Interpretable Graph Networks Formulate Universal Algebra Conjectures

Francesco Giannini[*]
CINI, Italy
francesco.giannini@unisi.it

Stefano Fioravanti[*]
Università di Siena, Italy, JKU Linz, Austria Italy
stefano.fioravanti@unisi.it

Oguzhan Keskin
University of Cambridge, UK
ok313@cam.ac.uk

Alisia Maria Lupidi
University of Cambridge, UK
aml201@cam.ac.uk

Lucie Charlotte Magister
University of Cambridge, UK
lcm67@cam.ac.uk

Pietro Lió
University of Cambridge, UK
pl219@cam.ac.uk

Pietro Barbiero[*]
Università della Svizzera Italiana, CH
University of Cambridge, UK
barbip@usi.ch

Abstract

The use of AI in Universal Algebra (UA)—one of the fields laying the foundations of modern mathematics—is still completely unexplored. While UA's topological representations would enable the analysis of such properties using graph neural networks, the limited transparency and brittle explainability of these models hinder their straightforward use to empirically validate existing conjectures or to formulate new ones. To bridge these gaps, we generate AI-ready datasets based on UA's conjectures, and introduce a novel neural layer to build fully interpretable graph networks. The results of our experiments demonstrate that interpretable graph networks strongly generalize when predicting universal algebra's properties, and generate simple explanations that empirically validate existing conjectures.

1 Introduction

Universal Algebra (UA, [1]) is one of the foundational fields of modern mathematics, yet the complexity of studying abstract algebraic structures hinders scientific progress and discourages many academics. As several UA conjectures equivalently characterize algebraic properties using equations or graphs [2], these properties could be studied with relational AI techniques, such as Graph Neural Networks (GNN, [3]). However, *GNNs' opaque reasoning* obstructs human understanding of their decision process [4]. Compounding the issue of GNNs' limited transparency, GNN explainability methods mostly rely on brittle and untrustworthy local/post-hoc methods [4–8] or pre-defined subgraphs for explanations [9, 10], which are often unknown in UA.

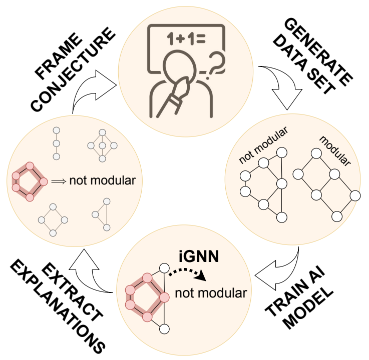

Figure 1: Interpretable graph networks support universal algebra research.

Contributions. In this work we study 5 properties of UA structures known as lattice varieties i.e., distributivity, modularity, meet/join-semi/semi-distributivity [11, 12] (see App. A for further details). To this end, we build a large dataset of lattice varieties (see App. B) and introduce a novel neural layer that makes GNNs fully interpretable, according to Rudin's [4] notion of interpretability (Sec. 2). The results of our experiments (Sec. 3) demonstrate that interpretable GNNs provide interpretable predictions without sacrificing task accuracy, and generate simple concept-based explanations that empirically validate existing conjectures.

2 Interpretable Graph Networks

In this section, we design an interpretable graph network (iGNN) that satisfies the notion of "interpretability" introduced by Rudin [4]. According to this definition, a machine learning (ML) system is interpretable if and only if (1) its inputs are semantically meaningful, and (2) its model inference is simple for humans to understand (e.g., linear/sparse/symbolic). To achieve this goal in GNNs, we introduce an interpretable graph layer that learns semantically meaningful concepts and uses them as inputs for an interpretable classification layer.

2.1 Interpretable Graph Layer

Node-level concepts. An interpretable concept space is the first step towards interpretability. Following Ghorbani et al. [13], a relevant concept is a “high-level human-understandable unit of information” shared by input samples and thus identifiable with clustering techniques. Message passing algorithms do cluster node embeddings based on the structure of node neighborhoods, as observed by Magister et al. [7]. However, the real-valued large embedding representations $\mathbf{h}_i \in \mathbb{R}^q, q \in \mathbb{N}$ generated by message passing can be challenging for humans to interpret. To address this, we use a hard Gumbel-Softmax activation $\Theta : \mathbb{R}^q \mapsto \{0,1\}^q$, following Azzolin et al. [9]:

$$\mathbf{c}_i = \Theta\big(\mathbf{h}_i\big) \qquad \mathbf{h}_i = \phi\Big(\mathbf{x}_i, \bigoplus_{j \in N_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\Big) \tag{1}$$

where $\mathbf{x}_* \in \mathbb{R}^n$ is a node’s feature vector, ψ and ϕ are learnable functions aggregating information from a node neighborhood N_i, and $\oplus$ is a permutation invariant aggregation function (such as sum or mean). During the forward pass, the Gumbel-Softmax activation Θ produces a one-hot encoded representation of each node embedding. Since nodes sharing the same neighborhood have similar embeddings $\mathbf{h}_i$ due to message passing, they will also have the same one-hot vector $\mathbf{c}_i$ due to the Gumbel-Softmax—allowing us to interpret nodes having the same one-hot concept $\mathbf{c}_i$ as nodes having similar embeddings $\mathbf{h}_i$ and thus sharing a similar neighborhood. More formally, we can assign a semantic meaning to a reference concept $\gamma \in \{0,1\}^q$ by visualizing concept prototypes corresponding to the inverse images of a node concept vector. In practice, we can consider a subset of the input lattices Γ corresponding to the node’s (p-hop) neighborhood covered by message passing:

$$\Gamma(\gamma, p) = \Big\{\mathbf{L}^{\langle i,p\rangle} \mid i \in L \wedge \ \mathbf{L} \in \mathcal{D} \wedge \ \mathbf{c}_i = \gamma\Big\} \tag{2}$$

where $\mathcal{D}$ is the set of all training lattices $\mathbf{L}$ (see App. A for lattices, and App. B for dataset), and $\mathbf{L}^{\langle i,p\rangle}$ is the graph corresponding to the p-hop neighborhood ($p \in \{1, \dots, |L|\}$) of the node $i \in L$, as suggested by Magister et al. [7], Ghorbani et al. [13]. This way, by visualizing concept prototypes as subgraph neighborhoods, the meaning of the concept representation becomes easily interpretable to humans, aiding in the understanding of the reasoning process of the network.

