Enhancing Logical Expressiveness in Graph Neural Networks via Path-Neighbor Aggregation

Han Yu[1], **Xiaojuan Zhao**[2 *], **Aiping Li**[1*], **Kai Chen**[1], **Ziniu Liu**[1], **Zhichao Peng**[2]

[1]College of Computer Science and Technology, National University of Defense Technology, Changsha, China.
[2]Information School, Hunan University of Humanities, Science and Technology, Loudi, China.
{yuhan17,zhaoxiaojuan18,liaiping,chenkai_, liuzn_nudt}@nudt.edu.cn, zcpeng@tju.edu.cn

Abstract

Graph neural networks (GNNs) can effectively model structural information of graphs, making them widely used in knowledge graph (KG) reasoning. However, existing studies on the expressive power of GNNs mainly focuses on simple single-relation graphs, and there is still insufficient discussion on the power of GNN to express logical rules in KGs. How to enhance the logical expressive power of GNNs is still a key issue. Motivated by this, we propose Path-Neighbor enhanced GNN (PN-GNN), a method to enhance the logical expressive power of GNN by aggregating node-neighbor embeddings on the reasoning path. First, we analyze the logical expressive power of existing GNN-based methods and point out the shortcomings of the expressive power of these methods. Then, we theoretically investigate the logical expressive power of PN-GNN, showing that it not only has strictly stronger expressive power than C-GNN but also that its $(k+1)$-hop logical expressiveness is strictly superior to that of k-hop. Finally, we evaluate the logical expressive power of PN-GNN on eight synthetic datasets and two real-world datasets. Both theoretical analysis and extensive experiments confirm that PN-GNN enhances the expressive power of logical rules without compromising generalization, as evidenced by its competitive performance in KG reasoning tasks.

Extended version — https://arxiv.org/abs/2511.07994

Introduction

A Knowledge Graph (KG) organizes and represents knowledge in a structured form. In a KG, the nodes represent entities (such as people, places, or concepts), while the edges capture relationships between entities (such as "belongs to" or "contains"). KG reasoning typically requires models to understand and infer complex relations within graph structures. In recent years, graph neural networks (GNNs) (Wu et al. 2020; Velickovic et al. 2018; Hamilton, Ying, and Leskovec 2017; Xu et al. 2019) have shown great potential in KG reasoning tasks due to their powerful ability to represent graph-structured data. GNNs can automatically learn intricate relations between nodes from data and effectively capture both global and local structural information via message passing mechanisms, thereby substantially enhancing reasoning capabilities.

With the widespread application of GNNs in KG reasoning, their expressive power is crucial and can be evaluated from two aspects. One is discrimination, that is, the ability to distinguish non-isomorphic graphs, which is closely related to the strength of the Weisfeiler-Leman test. Another is to directly characterize what specific function classes a model can approximate, that is, the ability to learn specific logical rule structures. In KG reasoning tasks, if GNN cannot learn certain key rule structures, its performance may be severely limited. As shown in Figure 1, to reason the triple $(Curry, Team, ?)$. Relying solely on either Path1 or Path2 is not sufficient to correctly infer the answer. Path1 only shows that Curry's family members are fans of G.S.W., but does not indicate whether Curry is an NBA player or which team he belongs to. Path2 shows that Curry is an NBA player, but not which specific team he plays for. Path2 and Path2' share the same relation path ($Play\_with \rightarrow NBA\_player \rightarrow Team$), and Path2' incorrectly links Curry to the Lakers. Only when both Path1 and Path2 are simultaneously satisfied can we infer the conclusion of ($Curry$, $Team$, $G.S.W$). Therefore, if the model's expressive power is insufficient to capture and reason about these relations, it may fail to make correct predictions in reasoning tasks. This not only limits the model's reasoning ability but also leads to poor performance in real scenarios. The limited expressive power of key rule structures significantly affects the model's effectiveness and applicability. Thus, this motivates developing more powerful GNN models by enhancing the expressiveness of logical rules.

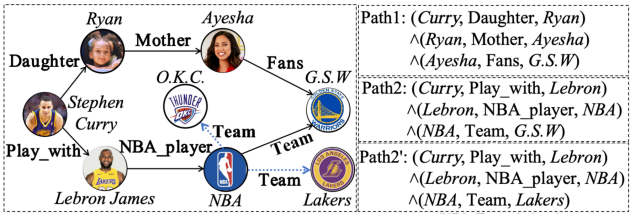


Figure 1: Schematic diagram of knowledge graph reasoning.

From the perspective of the logical expression abil-

* Corresponding authors.

ity of GNN, relational aggregation methods such as R-GCN (Schlichtkrull et al. 2018) and CompGCN (Vashishth et al. 2020), cannot perform rule learning well. The GNN method that dynamically adjusts based on conditions, such as NBFNet (Zhu et al. 2021), RED-GNN (Zhang and Yao 2022), and A*Net (Zhu et al. 2023), has been proven to have strong logical expressiveness (Qiu et al. 2024; Huang et al. 2023) and can learn basic logical relationships, such as unary prediction, negation, conjunction, disjunction, etc. Based on the Relational Weisfeiler-Leman (R-WL) (Barceló et al. 2022) test, the limitations of the logical rule learning ability of existing relational GNN (R-GNN) models were clarified, and it was proved that these models cannot capture higher-order logical rules (Huang et al. 2023). In other words, it was theoretically proved that the rule expression ability of conditional GNN (C-GNN) is strictly more powerful than that of R-GNN method. Although the labeling trick can enhance the rule learning ability of C-GNN in KG link prediction by introducing specific labels to nodes, it still faces two limitations: on the one hand, it will reduce the generalization ability of the model; on the other hand, the design of adding constant labels makes it difficult to apply to KG reasoning tasks in inductive settings. For example, in Figure

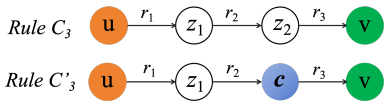


Figure 2: Rules C_3 and C'_3. Here, z_1, z_2 are variables and c is a constant.

