From Statistical Relational to Neural-Symbolic Artificial Intelligence

Luc De Raedt[1,2] , **Sebastijan Dumančić**[1] , **Robin Manhaeve**[1] and **Giuseppe Marra**[1]
[1]KU Leuven, Department of Computer Science and Leuven.AI
[2]Örebro University, Center for Applied Autonomous Sensor Systems
{luc.deraedt, sebastian.dumancic, robin.manhaeve, giuseppe.marra}@kuleuven.be

Abstract

Neural-symbolic and statistical relational artificial intelligence both integrate frameworks for learning with logical reasoning. This survey identifies several parallels across seven different dimensions between these two fields. These cannot only be used to characterize and position neural-symbolic artificial intelligence approaches but also to identify a number of directions for further research.

1 Introduction

The integration of learning and reasoning is one of the key challenges in artificial intelligence and machine learning today, and various communities have been addressing it. That is especially true for the field of neural-symbolic computation (NeSy) [Besold *et al.*, 2017; d'Avila Garcez *et al.*, 2019], where the goal is to integrate symbolic reasoning and neural networks. NeSy already has a long tradition, and it has recently attracted a lot of attention from various communities (cf. the keynotes of Yoshua Bengio and Henry Kautz on this topic at AAAI 2020).

Another domain that has a rich tradition in integrating learning and reasoning is that of statistical relational learning and artificial intelligence (StarAI) [Getoor and Taskar, 2007; De Raedt *et al.*, 2016]. But rather than focusing on integrating logic and neural networks, it is centred around the question of integrating logic with probabilistic reasoning, more specifically probabilistic graphical models. Despite the common interest in combining symbolic reasoning with a basic paradigm for learning, i.e., probabilistic graphical models or neural networks, it is surprising that there are not more interactions between these two fields.

This discrepancy is the key motivation behind this survey: it aims at pointing out the similarities between these two endeavours and in this way it wants to stimulate cross-fertilization. In doing so, we start from the literature on StarAI, following the key concepts and techniques outlined in a number of textbooks and tutorials such as [Russell, 2015; De Raedt *et al.*, 2016], because it turns out that the same issues and techniques that arise in StarAI apply to NeSy as well. As the key contribution of this survey, *we identify seven dimensions that these fields have in common and that can be used to categorize both StarAI and NeSy approaches*. These seven dimensions are concerned with (1) directed vs undirected models, (2) model vs proof-based inference, (3) integrating logic with probability and/or neural computation, (4) logical semantics, (5) learning parameters or structure, (6) representing entities as symbols or sub-symbols, and (7) the type of logic used. We provide evidence for our claim by positioning a wide variety of StarAI and NeSy systems along these dimensions and pointing out analogies between them. This provides not only new insights into the relationships between StarAI and NeSy, but it also allows one to carry over and adapt techniques from one field to another. Thus the insights provided in this paper can be used to create new opportunities for cross-fertilization between StarAI and NeSy, by focusing on those dimensions that have not been fully exploited yet. Of course, there are also important differences between StarAI and NeSy, the most important one being that the former operates more at the symbolic level, lending itself naturally to explainable AI, while the latter operates more at the sub-symbolic level, lending itself more naturally for computer vision and natural language processing.

Unlike some other recent surveys or perspectives on neural-symbolic computation [Besold *et al.*, 2017; d'Avila Garcez *et al.*, 2019], the present survey limits itself to a logical and probabilistic perspective, which it inherits from StarAI, and to developments in neural-symbolic computation that are consistent with this perspective. Furthermore, it focuses on representative and prototypical systems rather than aiming at completeness (which would not be possible given the fast developments in the field). Another early overview of neural-symbolic computation is that of [Bader and Hitzler, 2005]. Unlike the present survey it focuses very much on a logical and a reasoning perspective. Today, the focus has shifted very much to learning.

The following sections of the paper each describe one dimension. We summarize various neural-symbolic approaches along these dimensions in Table 1. For ease of writing, we do not always repeat the references to these approaches in the paper, the table mentions the key reference for each of them.

2 Directed vs Undirected

Within the graphical model community there is a distinction between the *directed* and *undirected* graphical models [Koller and Friedman, 2009], which has led to two distinct types of StarAI systems. The first generalizes directed models, and

resembles Bayesian networks; the second generalizes undirected models like Markov networks or random fields. The key difference between the two is that the first class of models indicates a natural direction (sometimes the term "causal" is used) between the different random variables, while the second one does not.

In StarAI, the first category includes well-known representations such as plate notation [Koller and Friedman, 2009], probabilistic relational models (PRMs) [Friedman *et al.*, 1999], probabilistic logic programs (PLPs) [De Raedt and Kimmig, 2015], and Bayesian logic programs (BLPs) [Kersting and De Raedt, 2007]. Today the most typical and popular representatives of this category are the probabilistic (logic) programs. The second category includes Markov Logic Networks (MLNs) [Richardson and Domingos, 2006] and Probabilistic Soft Logic (PSL) [Bach *et al.*, 2017]. They specify a set of weighted constraints, clauses or formulae.

