

Can Graph Neural Networks Help Logic Reasoning?

Yuyu Zhang*, Xinshi Chen*, Yuan Yang*, Arun Ramamurthy†, Bo Li‡, Yuan Qi◇, Le Song*◇

*Georgia Tech, †Siemens, ‡UIUC, ◇Ant Financial

Abstract

Effectively combining logic reasoning and probabilistic inference has been a long-standing goal of machine learning: the former has the ability to generalize with small training data, while the latter provides a principled framework for dealing with noisy data. However, existing methods for combining the best of both worlds are typically computationally intensive. In this paper, we focus on Markov Logic Networks and explore the use of graph neural networks (GNNs) for representing probabilistic logic inference. It is revealed from our analysis that the representation power of GNN alone is not enough for such a task. We instead propose a more expressive variant, called ExpressGNN, which can perform effective probabilistic logic inference while being able to scale to a large number of entities. We demonstrate by several benchmark datasets that ExpressGNN has the potential to advance probabilistic logic reasoning to the next stage.

1 Introduction

An elegant framework of combining logic reasoning and probabilistic inference is Markov Logic Network (MLN) [1], where logic predicates are treated as random variables and logic formulae are used to define the potential functions. It has greatly extended the ability of logic reasoning to handle noisy facts and partially correct logic formulae common in real-world problems. Furthermore, MLN enables probabilistic graphical models to exploit prior knowledge and learn in the region of small or zero training samples. This second aspect is important in the context of lifelong learning and massive multitask learning, where most prediction targets have insufficient number of labeled data.

However, a central challenge is that probabilistic inference in MLN is computationally intensive. It contains $\mathcal{O}(M^n)$ many random variables if there are M entities and the involved predicate has n arguments. Approximate inference techniques such as MCMC and belief propagation have been proposed, but the large MLN makes them barely scalable to hundreds of entities.

Graph neural network (GNN) is a popular tool of learning representation for graph data, including but not limited to social networks, molecular graphs, and knowledge graphs [2, 3, 4, 5, 6, 7]. It is natural to think that GNNs have the potential to improve the effectiveness of probabilistic logic inference in MLN. However, it is not clear *why* and *how* exactly GNNs may help.

In this paper, we explore the use of GNN for scalable probabilistic logic inference in MLN, and provide an affirmative answer on how to do that. In our method, GNN is applied to knowledge bases which can be orders of magnitude smaller than grounded MLN; and then GNN embeddings are used to define mean field distributions in probabilistic logic inference. However, our analysis reveals that GNN embeddings alone will lead to **inconsistent** parametrization due to the additional asymmetry created by logic Preprint. Under review.

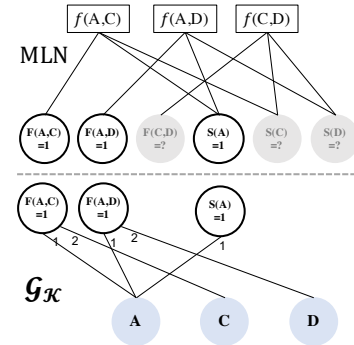


Figure 1: *Bottom:* A knowledge base as a factor graph. $\{A, C, D\}$ are entities, and F (Friend) and S (Smoke) are predicates. *Top:* Markov Logic Network (MLN) with formula $f(c, c') := \neg S(c) \vee \neg F(c, c') \vee S(c')$.

formulae. Motivated by this analysis, we propose a more expressive variant, called ExpressGNN, which consists of (1) an inductive GNN embedding component for learning representation from knowledge bases; (2) and a transductive and tunable embedding component for compensating the asymmetry created by logic formulae in MLN.

We show by experiments that mean field approximation with ExpressGNN enables efficient and effective probabilistic logic inference in modern knowledge bases. Furthermore, ExpressGNN can achieve these results with far fewer parameters than purely transductive embeddings, and yet it has the ability to adapt and generalize to new knowledge graphs.

Related work. Previous probabilistic logic inference techniques either use sampling methods or belief propagation. More advanced variants have been proposed to make use of symmetries in MLNs to reduce computation (e.g., the lifted inference algorithms [8, 9]). However, these inference methods still barely scale to hundreds of entities. We describe specific methods and compare their performances in Section 7. A recent seminal work explored the use of GNN for relation prediction, but the additional challenges from logic formula are not considered [10].

2 Knowledge Bases and Markov Logic Networks

Knowledge base. Typically, a knowledge base \mathcal{K} consists of a tuple $\mathcal{K} = (\mathcal{C}, \mathcal{R}, \mathcal{O})$, with a set $\mathcal{C} = \{c_1, \dots, c_M\}$ of M entities, a set $\mathcal{R} = \{r_1, \dots, r_N\}$ of N relations, and a collection $\mathcal{O} = \{o_1, \dots, o_L\}$ of L observed facts. In the language of first-order logic, entities are also called constants. For instance, a constant can be a person or an object. Relations are also called **predicates**. Each predicate is a logic function defined over \mathcal{C} , i.e., $r(\cdot) : \mathcal{C} \times \dots \times \mathcal{C} \mapsto \{0, 1\}$. In general, the arguments of predicates are asymmetric. For instance, for the predicate $r(c, c') := \text{L}(c, c')$ (L for Like) which checks whether c likes c' , the arguments c and c' are not exchangeable.

With a particular set of entities assigned to the arguments, the predicate is called a grounded predicate, and **each grounded predicate \equiv a binary random variable**, which will be used to define MLN. For a d -ary predicate, there are M^d ways to ground it. We denote an assignment as a_r . For instance, with $a_r = (c, c')$, we can simply write a grounded predicate $r(c, c')$ as $r(a_r)$. Each observed fact in knowledge bases is a truth value $\{0, 1\}$ assigned to a grounded predicate. For instance, a fact o can be $[\text{L}(c, c') = 1]$. The number of observed facts is typically much smaller than that of unobserved facts. We adopt the open-world paradigm and treat these **unobserved facts \equiv latent variables**.

As a more clear representation, we express a knowledge base \mathcal{K} by a bipartite graph $\mathcal{G}_{\mathcal{K}} = (\mathcal{C}, \mathcal{O}, \mathcal{E})$, where nodes on one side of the graph correspond to constants \mathcal{C} and nodes on the other side correspond to observed facts \mathcal{O} , which is called *factor* in this case (Fig. 1). The set of T edges, $\mathcal{E} = \{e_1, \dots, e_T\}$, will connect constants and the observed facts. More specifically, an edge

$e = (c, o, i)$ between node c and o exists, if the grounded predicate associated with o uses c as an argument in its i -th argument position. (See Fig. 1 for an illustration.)

Markov Logic Networks. MLNs use logic formulae to define potential functions in undirected graphical models. A logic formula $f(\cdot) : \mathcal{C} \times \dots \times \mathcal{C} \mapsto \{0, 1\}$ is a binary function defined via the composition of a few predicates. For instance, a logic formula $f(c, c')$ can be

$$\text{Smoke}(c) \wedge \text{Friend}(c, c') \Rightarrow \text{Smoke}(c') \iff \neg \text{Smoke}(c) \vee \neg \text{Friend}(c, c') \vee \text{Smoke}(c'),$$

where \neg is negation and the equivalence is established by De Morgan's law. Similar to predicates, we denote an assignment of constants to the arguments of a formula f as a_f , and the entire collection of consistent assignments of constants as $\mathcal{A}_f = \{a_f^1, a_f^2, \dots\}$. Given these logic representations, MLN can be defined as a joint distribution over all observed facts \mathcal{O} and unobserved facts \mathcal{H} as (Fig. 1)

$$P(\mathcal{O}, \mathcal{H}) := \frac{1}{Z} \exp \left(\sum_{f \in \mathcal{F}} w_f \sum_{a_f \in \mathcal{A}_f} \phi_f(a_f) \right), \quad (1)$$

where Z is a normalizing constant summing over all grounded predicates and $\phi_f(\cdot)$ is the potential function defined by a formula f . One form of $\phi_f(\cdot)$ can simply be the truth value of the logic formula f . For instance, if the formula is $f(c, c') := \neg \text{S}(c) \vee \neg \text{F}(c, c') \vee \text{S}(c')$, then $\phi_f(c, c')$ can simply take value 1 when $f(c, c')$ is true and 0 otherwise. Other more sophisticated ϕ_f can also be designed, which have the potential to take into account complex entities, such as images or texts, but will not be the focus of this paper. The weight w_f can be viewed as the confidence score of formulae f : the higher the weight, the more accurate the formula is.

3 Challenges for Inference in Markov Logic Networks

Inference in Markov Logic Networks can be very computationally intensive, since the inference needs to be carried out in the fully grounded network involving all grounded variables and formula nodes. Most previous inference methods barely scale to hundreds of entities.

Mean field approximation. We will focus on mean field approximation, since it has been demonstrated to scale up to many large graphical models, such as latent Dirichlet allocation for modeling topics from large text corpus [10, 11]. In this case, the conditional distribution $P(\mathcal{H}|\mathcal{O})$ is approximated by a product distribution, $P(\mathcal{H}|\mathcal{O}) \approx \prod_{r(a_r) \in \mathcal{H}} Q^*(r(a_r))$. The set of mean field distributions $Q^*(r(a_r))$ can be determined by KL-divergence minimization

$$\begin{aligned} \{Q^*(r(a_r))\} &= \operatorname{argmin}_{\{Q(r(a_r))\}} \operatorname{KL} \left(\prod_{r(a_r) \in \mathcal{H}} Q(r(a_r)) \parallel P(\mathcal{H}|\mathcal{O}) \right) \\ &= \operatorname{argmin}_{\{Q(r(a_r))\}} \sum_{r(a_r) \in \mathcal{H}} \mathbb{E}[\ln Q(r(a_r))] - \sum_{f \in \mathcal{F}} w_f \sum_{a_f \in \mathcal{A}_f} \mathbb{E}[\phi_f(a_f)|\mathcal{O}], \end{aligned} \quad (2)$$

where $\mathbb{E}[\phi_f(a_f)|\mathcal{O}]$ means that observed predicates in \mathcal{O} are fixed to their actual values with probability 1. For instance, a grounded formula $f(A, B) = \neg S(A) \vee \neg F(A, B) \vee S(B)$ with observations $S(A) = 1$ and $F(A, B) = 1$ will result in $\mathbb{E}[\phi_f(a_f)|\mathcal{O}] = \sum_{S(B) \in \{0,1\}} Q(S(B))(\neg 1 \vee \neg 1 \vee S(B))$. In theory, one can use mean field iteration to obtain the optimal $Q^*(r(a_r))$, but due to the large number of nodes in the grounded network, this iterative algorithm can be very inefficient.