Example 2.1 (Interpreting node-level concepts)**.** Consider the problem of classifying distributive lattices with a simplified dataset including the lattices $\mathbf{N}_5$ and $\mathbf{M}_3$ only, and where each node has a constant feature $x_i = 1$. As these two lattices only have nodes with 2 or 3 neighbours, one layer of message passing will then generate only two types of node embeddings e.g., $\mathbf{h}_{II} = [0.2, -0.4, 0.3]$ for nodes with a 2-nodes neighborhood (e.g.,), and $\mathbf{h}_{III} = [0.6, 0.2, -0.1]$ for nodes with a 3-nodes neighborhood (e.g.,). As a consequence, the Gumbel-Softmax will only generate two possible concept vectors e.g., $\mathbf{c}_{II} = [0, 0, 1]$ and $\mathbf{c}_{III} = [1, 0, 0]$. Hence, for instance the concept belongs to $\mathbf{c}_{II}$, while belongs to $\mathbf{c}_{III}$.

Graph-level concept embeddings. To generate a graph-level concept space in the interpretable graph layer, we can utilize the node-level concept space produced by the Gumbel-Softmax. Normally, graph-level embeddings are generated by applying a permutation invariant aggregation function on node embeddings. However, in iGNNs we restrict the options to (piece-wise) linear permutation invariant functions in order to follow our interpretability requirements dictated by Rudin [4]. This restriction still includes common options such as max or sum pooling. Max pooling can easily be interpreted by taking the component-wise max over the one-hot encoded concept vectors $\mathbf{c}_i$. After max pooling, the graph-level concept vector has a value of 1 at the k-th index if and only if at least one node activates the k-th concept i.e., $\exists i \in L, \mathbf{c}_{ik} = 1$. Similarly, we can interpret the output of a sum pooling: a graph-level concept vector takes a value $v \in \mathbb{N}$ at the k-th index after sum pooling if and only if there are exactly v nodes activating the k-th concept i.e., $\exists i_0, \ldots, i_v \in L, \mathbf{c}_{ik} = 1$.

Example 2.2 (Interpreting graph-level concepts)**.** Following Example 2.1, let us use sum pooling to generate graph-level concepts. For an $\mathbf{N}_5$ graph, we have 5 nodes with exactly the same 2-node neighborhood. Therefore, sum pooling generates a graph-level embedding $[0, 0, 5]$, which certifies that we have 5 nodes of the same type e.g., . For an $\mathbf{M}_3$ graph, the top and bottom nodes have a 3-node neighborhood e.g., , while the middle nodes have a 2-node neighborhood e.g., . This means that sum pooling generates a graph-level embedding $[2, 0, 3]$, certifying that we have 2 nodes of type and 3 nodes of type .

2.2 Interpretable architectures

The interpretable graph layer can be used to instantiate different types of iGNNs. One approach is to plug this layer as the last message passing layer of a standard GNN architecture:

$$\hat{y} = f\Big(\boxplus_{i \in K} \ \Big(\Theta\Big(\phi^{(K)}\Big(\mathbf{h}_i^{(K-1)}, \bigoplus_{j \in N_i} \psi^{(K)}(\mathbf{h}_i^{(K-1)}, \mathbf{h}_j^{(K-1)}) \Big) \Big) \Big) \Big) \tag{3}$$

$$\mathbf{h}_i^{(l)} = \phi^{(l)}\Big(\mathbf{h}_i^{(l-1)}, \bigoplus_{j \in N_i} \psi^{(l)}(\mathbf{h}_i^{(l-1)}, \mathbf{h}_j^{(l-1)}) \Big) \quad l = 1, \ldots, K \tag{4}$$

where f is an interpretable classifier (e.g., single-layer network), $\boxplus$ is an interpretable piece-wise linear and permutation-invariant function (such as max or sum), Θ is a Gumbel-Softmax hard activation function, and $\mathbf{h}_i^0 = \mathbf{x}_i$. In this way, we can interpret the first part of the network as a feature extractor generating well-clustered latent representations from which concepts can be extracted. This approach is useful when we only care about the most complex neighborhoods/concepts. Another approach is to generate a hierarchical transparent architecture where each GNN layer is interpretable:

$$\hat{y}^{(l)} = f\Big(\boxplus_{i \in K} \Big(\Theta\Big(\mathbf{h}_j^{(l)} \Big) \Big) \Big) \qquad l = 1, \ldots, K \tag{5}$$

In this case, we can interpret every single layer of our model with concepts of increasing complexity. The concepts extracted from the first layer represent subgraphs corresponding to the 1-hop neighborhood of a node, those extracted at the second layer will correspond to 2-hop neighborhoods, and so on. These hierarchical iGNNs can be useful to get insights into concepts with different granularities. By analyzing the concepts extracted at each layer, we gain a better understanding of the GNN inference and of the importance of different (sub)graph structures for the classification task.

3 Experiments and Key Findings

For our comparative study, we evaluate the performance of iGNNs and their hierarchical version against equivalent GNN models (i.e., having the same hyperparameters such as number layers, training epochs, and learning rate). We employ three quantitative metrics to assess a model's generalization and interpretability. We use the Area Under the Receiver Operating Characteristic (AUC ROC) curve to assess task generalization. To evaluate interpretability, we use standard metrics such as completeness [14] and fidelity [15]. Completeness assesses the quality of the concept space on a global scale using an interpretable model to map concepts to tasks, while fidelity measures the difference in predictions obtained with an interpretable surrogate model and the original model. All metrics are computed on test sets using 5 random seeds, and reported using the mean and 95% confidence interval. App C covers experimental details, and App. D shows additional results.

iGNNs improve interpretability without sacrificing task accuracy (Figure 2). Our experimental evaluation reveals that interpretable GNNs are able to strike a balance between completeness and fidelity, two crucial metrics that are used to assess generalization-interpretability trade-offs [15]. We observe that the multilabel classification scenario, which requires models to learn a more varied and diverse set of concepts, is the most challenging and results in the lowest completeness scores on average. On the contrary, interpretable surrogate models of black-box GNNs exhibit, as expected, lower fidelity scores, confirming analogous observations in the explainable AI literature [4, 15]. In practice, this discrepancy between the original black-box predictions and the predictions obtained with an interpretable surrogate model questions the actual usefulness of black-boxes when interpretable alternatives achieve similar results in solving the problem at hand, as extensively discussed by Rudin [4]. Overall, these results demonstrate how concept spaces are highly informative to solve universal algebra's tasks and how the interpretable graph layer may improve GNNs' interpretability without sacrificing task accuracy.

