The Expressivity and Explainability of Graph Neural Networks



Matthew Morris

Hertford College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2026

Raynalde, non possum, quia omnia quae scripsi videntur mihi palae. Videntur mihi palae respectu eorum quae vidi et revelata sunt mihi.

Acknowledgements

First, I would like to extend my sincere thanks and deepest gratitude to my supervisors, Professor Ian Horrocks and Professor Bernardo Cuenca Grau. They both provided so much support, wisdom, training, and edits to my work over the last three and a half years, for which I am very grateful. To Ian, I am particularly grateful for the encouragement given in moments where I was struggling, for teaching me to constantly question why I am trying to answer a particular question and how my findings can be best presented, and for all his support in enabling me to attend many conferences. To Bernardo, I am especially thankful for the high quality and detailed feedback given on my work: I am leaving the doctorate a far better mathematician than when I started, largely due to his advice, wisdom, and critical eye on my work. Thank you also to my examiners, Professor Egor V. Kostylev and Professor Giuseppe De Giacomo. Special thanks to Egor who went beyond the call of duty in his careful reading of my work, and who led a stimulating and honest debate during the viva.

I was funded by an EPSRC scholarship (CS2122_EPSRC_1339049), without which I would have been unable to pursue these studies. As such, I would like to express my thanks to my funders, who made this all possible. I also benefitted from funding via my supervisors, so would like to acknowledge the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project 237889), Samsung Research UK, and the EPSRC projects AnaLOG (EP/P025943/1), OASIS (EP/S032347/1), UKFIRES (EP/S019111/1), and ConCur (EP/V050869/1).

To members of the KRR group in the Oxford Computer Science department, I offer thanks for their support and companionship. In particular, I would like to thank Xiaxia Wang for her friendship, Shuwen Liu for her advice on administration matters relating to the DPhil, Jiaoyan Chen and Ismail Ilkan Ceylan for their examining my transfer and confirmation of status (as well as to Ismail for his early advice on an idea that became a paper), and Przemyslaw Walega for his friendship and feedback on a paper. Special thanks are due to David Tena Cucala, whose prior work is central to that of my thesis, whose friendship and support was greatly appreciated, and whose feedback and advice on papers was invaluable. He is an incredible academic, and I am so thankful to have been able to work with him.

Thank you to my wonderful friends in Oxford, who gave such amazing support and encouragement over the last few years, and who are far too many to name. I

would particularly like to thank my friends from the Computer Science department, the Oratory, the Catholic chaplaincy, the Oxford Companions of the Order of Malta, the university Ultimate Frisbee club, Hertford College (especially my wonderful Frisbee team), and the Canterbury Institute. Special thanks also to the Canterbury Institute for providing a space for my education not only in Computer Science, but also in philosophy, theology, politics, physics, and simply being a more virtuous human being. To my spiritual fathers and friends, Fr Jack Green (particularly for his spiritual direction), Fr Joseph Murphy, Fr Felipe Cinelli, and all the other priests and religious in Oxford who have been such a blessing to me: thank you. Thank you also to my South African friends, who supported me so well from the other side of the world, and who always made me very excited to visit South Africa.

I would also like to single out several particular friends to thank, including my housemates Francesco Banfi, Fernando Contreras, and Nathan Coundon, who formed the core of the amazing community I had here in Oxford, and with whom I shared many late-night conversations about life, love, and the Lord. My gratitude also to my dear friends William Nolan and Chase Mizzel, who I am very blessed to have in my life. Finally, my thanks also to Desmond Fairall and Jonathan Crowther, who the winds of providence brought to Oxford from South Africa, and whose deep friendship I have appreciated for the last 25 and 15 years, respectively.

To my Hannah: I cannot express in words how grateful I am for your love, support, and encouragement in the tail-end of this doctorate. You have been incredibly generous in your comforting me when I was frustrated, your gentle encouragement for my writing, your listening to me talking about my work, and all the time you spent proofreading at the end. I love you and am so thankful for you.

To my incredible family. First, thank you to Juliet for the fun times together in South Africa, the many wonderful phone calls, and for visiting me in Oxford. Second, to Emily: what a privilege to have been able to study together at Oxford. I am so grateful for all the dinners, fun times, and mutual support we were able to provide for each other whilst here. To my mom and dad: you raised me, you taught me the values I have today, you gave me an incredible education, you have supported and encouraged me my entire life (including during this doctorate), and you have always provided a safe and happy place to come home to. Thank you dad for proofreading my entire thesis, and thank you mom for visiting me in Oxford and reviewing part of my thesis. Thank you, Nonna, for all your selfless love: may you rest in peace. I love you all.

To my Creator, my Redeemer, and my King: may this work of my hands be fruitful, as I surrender it entirely into your care. All praise and glory and honour to you, Lord Jesus.

Abstract

Information stored as unary and binary facts is often referred to as a knowledge graph (KG). KGs are used in many applications, and different methods for learning on them have been proposed. Among these approaches, graph neural networks (GNNs) are effective and widely-used, due to their strong structural inductive biases. Message Passing Neural Networks (MPNNs) are a powerful variety of GNNs, of which Graph Convolutional Networks (GCNs) are a common instantiation. However, there are many open questions pertaining to the expressivity of GNNs: i.e., which functions GNNs are able to represent. A prominent subset of methods for analysing GNN expressivity is to compare them to formal logic. Furthermore, GNNs are black boxes that do not readily provide explanations for their predictions. Explainability and interpretability are crucial for models that operate in high-stakes domains. Thus, various methods seek to provide explanations for GNN predictions, including explanations based on logical rules. However, these rules are often not faithful to the predictions of the model, and there is increasing interest in logical explanations that faithfully characterize the predictions of GNNs. Such explanations have been shown to be attainable by restricting the weights of GCN models to be non-negative: these GCNs are called "monotonic".

In this thesis, I consider the expressivity and explainability of different MPNNs. First, I define a new class of graph functions that allow for symmetry-breaking and propose extensions to MPNNs to learn such functions. Next, I consider monotonic GCNs used with scoring functions for predicting binary facts, providing an upper-bound on their logical expressivity and recovering sound rules for predictions. I also relax the constraint of non-negative weights for GCNs with sum aggregation, showing that, without adaptions to their training regime, they provably learn no sound rules in practice. Finally, I also consider monotonic GCNs with mean aggregation, proving the limited class of simple monotonic rules that can be sound for them and providing a rule language for sound explanations. My experiments validate my theoretical results, showing that my novel MPNNs can learn symmetry-breaking graph functions, that monotonic GCNs still perform well on benchmark datasets, and that sound rules for explanations can be recovered in practice.

Contents

List of Abbreviations and Notation **xi**

1 Introduction **1**
1.1 Learning on Knowledge Graphs 1
1.2 Graph Neural Networks 2
1.3 My Contributions and Thesis Outline 6
1.4 Publications 8

2 Preliminaries **10**
2.1 Formal Logic 11
2.1.1 Datalog 11
2.1.2 First-Order Logic 12
2.1.3 Description Logics 13
2.2 Graph Theory 13
2.2.1 Vectors, Matrices, and Functions 13
2.2.2 Graphs 14
2.2.3 Graph Isomorphism and Automorphism 14
2.2.4 Graph Functions 15
2.3 Graph Neural Networks 16
2.3.1 Message Passing Neural Networks 17
2.3.2 Specific Graph Neural Networks 18
2.3.3 Scoring Functions 19
2.4 GCNs as Logical Functions 20
2.4.1 Correspondence Between Graphs and Datasets 20
2.4.2 GCN Dataset Transformations and Sound Rules 20
2.4.3 Link Prediction 22
2.4.4 Monotonicity 23
2.4.5 Relationship to Barcelo-style Expressivity 25
2.5 Experiments 26
2.5.1 Experimental Setup 27
2.5.2 Metrics 27
2.5.3 Datasets 29

3 Literature Review **31**
3.1 Non-GNN Approaches to Learning on KGs . . . 32
3.2 Graph Neural Networks . . . 35
3.2.1 Important Graph Neural Networks . . . 35
3.2.2 Knowledge Graph Encodings . . . 38
3.3 Expressivity of GNNs . . . 40
3.3.1 Graph Separation Power . . . 40
3.3.2 Approximation Ability . . . 42
3.3.3 Substructure Counting . . . 43
3.3.4 Other Non-Logic-Based Methods . . . 44
3.3.5 Formal Logic . . . 45
3.4 Explainability of GNNs . . . 49
3.4.1 Post-Hoc GNN Explainability . . . 50
3.4.2 Self-Explainable GNNs . . . 53
3.4.3 Logic-Based Self-Explainable GNNs . . . 55

4 Orbit-Equivariant Graph Neural Networks **57**
4.1 Introduction . . . 57
4.2 Orbit-Equivariance . . . 60
4.3 Non-Equivariant MPNN Architectures . . . 65
4.4 Experiments . . . 71
4.4.1 Datasets . . . 71
4.4.2 Loss . . . 73
4.4.3 Methodology . . . 74
4.4.4 Results . . . 75
4.5 Discussion . . . 78

5 Expressivity and Explanations for GCNs with Scoring Functions **79**
5.1 Introduction . . . 79
5.2 Sound Rules . . . 82
5.2.1 Monotonically Increasing Scoring Functions . . . 82
5.2.2 Checking Soundness of Rules . . . 84
5.2.3 Leveraging Scoring Function Results . . . 85
5.3 Equivalent Programs . . . 88
5.3.1 Monotonic Max-GCNs . . . 90
5.3.2 Monotonic Max-Sum-GCNs . . . 92
5.4 Experiments . . . 95
5.4.1 Experiment Setup . . . 95
5.4.2 Results . . . 98
5.5 Discussion . . . 100