Thus, we need to carefully think about how to parametrize the set of $Q(r(a_r))$, such that the parametrization is expressive enough for representing posterior distributions, while at the same time leading to efficient algorithms. Some common choices are

- **Naive parametrization.** Assign each $Q(r(a_r))$ a parameter $q_{r(a_r)} \in [0, 1]$. Such parametrization is very expressive, but the number of parameters is the same as MLN size $\mathcal{O}(M^n)$.
- **Tunable embedding parametrization.** Assign each entity c a vector embedding $\mu_c \in \mathbb{R}^d$, and define $Q(r(a_r))$ using involved entities. For instance, $Q(r(c, c')) := \operatorname{logistic}(\operatorname{MLP}_r(\mu_c, \mu_{c'}))$ where MLP_r is a neural network specific to predicate r and $\operatorname{logistic}(\cdot)$ is the standard logistic function. The number of parameters in such scheme is linear in the number of entities, $\mathcal{O}(d|\mathcal{C}|)$, but very high dimensional embedding μ_c may be needed to express the posteriors.

Note that both schemes are **transductive**, and the learned $q_{r(a_r)}$ or μ_c can only be used for the training graph, but can not be used for new entities or different but related knowledge graphs (**inductive setting**).

Stochastic inference. The objective in Eq. (2) contains an expensive summation, making its evaluation and optimization inefficient. For instance, for formula $f(c, c') := \neg S(c) \vee \neg F(c, c') \vee S(c')$, the number of terms involved in $\sum_{a_f \in \mathcal{A}_f}$ will be square in the number of entities. Thus, we approximate the objective function with stochastic sampling, and then optimize the parameters in $Q(r(a_r))$ via stochastic gradients, and various strategies can be used to reduce the variance of the stochastic gradients [10, 11].

4 Graph Neural Network for Inference

To efficiently parametrize the entity embeddings with less parameters than tunable embeddings, we propose to use a GNN on the knowledge graph \mathcal{G}_K , much smaller than the fully grounded MLN (Figure 1), to generate embeddings μ_c of each entity c , and then use these embeddings to define mean field distributions. The advantage of GNN is that the number of parameters can be independent of the number of entities. Any entity embedding μ_c can be reproduced by running GNN iterations online. Thus, GNN based parametrization can potentially be very memory efficient, making it possible to scale up to a large number of entities. Furthermore, the learned GNN parameters can be used for both transductive and inductive settings.

Algorithm 1: GNN and Color Refinement

Function $\text{GNN}(\mathcal{G}_K = (\mathcal{C}, \mathcal{O}, \mathcal{E}))$:

Initialize entity node: $\mu_c^{(0)} = \mu_c, \forall c \in \mathcal{C}$

Fact node: $\mu_o^{(0)} = \mu_r, \forall o \equiv [r(a_r) = v]$

▷ nodes of the same type are initialized with a uniform color

For $t = 0$ to $T - 1$ **do**

 Compute message $\forall (c, o, i) \in \mathcal{E}$:

$m_{o \rightarrow c}^{(t+1)} = \operatorname{MLP}_{1,i,v}(\mu_o^{(t)}, \mu_c^{(t)})$

$m_{c \rightarrow o}^{(t+1)} = \operatorname{MLP}_{2,i,v}(\mu_c^{(t)}, \mu_o^{(t)})$

 Aggregate message $\forall c \in \mathcal{C}, o \in \mathcal{O}$:

$m_c^{(t+1)} = \operatorname{AGG}_1(\{m_{o \rightarrow c}^{(t+1)}\}_{o \in \mathcal{N}(c)})$

$m_o^{(t+1)} = \operatorname{AGG}_2(\{m_{c \rightarrow o}^{(t+1)}\}_{c \in \mathcal{N}(o)})$

 ▷ aggregate colors of neighborhoods

 Update embeddings $\forall c \in \mathcal{C}, o \in \mathcal{O}$:

$\mu_c^{(t+1)} = \operatorname{MLP}_3(\mu_c^{(t)}, m_c^{(t+1)})$

$\mu_o^{(t+1)} = \operatorname{MLP}_4(\mu_o^{(t)}, m_o^{(t+1)})$

 ▷ hash colors of nodes and their neighborhoods into unique new colors

return node embeddings $\{\mu_c^{(T)}\}$ and $\{\mu_o^{(T)}\}$

The architecture of GNN over a knowledge graph \mathcal{G}_K is given in Algorithm 1, where the multilayer neural networks $\text{MLP}_{1,i,v}$ and $\text{MLP}_{2,i,v}$ take values v of observed facts and argument positions i into account, MLP_3 and MLP_4 are standard multilayer neural networks, and AGG_1 and AGG_2 are typically sum pooling functions. These embedding updates are carried out for a finite T times. In general, the more iterations are carried out, the larger the graph neighborhood around a node will be integrated into the representation. For simplicity of notation, we use $\{\mu_c\}$ and $\{\mu_o\}$ to refer to the final embeddings $\{\mu_c^{(T)}\}$ and $\{\mu_o^{(T)}\}$. Then these embeddings are used to define the mean field distributions. For instance, if a predicate $r(c, c')$ involves two entities c and c' , $Q(r(c, c'))$ can be defined as

$$Q(r(c, c')) = \text{logistic}(\text{MLP}_r(\mu_c, \mu_{c'})), \text{ where } \{\mu_c, \mu_{c'}\} = \text{GNN}(\mathcal{G}_K) \quad (3)$$

and MLP_r are predicate specific multilayer neural networks. For d dimensional embeddings, the number of parameters in GNN model is typically $O(d^2)$, independent of the number of entities.

5 Is GNN Expressive Enough?

Now a central question is: is GNN parametrization using the knowledge graph \mathcal{G}_K expressive enough? Or will there be situations where two random variables $r(c, c')$ and $r(c', c')$ have **different** distributions in MLN but $Q(r(c, c'))$ and $Q(r(c', c'))$ are forced to be **the same** due to the above characteristic of GNN embeddings? The question arises since the knowledge graph \mathcal{G}_K is different from the fully grounded MLN (Figure 1). We will use two theorems (proofs are all given in Appendix A) to analyze whether GNN is expressive enough. Our main results can be summarized as:

- (1) Without formulae tying together the predicates, GNN embeddings are expressive enough for feature representations of latent facts in the knowledge base.
- (2) With formulae in MLN modeling the dependency between predicates, GNN embedding becomes insufficient for posterior parametrization.

These theoretical analyses motivate a more expressive variant, called ExpressiveGNN in Section 6, to compensate for the insufficient representation power of GNN for posterior parametrization.

5.1 Property of GNN

Recent research shows that GNNs can learn to perform approximate graph isomorphism check [4, 12]. In our case, graph isomorphism of knowledge graphs is defined as follows.

Definition 5.1 (Graph Isomorphism). An isomorphism of graphs $\mathcal{G}_K = (\mathcal{C}, \mathcal{O}, \mathcal{E})$ and $\mathcal{G}_{K'} = (\mathcal{C}', \mathcal{O}', \mathcal{E}')$ is a bijection between the nodes $\pi : \mathcal{C} \cup \mathcal{O} \rightarrow \mathcal{C}' \cup \mathcal{O}'$ such that (1) $\forall o \in \mathcal{O}$, $\text{NodeType}(o) = \text{NodeType}(\pi(o))$; (2) c and o are adjacent in \mathcal{G}_K if and only if $\pi(c)$ and $\pi(o)$ are adjacent in $\mathcal{G}_{K'}$ and $\text{EdgeType}(c, o) = \text{EdgeType}(\pi(c), \pi(o))$.

In this definition, NodeType is determined by the associated predicate of a fact, i.e., $\text{NodeType}(o \equiv [r(a_r) = v]) = r$. EdgeType is determined by the observed value and argument position, i.e., $\text{EdgeType}((c, o \equiv [r(a_r) = v], i)) = (v, i)$. Our GNN architecture in Algorithm 1 is adapted to these graph topology specifications. More precisely, the initial node embeddings correspond to NodeTypes and different MLPs are used for different EdgeTypes .

It has been proved that GNNs with injections acting on neighborhood features can be as powerful as 1-dimensional Weisfeiler-Lehman graph isomorphism test [13, 12], also known as **color refinement**. The color refinement procedure is given in Algorithm 1 by red texts, which is analogous to updates of GNN embeddings. We use it to define indistinguishable nodes.

Definition 5.2 (Indistinguishable Nodes). Two nodes c, c' in a graph \mathcal{G} are **indistinguishable** if they have the same color after the color refinement procedure terminates and no further refinement of the node color classes is possible. In this paper, we use the following notation:

$$(c_1, \dots, c_n) \xleftrightarrow{\mathcal{G}} (c'_1, \dots, c'_n) : \text{for each } i \in [n], c_i \text{ and } c'_i \text{ are indistinguishable in } \mathcal{G}.$$

While GNN has the ability to do color refinement, it is *at most* as powerful as color refinement [12].

Therefore, if $c \xleftrightarrow{\mathcal{G}} c'$, their final GNN embeddings will be the same, i.e., $\mu_c = \mu_{c'}$. When we use GNN embeddings to define Q as in Eq. (3), it implies that $Q(r(c, c')) = Q(r(c', c'))$, $\forall c' \in \mathcal{C}$.

