mathcal{G}(\text{class}) = \mathbb{R}^2$. We learn embeddings in $\mathbb{R}^2$.

$\mathcal{G}(C_1 \mid \theta) = \mathbf{v}_\theta(C_1), \mathcal{G}(C_2 \mid \theta) = \mathbf{v}_\theta(C_2), \ldots, \mathcal{G}(C_{24} \mid \theta) = \mathbf{v}_\theta(C_{24})$; every class is associated with a vector of two real numbers. The embedding is initialized uniformly at random.

$\mathcal{G}(x \mid \theta) = \mathcal{G}(y \mid \theta) = \mathcal{G}(z \mid \theta) = \langle \mathbf{v}_\theta(C_1), \ldots, \mathbf{v}_\theta(C_{24}) \rangle$.

$\mathcal{G}(sub \mid \theta) : x, y \mapsto \texttt{sigmoid(MLP\_sub}_\theta(x, y))$, where $\texttt{MLP\_sub}_\theta$ has 1 output neuron.

Figure 2: Exemplary taxonomy as proposed in [6] to learn how LTN can represent ontology reasoning and learn all subclass relations by extrapolating to any depth. The figure shows (on the left) subclass relations inferred before and (on the right) after negative propositions $\mathcal{E}$ were added resulting in F1 scores of 0.64 and 0.97.

	F1 score	Precision	Recall
Prolog	1	1	1
NN Revision 1	0.635	0.436	1
NN Revision 2	0.968	1	0.93

Table 1
F1 score, Precision, and Recall of the deduced subclass relation for a symbolic reasoner in Prolog and an LTN for the revision steps without and with negative examples.

Figure 2 illustrates the difficulty of integrating logical statements that originate from primary logic-based approaches directly into DFL systems without considering the open or closed world assumptions. Since there are only positive propositions in $\mathcal{D}$, the network grounds every variable positively to the predicate $sub(x, z)$ when applying formula (3). Due to the fact that $\mathcal{D}$ only contains positive propositions, each variable in formula (3) is grounded positively to the predicate $sub(x, z)$ using the network. The formula does not provide any additional information for the gradient updates and, therefore, does not enhance the reasoning capabilities of the NN. Only formulas containing a negation in the consequent enable the network to acquire further relevant information in order to improve its reasoning capabilities. In the absence of such information, the network is unable to learn a meaningful interpretation of the vector embeddings. This is reflected in a low F1-score (F1=0.64) when considering all possible subclass relations (comparable to the results reported in [6]). Notably, the precision here is 0.436 but the recall is 1, indicating that the network is prone to false positives. Nevertheless, by assuming that knowledge is complete, it is possible to deduce negative propositions $\mathcal{E}$ on the basis of transitive closure. By generating negative propositions for learning, we impose data based on a closed-world assumption upon an open-world model, leading to a significant improvement of the F1 measure (F1=0.968).

4.2. Reasoning across Domains

The goal of this task is to be able to use quantified formulas to complete an existing domain but also to transfer information effectively into a new predicate domain. Again, we compare results with [6], this time on the well-known ancestor example from Inductive Logic Programming. We further compare results with logic programming using Prolog (henceforth; Prolog) and the open-world theorem prover Coq, for the sake of a comparison with symbolic reasoning approaches. Using subsets of the knowledge-base $\mathcal{K}$ in Prolog and Coq, we can derive comparisons on reasoning capability with incomplete information and open domains. We shall also investigate continual learning within the LTN iterative approach and its ability to reason with partial knowledge. The iterative application of LTN allows one to increase the size of the knowledge-base incrementally by querying the network. The following example illustrates how false positives may be exacerbated over multiple deductive reasoning steps as part of this iterative approach.

The ancestor example is a well-known family tree problem centred around the learning of recursive relations, in particular the definition of *ancestor* as the recursive application of a *parent* predicate. Given only positive propositions about the parent relation, the ancestor relation is expected to be learned and to be generalised to the recursive application of any instance of the parent relation[3].

Domain: people, to denote the individuals.

Constants: $C_1, .., C_{17}$ denoting the 17 individuals for which an embedding is to be learned (in the domain of people).