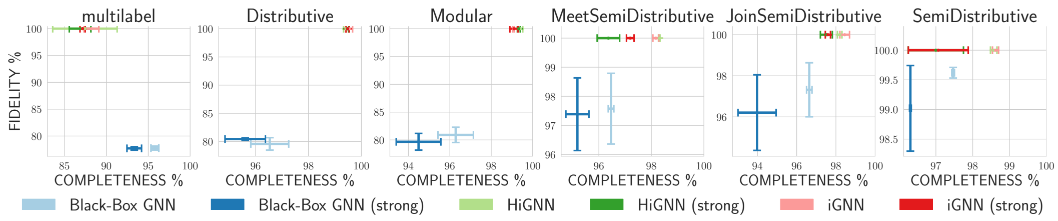

Figure 2: Accuracy-interpretability trade-off in terms of concept completeness (accuracy) and model fidelity (interpretability). iGNNs attain optimal fidelity as model inference is inherently interpretable, outmatching equivalent black-box GNNs. All models attain similar results in terms of completeness.

Concept-based explanations empirically validate universal algebra's conjectures (Figure 3). We present empirical evidence to support the validity of existing UA conjectures by examining the concepts generated for modular and distributive tasks. Similarly to Ribeiro et al. [15], we visualize in Figure 3 the weights of our trained linear classifier representing the relevance of each concept. We remark that the visualization is limited to the (top-5) most negative weights, as we are interested in those concepts that negatively affect the prediction of a property. In the same plot, we also show the prototype of each concept represented by the 2-hop neighborhood of a node activating the concept, following a similar procedure as Magister et al. [7], Azzolin et al. [9], Ghorbani et al. [13]. Using this visualization, we investigate the presence of certain concepts when classifying modular and distributive lattices. For the modularity task, our results show that the lattice $\mathbf{N}_5$ appears among non-modular concepts, but is never found in modular lattices, while the lattice $\mathbf{M}_3$ appears among both modular and non-modular concepts, which is consistent with Dedekind [11]. In the case of distributivity, we observe that both $\mathbf{M}_3$ and $\mathbf{N}_5$ are present among non-distributive concepts, and are never found in distributive lattices, which is also in line with Birkhoff [12].

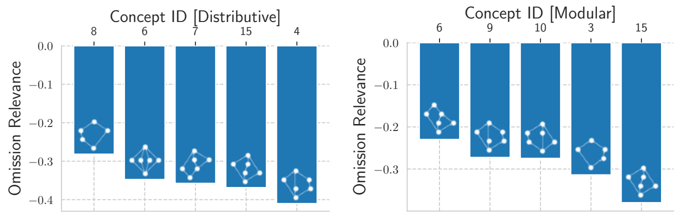

Figure 3: Ranking of relevant clusters of lattices (x-axis) according to the interpretable GNN linear classifier weights (y-axis, the lower the more relevant the cluster). $\mathbf{N}_5$ is always the most important lattice to omit for modularity, while both $\mathbf{M}_3$ and $\mathbf{N}_5$ are relevant for distributivity, thus validating theorems **??** and **??**.

4 Discussion

Relations with Graph Neural Network explainability. Inspired by vision approaches [13, 15, 16], early explainability techniques focused on feature importance [17], while subsequent works aimed to extract local explanations [6, 8, 10] or global explanations using conceptual subgraphs by clustering the activation space [7, 18, 19]. However, all these techniques either rely on pre-defined subgraphs for explanations (which are often unknown in UA) or provide post-hoc explanations which may be brittle and unfaithful as extensively demonstrated by Rudin [4]. On the contrary, our experiments show that iGNNs generate interpretable predictions according to Rudin [4] notion of interpretability via linear classifiers applied on sparse human-understandable concept representations.

Broader impact and perspectives. AI techniques are becoming increasingly popular for solving previously intractable mathematical problems and proposing new conjectures [20–23]. However, the use of modern AI methods in universal algebra was a novel and unexplored field until the development of the approach presented in this paper. To this end, our method uses interpretable graph networks to suggest graph structures that characterize relevant algebraic properties of lattices. With our approach, we empirically validated Dedekind [11] and Birkhoff [12] theorems on distributive and modular lattices, by recovering relevant lattices. This approach can be readily extended—beyond equational properties determined by the omission of a sublattice in a variety [24]—to any structural property of lattices, including the characterization of congruence lattices of algebraic varieties [24–27]. Our methodology can also be applied (beyond universal algebra) to investigate (almost) any mathematical property that can be topologically characterized on a graph, such as the classes of graphs/diagraphs with a fixed set of polymorphisms [28–30].

References

[1] Stanley Burris and H. P. Sankappanavar. *A Course in Universal Algebra*. Springer, 1981. 1, 7

[2] Peter Jipsen and Henry Rose. *Varieties of Lattices*. Springer Berlin, 1992. 1

[3] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. 1

[4] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019. 1, 2, 4

[5] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[6] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020. 4

[7] Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Liò. Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks. *arXiv preprint arXiv:2107.11889*, 2021. 2, 4, 10

[8] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks, 2019. 1, 4, 11

[9] Steve Azzolin, Antonio Longa, Pietro Barbiero, Pietro Liò, and Andrea Passerini. Global explainability of gnns via logic combination of learned concepts. *arXiv preprint arXiv:2210.07147*, 2022. 1, 2, 4

[10] Minh Vu and My T Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems*, 33:12225–12235, 2020. 1, 4

[11] Richard Dedekind. Über die von drei Moduln erzeugte Dualgruppe. *Math. Ann.*, 1900. 1, 4, 5, 9

[12] Garrett Birkhoff. On the structure of abstract algebras. In *Mathematical proceedings of the Cambridge philosophical society*, volume 31, pages 433–454. Cambridge University Press, 1935. 1, 4, 5, 9

[13] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 4

[14] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565, 2020. 3

[15] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 3, 4

[16] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 4

[17] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10772–10781, 2019. 4

[18] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. Protgnn: Towards self-explaining graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9127–9135, 2022. 4

[19] Lucie Charlotte Magister, Pietro Barbiero, Dmitry Kazhdan, Federico Siciliano, Gabriele Ciravegna, Fabrizio Silvestri, Mateja Jamnik, and Pietro Lio. Encoding concepts in graph neural networks. *arXiv preprint arXiv:2207.13586*, 2022. 4

[20] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*. 5

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

[22] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis, and Pushmeet Kohli. Advancing mathematics by guiding human intuition with AI. *Nature*, 600(7887):70–74, December 2021. doi: 10.1038/s41586-021-04086-x. URL `https://doi.org/10.1038/s41586-021-04086-x`.