6 Relational Graph Convolutional Networks Do Not Learn Sound Rules **101**
6.1 Introduction . . . 101
6.2 Partitioning the Channels of a GCN . . . 104
6.2.1 Safe Channels . . . 104
6.2.2 Stable and Increasing Channels . . . 108
6.2.3 Unbounded Channels . . . 111
6.3 Experiments . . . 115
6.3.1 Experiment Setup . . . 116
6.3.2 Training Paradigm Variations . . . 117
6.3.3 Results . . . 118
6.4 Discussion . . . 121

7 Sound Logical Explanations for Mean Aggregation GCNs **123**
7.1 Introduction . . . 123
7.2 Sound Rules and Explanations . . . 126
7.2.1 Relationship of MAGCNs with FOL . . . 126
7.2.2 Which Monotonic Rules can be Sound for MAGCNs . . . 128
7.2.3 Explaining MAGCN Predictions . . . 132
7.3 Experiments . . . 135
7.3.1 Experiment Setup . . . 135
7.3.2 Results . . . 137
7.4 Discussion . . . 140

8 Conclusions **143**
8.1 Summary . . . 143
8.2 Future Work . . . 146

Appendices

A Appendices for Chapter 4 **150**
A.1 Proofs . . . 151
A.1.1 Proposition 3 . . . 151
A.1.2 Proposition 5 . . . 151
A.1.3 Proposition 7 . . . 153
A.1.4 Theorem 8 . . . 154
A.1.5 Theorem 9 . . . 156
A.2 Orbit-1-WL . . . 170
A.2.1 Definition . . . 170
A.2.2 Successful Examples . . . 170

A.2.3 Counterexamples for Completeness . . . 171
A.2.4 Orbit-k-WL . . . 173
A.3 Experiments . . . 174
A.3.1 Computational Complexity . . . 174
A.3.2 Datasets . . . 174
A.3.2.1 Bioisostere . . . 175
A.3.2.2 Alchemy-Max-Orbit . . . 176
A.3.3 Hyperparameters . . . 177
A.3.4 Loss . . . 178
A.3.5 Full Results . . . 181
A.3.5.1 Bioisostere . . . 182
A.3.5.2 Alchemy-Max-Orbit-2 . . . 184
A.3.5.3 Alchemy-Max-Orbit-6 . . . 186

B Appendices for Chapter 5 **188**
B.1 Theory . . . 189
B.1.1 Non Monotonically Increasing Scoring Functions . . . 189
B.1.2 Monotonically Increasing Semantic Matching Models . . . 189
B.1.3 Monotonically Increasing Neural Network Models . . . 191
B.1.4 Initial Node Features without Unary Predicates . . . 192
B.2 Full Proofs . . . 194
B.2.1 Lemma 45 . . . 194
B.2.2 Lemma 11 . . . 194
B.2.3 Proposition 12 . . . 195
B.2.4 Proposition 15 . . . 197
B.2.5 Corollary 16 . . . 198
B.2.6 Corollary 17 . . . 198
B.2.7 Theorem 20 . . . 199
B.2.8 Capacity Setup . . . 205
B.2.9 Theorem 23 . . . 207
B.2.10 Theorem 24 . . . 209
B.3 Full Experiment Results . . . 217

C Appendices for Chapter 6 **221**
C.1 Full Proofs . . . 222
C.1.1 Lemma 29 . . . 222
C.1.2 Proposition 30 . . . 224
C.1.3 Theorem 31 . . . 225
C.1.4 Theorem 33 . . . 228
C.1.5 Theorem 34 . . . 233
C.2 More Details About Experiments . . . 235

D Appendices for Chapter 7 **236**
D.1 Proofs 237
D.1.1 Theorem 36 237
D.1.2 Proposition 37 240
D.1.3 Lemma 49 242
D.1.4 Theorem 39 243
D.2 Algorithms 248
D.2.1 Checking Soundness of ELUQ rules 248
D.2.2 Refining Explanations 248
D.3 Full Results 250

Bibliography **256**

List of Abbreviations and Notation

KG	Knowledge Graph
$\delta \in \mathbb{N}$	Number of unary predicates in the fixed signature
Dataset	Finite set of facts
$\mathtt{con}(D)$	Set of constants of a dataset D
Program	Finite set of rules
T_r	Immediate consequence operator for a rule r
$T_{\mathcal{P}}$	Immediate consequence operator for a program $\mathcal{P}$
FOL	First-order logic
Vectors	Denoted with round brackets: for example, $\mathbf{v} = (1, 2, 3, 4)$
$\mathbf{v}[i]$	Denotes the i-th element of vector $\mathbf{v}$
$\mathbf{A}[i, j]$	Denotes the element in row i and column j of matrix $\mathbf{A}$
Col	Fixed finite set of colours
Labels	Also referred to as "features" or "attributes"
$V(G)$	Denotes the vertices of a graph G
$E(G)$ **or** $E^c(G)$	Denotes the edges of a graph G
λ_G	Denotes the node labelling of a graph G
v^a	Vertex corresponding to constant a
$\mathbf{v}$ **for vertex** v	Label $\lambda(v)$ of vertex $v \in V(G)$
G_v **for vertex** v	Label $\lambda(v)$ of vertex $v \in V(G)$
GNN	Graph Neural Network
MPNN	Message Passing Neural Network
$f_{\mathbf{MPNN}}$	An MPNN
$f_\ell^{\mathbf{update}}$	MPNN update function
$f_\ell^{\mathbf{aggr}}$	MPNN aggregation function
$f_\ell^{\mathbf{read}}$	MPNN readout function

$\mathbf{v}_\ell$	Label of v after ℓ layers of an MPNN
GCN	Graph Convolutional Network
σ_ℓ	Activation function for layer ℓ of a GCN
$\mathbf{agg}_\ell$	Aggregation function for layer ℓ of a GCN
$\mathtt{cls}_t$	GCN step classification function with threshold $t \in \mathbb{R}$
$\mathcal{M}$	Typically denotes a GCN
sum-GCN	GCN that uses sum aggregation
MAGCN	Monotonic mean-GCN
d_f	Dimension of scoring function f
t_f	Threshold of scoring function f
$\mathtt{enc}(D)$	Canonical encoding of a dataset D
$\mathtt{dec}$	Canonical decoding
$T_{\mathcal{M}}$	Canonical dataset transformation induced by a GCN $\mathcal{M}$
$\mathtt{dec}_f$	Decoding using scoring function f
$T_{\mathcal{M},f}$	Transformation using GCN $\mathcal{M}$ and scoring function f

1 Introduction

Contents

1.1 Learning on Knowledge Graphs 1
1.2 Graph Neural Networks 2
1.3 My Contributions and Thesis Outline 6
1.4 Publications 8

1.1 Learning on Knowledge Graphs

When a knowledge base consists of only unary and binary facts, and can be expressed as a graph, it is often referred to as a *knowledge graph* (KG) – structured information is often stored in this way. In such a graph, nodes correspond to constants in the facts, node features correspond to unary facts, edges correspond to binary facts, and edge colours correspond to binary predicates. KGs have found use in diverse application areas, including general purpose information storage [272], semantic relationships between words [247], familial relations [132], geographical data [36], biomedical concepts [132], protein interactions [112], question answering [33, 35], semantic parsing [26, 115], disambiguating between named constants [72, 335], recommender systems [278], natural language processing [302], and information extraction [70, 118].

Machine learning methods have been applied to KGs in a variety of ways and for different purposes. As a prominent example, despite the prevalence and size of modern KGs, many remain incomplete and are missing true facts [54, 289, 319]. Thus, the task of KG completion arises, where the goal is to augment the existing KG through predicting new facts. Other important tasks include constant classification [197, 198], relation extraction [223, 290], graph classification [99, 300], node clustering / community detection [105, 208], rule mining [91, 179], and constant resolution [34, 197]. The evaluations in this thesis primarily focus on KG completion, although many of the techniques and models can be used for other tasks as well.

A number of different approaches have been proposed for learning on KGs. Some of these include symbolic and statistical methods, such as Inductive Logic Programming [62, 117, 195], rule mining [91, 179], and Statistical Relational Learning [16, 98, 165]. Other approaches use learning methods based on gradient descent, such as scoring functions, which can be distance-based [1, 32, 249] or semantic matching models [17, 159, 197]. Other examples of gradient descent -based learning methods include those using random walks [105, 214], Convolutional Neural Networks [75, 76], sequence models [107, 227], differentiable reasoning [83, 224, 304], and Large Language Models [51, 336]. Among these approaches, graph neural networks (GNNs) have emerged as an effective and widely-used model architecture.

1.2 Graph Neural Networks

Graph Neural Networks (GNNs) [48] are a class of neural network methods that operate upon graphs, and are the primary model architecture considered in my thesis. They use stacked layers of aggregation and combination operations that pool information from neighbouring nodes. GNNs have found popular use for learning on KGs and performing KG completion, since the input data is naturally structured as a graph. They have been successfully applied across a wide range of domains, including chemistry [80, 87, 153], healthcare [74, 88], computer vision [143, 252, 283], traffic [306, 323], anomaly detection [293], recommender systems [185, 310], combinatorial optimization [25], and natural language processing [173, 174].

