LogiX-Me: Logic-based Self-Explainable Message Passing GNN

Simone Palumbo[1], Alessio Ragno[2], Roberto Capobianco[3], Marc Plantevit[2] and Céline Robardet[4]

[1]Sapienza University of Rome
[2]EPITA Research Laboratory (LRE)
[3]Sony AI
[4]INSA Lyon, CNRS, LIRIS UMR 5205

Abstract

Graph Neural Networks (GNNs) have achieved state-of-the-art performance in learning from relational data, yet their decision-making processes remain largely opaque. While recent logic-based self-explainable approaches like LogiX-GIN have advanced transparency by grounding logical rules in the model's architecture, they are fundamentally restricted to unary predicates, limiting their ability to capture edge features and complex relational dependencies. To overcome this limitation, we present LogiX-Me, a novel self-explainable GNN grounded in Polyadic Graded Modal Logic. Unlike its predecessor, LogiX-Me employs a hierarchical dual-layer design that learns ternary predicates to explicitly model edge attributes and their interactions with node features. Extensive experiments demonstrate that LogiX-Me achieves predictive performance competitive with black-box baselines while significantly outperforming monadic logic-based models on edge-sensitive tasks.

Keywords

explainability, graph neural networks, logic

Declaration on Generative AI

During the preparation of this work, the author(s) used Perplexity and Grammarly in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

1. Introduction

Graph Neural Networks (GNNs) have emerged as powerful models for learning from relational data, capturing complex patterns in molecular structures, social networks, protein interactions, and knowledge graphs. Despite their success, GNNs inherit the black-box nature of deep learning, making their decision-making processes difficult to understand.

Explainable AI (XAI) seeks to make complex models more transparent through two main avenues: post-hoc and self-explainable approaches. Post-hoc methods analyze trained black-box models to highlight the most influential nodes or subgraphs, yet their explanations often fail to capture the model's true reasoning [1]. In contrast, self-explainable GNNs embed interpretability within their architectures, using strategies like prototypes [2], concepts [3], or information bottlenecks [4]. However, their focus on topology alone limits insight into the model's underlying logic. Logic-based models aim to overcome these limits by grounding explanations in formal reasoning. Early efforts, such as DC-GNNs [5], combine self-explainable GNNs with logic operators (e.g., LENs), introducing symbolic reasoning but still relying on purely topological representations. Building on this direction, LogiX-GIN [6] enables direct translation of its layers into Graded Modal Logic (GML) [7], marking a major advance

Published in the Proceedings of the Workshops of the EDBT/ICDT 2026 Joint Conference (March 24-27, 2026), Tampere, Finland
palumbo.1939378@studenti.uniroma1.it (S. Palumbo); alessio.ragno@epita.fr (A. Ragno); roberto.capobianco@sony.com (R. Capobianco); marc.plantevit@epita.fr (M. Plantevit); celine.robardet@liris.cnrs.fr (C. Robardet)

toward logical transparency. Yet, its logic remains restricted: it excludes edge features, supports only unary predicates, and captures reasoning at a single neighborhood scale.

We present LogiX-Me, a new self-explainable GNN that extends LogiX-GIN with a dual-layer design and a polyadic logical foundation. LogiX-Me uses two complementary transformation stages: the first operates before aggregation to learn edge-level rules, capturing binary $R(x, y)$ or ternary $R(x, y, e)$ predicates that describe relational properties between connected nodes; the second acts after aggregation to derive node-level rules, expressing neighborhood patterns such as "a node has k neighbors satisfying condition P." This hierarchical design mirrors human reasoning from local relations to global structures and naturally supports edge features within message passing. To formalize this expressiveness, we introduce Polyadic Graded Modal Logic (PGML), extending GML with polyadic predicates and counting quantifiers over nodes and edges, enabling statements like "there exist at least N neighbors y such that $R(x, y, e)$ holds." Overall, LogiX-Me is the first GNN to integrate polyadic logic-based reasoning directly into its architecture, enabling explicit, edge-aware, and human-intuitive explanations.