---

[3]In Prolog notation:
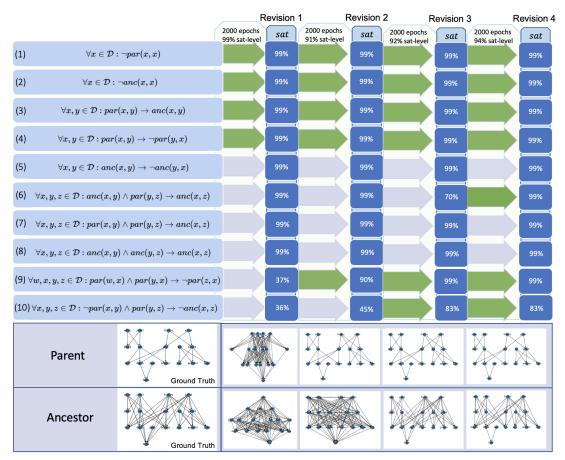ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).

Figure 3: Top: LTN solving the ancestor example [6] as part of an iterative revision process: $par(x, y)$ denotes that x is a parent of y; $anc(x, y)$ denotes that x is an ancestor of y. The figure shows the sat-level of each formula and the green arrows indicate the formulas used for training at each revision step. Bottom: Comparison of the graphs learned after each LTN revision with the ground truth of the ancestor example; the graphs approach the ground truth as more formulas that can generate negative propositions are retained by each revision.

	Revision 1		Revision 2		Revision 3		Revision 4	
	2000 epochs 99% sat-level	sat	2000 epochs 91% sat-level	sat	2000 epochs 92% sat-level	sat	2000 epochs 94% sat-level	sat
(1) $\forall x \in \mathcal{D} : \neg par(x,x)$		99%		99%		99%		99%
(2) $\forall x \in \mathcal{D} : \neg anc(x,x)$		99%		99%		99%		99%
(3) $\forall x,y \in \mathcal{D} : par(x,y) \rightarrow anc(x,y)$		99%		99%		99%		99%
(4) $\forall x,y \in \mathcal{D} : par(x,y) \rightarrow \neg par(y,x)$		99%		99%		99%		99%
(5) $\forall x,y \in \mathcal{D} : anc(x,y) \rightarrow \neg anc(y,x)$		99%		99%		99%		99%
(6) $\forall x,y,z \in \mathcal{D} : anc(x,y) \wedge par(y,z) \rightarrow anc(x,z)$		99%		99%		70%		99%
(7) $\forall x,y,z \in \mathcal{D} : par(x,y) \wedge par(y,z) \rightarrow anc(x,z)$		99%		99%		99%		99%
(8) $\forall x,y,z \in \mathcal{D} : anc(x,y) \wedge anc(y,z) \rightarrow anc(x,z)$		99%		99%		99%		99%
(9) $\forall w,x,y,z \in \mathcal{D} : par(w,x) \wedge par(y,x) \rightarrow \neg par(z,x)$		37%		90%		99%		99%
(10) $\forall x,y,z \in \mathcal{D} : \neg par(x,y) \wedge par(y,z) \rightarrow \neg anc(x,z)$		36%		45%		83%		83%

Predicates: $par(x, y), anc(x, y)$ for the parent and ancestor binary relation (both predicates in the domain of people).

Variables: x, y, z are variable ranging over the domain people.

Axioms: Let $\mathcal{D} = \{(x, y)\}$ denote the 22 parental relations, e.g. (joe, juliet), joe is a parent of juliet, (janice, juliet), janice is a parent of juliet, etc. $\mathcal{D}$ denotes the training data. The knowledge-base is incrementally expanded using the 10 quantifiable clauses listed in Figure 3 (left).

Grounding: $\mathcal{G}(\text{people}) = \mathbb{R}^2$. We learn embeddings in $\mathbb{R}^2$. $\mathcal{G}(C_1 \mid \theta) = \mathbf{v}_\theta(C_1), \mathcal{G}(C_2 \mid \theta) = \mathbf{v}_\theta(C_2), \ldots, \mathcal{G}(C_{17} \mid \theta) = \mathbf{v}_\theta(C_{17})$; every individual is associated with a vector of two real numbers. The embedding is initialized uniformly at

random.

$\mathcal{G}(x \mid \theta) = \mathcal{G}(y \mid \theta) = \mathcal{G}(z \mid \theta) = \langle \mathbf{v}_\theta(C_1), \ldots, \mathbf{v}_\theta(C_{17}) \rangle$.