5.2 GNN is expressive for feature representations in knowledge bases

GNN embeddings are computed based on knowledge bases $\mathcal{K} = (\mathcal{C}, \mathcal{R}, \mathcal{O})$ involving only observed facts \mathcal{O} , but the resulting entity embeddings $\{\mu_c\}$ will be used to represent a much larger set of unobserved facts \mathcal{H} . In this section, we will show that, when formulae are not considered, these entity embeddings $\{\mu_c\}$ are expressive enough for representing the latent facts \mathcal{H} .

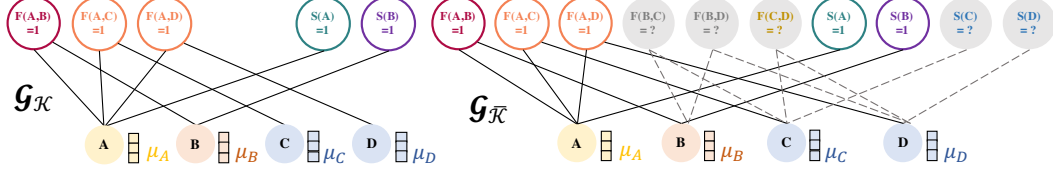


Figure 2: Factor graph $\mathcal{G}_{\mathcal{K}}$ and the corresponding augmented factor graph $\mathcal{G}_{\bar{\mathcal{K}}}$ after running coloring refinement.

To better explain our idea, we define a fully grounded knowledge base as: $\bar{\mathcal{K}} := (\mathcal{C}, \mathcal{R}, \mathcal{O} \cup \mathcal{H})$, where all unobserved facts are included and assigned a value of “?”. Therefore, the facts in $\bar{\mathcal{K}}$ can take one of three different values, i.e., $v \in \{0, 1, ?\}$. Its corresponding **augmented factor graph** is $\mathcal{G}_{\bar{\mathcal{K}}} = (\mathcal{C}, \mathcal{O} \cup \mathcal{H}, \mathcal{E} \cup \mathcal{E}_{\mathcal{H}})$. See Fig. 2 for an illustration of $\mathcal{G}_{\mathcal{K}}$ and $\mathcal{G}_{\bar{\mathcal{K}}}$.

Theorem 5.1. *Let $\mathcal{G}_{\mathcal{K}} = (\mathcal{C}, \mathcal{O}, \mathcal{E})$ be the factor graph for a knowledge base \mathcal{K} and $\mathcal{G}_{\bar{\mathcal{K}}} = (\mathcal{C}, \mathcal{O} \cup \mathcal{H}, \mathcal{E} \cup \mathcal{E}_{\mathcal{H}})$ be the corresponding augmented version. Then the following two statements are true:*

- (1) $c \xleftrightarrow{\mathcal{G}_{\mathcal{K}}} c'$ if and only if $c \xleftrightarrow{\mathcal{G}_{\bar{\mathcal{K}}}} c'$;
- (2) $[r(c_1, \dots, c_n) = v] \xleftrightarrow{\mathcal{G}_{\bar{\mathcal{K}}}} [r(c'_1, \dots, c'_n) = v]$ if and only if $(c_1, \dots, c_n) \xleftrightarrow{\mathcal{G}_{\bar{\mathcal{K}}}} (c'_1, \dots, c'_n)$.

Intuitively, Theorem 5.1 means that without considering the presence of formulae in MLN, unobserved predicates \mathcal{H} can be represented purely based on GNN embeddings obtained from $\mathcal{G}_{\mathcal{K}}$. For instance, to represent an unobserved predicate $r(c_1, \dots, c_n)$ in $\mathcal{G}_{\bar{\mathcal{K}}}$, we only need to compute $\{\mu_c, \mu_o\} = \text{GNN}(\mathcal{G}_{\mathcal{K}})$, and then use $\text{MLP}_r(\mu_{c_1}, \dots, \mu_{c_n})$ as its feature. This feature representation is as expressive as that obtained from $\{\mu_c, \mu_o\} = \text{GNN}(\mathcal{G}_{\bar{\mathcal{K}}})$, and thus drastically reducing the computation.

5.3 GNN is not expressive enough for posterior parametrization

Taking into account the influence of formulae on the posterior distributions of predicates, we can show that GNN embeddings alone become insufficient representations for parameterizing these posteriors. To better explain our analysis in this section, we first extend the definition of graph isomorphism to node isomorphism and then state the theorem.

Definition 5.3 (Isomorphic Nodes). Two ordered sequences of nodes (c_1, \dots, c_n) and (c'_1, \dots, c'_n) are **isomorphic** in a graph $\mathcal{G}_{\mathcal{K}}$ if there exists an isomorphism from $\mathcal{G}_{\mathcal{K}} = (\mathcal{C}, \mathcal{O}, \mathcal{E})$ to itself, i.e., $\pi : \mathcal{C} \cup \mathcal{O} \rightarrow \mathcal{C} \cup \mathcal{O}$, such that $\pi(c_1) = c'_1, \dots, \pi(c_n) = c'_n$. Further, we use the following notation

$$(c_1, \dots, c_n) \xleftrightarrow{\mathcal{G}_{\mathcal{K}}} (c'_1, \dots, c'_n) : (c_1, \dots, c_n) \text{ and } (c'_1, \dots, c'_n) \text{ are isomorphic in } \mathcal{G}_{\mathcal{K}}.$$

Theorem 5.2. *Consider a knowledge base $\mathcal{K} = (\mathcal{C}, \mathcal{R}, \mathcal{O})$ and any $r \in \mathcal{R}$. Two latent random variables $X := r(c_1, \dots, c_n)$ and $X' := r(c'_1, \dots, c'_n)$ have the same posterior distribution in **any** MLN if and only if $(c_1, \dots, c_n) \xleftrightarrow{\mathcal{G}_{\mathcal{K}}} (c'_1, \dots, c'_n)$.*

Remark. We say two random variables $X, X' \in \mathcal{H}$ have the same posterior if the marginal distributions $P(X|\mathcal{O})$ and $P(X'|\mathcal{O})$ are the same and, moreover, for any sequence of random variables (X_1, \dots, X_n) in $\mathcal{H} \setminus \{X\}$, there exists a sequence of random variables (X'_1, \dots, X'_n) in $\mathcal{H} \setminus \{X'\}$ such that the marginal distributions $P(X, X_1, \dots, X_n|\mathcal{O})$ and $P(X', X'_1, \dots, X'_n|\mathcal{O})$ are the same.

A proof is given in Appendix A. The proof of necessary condition is basically showing that, if (c_1, \dots, c_n) and (c'_1, \dots, c'_n) are NOT isomorphic in $\mathcal{G}_{\mathcal{K}}$, we can always define a formula which can make $r(c_1, \dots, c_n)$ and $r(c'_1, \dots, c'_n)$ distinguishable in MLN and have different posterior distributions. It implies an important fact that, to obtain an expressive representation for the posterior,

- (1) either GNN embeddings need to be powerful enough to distinguish non-isomorphic nodes;
- (2) or the information of formulae / MLN need to be incorporated into the parametrization.

The second condition (2) somehow defeats our purpose of using mean field approximation to speed up the inference, so it is currently not considered. Unfortunately, condition (1) is also not satisfied, because existing GNNs are *at most* as powerful as color refinement, which is not an exact graph isomorphism test. Besides, node isomorphism mentioned in Theorem 5.2 is even more complex than graph isomorphism because it is a constrained graph isomorphism.

We will interpret the implications of this theorem by an example. Figure 3 shows a factor graph representation for a knowledge base which leads to the following observations:

- Even though A and B have opposite relations with E , i.e., $F(A, E) = 1$ but $F(B, E) = 0$, A and B are indistinguishable in $\mathcal{G}_{\mathcal{K}}$ and thus have the same GNN embeddings, i.e., $\mu_A = \mu_B$.

• Suppose $f(c, c') := F(c, c') \Rightarrow L(c, c')$ is the only formula in MLN (L is Like and F is Friend). $L(A, E)$ and $L(B, E)$ apparently have different posteriors. However, using GNN embeddings, $Q(L(A, E)) = \text{logistic}(\text{MLP}_L(\mu_A, \mu_E))$ is always identical to $Q(L(B, E)) = \text{logistic}(\text{MLP}_L(\mu_B, \mu_E))$.

• There exists an isomorphism $\pi_1 : (A, E, B, F) \mapsto (B, F, A, E)$ such that $\pi_1(A) = B$, but no isomorphism π satisfies both $\pi(A) = B$ and $\pi(E) = E$. Therefore, we see how the isomorphism constraints in Theorem 5.2 make the problem even more complex than graph isomorphism check.

To conclude, it is revealed that node embeddings by GNN alone are not enough to express the posterior in MLN. We provide more examples in Appendix B to explain that this case is very common and not a rare case. In the next section, we will introduce a way of correcting the node embeddings.

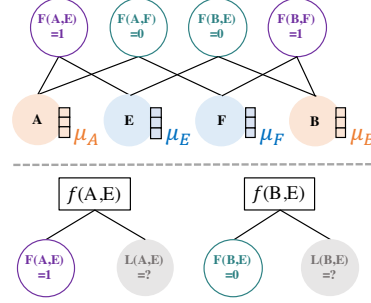


Figure 3: Top: A knowledge base with 0-1-0-1 loop. Bottom: MLN.

6 ExpressGNN: More Expressive GNN with Tunable Embeddings

It is currently challenging to design a new GNN that can check nodes isomorphism, because no polynomial-time algorithm is known even for unconstrained graph isomorphism test [14, 15].

In this section, we propose a simple yet effective solution. Take Figure 3 as an example:

To make $Q(L(A, E))$ different from $Q(L(B, E))$, we can simply introduce additional **low dimensional** tunable embeddings ω_A, ω_B , and ω_E and correct the parametrization as

$$\text{logistic}(\text{MLP}_L([\mu_A, \omega_A], [\mu_E, \omega_E])) \text{ and } \text{logistic}(\text{MLP}_L([\mu_B, \omega_B], [\mu_E, \omega_E])).$$

With tunable ω_A and ω_B , $Q(L(A, E))$ and $Q(L(B, E))$ can result in different values. In general, we can assign each entity $c \in \mathcal{C}$ a low-dimensional tunable embedding ω_c and concatenate it with the GNN embedding μ_c to represent this entity. We call this variant ExpressGNN and describe the parametrization of Q in Algorithm 2.