2. Background

This section establishes the mathematical framework linking GNNs, GML and the specific architectural choices in LogiX-GIN that enable faithful logical extraction.

GNNs operate on graph-structured data $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents edges, and $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{in}}}$ contains node feature vectors. For a node v, we denote its feature vector $\mathbf{x}_v$ and neighborhood $\mathcal{N}(v) = \{u \in \mathcal{V} : (u, v) \in \mathcal{E}\}$. GNNs process graphs via iterative message passing, updating node representations at each layer k:

$$\mathbf{h}_v^{(k)} = \varphi^{(k)} \left(\mathbf{h}_v^{(k-1)}, \mu^{(k)} \left(\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\} \right) \right), \tag{1}$$

where $\mu^{(k)}$ is a permutation-invariant aggregation function and $\varphi^{(k)}$ is a learnable update function. While general GNNs may use mean or max aggregation, LogiX-Me specifically employs sum aggregation ($\mu^{(k)} = \sum$) to enable counting-based reasoning. After K layers, $\mathbf{h}_v^{(K)}$ encodes the structure of the K-hop neighborhood.

GML offers a formal language that matches the expressivity of message passing by using counting constraints. We define GML formulas $\varphi(x)$ over binary edge relations $E(x, y)$ and unary node predicates $P_i(x)$ inductively:

$$\begin{aligned} \varphi(x) ::= & P(x) \mid \neg\psi(x) \mid \psi_1(x) \wedge \psi_2(x) \\ & \mid \exists^{\geq N} y \, (E(x, y) \wedge \theta(y)) \end{aligned} \tag{2}$$

The key operator is the *graded existential quantifier* $\exists^{\geq N}$, which asserts that "there exist at least N distinct neighbors y such that property $\theta(y)$ holds." The satisfaction relation is defined locally:

$$\begin{aligned} \mathcal{G}, v \models \exists^{\geq N} y \, (E(x, y) \wedge \theta(y)) \iff \\ |\{u \in \mathcal{N}(v) : \mathcal{G}, u \models \theta(y)\}| \geq N \end{aligned} \tag{3}$$

LogiX-GIN LogiX-GIN operationalizes this link by processing binary node features through three sequential stages at each layer: sum aggregation, Fourier binarization, and logical transformation.

First, the model aggregates the features of neighboring nodes using sum aggregation. This step produces integer-valued messages (counts) $\mathbf{a}_v^{(k)} \in \mathbb{N}^d$ representing how many neighbors satisfy specific properties:

$$\mathbf{a}_v^{(k)} = \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k-1)}. \tag{4}$$

Since the counts $\mathbf{a}_v^{(k)}$ need to be processed by the logic layer, which requires binary inputs, LogiX-GIN employs a *periodic* and *parametric Fourier step function* $\beta^{(k)}$ to map these counts to binary literals based

on learnable intervals:

$$\tilde{\mathbf{a}}_v^{(k)} = \beta^{(k)}(\mathbf{a}_v^{(k)}) = \text{clamp}\left(\frac{1}{2} \cdot \frac{\sum_{i=0}^{\tau} \frac{\sin((2i+1)(\mathbf{a}_v^{(k)} \odot \tilde{W}^{(k)} + \tilde{b}^{(k)}))}{2i+1}}{\sum_{i=0}^{\tau} \frac{\sin((2i+1)\frac{\pi}{2})}{2i+1}} + 1, 0, 1\right). \tag{5}$$

This function activates ($\tilde{\mathbf{a}}_v^{(k)}[j] = 1$) only when the neighbor count falls within specific learned intervals $I_{\beta^{(k)}}$:

$$\tilde{\mathbf{a}}_v^{(k)}[j] = 1 \iff \mathbf{a}_v^{(k)}[j] \in \bigcup_{n \in \mathbb{Z}} \left[\frac{2\pi n - \tilde{b}_j^{(k)}}{\tilde{W}_j^{(k)}}, \frac{(2\pi n + \pi) - \tilde{b}_j^{(k)}}{\tilde{W}_j^{(k)}}\right]. \tag{6}$$

This effectively creates literals representing graded properties like "the number of neighbors with property P is between 2 and 4."