[23] Yang-Hui He. Machine-learning mathematical structures. *International Journal of Data Science in the Mathematical Sciences*, pages 1–25, 2022. 5

[24] Philip M Whitman. Free lattices. *Annals of Mathematics*, pages 325–330, 1941. 5

[25] Paolo Aglianò, Stefano Bartali, and Stefano Fioravanti. On Freese's technique. *arXiv:2302.11452*, 2022.

[26] Keith Kearnes and Emil Kiss. The shape of congruence lattices. *Memoirs of the American Math. Soc.*, 2013.

[27] J. B. Nation. Varieties whose congruences satisfy certain lattice identities. *Algebra Universalis*, 1974. 5

[28] Barto Libor and Kozik Marcin. Absorbing Subalgebras, Cyclic Terms, and the Constraint Satisfaction Problem. *Logical Methods in Computer Science*, Volume 8, Issue 1, 2012. 5

[29] Libor Barto, Marcin Kozik, and Todd Niven. The csp dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of bang-jensen and hell). *SIAM Journal on Computing*, 38(5):1782–1802, 2009.

[30] Miroslav Olšák. The local loop lemma. *Algebra universalis*, 2020. 5

[31] Alan Day. A characterization of modularity for congruence lattices of algebras*. *Canadian Mathematical Bulletin*, 12(2):167–173, 1969. doi: 10.4153/CMB-1969-016-6. 7

[32] Bjarni Jonnson. Algebras whose congruence lattices are distributive. *MATHEMATICA SCANDINAVICA*, 21:110–121, Dec. 1967. doi: 10.7146/math.scand.a-10850. URL `https://www.mscand.dk/article/view/10850`. 7

[33] Martin Hyland and John Power. The category theoretic understanding of universal algebra: Lawvere theories and monads. *Electronic Notes in Theoretical Computer Science*, 172:437–458, 2007. ISSN 1571-0661. doi: https://doi.org/10.1016/j.entcs.2007.02.019. URL `https://www.sciencedirect.com/science/article/pii/S1571066107000874`. Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin. 8

[34] Joel Berman and Paweł Idziak. Counting finite algebras in the post varieties. *International Journal of Algebra and Computation*, 10(03):323–337, 2000. 9

[35] Bjarni Jónsson and Ivan Rival. Lattice varieties covering the smallest non-modular lattice variety. *Pacific J. Math.*, Volume 82, 1978. 10

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 11

[37] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 11

A Algebra definitions

A.1 Formal defintions for Universal Algebra

Universal algebra is the field of mathematics that studies algebraic structures, which are defined as a set A along with its own collection of operations. An n-ary operation on A is a function that takes n elements of A and returns a single element from the set. More formally [1, 31, 32]:

Definition A.1. N-ary function For a non-empty set A and n non-negative integer we define $A^0 = \{\emptyset\}$ and, for $n > 0$, A^n is the set of n-tuples of elements from A. An n-ary *operation* on A is any function f from A^n to A; n is the *arity* of f. An operation f on A is called an n-ary operation if its arity is n.

Definition A.2. Algebraic Structure An *algebra* $\mathbf{A}$ is a pair (A, F) where A is a non-empty set called *universe* and F is a set of finitary operations on A.

Apart from the operations on A, an algebra is further defined by axioms, that in the particular case of universal algebras are in the form of identities.

Definition A.3. A *lattice* $\mathbf{L}$ is an algebraic structure composed by a non-empty set L and two binary operations $\vee$ and $\wedge$ satisfying the following axioms and their duals obtained exchanging $\vee$ and $\wedge$:

$$x \vee y \approx y \vee x \qquad \text{(commutativity)}$$
$$x \vee (y \vee z) \approx (x \vee y) \qquad \text{(associativity)}$$
$$x \vee x \approx x \qquad \text{(idempotency)}$$
$$x \approx x \vee (x \wedge y) \qquad \text{(absorption)}$$

Theorem A.4 ([1]). *A partially ordered set L is a lattice if and only if for every $a, b \in L$ both* supremum *and* infimum *of $\{a, b\}$ exist (in L) with $a \vee b$ being the supremum and $a \wedge b$ the infimum.*

Definition A.5. Let $\mathbf{L}$ be a lattice. Then $\mathbf{L}$ is *modular* (*distributive*, $\vee$*-semi-distributive*, $\wedge$*-semi-distributive*) if it satisfies the following:

$$x \leq y \rightarrow x \vee (y \wedge z) \approx y \wedge (x \vee z) \qquad \text{(modularity)}$$
$$x \vee (y \wedge z) \approx (x \vee y) \wedge (x \vee z) \qquad \text{(distributivity)}$$
$$x \vee y \approx x \vee z \rightarrow x \vee (y \wedge z) \approx x \vee y \qquad \text{($\vee$-semi-distributivity)}$$
$$x \wedge y \approx x \wedge z \rightarrow x \wedge (y \vee z) \approx x \wedge y \qquad \text{($\wedge$-semi-distributivity).}$$

Furthermore a lattice $\mathbf{L}$ is semi-distributive if is both $\vee$-semi-distributive and $\wedge$-semi-distributive

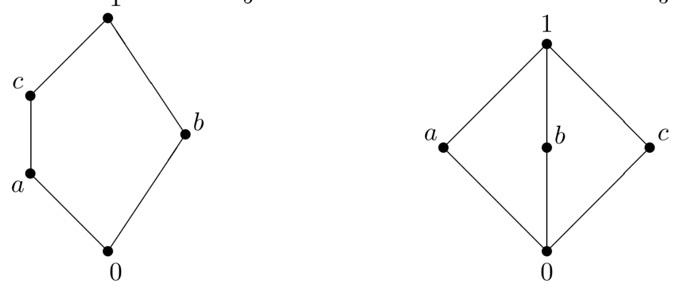

Figure 4: $\mathbf{N}_5$, a non-modular non-distributive and $\mathbf{M}_3$, a modular non-distributive lattice.