2, for the rule $C_3(u, v) := \exists\, z_1, z_2(r_1(u, z_1) \wedge r_2(z_1, z_2) \wedge r_3(z_2, v))$, when z_1 and z_2 in the rule are variables, it can match any element that satisfies the variable constraint; but once the variable is fixed to a constant c, that is, it becomes $C'_3(u, v) := \exists\, z_1(r_1(u, z_1) \wedge r_2(z_1, c) \wedge r_3(c, v))$, only the specific elements corresponding to c can satisfy the rule, so the scope of application of the rule is limited, and the coverage (Quinlan 1990; Muggleton and de Raedt 1994) naturally becomes smaller, thereby weakening the generalization ability of the model.

In this paper, we propose **P**ath-**N**eighbor enhanced **GNN** (**PN-GNN**), a method to enhance the logical expressive power of GNN. PN-GNN addresses the labeling trick's limitations while preserving the model's generalization. In the conditional message passing scenario, PN-GNN leverages neighboring nodes along the paths between the source node and each target node to enrich the model's structural information, thereby enhancing the logical expressiveness. PN-GNN not only improves logical reasoning capabilities but also preserves generalization ability, making it suitable for inductive settings. In summary, our key contributions are:

- We propose a method called Path-Neighbor Enhanced GNN (PN-GNN) that significantly improves the logical expressive power without compromising the generalization ability, and PN-GNN can be effectively applied in inductive settings.
- We theoretically analyze the logical expressive power of PN-GNN and demonstrate that it surpasses that of GNNs based on conditional message passing.
- We instantiate the proposed theoretical model PN-GNN and build a KG reasoning model based on PN-GNN. The effectiveness of PN-GNN is verified by constructing synthetic datasets and real-world datasets. It outperforms the benchmark dataset in all indicators.

Related Work

Expressivity of Graph Neural Networks

The expressive power of GNNs is often analyzed through graph isomorphism testing. Vanilla GNNs have been shown to match the 1-WL test (Xu et al. 2019), with extensions to KGs (Barceló et al. 2022). To enhance expressivity, two main directions have emerged: designing stronger WL variants (Morris et al. 2019; Otto 2023; Barceló et al. 2022), and applying initialization techniques such as node labeling or identity features (Abboud et al. 2021; You et al. 2021; Sato, Yamada, and Kashima 2021). In link prediction, the structural expressivity of GNN alone has been found to be insufficient (Srinivasan and Ribeiro 2020), and the incorporation of labeling enables GNNs to learn structural information, although the existing results are primarily limited to single relational graphs (Zhang et al. 2021). Beyond WL-based approaches, GNNs have been shown to model first-order logic (FOL) with two variables (FOC_2), and their expressivity in node classification can be described using the counting extension of modal logic (Barceló et al. 2020; Cai, Fürer, and Immerman 1992). This has been extended to KGs, linking logical and WL-based expressivity and identifying the rule structures that GNNs can learn (Huang et al. 2023). Inspired by Teru, Denis, and Hamilton (2020), further improvements combine conditional message passing with labeling to enhance the learning of logical rules in KG reasoning.

Knowledge Graph Reasoning Based on GNN

GNN-based KG reasoning can be categorized into R-GNN and C-GNN (Huang et al. 2023). R-GNN methods like R-GCN and CompGCN perform relation-aware message passing to update entity embeddings, which are then used to score candidate links between entities.

C-GNN methods generate query-aware representations by marking the head entity and propagating messages accordingly, and can be categorized into subgraph-based and progressive aggregation methods. Subgraph-based models, such as GraIL (Teru, Denis, and Hamilton 2020) and CoMPILE (Mai et al. 2021), extract local neighborhoods between entity pairs and apply GNNs to propagate messages within subgraphs. Progressive aggregation methods, such as NBFNet (Zhu et al. 2021), RED-GNN (Zhang and Yao 2022), and A*Net (Zhu et al. 2023), further enhance reasoning capabilities. NBFNet incorporates the Bellman-Ford (Goldberg and Radzik 1993) to propagate messages along multi-hop paths. RED-GNN determines the next hop based on the current node, the query relation, and the next relation embedding. A*Net introduces A* search to enhance inference efficiently.

Background

Knowledge Graphs and Binary Predicate

A knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{E}$ represents the set of entities (nodes), $\mathcal{R}$ represents the set of relations (edges). $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ represents set of triplets and each triplet can be represented as (s, r, o), $s, o \in \mathcal{E}$, $r \in \mathcal{R}$. A constant ($c \in \mathcal{E}$) is a unique identifier for an entity in KG, while a variable ($z_i \in \mathcal{E}$) represents any entity in KG. The binary predicate $r_j(s, o)$, where r_j ($r_j \in \mathcal{R}$) represents a certain relation or attribute.