Different GNN Models GNNs vary based on their architecture and the encoding of the KG that they are applied to. Commonly used GNNs such as GCNs [138, 234], GraphSAGE [112], and GATs [268] are defined within the Message Passing framework [99, 102, 294]: I refer to such models as Message Passing Neural Networks (MPNNs). MPNNs leverage local message-passing operations that are invariant to node permutations so that the resulting output does not fundamentally depend on the particular nodes in the input graph, but only the features and graph structure. When the MPNN yields an output for each node, this property is called "equivariance". This strong inductive bias enables MPNNs to effectively and efficiently learn functions over graphs [21, 42].

Graph Convolutional Networks (GCNs) [80, 138] are a common instantiation of MPNNs. I use the term "GCN" to refer to a particular class of MPNN models described in Section 2.3.2, which allow for different aggregation and activation functions, and aggregate graph edges of each distinct colour separately. A model subsumed under GCNs is R-GCN [234], a popular and influential GNN which has many extensions [45, 161, 238, 259, 264, 313] and remains a common baseline for evaluating against or as part of a larger system [110, 152, 156, 162, 251, 325]. Within the class of GCN models, there are different options for how information is aggregated from neighbouring nodes. The most common choices are sum [19], max [254], mean [234], and attention [268]. I refer to GCNs using sum, max, and mean aggregation as sum-GCNs, max-GCNs, and mean-GCNs respectively.

Expressivity There are many open questions regarding which functions GNNs are able to represent, often referred to as their "expressivity". This has been approached in various ways, including the ability of GNNs to distinguish between different non-isomorphic graphs [37, 186, 300], approximate functions [2, 12, 177], and count substructures [55, 205, 322].

Expressivity is an important property for several reasons. First, if a model has restricted expressivity and the function the model is trying to learn also falls within the restriction, then it can serve as a useful inductive bias. For example, the

equivariance of MPNNs can be considered a limitation in their expressivity, as they can only learn equivariant functions. However, it is a useful limitation, since most practical functions on graphs are equivariant. There is also evidence that more expressive models can struggle to generalize [178]. On the other hand, studying expressivity has exposed the limits of GNNs [186, 300] and thus inspired extensions to GNNs to learn richer classes of functions [28, 37, 176, 188].

A prominent subset of methods for analysing GNN expressivity is to compare them to formal logic. In a foundational paper, Barceló et al. [19] consider GNNs as logical classifiers and compare them to logical classifiers from first-order logic (FOL). Within related work that is foundational to this thesis, Tena Cucala et al. [254, 256] consider GNNs as computing transformations over logical datasets, where the transformations can be represented as the immediate consequences of Datalog programs. In particular, the class of *monotonic max-GCNs* [254], which restrict all matrix weights in the model to be non-negative and impose certain requirements on activation and classification functions, can be faithfully captured by means of tree-like Datalog programs. Likewise, *monotonic max-sum-GCNs* [256], which relax the requirements on the aggregation function to encompass both max and sum aggregation, can be faithfully characterized by means of tree-like Datalog programs with inequalities in the body of rules. These results provide upper bounds on the logical expressivity of these GCNs.

Explainability Despite their success for learning on graphs, GNNs and other neural models are black boxes that do not readily provide explanations for their predictions [95]. Explainability and interpretability are crucial for models that operate in high-stakes domains, such as precision medicine, drug discovery, and autonomous vehicles [9, 318]. Furthermore, explanations can provide novel insights about a particular domain that can be leveraged by human experts. It is also important for domains that require compliance with external constraints and legislation, such as criminal justice, security, and finance. The domain of explainable AI [78] seeks to address these concerns by providing explanations

for model predictions, and has been widely used across various fields of machine learning. Explainable methods have been used for GNNs in different application areas, including biomedical [121], neuroscientific [334], molecular sciences [5], drug discovery [331], material sciences [68], environmental [71], transportation [263], energy [269], fraud detection [314], and software security [46].

A popular method for providing explanations for models that learn over KGs is through the use of logical rules [201, 210, 217, 254, 256]. Two prominent examples of neural models for KGs are Neural-LP [304] and DRUM [227], which both extract Datalog rules from the models that serve as explanations. As shown by Tena Cucala et al. [255], Wang et al. [282], however, the extracted Datalog rules are not faithful to the model in the sense that the rules may derive, for some KG, facts that are not predicted by the model (*unsoundness*) as well as failing to derive predicted facts (*incompleteness*). Thus, there is increasing interest in the development of neural KG completion methods whose predictions can be faithfully characterized by means of rule-based reasoning. As shown by Tena Cucala et al. [255], Wang et al. [282], the Neural-LP and DRUM approaches can be adapted to ensure soundness and completeness of the generated rules with respect to the model.

When it comes to GNNs, the works of Tena Cucala et al. [254, 256], which we discussed above, both allow for the extraction of sound rules from the GNNs, which serve as robust and provably faithful explanations. Other works also extract logical explanations for GNN predictions [6, 13, 139, 210]. However, these do not provide theoretical guarantees that the explanations are sound for the model, only empirical evidence that they are approximately faithful. Pluska et al. [210], for example, learn decision trees from GNNs and then extract logical rules from the decision trees — they are not equivalent to the GNNs they are derived from, so the extracted logical rules are not provably sound.

Logical rules that are sound for a GNN serve as explanations for the facts they predict on logical datasets, since they prove that whenever the body of the rule is satisfied in an input dataset, the GNN will always make a prediction

corresponding to the head of the rule. The usefulness of sound rules are explanations is further discussed in Section 2.4.2.

1.3 My Contributions and Thesis Outline

In my thesis, I consider the expressivity of MPNNs in relation to equivariance, and the logical expressivity and explainability of monotonic GCNs [256] when various restrictions are relaxed, or when they are combined with scoring functions. I find that MPNNs can be extended to express a richer class of functions beyond equivariance, and that the extraction of sound logical rules from GCN models becomes more difficult (and sometimes impossible for certain classes of rules) as their monotonicity restrictions are relaxed.

First, in Chapter 2, I cover the preliminaries needed for most of the theory presented in my thesis. In particular, I cover topics pertaining to formal logic, graph theory, graph neural networks, and the relationship between GCNs and formal logic. I also describe common features and datasets of the experiments run in my thesis.

Next, in Chapter 3, I provide a literature review of papers relevant to the work of my thesis. I summarize various non-GNN approaches to learning on KGs, discuss some important GNN models, compare works that consider the expressivity of GNNs, and finally cover a number of methods for yielding explanations for GNN predictions.

In Chapter 4, I consider the expressivity of MPNNs with respect to equivariant functions. Equivariance is an important structural property that is captured by architectures such as MPNNs. However, equivariant graph functions cannot produce different outputs for similar nodes, which may be undesirable when the function is trying to optimize some global graph property. Thus, I define orbit-equivariance, a relaxation of equivariance which allows for such functions whilst retaining important structural inductive biases. I situate the property in the hierarchy of graph functions, define a taxonomy of orbit-equivariant functions, and provide four different ways to achieve non-equivariant MPNNs. For each, I analyse their expressivity with respect to orbit-equivariance and evaluate them on two novel datasets, one of which stems from a real-world use case of designing optimal bioisosteres.

In Chapter 5 and onwards, I consider a more restricted class of MPNNs – GCNs – to perform a more fine-grained analysis of their expressivity and to extract sound logical rules from them for explainability. Given the lack of explainability of GCNs on KGs, recent work [254, 256] has used sound Datalog rules and equivalent programs for explanations, as well as descriptions of the expressivity for various models. However, this work considers a limited form of link prediction relying on special encodings. Thus, in Chapter 5, I consider a popular method for performing link prediction with GCNs, where a scoring function is used to decode the GCN output into predictions for candidate binary facts. I show how GCNs and scoring functions can be adapted to be monotonic (by restricting them to have non-negative weights, among other constraints), use the monotonicity to extract sound rules for explaining predictions, and leverage existing results about which rules scoring functions can capture. I also define procedures for obtaining equivalent programs for certain classes of monotonic GCNs with scoring functions. My experiments show that, on link prediction benchmarks, monotonic GCNs and scoring functions perform well in practice and yield many sound rules.

Whilst the restriction of having only non-negative weights allows one to recover sound rules for explanations, it is not used in practice for standard and popular GCNs. Furthermore, it may be too restrictive in scenarios where the function required to solve the task is not monotonic. Thus, in Chapter 6, I consider sum-GCNs, which do not impose restrictions on model weights. The functions learnt by sum-GCNs may be non-monotonic: predicted facts may be invalidated by adding new facts to the input KG. As a result, these GCNs cannot, in general, be faithfully captured by (negation-free) Datalog programs. However, some subset of their predictions may still be explainable by sound Datalog rules. Thus, I provide two methods to extract sound rules that explain the prediction of sum-GCNs. Furthermore, I provide a method that can verify that certain classes of Datalog rules are provably not sound for a sum-GCN. In my experiments, I train sum-GCNs on KG completion benchmarks, and I am able to verify that no Datalog rule is sound for these models, even though the models often obtain high to near-perfect accuracy.

This raises some concerns about the ability of sum-GCN models to generalize and about the explainability of their predictions. This stands in stark contrast to the monotonic GCNs in Chapter 5, whose entire operation could be characterized by sound Datalog rules. To address this, I provide a variation to the training paradigm of sum-GCN that encourage it to learn sound rules, and find a trade-off between model accuracy and the number of learned sound rules.