$\mathcal{G}(parent \mid \theta) : x, y \mapsto \texttt{sigmoid(MLP\_parent}_\theta(x, y))$, where $\texttt{MLP\_parent}_\theta$ has 1 output neuron.

$\mathcal{G}(ancestor \mid \theta) : x, y \mapsto \texttt{sigmoid(MLP\_ancestor}_\theta(x, y))$, where $\texttt{MLP\_ancestor}_\theta$ has 1 output neuron.

As shown in Figure 3, the full knowledge-base is split into partial knowledge-bases and queried incrementally on knowledge that does not form part of the training process. The formulas used here correspond to those in the *small* and *extended* knowledge-bases used in [6]. For efficiency, we use an embedding of size 2 as opposed to 10 in [6]. Furthermore, we employ the product Real Logic operators introduced in [5]. With the theorem prover Coq, we can derive proofs of the ancestor relation given a knowledge-base and propositions (i.e., examples). Coq derives these proofs under an open-world assumption. The information is incomplete, although each revision step contributes to the knowledge-base (Table 2).

	F1 score	Precision	Recall	Sat Level	Coq True Positives	Coq True Negatives	Coq Unknowns	Positive Unknowns	Negative Unknowns
LTN Rev. 1	0.597	0.426	1	1	22	17	250 (86.5%)	9.6%	90.4%
LTN Rev. 2	0.631	0.46	1	0.907	22	17	250 (86.5%)	9.6%	90.4%
LTN Rev. 3	0.875	0.84	0.913	0.912	22	33	234 (81%)	10.26%	89.74
LTN Rev. 4	0.884	0.84	0.933	0.945	46	57	186 (74.4%)	0%	100%
Prolog Rev. 1,2,3	0.647	1	0.47						
Prolog Rev. 4	1	1	1						

Table 2

F1 score, Precision and Recall of the deduced ancestor relations using Prolog and LTN for the revision steps depicted in Figure 3. Prolog achieves perfect scores as expected with all the required information provided under the closed-world assumption. Coq is unable to prove a large number of cases under open-world assumption. LTN is able to perform well under open-world assumption given incomplete information, while improving on the reasoning capabilities reported in [6] with each revision step of the iterative LTN.

In a first revision step, we initialise the task with minimal background knowledge consisting of the 4 formulas highlighted in the first column of Figure 3 and the 22 propositions ($\mathcal{D}$) of the parent relation. We train the network to learn an embedding for each clause indicated by a green arrow in the figure at each revision step. We observe fast convergence at training to the level of 99% of satisfiability after 2,000 epochs, whilst in [6] networks were optimised up to 10,000 and 20,000 epochs. In a first revision step, we initialise the task with minimal background knowledge consisting of the 4 formulas highlighted in the first column of Figure 3 and the 22 propositions ($\mathcal{D}$) of the parent relation. The network is trained at each revision step to learn a grounding for the clauses indicated by a green arrow in the figure. Upon training, we observe a fast convergence to 99% satisfiability of $\mathcal{K}$ after 2,000 epochs, whilst in [6] networks were optimised up to 10,000 and 20,000 epochs to converge. By querying each clause after learning from clauses 1 to 4, it is evident that the network has acquired knowledge of clauses 5 to 8, as indicated by their satisfiability levels at revision 1 in Fig.3. Thus, using the extended knowledge-base as presented in [6] would not enhance the deductive reasoning capability in this

case (with the use of the product real logic operators [5]). The network is generating many false positives (see Table 2) as numerous propositions and quantified formulas in $\mathcal{K}$ specify positive relations. The majority of unknown relations which cannot be deduced, as determined by the theorem prover Coq, are inferred to be true by the network. Although, under the assumption of a closed world, Prolog assigns all unknowns to false and achieves high precision (see Table 2). By querying clauses 9 and 10, whose consequents are negative literals, a low satisfiability value of around 37 % indicates also that the majority of parent relations that are not part of the training data are assumed to be true by the network. Therefore, either more data containing negative examples should be considered or specific quantified formulas should be added to LTN, both will yield the same effect of improving the groundings. Here we opted for the latter option due to the simplicity of adding a clause as a constraint to LTN. The incremental addition of clauses and the monitoring of performance improvements should make revision easier when clauses (rather than large chunks of data) may need to be removed later.