Conditional Graph Neural Networks

Graph neural networks based on conditional message passing mechanisms (Zhu et al. 2021; Zhang and Yao 2022) have a common message passing mechanism. Given a triple query $(u, q, ?)$, where u is the head entity and q is the query relation. Conditional Graph Neural Networks (C-GNN) first mark the head entity u with the query relation q to distinguish it from all other nodes in the KG. Then, it iteratively calculates and updates the representations of entity pairs $\mathbf{h}_{v|u,q}$ of all entities in the KG that can reach the tail v from the head u:

$$\begin{aligned}\mathbf{h}^{(0)}_{v|u,q} &= \text{INIT}(u, v, q) \\ \mathbf{h}^{(t+1)}_{v|u,q} &= \underset{\substack{w \in N_r(v) \\ r \in R}}{\text{UPDATE}}\left(\mathbf{h}^{(t)}_{v|u,q}, \text{AGG}\left(\{\{\text{MSG}_r(\mathbf{h}^{(t)}_{w|u,q}, \mathbf{z_q})\}\}\right)\right)\end{aligned} \tag{1}$$

Here, INIT denotes the initialization function, UPDATE represents the update function, $N_r(v)$ denotes the set of neighbors connected to v via relation r, $\mathbf{z_q}$ denotes the embedding of the query relation. AGG is the aggregation function, while MSG refers to the message passing function. The notation $\{\{\cdot\}\}$ represents a multi-set. C-GNN obtains the pairwise-entity representations $\mathbf{h}^{(L)}_{v|u,q}$ of the tail entity v, conditioned on the head entity u and the query relation q, through an L-layer message passing process. Finally, C-GNN predicts the target based on the tail representation:

$$score(u, q, v) = \sigma(\text{MLP}(\mathbf{h}^{(L)}_{v|u,q})) \tag{2}$$

where $\text{MLP}(\cdot)$ represents a trainable multi-layer perceptron.

Logical Expressive Power of Conditional GNN

The logical expressivity of a GNN refers to the set of logical formulas it can learn (Qiu et al. 2024). C-GNN demonstrates superior logical expressive power compared to R-GNN, leading to enhanced performance (Huang et al. 2023; Qiu et al. 2024). We use the counting extension of modal logic (CML), which extends the counting expressiveness of graded modal logic, to describe FOL properties in KG. CML consists of propositional constants (always true $\top$, always false $\bot$), unary logical predicates $P_i(x)$, and recursive rules. Given FOL $\varphi(x, y)$, $\psi(x, y)$, positive integer variables $N \geq 1$ and $r \in R$, according to the recursive rules, $\neg\varphi(x, y)$, $\varphi(x, y) \wedge \psi(x, y)$, and $\exists^{N \geq n} z(\varphi(x, z) \wedge r(z, y))$ are also in CML. Based on the CML in KG, the logical rule expressive power of C-GNN can be expressed as follows:

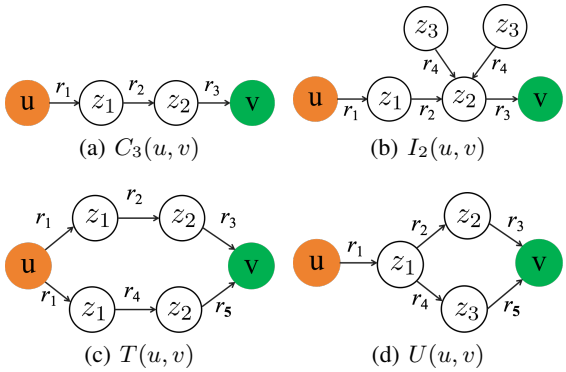


Figure 3: The rule structure that C-GNN can learn, taking 3-hop as an example.

Theorem 1. *C-GNN can learn every CML-based formula $\varphi(x)$ by leveraging its entity representations.*

Theorem 2. *A formula $\varphi(x)$ is learned by C-GNN if it can be expressed as a formula in CML.*

Theorem 1 shows that C-GNN has the ability to learn all CML formulas, implying that its logical expressive power of C-GNN at least covers the set of CML formulas. Conversely, Theorem 2 restricts the expressive power of C-GNN, by showing that it can only learn formulas that can be converted into CML formulas. Together, these two theorems characterize the logical expressive power of C-GNN, which is equivalent to that of CML formulas. See the appendix of the extended version for the complete proofs.

In KGs, CML can represent several common logical rule structures, as shown in Figure 3. Chain-like rules C_3, as illustrated in Figure 3(a), represent a commonly used rule structure in KG reasoning methods that rely on paths or rules (Qiu et al. 2024; Huang et al. 2023). The corresponding rule formula is expressed as follows:

$$C_3(u, v) := \exists\, z_1, z_2(r_1(u, z_1) \wedge r_2(z_1, z_2) \wedge r_3(z_2, v)). \tag{3}$$

Since C_3 can be described by CML based on recursion, C-GNN can learn the chain logic rule structure C_3.

The second rule structure I_N adds one or more relations $r \in \mathcal{R}$ that need to be satisfied to the chain logic rule, as shown in Figure 3(b), the case for $N = 2$ is illustrated and denoted as I_2. In the KG, I_N represents the scenario in which the rule is valid only if specific conditions are satisfied on certain entities. I_N can be described by CML, so it can be learned by C-GNN.

$$\begin{aligned}I_N(u, v) := \exists\, z_1, z_2(r_1(u, z_1) \wedge r_2(z_1, z_2)) \\ \wedge \{\exists^{\geq N} z_3(r_4(z_3, z_2))\} \wedge r_3(z_2, v).\end{aligned} \tag{4}$$

The third rule structure T, as shown in Figure 3(c), the upper and lower parts of the structure can be regarded as chain rules respectively, which can be described by CML. According to the recursive rule of CML, we can merge these two parts, so T can also be learned by C-GNN.