Finally, the binarized count features $\tilde{\mathbf{a}}_v^{(k)}$ are processed by a Transparent Explainable Logic Layer (TELL) [8] to compute the node's updated binary representation $\mathbf{h}_v^{(k)}$. This layer computes activations based on a weighted sum and bias, identifying minimal subsets of active inputs sufficient to trigger an output:

$$\mathbf{h}_v^{(k)}[j] = 1 \iff \sum_{i \in S} \mathbf{W}[i, j] \cdot \tilde{\mathbf{a}}_v^{(k)}[i] > -\mathbf{b}[j]. \tag{7}$$

This structure allows for the extraction of complex rules in Disjunctive Normal Form (DNF), $E_j = \bigvee_{S \in \mathcal{S}_j} (\bigwedge_{i \in S} \tilde{a}_i)$, combining the interval-based counting literals into higher-order logical formulas.

3. Polyadic Graded Modal Logic (PGML)

While GML provides foundations for expressing node-level properties with counting, it is fundamentally limited to unary predicates and cannot express edge properties explicitly. Our work extends this framework through PGML, which augments GML with polyadic predicates.

Consider a molecular graph where we wish to express "atom v is a carbon connected to at least 3 carbons by single bonds". In standard GML, we can only write:

$$Carbon(x) \wedge \exists^{\geq 3} y\, (E(x, y) \wedge Carbon(y)), \tag{8}$$

which states that v has at least 3 carbon atoms in its neighborhood, but this only formalizes that v should be a carbon and that the neighbor nodes should be carbons, missing the information on the single bonds. GML fundamentally cannot express properties that depend on edge features or relationships between node pairs. In molecular property prediction, the outcome often depends on specific bond types (single, double, ...) connecting atoms, not just atom types. GML cannot distinguish "carbon connected to an oxygen via double bond" from "carbon connected to an oxygen via single bond" so it is a critical limitation for chemistry applications, as well as many others.

In PGML, we introduce ternary predicates $R(x, y, e)$, where x and y are nodes and e represents the edge feature vector of the edge connecting them. This enables direct expression:

$$\exists^{\geq 3} y\, (E(x, y) \wedge Single(e_{xy}) \wedge Carbon(x) \wedge Carbon(y))), \tag{9}$$

which unambiguously states that v has at least 3 C-C edges, cleanly distinguishing edge-level from node-level predicates. Formally, a PGML graph structure is a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{P_i\}_{i=1}^{I}, \{R_j\}_{j=1}^{K})$ where:

- $\mathcal{V}$ is a finite set of nodes
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge relation

- $P_i \subseteq \mathcal{V}$ are unary predicates (node properties)
- $R_j \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{E}$ are ternary predicates (edge properties)

PGML formulas are a syntactic restriction of first-order logic with exactly one free variable x. The language includes:

- Unary predicate symbols $\{P_1(x), \ldots, P_I(x)\}$ representing node properties
- Ternary relation symbols $\{R_1(x, y, e_{xy}), \ldots, R_K(x, y, e_{xy})\}$ representing edge properties, where e_{xy} denotes the edge between x and y
- Binary relation $E(x, y)$ representing graph adjacency

PGML formulas $\varphi(x)$ with free variable x are defined inductively:

$$\begin{aligned}\varphi(x) ::= & P(x) \mid \neg\psi(x) \mid \psi_1(x) \wedge \psi_2(x) \\ & \mid \psi_1(x) \vee \psi_2(x) \\ & \mid \exists^{\geq N} y \, (E(x, y) \wedge R(x, y, e_{xy}))\end{aligned}$$

The key extension beyond GML is the inclusion of ternary predicate $R(x, y, e_{xy})$ in the quantifier, enabling counting conditioned simultaneously on source node x, target node y, and edge features e_{xy}.

4. LogiX-Me

This section presents the LogiX-Me architecture, detailing its dual-layer components and training procedures.

