Grounding Methods for Neural-Symbolic AI

Rodrigo Castellano Ontiveros[1] , **Francesco Giannini**[2] , **Marco Gori**[1] ,
Giuseppe Marra[3] and **Michelangelo Diligenti**[1]
[1]University of Siena, Italy
[2]Scuola Normale Superiore, Italy
[3]KU Leuven, Belgium
{rodrigo.castellano,marco.gori,michelangelo.diligenti}@unisi.it,
francesco.giannini@sns.it, giuseppe.marra@kuleuven.be

Abstract

A large class of Neural-Symbolic (NeSy) methods employs a machine learner to process the input entities, while relying on a reasoner based on First-Order Logic to represent and process more complex relationships among the entities. A fundamental role for these methods is played by the process of logic grounding, which determines the relevant substitutions for the logic rules using a (sub)set of entities. Some NeSy methods use an exhaustive derivation of all possible substitutions, preserving the full expressive power of the logic knowledge. This leads to a combinatorial explosion in the number of ground formulas to consider and, therefore, strongly limits their scalability. Other methods rely on heuristic-based selective derivations, which are generally more computationally efficient, but lack a justification and provide no guarantees of preserving the information provided to and returned by the reasoner. Taking inspiration from multi-hop symbolic reasoning, this paper proposes a parametrized family of grounding methods generalizing classic Backward Chaining. Different selections within this family allow us to obtain commonly employed grounding methods as special cases, and to control the trade-off between expressiveness and scalability of the reasoner. The experimental results show that the selection of the grounding criterion is often as important as the NeSy method itself.

1 Introduction

Neural-Symbolic (NeSy) methods [d'Avila Garcez *et al.*, 2009; Marra *et al.*, 2024] integrate the strengths of neural networks and symbolic reasoning, and heavily rely on the process of logic grounding, which bridges the semantic gap between neural representations and symbolic logic. However, classical grounding techniques in NeSy are impractical, and they do not take advantage of the generalization capabilities of NeSy approaches, which can deduce true facts using statistical methods without the need for a classical proof. In fact, the selection of appropriate grounding represents a significant challenge for NeSy methodologies. Existing approaches exhibit a dichotomy between exhaustive derivations that maintain full expressiveness of logical knowledge but suffer from scalability issues, and heuristic-based selective derivations that sacrifice logical coherence for computational efficiency. Indeed, the employed grounding process is often completely neglected in the NeSy literature, or relegated to a single sentence in the appendix. This hinders the possibility to reproduce, compare, and even fully understand the strengths and weaknesses of the different methods.

This paper aims to fill this gap by exploring diverse grounding methods in NeSy AI. We contend that the selection of a suitable grounding criterion is as pivotal as the design of the NeSy method itself. To support this claim, we propose a parameterized generalization of the classic Backward Chaining algorithm, which leverages the well-studied grounding approaches employed in logic inference, but it can be relaxed to take advantage of the statistical nature of NeSy approaches. This allows the control of the trade-off between scalability and preservation of the logical context. Experimentally, we show the efficacy of the proposed grounding methodologies across various NeSy methods on the link prediction task in Knowledge Graphs (KGs).

Contributions. The main contributions of this paper are: (i) we formalize a variety of parametrized grounding methods for NeSy models. The parametrization controls the trade-off between scalability, ensuring the applicability of NeSy methods to large datasets, and expressiveness, hence allowing the model to take full advantage of the knowledge; (ii) we show that our results are general and not specific to a single NeSy method, and we provide an extensive experimental evaluation to support this claim, (iii) we redefine a large class of popular NeSy methods within a common message passing schema, allowing us to share a common implementation and reuse the same grounding and scaling techniques.

2 Preliminaries

First-Order Logic (FOL). A function-free FOL language [Barwise, 1977] is defined by a set of constants (entities) $\mathcal{C}$, variables $\mathcal{V}$ and predicates (relations) $\mathcal{R}$. An atomic formula, or atom, is $r(a_1, ..., a_k)$ with $a_1, ..., a_k \in \mathcal{C} \cup \mathcal{V}$, $r \in \mathcal{R}$ and $k = arity(r)$. A literal is an atom or its negation, and each compound FOL formula α is obtained by recursively combining atomic formulas with connectives ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$) and quantifiers ($\forall$ and $\exists$). Examples of FOL

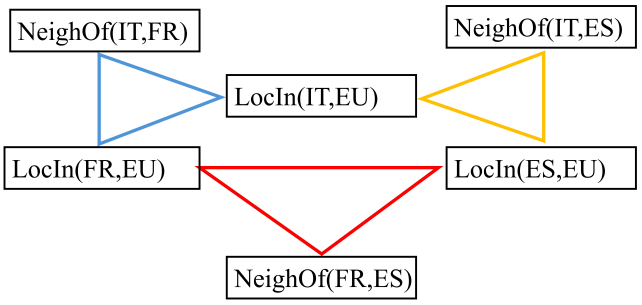

Figure 1: Portion of a GMN. We used the same color for atoms occurring in the same ground formula. The FOL theory includes the constants Italy (IT), France (FR), Spain (ES) and Europe (EU), the predicates LocatedIn (LocIn) and NeighbourOf (NeighOf), and the rule $\forall x \forall y \forall z\ LocIn(x,z) \wedge NeighOf(x,y) \rightarrow LocIn(y,z)$.

formulas are $\forall x Nation(x) \rightarrow \exists y CapitalOf(y,x)$, expressing that "All nations have a capital" or $\forall x \forall y \forall z LocIn(x,y) \wedge LocIn(y,z) \rightarrow LocIn(x,z)$, expressing the transitiveness of being located in a place. Given a formula α, we define *vars*(α) as the set of variables contained in α. For simplicity, in the following sections we will focus on FOL theories, i.e. sets of formulas, containing only specific formulas called Horn clauses, which play a fundamental role in computational logic. A Horn clause is a disjunction of literals with a single positive literal, which is equivalent to formulas in the form $b_1 \wedge \ldots \wedge b_n \rightarrow h$, where the atoms $b_1, \ldots, b_n$ are called *body* atoms and the single atom h is called *head*.

