Neural Semirings

Pedro Zuidberg Dos Martires[1]

[1]KU Leuven, Belgium

Abstract

From an abstract algebra perspective, reasoning tasks in symbolic artificial intelligence can be formulated in terms of commutative semiring operations. Usually, the reasoning task, e.g. probabilistic reasoning or fuzzy reasoning, is fixed upfront as part of the problem specification. In this paper we relax this assumption and render the reasoning operators (the semiring operations) subject to learning by replacing predefined operations with learnable neural networks. This unlocks the **learn to reason** paradigm in the semiring reasoning setting.

Keywords

neuro-symbolic, semirings, learn to reason

1. Introduction

At the core of symbolic AI lie Boolean functions consisting of Boolean literals and Boolean connectives between these literals. An interesting representation of Boolean functions, for our purposes, is the so-called negation normal form: the only connectives allowed are AND-gates, OR-gates, and negations. The latter can only occur on the literals themselves. These three gates are universal, which means they allow us to represent any Boolean function. At an abstract level, reasoning tasks in symbolic AI are concerned with computing quantities of these logic functions.

In a symbolic AI system, the process of reasoning can be formulated using algebraic structures called (commutative) semirings. This means that a symbolic AI is a computer program that performs its computations while adhering to the **structure imposed by semirings** [1, 2, 3, 4].

A well-known semiring in symbolic AI is the probability semiring. This is the semiring used for inference in the probabilsitic programming language ProbLog [5] but also in its neuro-symbolic extension DeepProbLog [6]. In the case of the Boolean literals representing probabilistic Boolean variables, the probability semiring allows for the computation of the probability of a Boolean function being satisfied:

$$p\left(\bigvee_{I\in M(\psi)}\bigwedge_{\ell\in I}\ell\right)=\sum_{I\in M(\psi)}\prod_{\ell\in I}p(\ell) \tag{1}$$

On the left hand-side we have the (probabilistic) Boolean function for which we want to compute the probability. The symbol ψ denotes the entire Boolean function, $M(\psi)$ denotes the models of ψ, and the ℓ's are the literals, i.e. Boolean atoms and their negation.

15th International Workshop on Neural-Symbolic Learning and Reasoning

pedro.zudo@kuleuven.be (. P. Zuidberg Dos Martires)

Note how, on the right, we push the probability functional p inside and apply it directly to the literals instead of the entire formula. This is not always possible but for Boolean functions that adhere to specific constraints [7] it enables poly-time probabilistic inference (in contrast to the usual computational hardness of probabilistic reasoning). Note also how the $\bigvee$ changed to a sum operation and the $\bigwedge$ changed to a product operation. For probabilities, these operations are simply the addition and multiplication of real values between 0 and 1. In the probability semiring the real numbers between 0 and 1 are called the elements of the semiring.

Researchers soon enough realized [8, 9, 2, 1] that different reasoning tasks common in AI, such as probabilistic inference, the Viterbi algorithm, or forward differentiation, can be formulated as computations in different semirings. We denote this generalized computation by α and we call α the labeling functional:

$$\alpha\left(\bigvee_{I\in M(\psi)}\bigwedge_{\ell\in I}\ell\right)=\bigoplus_{I\in M(\psi)}\bigotimes_{\ell\in I}\alpha(\ell) \tag{2}$$

Comparing Equation 2 to Equation 1, we see that, structurally, nothing changed. The only difference is the quantity that we compute (denoted by $\alpha(\cdot)$) and the generalization of the summation and the product in the first equation to an abstract summation and multiplication. This general framework to perform reasoning tasks was coined algebraic model counting (AMC) in [2], where the authors do also give an overview of which reasoning tasks can be formulated as a semiring computation and under which conditions.

Contribution

Learning within the semiring reasoning framework of algebraic model counting has mainly been concerned with learning a mapping from literals to elements of a given semiring. For instance, DeepProbLog takes high-dimensional data, such as the image of a number  , and uses a neural net to map it to the probability of the image showing the number 7.

We will lift this restriction and allow the semiring operations (i.e. the reasoning operations) to be subject to learning as well. We replace predefined semiring operations by neural networks that adhere to the algebraic properties of semirings, while still being trainable. Swapping out predefined semiring operations for learnable neural networks unlocks the **learn to reason** paradigm [10] in the semiring reasoning framework.