$$\begin{aligned}T(u, v) := \exists\, z_1, z_2(r_1(u, z_1) \wedge r_2(z_1, z_2) \wedge r_3(z_2, v)) \\ \wedge \exists\, z_1, z_2(r_1(u, z_1) \wedge r_4(z_1, z_2) \wedge r_5(z_2, v)).\end{aligned} \tag{5}$$

When the relations r_2 and r_4 are outgoing edges of the same node, we refer to this modified structure as *U*, as shown in Figure 3(d) and the corresponding rule formula is given as follows:

$$U(u,v) := \exists\, z_1 r_1(u,z_1) \wedge (\exists\, z_2, z_3(r_2(z_1,z_2) \wedge r_3(z_2,v) \wedge r_4(z_1,z_3) \wedge r_5(z_3,v)) \quad (6)$$

In this case, if we do not add an additional variable to indicate that the edges r_2 and r_4 come from the same node, the logical rule structures T and U have the same CML expression. That is, C-GNN cannot distinguish between U and T under CML, so we have the following corollary:

Corollary 3. *C-GNN cannot learn the logical rule structure U.*

This limitation highlights the need for additional mechanisms to enhance the model's expressive power. To address this issue, the labeling trick is introduced, a technique that augments the input representation to differentiate previously indistinguishable structures. In the following, we formally define the labeling trick and provide an example demonstrating how it enables C-GNN to overcome the limitation identified in Corollary 3.

Definition 4. ***(Labeling Trick)**. Given a rule structure R(x,y) and an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n\times k}$ for the graph $\mathcal{G}$, we assign a constant to each node whose out-degree is greater than one. Construct a labeled tensor $L^{(S)} \in \mathbb{R}^{n\times n\times d}$ to distinguish target nodes from non-target nodes. Concatenate $L^{(S)}$ and $\boldsymbol{A}$ to get a new graph representation $A^{(S)} = [\boldsymbol{A}\ L^{(S)}] \in \mathbb{R}^{n\times n\times(k+d)}$, which allows C-GNN to learn the rules to distinguish them.*

For example, in Figure 3, U and T cannot be effectively distinguished. When a constant c is added to z_1 and a specific initial representation is given, the C-GNN can learn the structure of the rules $U'(u,v) := r_1(u,c) \wedge (\exists\, z_2, z_3(r_2(c,z_2) \wedge r_3(z_2,v) \wedge r_4(c,z_3) \wedge r_5(z_3,v))$. EL-GNN (Qiu et al. 2024) improves the rule learning ability by using the labeling trick based on C-GNN. However, when nodes are given unique labels, the model may over-rely on this "artificial" information during training, thereby ignoring other potential topological features or relationship clues, resulting in a decrease in the model's generalization ability.

Next, we introduce the concept of "coverage" (Quinlan 1990; Muggleton and de Raedt 1994) to assess the logical expressive power of these rules and explain why assigning constants to variables reduces coverage.

Definition 5. ***(Coverage)**. Given a knowledge graph $\mathcal{G}$, for the rule $\varphi(x,y) := \exists z_1, z_2, \cdots (r_1(x,z_1) \wedge r_2(z_1,z_2) \wedge \cdots \wedge r_m(z_k,y))$, $\{z_1, z_2, \cdots, z_k\} \in \mathcal{E}$, we define coverage $Cov(\varphi)$ as the set of all pairs of entities (x,y) that can satisfy the rule $\varphi(x,y)$ in $\mathcal{G}$. Formally,*

$$Cov(\varphi) = \{(x,y) \in \mathcal{E}\times\mathcal{E} \mid \mathcal{G} \models \varphi(x,y)\}. \quad (7)$$

Where $\mathcal{G} \models \varphi(x,y)$ means that there exist entities $\{z_1, z_2, \cdots\}$ in $\mathcal{G}$ that satisfy all the relational conditions in $\varphi(x,y)$. In other words, $Cov(\varphi)$ measures how many valid pairwise entities (x,y) in $\mathcal{G}$ are "covered" by the rule φ. The greater the coverage, the stronger the generalizability of the rule.

Regardless of the structure of the rule, as long as a variable is replaced by a constant in the rule, there will be a coverage reduction effect. That is to say, for any structure, if the original rule $\varphi(x,y)$ contains one or more variables, these variables determine which elements the rule can adapt to (larger coverage). After replacing the variable with a constant to obtain a new rule $\varphi'(x,y)$, it can only match the specific elements corresponding to the constant (smaller coverage). Therefore, it can be summarized as follows:

Corollary 6. ***(Adding Constants Restricts Coverage)** If we start from the original rules $\varphi(x,y,\cdots)$ and replace at least one of them with a constant c to obtain new rules $\varphi'(x,c,\cdots)$, the coverage of the new rules is less than that of the original rules. Formally expressed as:*

$$Cov\Big(\varphi'(x,c,\cdots)\Big) \subseteq Cov\Big(\varphi(x,y,\cdots)\Big). \quad (8)$$

Rules with constants (labels) become more specific, resulting in a smaller coverage and reduced generalization ability.