One can think of ExpressGNN as a hierarchical encoding of entities: GNN embeddings assign similar codes to nodes similar in knowledge graph neighborhoods, while the tunable embeddings provide additional capacity to code variations beyond knowledge graph structures. The hope is only a very low dimensional tunable embedding is needed to fine-tune individual differences. Then the total number of parameters in ExpressGNN could be much smaller than using tunable embedding alone.

ExpressGNN also presents an interesting trade-off between induction and transduction ability. The GNN embedding part allows ExpressGNN to possess some generalization ability to new entities and different knowledge graphs; while the tunable embedding part gives ExpressGNN the extra representation power to perform accurate inference in the current knowledge graph.

7 Experiments

Our experiments show that mean field approximation with ExpressGNN enables efficient and effective probabilistic logic inference and lead to to state-of-the-art results in several benchmark and large datasets.

Benchmark datasets. (i) UW-CSE contains information of students and professors in five department (AI, Graphics, Language, System, Theory) [1]. (ii) Cora [16] contains a collection of citations to computer science research papers. It is split into five subsets according to the research field. (iii) synthetic Kinship datasets contain kinship relationships (e.g., Father, Brother) and resemble the popular Kinship dataset [17]. (iv) FB15K-237 is a large-scale knowledge base [18]. Statistics of datasets are provided in Table 1. See more details of datasets in Appendix C.

Table 1: Statistics of datasets.

Dataset		#entity	#ground predicate	#ground formula
FB15K-237		15K	50M	679B
Cora (avg)		616	157K	457M
Kinship	S1	62	50K	550K
	S2	110	158K	3M
	S3	160	333K	9M
	S4	221	635K	23M
	S5	266	920K	39M
UW-CSE	AI	300	95K	73M
	Graphics	195	70K	64M
	Language	82	15K	9M
	Systems	277	95K	121M
	Theory	174	51K	54M

7.1 Ablation study and comparison to strong MLN inference methods

We conduct experiments on Kinship, UW-CSE and Cora, since other baselines can only scale up to these datasets. We use the original logic formulae provided in UW-CSE and Cora, and use hand-coded rules for Kinship. The weights for all formulae are set to 1. We use area under the precision-recall curve (AUC-PR) to evaluate deductive inference accuracy for **predicates never seen during training**, and under the **open-world** setting.¹ See Appendix C for more details. Before comparing to other baselines, we first perform an ablation study for ExpressGNN in Cora to explore the trade-off between GNN and tunable embeddings.

Ablation study. The number of parameters in GNN is independent of entity size, but it is less expressive. The number of parameters in the tunable component is linear in entity size, but it is more expressive. Results on different combinations of these two components are shown in Table 2, which are consistent with our analytical result: GNN alone is not expressive enough.

It is observed that GNN64+Tune4 has comparable performance with Tune64, but consistently better than GNN64. However, the number of parameters in GNN64+Tune4 is $O(64^2 + 4|C|)$, while that in Tune64 is $O(64|C|)$. A similar result is observed for GNN64+Tune64 and Tune128. Therefore, ExpressGNN as a combination of two types of embeddings can possess the advantages of both having a small number of parameters and being expressive. Therefore, we will use ExpressGNN throughout the rest of the experiments with hyperparameters optimized on the validation set. See Appendix C for details.

Inference accuracy. We evaluate the inference accuracy of ExpressGNN against a number of state-of-the-art MLN inference algorithms: (i) MCMC (Gibbs Sampling) [19, 1]; (ii) Belief Propagation (BP) [20]; (iii) Lifted Belief Propagation (Lifted BP) [8]; (iv) MC-SAT [21]; (v) Hinge-Loss Markov Random Field (HL-MRF) [22]. Results are shown in Table 3.

Table 3: Inference accuracy (AUC-PR) of different methods on three benchmark datasets.

Method	Kinship					UW-CSE					Cora	
	S1	S2	S3	S4	S5	AI	Graphics	Language	Systems	Theory	(avg)	
MCMC	0.53	-	-	-	-	-	-	-	-	-	-	
BP / Lifted BP	0.53	0.58	0.55	0.55	0.56	0.01	0.01	0.01	0.01	0.01	-	
MC-SAT	0.54	0.60	0.55	0.55	-	0.03	0.05	0.06	0.02	0.02	-	
HL-MRF	1.00	1.00	1.00	1.00	-	0.06	0.06	0.02	0.04	0.03	-	
ExpressGNN	0.97	0.97	0.99	0.99	0.99	0.09	0.19	0.14	0.06	0.09	0.64	

A hyphen in the entry indicates that the inference is either out of memory or exceeds the time limit (24 hours). Note that since the lifted BP is guaranteed to get identical results as BP [8], the results of these two methods are merged into one row. For UW-CSE, the results suggest that ExpressGNN consistently outperforms all baselines. On synthetic Kinship, since the dataset is noise-free, HL-MRF achieves the score of 1 for the first four sets. ExpressGNN yields similar but not perfect scores for all the subsets, presumably caused by the stochastic nature of our sampling and optimization method.

Inference efficiency. The inference time on UW-CSE and Kinship are summarized in Figure 4 (Cora is omitted as none of the baselines is feasible). As the size of the dataset grows linearly, inference time of all baseline methods grows exponentially. ExpressGNN maintains a nearly constant inference time with the increasing size of the dataset, demonstrating strong scalability. For HL-MRF, while maintaining a comparatively short wall-clock time, it exhibits an exponential increase in the space complexity. Slower methods such as MCMC and BP becomes infeasible for large datasets. ExpressGNN outperforms all baseline methods by at least one or two orders of magnitude.

7.2 Large-scale knowledge base completion

We use a large-scale dataset, FB15K-237 [18], to show the scalability of ExpressGNN. Since none of the aforementioned probabilistic inference methods are tractable on this dataset, we compare with several state-of-the-art supervised methods for knowledge base completion: (i) Neural Logic

¹In Appendix E, we report the performance under the closed-world setting as in the original works.

Method	Inference Time (minutes)				
	AI	Graphics	Language	Systems	Theory
MCMC	>24h	>24h	>24h	>24h	>24h
BP	408	352	37	457	190
Lifted BP	321	270	32	525	243
MC-SAT	172	147	14	196	86
HL-MRF	135	132	18	178	72
ExpressGNN	14	20	5	7	13

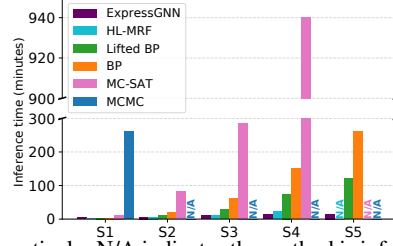


Figure 4: *Left / Right*: Inference time on UW-CSE / Kinship respectively. N/A indicates the method is infeasible.

Programming (Neural LP) [23]; (ii) Neural Tensor Network (NTN) [24]; (iii) TransE [25]. In these knowledge completion experiments, we follow the setting in [10] to add a discriminative loss $\sum_{r(a_r) \in \mathcal{O}} \log Q(r(a_r))$ to better utilize observed data. We use Neural LP to generate candidate rules and pick up those with high confidence scores for ExpressGNN. See Appendix F for examples of logic formulae used in experiments. For competitor methods, we use default tuned hyperparameters, which can reproduce the experimental results reported in their original works.

Evaluation. Given a query, e.g., $r(c, c')$, the task is to rank the query on top of all possible grounding of r . For evaluation, we compute the Mean Reciprocal Ranks (MRR), which is the average of the reciprocal rank of all the truth queries, and Hits@10, which is the percentage of truth queries that are ranked among top 10. Following the protocol proposed in [23, 25], we also use *filtered* rankings.

Table 4: Comparison on FB15K-237 with varied training set size.

Model	MRR					Hits@10				
	0%	5%	10%	20%	100%	0%	5%	10%	20%	100%
Neural LP	0.01	0.13	0.15	0.16	0.24	1.5	23.2	24.7	26.4	36.2
NTN	0.09	0.10	0.10	0.11	0.13	17.9	19.3	19.1	19.6	23.9
TransE	0.21	0.22	0.22	0.22	0.28	36.2	37.1	37.7	38.0	44.5
ExpressGNN	0.41	0.42	0.43	0.49	0.52	54.0	58.2	61.1	62.2	62.5

Table 5: Inductive knowledge completion on FB15K-237.

Model	MRR	Hits@10
Neural LP	0.18	31.6
NTN	0.00	0.0
TransE	0.00	0.0
ExpressGNN	0.26	41.0

Data efficiency in transductive setting. We demonstrate the data efficiency of using logic formula and compare ExpressGNN with aforementioned supervised approaches. More precisely, we follow [23] to split the knowledge base into facts / training / validation / testing sets, vary the size of the training set from 0% to 100%, and feed the varied training set with the same complete facts set to models for training. Evaluations on testing set are given in Table 4. It shows with small training data ExpressGNN can generalize significantly better than supervised methods. With more supervision, supervised approaches start to close the gap with ExpressGNN. This also suggests that high confidence logic rules indeed help us generalize better under small training data.

Inductive ability. To demonstrate the inductive learning ability of ExpressGNN, we conduct experiments on FB15K-237 where training and testing use disjoint sets of both entities and relations. To prepare data for such setting, we first randomly select a subset of relations, and restrict the test set to relations in this selected subset, which is similar to [25]. Then we exclude from the training data any tuples that share at least one entity with any tuple in the test set. After we learn ExpressGNN, only the inductive GNN embedding part is used for predicting the test data. Table 5 shows the experimental results. As expected, in this inductive setting, supervised transductive learning methods such as NTN and TransE drop to zero in terms of MRR and Hits@10². In contrast, Neural LP and ExpressGNN are much less affected by the number of unseen entities and relations, and both achieve reasonable performance at the same scale as the non-inductive setting. Since our method can directly exploit first-order logic, we achieve better performance than Neural LP.