Whilst Chapter 5 considers purely monotonic GCNs, in Chapter 6 I relax the monotonicity requirement by considering sum-GCNs with negative weights. In Chapter 7, I consider a different variation of GCNs that leads to them being non-monotonic: using mean as the aggregation function. In practice, mean aggregation often emerges as the default choice [112, 138, 225, 234, 294] due to its simplicity, stability during training, and good empirical performance. Despite the prevalence of GNNs that use mean as an aggregation function, explainability and expressivity results are lacking for them. Thus, in Chapter 7, I consider GCNs with mean aggregation and non-negative weights (MAGCNs), proving the precise class of monotonic rules that can be sound for them. These simple monotonic rules provide explanations for a subset of the MAGCN predictions. I also provide a restricted fragment of first-order logic containing rules that can explain any MAGCN prediction. My experiments show that restricting mean-GCNs to have non-negative weights yields comparable or improved performance on standard inductive benchmarks, that sound monotonic rules are obtained when using datasets with monotonic structures, that insightful explanations can be generated in practice, and that the sound rules can expose issues in the trained models.

Finally, in Chapter 8, I summarize the thesis with a brief review of the contributions, draw some overall conclusions, and discuss potential for future work.

1.4 Publications

This thesis is based on my publications listed below:

- Chapter 4 is based on Morris et al. [192]: Matthew Morris, Bernardo Cuenca Grau, and Ian Horrocks. Orbit-equivariant graph neural networks. In *International Conference on Learning Representations*, pages 5221–5259, 2024

- Chapter 5 is based on Morris et al. [194]: Matthew Morris, David Tena Cucala, and Bernardo Cuenca Grau. Logical expressivity and explanations for monotonic GNNs with scoring functions. In *Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning*, volume 22, pages 477–488, 2025

- Chapter 6 is based on Morris et al. [193]: Matthew Morris, David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. Relational graph convolutional networks do not learn sound rules. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning*, volume 21, pages 897–908, 2024

- Chapter 7 is based on Morris and Horrocks [190]: Matthew Morris and Ian Horrocks. Sound logical explanations for mean aggregation graph neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025

2 Preliminaries

Contents

2.1 Formal Logic 11
2.1.1 Datalog 11
2.1.2 First-Order Logic 12
2.1.3 Description Logics 13
2.2 Graph Theory 13
2.2.1 Vectors, Matrices, and Functions 13
2.2.2 Graphs 14
2.2.3 Graph Isomorphism and Automorphism 14
2.2.4 Graph Functions 15
2.3 Graph Neural Networks 16
2.3.1 Message Passing Neural Networks 17
2.3.2 Specific Graph Neural Networks 18
2.3.3 Scoring Functions 19
2.4 GCNs as Logical Functions 20
2.4.1 Correspondence Between Graphs and Datasets 20
2.4.2 GCN Dataset Transformations and Sound Rules 20
2.4.3 Link Prediction 22
2.4.4 Monotonicity 23
2.4.5 Relationship to Barcelo-style Expressivity 25
2.5 Experiments 26
2.5.1 Experimental Setup 27
2.5.2 Metrics 27
2.5.3 Datasets 29

In this chapter, I cover the preliminaries needed for most of the theory presented in my thesis. First, I cover topics in formal logic, particularly Datalog, first-order

logic, and description logics. Next, I cover some graph theory, culminating in functions on graphs and their properties. Then I introduce Graph Neural Networks (GNNs), providing a formal definition of a broad class of message-passing GNNs, defining GCNS (a specific GNN architecture), and laying out the background for scoring functions, which are used with GCNs to perform link prediction. Next, I cover the relationship between GCNs and formal logic, showing the correspondence between graphs and sets of logical facts, defining notions of sound, complete, and equivalent logical rules for GCNs, illustrating how link prediction can be performed, summarizing how monotonicity properties can be used for rule extraction, and explaining the relationship between the style of logical expressivity used in this thesis and another prominent style in the literature. Finally, I describe common features of the experiments run in my thesis, define metrics that I use for the evaluation of GNN models, and introduce the KGs/datasets on which I run experiments.

2.1 Formal Logic

2.1.1 Datalog

We fix a signature of $\delta \in \mathbb{N}$ unary predicates and a finite but arbitrary set of binary predicates (also referred to as relations). We also consider countably infinite sets of variables and constants, disjoint with one another and the set of predicates. A term is a variable or a constant. An atom is an expression of the form $R(t_1, ..., t_n)$, where each t_i is a term and R is a predicate with non-negative integer arity $n \in \{1, 2\}$.

A literal is an atom or any inequality $t_1 \not\approx t_2$ for terms t_1, t_2. A literal is ground if it contains no variables. A fact is a ground atom, and a dataset D is a finite set of facts. We denote the set of constants of a dataset D with $\texttt{con}(D)$. A (Datalog) rule is a constant-free expression of the form

$$B_1 \wedge ... \wedge B_n \rightarrow H, \tag{2.1}$$

where $n \in \mathbb{N}_0$, $B_1, ..., B_n$ are its body literals, and H is its head atom. We do not require rules to be safe: there may be variables in the head that do not occur in a body atom. Furthermore, to avoid redundant rules, we require that each inequality

in the body of a rule mentions two different terms. A (Datalog) program is a finite set of rules. A substitution ν maps finitely many variables to constants. For literal α and a substitution ν defined on each variable in α, $\alpha\nu$ is obtained by replacing each occurrence of a variable x in α with $\nu(x)$.

For a dataset D and a fact B, we write $D \models B$ if $B \in D$; furthermore, given constants a_1 and a_2, we write $D \models a_1 \not\approx a_2$ if $a_1 \neq a_2$, for uniformity. We write $D \models A$ for a set or conjunction of facts A if D contains each fact in A. The immediate consequence operator T_r for a rule r of form (2.1) maps a dataset D to dataset $T_r(D)$ containing $H\nu$ for each substitution ν such that $D \models B_i\nu$ for each $i \in \{1, \ldots, n\}$. For a program $\mathcal{P}$, $T_\mathcal{P}(D) = \bigcup_{r\in\mathcal{P}} T_r(D)$. Notice that T_r and $T_\mathcal{P}$ are transformations from datasets to datasets.

2.1.2 First-Order Logic

We also extend these ideas to first-order logic (FOL). We use standard FOL syntax [180], including equality, allowing for the following operators: $\forall, \exists, \neg, \wedge, \vee, =$. Variables in a FOL formula are free if they do not have a matching quantifier. Barceló et al. [19] define FOL classifiers over coloured graphs (see Section 2.2.2), where for graph G, node v, and FOL formula φ with a free variable x, $(G, v) \models \varphi$ denotes φ being satisfied by node v of G (i.e. v satisfies φ when "substituted" for x). Similarly, for a dataset D, constant $a \in \texttt{con}(D)$, and FOL formula φ with free variable x, we let $(D, a) \models \varphi$ denote φ being satisfied by constant a of D. More precisely, $(D, a) \models \varphi$ if for variable substitution μ mapping $x \mapsto a$, we have $D \models \varphi\mu$ (under the standard Tarskian semantics). As usual in this setting, D is treated as a finite Herbrand interpretation and $D \models \varphi\mu$ iff $\varphi\mu$ evaluates to true in D.

We define a FOL *rule* to be of the form $B \to A(x)$, where B is any FOL formula with a single free variable x and A a unary predicate. A (FOL) program is a finite set of (FOL) rules. Given a dataset D and FOL rule $r : B \to A(x)$, we define the immediate consequence operator $T_r(D) = \{A(a) \mid a \in \texttt{con}(D), (D, a) \models B\}$. For a program $\mathcal{P}$, we likewise define $T_\mathcal{P}(D) = \bigcup_{r\in\mathcal{P}} T_r(D)$.

2.1.3 Description Logics

We also consider fragments of FOL defined by description logics such as $\mathcal{ALCQ}$ [15], the concepts of which are defined by:

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall P.C \mid \exists P.C \mid \ \leq_n P.C \mid \ \geq_n P.C,$$

for any relation[1] P (binary predicate), $\mathcal{ALCQ}$ concepts C, D, and atomic concept (unary predicate) A. We define an $\mathcal{ALCQ}$ rule to be a subsumption of the form $C \sqsubseteq A$. Immediate consequences are defined in the same way as for FOL, since every $\mathcal{ALCQ}$ concept corresponds to a FOL formula with one free variable. I.e. given a dataset D and $\mathcal{ALCQ}$ rule $r : C \sqsubseteq A$, we define the immediate consequence operator $T_r(D) = \{A(a) \mid a \in \texttt{con}(D), (D, a) \models C\}$.

For example, the $\mathcal{ALCQ}$ concept B, defined as

$$\text{Car} \sqcap \ \geq_2 \text{hasTire.Red} \ \sqcap \ \exists \text{hasSpoiler.}\top,$$

identifies every car with at least two red tires and a spoiler. Consider rule $r : B \sqsubseteq \text{SportsCar}$ and dataset

$$D = \{\text{Car}(a), \text{Car}(d), \text{hasTire}(a, b), \text{hasTire}(a, c), \text{Red}(b), \text{Red}(c), \text{hasSpoiler}(a, e)\}.$$

Then $T_r(D) = \{\text{SportsCar}(a)\}$.