Definition A.6. Congruence Lattice
An *equivalence relation* on a set A is a binary relation $\sim$ that satisfies three properties: reflexivity, symmetry, and transitivity.

Reflexivity: For every element a in A, a is related to itself, denoted as $a \sim a$;

Symmetry: For any elements a and b in A, if $a \sim b$, then $b \sim a$;

Transitivity: For any elements a, b, and c in A, if $a \sim b$ and $b \sim c$, then $a \sim c$.

In other words, an equivalence relation partitions the set A into subsets, called *equivalence classes*, such that elements within the same class are equivalent to each other under the relation $\sim$.

Let $\mathbf{A}$ be an algebra. A *congruence* θ of $\mathbf{A}$ is a equivalent relation on A, that is compatible with the operations of $\mathbf{A}$. Formally, for every n-ary operation f of $\mathbf{A}$: if $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n) \in \theta$, then $(f(a_1, a_2, \ldots, a_n), f(b_1, b_2, \ldots, b_n)) \in \theta$. For every algebra $\mathbf{A}$ on the set A, the identity relation on A, and $A \times A$ are trivial congruences. An algebra with no other congruences is called *simple*. Let $\mathrm{Con}(\mathbf{A})$ be the set of congruences on the algebra $\mathbf{A}$. Since congruences are closed under intersection, we can define a meet operation: $\wedge : \mathrm{Con}(\mathbf{A}) \times \mathrm{Con}(\mathbf{A}) \to \mathrm{Con}(\mathbf{A})$ by simply taking the intersection of the congruences $E_1 \wedge E_2 = E_1 \cap E_2$. Congruences are not closed under union, however we can define the following closure operator of a binary relation E, with respect to a fixed algebra $\mathbf{A}$, such that its image is congruence: $\langle E \rangle_{\mathbf{A}} = \bigcap \{F \in \mathrm{Con}(\mathbf{A}) \mid E \subseteq F\}$. Note that the closure of a binary relation is a congruence and thus depends on the operations in $\mathbf{A}$, not just on the base set. Now define $\vee : \mathrm{Con}(\mathbf{A}) \times \mathrm{Con}(\mathbf{A}) \to \mathrm{Con}(\mathbf{A})$ as $E_1 \vee E_2 = \langle E_1 \cup E_2 \rangle_{\mathbf{A}}$. For every algebra $\mathbf{A}$, $(\mathrm{Con}(\mathbf{A}), \wedge, \vee)$ with the two operations defined above forms a lattice, called the *congruence lattice* of $\mathbf{A}$.

A *type* $\mathcal{F}$ is defined as a set of operation symbols along with their respective arities. Each operation symbol represents a specific operation that can be performed on the elements of the algebraic system. To refer to the specific operation performed by a given symbol f on an algebra $\mathbf{A}$ of type $\mathcal{F}$, we denote it as $f^{\mathbf{A}}$. This notation allows us to differentiate and access the particular operation carried out by f within the context of $\mathbf{A}$.

Definition A.7. Subalgebra Let $\mathbf{A}$ and $\mathbf{B}$ be two algebras of the same type. Then $\mathbf{B}$ is a *subalgebra* of $\mathbf{A}$ if $B \subseteq A$ and every fundamental operation of $\mathbf{B}$ is the restriction of the corresponding operation of $\mathbf{A}$, i.e., for each function symbol f, $f^{\mathbf{B}}$ is $f^{\mathbf{A}}$ restricted to $\mathbf{B}$.

Definition A.8. Homomorphic image Suppose $\mathbf{A}$ and $\mathbf{B}$ are two algebras of the same type $\mathcal{F}$, i.e. for each operation of $\mathbf{A}$, there exists a corresponding operation $\mathbf{B}$ with the same arity, and vice versa. A mapping $\alpha : A \to B$ is called a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ if

$$\alpha f^{\mathbf{A}}(a_1, \ldots, a_n) = f^{\mathbf{B}}(\alpha a_1, \ldots, \alpha a_n)$$

for each n-ary f in $\mathcal{F}$ and each sequence $a_1, \ldots, a_n$ from $\mathbf{A}$. If, in addition, the mapping α is onto then $\mathbf{B}$ is said to be a *homomorphic image* of $\mathbf{A}$.

Definition A.9. Direct product Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be two algebras of the same type $\mathcal{F}$. We define the direct product $\mathbf{A}_1 \times \mathbf{A}_2$ to be the algebra whose universe is the set $A_1 \times A_2$, and such that for $f \in \mathcal{F}$ and $a_i \in A_1$, $a'_i \in A_2$, $1 \leq i \leq n$,

$$f^{\mathbf{A}_1 \times \mathbf{A}_2}(\langle a_1, a'_1 \rangle, \ldots, \langle a_n, a'_n \rangle) = \langle f^{\mathbf{A}_1}(a_1, \ldots, a_n), f^{\mathbf{A}_2}(a'_1, \ldots, a'_n) \rangle$$

The collection of algebraic structures defined by equational laws are called varieties. [33]

Definition A.10. Variety A nonempty class K of algebras of type $\mathcal{F}$ is called a *variety* if it is closed under subalgebras, homomorphic images, and direct products.