Grounding FOL Theories. Let us fix a FOL language. A substitution θ is an assignment from variables to domain constants, like $\theta = \{x : Italy, y : Europe\}$. A *ground formula* is a FOL formula where all the variables are constants. For example, given $\alpha = LocIn(x,y)$, we can *ground* it with θ and obtain $\theta\alpha = LocIn(Italy, Europe)$. The *Herbrand Base* $HB = \{r(c_1,..c_k) | r \in \mathcal{R}, k = arity(r), (c_1,..c_k) \in \mathcal{C}^k\}$ is the set of all possible ground atoms formed using the predicate and constant symbols. Given a FOL theory Γ, its *Herbrand Universe* is defined as $HU_\Gamma = \{\theta\alpha : \alpha \in \Gamma, \theta \text{ is a substitution}\}$, i.e. HU_Γ is the set of all possible ground formulas that can be formed from Γ. The process of constructing the entire Herbrand Universe is called *(full) grounding*. In general, we refer to *grounding* as the process of constructing a subset of the Herbrand Universe for a given FOL Theory.

Some methods like Markov Logic Networks (MLNs) [Richardson and Domingos, 2006], use a graphical representation of the entire Herbrand Universe (or a subset) called a *Grounded Markov Network* (GMN). In the GMN, each node represents a ground atom, and there is an edge between two nodes only if the corresponding ground atoms appear together in at least one grounding of a formula (Figure 1).

Knowledge Graph Embeddings. A Knowledge Graph (KG) represents knowledge as a graph of entities (nodes) and relations (edges), with facts as (subject, relation, object) triples. Knowledge Graph Embeddings (KGEs) [Wang *et al.*, 2017] map KG entities and relations to low-dimensional vectors, encoding semantic meaning and statistical dependencies. KGEs enable effective reasoning even with incomplete/noisy graphs, for tasks like link prediction and query answering. Prominent examples include DistMult [Yang *et al.*, 2015] and ComplEx [Trouillon *et al.*, 2016].

3 Neural-Symbolic Models

Neural networks, while effective at feature processing, are limited by opaque decision-making, large data requirements, and poor generalization. Symbolic AI, though interpretable, struggles with real-world complexity. Neuro-symbolic (NeSy) models combine their strengths. This paper focuses on NeSy methods using a neural network to process FOL constants/predicates, using the GMN topology to define the reasoner.

The overall structure of these methods, inspired by [Barbiero *et al.*, 2024], is shown in Figure 2, and it is composed of two high-level components:

1. **Atom Processing Layer** mapping each ground atom into an embedding and computing its initial prediction.
2. **Reasoning Layers** updating the atoms' embedding and score according to the structure imposed by the GMN associated to the logic theory, and computing a final updated version of the predictions for each ground atom.

3.1 Atom Processing Layer

The class of considered neural-symbolic methods assigns a numeric representation and an initial output prediction to all the ground atoms in the HB. This can be formalized as: given a ground atom $\alpha = P(c_1, \ldots, c_n)$ and $Emb(c_i)$ the embedding of $c_i \in \mathcal{C}$, the atom processing layer outputs the embedding $Emb^0(\alpha)$ and the initial output prediction $o^0(\alpha)$ as:

$$\begin{aligned} Emb^0(\alpha) &= Enc_P(Emb(c_1), ..., Emb(c_n)) \\ o^0(\alpha) &= Score(Emb^0(\alpha)) \end{aligned}$$

where Enc_P is a generic encoding function associated with the predicate $P \in \mathcal{R}$, and $Score$ is a generic scoring function. Many different representation learning approaches can be employed to implement and co-train the functions $Enc, Score$. While this paper focuses on KGEs and their scoring functions, the approach can be trivially applied to pattern recognition problems using neural networks.

3.2 Reasoning Layers

The reasoning layers take the outputs of the atom processing layer and provide an updated representation (both as embedding and prediction) for each ground atom in the head of the rules. Finally, it outputs the predictions for all atoms. In the following, we indicate the set of rules of a FOL theory having an atom h as head in the considered FOL grounding as $\mathcal{R}(h)$. The reasoner can be seen as the unfolding network of a message passing process [Gilmer *et al.*, 2017] over the GMN, where messages flow from atoms appearing in the body of a ground formula to the atom in the head of the same ground formula. Below, we provide the message passing equations for some of the methods tested in this paper.

The messages flow from an atom node b in the body of the j-th formula r_j to the atom node h in the head of the same