From a logical perspective, the difference amounts to using a form of definite clauses (as in the programming language Prolog) versus the use of full clausal logic or even first order logic. On the one hand, a definite clause is an expression of the form $h \leftarrow b_1 \wedge ... \wedge b_n$ where h and the b_i are logical atoms of the form $p(t_1, ..., t_m)$, with p being a predicate of arity m and the t_i being terms, that is, constants, variables, or structured terms of the form $f(t_1, ..., t_n)$, where f is a functor and the t_i are again terms. On the other hand, full clausal logic also allows for formulae of the form $h_1 \vee ... \vee h_m \leftarrow b_1, ..., b_n$.

The first type of rule forms the basis of programming and database languages such as Prolog and Datalog. It is typically used for forward or backward inference to prove that certain atoms hold. The second type of clause specifies a more general relationship between two sets of atoms, the ones in the condition and in the conclusion part. While such clauses can also be used in (resolution) theorem provers, they can also be viewed as constraints that relate these two sets of atoms as is common in Answer Set Programming [Gebser *et al.*, 2012]. This difference reflects the kind of knowledge that the user has about the problem. With directed models, one can express that a set of variables has a direct "causal" influence on another one, while with undirected ones one expresses a kind of (soft) constraints on a set of variables, that is, that the variables are related to one another. More details on these connections can be found in [De Raedt *et al.*, 2016].

Borrowing this view from StarAI, we can devise a first dimension for neural-symbolic approaches, which relies entirely on the logical perspective outlined above.

The first category includes systems which retain the directed nature of logical inference as they exploit backward chaining. The most prominent members of this category are NeSy systems based on Prolog or Datalog, such as Neural Theorem Provers (NTPs) [Rocktäschel and Riedel, 2017], NLProlog [Weber *et al.*, 2019], DeepProbLog [Manhaeve *et al.*, 2018] and DiffLog [Si *et al.*, 2019]. Lifted Relational Neural Networks (LRNNs) [Šourek *et al.*, 2018] and ∂ILP [Evans and Grefenstette, 2018] are other examples of non-probabilistic directed models, where definite clauses are compiled into a neural network architecture in a forward chaining fashion. The systems that imitate logical reasoning with tensor calculus, Neural Logic Programming (NeuralLP) [Yang *et al.*, 2017] and Neural Logic Machines (NLM) [Dong *et al.*, 2019], are likewise instances of directed logic.

The undirected NeSy approaches consider logic as a constraint on the behaviour of a predictive model. A large group of approaches, including Semantic Based regularization (SBR) [Diligenti *et al.*, 2017], Logic Tensor Networks(LTN) [Donadello *et al.*, 2017] and Semantic Loss (SL) [Xu *et al.*, 2018], exploits logical knowledge as a soft constraint over the hypothesis space in a way that favours solutions consistent with the encoded knowledge. SBR and LTN implement predicates as neural networks and translates the provided logical formulas into a real valued regularization by means of fuzzy logic, while SL uses marginal probabilities of the target atoms to define the regularization term and relies on arithmetic circuits [Darwiche, 2011] to evaluate it efficiently. Similarly, another group of approaches, including Neural Markov Logic Networks (NMLN) [Marra and Kuželka, 2019] and Relational Neural Machines (RNM) [Marra *et al.*, 2020] extend MLNs, allowing factors to be implemented as neural architectures. Finally, [Rocktäschel *et al.*, 2015; Demeester *et al.*, 2016] compute ground atoms scores as dot products between relation and entities embeddings; implication rules are then translated into a logical loss by means of a continuous relaxation of the implication operator.

3 Model-based vs Proof-based Inference

The distinction between directed and undirected models is closely related to the distinction between a model-theoretic and proof-theoretic approach to inference. This is clear when looking at the difference between Answer Set Programming and the programming language Prolog. In the model theoretic perspective, one first grounds out the clauses in the theory and then calls a SAT solver (possibly after breaking cycles), while in a proof-theoretic perspective one performs a sequence of inference steps in order to obtain a proof.

Grounding is the step whereby a clause c (or formula) containing variables $\{V_1, ..., V_k\}$ is replaced by all instances $c\theta$ where θ is a substitution $\{V_1 = c_1, ...V_k = c_k\}$ and the c_i are constants (or other ground terms) appearing in the domain. The resulting clause $c\theta$ is that obtained by simultaneously replacing all variables by the corresponding constants. Usually the grounding process is optimised in order to obtain only those ground clauses that are relevant for the considered inference task.

Many StarAI systems use logic as a kind of template to ground out the relational model in order to obtain a grounded model and perform inference. This grounded model can be a graphical model, or alternatively, a ground weighted logical theory on which traditional inference methods apply, such as belief propagation or weighted model counting. This is used in well known systems such as MLNs, PSL, BLPs, and PRMs. Some systems like PRMs and BLPs additionally use aggregates, or combining rules, in order to combine multiple conditional probability distributions into one using, e.g., noisy-or.