LogiX-Me consists of a layer made of two distinct logical transformation components, both implemented as Transparent Explainable Logic Layers (TELL), that process information at different levels: $\lambda_E^{(k)}$ operates at the edge level to learn relational predicates, while $\lambda_N^{(k)}$ operates at the node level to learn neighborhood counting patterns.

$$\mathbf{h}_v^{(k)} = \lambda_N^{(k)} \left(\beta^{(k)} \left(\sum_{u \in \mathcal{N}(v)} \lambda_E^{(k)}([\mathbf{h}_v^{(k-1)} \parallel \mathbf{h}_u^{(k-1)} \parallel \mathbf{e}_{vu}]) \right) \right) \tag{10}$$

For each edge (v, u) in the graph, LogiX-Me computes a message by first concatenating features, then applying λ_E:

$$\mathbf{z}_{vu} = [\mathbf{h}_v^{(k-1)} \parallel \mathbf{h}_u^{(k-1)} \parallel \mathbf{e}_{vu}] \tag{11}$$

where $\mathbf{h}_v^{(k-1)} \in \mathbb{R}^d$ are source node features, $\mathbf{h}_u^{(k-1)} \in \mathbb{R}^d$ are target node features, and $\mathbf{z}_{vu} \in \mathbb{R}^{2d+d_e}$ are concatenated features. If edge features are not present, $\mathbf{z}_{vu} = [\mathbf{h}_v^{(k-1)} \parallel \mathbf{h}_u^{(k-1)}]$. The concatenated features pass through λ_E:

$$\mathbf{m}_{vu} = \sigma(\mathbf{z}_{vu} \cdot \mathbf{W}_E^{+(k)} + \mathbf{b}_E^{(k)}) \tag{12}$$

where $\mathbf{W}_E^{+(k)} \in \mathbb{R}_{\geq 0}^{(2d+d_e) \times d'}$ is a non-negative weight matrix, $\mathbf{b}_E^{(k)} \in \mathbb{R}^{d'}$ is bias vector, σ is sigmoid activation function and $\mathbf{m}_{vu} \in [0, 1]^{d'}$ is the message from u to v. Each output dimension j of $\mathbf{m}_{vu}$ corresponds to a learned ternary predicate $R_j(v, u, \mathbf{e}_{vu})$. The message $\mathbf{m}_{vu}[j] \approx 1$ indicates that the triple $(v, u, \mathbf{e}_{vu})$ satisfies predicate R_j. This directly corresponds to PGML's ternary predicates $R(x, y, e)$. Then, the messages from all neighbors are summed:

$$\mathbf{a}_v^{(k)} = \sum_{u \in \mathcal{N}(v)} \mathbf{m}_{vu} \tag{13}$$

where $\mathbf{a}_v^{(k)} \in \mathbb{R}^{d'}$ are the aggregated messages and each dimension $\mathbf{a}_v^{(k)}[j] \approx$ count of neighbors u such that $R_j(v, u, \mathbf{e}_{vu})$ holds. Since $\mathbf{m}_{vu}$ values are approximately binary (due to steep sigmoid), $\mathbf{a}_v^{(k)}[j]$

represents the number of edges (v, u) satisfying predicate $R_j(v, u, \mathbf{e}_{vu})$. This directly corresponds to PGML's counting quantifiers over ternary predicates $\mathbf{a}_v^{(k)}[j] \approx |\{u \in \mathcal{N}(v) : (v, u, \mathbf{e}_{vu}) \in R_j\}|$. After aggregation, counts are re-binarized using the Fourier-based step function. The binarized finally counts pass through λ_N:

$$\mathbf{h}_v^{(k)} = \sigma(\tilde{\mathbf{a}}_v^{(k)} \cdot \mathbf{W}_N^{+(k)} + \mathbf{b}_N^{(k)}) \tag{14}$$