According to corollary 6, we can conclude that $Cov(EL\text{-}GNN) \subseteq Cov(C\text{-}GNN)$

Path Neighbors Enhanced GNN

Method

To improve the logical expressive power of C-GNN without weakening its generalization ability, we propose PN-GNN, which uses a flexible plug-and-play approach without adding constant labels. PN-GNN iteratively updates L steps according to formula (1) to obtain the pairwise entity representation $\mathbf{h}^{(L)}_{v|u,q}$, and aggregates the representations of neighbor nodes along the path passing through the head and tail entities in the KG, leveraging these neighbors to enhance the structural expressive power of the GNN:

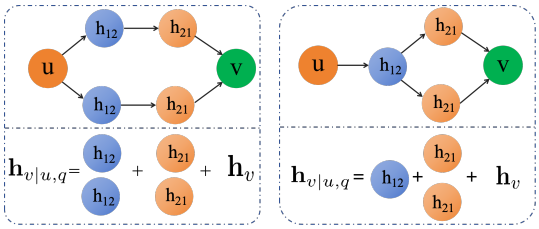
$$\mathbf{h}_{ij} = \text{POOL}(\{\mathbf{h}^{(L)}_{w|u,q} | w \in P_{uv}, d(u,w) = i, d(w,v) = j\}) \quad (9)$$

The aggregated nodes are represented by $\mathbf{h}_{ij}$. Among them, i represents the distance from the head entity to the node w to be aggregated, and j represents the distance from the node w to the tail entity v. P_{uv} represents all paths with u as the head source node and v as the target node. $d(\cdot,\cdot)$ represents the distance between nodes. POOL is a pooling function that aggregates the representations of neighborhoods along the path with the same distance. It can be set to any commonly used function in GNN, such as *max*, *min*, *mean*.

Using the aggregated representations $\mathbf{h}_{ij}$ of the neighbors of nodes on the path, PN-GNN only needs to fuse the representation $\mathbf{h}^{(L)}_{v|u,q}$ learned by C-GNN:

$$\begin{aligned} \mathbf{h}_d &= \bigoplus_{i+j\leqslant d} \text{MLP}(\mathbf{h}_{ij}) \\ \mathbf{h}_{v|u,q} &= \mathbf{h}_d \odot \mathbf{h}^{(L)}_{v|u,q} \end{aligned} \quad (10)$$

Formula (10) computes the node representation $\mathbf{h}_{v|u,q}$ of each target tail entity v. MLP$(\cdot)$ is a multi-layer perceptron. The operator $\bigoplus$ aggregates all neighbors in Equation (9), using operations like addition or concatenation. Similarly, $\odot$



(a) T (b) U

Figure 4: Example of the aggregation process

Algorithm 1: PN-GNN

Require: Source node u, query relation q, KG $\mathbb{G}$
Ensure: Node pair representation $\mathbf{h}_{v|u,q}$, $v \in V$

for $v \in V$ **do**
  $\mathbf{h}^{(0)}_{v|u,q} \leftarrow \text{INIT}(u, v, q)$
end for
for $t \in [1, T]$ **do**
  for $v \in V$ **do**
    $\mathbf{h}^{(t)}_{v|u,q} \leftarrow \text{C-GNN}(u, q, KG)$
  end for
end for
$\mathbf{h}_{ij} = \text{POOL}(\{\mathbf{h}^{(L)}_{w|u,q} | w \in P_{uv}, d(u,w) = i, d(w,v) = j\})$
$\mathbf{h}_d = \bigoplus_{i,j \leqslant d} \text{MLP}(\mathbf{h}_{ij})$
$\mathbf{h}_{v|u,q} = \mathbf{h}_d \odot \mathbf{h}^{(L)}_{v|u,q}$
return $\mathbf{h}_{v|u,q}$

fuses the C-GNN pairwise representation with the neighbor vector, also via addition or concatenation. To reduce computational overhead while ensuring the expressive power of PN-GNN, we select 1-hop and 2-hop neighbors along the path. Here, $\mathbf{h}_{11}$ denotes the aggregation of neighbors with 1-hop, which means that neighbor node w is 1-hop away from both the source node u and the target node v. The 2-hop neighborhoods include $\mathbf{h}_{12}$ and $\mathbf{h}_{21}$, which indicate that w is 1-hop from u and 2-hop from v, or 2-hop from u and 1-hop from v, respectively. As shown in Figure 4, Equation (10) now only involves path neighborhoods within two hops.

$$\begin{aligned} \mathbf{h}_d &= \text{MLP}_{11}(\mathbf{h}_{11}) \odot \text{MLP}_{12}(\mathbf{h}_{12}) \odot \text{MLP}_{21}(\mathbf{h}_{21}) \\ \mathbf{h}_{v|u,q} &= \mathbf{h}_d \odot \mathbf{h}^{(L)}_{v|u,q} \end{aligned} \tag{11}$$

The iterative update process of the pairwise-entity representation $\mathbf{h}_{v|u,q}$ can be expressed by algorithm 1.

After L iterations, the model obtains the pairwise-entity representation $\mathbf{h}_{v|u,q}$ for every entity v in the KG. Similar to the C-GNN, PN-GNN directly uses this representation to compute the conditional probability of the tail entity v.

$$p(v|u,q) = \sigma(\text{MLP}(\mathbf{h}_{v|u,q})) \tag{12}$$

Here, $\sigma(\cdot)$ denotes the sigmoid function. PN-GNN is trained by minimizing the negative log-likelihood of positive and negative triples:

$$L_{KG} = -logp(u,q,v) - \sum_{i=1}^{n} \frac{1}{n} log(1 - p(u_i^{'}, q, v_i^{'})) \tag{13}$$

Here, $(u_i^{'}, q, v_i^{'})$ denotes a negative sample. Negative samples are constructed under the Partial Completeness Assumption (PCA) (Bordes et al. 2013; Sun et al. 2019) by randomly replacing the head (or tail) entity with an incorrect one from the knowledge graph, and n denotes the number of negative samples. In order to balance logical expressive power and efficiency, we use a 2-hop PN-GNN.