Alternatively, one can follow a proof or trace based approach to define the probability distribution and perform inference. This is akin to what happens in probabilistic program-

ming (cf. also [Russell, 2015]), in StarAI frameworks such as PLPs, probabilistic databases [Van den Broeck *et al.*, 2017] and probabilistic unification based grammars such as Stochastic Logic Programs (SLPs) [Muggleton, 1996]. Just like pure logic supports the model-theoretic and proof-theoretic perspectives, both perspectives have been explored in parallel for some of the probabilistic logic programming languages such as ICL [Poole, 2008] and ProbLog [Fierens *et al.*, 2015].

These two perspectives carry over to the neural-symbolic methods. Approaches like NTPs, DeepProblog, ∂ILP and DiffLog are proof-based. The probabilities or certainties that these systems output are based on the enumerated proofs, and they are also able to learn how to combine them. In contrast, approaches of LRNN, LTNs, RNM, NMLN, NLM and NeuralLP are all based on the model-theoretic perspective. Learning in these models is done through learning the (shared) parameters over the ground model and inference is based on possible groundings of the model.

4 Logic vs Probability vs Neural

When two paradigms are integrated, examining which of the base paradigms are preserved, and to which extent, tells us a lot about the strengths and weaknesses of the resulting paradigm. In StarAI, the traditional knowledge-based model construction approach is to use the logic only to generate a probabilistic graphical model. Thus the graphical model can be used to define the semantics of the model and also to perform inference. This can make it more difficult to understand the effects of applying logical inference rules such as resolution. For instance, in MLNs the addition of the resolvent of two weighted rules, makes it hard to predict the effect on the distribution.

On the other hand, the opposite holds for PLPs and its variants. While it is clear what the effect of a logical operation is, it is often harder to directly identify and exploit properties such as conditional or contextual independencies, which are needed for efficient probabilistic inference.

The position on the spectrum between logic and probability has a profound influence on the properties of the underlying model. For NeSy, the spectrum involves not only logic and neural networks, but also probability. It has been argued that when combining different perspectives in one model or framework, such as neural, logic and probabilistic ones, it is desirable to have the originals or base paradigms as a special case, see also [De Raedt *et al.*, 2019].

The vast majority of current NeSy approaches focuses on the neural aspect (i.e., they originated as a fully neural method to which logical components have been added). Some of these approaches like LTNs and TensorLog [Cohen *et al.*, 2017] pursue a kind of knowledge-based model construction approach in which the logic is compiled away into the neural network architecture. A different family of NeSy approaches, which includes SL and SBR, turns the logic into a regularization function to provide a penalty whenever the desired logical theory or constraints are violated. This leads to the logic being compiled into the weights of the trained neural network.

A small number of NeSy methods do retain the focus on logic. Some of these methods start from existing logic (programming) frameworks and extend them with primitives that allow them to interface with neural networks and allow for differentiable operations. Examples include DeepProbLog and DiffLog. Other methods instead take an existing framework and turn it into a differentiable version. The key inference concepts are mapped onto an analogous concept that behaves identically for the edge cases, but is continuous and differentiable in non-deterministic cases. Such methods include ∂ILP, ∂4 [Bošnjak *et al.*, 2017] and NTPs.

An aspect that significantly aids in developing a common framework, and analysing its properties, is the development of an intermediate representation language that can serve as a kind of *assembly language* [Zuidberg Dos Martires *et al.*, 2019]. One such idea concerns performing probabilistic inference by mapping it onto a weighted model counting (WMC) problem. This can then in turn be solved by compiling it into a structure (e.g. an arithmetic circuit) that allows for efficient inference. This has the added benefit that this structure is differentiable, which can facilitate the integration between logic based systems and neural networks. DeepProbLog, for example, uses this approach.

5 Semantics

Traditionally, StarAI combines two semantics: a logical and a probabilistic one. In the *logical* semantics, atoms are assigned a truth value in the $\{$*true*,*false*$\}$ set (i.e. $\{0, 1\}$). In a *probabilistic* semantics, probability is defined as a measure over sets of possible worlds, where each possible world is an assignment of values to the random variables. This implies that a probabilistic logic semantics defines probability distributions over ground logical interpretations, that is, over sets of ground facts. Prominent examples in StarAI are ProbLog (from the directed side) and MLNs (from the undirected one). Incorporating probabilistic semantics into a logical one is natural when one wants to perform logical reasoning under uncertainty. Moreover, it "only" extends Boolean logic with probabilities, thus it preserves the original logical semantics. However, this extension comes at the price of more complex inference, which makes probabilistic StarAI models intractable in large scale domains.