8 Conclusion

Our analysis shows that GNN while being suitable for probabilistic logic inference in MLN, is not expressive enough. Motivated by this analysis, we propose ExpressGNN, an integrated GNN and tunable embedding approach, which has a trade-off between model size and expressiveness, and leads to scalable and effective logic inference in both transductive and inductive experiments. ExpressGNN opens up many possibilities for future research such as formula weight learning with variational inference, incorporating entity features and neural tensorized logic formulae, and addressing challenging datasets such as GQA [26].

²Since we use the worst-case ranking to break the tie, the MRR is smaller than 1e-5 for NTN and TransE.

References

- [1] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2): 107–136, 2006.
- [2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.
- [3] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.
- [4] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711, 2016.
- [5] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.
- [6] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
- [7] Will Hamilton, Zhitaoying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.
- [8] Parag Singla and Pedro M Domingos. Lifted first-order belief propagation. In *AAAI*, volume 8, pages 1094–1099, 2008.
- [9] Parag Singla, Aniruddh Nath, and Pedro M Domingos. Approximate lifting techniques for belief propagation. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.
- [10] Qu Meng, Bengio Yoshua, and Tang Jian. Gmn: Graph markov neural networks. *arXiv preprint [arXiv:1905.06214](https://arxiv.org/abs/1905.06214)*, 2019.
- [11] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.
- [12] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint [arXiv:1810.00826](https://arxiv.org/abs/1810.00826)*, 2018.
- [13] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep): 2539–2561, 2011.
- [14] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.
- [15] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697. ACM, 2016.
- [16] Parag Singla and Pedro Domingos. Discriminative training of markov logic networks. In *AAAI*, volume 5, pages 868–873, 2005.
- [17] Woodrow W Denham. *The detection of patterns in Alyawara nonverbal behavior*. PhD thesis, University of Washington, Seattle., 1973.
- [18] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, 2015.
- [19] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC, 1995.
- [20] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Generalized belief propagation. In *Advances in neural information processing systems*, pages 689–695, 2001.

- [21] Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, volume 6, pages 458–463, 2006.
- [22] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *arXiv preprint [arXiv:1505.04406](#)*, 2015.
- [23] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base completion. *CoRR*, *abs/1702.08367*, 2017.
- [24] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.
- [25] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.
- [26] Drew A Hudson and Christopher D Manning. Gqa: a new dataset for compositional question answering over real-world images. *arXiv preprint [arXiv:1902.09506](#)*, 2019.
- [27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint [arXiv:1412.6980](#)*, 2014.

Appendix

A Proof of Theorems

Theorem 5.1. Let $\mathcal{G}_\mathcal{K} = (\mathcal{C}, \mathcal{O}, \mathcal{E})$ be the factor graph for a knowledge base \mathcal{K} and $\mathcal{G}_{\bar{\mathcal{K}}} = (\mathcal{C}, \mathcal{O} \cup \mathcal{H}, \mathcal{E} \cup \mathcal{E}_\mathcal{H})$ be the corresponding augmented version. Then the following two statements are true:

- (1) $c \xleftrightarrow{\mathcal{G}_\mathcal{K}} c'$ if and only if $c \xleftrightarrow{\mathcal{G}_{\bar{\mathcal{K}}}} c'$;
- (2) $[r(c_1, \dots, c_n) = v] \xleftrightarrow{\mathcal{G}_{\bar{\mathcal{K}}}} [r(c'_1, \dots, c'_n) = v]$ if and only if $(c_1, \dots, c_n) \xleftrightarrow{\mathcal{G}_{\bar{\mathcal{K}}}} (c'_1, \dots, c'_n)$.

Proof. For simplicity, we use \mathcal{G} and \mathcal{G}' to represent $\mathcal{G}_\mathcal{K}$ and $\mathcal{G}_{\bar{\mathcal{K}}}$ in this proof.

2. Let us first assume statement 1 is true and prove statement 2.

The neighbors of $H := [r(c_1, \dots, c_n) = v]$ and $H' := [r(c'_1, \dots, c'_n) = v]$ are

$$\mathcal{N}(H) = \{(c_i, i) : i = 1, \dots, n\} \text{ and } \mathcal{N}(H') = \{(c'_i, i) : i = 1, \dots, n\} \quad (4)$$

where i represents the edge type. It is easy to see that H and H' are indistinguishable in \mathcal{G}' if and only if c'_i and c_i are indistinguishable in \mathcal{G}' for $i = 1, \dots, n$. By statement 1, c'_i and c_i are indistinguishable in \mathcal{G}' if and only if c'_i and c_i are indistinguishable in \mathcal{G} . Hence, statement 2 is true. Now it remains to prove statement 1.

1. (\Leftarrow) If c and c' are distinguishable in \mathcal{G} , it is easy to see c and c' are also distinguishable in the new graph \mathcal{G}' . The reason is that the newly added nodes \mathcal{H} are of different types from the observed nodes \mathcal{O} in \mathcal{G} , so that these newly added nodes can not make two distinguishable nodes to become indistinguishable.

(\Rightarrow) Assume that c and c' are indistinguishable in \mathcal{G} , we will prove they are indistinguishable in \mathcal{G}' using MI (mathematical induction). The idea is to construct the new graph \mathcal{G}' by connecting the unobserved nodes in \mathcal{H} in a particular order. More specifically, we first connect all unobserved grounded predicates $[r(c_1, \dots, c_n) = ?] \in \mathcal{H}$ to their first arguments c_1 , and the resulting graph is denoted by $\mathcal{G}^{(1)}$. Then we can connect all $[r(c_1, c_2, \dots, c_n) = ?] \in \mathcal{H}$ to their second arguments c_2 and denote the resulting graph by $\mathcal{G}^{(2)}$. In this way, we obtain a sequence of graphs $\{\mathcal{G}^{(k)}\}_{k=1}^R$ where $R := \max\{n : r \in \mathcal{R}\}$ is the maximal number of arguments. It is clear that $\mathcal{G}' = \mathcal{G}^{(R)}$. In the following, we will use MI to prove that for all $k = 1, \dots, R$, if c and c' are indistinguishable in \mathcal{G} , then they are indistinguishable in $\mathcal{G}^{(k)}$.

Proof of (MI 1):

Consider any predicate $r \in \mathcal{R}$. For any two indistinguishable nodes c, c' in \mathcal{G} , $\#\{r(c, \dots) : \text{observed}\} = \#\{r(c', \dots) : \text{observed}\}$. Hence, it is obvious that

$$\#\{r(c, \dots) : \text{unobserved}\} = \#\{r(c', \dots) : \text{unobserved}\} = M. \quad (5)$$

Before connected to the graph \mathcal{G} , the unobserved nodes $\{r(\cdot) : \text{unobserved}\}$ are all indistinguishable because they are of the same node-type. Now we connect all these unobserved nodes to its first argument. Then c is connected to $\{r(c, \dots) : \text{unobserved}\}$ and c' is connected to $\{r(c', \dots) : \text{unobserved}\}$. Since both c and c' are connected M unobserved nodes and these nodes are indistinguishable, c and c' remain to be indistinguishable. Also, after connected to its first argument, $r(c, \dots)$ and $r(c', \dots)$ are indistinguishable if and only if c and c' are indistinguishable, which is obvious.

Similarly, we can connect all unobserved grounded predicates to its first argument. In the resulting graph, two nodes are indistinguishable if they are indistinguishable in \mathcal{G} .

Assumption (MI k):

Assume that after connecting all unobserved grounded predicates to their first k arguments, the constant nodes in the resulting graph, $\mathcal{G}^{(k)}$, are indistinguishable if they are indistinguishable in \mathcal{G} .

Proof of (MI $k + 1$):

The constants \mathcal{C} in \mathcal{G} can be partitioned into N groups $\mathcal{C} = \bigcup_{i=1}^N \mathcal{C}^{(i)}$, where the constants in the same group $\mathcal{C}^{(i)}$ are indistinguishable in \mathcal{G} (and also indistinguishable in $\mathcal{G}^{(k)}$).

Consider a predicate $r^* \in \mathcal{R}$. The set of unobserved grounded predicates where the $(k+1)$ -th argument is c can be written as

$$\begin{aligned} & \{r^*(\dots, c_{k+1} = c, \dots) : \text{unobserved}\} \\ &= \bigcup_{i_1=1}^N \dots \bigcup_{i_k=1}^N \left\{ r^*(c_1, \dots, c_k, c_{k+1} = c, \dots) : c_i \in \mathcal{C}^{(i_1)}, \dots, c_k \in \mathcal{C}^{(i_k)}, \text{unobserved} \right\}. \end{aligned} \quad (6)$$

Similar to the arguments in **(MI 1)**, for any two indistinguishable nodes c and c' , for any fixed sequence of groups i_1, \dots, i_k , the size of the following two sets are the same:

$$\begin{aligned} M(i_1, \dots, i_k) &= \# \{r^*(c_1, \dots, c_k, c_{k+1} = c, \dots) : c_i \in \mathcal{C}^{(i_1)}, \dots, c_k \in \mathcal{C}^{(i_k)}, \text{unobserved}\} \\ &= \# \{r^*(c_1, \dots, c_k, c_{k+1} = c', \dots) : c_i \in \mathcal{C}^{(i_1)}, \dots, c_k \in \mathcal{C}^{(i_k)}, \text{unobserved}\}. \end{aligned} \quad (7)$$

Also, all grounded predicates in the above two sets are indistinguishable in $\mathcal{G}^{(k)}$ because their first k arguments are indistinguishable. Hence, these are two sets of $M(i_1, \dots, i_k)$ many indistinguishable nodes. In conclusion, the two sets $\{r^*(\dots, c_{k+1} = c, \dots) : \text{unobserved}\}$ and $\{r^*(\dots, c_{k+1} = c', \dots) : \text{unobserved}\}$ are indistinguishable in $\mathcal{G}^{(k)}$ if c and c' are indistinguishable.