B A Tool to Generate Datasets of Lattice Varieties

Algorithm 1: Generate dataset of lattice varieties.

```
Input: $n \geq 1$, hasProperty(·,·,·)                                  // $n$: cardinality
Dataset = []
AllFuncs ← genAllFuncs($n$)                     // binary functions as $n \times n$ matrices
for $L \in$ AllFuncs do                                        // $L(i,j) = 1$ meaning $i \leq_L j$
  if isPartialOrder($L$) then             // check if $\leq_L$ is refl., antisym. and trans.
    if isLattice($L$) then                                   // check if $L$ is a lattice
      for $i, j \leq n$ do
        $\wedge_L[i,j] \leftarrow \sup_{x \leq L}\{x \leq_L i \text{ and } x \leq_L j\}$
        $\vee_L[i,j] \leftarrow \inf_{x \leq L}\{i \leq_L x \text{ and } j \leq_L x\}$
      if hasProperty($L, \wedge_L, \vee_L$) then                 // check $\wedge_L$,$\vee_L$ properties
        Dataset.$append$([$L$, True])
      else
        Dataset.$append$([$L$, False])
```

We propose a general methodology to investigate any algebraic property whose validity can be verified on a finite lattice. In this work, we focus on properties that can be characterized via equations and quasi-equations. To train AI models, we propose a general dataset generator[1] for lattice varieties (Algorithm 1). The generator takes as input the number of nodes n in the lattices and a function to check whether a lattice satisfies a given property. We generate $2^{n \times n}$ matrices of size $n \times n$, containing all binary functions definable on $\{1, \dots, n\}^2$, and filter only binary matrices representing partial orders[2]. Then, we verify that the partial ordered set L is a lattice, by checking that any pair of nodes always has a unique infimum and supremum. This directly verifies that $\wedge_L$ and $\vee_L$ satisfy Definition A.3. Finally, we check whether the lattice satisfies the target property or not, and append it and the property label to our dataset. We remark that checking the validity of a single ternary equation on a medium-size lattice is not computationally prohibitive (i.e., it "only" requires checking n^3 identities), but the number of existing lattices increases exponentially as n increases. For instance, it is known that there are at least 2,000,000 non-isomorphic lattices with $n = 10$ elements [34]. Therefore, we only sample a fixed number of lattices per cardinality starting from a certain node cardinality. While this may seem a strong bias, we notice that known and relevant lattice omissions often rely on lattices with few nodes [11, 12]. To empirically verify that this is not a significant limitation, in our experiments we deliberately investigate the generalization capacity of GNNs when trained on small-size lattices and tested on larger ones. This way we can use GNNs to predict the satisfiability of equational properties on large graph structures without explicitly checking them. Using Algorithm 1, we generated the first large-scale AI-compatible datasets of lattices containing more than $29,000$ graphs and the labels of 5 key properties of lattice (quasi-)varieties i.e., modularity, distributivity, semi-distributivity, join semi-distributivity, and meet semi-distributivity.

C Experimental details

In practice, we train all models using eight message passing layers and different embedding sizes ranging from 16 to 64. We train all models for 200 epochs with a learning rate of 0.001. For interpretable models, we set the Gumbel-Softmax temperature to the default value of 1 and the activation behavior to "hard," which generates one-hot encoded embeddings in the forward pass, but computes the gradients using the soft scores. For the hierarchical model, we set the internal loss weight to 0.1 (to score it roughly 10% less w.r.t. the main loss). Overall, our selection of baselines aims at embracing a wide set of training setups and architectures to assess the effectiveness and versatility of GNNs for analyzing lattice properties in universal algebra. To demonstrate the robustness of our approach, we implemented different types of message passing layers, including graph convolution and GIN.

D Additional results

D.1 Contrastive explanations highlight topological differences between properties of lattice varieties (Figure 5).

We leverage interpretable GNNs to analyze the key topological differences of classical lattice properties such as join and meet semi-distribuitivity characterized by relevant quasi-equations

[1]The dataset generator code and the generated datasets will be made public in case of paper acceptance.

[2]Our algorithm optimizes this step considering only reflexive and antisymmetric binary relations, and enforces transitivity with an easy fix-point calculation.

(cf. Appendix A.5). To this end, we visualize specific concept prototypes corresponding to lattices that are not meet semi-distributive against lattices that are meet semi-distributive. We observe $\mathbf{N}_5$ but not $\mathbf{M}_3$ among the concepts of meet semi-distributive lattices, while we observe both $\mathbf{N}_5$ and $\mathbf{M}_3$ only in concepts that are not meet semi-distributive. This observation suggests that $\mathbf{N}_5$ is not a key lattice for meet semi-distributive lattices, *unlike distributive lattices*. Furthermore, we find that the lattice pattern is relevant for non meet semidistributivity, while its dual is relevant for non join semidistributivity, thus empirically confirming the hypotheses of Jónsson and Rival [35]. These findings are significant because they demonstrate how analyzing concepts in interpretable GNNs can provide universal algebraists with a powerful and automatic tool to formulate new conjectures based on identifying simple lattices that play a role in specific properties. By leveraging the power of interpretable GNNs, we may uncover previously unknown connections between different properties and identify new patterns and structures that could lead to the development of new conjectures and theorems in universal algebra, providing exciting opportunities for future research in universal algebra.

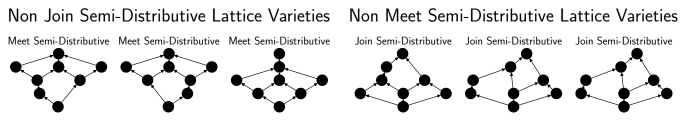

Figure 5: Contrastive explanations showing lattice varieties with a pair of discording labels to highlight the key difference between join and meet semi-distributivity.

D.2 Concept completeness and purity

Our experimental results (Tables 1 & 3) demonstrate that interpretable GNNs produce concepts with high completeness and low purity, which are standard quantitative metrics used to evaluate the quality of concept-based approaches. Completeness score is the accuracy of a classifier, such as decision tree, which takes concepts as inputs and predicts a label. Purity score is the number of graph edits, such as node/edge addition/eliminations, necessary to match two graphs in a cluster. A concept space is said to be pure if the purity score is zero.

We employ decision tree as the classifier, but compute recall instead of accuracy to calculate completeness score since the datasets are heavily unbalanced towards the negative labels. We compute purity scores for each cluster and report the average of those scores as the final purity score. Our approach achieves at least 73% and up to 87% recall, which shows that our interpretable models consistently avoid false negatives in the abundance of negative labels. We obtain around 3-4 purity scores, which suggests that our interpretable models extract relatively pure concept spaces in the presence of large lattices.