Logical Expressive Power of PN-GNN

The matching of logical rule structures in KG is regarded as a binary logical classification task. In order to prove that the logical expressive of PN-GNN is more powerful than that of C-GNN, it is first proved that PN-GNN can learn all logical rule structures that C-GNN can learn. According to corollary 3, it is necessary to first prove that PN-GNN can learn the logical rule structure in CML in KG.

Let $\varphi(x)$ be a FOL expression in the CML of the KG. φ can be decomposed into a series of sub-expressions, namely $sub(\varphi) = (\varphi_1, \varphi_2, ..., \varphi_L)$. If φ_k is a sub-expression of φ_l, then $k \leqslant l$, and finally $\varphi = \varphi_L$. $\mathbf{h}^{(t)}_v \in \mathbb{R}^L$ is the representation learned by GNN through message passing, and L is the total number of iterations. The iterative calculation of PN-GNN can be simplified to the following formula:

$$\mathbf{h}^{(t)}_v = \sigma(\mathbf{h}^{(t-1)}_v \mathbf{C} + \sum_{j=1}^{r} \sum_{u \in v} \mathbf{h}^{(t-1)}_u \mathbf{A}_r + \mathbf{b}) \tag{14}$$

$$\mathbf{h}_v = \sigma(\mathbf{h}^{(L)}_v + \sum_{i+j=|d|} (\mathbf{h}^{(L)}_{ij} \mathbf{A}_{pn} + \mathbf{b}_{pn})) \tag{15}$$

The formula (14) represents the process of C-GNN iteratively calculating entity representation. Formula (15) represents the process of PN-GNN aggregating neighborhoods representations in the path, where $|d|$ represents the maximum length of the allowed path. For example, when $|d| = 2$, the neighborhoods with a distance of 1-hop from the head and tail entities are aggregated. When $|d| = 3$, the neighborhoods with a distance of 1-hop from the head entity and 2-hop from the tail entity and 2-hop from the head entity and 1-hop from the tail entity are aggregated. $\sigma = min(max(0, x), 1)$ is the ReLU activation function.

Based on formula (14) and (15), we can derive the following two lemmas:

Lemma 7. *If $\varphi(x)$ is a CML formula that can be learned by C-GNN, then PN-GNN can also learn $\varphi(x)$.*

Lemma 8. *PN-GNN can learn the logical rule structure U.*

Based on these lemmas, we show that PN-GNN can learn the logical rule structure U, while C-GNN cannot (see Corollary 3). This leads to the conclusion that PN-GNN has stronger logical expressiveness than C-GNN. The theorem is stated below (see Appendix for complete proofs):

Theorem 9. *PN-GNN has strictly more powerful logical expressiveness than C-GNN.*

The expressive power of PN-GNN depends on both its architecture and propagation range (number of hops). As the propagation range increases, its logical expressiveness improves. The following results formalize this relationship:

Lemma 10. *(k+1)-hop can represent all logical expressions that k-hop can learn.*

Lemma 11. *(k+1)-hop can learn logical rules that cannot be expressed by k-hop.*

Based on Lemmas 10 and 11, we obtain the following theorem (see Appendix for the complete proof):

Theorem 12. *For PN-GNN, the logical expressive power of (k+1)-hop is strictly more powerful than that of k-hop.*

Although increasing the number of hops enhances the model's logical expressive capability, it also reduces computational efficiency. Therefore, to balance expressiveness and efficiency, we set k = 2.

Experiments

Experiments Setting

Datasets To evaluate the logical expressive power of PN-GNN, we conducted experiments on two real-world datasets (WN18RR and FB15K237) in the transductive setting and their inductive variants (v1-v4), as well as six synthetic datasets provided by Qiu et al. (2024). We also constructed two additional synthetic datasets, T_{label} and U_{label}, to examine how labeling strategies affect generalization and ensure predictions rely on logical structure rather than specific labels. Details on dataset construction, statistics, hyperparameters, and evaluation metrics are provided in Appendix.

Baselines We compare PN-GNN with several baselines, including R-GNN-based methods, C-GNN-based methods, and labeling strategy-based EL-GNN. In addition, we use representation learning, path-based, and GNN-based methods as baselines in both inference and inductive settings. See Appendix for more details.

Experiments On Synthetic Datasets

Results on synthetic datasets are shown in Table 1. The R-GNN results are taken from Qiu et al. (2024), while the EL-GNN results are obtained by using NBFNet as the backbone and carefully searching for parameters on the validation using the Ray (Liaw et al. 2018). Through the analysis of the results, we can observe that:

(1)Theoretical analysis indicates that the logical expressive of C-GNN is strictly more powerful than that of R-GNN. Experimental results validate this conclusion: methods based on C-GNN, such as NBFNet, RED-GNN, EL-GNN, and the proposed PN-GNN, achieved 100% accuracy on the C_3, C_4, I_1, I_2, and T. In contrast, the performance of R-GNN confirms that it is nearly incapable of learning these logical rules. Notably, PN-GNN also achieved 100% accuracy on all datasets except U and the two datasets (T_{label} and U_{label}) that we constructed, demonstrating that its logical expressive power is comparable to that of C-GNN and that it can learn the logical rules in CML.

(2) EL-GNN and PN-GNN exhibit strictly more powerful logical expressive compared to C-GNN. Analysis indicates that C-GNN does not distinguish between U and T, making it impossible to learn the logical rule structures in U. In contrast, EL-GNN, leveraging a labeling strategy, achieved the best performance on the U dataset, with a 21.6% improvement over C-GNN. Similarly, PN-GNN also outperformed C-GNN, achieving a 15.8% improvement. These results demonstrate that PN-GNN enhances the logical expressive power of C-GNN.