Another approach is to turn the logical operators into real-valued functions, and in doing so relax the Boolean truth values to the continuous $[0, 1]$ interval. This introduces the semantics of *fuzzy logic* (or soft logic), which is mathematically grounded in the t-norm theory. The fuzzy semantics can be used alone or in conjunction with the probabilistic one (e.g. [Bach *et al.*, 2017]). The algebraic and geometric properties of t-norms (including especially convexity and differentiability) results in a reduction in the complexity of logical and/or probabilistic inference. However, differently from the probabilistic case, the semantics of the original Boolean theory is not preserved. Indeed, the *fuzzification* procedure can introduce undesirable effects. In particular, improper choices in the fuzzification could lead to behaviours that are different from the ones in the original theory [Giannini *et al.*, 2018] and particular attention should be paid to assessing that any desired property is preserved. For example, when translating the implication $A \rightarrow B$ with its fuzzy material implication (i.e. $\neg A \vee B$), one may lose the transitivity property (e.g. the material implication

in the minimum logic). Another class of anomalies concerns the way t-norms behave when aggregating a large number of fuzzy truth degrees. An example is the n-ary Łukasiewicz strong disjunction $F_{\oplus}(x_1, \ldots, x_n) = min(1, x_1 + \cdots + x_n)$. It can evaluate to 1 (i.e. true) also when all x_i are very small (i.e. false), e.g. $n = 10$ and $x_i = 0.1$. The use of this operator would lead to poor approximations when disjointing multiple atoms, e.g. in existentially quantified formulas or in the aggregation of several alternative proofs. [van Krieken *et al.*, 2020] analyse similar issues, also in connection to differentiability.

Neural-symbolic approaches can easily be categorized along the same lines. Neural enhancements of the *logic* semantics either use neural networks to turn perceptive input to a logical atom, or relax the logical reasoning through tensor calculus. An instance of the former is ABL [Dai *et al.*, 2019], which use logical abduction to provide the feedback for a neural model processing the perceptive input. Tensor calculus approaches, such as NLM and NeuralLP, interpret predicates as tensors grounded over all constants in a domain and interpret clauses as a product of those matrices.

Neural enhancements of the *probabilistic* semantics usually parameterize the underlying distribution in terms of neural components. In particular, DeepProbLog exploits neural predicates to compute the probabilities of probabilistic facts as the output of neural computations over vectorial representations of the constants, which is similar to SL in the propositional counterpart. NMLN and RNM use neural potentials in order to implement factors (or their weights) as neural networks. [Rocktäschel *et al.*, 2015] computes marginal probabilities as logistic functions over similarity measures between embeddings of entities and relations.

In neural enhancements of the *fuzzy* semantics, neural networks produce continuously-valued truth assignments. The differentiability of the t-norms allows to easily integrate neural frameworks. In particular, SBR and LTN turn atoms into neural networks taking as inputs the feature representation of the constants and returning the corresponding truth value. Similarly, in LRNN, ∂ILP, DiffLog and [Wang and Pan, 2019], the scores of the proofs are computed by using fuzzy logic connectives.

Finally, a large class of methods [Minervini *et al.*, 2017; Demeester *et al.*, 2016; Cohen *et al.*, 2017; Weber *et al.*, 2019] relaxes logical statements in a numeric way, without giving any other specific semantics. Here, atoms are assigned scores in $\mathbb{R}$ computed by a neural scoring function over embeddings. Numerical approximations are then applied either to combine these scores according to logical formulas or to aggregate proofs scores. The resulting neural architecture is usually differentiable and, thus, trained end-to-end. It is however hard to interpret the numbers generated by such approaches.

6 Structure vs Parameter Learning

StarAI distinguishes between two types of learning: structure learning, which corresponds to learning the logical clauses of the model [Kok and Domingos, 2005], and parameter learning in which the probabilities or weights of the clauses have to be estimated [Gutmann *et al.*, 2008; Lowd and Domingos, 2007]. The former is typically achieved by means of a combinatorial search over the space of possible clauses, while in the latter one an expert user provides a set of informative clauses for which only the probabilities have to be estimated.

Learning in NeSy approaches blurs this distinction and is positioned somewhere mid-ground: the model structure is learned though parameter learning. In contrast to StarAI parameter learning in which the user carefully selects the informative clauses, the set of clauses in NeSy approaches is typically enumerated from the user-provided rule templates of predefined complexity. As such, the enumerated rules contain noisy and erroneous patterns that are corrected by learning the corresponding probabilities or weights. The structure learning is therefore not performed explicitly as none of the given rules is removed. While it is certainly possible to extract the most important clauses, the inference is still performed considering all the enumerated clauses. Examples of such systems include NTPs, ∂ILP, DeepProbLog, NeuralLP and DiffLog. A related way of learning the structure is that of *program sketching*, in which a user provides a sketch of the target model and leaves certain parts of the model unspecified. The learning task corresponds to filling the blanks. NeSy systems based on sketching, such as DeepProbLog and ∂4, perform a simplified version of sketching in which they fill in a single operation instead of an entire program.

A substantial number of approaches tries to leverage the best of both worlds. These ideas include using neural models to guide the symbolic search [Kalyan *et al.*, 2018; Ellis *et al.*, 2018a; Valkov *et al.*, 2018], or using a neural model to produce a program that is then executed symbolically [Ellis *et al.*, 2018b; Mao *et al.*, 2019].

7 Symbols vs Sub-symbols

Perhaps the biggest difference between StarAI and neural methods is how they represent entities. StarAI generally represents entities by constants (symbols). However, neural methods are unable to represent symbols directly and exactly, and therefore represent them with sub-symbolic formats, such as vectorized representations. For instance, vectorized representations can be created through one-hot encodings, inherent numerical properties of symbols (e.g. the pixel data of an image), or by learning a mapping from one-hot encodings to a dense feature space. This, of course, has an impact on the generalizability of the system towards unseen entities, as no vectorized representation is available for an unseen entity.