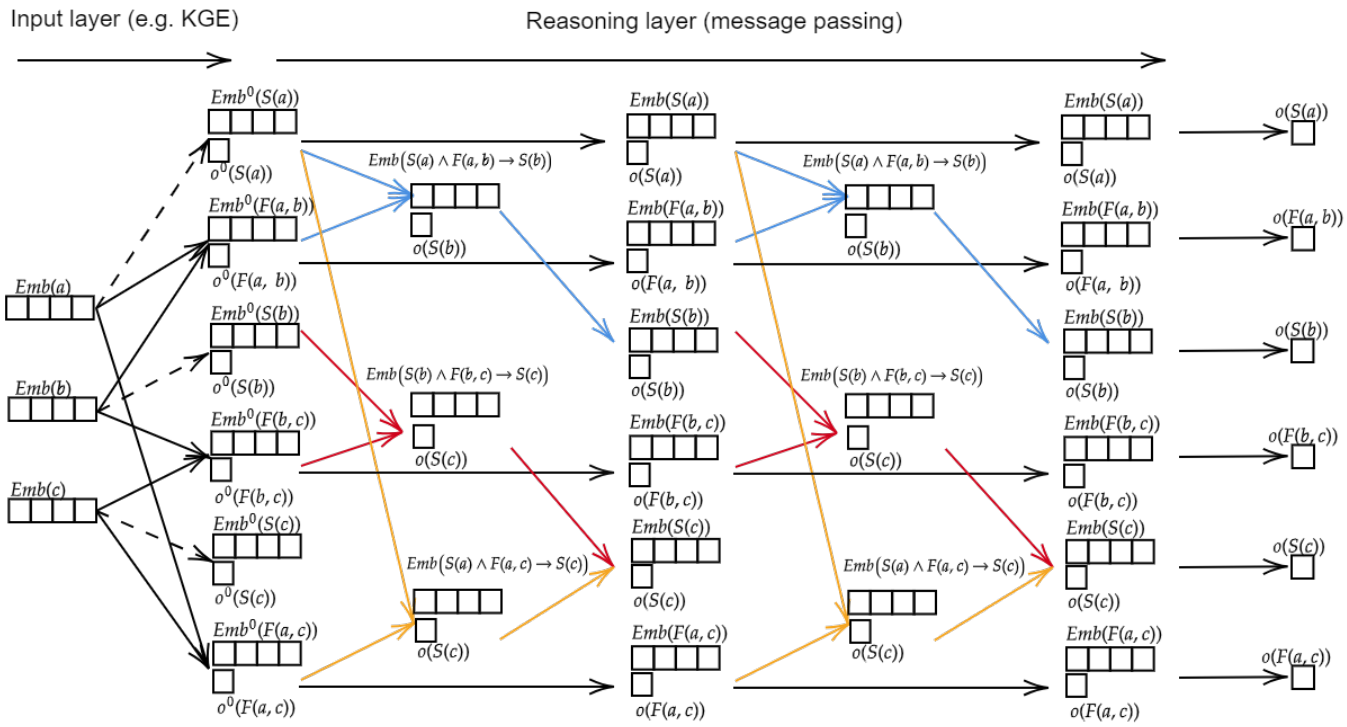

Figure 2: In the considered NeSy models an atom processor takes as input the representation of the constants and compute an embedding and an initial prediction of the atoms. The representations and predictions are refined via message passing, thus performing the reasoning process.

ground formula as:

$$\begin{aligned} Emb_j(h) &= F_j(M_{ne(h,1)\to h}, ..., M_{ne(h,d_j)\to h}) \\ o_j(h) &= G_j(M_{ne(h,1)\to h}, ..., M_{ne(h,d_j)\to h}) \end{aligned}$$

where $M^j_{b\to h}$ is the message defining the different neural-symbolic methods, F_j, G_j are generic functions aggregating back the encoded representation and the output of a node, $ne(h,1), \ldots, ne(h,d_j)$ are the neighbours of node h for the j-th rule, i.e. the set of body atoms associated to atom h for the rule r_j, with $d_j = |ne(h)|$ the number of atoms occurring in the body of the j-th rule r_j.

The embedding of an atom node h is updated by aggregating the representation for all the rules having atom h as head: $Emb(h) = \underset{r_j \in \mathcal{R}(h)}{\mathcal{A}^e} Emb_j(h)$, where $\mathcal{A}^e$ is an aggregation operator like sum, mean or max. Other methods instead aggregate directly over the outputs of a node: $o(h) = \underset{r_j \in \mathcal{R}(h)}{\mathcal{A}^o} o_j(h)$, where $\mathcal{A}^o$ can be any aggregation operator, even if most methods set it as a disjunctive operator, like a max, as shown below.

Relational Reasoning Networks (R2N). R2N [Marra *et al.*, 2025] propagates the embeddings as representations of the atoms, using the following message passing schema:

$$\begin{aligned} M_{b\to h} &= Emb(b) \\ Emb_j(h) &= MLP^e_j(M_{ne(h,1)\to h}, ..., M_{ne(h,d_j)\to h}) \\ Emb(h) &= \sum_{r_j \in \mathcal{R}(h)} Emb_j(h) \\ o(h) &= MLP^o(Emb(h)) \end{aligned}$$

where the embeddings are initialized using Emb^0, and MLP^e_j, MLP^o are multi-layer perceptrons (MLP) processing the embeddings and computing the outputs, respectively.

Logic Tensor Networks (LTN). LTN [Badreddine *et al.*, 2022] and Semantic Based Regularization (SBR) [Diligenti *et al.*, 2017] aggregate the predictions of the body atoms of the j-th rule using a selected t-norm and enforce the consistency of the predictions with the logic rule. These methods were limited to unary predicates in the original papers, but here we propose a relational extension based on message passing:

$$\begin{aligned} M_{b\to h} &= o(b) \\ o_j(h) &= t\text{-}norm(M_{ne(h,1)\to h}, \ldots, M_{ne(h,d_j)\to h}) \\ o(h) &= \max_{r_j \in \mathcal{R}(h)} o_j(h) \end{aligned}$$

where $o(\alpha)$ is initialized using $o^0(\alpha)$ and the original formulation was running a single propagation step.

Deep Concept Reasoners (DCR). DCR [Barbiero *et al.*, 2023] builds a formula from a set of candidate atoms before computing the output of a formula using a t-norm:

$$\begin{aligned} M_{b\to h} &= \Psi_j(Emb^0(b), \Phi_j(Emb^0(b), o(b))) \\ o_j(h) &= t\text{-}norm\left(M_{ne(h,1)\to h}, \ldots, M_{ne(h,d_j)\to h}\right) \\ o(h) &= \max_{r_j \in \mathcal{R}(h)} o_j(h) \end{aligned}$$

where $o_j(\alpha)$ is initialized using $o^0_j(\alpha)$, and the functions $\Phi_j : \mathbb{R}^{n+1} \to [0,1]$ and $\Psi_j : \mathbb{R}^{n+1} \to [0,1]$, assuming $Emb^0(\alpha) \in \mathbb{R}^n$, process the embedded representation of