Now we connect all unobserved $r^*(\cdot)$ to their $(k+1)$ -th arguments. Then the constant node c is connected to $\{r^*(\dots, c_{k+1} = c, \dots) : \text{unobserved}\}$ and c' is connected to $\{r^*(\dots, c_{k+1} = c', \dots) : \text{unobserved}\}$. Since these two sets are indistinguishable, then c and c' remain to be indistinguishable.

Similarly, for other predicates $r \in \mathcal{R}$, we can connect all unobserved grounded predicates $r(\cdot)$ to their $(k+1)$ -th arguments. In the resulting graph, $\mathcal{G}^{(k+1)}$, any pair of two nodes will remain indistinguishable if they are indistinguishable in \mathcal{G} .

□

Theorem 5.2. Consider a knowledge base $\mathcal{K} = (\mathcal{C}, \mathcal{R}, \mathcal{O})$ and any $r \in \mathcal{R}$. Two latent random variables $X := r(c_1, \dots, c_n)$ and $X' := r(c'_1, \dots, c'_n)$ have the same posterior distribution in **any** MLN if and only if $(c_1, \dots, c_n) \xleftrightarrow{\mathcal{G}_{\mathcal{K}}} (c'_1, \dots, c'_n)$.

Proof. A graph isomorphism from G to itself is called automorphism, so in this proof, we will use the terminology - automorphism - to indicate such a self-bijection.

(\Leftarrow) We first prove the sufficient condition:

If \exists automorphism π on the graph $\mathcal{G}_{\mathcal{K}}$ such that $\pi(c_i) = c'_i, \forall i = 1, \dots, n$, then for any $r \in \mathcal{R}$, $r(c_1, \dots, c_n)$ and $r(c'_1, \dots, c'_n)$ have the same posterior in any MLN.

MLN is a graphical model that can also be represented by a factor graph $\text{MLN} = (\mathcal{O} \cup \mathcal{H}, \mathcal{F}_g, \mathcal{E})$ where grounded predicates (random variables) and grounded formulae (potential) are connected. We will show that \exists an automorphism ϕ on MLN such that $\phi(r(c_1, \dots, c_n)) = r(c'_1, \dots, c'_n)$. Then the sufficient condition is true. This automorphism ϕ is easy to construct using the automorphism π on $\mathcal{G}_{\mathcal{K}}$. More precisely, we define $\phi : (\mathcal{O} \cup \mathcal{H}, \mathcal{F}_g) \rightarrow (\mathcal{O} \cup \mathcal{H}, \mathcal{F}_g)$ as

$$\phi(r(a_r)) = r(\pi(a_r)), \quad \phi(f(a_f)) = f(\pi(a_f)), \quad (10)$$

for any predicate $r \in \mathcal{R}$, any assignments a_r to its arguments, any formula $f \in \mathcal{F}$, and any assignments a_f to its arguments. It is easy to see ϕ is an automorphism:

1. Since π is a bijection, apparently ϕ is also a bijection.
2. The above definition preserves the binding of the arguments. $r(a_r)$ and $f(a_f)$ are connected if and only if $\phi(r(a_r))$ and $f(\pi(a_f))$ are connected.
3. Given the definition of π , we know that $r(a_r)$ and $r(\pi(a_r))$ have the same observation value. Therefore, in MLN, $\text{NodeType}(r(a_r)) = \text{NodeType}(\phi(r(a_r)))$.

This completes the proof of the sufficient condition.

(\implies) To prove the necessary condition, it is equivalent to show given the following assumption

(A 1): there is no automorphism π on the graph $\mathcal{G}_{\mathcal{K}}$ such that $\pi(c_i) = c'_i, \forall i = 1, \dots, n$,

the following statement is true:

there must exists a MLN and a predicate r in it such that $r(c_1, \dots, c_n)$ and $r(c'_1, \dots, c'_n)$ have different posterior.

Before showing this, let us first introduce the **factor graph representation of a single logic formula** f .

A logic formula f can be represented as a factor graph, $\mathcal{G}_f = (\mathcal{C}_f, \mathcal{R}_f, \mathcal{E}_f)$, where nodes on one side of the graph is the set of distinct constants \mathcal{C}_f needed in the formula, while nodes on the other side is the set of predicates \mathcal{R}_f used to define the formula. The set of edges, \mathcal{E}_f , will connect constants to predicates or predicate negation. That is, an edge

$e = (c, r, i)$ between node c and predicate r exists, if the predicate r use constant c in its i -th argument.

We note that the set of distinctive constants used in the definition of logic formula are templates where actual constant can be in-

stantiated from \mathcal{C} . An illustration of logic formula factor graph can be found in Figure 5. Similar to the factor graph for the knowledge base, we also differentiate the type of edges by the position of the argument.

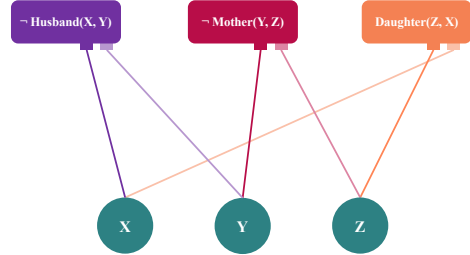


Figure 5: An example of factor graph for the logic formula $\neg\text{Husband}(X, Y) \vee \neg\text{Mother}(Y, Z) \vee \text{Daughter}(Z, X)$.

Therefore, every single formula can be represented by a factor graph. We will construct a factor graph representation to define a particular formula, and show that the MLN induced by this formula will result in different posteriors for $r(c_1, \dots, c_n)$ and $r(c'_1, \dots, c'_n)$. The factor graph for the formula is constructed in the following way (See Figure 8 as an example of the resulting formula constructed using the following steps):

(i) Given the above assumption (A 1), we claim that:

\exists a subgraph $\mathcal{G}_{c_{1:n}}^* = (\mathcal{C}_c^*, \mathcal{O}_c^*, \mathcal{E}_c^*) \subseteq \mathcal{G}_{\mathcal{K}}$ such that all subgraphs $\mathcal{G}_{c'_{1:n}} = (\mathcal{C}_{c'}^*, \mathcal{O}_{c'}^*, \mathcal{E}_{c'}^*) \subseteq \mathcal{G}_{\mathcal{K}}$ satisfy:

(Condition) if there exists an isomorphism $\phi : \mathcal{G}_{c_{1:n}}^* \rightarrow \mathcal{G}_{c'_{1:n}}^*$ satisfying $\phi(c_i) = c'_i, \forall i = 1, \dots, n$ after the observation values are IGNORED (that is, $[r_j(\dots) = 0]$ and $[r_j(\dots) = 1]$ are treated as the SAME type of nodes), then the set of fact nodes (observations) in these two graphs are different (that is, $\mathcal{O}_c^* \neq \mathcal{O}_{c'}^*$).

The proof of this claim is given at the end of this proof.

(ii) Next, we use $\mathcal{G}_{c_{1:n}}^*$ to define a formula f . We first initialize the definition of the formula value as

$$f(c_1, \dots, c_n, \tilde{c}_1, \dots, \tilde{c}_n) = (\wedge \{ \tilde{r}(a_{\tilde{r}}) : \tilde{r}(a_{\tilde{r}}) \in \mathcal{G}_{c_{1:n}}^* \}) \Rightarrow r(c_1, \dots, c_n). \quad (11)$$

Then, we change $\tilde{r}(a_{\tilde{r}})$ in this formula to the negation $\neg\tilde{r}(a_{\tilde{r}})$ if the observed value of $\tilde{r}(a_{\tilde{r}})$ is 0 in $\mathcal{G}_{c_{1:n}}^*$.

We have defined a formula f using the above two steps. Suppose the MLN only contains this formula f . Then

the two nodes $r(c_1, \dots, c_n)$ and $r(c'_1, \dots, c'_n)$ in this MLN must be distinguishable.

The reason is, in MLN, $r(c_1, \dots, c_n)$ is connected to a grounded formula $f(c_1, \dots, c_n, \tilde{c}_1, \dots, \tilde{c}_n)$, whose factor graph representation is $\mathcal{G}_{c_{1:n}}^* \cup r(c_1, \dots, c_n)$. In this formula, all variables are observed in the knowledge base \mathcal{K} except for $r(c_1, \dots, c_n)$ and the observation set is \mathcal{O}_c^* . The formula value is

$$f(c_1, \dots, c_n, \tilde{c}_1, \dots, \tilde{c}_n) = (1 \Rightarrow r(c_1, \dots, c_n)). \quad (12)$$

Clarification: Equation 11 is used to **define** a formula and c_i in this equation can be replaced by other constants, while Equation 12 represents a **grounded** formula whose arguments are exactly $c_1, \dots, c_n, \tilde{c}_1, \dots, \tilde{c}_n$. Based on **(Condition)**, there is NO formula $f(c'_1, \dots, c'_n, \tilde{c}'_1, \dots, \tilde{c}'_n)$ that contains $r(c'_1, \dots, c'_n)$ has an observation set the same as \mathcal{O}_c^* . Therefore, $r(c_1, \dots, c_n)$ and $r(c'_1, \dots, c'_n)$ are distinguishable in this MLN.

Proof of claim:

We show the existence by constructing the subgraph $\mathcal{G}_{c_{1:n}}^* \subseteq \mathcal{G}_{\mathcal{K}}$ in the following way:

(i) First, we initialize the subgraph as $\mathcal{G}_{c_{1:n}}^* := \mathcal{G}_{\mathcal{K}}$. Given assumption **(A 1)** stated above, it is clear that

(S 1) \forall subgraph $\mathcal{G}' \subseteq \mathcal{G}_{\mathcal{K}}$, there is no isomorphism $\pi : \mathcal{G}_{c_{1:n}}^* \rightarrow \mathcal{G}'$ satisfying $\pi(c_i) = c'_i, \forall i = 1, \dots, n$.

(ii) Second, we need to check whether the following case occurs:

(C 1) \exists a subgraph $\mathcal{G}' = (\mathcal{C}', \mathcal{O}', \mathcal{E}')$ such that (1) there EXISTS an isomorphism $\phi : \mathcal{G}_{c_{1:n}}^* \rightarrow \mathcal{G}'$ satisfying $\phi(c_i) = c'_i, \forall i = 1, \dots, n$ after the observation values are IGNORED (that is, $[r_j(\dots)] = 0$ and $[r_j(\dots)] = 1$ are treated as the same type of nodes); and (2) the set of factor nodes (observations) in these two graphs are the same (that is, $\mathcal{O}_c^* = \mathcal{O}'$).