An especially interesting aspect of NeSy methods is that they combine the best of both worlds by introducing a variety of ways to combine symbols and sub-symbols for task representation and reasoning. The idea of mapping entities onto sub-symbols is made very explicit in LTNs, where in a first step, all symbols are replaced with sub-symbols. In DeepProbLog, entities are represented using symbols, but they sometimes have sub-symbolic representations that are only used inside the neural networks. Similarly, in [Lippi and Frasconi, 2009] and RNM, MLNs are conditioned on a feature representation of constants (e.g. images, audio signals). Finally, among those models exploiting learned embeddings, we find [Rocktäschel *et al.*, 2015; Minervini *et al.*, 2017; Demeester *et al.*, 2016].

A powerful and elegant mechanism for reasoning about symbols matching in logic is *unification*. For instance, the atomic expressions $p(a, Y)$ and $p(X, b)$ can be unified using the substitution $\{X = a, Y = b\}$. Unification not only works for constants but also for structured terms $f(t_1, ..., t_n)$ where f is a structured term and the t_i are constants, variables or structured terms themselves. While unification is not supported by standard neural networks, matching of the symbols can be performed based on their similarity in embedding space. Entities are typically embedded in some metric space, and represented through their embeddings, that is, through sub-symbols. Reasoning typically proceeds by performing algebraic operations (such as vector addition) on these embeddings, and considering the similarity between two entities by using their distance in embedding space. It is quite interesting to see to what extent current neural-symbolic approaches support unification on the one hand, and to what extent the use of embeddings has been integrated into the neural-symbolic logics as a kind of *soft* equality or unification

This idea was implemented in NTPs and NLProlog as *soft* or *weak unification*. In these systems, two entities can be unified if they are similar, and not just if they are identical. As such, this system can interweave both symbols and sub-symbols during inference. For each entity, an embedding is learned and their similarity is determined based on the distance between the embeddings using a radial basis function. However, this potentially adds a lot of different proof paths, which can result in computational issues for larger programs. This problem was solved in later iterations of the system [Minervini *et al.*, 2020].

8 Type of Logic

StarAI approaches have explored various types of logical representations, following a natural ordering [De Raedt, 2008; Flach, 1994] starting with propositional logic (symbols without arguments), to relational logic (with only constants and variables as terms, without any structured terms) which forms the basis for the Datalog database language, to general first order logic, and then to logic programs as in the programming language Prolog. Logic programs are usually restricted to definite clauses. The semantics of a definite clause program is given by its least Herbrand model, the set of all ground facts that are logically entailed by the program. This contrasts with the standard semantics of first order logic that would also allow for other models. This difference carries over to StarAI, where probabilistic logic programs and Markov Logic inherit their semantics from logic programming, respectively first order logic. This explains, for instance, why Markov Logic's semantics boils down to a maximum entropy approach when a theory has multiple models (such as $a \vee b$), cf. [De Raedt and Kimmig, 2015; De Raedt *et al.*, 2016] for more details. On the other hand, logic programs are also the basis for the programming language Prolog, which implies that they can be used to specify traditional programs such as sorting and data structures such as lists through structured terms. This is relevant especially for those approaches to neural-symbolic computation that are used to synthesize programs from examples.

NeSy approaches follow the same natural expressivity order of the underlying logics. For instance, SL focuses only on the propositional setting. On the other hand, ∂ILP, NTPs and DiffLog are based on Datalog, which belongs to relational logic segment. LTNs and SBR use fuzzy logic to translate a general first-order logic theory into a training objective, either isolated or in conjunction with a supervised criterion. Just like Markov Logic, also RNM and NMLN use first-order logic to generate a random field. Finally, DeepProbLog, NLProlog and LRNN are examples of neural-symbolic logic programming frameworks.

The chosen type of logic has a significant impact on inference and learning. The more expressive a logic, the harder the inference and learning become. For instance, for structure learning, the space of possible clauses for structure learning typically becomes exponentially larger as a more expressive class of logic is used. At the same time, many theories require a certain level of representation expressivity and cannot be expressed using simpler types of logic. For instance, programs cannot be represented using relational and propositional representations.

9 Open Challenges

To conclude, we list a number of challenges for NeSy, which deserve, in our opinion, more attention.

Semantics The statistical relational AI community and the probabilistic graphical model communities have devoted a lot of attention to the semantics of its models. This has resulted in a number of clear choices (such as directed vs. undirected, trace-based vs. possible world [Russell, 2015]), with corresponding strengths and weaknesses, which has allowed to clarify the relationships between the different models. Workshops have been held on that topic[1]. Furthermore, some researchers have investigated how to transform one type of model into another [Jaeger, 2008]. At the same time, the framework of weight model counting has emerged as a common assembly language for inference in many of these languages. The situation in neural-symbolic computation today is very much that of the early days in statistical relational learning, in which there were many competing formalisms, (sometimes characterized as the statistical relational learning alphabet soup). It would be great to get more insight into the semantics of neural-symbolic approaches and their relationships. This survey hopes to contribute towards this goal.