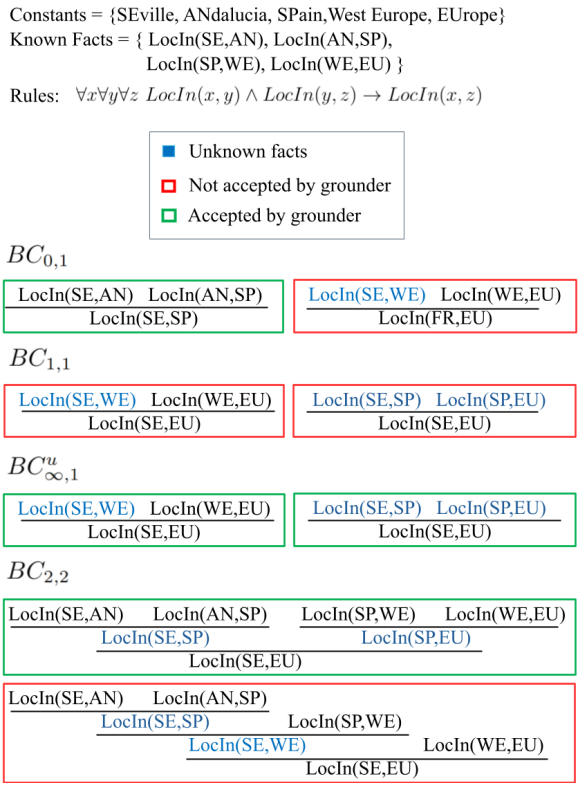

Figure 3: Illustration of some accepted/rejected proofs for different queries with respect to w, d ($BC_{w,d}$) parameters.

each atom in each rule j, to compute its polarity and relevance values, respectively, to get a learned Horn Clause. The process can possibly be iterated, even if the original formulation is defined for a single step of propagation.

4 Grounding Methods

Logical reasoning is often used as a form of theorem proving [Loveland, 2016] in computational logic. An example is query answering in Prolog, where the answer to a query is computed by searching for supporting facts given the set of rules of the program. Hence, proving can be cast to a search over the GMN, looking for subtrees (i.e., proofs) that connect the query to a set of supporting known facts. The known facts in a logic theory are typically much smaller than the HB. Classic Backward Chaining in Prolog considers unknown facts as possibly provable, and recursively builds a sub-proof for them if they are the head of a logic rule. While this process is effective in finding a formal proof for a query given the available known facts, it can be very time-consuming. A commonly used strategy consists in fixing a maximum depth for the backward search, thus admitting to find only proofs within a maximum number of reasoning steps.

In NeSy, the logic facts can be neither true nor false, as they are assigned a score (e.g. a probability of being true) by the corresponding input layer. Therefore, it would be limiting to only consider known true facts in the proving process. Many other facts can provide uncertain information (i.e. context) to predict a query and should be considered. Unfortunately, the size of the portion of the ground network relevant to the query may grow exponentially. However, such a limitation also introduces a new opportunity. Since NeSy models can score any fact in the knowledge base, one does not necessarily need to search the supporting known facts, as the scored facts at shallow depths in a proof can provide enough context for the learning to happen at a statistical level. Nevertheless, the majority of existing NeSy models completely neglect such opportunity, requiring very deep proving processes and resulting in intractable methods for inference.

4.1 Parameterized Backward Chaining Grounder

The discussed limitations of classical Backward Chaining for NeSy motivate a novel definition of a flexible parameterized class of Backward Chaining grounders, which may take advantage of the generalization capabilities of NeSy methods over partially proved trees. A grounder in this parametrized class is defined as $BC_{w,d}$, where $w, d \in \mathbb{N}$ are referred to as the **width** and **depth** of the grounder, respectively. The width w determines the maximum number of atoms in the body of each ground rule that is allowed to not be in the training data and, in turn, to be proved as sub-goals. The depth d controls the maximum depth that can be achieved by any proof for any query passing through its sub-goals. If the number of unknown ground atoms is bigger than w in any ground rule, the proof is immediately discarded even if more depth is allowed. Moreover, once the proof has been built by the grounder, we allow two options: i) to discard the proof if it contains unknown atoms (as done by Backward Chaining), ii) to accept the proof by using the scores of the unknown ground atom. We denote these grounders allowing uncertain atoms as $BC^{u}_{w,d}$. For simplicity, if not stated differently, we will assume option i). When using option i), $BC_{0,1} \equiv BC_{1,1}$, so we will use $BC_{0,1}$ in the rest of the paper.

To illustrate the effects of depths and widths, Figure 3 compares the acceptance of diverse proofs across different parameter values. Section 5 discusses the experimental results for different versions of $BC_{w,d}$. Here below, we show how $BC_{w,d}$ subsumes as special cases grounding techniques commonly used in the literature.

Backward Chaining Grounder $(w = \infty, d = \infty)$**.** Backward Chaining is an inference strategy used in logic programming and automated theorem provers to efficiently prove unknown facts. Given a query to prove, it searches for a rule, whose head atom matches the query. If some atom in the body of the rule is not in the training data, it becomes a new sub-goal that needs to be proven. The process is repeated recursively, until the query is proved with an associated proof tree. Notice that, for $w = d = \infty$ we mean $w = B$, being B the maximum amount of atoms in the body of all the rules. This is the grounding strategy employed by DeepProbLog [Manhaeve *et al.*, 2018], but its application is unfeasible at the scale of the learning tasks considered in this paper.

Backward Chaining Grounder with limited depth $(w = \infty, d = n)$**.** In practical settings, the growth of the admissible proofs is limited, as often done in logic programming. This can be accomplished by setting $d = n$, where $n > 0$ determines the maximum depth of the proofs.

Known Body Grounder $(w = 0, d = 1)$**.** Several popular NeSy models, like e.g. ExpressGNN [Zhang *et al.*, 2020] and RNNLOGIC [Qu *et al.*, 2020], accept a grounding only if all body atoms are known facts, and recursion is limited to depth 1. Therefore, these approaches are equivalent to $BC_{0,1}$.

Full Grounder $(w = \infty, d = 1)$**.** Full grounding of a FOL theory Γ involves generating the HU_Γ, and it is used by models like LTN, SBR or MLNs. This grounder assumes option ii) for proofs, i.e. proofs are accepted even with unknown ground atoms, and thus it is denoted as $BC^u_{\infty,1}$. Roughly, the Full Grounder considers each atom in the HB as a possible query. Hence, all possible ground atoms are represented in the graph, often including irrelevant information, but at the same time ensuring that all the required information needed to perform a correct inference could be recovered.