2.2 Graph Theory

2.2.1 Vectors, Matrices, and Functions

We consider real-valued vectors, matrices, and higher-order tensors, which we denote with bold letters. We use round brackets for vectors and matrices: for example, $\mathbf{v} = (1, 2, 3, 4)$. For $\mathbf{v}$ a vector and $i > 0$, $\mathbf{v}[i]$ denotes the i-th element of $\mathbf{v}$. For $\mathbf{A}$ a matrix and $i, j > 0$, $\mathbf{A}[i, j]$ denotes the element in row i and column

[1] In description logics, relations are often referred to as "roles".

j of $\mathbf{A}$ (this notation generalizes to higher-order tensors). A function $\sigma : \mathbb{R} \to \mathbb{R}$ is *monotonically increasing* if $x \leq y$ implies $\sigma(x) \leq \sigma(y)$. We apply univariate functions to vectors element-wise.

2.2.2 Graphs

We fix a finite set Col of colours, with |Col| equal to the number of binary predicates in the fixed signature defined in Section 2.1.1. A graph G is a tuple $(V, \{E^c\}_{c\in\text{Col}}, \lambda)$ where V is a finite vertex (node) set, each $E^c \subseteq V \times V$ is a set of directed edges with the colour c, and λ assigns to each $v \in V$ a real vector label of dimension δ (the number of unary predicates in the signature). Labels are also referred to as "features" or "attributes". If $|\text{Col}| = 1$, then we denote $\{E^c\}_{c\in\text{Col}}$ with E, as there is only one set of edges. We also use $V(G)$, $E(G)$ or $E^c(G)$, and λ_G to respectively denote the vertices, edges, and node labelling of a graph G.

We assume that there is a one-to-one correspondence between potential graph vertices and constants in the signature; we typically represent the vertex corresponding to constant a by v^a. When λ is clear from the context, we abbreviate $\lambda(v)$ as $\mathbf{v}$ for each $v \in V$. When needing to clarify which graph the node belongs to, we also abbreviate $\lambda(v)$ as G_v.

Graph G is undirected if E^c is symmetric for each $c \in$ Col and is Boolean if $\mathbf{v}[i] \in \{0, 1\}$ for each $v \in V$ and $i \in \{1, ..., \delta\}$. Note that since Col and δ are arbitrary (but fixed), all results using these definitions generalize to graphs with an arbitrary number of edge colours and vertex embedding dimensions.

2.2.3 Graph Isomorphism and Automorphism

Two graphs G, H are isomorphic (denoted $G \equiv H$) if there exists a bijection $\pi : V(G) \to V(H)$ that is edge-preserving and node label -preserving. Formally, we require that:

1. for all $u, v \in V(G)$, $(u, v) \in E(G)$ if and only if $(\pi(u), \pi(v)) \in E(H)$, and

2. for all $v \in V(G)$, $\lambda_G(v) = \lambda_H(\pi(v))$.

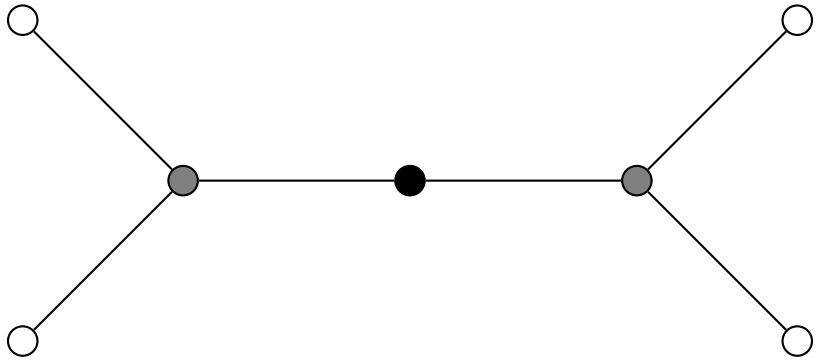


Figure 2.1: Graph with nodes coloured by which orbit they belong to, where the nodes have no labels.

A permutation on a set X is a bijection $\sigma : X \to X$. A permutation σ on a tuple $(x_1, ..., x_n)$ yields the tuple $(\sigma(x_1), ..., \sigma(x_n))$. A permutation (of nodes) σ on a graph $G = (V, E, \lambda)$ yields the graph $H := \sigma \cdot G$, where $V(H) = V(G)$, $E(H) = \{(\sigma(v),\ \sigma(w)) \mid (v,\ w) \in E(G)\}$, and $G_v = H_{\sigma(v)}$ for each $v \in V(G)$. An automorphism of G is a permutation σ on G satisfying $\sigma \cdot G = G$ (equivalently, it is an isomorphism from the graph to itself). Nodes $v, w \in V(G)$ are *similar* if there is an automorphism σ of G such that $\sigma(v) = w$. Node similarity induces an equivalence relation on $V(G)$, where each equivalence class is an *orbit*. The set $R(G)$ of orbits of G forms a partition of $V(G)$. Figure 2.1 depicts an example of graph orbits.

2.2.4 Graph Functions

Permutation equivariance is an important property for models realizing functions on graphs [21, 42]. Intuitively, if the function preserves the graph structure and assigns (potentially new) labels to each node, then permuting the input graph should be equivalent to applying the same permutation to the output graph (see Figure 2.2). A *node-labelling* function f maps each graph $G = (V, E, \lambda_G)$ in its domain $\mathcal{G}$ to a graph $H = (V, E, \lambda_H)$ with the same vertices and edges. Node-labelling function f on a domain $\mathcal{G}$ closed under permutation is *equivariant* if $f(\sigma \cdot G) = \sigma \cdot f(G)$ holds for all $G \in \mathcal{G}$ and permutations σ on G.

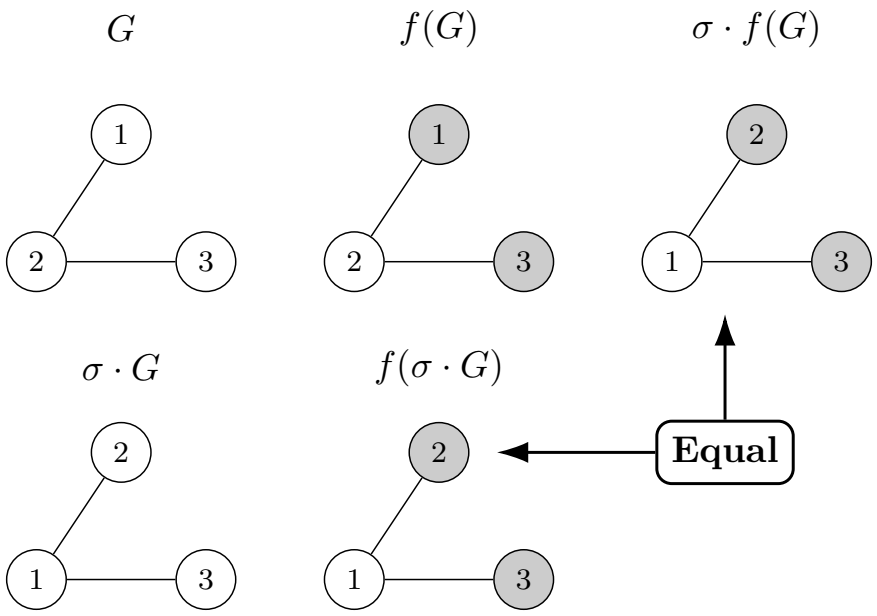


Figure 2.2: Example of an equivariant function f.

2.3 Graph Neural Networks

There are many definitions of Graph Neural Networks (GNNs). In this thesis, I use the term GNN to generally refer to neural models that use aggregation and combination operations to perform computation on graphs. The terms sum-GNN, max-GNN, and mean-GNN refer to such neural models that use sum, max, and mean aggregation to pool neighbourhood information respectively.

Graph neural networks are used for a variety of tasks. They can be broadly separated into node-level, edge-level, and graph-level tasks. KG completion can either be a node-level or an edge-level task. After a GNN has been applied to a graph, each node has a new (latent) feature, which can be used for these downstream tasks. This is shown visually in Figure 2.3.

An example of a node-level task is predicting the traffic flow through a junction, where the input graph consists of junctions and the roads connecting them. To solve node-level tasks, a final function is applied to the feature of the node being considered. An example of a graph-level task is predicting the acidity of a molecule, where the molecule is represented as a graph. To solve such a task, all node features are pooled and passed through a function to produce a final output for the graph. Finally, an

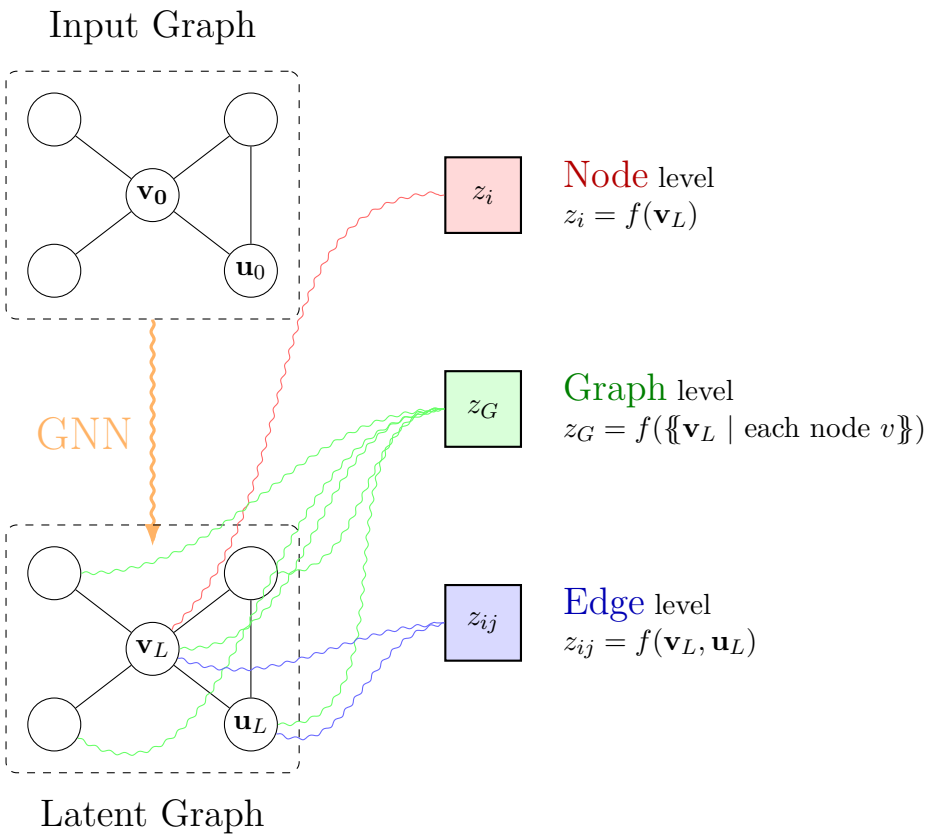


Figure 2.3: Primary tasks that a GNN is used for.

example of an edge-level task is predicting whether two people in a social network are friends. To achieve this, a function operating on two node features is used.