As a part of revision step two, we add clause 9 having $\neg par(z, x)$ in the consequent to the LTN and continue to train the LTN for 2,000 additional epochs. As shown in Figure 3 bottom, the added knowledge enables the network to eliminate false positives from the par relation. Notably, this translates into an improved inference of the anc relation by the network, as can be seen from the Table 2, despite Coq finding no additional proofs for these anc relation.

During revision step three, following an additional 2,000 epochs of training with the clauses shown in Figure 3, the $\neg anc$ relation is further specified. An examination of the F1 score in Table 2 indicates that the provision of negative examples significantly enhances the reasoning capabilities of DFL-models. Further specification of $\neg anc$ cannot improve Prolog's results where unknowns are assumed to be false under the closed-world assumption.

Lastly, revision four adds the clauses that fully specify all ancestor relations within the set of examples and all 46 positive anc relations can be derived using Coq. However, the $\neg anc$ relations remain under-specified with 186 examples not being provable under an open-world assumption. The continuous revision of the model has demonstrated that one can eliminate an undesirable model property by adding constraints that will increase the deductive capabilities reported in [6]. Using less than 1% trainable parameters, we obtain a Mean Absolute Error of 0.04 and F1 of 0.884, while [6] achieved 0.14 and 0.85. The results indicate that open-world DFL methods are prone to recall while logical reasoners under a closed-world assumption are better at attaining high precision when the domain is underspecified. Considering this point is essential when making assumptions about the information required to ground a domain through learning. Our findings also indicate that DFL methods outperform logic-based methods when the information is incomplete.

5. Conclusion

Adding logically-redundant knowledge to a neurosymbolic system can be beneficial to improve overall performance. When such performance is intended to measure the so-called reasoning capability of the system, it is important to make explicit the underlying assumption of a closed-world or open-world reasoning. We began by providing a definition of reasoning in NNs, which can be used to formalise, measure, and compare results. The first reasoning example used in this

paper (ontology reasoning) showed how an open-world neurosymbolic model trained without sufficient negative information fails to learn simple relational knowledge. As expected, the iterative provision of negative information can improve reasoning capabilities dramatically. The second example used in this paper (the ILP ancestor example) showed the same trend, with the iterative application of knowledge improving the reasoning capability of the neural network. The neurosymbolic approach showed also a superior reasoning performance than purely symbolic methods in the presence of incomplete information, indicating how the combination of data and knowledge may benefit purely neural or purely symbolic (reasoning) approaches. Comparisons with purely-symbolic learning approaches and other neurosymbolic approaches are planned as future work. Our results showed that the network was capable of capturing *multi-hop* steps and we intend to extend the evaluation to a large-scale knowledge graph with increasingly sophisticated relations.

References

[1] A. d. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, S. N. Tran, Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning, arXiv:1905.06088 (2019). URL: http://arxiv.org/abs/1905.06088.

[2] T. R. Besold, A. d. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kuehnberger, L. C. Lamb, D. Lowd, P. M. V. Lima, L. de Penning, G. Pinkas, H. Poon, G. Zaverucha, Neural-Symbolic Learning and Reasoning: A Survey and Interpretation (2017) 1–58. URL: http://arxiv.org/abs/1711.03902.

[3] B. Wagner, A. d. Garcez, Neural-Symbolic Integration for Fairness in AI, in: AAAI Spring Symposium AAAI-MAKE, 2021. URL: http://ceur-ws.org/Vol-2846/paper5.pdf.

[4] B. Wagner, A. d'Avila Garcez, Neural-Symbolic Integration for Interactive Learning and Conceptual Grounding, in: NeurIPS, Workshop on Human and Machine Decisions, 2021. URL: https://arxiv.org/abs/2112.11805.

[5] E. v. Krieken, E. Acar, F. v. Harmelen, Analyzing Differentiable Fuzzy Logic Operators (2020). URL: https://arxiv.org/abs/2002.06100v1.

[6] F. Bianchi, P. Hitzler, On the Capabilities of Logic Tensor Networks for Deductive Reasoning, AAAI Spring Symposium MAKE (2019).

[7] S. Badreddine, A. d. Garcez, L. Serafini, M. Spranger, Logic Tensor Networks (2020). URL: http://arxiv.org/abs/2012.13635.

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

[9] S. Guo, Q. Wang, L. Wang, B. Wang, L. Guo, Jointly embedding knowledge graphs and logical rules, in: EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings, 2016. doi:10.18653/v1/d16-1019.