Furthermore, the hierarchical structure of interpretable GNNs enables us to evaluate the quality of intermediate concepts layer by layer. This hierarchy provides insights into why we may need more layers, and it can be used as a valuable tool to find the optimal setup and tune the size of the architecture. Additionally, it can also be used to compare the quality of concepts at different layers of the network. To that end, we compare the purity scores of the concept spaces obtained by the second layer and the final layer of HiGNN. As shown in Table 2, deeper layers may produce higher quality concepts for distributivity and join semi-distributivity whereas earlier layers may result in more reliable concepts for the remaining properties. Overall, these results quantitatively assess and validate the high quality of the concepts learned by the interpretable GNNs, highlighting the effectiveness of this approach for learning and analyzing complex algebraic structures.

Table 1: Concept purity scores of graph neural models in solving universal algebra's tasks. Lower is better.

	WEAK PURITY			STRONG PURITY		
	GNN	**iGNN**	**HiGNN**	**GNN**	**iGNN**	**HiGNN**
Distributive	3.30 ± 0.36	3.64 ± 0.30	3.09 ± 0.56	3.29 ± 0.38	4.00 ± 0.77	4.15 ± 0.67
Join Semi Distributive	2.38 ± 0.37	3.96 ± 0.51	3.74 ± 0.62	3.45 ± 0.34	3.98 ± 0.68	4.29 ± 0.61
Meet Semi Distributive	3.24 ± 0.63	3.55 ± 0.62	3.39 ± 0.29	3.36 ± 0.32	4.25 ± 0.39	4.97 ± 0.44
Modular	3.10 ± 0.35	3.50 ± 0.46	4.44 ± 0.56	3.14 ± 0.24	3.19 ± 1.01	4.25 ± 0.69
Semi Distributive	2.84 ± 0.51	3.70 ± 0.54	4.11 ± 0.46	3.70 ± 0.55	3.92 ± 0.28	4.08 ± 0.85

D.3 Concept visualization

Figure 6 visualizes 18 randomly sampled graph concepts (out of the 7896 graph concepts represented by different graph encodings) following the visualization procedure introduced by [7]. The figure

Table 2: Concept purity scores of different layers of HiGNN. Lower is better.

	WEAK PURITY		STRONG PURITY	
	2nd Layer	**Last Layer**	**2nd Layer**	**Last Layer**
Distributive	3.26 ± 0.43	3.09 ± 0.56	4.66 ± 0.98	4.15 ± 0.67
Join Semi Distributive	4.25 ± 0.69	3.74 ± 0.62	4.30 ± 0.39	4.29 ± 0.61
Meet Semi Distributive	3.64 ± 0.39	3.39 ± 0.29	4.41 ± 0.27	4.97 ± 0.44
Modular	3.89 ± 0.63	4.44 ± 0.56	4.19 ± 0.56	4.25 ± 0.69
Semi Distributive	3.55 ± 0.58	4.11 ± 0.46	3.16 ± 0.59	4.08 ± 0.85

Table 3: Concept completeness scores of graph neural models in solving universal algebra's tasks. Higher is better.

	WEAK COMPLETENESS		STRONG COMPLETENESS	
	iGNN	**HiGNN**	**iGNN**	**HiGNN**
Distributive	77.30 ± 0.20	76.60 ± 2.35	78.19 ± 1.71	73.16 ± 3.63
Join Semi Distributive	85.20 ± 0.86	86.84 ± 0.37	81.08 ± 0.18	79.76 ± 0.38
Meet Semi Distributive	84.21 ± 0.68	84.34 ± 1.08	80.86 ± 1.32	79.68 ± 0.22
Modular	76.98 ± 0.28	73.77 ± 3.14	81.36 ± 0.70	77.61 ± 2.37
Semi Distributive	87.33 ± 1.16	85.62 ± 0.27	84.03 ± 0.89	82.26 ± 0.21

shows for each concept an example of four (randomly sampled) graphs having the same concept label in the 7-th layer of the hierarchical iGNN trained on the multilabel dataset. Graphs belonging to the same concept show a coherency in their structure and similar patterns. These patterns represent the knowledge extracted and discovered by the hierarchical iGNN.

D.4 Explanations of post-hoc explainers

We compared our Explainable Hierarchical GNN against a standard explainer (namely GNNExplainer [8]) to further support our results. GNNExplainer is the first general, model-agnostic approach for providing interpretable explanations for predictions of any GNN-based model on any graph-based machine learning task and it is widely used in the scientific community as one of the staple explainers in GNN's XAI. In this particular setting, GNNExplainer was configured as follows: model-wise explanation on multiclass-node level classification task, with HiGNN as the model of choice, and GNNExplainer as the desired algorithm, trained for 200 epochs. The explainer takes as input a single graph in the dataset and outputs and explanation for its classification. GNNExplainer will enforce a classification based on the presence or omission of M_3 and/or N_5 and it is possible to visualize the subgraph that lead to this classification by leveraging the `visualize_graph` function. By doing this, we retrieve the following visualizations:

On the right, the substructure identified as $N5$ by `GNNExplainer` which lead to the classification of said graph as non modular and non distributive. On the right, in green $M3$. Our hierarchical model arrives to the same conclusions as the standard explainer but can also be augmented with a standard explainer.

E Code, Licences, Resources

Libraries. For our experiments, we implemented all baselines and methods in Python 3.7 and relied upon open-source libraries such as PyTorch 1.11 [36] (BSD license) and Scikit-learn [37] (BSD license). To produce the plots seen in this paper, we made use of Matplotlib 3.5 (BSD license). We will release all of the code required to recreate our experiments in an MIT-licensed public repository.

Resources. All of our experiments were run on a private machine with 8 Intel(R) Xeon(R) Gold 5218 CPUs (2.30GHz), 64GB of RAM, and 2 Quadro RTX 8000 Nvidia GPUs. We estimate that approximately 100-GPU hours were required to complete all of our experiments.