Probabilistic reasoning Although relatively few methods explore the integration of logical and neural methods from a probabilistic perspective, we believe that a probabilistic approach is a very good way to integrate the two [De Raedt *et al.*, 2019], but many open questions remain. Probabilistic inference is computationally more expensive than other approaches discussed in this paper. It would be interesting in future work to determine exactly what the benefit of this probabilistic approach is, and in which circumstances it arises.

Fuzzy semantics The selection of the t-norm fuzzy logic and the corresponding translation of the connectives is very heterogeneous in the literature.It is often not well clear which

[1]For instance, https://pps2018.luddy.indiana.edu/

	Dimension 1 (D)irected (U)ndirected	**Dimension 2** (M)odel-based (P)roof -based	**Dimension 3** (L)ogic (P)robability (N)eural	**Dimension 4** (L)ogic (P)robability (F)uzzy	**Dimension 5** (P)arameter (S)tructure	**Dimension 6** (S)ymbols (Sub)symbols	**Dimension 7** (P)ropositional (R)elational (FOL) (LP)
∂ILP [Evans and Grefenstette, 2018]	D	P	L+N	F	P + S	S	R
DeepProbLog [Manhaeve *et al.*, 2018]	D	P	L+P+N	P	P	S+Sub	LP
DiffLog [Si *et al.*, 2019]	D	P	L+N	F	P+S	S	R
LRNN [Šourek *et al.*, 2018]	D	P	L+N	F	P+S	S+Sub	LP
LTN [Donadello *et al.*, 2017]	U	M	L+N	F	P	Sub	FOL
NeuralLP [Yang *et al.*, 2017]	D	M	L+N	L	P	S	R
NLM [Dong *et al.*, 2019]	D	M	L+N	L	P+S	S	R
NLProlog [Weber *et al.*, 2019]	D	P	L+P+N	P	P+S	S+Sub	LP
NMLN [Marra and Kuželka, 2019]	U	M	L+P+N	P	P+S	S+Sub	FOL
NTP [Rocktäschel and Riedel, 2017]	D	P	L+N	L	P+S	S+Sub	R
RNM [Marra *et al.*, 2020]	U	M	L+P+N	P	P	S+Sub	FOL
SL [Xu *et al.*, 2018]	U	M	L+P+N	P	P	S+Sub	P
SBR [Diligenti *et al.*, 2017]	U	M	L+N	F	P	Sub	FOL
Tensorlog [Cohen *et al.*, 2017]	D	P	L+N	P	P	S+Sub	R

Table 1: Taxonomy of a (non-exhaustive) list of NeSy models according to the 7 dimensions outlined in the paper.

properties of Boolean logic a model is preserving, while there is the general tendency to consider fuzzy logic a continuous surrogate of Boolean logic without considering the well-known differences in semantics. There is a clear need for further studies in this field. On one hand, one could want to define new models which are natively fuzzy, thus not requiring the translation from Boolean logic. On the other hand, an interesting research direction concerns the characterisation of what is a good fuzzy approximation of Boolean logic in relation to a set of properties that one wishes to preserve.

Applications The current NeSy systems are not yet very mature from an application perspective and there are no real showcase applications yet. However, there have been promising proof-of-concepts in various fields. First, knowledge-base completion is a natural application for NeSy methods as the knowledge base is inherently symbolic and neural methods can be leveraged to generalise over the facts and predict those that are missing [Rocktäschel and Riedel, 2017; Donadello *et al.*, 2017]. Second, NeSy has contributed to various computer vision tasks by incorporating background knowledge in, for instance, image segmentation and semantic image interpretation [Donadello *et al.*, 2017; Alirezaie *et al.*, 2019]. Lastly, NeSy methods were used to aid natural language tasks such as question answering and visual question answering [Weber *et al.*, 2019; Yi *et al.*, 2018], where the latter is also closely related to the computer vision domain. Some of the methods mentioned show some promising early results in the domain of inductive programming [Evans and Grefenstette, 2018; Si *et al.*, 2019; Rocktäschel and Riedel, 2017], although they are still limited when compared to standard inductive logic programming systems. Developing real-life applications of NeSy is one of the most challenging and pressing open questions for the field.

Structure learning While significant progress has been made on learning the structure of purely relational models (without probabilities), learning StarAI models remains a major challenge due to the complexity of inference and the combinatorial nature of the problem. Incorporating neural aspects complicates the problem even more. NeSy methods have certainly shown potential for addressing this problem (Section 6), but the existing methods are still limited and mostly domain-specific which impedes their wide application. For instance, the current systems that support structure learning require user effort to specify the clause templates or write a sketch of a model.

Scaling inference Scalable inference is a major challenge for StarAI and therefore also for NeSy approaches with an explicit logical or probabilistic reasoning component. Investigating to which extent neural methods can help with this challenge by means of lifted (exploiting symmetries in models) or approximate inference, as well as reasoning from the intermediate representations [Abboud *et al.*, 2020], are promising future research directions.