[10] M. Diligenti, M. Gori, C. Saccà, Semantic-based regularization for learning and inference, Artificial Intelligence (2017). doi:10.1016/j.artint.2015.08.011.

[11] G. Marra, F. Giannini, M. Diligenti, M. Gori, LYRICS: a General Interface Layer to Integrate Logic Inference and Deep Learning (2019). URL: http://arxiv.org/abs/1903.07534.

[12] I. Donadello, L. Serafini, A. d. Garcez, Logic tensor networks for semantic image in-

terpretation, in: IJCAI International Joint Conference on Artificial Intelligence, 2017. doi:`10.24963/ijcai.2017/221`.

[13] L. Bottou, From machine learning to machine reasoning: An essay, Machine Learning (2014). doi:`10.1007/s10994-013-5335-x`.

[14] M. Ebrahimi, A. Eberhart, F. Bianchi, P. Hitzler, Towards bridging the neuro-symbolic gap: deep deductive reasoners, Applied Intelligence (2021). doi:`10.1007/s10489-020-02165-6`.

[15] C. G. Hempel, I.—STUDIES IN THE LOGIC OF CONFIRMATION (I.), Mind (1945). doi:`10.1093/mind/liv.213.1`.

[16] H. Lipson, Foundations of Artificial Intelligence, Cornell University CS4700, 2011, p. Lecture 16 Slides.

A. Appendix

A.1. On the Raven's paradox

Krieken et al. [5] illustrate the reasoning capabilities of DFL systems using the well-known raven paradox [15]. They approach the raven paradox by specifying and measuring differential updates that have particular properties. Their focus is specifically on the fuzzy implications and adjustments to the relative importance of Modus Ponens (MP) with respect to Modus Tollens (MT) updates, which is a trade-off at the heart of the the raven's paradox[4].

The paradox centres around the question: What constitutes evidence for the statement "all ravens are black"? In first-order logic, $\forall x : raven(x) \rightarrow black(x)$.

The authors observe that the average DFL gradient update due to MT is, in the case of this problem, around 100 times larger than the average MP gradient update, i.e. "it uses far more contrapositive reasoning" [5]. They propose to handle positive and contrapositive reasoning separately. MP and MT differential updates in the context of the paradox can be specified as:

- Modus Ponens: "It's a raven so it has to be black"; increase the truth-value of *black* during semi-supervised learning.
- Modus Tollens: "It is not black so it cannot be a raven"; decrease the truth-value of *raven* by contraposition of the implication.

Most fuzzy implications will focus on MP reasoning (first bullet point above). Krieken et al. [5] argue that this is undesirable as MT reasoning (second bullet point) can be more representative of what goes on in the real world.

Although the argument about the importance of MT is bound to continue (MT does not feature in Prolog, for example), the difference between MP and MT is particularly noticeable in semi-supervised settings because the choice of Modus influences directly how unlabelled data is used for optimisation of the system. We argue that MT may need to be provided explicitly to the system in the form of a formula $\forall x : \neg black(x) \rightarrow \neg raven(x)$, allowing one to generate

---

[4]Does the observation of non-black objects varied in colour and unrelated to ravens provide evidence that should increase the likelihood that all ravens are black?

an adequate number of negative examples (non-ravens of various colours) to counteract the problem with the gradients.

In the context of learning given the raven's paradox, examples of non-ravens become relevant and should be provided as input to the system

If possible, providing such data directly (i.e. instances of non-ravens, non-black things $\neg black(x) \wedge \neg raven(x)$) may be preferred to allow for an adequate interpolation or clustering of representations, but the measure of this may be highly domain dependent. There are limitations that remain as the non-black non-ravens may be difficult to specify in an open world with the DFL system not being able to ground non-black things equally well as black things using a data-driven approach. However, it should be noted that in the LTN iterative approach, the observed phenomenon can be tackled at the fuzzy logic knowledge level as well as at the data-driven level. Notice how, when a closed-world assumption can be adopted, in particular when Clark's completion can be assumed valid (not the case of the raven's paradox), the problem is simplified greatly. This can be illustrated with the well-known example of human reasoning whereby participants are told that *if there's an exam tomorrow then Lisa studies late in the library*, $\forall x : exam(x) \rightarrow library(Lisa)$. When informed in addition that there are no exams tomorrow, most participants conclude that Lisa is not in the library, which is a fallacy. A possible explanation is that most participants assume, when told *if there's an exam tomorrow then Lisa studies late in the library*, that the completion holds, i.e. *Lisa studies late in the library if and only if there's an exam tomorrow*, $\forall x : exam(x) \leftrightarrow library(Lisa)$. Given this closed-world assumption, it is not difficult to derive $\forall x : \neg exam(x) \rightarrow \neg library(Lisa)$, which provides an adequate knowledge-base for the generation of negative examples required for DFL learning.