Paper Organization

We start with the necessary mathematical background on algebraic model counting (Section 2), which also includes a formal definition of semirings. In Section 3, we introduce learnable semirings by showing how the semiring operations can be represented through neural networks and in Section 4 we sketch a possible learning set-up for neural semirings. We end the paper by discussing related work (including graph neural networks) in Section 5, and giving concluding thoughts in Section 6.

2. Preliminaries on Algebraic Model Counting

Here we give the formal definitions of algebraic structures called monoids and semirings, and also of algebraic model counting. These will allow us to learn to reason in an algebraic model counting context by introducing neural semirings.

Definition 2.1 (Monoid). *A monoid is an algebraic structure* $(\mathbb{S}, \circ, e)$ *equipping a set of elements* $\mathbb{S}$ *with a closed binary operation* $\circ : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$ *and where* $e \in \mathbb{S}$ *is the so-called neutral element. Furthermore the following properties have to be satisfied:*

1. *associativity:* $\forall a, b, c \in \mathbb{S}$ *it holds that* $(a \circ b) \circ c = a \circ (b \circ c)$.
2. *neutral element: there exists an element* $e \in \mathbb{S}$ *called the neural such that* $\forall s \in \mathbb{S}$ *it holds that* $e \circ s = s \circ a = s$.

Additionally, a monoid is called commutative if the structure $(\mathbb{S}, \circ, e)$ *satisfies:*

3. *commutativity:* $\forall a, b \in \mathbb{S}$ *it holds that* $a \circ b = b \circ a$.

Definition 2.2 (Semiring). *A semiring is an algebraic structure* $(\mathbb{S}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$ *equipping a set of elements* $\mathbb{S}$ *with closed binary operations* $\oplus$ *and* $\otimes$*, which are usually referred to as addition and multiplication, respectively. Furthermore the following properties hold:*

1. *the algebraic structure* $(\mathbb{S}, \oplus, e^{\oplus})$ *is a commutative monoid.*
2. *the algebraic structure* $(\mathbb{S}, \otimes, e^{\otimes})$ *is a monoid.*
3. *distributivity: the multiplication distributes over the addition, i.e.* $\forall a, b, c \in \mathbb{S}$ *it holds that* $a \otimes (b \oplus c) = a \otimes b \oplus a \otimes c$.
4. *annihilating element: the neutral element of the addition monoid* $e^{\oplus}$ *is the annihilating element of the multiplication, i.e.* $\forall s \in \mathbb{S}$ *it holds that* $e^{\oplus} \otimes s = s \otimes e^{\oplus} = e^{\oplus}$.

Additionally, a semiring is called commutative if the structure $(\mathbb{S}, \otimes, e^{\otimes})$ *is a commutative monoid:*

5. *commutativity of multiplication:* $\forall a, b \in \mathbb{S}$ *it holds that* $a \otimes b = b \otimes a$.

Definition 2.3 (Algebraic model counting [2]). *Given are a propositional logic theory* ψ *over a set of variables* $\mathscr{B}$*, a commutative semiring* $(\mathbb{S}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$*, and a labeling function* $\alpha : \mathbb{L} \to \mathbb{S}$*, with* $\mathbb{L}$ *being the set of variables in* $\mathscr{B}$ *and their negations. The algebraic model count of a theory* ψ *is then defined as:*

$$AMC(\psi, \alpha|\mathscr{B}) = \bigoplus_{I \in M(\psi)} \bigotimes_{\ell \in I} \alpha(\ell) \tag{3}$$

While elegant in its formulation, computing the algebraic model count is generally a computationally hard problem, #P-hard to be precise [11]. This is where knowledge compilation [12] comes to our rescue.

Knowledge compilation is the process of transforming a propositional logic formula into a form that allows for **polytime computation** of algebraic model counts. Although the knowledge compilation step is itself computationally hard, the overall procedure yields a net benefit when a logic formula has to be evaluated multiple times, possibly with different labeling

functionals. Combining AMC and knowledge compilation is a perfect match in an iterative learning setting such as stochastic gradient descent, which has already been demonstrated in the neuro-symbolic literature [13, 6].