where $\tilde{\mathbf{a}}_v^{(k)} = \beta^{(k)}(\mathbf{a}_v^{(k)}) \in [0, 1]^{d'}$ are the binarized counts, $\mathbf{W}_N^{+(k)} \in \mathbb{R}_{\geq 0}^{d' \times d}$ is a non-negative weight matrix, $\mathbf{b}_N^{(k)} \in \mathbb{R}^d$ is the bias vector and $\mathbf{h}_v^{(k)} \in [0, 1]^d$ is the output node embedding. Each output dimension j of $\mathbf{h}_v^{(k)}$ corresponds to a PGML formula combining multiple counting conditions:

$$\mathbf{h}_v^{(k)}[j] = 1 \iff \mathcal{G}, v \models \bigvee_{S \in \mathcal{S}_j} \left(\bigwedge_{i \in S} \exists^{\geq N_i} y \, (E(x, y) \wedge R_i(x, y, e_{xy})) \right) \tag{15}$$

where $\mathcal{S}_j$ is the set of minimal activating sets for the j-th output and R_i are predicates learned by $\lambda_E^{(k)}$.

LogiX-Me requires binary or approximately binary features to enable faithful logical interpretation. However, many real-world datasets contain continuous node or edge features. To handle this, LogiX-Me introduces optional binarization layers that apply the Fourier step function before passing through the first convolution layer.

To increase model expressiveness, LogiX-Me optionally concatenates negated features:

$$\mathbf{h}_{\text{aug}} = [\mathbf{h} \parallel (1 - \mathbf{h})] \tag{16}$$

This enables learning rules with explicit negation, e.g., "NOT carbon". The rationale stems from the non-negative weight constraint in TELL layers: with $\mathbf{W}^+ \in \mathbb{R}_{\geq 0}$, the model can only learn positive conjunctions. To express a negated literal $\neg R(x, y, e)$ in a rule, the model would need negative weights, which are not permitted in the TELL formulation. Negative concatenation circumvents this limitation by explicitly providing both $R(x, y, e)$ and $\neg R(x, y, e) = 1 - R(x, y, e)$ as separate input features, allowing the model to select whichever is needed through positive weights. Three modes are available: no negative concatenation (model limited to only positive conjunctions), negative concatenation at the first layer (it enables negation at the input level (e.g., "NOT carbon"), but higher layers cannot negate learned patterns from previous layers) and negative concatenation at all layers (it enables complex rules like "node does NOT have pattern A" where pattern A itself was learned in previous layers). The last mode is used in LogiX-GIN. We experimented with these modes in order to see if negation is truly necessary and in order to reduce the weights of the model.

We also control whether edge features are reused at every layer or only at the first layer. Unlike node features, which are updated at each layer to encode increasingly abstract patterns, edge features remain static throughout the network. Since edge features are constant across layers, any edge property expressible using $\mathbf{e}_{vu}$ at layer $k > 1$ could equivalently be learned at layer 1. If layer 1 learns all relevant ternary predicates $R(x, y, e)$ over the edge features, subsequent layers can reference these learned predicates through node embeddings without requiring direct access to the raw edge features. In other words, the information in $\mathbf{e}_{vu}$ should already be encoded in the patterns learned by earlier layers

In our implementation, K LogiX-Me layers are stacked (typically $K = 5$) to enable hierarchical feature composition: the first layers learns atomic ternary predicates over node pairs and edges that are then combined into higher-order patterns. Since each layer's predicates can reference previous layers' predicates, the final rules can be recursively expanded back to input features, providing complete explanations.

For graph-level tasks, node embeddings from each layer are aggregated using three readout functions (sum, mean and max) and then are passed through a final TELL layer. In node classification, node embeddings from each layer are directly passed as the input of a final TELL layer. For link prediction, we compute the element-wise product of the embeddings of the two endpoint nodes from each layer

and pass the resulting representation to a TELL layer. This final layer also produces interpretable rules explaining how graph-level or node-level features combine to predict classes.

Training deep logic-constrained networks presents significant challenges due to vanishing gradients from multiple sigmoid activations and the constraint of non-negative weights in TELL layers. LogiX-Me employs a 3-phase training strategy with knowledge distillation at both node-level and message-level representations. The first phase is knowledge distillation pretraining, where we train an unconstrained model with an identical architecture, but with arbitrary MLPs (no non-negative weight constraints), Gumbel-Sigmoid activations to encourage binary-like representations, and standard cross-entropy loss for the classification task. We use this model as a teacher to supervise the logic-constrained student model. For each LogiX-Me layer k, we distill both the post-aggregation output and the intermediate message output.