(3) We demonstrate the impact of constant labels on EL-GNN using the T_{label} and U_{label}. The performance of EL-GNN significantly degrades on both datasets because it learns node representations from fixed labels during training. However, at test time the model encounters different constant labels, causing its learned rules to miss some cases and leading to degraded performance, indicating weakened generalization. In particular, for the T_{label}, where labels are not required, it is affected by the label vector during training, resulting in reliance on labels for prediction during testing, which leads to a significant drop in performance. In contrast, PN-GNN exhibits a much stronger performance: on the T_{label}, it performs on par with NBFNet and far outperforms EL-GNN, and on the U_{label}, it not only surpasses EL-GNN but also outperforms NBFNet. This shows that PN-GNN can effectively alleviate the negative impact of constant labels on generalization.

Method	C_3	C_4	I_1	I_2	T	U	T_{label}	U_{label}
R-GCN	1.6	3.1	4.4	2.4	6.7	1.4	-	-
CompGCN	1.6	2.1	5.3	3.9	6.7	2.7	-	-
NBFNet	100	100	100	100	100	54.1	60.0	56.8
EL-GNN	100	100	100	100	100	75.7	22.0	59.5
PN-GNN	100	100	100	100	100	69.9	60.0	68.9

Table 1: Results on synthetic datasets (Hits@1).

Transductive Results

The results of the transductive setting are shown in Table 2. Across all evaluation metrics, C-GNN based methods surpass those based on representation learning, paths, and R-GNNs. This outcome aligns with C-GNN theory, showing that C-GNN can learn and leverage logical rule structures. They also indicate that, although GNNs capture more graph structures than path-based methods, the ability to learn logical rule structures is crucial for KG reasoning. PN-GNN and EL-GNN further improve upon C-GNN, demonstrating that enhancing the representation of logical rule structures leads to better inference performance. On the real-world dataset FB15K237, which contains U and T structures, EL-GNN and PN-GNN outperform C-GNN. Notably, the node out-degree d is a sensitive hyperparameter for EL-GNN and has a great impact on performance (Qiu et al. 2024), whereas PN-GNN does not require this parameter.

Method	FB15K237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	0.294	-	-	46.5	0.226	-	40.3	53.2
RotatE	0.338	24.1	37.5	53.3	0.476	42.8	49.2	57.1
HAKE	0.341	24.3	37.5	53.5	0.496	45.1	51.3	58.2
RotH	0.344	24.6	38.0	53.5	0.495	44.9	51.4	58.6
ComplE+RP	0.388	29.8	42.5	56.8	0.488	44.3	50.5	57.8
ConE	0.345	24.7	38.1	54.0	0.496	45.3	51.5	57.9
NeuralLP	0.240	-	-	36.2	0.435	37.1	43.4	56.6
DRUM	0.343	25.5	37.8	51.6	0.486	42.5	51.3	58.6
R-GCN	0.273	18.2	30.3	45.6	0.402	34.5	43.7	49.4
CompGCN	0.355	26.4	39.0	53.5	0.479	44.3	49.4	54.6
NBFNet	0.415	32.1	45.4	59.9	0.551	49.7	57.3	66.6
RED-GNN	0.374	28.3	-	55.8	0.533	48.5	-	62.4
A*Net	0.411	32.1	45.3	58.6	0.549	49.5	57.3	65.9
EL-GNN	**0.421**	**33.2**	45.9	59.8	**0.555**	**49.9**	**57.8**	66.4
PN-GNN	**0.423**	**33.1**	**46.5**	**60.2**	**0.555**	**49.9**	**57.9**	**66.9**

Table 2: Results for transductive setting on real-world datasets. The best results are shown in bold.

Method	FB15K237				WN18RR			
	v1	v2	v3	v4	v1	v2	v3	v4
NeuralLP	52.9	58.9	52.9	55.9	74.4	68.9	46.3	67.1
DRUM	52.9	58.7	52.9	55.9	74.4	68.9	46.2	67.1
RuleN	49.8	77.8	87.7	85.6	80.9	78.2	53.4	71.6
GraIL	64.3	81.8	82.8	89.3	82.4	78.7	58.4	73.4
CoMPILE	67.6	82.9	84.7	87.4	83.6	79.8	60.7	75.5
RED-GNN	65.7	82.9	81.1	89.9	86.7	83.1	70.8	78.6
NBFNet	83.5	94.9	95.2	96.1	94.8	90.1	89.3	89.1
PN-GNN	**85.1**	**95.8**	**96.2**	**96.5**	**95.2**	**90.8**	**89.8**	**89.3**

Table 3: Results for inductive setting (evaluated with Hits@10). The best results are shown in bold.

Method	T	U	FB15K237v1	WN18RRv1
NBFNet	100	54.1	17.1	60.1
PN-GNN	100	69.9	22.1	62.5
PN-GNN$_{11}$	100	48.7	20.2	62.8
PN-GNN$_{12-21}$	100	69.9	18.5	62.5

Table 4: Ablation study results on dataset using the Hits@1.

Inductive Results

Table 3 presents the experimental results under the inductive setting. EL-GNN relies on entity representations to annotate entities, rendering it incapable of handling unseen entities in the inductive setting. In the inductive setting, models must infer unseen entities from existing relationships, directly reflecting their logical expressiveness. Experimental results show that C-GNN-based methods, such as NBFNet and RED-GNN, outperform path-based approaches. Building on NBFNet, PN-GNN achieves further improvements across datasets, verifying the effectiveness of its enhanced logical expressiveness in inductive scenarios. We also conduct experiments to test the effectiveness of PN-GNN and provide detailed comparisons in Appendix.