Data efficiency A major advantage of StarAI methods, as compared to neural ones, is their data efficiency – StarAI methods can efficiently learn from small amounts of data, whereas neural methods are data hungry. On the other hand, StarAI methods do not scale to big data sets, while neural methods can easily handle them. We believe that understanding how these methods can help each other to overcome their complementary weaknesses, is a promising research direction.

Symbolic representation learning The effectiveness of deep learning comes from the ability to change the representa-

tion of the data so that the target task becomes easier to solve. The ability to change the representation on the symbolic level as well would significantly increase the capabilities of NeSy systems. This is a major open challenge for which neurally inspired methods could help achieve progress [Cropper, 2019; Dumančić *et al.*, 2019].

Acknowledgements

Robin Manhaeve and Sebastijan Dumančić are funded by the Research Foundation-Flanders (FWO). This work has also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No [694980] SYNTH: Synthesising Inductive Data Models) and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[Abboud *et al.*, 2020] Ralph Abboud, İsmail İlkan Ceylan, and Thomas Lukasiewicz. Learning to reason: Leveraging neural networks for approximate DNF counting. In *AAAI*, 2020.

[Alirezaie *et al.*, 2019] Marjan Alirezaie, Martin Längkvist, Michael Sioutis, and Amy Loutfi. Semantic referee: A neural-symbolic framework for enhancing geospatial semantic segmentation. *Semantic Web*, 10, 2019.

[Bach *et al.*, 2017] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *J. Mach. Learn. Res.*, 18, 2017.

[Bader and Hitzler, 2005] Sebastian Bader and Pascal Hitzler. Dimensions of neural-symbolic integration - A structured survey. *CoRR*, abs/cs/0511042, 2005.

[Besold *et al.*, 2017] Tarek R. Besold, Artur S. d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro M. Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *CoRR*, abs/1711.03902, 2017.

[Bošnjak *et al.*, 2017] Matko Bošnjak, Tim Rocktäschel, Jason Naradowsky, and Sebastian Riedel. Programming with a differentiable forth interpreter. In *ICML*, 2017.

[Cohen *et al.*, 2017] William W. Cohen, Fan Yang, and Kathryn Mazaitis. Tensorlog: Deep learning meets probabilistic dbs. *CoRR*, abs/1707.05390, 2017.

[Cropper, 2019] Andrew Cropper. Playgol: Learning programs through play. In *IJCAI 2019*, 2019.

[Dai *et al.*, 2019] Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. Bridging machine learning and logical reasoning by abductive learning. In *NeurIPS*, 2019.

[Darwiche, 2011] Adnan Darwiche. Sdd: A new canonical representation of propositional knowledge bases. In *IJCAI*, 2011.

[d'Avila Garcez *et al.*, 2019] Artur S. d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *FLAP*, 6, 2019.

[De Raedt and Kimmig, 2015] Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100, 2015.

[De Raedt *et al.*, 2016] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Morgan & Claypool Publishers, 2016.

[De Raedt *et al.*, 2019] Luc De Raedt, Robin Manhaeve, Sebastijan Dumančić, Thomas Demeester, and Angelika Kimmig. Neuro-symbolic= neural+ logical+ probabilistic. In *NeSy @ IJCAI*, 2019.

[De Raedt, 2008] Luc De Raedt. *Logical and relational learning*. Springer, 2008.

[Demeester *et al.*, 2016] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. In *EMNLP*, 2016.

[Diligenti *et al.*, 2017] Michelangelo Diligenti, Marco Gori, and Claudio Saccà. Semantic-based regularization for learning and inference. *Artif. Intell.*, 244, 2017.

[Donadello *et al.*, 2017] Ivan Donadello, Luciano Serafini, and Artur S. d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *IJCAI*, 2017.

[Dong *et al.*, 2019] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. In *ICLR*, 2019.

[Dumančić *et al.*, 2019] Sebastijan Dumančić, Tias Guns, Wannes Meert, and Hendrik Blockeel. Learning relational representations with auto-encoding logic programs. In *IJCAI*, 2019.

[Ellis *et al.*, 2018a] Kevin Ellis, Lucas Morales, Mathias Sablé-Meyer, Armando Solar-Lezama, and Josh Tenenbaum. Learning libraries of subroutines for neurally-guided bayesian program induction. In *NeurIPS*, 2018.

[Ellis *et al.*, 2018b] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. Learning to infer graphics programs from hand-drawn images. In *NeurIPS*, 2018.

[Evans and Grefenstette, 2018] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *J. Artif. Intell. Res.*, 61, 2018.

[Fierens *et al.*, 2015] Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming*, 15, 2015.

[Flach, 1994] Peter Flach. *Simply Logical: Intelligent Reasoning by Example*. John Wiley & Sons, Inc., 1994.

[Friedman *et al.*, 1999] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, 1999.

[Gebser *et al.*, 2012] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Answer set solving in practice. *Synthesis lectures on artificial intelligence and machine learning*, 6, 2012.