A well-known target representation for logic formulas used in knowledge compilation are so-called *binary decision diagrams* (BDD) [14]. The advantage of using BDDs over formulas in conjunctive normal forms (c.f. Equation 2) is that we can alternate conjunctions and disjunctions, which in turn means that we can compute algebraic model counts by repeatedly nesting $\oplus$ and $\otimes$ operations. Fundamentally, knowledge compilation turns the computation of an algebraic model count into a dynamic programming algorithm, where the target representation of the propositional formula, e.g. a BDD, determines the order in which the $\oplus$ adn $\otimes$ operations are performed. We refer the reader to [2] for a detailed account.

3. Neural Semirings

Commutative neural semirings are algebraic structures of the form $(\mathbb{S}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$ (cf. Definition 2.2), where the binary operations $\otimes$ and $\oplus$ are neural networks, and which adhere to the properties of commutative semirings. In order to construct neural semirings we draw inspirations from so-called deeps sets [15, 16]. Deep sets are neural networks that are invariant under permutations of the inputs:

$$f(x_1, \ldots, x_M) = f(x_{\pi(1)}, \ldots, x_{\pi(M)}) \tag{4}$$

for all possible permutations π. We can now explicitly construct a permutation-invariant function f by assuming sum-decomposability:

$$f(X = \{x_1, \ldots, x_M\}) = \rho\left(\sum_{x \in X} \phi(x)\right) \tag{5}$$

Here, $\phi(x)$ is a vector and the summation is performed element-wise. The tuple (ρ, ϕ) is called a sum-decomposition of f. Reducing the number of inputs to two immediately yields commutative binary operations that we can use for the definition of the $\otimes$ and $\oplus$ operations of neural semirings:

$$\oplus(\alpha_1, \alpha_2) = \rho_{\oplus}\left(\phi_{\oplus}(\alpha_1) + \phi_{\oplus}(\alpha_2)\right) = \rho_{\oplus}\left(\phi_{\oplus}(\alpha_2) + \phi_{\oplus}(\alpha_1)\right) = \oplus(\alpha_2, \alpha_1) \tag{6}$$

$$\otimes(\alpha_1, \alpha_2) = \rho_{\otimes}\left(\phi_{\otimes}(\alpha_1) + \phi_{\otimes}(\alpha_2)\right) = \rho_{\otimes}\left(\phi_{\otimes}(\alpha_2) + \phi_{\otimes}(\alpha_1)\right) = \otimes(\alpha_2, \alpha_1) \tag{7}$$

where $\alpha_1, \alpha_2 \in \mathbb{S}$. Assuming sum decomposability of the neural semiring operations results in satisfying the commutativity property.

The next property that we would like the neural semiring operators to satisfy is associativity. That is, we would like to following equation to hold:

$$\oplus\left(\alpha_1, \oplus(\alpha_2, \alpha_3)\right) = \oplus(\oplus(\alpha_1, \alpha_2), \alpha_3)) \tag{8}$$

$$\Leftrightarrow \rho_{\oplus}\left[\phi_{\oplus}(\alpha_1) + \phi_{\oplus}\left(\rho_{\oplus}\left[\phi_{\oplus}(\alpha_2) + \phi_{\oplus}(\alpha_3)\right]\right)\right] = \rho_{\oplus}\left[\phi_{\oplus}\left(\rho_{\oplus}\left[\phi_{\oplus}(\alpha_1) + \phi_{\oplus}(\alpha_2)\right]\right) + \phi_{\oplus}(\alpha_3)\right] \tag{9}$$

We satisfy Equation 9 by imposing $\rho_{\oplus} = \phi_{\oplus}^{-1}$. Imposing additionally an analog condition for the $\otimes$ operation we get the following sum-decomposition of our neural semiring operators:

$$\oplus(\alpha_1, \alpha_2) = \phi_{\oplus}^{-1}\Big[\phi_{\oplus}(\alpha_1) + \phi_{\oplus}(\alpha_2)\Big] \tag{10}$$

$$\otimes(\alpha_1, \alpha_2) = \phi_{\otimes}^{-1}\Big[\phi_{\otimes}(\alpha_1) + \phi_{\otimes}(\alpha_2)\Big] \tag{11}$$