Complexity Analysis. The performances of $BC_{w,d}$ for NeSy applications are bound by two conflicting effects. On one hand, a grounder should be able to find the proofs needed to prove the queries. In this regard, increasing the width and the depth will result in extending the number of admissible proofs, thus increasing the number of provable queries.

Proposition 1. *The set of facts that are provable via $BC_{w,d}$ is a subset of the provable facts via $BC_{w,d+1}$ and of the provable facts via $BC_{w+1,d}$.*

Proof. The proof trivially follows by observing that the set of buildable proofs in $BC_{w,d}$ is monotone w.r.t. w and d. Indeed, any proof with maximum width w (or depth d) is also a proof with maximum width $w + 1$ (or depth $d + 1$). □

However, a grounder should not instantiate more nodes than needed, as this may affect the generalization capabilities, as stated by the following theorem.

Theorem 1. *[Scarselli* et al.*, 2018] The Vapnik–Chervonenkis dimension (VCD) of the function ϕ computed by a GNN with sigmoidal outputs on the domain of graphs with up to N nodes satisfies: $VCD(\phi) = O(p^2 \cdot N^2)$, where p is the number of parameters of the GNN.*

The Vapnik–Chervonenkis theory states that the generalization error grows as the square root of the VCD, therefore the error bound grows linearly with the graph size. The number of nodes of the underlying reasoning graph built by $BC_{w,d}$ increases as $O(G^d w^{d-1} m)$, where G is the number of groundings with at most w atoms not in the known KG, and m is the maximum number of body atoms within any rule. For instance, if the rule with the longest body is a chain in the form of $p(x_1, x_2) \wedge \ldots \wedge p(x_{n-1}, x_n) \rightarrow p(x_1, x_n)$ with n representing the number of variables, the number of groundings grows as: $G \propto C^{min(w,n-2)}$ for a single reasoning step, where C is the number of constants, and the overall ground network size grows as $O(C^{min(w,n-2)\cdot d} w^{d-1} m)$. Theorem 1 shows that the generalization error is bound to grow linearly with the size of the grounded network. Higher values for w or d lead to an increase in the size of the GMN, therefore it is fundamental for the success of the NeSy approach to select values for w, d such that the logical context is preserved, while the generalization capabilities of the GNN are not hindered.

Dataset	#Entities	#Relations	#Facts	#Degree	#Rules
Countries S1	272	3	1,110	4.28	1
Countries S2	272	4	1,062	4.35	2
Countries S3	272	4	978	4.35	3
Kinship	104	25	28,356	272.7	48
WN18RR	40,943	18	93,003	2.3	17
FB15k-237	14505	237	310079	21.4	500

Table 1: Basic statistics for the employed datasets.

5 Experiments

Several experiments have been performed on KG link prediction to compare the effectiveness of the grounders.[1]

Datasets. The Countries [Bouchard *et al.*, 2015], Kinship [Kok and Domingos, 2007], WN18RR [Dettmers *et al.*, 2018] and FB15k-237 [Dou *et al.*, 2021] datasets are used to capture diverse sizes and complexities. Countries is split into three tasks S{1,2,3} of increasing complexity, predicting country location based on regions and neighborhoods. Kinship represents familial relationships. Dataset statistics are in Section 5.

Rules. The rules associated with the Countries dataset are $R_1 = LocIn(x, w) \wedge LocIn(w, z) \rightarrow LocIn(x, z)$, $R_2 = NeighOf(x, y) \wedge LocIn(y, z) \rightarrow LocIn(x, z)$ and $R_3 = NeighOf(x, y) \wedge NeighOf(y, k) \wedge LocIn(k, z) \rightarrow LocIn(x, z)$, where $LocIn$ refers to the predicate LocatedIn and $NeighOf$ refers to NeighbourOf. The task S_1 takes R_1, S_2 takes R_1, R_2, and $S3$ all of them. The Countries_abl dataset is based only on R_2. Rules for the remaining datasets were extracted using AMIE [Galárraga et al., 2013], selecting the top rules based on confidence. The number of selected rules per dataset is shown in Section 5.

Hyperparameters. The KGE uses a fixed embedding size of 100, with overall 1000 head and tail corruptions for WN18RR. Adam optimizer (10^{-2} learning rate) and binary cross-entropy loss were used over 100 epochs.

Models. ComplEx is used as the KGE baseline and as the input layer for NeSy methods to enable a reasoning ablation study. The KGE baseline is compared with NeSy models of varying characteristics: R2N, which is highly expressive but less interpretable as it learns rules over groups of atoms, excels in complex, loosely formalized reasoning but risks overfitting when flexibility is unnecessary. SBR, less expressive but highly interpretable, performs best in simpler, well-formalized tasks as it strictly applies logic rules. DCR serves as a middle ground between these two approaches.

Metrics. For the evaluation, we use Mean Reciprocal Rank (MRR) and Hits@N metrics. All metric evaluations have been averaged over five runs for the countries dataset.

5.1 Experimental Results

The first experiment serves as a proof of concept using the Countries_abl dataset. Three splits (AS_1, AS_2, AS_3) are generated by ablating facts from the original dataset. The splits are designed with queries requiring one, two, and three reasoning steps, respectively, based on consecutive applications

[1] https://github.com/rodrigo-castellano/Grounding-Methods

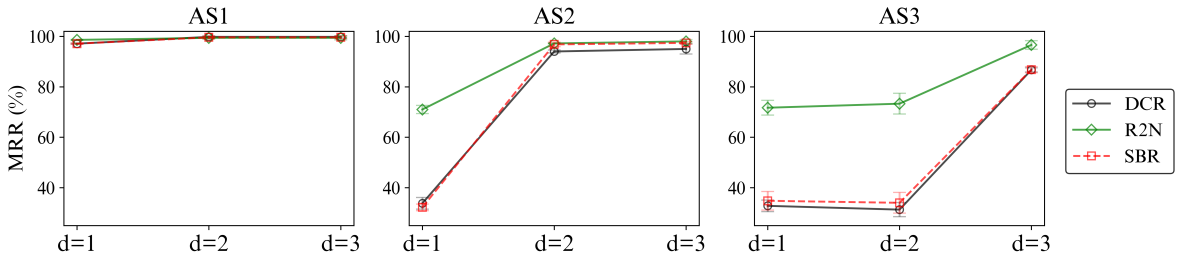

Figure 4: MRR results for different methods using $BC_{1,d}$ with different depths for the Countries_abl datasets AS_1, AS_2, AS_3.