2.3.1 Message Passing Neural Networks

Commonly used GNNs such as GCNs [138, 234], GraphSAGE [112], and GATs [268] are defined within the Message Passing framework [99]: I refer to such models as Message Passing Neural Networks (MPNNs).

An MPNN consists of multiple local message-passing layers that can include global readouts: Gilmer et al. [99] originally defined them with only local message passing, but subsequent works [19] have extended them to include global readouts. I present the full definition here, but note that many works exclude the readout functions f_ℓ^{read} from the model. We define them for graphs with only a single edge colour, but note that they can be extended to incorporate graphs with multiple edge colours or graph labels [21]. We define an MPNN f_{MPNN} with

$L \geq 1$ layers to be a tuple

$$\langle\ \{f_\ell^{\text{update}}\}_{1\leq\ell\leq L},\ \{f_\ell^{\text{aggr}}\}_{1\leq\ell\leq L},\ \{f_\ell^{\text{read}}\}_{1\leq\ell\leq L}\ \rangle,$$

where each f_ℓ^{aggr} and f_ℓ^{read} are functions mapping multisets of real vectors to real vectors, and each f_ℓ^{update} is a function mapping a triple of real vectors to a real vector (also known as a *combination* function).

Applying f_{MPNN} to a graph G induces a sequence of labels $\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_L$ (sometimes denoted as $\mathbf{v}_{\lambda_0}, \mathbf{v}_{\lambda_1}, ..., \mathbf{v}_{\lambda_L}$) for each vertex v in the graph as follows. First, $\mathbf{v}_0$ is the initial labelling of node v in the input graph; then, for each $1 \leq \ell \leq L$, $\mathbf{v}_\ell$ is defined by the following expression:

$$\mathbf{v}_\ell = f_\ell^{\text{update}}\Big(\mathbf{v}_{\ell-1},\ f_\ell^{\text{aggr}}(\{\!\{\mathbf{u}_{\ell-1} \mid (v,u) \in E(G)\}\!\}),\ f_\ell^{\text{read}}(\{\!\{\mathbf{u}_{\ell-1} \mid u \in V(G)\}\!\})\Big).$$

The output of f_{MPNN} is a graph with the same vertices and edges as the input graph, but where each vertex is labelled by $\mathbf{v}_L$. Importantly, MPNNs are equivariant node-labelling functions for graphs labelled with real vectors.

2.3.2 Specific Graph Neural Networks

We consider concrete instantiations of the above framework that also allow for graph edges of varying colour. In particular, we define a graph convolutional network[2] (GCN) $\mathcal{M}$ with $L \geq 1$ layers to be a tuple

$$\langle\ \{\mathbf{A}_\ell\}_{1\leq\ell\leq L},\ \{\mathbf{B}_\ell^c\}_{c\in\text{Col},1\leq\ell\leq L},\ \{\mathbf{b}_\ell\}_{1\leq\ell\leq L},\ \{\sigma_\ell\}_{1\leq\ell\leq L},\ \{\text{agg}_\ell\}_{1\leq\ell\leq L},\ \texttt{cls}_t\ \rangle, \quad (2.2)$$

where matrices $\mathbf{A}_\ell$ and $\mathbf{B}_\ell^c$ are of dimension $\delta_\ell \times \delta_{\ell-1}$ with $\delta_0 = \delta = \delta_L$, $\mathbf{b}_\ell$ is a vector of dimension δ_ℓ, $\sigma_\ell : \mathbb{R} \to \mathbb{R}$ is an activation function, agg_ℓ is a function mapping real-valued multisets to real values, and $\texttt{cls}_t : \mathbb{R} \to \{0,1\}$ is a step classification function with threshold $t \in \mathbb{R}$ such that $\texttt{cls}_t(x) = 1$ if $x \geq t$ and $\texttt{cls}_t(x) = 0$ otherwise.

Applying $\mathcal{M}$ to a graph induces a sequence of labels $\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_L$ (sometimes denoted as $\mathbf{v}_{\lambda_0}, \mathbf{v}_{\lambda_1}, ..., \mathbf{v}_{\lambda_L}$) for each vertex v in the graph as follows. First, $\mathbf{v}_0$

[2]This is a generalization of the model R-GCN [234], allowing for biases and different aggregation functions.

is the initial labelling of node v in the input graph; then, for each $1 \leq \ell \leq L$, $\mathbf{v}_\ell$ is defined by the following expression:

$$\sigma_\ell(\mathbf{b}_\ell + \mathbf{A}_\ell \mathbf{v}_{\ell-1} + \sum_{c \in \mathrm{Col}} \mathbf{B}_\ell^c \, \mathrm{agg}_\ell(\{\!\{\mathbf{u}_{\ell-1} \mid (v,u) \in E^c\}\!\})) \tag{2.3}$$

The output of $\mathcal{M}$ is a graph with the same vertices and edges as the input graph, but where each vertex is labelled by $\mathtt{cls}_t(\mathbf{v}_L)$.

In Chapter 5, I use scoring functions with GCNs for link prediction. In this case, the classification function $\mathtt{cls}_t$ is not used, and the output of $\mathcal{M}$ is a graph with the same vertices and edges as the input graph, but where each vertex is labelled by $\mathbf{v}_L$.

We consider GCNs with different aggregation functions. This includes so-called "max-sum" aggregation functions, which generalize both max and sum aggregation. For $k \in \mathbb{N}_0 \cup \infty$, a finite real multiset S, and $j := \min(k, |S|)$, we define max-k-sum$(S) = \sum_{i=1}^{j} s_i$, where $s_1, ..., s_j$ are the j largest numbers of S, including repeated values. If $j = 0$, the sum is defined as 0. Note that max-1-sum is equivalent to max (assuming that the maximum of the empty multiset evaluates to zero) and max-∞-sum is equivalent to sum. If a GCN has $\mathrm{agg}_\ell =$ max-k_ℓ-sum for all $\ell \in \{1, ..., L\}$, where $k_\ell \in \mathbb{N}_0 \cup \infty$, we refer to it as a max-sum-GCN. Similarly, when each agg_ℓ is mean, we call it a mean-GCN (likewise for max and sum).

Finally, we say that a GCN is *monotonic* if for all $\ell \in \{1, ..., L\}$ and $c \in \mathrm{Col}$, all elements of $\mathbf{A}_\ell, \mathbf{B}_\ell^c$ are non-negative and σ_ℓ is monotonically increasing with non-negative range.

2.3.3 Scoring Functions

Scoring functions are often used in conjunction with GCNs to solve tasks related to edges in the graph (for example, for predicting whether two given nodes are adjacent). A *scoring function* f with dimension $d_f \in \mathbb{N}$ defines $f(R, \mathbf{h}, \mathbf{t}) \mapsto \mathbb{R}$ for all vectors $\mathbf{h}, \mathbf{t} \in \mathbb{R}^{d_f}$ and relations R in the fixed signature. It also defines a threshold $t_f \in \mathbb{R}$. For a set of constants C and function $e : C \to \mathbb{R}^{d_f}$ that assigns an embedding to each constant, we define

$$\mathcal{F}_f(C, e) = \{R(a, b) \mid a, b \in C, R \text{ is a relation}, f(R, e(a), e(b)) \geq t_f\},$$

i.e. the set of facts over constants of C scored as true by f. GCNs, as well as other methods, can be used to learn an embedding functions e.

We also consider *families* of scoring functions—sets of functions sharing a common form and parameterized by one or more values—where each family comprises all functions generated by varying the parameter values. For example, RESCAL is the family of scoring functions of the form $f(R, \mathbf{h}, \mathbf{t}) = \mathbf{h}^\top \mathbf{M}_R \mathbf{t}$, where $\mathbf{M}_R$ is a real matrix of dimension $d_f \times d_f$.

2.4 GCNs as Logical Functions

2.4.1 Correspondence Between Graphs and Datasets

Several schemes have been proposed in the literature for converting datasets to and from graphs. As the primary approach, we adopt the so-called *canonical scheme* [256], which is a straightforward way of converting datasets to coloured graphs. In particular, colours in graphs correspond to binary predicates in the signature and indices of feature vectors in the input and output layers of the sum-GCN to unary predicates. For each $p \in \{1, \ldots, \delta\}$, we denote the unary predicate corresponding to index p as U_p. More precisely, the canonical encoding $\texttt{enc}(D)$ of a dataset D is the Boolean (Col, δ)-graph with a vertex v_a for each constant a in D and a c-coloured edge (v_a, v_b) for each fact $R^c(a, b) \in D$. Furthermore, given a vertex v_a corresponding to constant a, vector component $\mathbf{v}_a[p]$ is set to 1 if and only if $U_p(a) \in D$, for $p \in \{1, \ldots, \delta\}$. The decoder $\texttt{dec}$ is the inverse of the encoder.