Writing the neural semiring operations in this fashion alludes to an encoder/decoder interpretation. Let us consider the $\otimes$ operation: $\phi_{\otimes}$ takes an input α, which is an element of the semiring, and enccodes it into a latent space where the $\otimes$-operation reduces to a mere element-wise addition. $\phi_{\otimes}^{-1}$ then takes this results of the element-wise addition and decodes it back into the space of semiring elements. This interpretation opens up an avenue for implementing associative neural operators. We could, for instance, fall back on an encoder/decoder architecture. While this is a standard architecture, we would only have an approximation of associativity: the decoder will never be the exact inverse of the encoder. Alternatively, we could enforce $\rho = \phi^{-1}$ by construction. The key challenge in this case would be the efficient construction of the inverse of ϕ. Here, techniques from the neural network community [17] or the normalizing flow community [18] could be used to construct neural networks that are easily invertible.

Next, we turn our attention to the neutral elements and more specifically to their existence. Assuming the sum-decomposition in Equation 10 we have to satisfy the following equality:

$$\oplus(\alpha, e^{\oplus}) = \alpha \tag{12}$$

This is satisfied for every α if we simply take $\phi_{\oplus}(e^{\oplus}) = \vec{0}$. We then obtain the neutral element of the addition with $\phi_{\oplus}^{-1}(\vec{0})$. This means that a neutral element exists if $\vec{0}$ is included in the co-domain of $\phi_{\oplus}$. We can also follow a similar reasoning for the $\otimes$-operation.

Until now we have only been considering properties of neural semiring operations that only concern one of the two operations at a time. However, what distinguishes semiring from monoids (and groups from that matter, which are moinoids with invertible elements) is the presence of properties that intertwingle the $\oplus$ and the $\otimes$ operations. Let us first have a look at the annihilating property of semirings and then study the distributivity property.

With the commutativity of the $\otimes$-operation already holding, we need to have $e^{\oplus} \otimes \alpha = e^{\oplus}$ in order to satisfy the annihilating property. Plugging in the sum-decomposition of Equation 11 we get:

$$e^{\oplus} \otimes \alpha = e^{\oplus} \Leftrightarrow \phi_{\otimes}^{-1}\Big[\phi_{\otimes}(e^{\oplus}) + \phi_{\otimes}(\alpha)\Big] = e^{\otimes} \tag{13}$$

$$\Leftrightarrow \phi_{\otimes}(e^{\oplus}) + \phi_{\otimes}(\alpha) = \phi_{\otimes}(e^{\otimes}) \tag{14}$$

$$\Leftrightarrow \phi_{\otimes}(e^{\oplus})_i = \pm\infty \tag{15}$$

The index i in the last line indexes the elements of the vector $\phi_{\otimes}(e^{\oplus})$ and the equation itself tells us that each element of the vector has to be either plus or minus infinity. At first this result seems unhinged. However, remembering that the neutral element of the addition in the log-probability semiring is also $-\infty$ makes this result plausible. Equation 15 is the vector version of the neutral element of the addition in the log-probability semiring.

The trickiest property to enforce by construction in neural semirings is the distributivity of the multiplication over the addition. Or more formally:

$$\begin{aligned}&\phi_{\otimes}^{-1}\Big[\phi_{\otimes}(\alpha_1) + \phi_{\otimes}\Big(\phi_{\oplus}^{-1}(\phi_{\oplus}(\alpha_2) + \phi_{\oplus}(\alpha_3))\Big)\Big]\\ =\ &\phi_{\oplus}^{-1}\Big[\phi_{\oplus}\Big(\phi_{\otimes}^{-1}(\phi_{\otimes}(\alpha_1) + \phi_{\otimes}(\alpha_2))\Big) + \phi_{\oplus}\Big(\phi_{\otimes}^{-1}(\phi_{\otimes}(\alpha_1) + \phi_{\otimes}(\alpha_3))\Big)\Big]\end{aligned} \tag{16}$$

where we again used the sum-decompositions from Equations 10 and 11. One possibility to enforce the properties is to, again pick functions $\phi_{\otimes}^{-1}$, $\phi_{\otimes}$, $\phi_{\oplus}^{-1}$ and $\phi_{\oplus}$ that satisfy Equation 16. For instance, we could pick:

$$\phi_{\otimes}^{-1}(\alpha) = \exp(\alpha) \qquad \phi_{\otimes}(\alpha) = \log(\alpha)$$
$$\phi_{\oplus}^{-1}(\alpha) = |\alpha|^{1/p} \qquad \phi_{\oplus}(\alpha) = \alpha^p$$