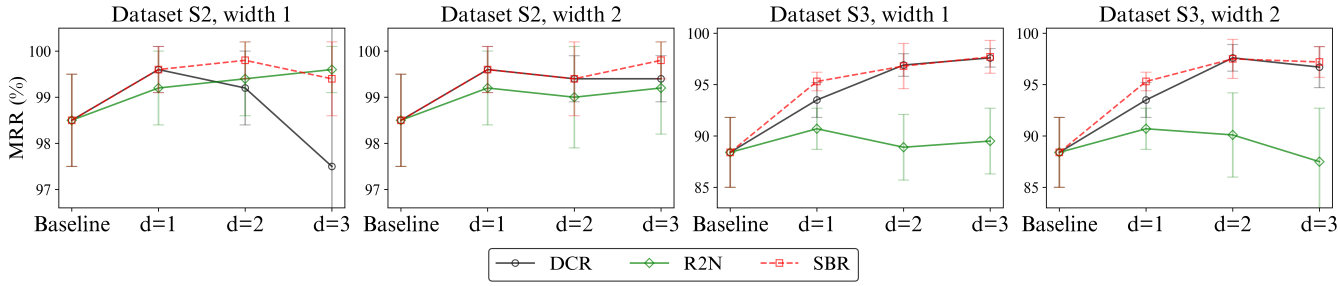

Figure 5: Results for the studied NeSy models and $BC_{w,d}$ grounders for Countries S2,S3 (S1 omitted as it is perfectly solved by all methods).

of the rule $LocIn(x, w) \wedge LocIn(w, z) \rightarrow LocIn(x, z)$. The $BC_{0,1}$ grounder is sufficient to solve the simple one-hop reasoning task defined by AS_1, whereas $BC_{1,2}$ or $BC_{1,3}$ are required to provide good MRR performances in AS_2, as shown in Figure 4. Finally, $BC_{1,3}$ is the only grounder capable of solving AS_3 with high MRR with all methods. This is consistent with our expectations, as the grounding depth defines the number of hops preserved in the context of a query.

The results for the Country dataset are presented in Figure 5, where the baseline is placed as $d = 0$ (this is consistent with the model definitions as the baseline is the model output if no message passing of the reasoner is performed on top of the input atom processing layer). The task S1 is quite simple so that all methods including the baseline are able to fully solve it with $MRR = 1$, therefore it is omitted in the plots. For S2, all NeSy methods outperform the baseline, with depth 1 ($BC_{w,1}$) sufficient for strong performance, and higher depths plateauing. Regarding the more complex reasoning required by the S3 task, we observed that R2N tends to overfit for this dataset in the tested configuration. All other NeSy methods provide a large improvement over the baseline. Optimal results are achieved at either depth 2 or 3, which is indeed the number of reasoning hops required to solve the task. Figure 6 compares the NeSy methods against the baseline when using the Full grounder for the S1,S2,S3 tasks. The performances of the Full grounder are, in general, not improving over the $BC_{w,d}$ grounders in Figure 5, with actually smaller average gains over the baseline. The Full grounder tends to ground many unnecessary atoms with respect to the focused $BC_{w,d}$ grounders, and this negatively affects model generalization (as stated by Theorem 1). The results for the WN18RR and Kinship are presented in Table 2, where we report only the grounders that could be applied in less than 10h

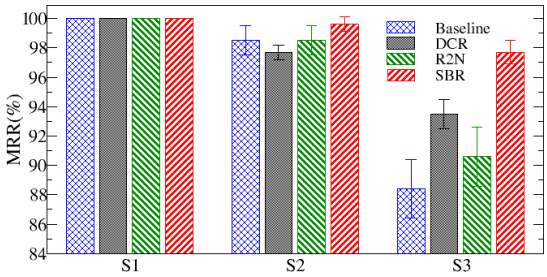

Figure 6: MRR results for the Country datasets and the Full grounder using different NeSy methods and the baseline.

of computation on our workstation (12 core i7 CPU, 64GB RAM, OS Linux). For example, the application of the Full grounder is unfeasible for WN18RR or Kinship.

MRR gains over the baseline of approximately 9%, and 3% for Kinship, and WN18RR, respectively. As shown by the inference times in Table 2, increasing the depth of the grounder in the larger datasets strongly impacts the required inference times. In practice, any grounder with a depth $d > 2$ would be very impractical to use for large KGs. Therefore we limited the experiments to $BC_{0,1}$ and $BC_{1,2}$. We observed generally improved results are obtained as the depth increases from $d = 1$ to $d = 2$ to retain enough information to improve the classification. However, the statistical nature of NeSy tasks makes it possible to get competitive performances even when limited to $BC_{0,1}$. For instance, in the Kinship dataset DCR works very well already at depth 1, even if R2N and SBR can provide slightly improved results at depth 2. In the WN18RR dataset, $BC_{0,1}$ already provides significant gains over the baseline, even if $BC_{1,2}$ provides the best results both for DCR and SBR. These differences are also found in larger