The second phase is fine-tuning, where the logic model is trained end-to-end on the actual task labels. We use a hyperparameter to regulate whether we want to maintain the distillation loss also in this phase. In this phase, the Fourier step function increases its value τ, encouraging sharp binarization.

The third phase is a post-training pruning where we apply Hoyer regularization to further reduce rule complexity. For each layer individually (in reverse order $K \to 1$) we use $\mathcal{L}_{\text{Hoyer}} = \frac{||\mathbf{W}^+||_1}{||\mathbf{W}^+||_2}$. This technique encourages the use of a few large weights while pushing the remaining ones toward zero.

5. Experiments

We evaluate LogiX-Me across 13 diverse graph learning tasks, comparing it against the teacher model and LogiX-GIN. We evaluate across four distinct domains: Synthetic tasks include *BA2Motifs* and *BAMultiShapes* (graph classification), plus *BAShapes*, *BACommunity*, and *TreeGrid* (node classification). Molecular benchmarks comprise *MUTAG*, *Mutagenicity*, *NCI1*, and *BBBP* for property prediction, alongside *PROTEINS*. We also include Social networks (*Reddit-Binary*) and Citation networks (*Cora*, *CiteSeer*) for link prediction. We report accuracy for classification and AUC-ROC for link prediction (averaged over 5 seeds).

Table 1 presents the complete performance comparison. For graph classification, LogiX-Me matches the teacher on synthetic tasks and substantially improves on *MUTAG* and *PROTEINS*, though it sees moderate decay on social networks. Notably, in link prediction tasks, LogiX-Me significantly outperforms the teacher on both Cora and CiteSeer.

LogiX-Me also consistently outperforms LogiX-GIN, with dramatic improvements on *BACommunity*, *PROTEINS*, and *MUTAG*. These gains confirm that edge-aware predicates capture relational patterns (e.g., specific bond types or community roles) that are inexpressible in LogiX-GIN's monadic framework.

Table 1
Performance comparison. Accuracy (%) for Classification, AUC for Link Prediction. **Diff** indicates improvement of LogiX-Me over LogiX-GIN. Best results in bold.

Dataset	Task	Teacher	LogiX-GIN	LogiX-Me	Diff
BA2Motifs	GC	100.0 ± 0.0	100.0 ± 0.0	$\mathbf{100.0 \pm 0.0}$	0.0
BAMultiShapes	GC	100.0 ± 0.0	100.0 ± 0.0	$\mathbf{100.0 \pm 0.0}$	0.0
MUTAG	GC	71.6 ± 8.0	81.1 ± 4.7	$\mathbf{85.3 \pm 4.4}$	+4.2
Mutagenicity	GC	82.5 ± 2.0	$\mathbf{79.3 \pm 1.3}$	78.6 ± 3.0	−0.7
NCI1	GC	80.2 ± 1.7	$\mathbf{78.4 \pm 1.3}$	77.6 ± 1.7	−0.9
BBBP	GC	85.2 ± 1.8	86.4 ± 0.9	$\mathbf{86.4 \pm 2.0}$	0.0
PROTEINS	GC	72.9 ± 6.8	70.9 ± 5.6	$\mathbf{75.2 \pm 6.2}$	+4.3
Reddit-Binary	GC	94.6 ± 1.2	85.6 ± 3.2	$\mathbf{87.8 \pm 3.3}$	+2.2
BAShapes	NC	97.1 ± 1.0	93.4 ± 3.0	$\mathbf{97.4 \pm 1.6}$	+4.0
BACommunity	NC	95.9 ± 1.4	85.7 ± 2.7	$\mathbf{94.3 \pm 1.0}$	+8.6
TreeGrid	NC	99.4 ± 0.4	98.2 ± 0.9	$\mathbf{99.4 \pm 0.4}$	+1.1
Cora	LP	86.1 ± 1.2	-	$\mathbf{88.7 \pm 0.7}$	-
CiteSeer	LP	79.4 ± 1.3	-	$\mathbf{84.2 \pm 2.7}$	-