Here we assume that α is a vector and that the functions operate element-wise. In this case the $\otimes$-operation is equivalent to an element-wise multiplication of two vectors and the addition is in fact an element-wise p-norm. Note that each dimension of the α-vector can have a different parameter p. This means that every dimension is aggregated using a different p-norm. In a neural semiring it would be this vector of parameters that would be learned, possibly via a neural network.

Alternatively, we could enforce distributivity through an extra loss term. Assume therefore that we have two neural networks $\phi_{\otimes}$ and $\phi_{\oplus}$ and we know how to efficiently compute their inverses $\phi_{\otimes}^{-1}$ and $\phi_{\oplus}^{-1}$. Distributivity can then be enforced approximately by adding a term of the form:

$$\begin{aligned}\mathscr{L}_{dist}(\alpha_1, \alpha_2, \alpha_3) = \Bigg[\quad &\phi_{\otimes}^{-1}\Big[\phi_{\otimes}(\alpha_1) + \phi_{\otimes}\Big(\phi_{\oplus}^{-1}(\phi_{\oplus}(\alpha_2) + \phi_{\oplus}(\alpha_3))\Big)\Big]\\ &- \phi_{\oplus}^{-1}\Big[\phi_{\oplus}\Big(\phi_{\otimes}^{-1}(\phi_{\otimes}(\alpha_1) + \phi_{\otimes}(\alpha_2))\Big) + \phi_{\oplus}^{-1}\Big[\phi_{\oplus}\Big(\phi_{\otimes}^{-1}(\phi_{\otimes}(\alpha_1) + \phi_{\otimes}(\alpha_3))\Big)\Big]^2\end{aligned}$$

to the learning objective, which we would like to optimize. Here we picked the squared error loss but other options might be viable as well. We will elaborate more on this distributivity loss term in the next section.

4. Learning with Neural Semirings

The conceptually easiest learning paradigm applicable to learning with neural semirings is fully supervised learning. In such a setting we assume to be given a set of literals and computation graphs as input and a corresponding set of labels as supervision. The computation graph of each training example first takes the literals and maps (encodes) them to a set of vectors. These vectors are then combined using learnable sum-product operations. Figure 1 gives a conceptual overview of the learning setup. An open question is where we obtain the computation graphs from. One possibility would be to have as input a weighted Boolean formula (for instance representing a weighted graph) and knowledge-compiling the formula into a tractable and differentiable representation (cf. end of Section 2).

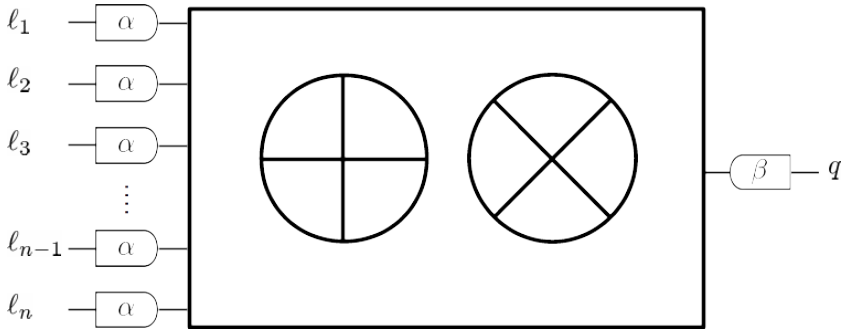

Figure 1: Schematic overview of a neural semiring computation. The diagram is read left to right. We have as input the literals ℓ_i, which are passed through the labeling functional α. The middle part abstractly represents recursively nested sum-product computations, i.e. it represents the the computation graph given as as input to the learning problem. In the end, we obtain the value of the query q by passing the output of the nested sum-product through a learnable decoding function β whose result we can compare to the supervision (the label in the training data set).