Model	Grounder	MRR	Hits@1	Hits@3	Hits@10	Train Time(s)	Test Time(s)
Kinship							
DCR	$BC_{0,1}$	90.1	84.1	**95.9**	97.0	16479.6	7658.7
	$BC_{1,2}$	90.1	84.1	**95.9**	97.0	16295.2	7517.4
R2N	$BC_{0,1}$	**94.0**	**92.1**	95.6	96.5	9573.3	6616.2
	$BC_{1,2}$	91.8	87.1	96.4	97.4	48808.8	28249.2
SBR	$BC_{0,1}$	86.9	78.0	95.6	97.1	9066.8	6208.7
	$BC_{1,2}$	87.7	79.1	96.0	**97.3**	43354.7	27447.9
ComplEx	-	85.9	79.2	92.2	94.5	773.3	284.9
WN18RR							
DCR	$BC_{0,1}$	44.2	42.2	44.8	47.6	26133.3	2337.9
	$BC_{1,2}$	**45.6**	42.9	**47.1**	**50.2**	74626.7	6944.2
R2N	$BC_{0,1}$	44.2	42.3	44.6	47.3	20613.7	2183.4
	$BC_{1,2}$	44.1	41.4	45.4	48.1	72213.3	7353.4
SBR	$BC_{0,1}$	44.0	42.3	44.2	46.6	21940.5	1909.8
	$BC_{1,2}$	44.7	**42.5**	45.2	48.2	67851.6	6666.0
ComplEx	-	42.7	40.8	42.9	45.9	1079.2	138.7

Table 2: Results for different NeSy methods and grounders for the Kinship and WN18RR datasets compared against the baseline.

	Grounder	MRR	Hits@1	Hits@3	Hits@10
ComplEx	-	25.9	16.3	29.2	45.2
RNNLogic(emb.)	$BC_{0,1}$	34.4	25.2	38.0	53.0
R2N	$BC_{0,1}$	**34.7**	**25.4**	**38.2**	**53.1**

Table 3: MRR, Hits@N metrics on the FB15k-237 dataset.

datasets. For WN18RR, moving from $BC_{0,1}$ to $BC_{1,2}$ with the DCR model improved the MRR metric by 1.6 points, but increased training and inference times by a 3x factor. This observation underscores the critical importance of efficient and effective grounding techniques for scaling NeSy methods to larger real-world KGs. Results for the larger FB15k-237 dataset are provided in Section 5.1, where the computational constraints limited experiments to $BC_{0,1}$, which still showed substantial improvements over the baseline for R2N. We did not add SBR and DCR to the table as the available rules were not covering enough atoms to provide a fair comparison.

6 Related Work

Full Groundings Approaches. A large class of NeSy models was defined over a full grounding approach, this includes SBR [Diligenti *et al.*, 2017], LTN [Badreddine *et al.*, 2022], and Relational Neural Machines [Marra *et al.*, 2020]. The main problem with this grounding method is its lack of scalability, as shown in the experimental section.

Theorem proving approaches. Automatically answering a query (i.e. proving a theorem) given a set of definite clauses is at the core of logic programming [Clocksin and Mellish, 2012]. When proving a fact, only a portion of the GMN is relevant, thus a full grounding is usually not necessary. Standard techniques to avoid the full grounding include *Backward Chaining* [Kapoor and Bahl, 2016] and *Forward Chaining* [Al-Ajlan, 2015] or a mix of the two [Mumick and Pirahesh, 1994]. Since known facts are in general sparse over the HB, the relevant portion of the Markov network is much smaller than the full GMN obtained by grounding the rules over the entire domain. A large class of NeSy systems is based on theorem proving, this class includes systems exploiting a Prolog engine like DeepProbLog [Manhaeve *et al.*, 2018], DeepStochLog [Winters *et al.*, 2022], an ASP engine [Shakarian *et al.*, 2023] or Datalog (e.g. LRNN [Sourek *et al.*, 2018]). However, the exact proving approach does not allow the methods to scale beyond small inference problems. Therefore, some methods try to further limit the graph size. For example, DPLA* [Manhaeve *et al.*, 2021] uses an A* approach to select the most promising portions of the GMN.

Heuristics-based approaches. The most common approach used in the literature is to rely on some ad-hoc heuristics to make grounding tractable. However, it did not emerge in the literature a clear understanding of the trade-off between performance and computing time. *RNNLogic* [Qu *et al.*, 2020] generates logic rules and then scores the query triplets using the grounding paths in the training knowledge. This makes the model account for the grounded rules where all body atoms are true in the training facts, which is the special case where Backward Chaining is limited to depth 1. *Discriminative Gaifman Models* [Niepert, 2016] considers local neighbourhoods of a KG as features to predict queries. The local neighbourhoods can be seen as a generalization of a Horn clause, representing a chain of facts in the knowledge that imply the query. Similarly to RNN Logic, this considers only the grounded rules where all body atoms are true in the training facts. *ExpressGNN* [Zhang *et al.*, 2020] operates directly on the knowledge graph rather than the extensive GMN grounding graph. This is similar to the approach of *UniKER* [Cheng *et al.*, 2021], which focuses on instantiating rules only with the training triples in the knowledge graph but iteratively enriching the training data with new facts derived using forward chaining. *KALE* [Guo *et al.*, 2016] includes groundings where at least one atom is true in the training data, and uses a full grounding approach for the other variables. *RUGE* [Guo *et al.*, 2018] considers a grounding when the triples in the rule's body are observed in the training data, and the head triple is not.

7 Conclusions and Future Work

This paper studies the effect of different grounding schemas on the predictive accuracy of NeSy methods. The main contributions of the work are in the definition of new parametrized grounding approaches, tailored to NeSy methods, and in the extensive experimental comparison of these grounders, offering insights into the trade-offs between grounding depth, model performance, and computational efficiency. The results on link prediction on KGs show that increasing the depth of the parameterized grounder enhances performance in complex reasoning tasks, but the statistical nature of NeSy allows preserving a good prediction accuracy even using shallow and efficient grounders. This contribution is not only methodological but also practical, highlighting the critical role of the grounding schema in bridging the gap between symbolic reasoning and neural learning.

This study is limited to static grounders without any feedback from the NeSy method, which is the most used approach. In future work, we plan to explore dynamic grounders, which iteratively exploit the predictions of the learner to dynamically incorporate new atoms into the set of known facts. This could lead to the definition of parametric grounders which can be trained end-to-end with the learner.