Ablation Study

To examine the effect of neighborhood hops on PN-GNN's performance, we conducted ablation studies on T, U, FB15K237 v1, and WN18RR v1, comparing PN-GNN$_{11}$ (using only 1-hop) and PN-GNN$_{12-21}$ (using only 2-hop). Results are shown in Table 4. C-GNN is sufficient for the T dataset, with all models achieving 100% accuracy. For U, PN-GNN$_{12-21}$ outperforms NBFNet by 15.8%, consistent with Lemma 8, as distinguishing the logical rules in U mainly relies on 3-hop neighborhoods. On FB15K237 v1, PN-GNN and its variants all outperform NBFNet, with the best improvement reaching 5%, demonstrating the benefits of improved logical expressiveness in complex real-world datasets. On WN18RR v1, PN-GNN$_{11}$ achieves the highest performance, outperforming NBFNet by 2.7%. This suggests that for datasets with fewer relations and predominantly short-path logical structures, leveraging only 1-hop neighborhoods provides more relevant information than considering longer paths, which may introduce noise.

Conclusion

In this work, we analyze the logical expressiveness of GNNs in KG reasoning. We highlight the limitations of the labeling trick—commonly used to boost link prediction—which can hinder model generalization. To address this, we propose Path-Neighbor enhanced GNN (PN-GNN), which improves the logical expressive power of GNN by aggregating node-neighbor embeddings on the reasoning paths and avoids the generalization issues of the labeling trick, while still supporting inductive settings. PN-GNN faces increased computational costs in multi-hop scenarios as resource demands grow with hops.

Acknowledgements

This work is particularly supported by the National Key Research and Development Program of China (No. 2022YFB3104103), the National Natural Science Foundation of China (No. 62302507), the Provincial Natural Science Foundation of Hunan (No. 2024JJ7249, 2025JJ70302), and the National First-Class Undergraduate Major (Network Engineering) Fund.

References

Abboud, R.; Ceylan, İ. İ.; Grohe, M.; and Lukasiewicz, T. 2021. The Surprising Power of Graph Neural Networks with Random Node Initialization. In *IJCAI 2021*, 2112–2118.

Barceló, P.; Galkin, M.; Morris, C.; and Orth, M. A. R. 2022. Weisfeiler and Leman Go Relational. In *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event*, volume 198 of *Proceedings of Machine Learning Research*, 46. PMLR.

Barceló, P.; Kostylev, E. V.; Monet, M.; Pérez, J.; Reutter, J.; and Silva, J.-P. 2020. The logical expressiveness of graph neural networks. In *ICLR 2020*.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. *NeurIPS*, 26.

Cai, J.-Y.; Fürer, M.; and Immerman, N. 1992. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4): 389–410.

Goldberg, A. V.; and Radzik, T. 1993. A heuristic improvement of the Bellman-Ford algorithm. *Applied Mathematics Letters*, 6(3): 3–6.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS 2017*, 1024–1034.

Huang, X.; Romero, M.; Ceylan, İ. İ.; and Barceló, P. 2023. A Theory of Link Prediction via Relational Weisfeiler-Leman on Knowledge Graphs. In *NeurIPS*.

Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J. E.; and Stoica, I. 2018. Tune: A Research Platform for Distributed Model Selection and Training. arXiv:1807.05118.

Mai, S.; Zheng, S.; Yang, Y.; and Hu, H. 2021. Communicative message passing for inductive relation reasoning. In *AAAI 2021*, volume 35, 4294–4302.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *AAAI 2019*, 4602–4609. AAAI Press.

Muggleton, S.; and de Raedt, L. 1994. Inductive Logic Programming: Theory and methods. *The Journal of Logic Programming*, 19-20: 629–679. Special Issue: Ten Years of Logic Programming.

Otto, M. 2023. Graded modal logic and counting bisimulation. arXiv:1910.00039.

Qiu, H.; Zhang, Y.; Li, Y.; and Yao, Q. 2024. Understanding Expressivity of GNN in Rule Learning. In *ICLR 2024*.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning*, 5: 239–266.

Sato, R.; Yamada, M.; and Kashima, H. 2021. Random Features Strengthen Graph Neural Networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, 333–341. SIAM.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *ESWC 2018*, 593–607. Springer.

Srinivasan, B.; and Ribeiro, B. 2020. On the Equivalence between Positional Node Embeddings and Structural Graph Representations. In *ICLR 2020*.

Sun, Z.; Deng, Z.; Nie, J.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR 2019*.

Teru, K.; Denis, E.; and Hamilton, W. 2020. Inductive relation prediction by subgraph reasoning. In *ICML 2020*, 9448–9457. PMLR.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. P. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR 2020*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

You, J.; Gomes-Selman, J. M.; Ying, R.; and Leskovec, J. 2021. Identity-aware graph neural networks. In *AAAI 2021*, volume 35, 10737–10745.

Zhang, M.; Li, P.; Xia, Y.; Wang, K.; and Jin, L. 2021. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. In *NeurIPS*, 9061–9073.

Zhang, Y.; and Yao, Q. 2022. Knowledge graph reasoning with relational digraph. In *Proceedings of the ACM web conference 2022*, 912–924.

Zhu, Z.; Yuan, X.; Galkin, M.; Xhonneux, L. A. C.; Zhang, M.; Gazeau, M.; and Tang, J. 2023. A*Net: A Scalable Path-based Reasoning Approach for Knowledge Graphs. In *NeurIPS*.

Zhu, Z.; Zhang, Z.; Xhonneux, L.-P.; and Tang, J. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *NeurIPS 2021*, 34: 29476–29490.