Using the notation presented in Figure 1, supervised learning in a neural semiring setting consists of minimizing a loss function where we compare $\beta(\ell_1, \ldots, \ell_n)$ and the corresponding label L for each datum in the training data set. The parameters to be optimized are the parameters of the encoder α, of the $\oplus$ and $\otimes$ operators, and of the decoder β. We call this loss function the *commutative semiring computation graph loss* (CSCG-loss) and denote it by $\mathscr{L}_{CSCG}$.

At the end of Section 3 we discussed the possibility of relaxing the semiring properties on the $\oplus$ and $\otimes$ operator by enforcing distributivity through an extra term in the loss. The CSCG-loss is then written as:

$$\mathscr{L}_{CSCG}(\ell_1, \ldots, \ell_2, L) \approx \mathscr{L}_{CSCG^*}(\ell_1, \ldots, \ell_2, L) + \sum_{\ell_i, \ell_j, \ell_k} \mathscr{L}_{dist}(\alpha(\ell_i), \alpha(\ell_j), \alpha(\ell_k)) \tag{17}$$

In above equation $\mathscr{L}_{CSCG^*}$ denotes the CSCG-loss but without the distributivity being enforced by construction on the semiring operators. Furthermore, we can interpret the distributivity loss (on the right) as a regularization term for $\mathscr{L}_{CSCG^*}$.

5. Related Work

Historically, artificial intelligence can be divided into two main camps. The so-called symbolic camp and the sub-symbolic or connectionist camp. The synthesis of both streams has been dubbed neuro-symbolic AI [19, 20, 21].

Formulating neuro-symbolic AI as semiring computations has usually been done in approaches originating in the symbolic camp, such as DeepProbLog [6], Tensorlog [22], Modular Neural Networks [23], or also Logic Tensor Networks [24].

All four approaches have in common that they are able to learn the labeling functional, which we denote by α, but also that they pre-define the reasoning operations ($\oplus$ and $\otimes$). For example, in TensorLog the labeling functional α produces vectors (elements of the semiring). The $\oplus$ and

$\otimes$ operations are then defined as element-wise addition and multiplication of these vectors. This effectively means the embedding space in which reasoning is performed is chosen up front.

A disadvantage of fixing the semiring operations (i.e. reasoning operations) up front is that this severely restricts the embedding spaces that can be learned. For example, the modular neural networks proposed by Andreas et al. [23] subdivide an image by overlaying a grid. Each grid cell then corresponds to an entry in a tensor. In semiring terminology this is the labeling step producing elements of the semiring. Elements are combined using predefined point-wise tensor operations. This restricts reasoning and learning to grid worlds.

In recent years neuro-symbolic AI has also experienced a revival in the connectionist camp. The most prominent approach are probably graph neural networks [25], which were born out of necessity to use structured data within neural networks. Inference and learning in graph neural networks is performed by running a message passing algorithm on the structure provided by a graph. Interestingly, message passing is nothing but a sum-product algorithm that can be formulated as a semiring computation.

In comparison to the research originating in the symbolic camp, we make two interesting observations for GNNs and related approaches [26, 27]. 1) Contrary to symbolic approaches to neuro-symbolic AI, connectionist approaches have been paying attention to learnable aggregation functions, i.e. a learnable addition operator [15, 16, 28]. 2) Although the connectionist camp has started to formulate deep learning in terms of abstract algebra, cf. geometric deep leanring [29], this has only been done with regards to groups (monoids with an invertible elements). Commutative semirings have been ignored so far.

6. Conclusions and Future Work

This paper introduces the concept of neural semirings, a framework that has the potential to unlock the learn to reason paradigm in a semiring setting. In contrast to prior work, neural semirings are not fixed up front anymore but learned from data and contrary to geometric deep learning we formulate neuro-symbolic AI in terms of semirings instead of groups, where the additional complication of distributivity emerges.

Future work will obviously include an experimental evaluation of neural semirings and a more detailed study of the ingredients to neural semirings. For instance, we might want to investigate alternatives to sum-decompostions of the semiring operators such as product-decompostions. We will also have to study in more detail the connections of neural semirings to approaches in the connectionsist camp of neuro-symbolic AI, in particular the work of Xu et al. [10].