Ethical Statement

There are no ethical issues.

Acknowledgments

This work has been partially supported by the project PNRR M4C2 "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI" , code PE0000013, CUP E53C22001610006. This work was also supported by the EU Framework Program for Research and Innovation Horizon under the Grant Agreement No 101073307 (MSCA-DN LeMuR). This work has been partially supported by the project "CONSTR: a COllectionless-based Neuro-Symbolic Theory for learning and Reasoning", PARTENARIATO ESTESO "Future Artificial Intelligence Research - FAIR", SPOKE 1 "Human-Centered AI" Università di Pisa, "NextGenerationEU", CUP I53C22001380006.

References

[Al-Ajlan, 2015] Ajlan Al-Ajlan. The comparison between forward and backward chaining. *International Journal of Machine Learning and Computing*, 5(2):106, 2015.

[Badreddine *et al.*, 2022] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

[Barbiero *et al.*, 2023] Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Mateo Espinosa Zarlenga, Lucie Charlotte Magister, Alberto Tonda, Pietro Lió, Frederic Precioso, Mateja Jamnik, and Giuseppe Marra. Interpretable neural-symbolic concept reasoning. In *ICML*, pages 1801–1825, 2023.

[Barbiero *et al.*, 2024] Pietro Barbiero, Francesco Giannini, Gabriele Ciravegna, Michelangelo Diligenti, and Giuseppe Marra. Relational concept bottleneck models. *Advances in Neural Information Processing Systems*, 37:77663–77685, 2024.

[Barwise, 1977] Jon Barwise. An introduction to first-order logic. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 5–46. Elsevier, 1977.

[Bouchard *et al.*, 2015] Guillaume Bouchard, S. Singh, and Theo Trouillon. On approximate reasoning capabilities of low-rank vector spaces. In *AAAI Spring Symposia*, 2015.

[Cheng *et al.*, 2021] Kewei Cheng, Ziqing Yang, Ming Zhang, and Yizhou Sun. Uniker: A unified framework for combining embedding and definite horn rule reasoning for knowledge graph inference. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9753–9771, 2021.

[Clocksin and Mellish, 2012] William F Clocksin and Christopher S Mellish. *Programming in Prolog: Using the ISO standard*. Springer Science & Business Media, 2012.

[Dettmers *et al.*, 2018] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference*, 2018.

[Diligenti *et al.*, 2017] Michelangelo Diligenti, Marco Gori, and Claudio Sacca. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017.

[Dou *et al.*, 2021] Jiaheng Dou, Bing Tian, Yong Zhang, and Chunxiao Xing. A novel embedding model for knowledge graph completion based on multi-task learning. In *Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part I 26*, pages 240–255. Springer, 2021.

[d'Avila Garcez *et al.*, 2009] Artur S d'Avila Garcez, Luís C Lamb, and Dov M Gabbay. *Neural-symbolic learning systems*. Springer, 2009.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ProceedingsICML*. PMLR, 2017.

[Guo *et al.*, 2016] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 192–202, 2016.

[Guo *et al.*, 2018] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Kapoor and Bahl, 2016] Namarta Kapoor and Nischay Bahl. Comparative study of forward and backward chaining in artificial intelligence. *International journal of engineering and computer science*, 5(4):16239–16242, 2016.

[Kok and Domingos, 2007] Stanley Kok and Pedro Domingos. Statistical predicate invention. In *ICML*, 2007.

[Loveland, 2016] Donald W Loveland. *Automated theorem proving: A logical basis*. Elsevier, 2016.

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

[Manhaeve *et al.*, 2021] Robin Manhaeve, Giuseppe Marra, and Luc De Raedt. Approximate inference for neural probabilistic logic programming. In *KR*, pages 475–486, 2021.

[Marra *et al.*, 2020] Giuseppe Marra, Michelangelo Diligenti, Francesco Giannini, Marco Gori, and Marco Maggini. Relational neural machines. In *ECAI*, pages 1340–1347. IOS Press, 2020.

[Marra *et al.*, 2024] Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. From statistical relational to neurosymbolic artificial intelligence. *Artificial Intelligence*, page 104062, 2024.

[Marra *et al.*, 2025] Giuseppe Marra, Michelangelo Diligenti, and Francesco Giannini. Relational reasoning networks. *Knowledge-Based Systems*, page 112822, 2025.

[Mumick and Pirahesh, 1994] Inderpal Singh Mumick and Hamid Pirahesh. Implementation of magic-sets in a relational database system. *ACM SIGMOD Record*, 23(2):103–114, 1994.

[Niepert, 2016] Mathias Niepert. Discriminative gaifman models. In *NeurIPS*, 2016.

[Qu *et al.*, 2020] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. *arXiv preprint arXiv:2010.04029*, 2020.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1):107–136, 2006.

[Scarselli *et al.*, 2018] Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. The vapnik–chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248–259, 2018.

[Shakarian *et al.*, 2023] Paulo Shakarian, Chitta Baral, Gerardo I Simari, Bowen Xi, and Lahari Pokala. Neurasp. In *Neuro Symbolic Reasoning and Learning*, pages 63–74. Springer, 2023.

[Sourek *et al.*, 2018] Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezny, Steven Schockaert, and Ondrej Kuzelka. Lifted relational neural networks: Efficient learning of latent relational structures. *Journal of Artificial Intelligence Research*, 62:69–100, 2018.

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080, 2016.

[Wang *et al.*, 2017] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE TKDE*, 29(12):2724–2743, 2017.

[Winters *et al.*, 2022] Thomas Winters, Giuseppe Marra, Robin Manhaeve, and Luc De Raedt. Deepstochlog: Neural stochastic logic programming. In *Proceedings of the AAAI Conference*, 2022.

[Yang *et al.*, 2015] Bishan Yang, W. Yih, X. He, J. Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.

[Zhang *et al.*, 2020] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Efficient probabilistic logic reasoning with graph neural networks. In *ICLR*, 2020.