2.4.2 GCN Dataset Transformations and Sound Rules

A GCN $\mathcal{M}$ and the canonical dataset encoding/decoding scheme can be used to realize a transformation $T_\mathcal{M}$ from datasets to datasets over a given finite signature [256]. The canonical dataset transformation induced by a GCN $\mathcal{M}$ is defined as: $T_\mathcal{M}(D) = \texttt{dec}(\mathcal{M}(\texttt{enc}(D)))$. We abbreviate $\mathcal{M}(\texttt{enc}(D))$ by $\mathcal{M}(D)$.

A rule or program α is sound for a GCN $\mathcal{M}$ if $T_\alpha(D) \subseteq T_\mathcal{M}(D)$ for each dataset D. Conversely, α is complete for $\mathcal{M}$ if $T_\mathcal{M}(D) \subseteq T_\alpha(D)$ for each dataset D. We say that α is equivalent to $\mathcal{M}$ if it is both sound and complete for $\mathcal{M}$. Finally, we say that a rule or program α subsumes a rule r if for any dataset D, $T_r(D) \subseteq T_\alpha(D)$. Note that these definitions apply to Datalog, FOL, and description logic rules/programs.

Sound Rules as Explanations Some sound rules are more useful as explanations than others: for example, a rule with a body that is never satisfied (such as $\bot \rightarrow A(x)$) is trivially sound but predicts no facts and thus does not explain anything. Furthermore, consider an input dataset D, and a rule r which simply instantiates D as its body and a prediction of $\mathcal{M}$ as its head. Then r will explain that prediction and will be sound: however, it only explains one prediction, the body will not be satisfied in other datasets, and r will be too large to be understandable by a human.

Thus, what we require for explainability are sound rules with satisfiable bodies that are as small as possible. In my experiments in Chapters 5 to 7, I mostly check the soundness of rules with few body atoms, and also check specific rules which have been injected into synthetic datasets. This means that the sound rules obtained are useful as explanations.

Sound Rules Under Rule/Program Composition The following proposition establishes that the containment relation that defines soundness for a *monotonic* rule or program still holds when the operators are composed a finite number of times. We say that a FOL / $\mathcal{ALCQ}$ / Datalog rule (or program) r is monotonic (under dataset inclusion) if for all datasets D, D' such that $D \subseteq D'$, $T_r(D) \subseteq T_r(D')$. For example, any Datalog rule or program is monotonic. The property does not hold in general for FOL or description logic rules or programs, as their immediate consequence operators can be non-monotonic (see Section 2.4.4).

Proposition 1. *If α is a monotonic (under dataset inclusion) rule or program sound for GCN $\mathcal{M}$, then for any dataset D and $k \in \mathbb{N}$, the dataset containment holds when T_α and $T_\mathcal{M}$ are composed k times: $T^k_\alpha(D) \subseteq T^k_\mathcal{M}(D)$.*

Proof. Let D be an arbitrary dataset. We prove the claim by induction on k. The base case of $k = 1$ follows trivially, since $T_\alpha(D) \subseteq T_\mathcal{M}(D)$ holds from the soundness of α.

For the inductive step, let $k \geq 2$. Let $D_\alpha = T_\alpha^{k-1}(D)$ and $D_\mathcal{M} = T_\mathcal{M}^{k-1}(D)$. Then from the soundness of α, $T_\alpha(D_\mathcal{M}) \subseteq T_\mathcal{M}(D_\mathcal{M})$, and from the induction hypothesis, $D_\alpha \subseteq D_\mathcal{M}$.

But then $T_\alpha(D_\alpha) \subseteq T_\alpha(D_\mathcal{M})$, since operator T_α is monotonic. So we have $T_\alpha(D_\alpha) \subseteq T_\alpha(D_\mathcal{M}) \subseteq T_\mathcal{M}(D_\mathcal{M})$, and thus $T_\alpha^k(D) \subseteq T_\mathcal{M}^k(D)$. □

2.4.3 Link Prediction

The link prediction task assumes a given incomplete dataset D and an (unknown) completion D^* of D containing all missing *binary* facts that are considered true over the predicates and constants of D. Thus, given a fact $P(a, b)$ involving a binary predicate P and constants a, b from D, the task is to predict whether $P(a, b) \in D^*$.

Dataset Encoding Trick To perform link prediction with a standard node-labelling GCN $\mathcal{M}$, we use the (non-canonical) encoding and decoding from Tena Cucala et al. [256], which differs from the canonical encoding in that vertices can now also encode pairs of constants, so that both unary and binary facts can be represented in the vertex features of the input and output layers. This allows us to derive new binary facts (which the canonical transformation cannot do). As shown by Tena Cucala et al. [256, Section 3.2], this non-canonical encoding can be expressed as the composition of a Datalog program $\mathcal{P}_{\texttt{enc}}$ and the canonical encoding; similarly, the decoding can be seen as the composition of the canonical decoding followed by the application of a Datalog program $\mathcal{P}_{\texttt{dec}}$. Hence, we can use these programs to lift any rule extraction results obtained for the canonical transformation $T_\mathcal{M}$ to the end-to-end transformation $\mathcal{P}_{\texttt{dec}}(\texttt{dec}(T_\mathcal{M}(\texttt{enc}(\mathcal{P}_{\texttt{enc}}(D)))))$.

Scoring Functions A more common method for performing link prediction is to use a scoring function. A GCN $\mathcal{M}$ with L layers and a scoring function f with dimension $d_f = \delta_L$ can be used to realize a transformation $T_{\mathcal{M},f}$ from datasets to datasets containing only binary facts — going forward, when we mention a GCN and scoring function together, we assume $d_f = \delta_L$.

Given a scoring function f, we define the application of the decoder $\mathtt{dec}_f$ to a labelled graph G as producing the dataset containing $R^c(a, b)$ for each relation R^c and pair of constants a, b corresponding to vertices in G such that $f(R^c, \mathbf{v}^a, \mathbf{v}^b) \geq t_f$. Thus, we have $\mathtt{dec}_f(\mathcal{M}(\mathtt{enc}(D)))$

$$= \{R^c(a, b) \mid a, b \in \mathtt{con}(D), c \in \mathrm{Col}, f(R^c, \mathbf{v}^{\mathbf{a}}_{\mathbf{L}}, \mathbf{v}^{\mathbf{b}}_{\mathbf{L}}) \geq t_f\}.$$

The *canonical* dataset transformation induced by a GCN $\mathcal{M}$ and a scoring function f is then defined as: $T_{\mathcal{M},f}(D) = \mathtt{dec}_f(\mathcal{M}(\mathtt{enc}(D)))$. We often abbreviate $\mathcal{M}(\mathtt{enc}(D))$ by $\mathcal{M}(D)$. The GCN and scoring function combination can be trained end-to-end.

Soundness, completeness, and equivalence are defined in the same way as for standard GCNs: for example, a program or rule α is sound for a combination $(\mathcal{M}, f)$ of a GCN $\mathcal{M}$ and a scoring function f if $T_\alpha(D) \subseteq T_{\mathcal{M},f}(D)$ for each dataset D.

2.4.4 Monotonicity

Many of the rule extraction techniques used in this thesis and in prior work [254, 256] depend on monotonicity properties of certain logical fragments and GCN-based dataset transformations. Intuitively, a rule or program is *monotonic* if, for all datasets, the immediate consequences it yields on the dataset are also yielded in any extension of the dataset.

Formally, a homomorphism from a dataset D to a dataset D' is a mapping h of constants to constants that is defined on at least $\mathtt{con}(D)$, and that satisfies $h(D) \subseteq D'$, where $h(D)$ is the dataset obtained by replacing each constant a in D with $h(a)$ (and removing all duplicate facts). Homomorphism h is injective

if $h(a) \neq h(b)$ for all distinct pairs a, b of constants in the domain of h. An operator T from datasets to datasets over the same constants is then *monotonic under homomorphisms* if, for all datasets D and D to which T is applicable, each homomorphism from D to D' is also a homomorphism from $T(D)$ to $T(D')$; moreover, T is *monotonic under injective homomorphisms* if the property holds only for injective homomorphisms.

Then, for example, for any inequality-free Datalog program $\mathcal{P}$, the operator $T_\mathcal{P}$ is monotonic under homomorphisms [254]. For any Datalog program $\mathcal{P}$ (possibly containing inequalities), the operator $T_\mathcal{P}$ is instead monotonic under injective homomorphisms. Tena Cucala et al. [254] show that the dataset transformations resulting from monotonic max-GCNs are monotonic under homomorphisms, and Tena Cucala et al. [256] prove that monotonic max-sum-GCNs are monotonic under injective homomorphisms. Such GCNs satisfy the properties due to (1) the fact that node feature values in the GCN output do not decrease as facts are added to the input dataset and (2) GCNs / the canonical encoding being agnostic to the particular constants used in the input dataset (i.e. GCNs do not depend on the nodes themselves, merely their features and the graph structure).