(iii) Third, we need to modify the subgraph if the case **(C 1)** is observed. Since $|\mathcal{G}_{c_{1:n}}^*| \geq |\mathcal{G}'|$, the only subgraph that will lead to the case **(C 1)** is the maximal subgraph $\mathcal{G}_{c_{1:n}}^*$. The isomorphism ϕ is defined by ignoring the observation values, while the isomorphism π in **(S 1)** is not ignoring them. Thus,

(S 1) and **(C 1)** $\implies \exists$ a set of nodes $S := \{[r_j(a^{(1)}) = 0], \dots, [r_j(a^{(n)}) = 0]\}$ such that for any isomorphism ϕ satisfying the conditions in **(C 1)**, the range $\phi(S)$ contains at least one node $[r_j(\cdot) = 1]$ which has observation value 1.

Otherwise, it is easy to see a contradiction to statement **(S 1)**.

(M 1) Modify the subgraph by $\mathcal{G}_{c_{1:n}}^* \leftarrow \mathcal{G}_{c_{1:n}}^* \setminus S$. The nodes (and also their edges) in the set $S := \{[r_j(a^{(1)}) = 0], \dots, [r_j(a^{(n)}) = 0]\}$ are removed.

For the new subgraph $\mathcal{G}_{c_{1:n}}^*$ after the modification **(M 1)**, the case **(C 1)** will not occur. Thus, we've obtained a subgraph that satisfies the conditions stated in the claim. Finally, we can remove the nodes that are not connected with $\{c_1, \dots, c_n\}$ (that is, there is no path between this node and any one of $\{c_1, \dots, c_n\}$). The remaining graph is connected to $\{c_1, \dots, c_n\}$ and still satisfies the conditions that we need.

□

B Counter Examples

We provide more examples in this section to show that it is more than a rare case that GNN embeddings alone are not expressive enough.

B.1 Example 1

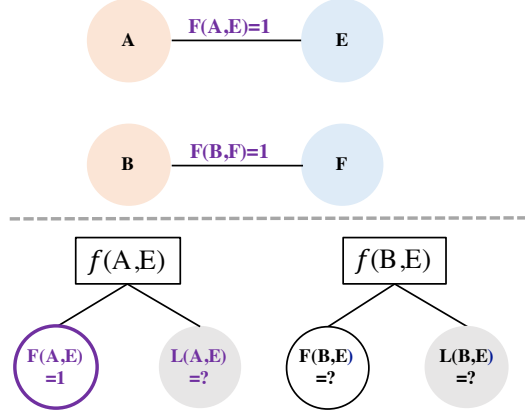


Figure 6: Example 1. *Top*: Knowledge base. *Bottom*: MLN

Unlike the example shown in main text, where A and B have OPPOSITE relation with E, Figure 6 shows a very simple example where A and B have exactly the same structure which makes A and B indistinguishable and isomorphic. However, since (A,E) and (B,E) are not isomorphic, it can be easily seen that $L(A,E)$ has different posterior from $L(B,E)$.

B.2 Example 2

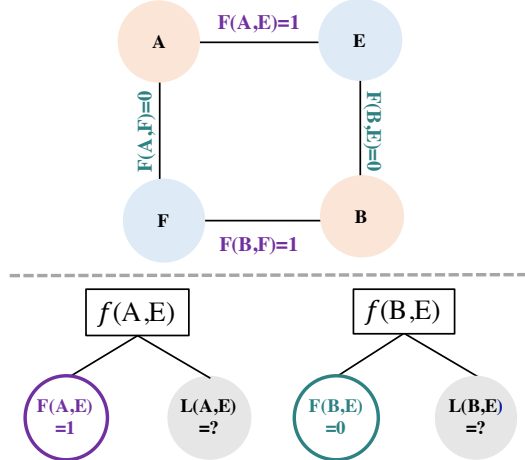


Figure 7: The same example as in Figure 3. *Top*: Knowledge base. *Bottom*: MLN

Figure 7 shows an example which is the same as in Figure 3. However, in this example, it is already revealed in the knowledge base that (A,E) and (B,E) have different local structures as they are connected by different observations. That is, $(A, [F(A,E) = 1], E)$ and $(B, [F(B,E) = 0], E)$ can be distinguished by GNN.

Now, we use another example in Figure 8 to show that even when the local structures are the same, the posteriors can still be different, which is caused by the formulae.

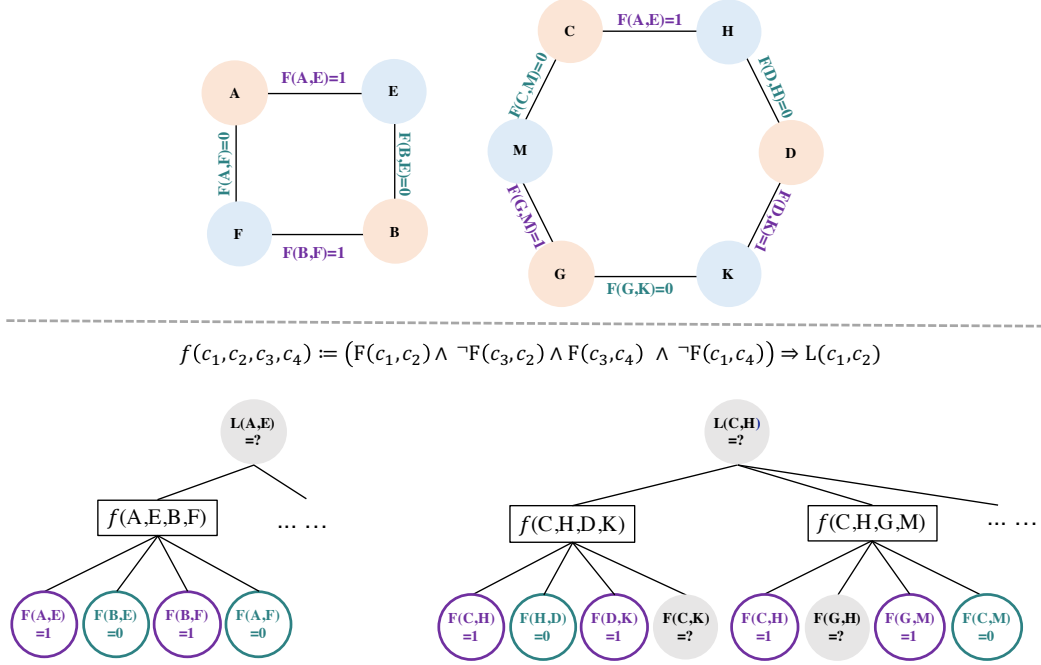


Figure 8: Example 2. *Top*: Knowledge base. *Bottom*: MLN

In Figure 8, (A, E) and (C, H) have the same local structure, so that the tuple $(A, [F(A, E) = 1], E)$ and $(C, [F(C, H) = 1], H)$ can NOT be distinguished by GNN. However, we can make use of subgraph (A, E, B, F) to define a formula, and then the resulting MLN gives different posterior to $L(A, E)$ and $L(C, H)$, as can be seen from the figure. Note that this construction of MLN is the same as the construction steps stated in the proof in Section A.

C Experiment Settings

Experimental setup. All the experiments are conducted on a GPU-enabled (Nvidia RTX 2080 Ti) Linux machine powered by Intel Xeon Silver 4116 processors at 2.10GHz with 256GB RAM. We implement ExpressGNN using PyTorch and train it with Adam optimizer [27]. To ensure a fair comparison, we allocate the same computational resources (CPU, GPU and memory) for all the experiments. We use default tuned hyperparameters for competitor methods, which can reproduce the experimental results reported in their original works. For ExpressGNN, we use 0.0005 as the initial learning rate, and decay the learning rate by half for every 10 epochs without improvement in terms of validation loss. For Kinship, UW-CSE and Cora, we run ExpressGNN with a fixed number of iterations, and use the smallest subset from the original split for hyperparameter tuning. For FB15K-237, we use the original validation set to tune the hyperparameters.

Here are more details of the setup of ExpressGNN. We use a two-layer MLP with ReLU activation function as the nonlinear transformation for each embedding update step in the graph neural network in Algorithm 1. For different steps, we learn different MLP parameters. To increase the model capacity of ExpressGNN, we also use different MLP parameters for different edge type, and for a different direction of embedding aggregation. As we discussed in Section 7.1, the number of trainable parameters in GNN alone is independent of the number of entities in the knowledge base. Therefore, it has a minor impact on the computational cost by using different MLP parameters as described above. For each dataset, we search the configuration of ExpressGNN on either the validation set or the smallest subset. The configuration we search includes the embedding size, the split point of tunable embeddings and GNN embeddings, the number of embedding update steps, and the sampling batch size.

Task and evaluation metrics. The deductive logic inference task is to answer queries that typically involve single predicate. For example in UW-CSE, the task is to predict the $\text{AdvisedBy}(c, c')$ relation for all persons in the set. In Cora, the task is to de-duplicate entities, and one of the query predicates is $\text{SameAuthor}(c, c')$. As for Kinship, the task is to predict whether a person is male or female, i.e., $\text{Male}(c)$. For each possible substitution of the query predicate with different entities, the model is tasked to predict whether it’s true or not. Then we use the area under the precision-recall curve (AUC-PR) as the evaluation metric for inference accuracy. Due to the severe imbalance of positive and negative samples in typical logic reasoning tasks, the AUC-PR is better than the AUC-ROC to reflect the actual model performance and is widely used in the literature [1].