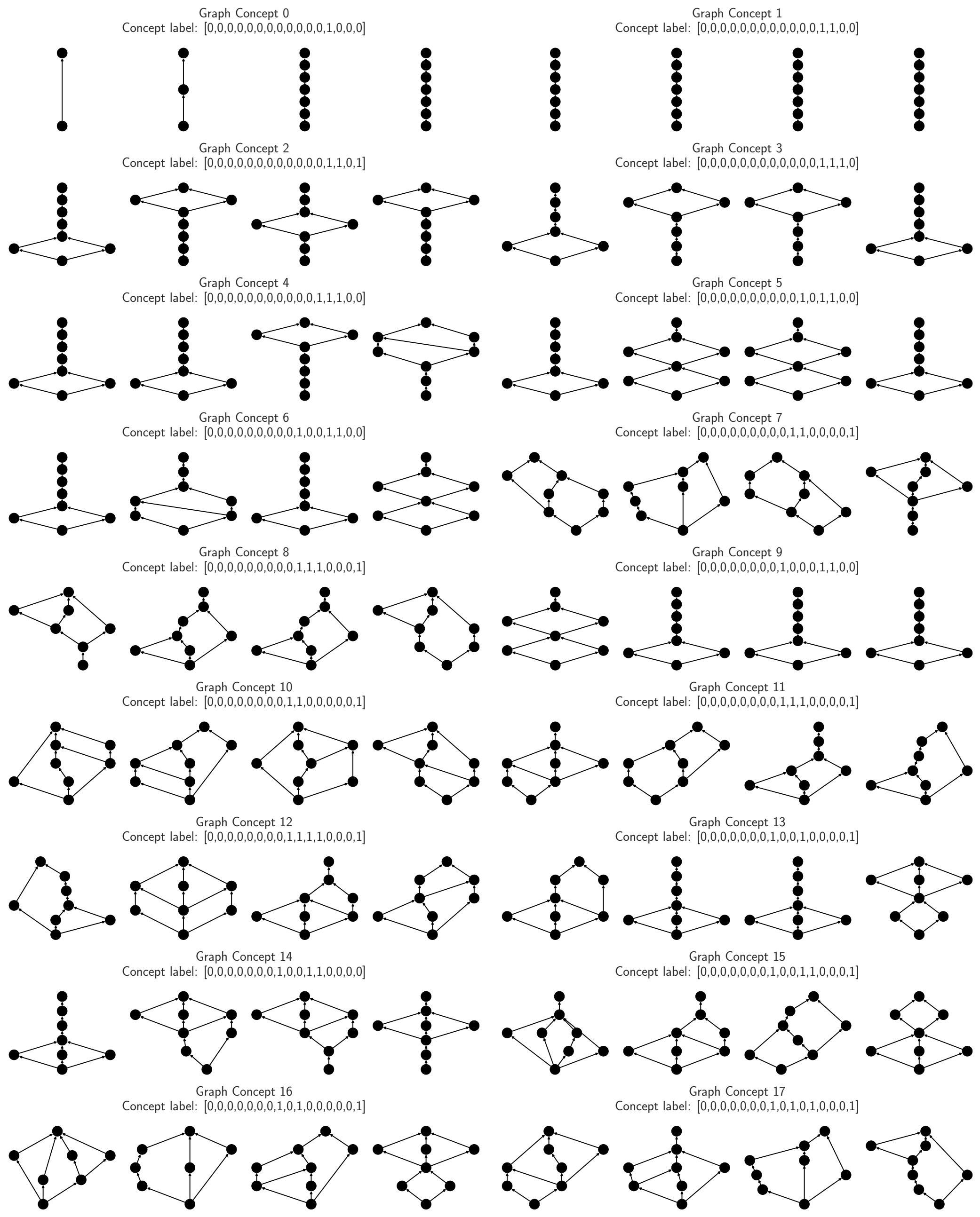
Graph Concept 0
Concept label: [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0]
Graph Concept 1
Concept label: [0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0]
Graph Concept 2
Concept label: [0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1]
Graph Concept 3
Concept label: [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0]
Graph Concept 4
Concept label: [0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0]
Graph Concept 5
Concept label: [0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0]
Graph Concept 6
Concept label: [0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0]
Graph Concept 7
Concept label: [0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1]
Graph Concept 8
Concept label: [0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,1]
Graph Concept 9
Concept label: [0,0,0,0,0,0,0,0,1,0,0,0,1,1,0,0]
Graph Concept 10
Concept label: [0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,1]
Graph Concept 11
Concept label: [0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,1]
Graph Concept 12
Concept label: [0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,1]
Graph Concept 13
Concept label: [0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1]
Graph Concept 14
Concept label: [0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,0]
Graph Concept 15
Concept label: [0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1]
Graph Concept 16
Concept label: [0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1]
Graph Concept 17
Concept label: [0,0,0,0,0,0,0,1,0,1,0,1,0,0,0,1]

Figure 6: Examples of graph concepts.

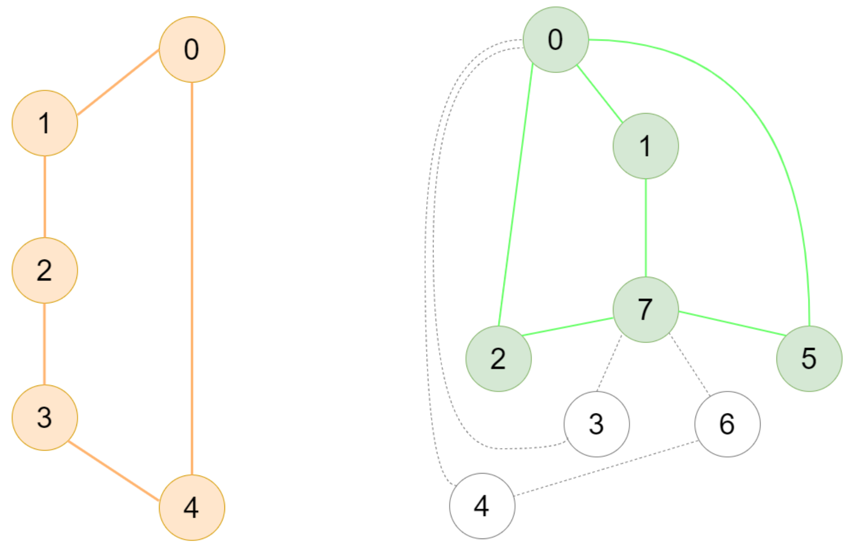

Figure 7: Visualizations obtained with GNNExplainer on weak distributive generalization (on the left) and strong multiclass generalization (on the right)