A.2. Detailed description of the ontology and ancestor examples

In this section, we detail the setup of the examples used in the paper. We define the domain and all knowledge-base formulas and give a description that follows the same structure as that used in [7]. All experiments were run using the Tensorflow version 2.0 and reproduced in PyTorch 1.2. The processor used i9 (4.5Ghz) with a NVIDIA 2080 TI GPU and 32GB Ram using Windows 10[5].

A.2.1. Ontology Reasoning

The task is learning an ontology's *subclass* relations. Direct (one-hop) relations in the form of propositions are given and quantified formulas are used to derive and complement data to infer multi-hop inferences of any length in the ontology's knowledge graph.

---

[5]The code for the experiments can be found here: https://github.com/benediktwagner/ltn-reasoning.
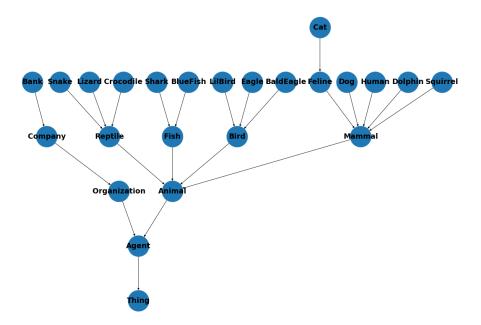
Figure 4: Exemplary taxonomy as proposed in [6] to learn how a NN trained using the LTN framework can perform on ontology reasoning. The task is to learn all subclass relations by extrapolating these relations to any length.

A.2.2. Cross-domain Reasoning

The task is a combination of transferring information from the parent predicate to the ancestor predicate while at the same time inferring the complement of the existing formulas (non-parents). This is achieved by the quantified formulas and it illustrates how false positives may be exacerbated over multiple reasoning steps.
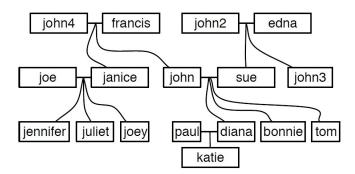

Figure 5: Exemplary family tree as proposed in [16] to learn how a NN trained using the LTN framework can perform on multi-hop reasoning. The task is to learn all ancestor relations by deducing these from the given parent relation.

	Known True	Known False	Unknowns	Total
Revision 1	22	17	250	289
Revision 2	22	17	250	289
Revision 3	22	33	234	289
Revision 4	46	57	186	289

Table 3
Using the theorem prover Coq, we can derive proofs for true and false ancestor relation given axioms and propositions. It shows that although the information is incomplete, each revision step adds to what can be deduced.

	F1 score	Precision	Recall	Sat Level
Prolog Rev. 1,2,3	0.647	1	0.47	
Prolog Rev. 4	1	1	1	
Coq (unk assumed false) Rev. 1,2	0.647	1	0.47	
Coq (unk assumed false) Rev. 3	0.647	1	0.47	
Coq (unk assumed false) Rev. 4	1	1	1	
Coq (unk assumed true) Rev. 1,2	0.289	0.169	1	
Coq (unk assumed true) Rev. 3	0.305	0.18	1	
Coq (unk assumed true) Rev. 4	0.33	0.197	1	
LTN Rev. 1	0.597	0.426	1	1
LTN Rev. 2	0.631	0.46	1	0.907
LTN Rev. 3	0.875	0.84	0.913	0.912
LTN Rev. 4	0.884	0.84	0.933	0.945

Table 4
F1 score, Precision and Recall of the deduced ancestor relation; Prolog compared with LTN for the revision steps depicted in Figure 3. Prolog is performing well as all required knowledge is given under closed-world-assumption. LTN is able to perform well under open-world assumption with incomplete information when compared with open-world Coq. Unknowns refer to all relations that are not provable under open-world assumption.