For knowledge base completion, all methods are evaluated on test sets with Mean Reciprocal Rank (MRR) and Hits@10. Both are commonly used metrics for knowledge base completion. For each test query $r(c, c')$ with respect to relation r , the model is tasked to generate a rank list over all possible instantiations of r and sort them according to the model’s confidence on how likely this instantiation is true. Then MRR is computed as the average over all the reciprocal ranks of each $r(c, c')$ in its corresponding rank list, and Hits@10 is computed as the average times of ranking of the true fact in top 10 predictions. If two candidate entities have the same score, we break the tie by ranking the wrong ones ahead. This ensures a fair comparison for all methods. Additionally, before evaluation, the rank list will be filtered [23, 25] so that it does not contain any true fact other than $r(c, c')$ itself.

Sampling method. As there are exponential many formulae in the sampling space, one cannot sample by explicitly enumerating all the formulae and permute them. On the other hand, not all ground formulae will contribute to the optimization during training. For example, a ground formula that only contains observed variables will not contribute gradients, as evaluating this formula is independent of the latent variable posterior.

To overcome these challenges, we propose the following efficient sampling scheme: 1) to sample a ground formula we start from uniformly sampling a formula f from the space of \mathcal{F} ; 2) shuffle its predicate space \mathcal{R}_f into a sequence; 3) for each predicate r popped from the top of the \mathcal{R}_f , with a probability of p_{obs} we instantiate it as an observed variable and with a probability of $1 - p_{obs}$ it will become a uniformly sampled variable; 4) to instantiate an observed variable, we list all facts stored in the knowledge base with respect to predicate r and uniformly sample from it. In the case where no fact can be found, or if we hit probability $1 - p_{obs}$, then the predicate r will be instantiated with a random constant. Once a formula is fully instantiated, we examine its form and reject those without any latent variable.

Table 6: Complete statistics of the benchmark and synthetic datasets.

Dataset	# entity	# relation	# fact	# query	# ground predicate	# ground formula
FB15K-237	15K	237	272K	20K	50M	679B
Kinship-S1	62	15	187	38	50K	550K
Kinship-S2	110	15	307	62	158K	3M
Kinship-S3	160	15	482	102	333K	9M
Kinship-S4	221	15	723	150	635K	23M
Kinship-S5	266	15	885	183	920K	39M
UW-CSE-AI	300	22	731	4K	95K	73M
UW-CSE-Graphics	195	22	449	4K	70K	64M
UW-CSE-Language	82	22	182	1K	15K	9M
UW-CSE-Systems	277	22	733	5K	95K	121M
UW-CSE-Theory	174	22	465	2K	51K	54M
Cora-S1	670	10	11K	2K	175K	621B
Cora-S2	602	10	9K	2K	156K	431B
Cora-S3	607	10	18K	3K	156K	438B
Cora-S4	600	10	12K	2K	160K	435B
Cora-S5	600	10	11K	2K	140K	339B

In experiments, we set p_{obs} as 0.9. The intuition is that one wants to prioritize on sampling formulae containing both observed and latent variables. Otherwise, in the cases where a formula is fully latent, GNN is essentially optimizing towards learning the prior distribution determined by the form of formula and its weight, which is unlikely to be close to the actual posterior distribution. Additionally, for knowledge completion on FB15K-237, we further control the sample space to be query-related. Each time the model is fed with a query $r(c, c')$, we sample only the ground formulae with r as the positive literal and containing constants c and c' .

D Dataset Details

For experiments, we use four benchmark datasets: 1) The social network dataset UW-CSE [1] contains publicly available information of students and professors in the CSE department of UW. The dataset is split into five sets according to the home department of the entities. 2) The entity resolution dataset Cora [16] consists of a collection of citations to computer science research papers. The dataset is also split into five subsets according to the field of research. 3) We introduce a synthetic dataset that resembles the popular Kinship dataset [17]. The original dataset contains kinship relationships (e.g., Father, Brother) among family members in the Alyawarra tribe from Central Australia. We generate five sets by linearly increasing the number of entities. 4) The knowledge base completion benchmark FB15K-237 [18] is a generic knowledge base constructed from Freebase, which is designed to a more challenging variant of FB15K. More specifically, FB15K-237 is constructed by removing near-duplicate and inverse relations from FB15K. The dataset is split into training / validation / testing and we use the same split of facts from training as in prior work [23].

D.1 Datasets statistics

The complete statistics of the benchmark and synthetic datasets are shown in Table 6. The statistics of Cora is averaged over its five splits. Examples of logic formulae used in four benchmark datasets are listed in Table 8.

D.2 Synthetic Kinship Dataset

The synthetic dataset closely resembles the original Kinship dataset but with a controllable number of entities. To generate a dataset with n entities, we randomly split n entities into two groups which represent the first and second generation respectively. Within each group, entities are grouped into a few sub-groups representing the sister- and brother-hood. Finally, entities from different sub-groups in the first generation are randomly coupled and a sub-group in the second generation is assigned to

Table 7: Inference performance of competitors and our method under the closed-world semantics.

Method	Cora					UW-CSE				
	S1	S2	S3	S4	S5	AI	Graphics	Language	Systems	Theory
MCMC	0.43	0.63	0.24	0.46	0.56	0.19	0.04	0.03	0.15	0.08
BP / Lifted BP	0.44	0.62	0.24	0.45	0.57	0.21	0.04	0.01	0.14	0.05
MC-SAT	0.43	0.63	0.24	0.46	0.57	0.13	0.04	0.03	0.11	0.08
HL-MRF	0.60	0.78	0.52	0.70	0.81	0.26	0.18	0.06	0.27	0.19

them as their children. To generate the knowledge base, one traverse this family tree, and record all kinship relations for each entity. In this experiment, we generate five datasets by linearly increasing the number of entities. Examples of the first-order logic formulae used in the Kinship dataset is summarized in Table 8.

E Inference with Closed-World Semantics for Baseline Methods

In Section 7.1 we compare ExpressGNN with five probabilistic inference methods under open-world semantics. This is different from the original works, where they generally adopt the closed-world setting due to the scalability issues. More specifically, the original works assume that the predicates (except the ones in the query) observed in the knowledge base is *closed*, meaning for all instantiations of these predicates that do not appear in the knowledge base are considered *false*. Note that only the query predicates remain open-world in this setting.

For sanity checking, we also conduct these experiments with a closed-world setting. We found the results summarized in Table 7 are close to those reported in the original works. This shows that we have a fair setup (including memory size, hyperparameters, etc.) for those competitor methods. Additionally, one can find that the AUC-PR scores compared to those (Table 3) under open-world setting are actually better. This is due to the way the datasets were originally collected and evaluated generally complies with the closed-world assumption. But this is very unlikely to be true for real-world and large-scale knowledge base such as Freebase and WordNet, where many *true* facts between entities are not observed. Therefore, in general, the open-world setting is much more reasonable, which we follow throughout this paper.

F Logic Formulae

We list some examples of logic formulae used in four benchmark datasets in Table 8. The full list of logic formulae is available in our source code repository. Note that these formulae are not necessarily as clean as being always true, but are typically true.

For UW-CSE and Cora, we use the logic formulae provided in the original dataset. UW-CSE provides 94 hand-coded logic formulae, and Cora provides 46 hand-coded rules. For Kinship, we hand-code 22 first-order logic formulae. For FB15K-237, we first use Neural LP [23] on the full data to generate candidate rules. Then we select the ones that have confidence scores higher than 90% of the highest scored formulae sharing the same target predicate. We also de-duplicate redundant rules that can be reduced to other rules by switching the logic variables. Finally, we have generated 509 logic formulae for FB15K-237.

Table 8: Examples of logic formulae used in four benchmark datasets.

Dataset	First-order Logic Formulae
Kinship	$\text{Father}(X,Z) \wedge \text{Mother}(Y,Z) \Rightarrow \text{Husband}(X,Y)$ $\text{Father}(X,Z) \wedge \text{Husband}(X,Y) \Rightarrow \text{Mother}(Y,Z)$ $\text{Husband}(X,Y) \Rightarrow \text{Wife}(Y,X)$ $\text{Son}(Y,X) \Rightarrow \text{Father}(X,Y) \vee \text{Mother}(X,Y)$ $\text{Daughter}(Y,X) \Rightarrow \text{Father}(X,Y) \vee \text{Mother}(X,Y)$
UW-CSE	$\text{taughtBy}(c, p, q) \wedge \text{courseLevel}(c, \text{Level500}) \Rightarrow \text{professor}(p)$ $\text{tempAdvisedBy}(p, s) \Rightarrow \text{professor}(p)$ $\text{advisedBy}(p, s) \Rightarrow \text{student}(s)$ $\text{tempAdvisedBy}(p, s) \Rightarrow \text{student}(s)$ $\text{professor}(p) \wedge \text{hasPosition}(p, \text{Faculty}) \Rightarrow \text{taughtBy}(c, p, q)$
Cora	$\text{SameBib}(b1,b2) \wedge \text{SameBib}(b2,b3) \Rightarrow \text{SameBib}(b1,b3)$ $\text{SameTitle}(t1,t2) \wedge \text{SameTitle}(t2,t3) \Rightarrow \text{SameTitle}(t1,t3)$ $\text{Author}(bc1,a1) \wedge \text{Author}(bc2,a2) \wedge \text{SameAuthor}(a1,a2) \Rightarrow \text{SameBib}(bc1,bc2)$ $\text{HasWordVenue}(a1, +w) \wedge \text{HasWordVenue}(a2, +w) \Rightarrow \text{SameVenue}(a1, a2)$ $\text{Title}(bc1,t1) \wedge \text{Title}(bc2,t2) \wedge \text{SameTitle}(t1,t2) \Rightarrow \text{SameBib}(bc1,bc2)$
FB15K-237	$\text{position}(B, A) \wedge \text{position}(C, B) \Rightarrow \text{position}(C, A)$ $\text{ceremony}(B, A) \wedge \text{ceremony}(C, B) \Rightarrow \text{categoryOf}(C, A)$ $\text{film}(B, A) \wedge \text{film}(C, B) \Rightarrow \text{participant}(A, C)$ $\text{storyBy}(A, B) \Rightarrow \text{participant}(A, B)$ $\text{adjoins}(A, B) \wedge \text{country}(B, C) \Rightarrow \text{serviceLocation}(A, C)$