Acknowledgments

Pedro Zuidberg Dos Martires has received support from the KU Leuven Special Research Fund.

References

[1] Z. Li, J. Eisner, First-and second-order expectation semirings with applications to minimum-risk training on translation forests, in: Conference on Empirical Methods in Natural

Language Processing, 2009.
[2] A. Kimmig, G. Van den Broeck, L. De Raedt, Algebraic model counting, Journal of Applied Logic (2017).
[3] F. Orsini, P. Frasconi, L. De Raedt, kproblog: an algebraic prolog for machine learning, Machine Learning (2017).
[4] V. Belle, L. De Raedt, Semiring programming: A semantic framework for generalized sum product problems, International Journal of Approximate Reasoning (2020).
[5] D. Fierens, G. Van den Broeck, J. Renkens, D. Shterionov, B. Gutmann, I. Thon, G. Janssens, L. De Raedt, Inference and learning in probabilistic logic programs using weighted boolean formulas, Theory and Practice of Logic Programming (2015).
[6] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. De Raedt, Deepproblog: Neural probabilistic logic programming, in: Advances in Neural Information Processing Systems, 2018.
[7] A. Darwiche, Modeling and reasoning with Bayesian networks, Cambridge university press, 2009.
[8] J. Goodman, Semiring parsing, Computational Linguistics (1999).
[9] A. Kimmig, G. Van den Broeck, L. De Raedt, An algebraic prolog for reasoning about possible worlds, in: AAAI, 2011.
[10] K. Xu, J. Li, M. Zhang, S. S. Du, K.-i. Kawarabayashi, S. Jegelka, What can neural networks reason about?, in: ICLR, 2020.
[11] L. G. Valiant, The complexity of computing the permanent, Theoretical Computer Science 8 (1979) 189–201.
[12] A. Darwiche, P. Marquis, A knowledge compilation map, Journal of Artificial Intelligence Research 17 (2002) 229–264.
[13] J. Xu, Z. Zhang, T. Friedman, Y. Liang, G. Broeck, A semantic loss function for deep learning with symbolic knowledge, in: International Conference on Machine Learning, PMLR, 2018, pp. 5502–5511.
[14] R. E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation, IEEE Transactions on Computers 35 (1986).
[15] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, A. J. Smola, Deep sets, in: NIPS, 2017.
[16] E. Wagstaff, F. Fuchs, M. Engelcke, I. Posner, M. A. Osborne, On the limitations of representing functions on sets, in: ICML, 2019.
[17] Y. Song, C. Meng, S. Ermon, Mintnet: Building invertible neural networks with masked convolutions, in: Advances in Neural Information Processing Systems, 2019.
[18] D. Rezende, S. Mohamed, Variational inference with normalizing flows, in: ICML, 2015.
[19] T. R. Besold, A. d. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kühnberger, L. C. Lamb, D. Lowd, P. M. V. Lima, et al., Neural-symbolic learning and reasoning: A survey and interpretation, arXiv preprint arXiv:1711.03902 (2017).
[20] A. Garcez, M. Gori, L. Lamb, L. Serafini, M. Spranger, S. Tran, Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning, Journal of Applied Logics (2019).
[21] A. d. Garcez, L. C. Lamb, Neurosymbolic ai: The 3rd wave, arXiv preprint arXiv:2012.05876 (2020).

[22] W. Cohen, F. Yang, K. R. Mazaitis, Tensorlog: A probabilistic database implemented using deep-learning infrastructure, Journal of Artificial Intelligence Research (2020).
[23] J. Andreas, M. Rohrbach, T. Darrell, D. Klein, Learning to compose neural networks for question answering, in: NAACL, 2016.
[24] I. Donadello, L. Serafini, A. D. Garcez, Logic tensor networks for semantic image interpretation, IJCAI (2017).
[25] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: IEEE International Joint Conference on Neural Networks, 2005.
[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural message passing for quantum chemistry, in: ICML, 2017.
[28] G. Pellegrini, A. Tibo, P. Frasconi, A. Passerini, M. Jaeger, Learning aggregation functions, arXiv preprint arXiv:2012.08482 (2020).
[29] M. M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, arXiv preprint arXiv:2104.13478 (2021).