[Getoor and Taskar, 2007] L. Getoor and B. Taskar, editors. *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

[Giannini *et al.*, 2018] Francesco Giannini, Michelangelo Diligenti, Marco Gori, and Marco Maggini. On a convex logic fragment for learning and reasoning. *IEEE TFS*, 27, 2018.

[Gutmann *et al.*, 2008] Bernd Gutmann, Angelika Kimmig, Kristian Kersting, and Luc De Raedt. Parameter learning in probabilistic databases: A least squares approach. In *ECML&PKDD*, 2008.

[Jaeger, 2008] Manfred Jaeger. Model-theoretic expressivity analysis. In Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors, *Probabilistic Inductive Logic Programming - Theory and Applications*, volume 4911 of *LNCS*. Springer, 2008.

[Kalyan *et al.*, 2018] Ashwin Kalyan, Abhishek Mohta, Oleksandr Polozov, Dhruv Batra, Prateek Jain, and Sumit Gulwani. Neural-guided deductive search for real-time program synthesis from examples. In *ICLR*, 2018.

[Kersting and De Raedt, 2007] Kristian Kersting and Luc De Raedt. Bayesian logic programming: Theory and tool. In L. Getoor and B. Taskar, editors, *An introduction to Statistical Relational Learning*. MIT Press, 2007.

[Kok and Domingos, 2005] Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In *ICML*, 2005.

[Koller and Friedman, 2009] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[Lippi and Frasconi, 2009] Marco Lippi and Paolo Frasconi. Prediction of protein beta-residue contacts by markov logic networks with grounding-specific weights. *Bioinform.*, 25, 2009.

[Lowd and Domingos, 2007] Daniel Lowd and Pedro Domingos. Efficient weight learning for markov logic networks. In *ECML&PKDD*, 2007.

[Manhaeve *et al.*, 2018] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *NeurIPS*, 2018.

[Mao *et al.*, 2019] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*, 2019.

[Marra and Kuželka, 2019] Giuseppe Marra and Ondrej Kuželka. Neural markov logic networks. *CoRR*, abs/1905.13462, 2019.

[Marra *et al.*, 2020] Giuseppe Marra, Michelangelo Diligenti, Francesco Giannini, Marco Gori, and Marco Maggini. Relational neural machines. In *ECAI in press*, 2020.

[Minervini *et al.*, 2017] Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. In *UAI*, 2017.

[Minervini *et al.*, 2020] Pasquale Minervini, Matko Bošnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. Differentiable reasoning on large knowledge bases and natural language. In *AAAI*, 2020.

[Muggleton, 1996] Stephen Muggleton. Stochastic logic programs. *Advances in inductive logic programming*, 32, 1996.

[Poole, 2008] David Poole. The independent choice logic and beyond. In *Probabilistic Inductive Logic Programming - Theory and Applications*, volume 4911 of *LNCS*. Springer, 2008.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro M. Domingos. Markov logic networks. *Machine Learning*, 62, 2006.

[Rocktäschel and Riedel, 2017] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *NIPS*, 2017.

[Rocktäschel *et al.*, 2015] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *NAACL HLT*, 2015.

[Russell, 2015] Stuart Russell. Unifying logic and probability. *Communications of the ACM*, 58, 2015.

[Si *et al.*, 2019] Xujie Si, Mukund Raghothaman, Kihong Heo, and Mayur Naik. Synthesizing datalog programs using numerical relaxation. In *IJCAI*, 2019.

[Valkov *et al.*, 2018] Lazar Valkov, Dipak Chaudhari, Akash Srivastava, Charles A. Sutton, and Swarat Chaudhuri. Houdini: Lifelong learning as program synthesis. In *NeurIPS*, 2018.

[Van den Broeck *et al.*, 2017] Guy Van den Broeck, Dan Suciu, et al. Query processing on probabilistic data: A survey. *Foundations and Trends® in Databases*, 7, 2017.

[van Krieken *et al.*, 2020] Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy logic operators. *CoRR*, abs/2002.06100, 2020.

[Wang and Pan, 2019] Wenya Wang and Sinno Jialin Pan. Integrating deep learning with logic fusion for information extraction. *CoRR*, abs/1912.03041, 2019.

[Weber *et al.*, 2019] Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. Nlprolog: Reasoning with weak unification for question answering in natural language. In *ACL*, 2019.

[Xu *et al.*, 2018] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, 2018.

[Yang *et al.*, 2017] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, 2017.

[Yi *et al.*, 2018] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *NeurIPS*, 2018.

[Zuidberg Dos Martires *et al.*, 2019] Pedro Zuidberg Dos Martires, Vincent Derkinderen, Robin Manhaeve, Wannes Meert, Angelika Kimmig, and Luc De Raedt. Transforming probabilistic programs into algebraic circuits for inference and learning. In *Program Transformations for ML Workshop at NeurIPS*, 2019.

[Šourek *et al.*, 2018] Gustav Šourek, Vojtech Aschenbrenner, Filip Železný, Steven Schockaert, and Ondrej Kuželka. Lifted relational neural networks: Efficient learning of latent relational structures. *J. Artif. Intell. Res.*, 62, 2018.