To explain the quantitative performance gap, we analyse the logical formulas extracted by both models. On the *MUTAG* dataset, which classifies mutagenic nitroaromatic compounds, LogiX-GIN is constrained to unary predicates. It identifies patterns by nesting modal operators, typically generating rules like:

$$\exists^{>0} y \left(E(x, y) \land C(y) \land \exists^{>0} z \left(E(y, z) \land N(z) \land \exists^{>1} w \left(E(z, w) \land O(w) \right) \right) \right) \tag{17}$$

This formula describes a topological structure (Carbon connected to Nitrogen, connected to Oxygens) but fails to specify the nature of these connections (single vs. double bonds), which is chemically vital. In contrast, LogiX-Me, leveraging PGML, utilizes ternary predicates $R(x, y, e)$ to explicitly incorporate edge features. Our analysis indicates that it successfully extracts rules capturing precise bond information, such as:

$$\exists^{\geq 1} y \left(E(x, y) \land C(x) \land N(y) \land \text{single}(e_{xy}) \right) \land \exists^{\geq 1} z \left(E(y, z) \land N(y) \land O(z) \land \text{double}(e_{yz}) \right) \tag{18}$$

This enhanced expressivity allows LogiX-Me to distinguish between chemically distinct substructures (e.g., actual nitro groups vs. generic C-N-O chains) that appear topologically identical to the monadic baseline, directly contributing to its higher accuracy.

We further evaluate whether this increased expressiveness comes at the cost of interpretability. We define complexity as the length of the induced logical rule (number of literals and operators). Despite the richer semantics of PGML, LogiX-Me does not produce significantly more complex rules. In fact, the introduction of polyadic predicates often leads to more concise explanations. Where LogiX-GIN requires deep nesting of modal operators to approximate a relational pattern, LogiX-Me can express the same pattern with a shallower formula using explicit edge constraints. Empirically, while the average rule length is slightly higher due to the inclusion of edge predicates, the effective depth required to achieve peak performance is lower. This suggests that PGML primitives are more semantically dense for graph tasks, allowing the model to provide explanations that are easier for domain experts.

6. Conclusions

We introduced LogiX-Me, a self-explainable GNN that integrates Polyadic Graded Modal Logic (PGML) into its architecture to capture edge-aware and relational patterns. By employing a dual-layer design, the model explicitly learns ternary predicates $R(x, y, e)$ and neighborhood counting constraints, overcoming the expressivity limits of prior monadic approaches.

Our experiments demonstrate that LogiX-Me achieves competitive performance with black-box baselines while significantly outperforming the LogiX-GIN baseline on tasks requiring edge differentiation. Crucially, this expressiveness yields concise, human-readable logical rules without sacrificing interpretability. LogiX-Me thus offers a robust framework for transparent graph learning, proving that self-explanation can coexist with high predictive accuracy.

References

[1] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence (2019).

[2] A. Ragno, B. La Rosa, R. Capobianco, Prototype-based interpretable graph neural networks, IEEE Transactions on Artificial Intelligence (2022).

[3] T. Bui, W. Li, Toward interpretable graph neural networks via con cept matching model, ICDM (2023).

[4] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, R. He, Graph information bottleneck for subgraph recognition, ICLR (2021).
[5] S. Azzolin, S. Malhotra, A. Passerini, S. Teso, Beyond topological self-explainable gnns: A formal explainability perspective, in: ICML, 2025.
[6] A. Ragno, M. Plantevit, C. Robardet, On logic-based self-explainable graph neural networks, NeurIPS (2025).
[7] M. De Rijke, A note on graded modal logic, Studia Logica (2000).
[8] A. Ragno, M. Plantevit, C. Robardet, R. Capobianco, Transparent explainable logic layers, ECAI (2024).