In this thesis, I also use a simpler definition of monotonicity based on dataset inclusion instead of homomorphisms:

Definition 2. *A FOL / $\mathcal{ALCQ}$ rule (or program) r is* monotonic *(under dataset inclusion) if for all datasets D, D' such that $D \subseteq D'$, $T_r(D) \subseteq T_r(D')$.*

If the rule or program is also agnostic to the particular constants in the dataset, then the resulting operator is monotonic under injective homomorphisms.

Notice that due to the canonical correspondence between graphs and datasets, we can equivalently say that a FOL formula φ with a free variable is monotonic (under "graph extension") if for all Boolean graphs G, G' and vertices $v \in V(G)$ such that

1. $V(G) \subseteq V(G')$,
2. $E(G) \subseteq E(G')$,

3. any node label entries that are 1 in λ_G are also 1 in $\lambda_{G'}$, and

4. $(G, v) \models \varphi$,

we have that $(G', v) \models \varphi$.

2.4.5 Relationship to Barcelo-style Expressivity

Most of the definitions used in this thesis for sound rules and equivalent programs are based on the work of Tena Cucala et al. [254, 256], which I term "Cucala-style expressivity". These definitions are based on logical rules. However, there are wider-used definitions of expressivity going back to the seminal work of Barceló et al. [19], which I term "Barcelo-style expressivity". In contrast to Cucala-style expressivity, these definitions are based on capturing logical classifiers. However, the notions of equivalent rules/programs and capturing a logical classifier are essentially the same, and results from either setting can be lifted to the other. For completeness, I will illustrate the equivalence.

Barceló et al. [19] define a logical classifier to be a FOL formula with a single free variable. They define the "capturing" of logical classifiers in terms of graphs; however, given the correspondence between graphs and datasets, I provide an equivalent definition in terms of datasets:

Definition. *A GCN $\mathcal{M}$ that outputs a value in $\{0,1\}$[3] for each node* captures *a logical classifier φ if for every dataset D and constant $a \in \texttt{con}(D)$, it holds that $\mathcal{M}(D)_{v^a} = 1$ if and only if $(D, a) \models \varphi$.*

Consider now the Cucala-style setting, but where the signature contains only one unary predicate (i.e. $\delta = 1$). Notice that a FOL logic rule $r : B \to A(x)$ (where B has free variable x and A is the single unary predicate) is equivalent to a GCN $\mathcal{M}$ if for every dataset D, $T_{\mathcal{M}}(D) = T_r(D)$. I.e. for every dataset D, $\{A(a) \mid a \in \texttt{con}(D),\ \mathcal{M}(D)_{v^a} = 1\} = \{A(a) \mid a \in \texttt{con}(D),\ (D, a) \models B\}$. This is

[3]Technically, such a GCN would output a Boolean vector of size 1 for each node, but I present them as single values for readability.

equivalent to saying that for every dataset D and constant $a \in \texttt{con}(D)$, it holds that $\mathcal{M}(D)_{v^a} = 1$ if and only if $(D, a) \models B$.

Notice that this is the same as the Barcelo-style definition of capturing a classifier. Thus, the Cucala-style setting is identical to Barcelo-style if the signature contains only one unary predicate. Cucala-style expressivity allows one to directly characterize GCNs that output more than just $\{0, 1\}$ for each node. In other words, Barcelo-style deals specifically with binary classification, where Cucala-style describes the more general problem of node classification with an arbitrary number of classes (the number δ of unary predicates). However, Barcelo-style can also characterize node classification GCNs by using a different binary classifier for each class. Equivalent notions of Cucala-style soundness and completeness can also be defined for the Barcelo-style approach. Thus, these approaches to expressivity are essentially equivalent.

Finally, I note that both Cucala-style and Barcelo-style expressivity are concerned with *uniform* expressivity: i.e. the equivalence between the logical program / classifier and the GCN must hold for *every* dataset. In Chapter 4, I also consider uniform expressivity using the 1-WL graph colouring algorithm [189, 287]. An alternative form of expressivity that is sometimes considered, such as by Schönherr and Lutz [235], is *non-uniform* expressivity. Non-uniform expressivity only requires equivalence on datasets with a bounded number of constants. In the non-uniform case, a model architecture $\mathcal{M}$ is said to be equivalent to a logical language $\mathcal{L}$ if for every $n \in \mathbb{N}$, $\mathcal{M}$ and $\mathcal{L}$ can express exactly the same logical classifiers across all graphs with n vertices.

2.5 Experiments

Across Chapters 4 to 7, I run experiments on GCNs to test my theoretical findings in practice. Whilst the models, datasets, training techniques, and experiment goals differ across each chapter, there are some common features which I will outline in this section.

2.5.1 Experimental Setup

Experiments are run using PyTorch Geometric, with 2 CPUs and 16GB of memory on a Linux server. I run each experiment across different random seeds (5-10, depending on the chapter) and present the aggregated metrics: an experiment consists of a specific set of hyperparameters and a single trained GCN model (sometimes with a scoring function). I train each model up to a specified number of epochs, sometimes employing early stopping if the training loss deteriorates for 50 epochs. Then, I evaluate the model on the validation dataset, maximizing for the accuracy metric (see Section 2.3.2) to choose a classification threshold (if the model has one) and to optimize other hyperparameters. Finally, with a fixed classification threshold, I evaluate the trained model on the test dataset and report the metrics described in Section 2.5.2.

The GCN definition given in Section 2.3.2, chosen to correspond to that of Tena Cucala et al. [256] and for ease of presentation, describes aggregation in the reverse direction of the edges. In my experiments, I use the standard approach and aggregate in the direction of the edges. I train monotonic models (i.e. those with only non-negative matrix weights) by clamping negative matrix weights to 0 after each optimizer step, as in the approach of Tena Cucala et al. [254]. In Chapters 5 to 7, I provide theoretical results which allow for the checking of sound rules for the GCNs covered in each chapter: in my experiments in these chapters, I extract sound rules from the trained models in practice.

2.5.2 Metrics

There are a variety of metrics used for evaluating the performance of KG completion models [162]. To define the metrics, let P_{test} and N_{test} be the sets of positive and negative facts used for testing, respectively.

Standard classification metrics can only be calculated when the model makes predictions of P_{predict} and N_{predict} of true and false facts, respectively. If a KG completion model produces a confidence score for candidate facts, then a confidence score threshold is typically needed if one wants to compute classification metrics.

Let tp, tn, fp, and fn respectively be the counts of the number of true positives (facts predicted to be true by the model that are actually true), true negatives, false positives, and false negatives. These counts can be computed as follows:

$$tp = |P_{\text{predict}} \cap P_{\text{test}}| \qquad tn = |N_{\text{predict}} \cap N_{\text{test}}|$$
$$fp = |P_{\text{predict}}| - tp \qquad fn = |N_{\text{predict}}| - tn$$

From these values, standard classification metrics can be computed as follows:

$$\text{precision} = \frac{tp}{tp + fp} \qquad \text{recall} = \frac{tp}{tp + fn}$$
$$\text{accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \qquad \text{F1} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

When a model uses a threshold value to make its predictions (e.g. by assigning all candidate facts with a confidence score over a given threshold to be true), then precision and recall can be plotted on a graph for different threshold values, as lower threshold values will typically lead to higher recall and lower precision, and vice-versa for higher threshold values. This yields a precision-recall curve, where precision is given on the y-axis and recall on the-x axis. The area under this curve yields the metric $AUPRC$ [38].

When presenting an aggregated metric, I always present the mean. In Chapter 4, I also present the standard deviation. In Chapter 7, to compute 95% confidence intervals, I assume a normal distribution and compute the interval as 1.97× SEM (standard error of the mean).

When evaluating link prediction models, other "ranking-based" metrics are often reported, including mean rank, mean reciprocal rank, and Hits@k [162]. However, given that prior works in this area exclusively use classification metrics [254], and that the non-canonical dataset transformation described in Section 2.4.3

	Pattern
Hierarchy (hier)	$R(x,y) \rightarrow S(x,y)$
Symmetry (sym)	$R(x,y) \rightarrow R(y,x)$
Cup	$R(x,y) \wedge S(y,z) \wedge T(w,x) \rightarrow P(x,y)$
NM-Hierarchy (nmhier)	$R(x,y) \wedge \neg S(y,z) \rightarrow T(x,y)$

Table 2.1: LogInfer inference patterns used in my thesis.

converts link prediction problems into node classification ones, I report only the above classification metrics.

2.5.3 Datasets

For the experiments of Chapters 5 to 7, I use three standard link prediction benchmarks: WN18RRv1, FB237v1, and NELLv1 [258], each of which provides datasets for training, validation, and testing. I use the negative examples (for validation and testing) and positive targets sampled by Tena Cucala et al. [254], so that the results can be compared with directly. In Chapter 7, I also use the LUBM dataset [108, LUBM(1,0)], with the train/test split from Liu et al. [162]; this is a dataset where the task is node classification.

In Chapters 5 to 7, I also use LogInfer [162], a framework which augments a dataset by applying Datalog rules of a certain shape—called a "pattern"—and adding the consequences of the rules back to the dataset. Such a rule is referred to as an *injected* rule. I apply the LogInfer framework to the datasets FB15K-237 [260] and WN18RR [76] (the datasets from which FB237v1 and WN18RRv1 were obtained). I consider the rule patterns *hierarchy* and *symmetry* defined by Liu et al. [162]; I also use a new monotonic pattern *cup*, which has a tree-like structure, and a non-monotonic rule pattern *non-monotonic hierarchy* (nmhier). All patterns are shown in Table 2.1.

For each pattern P, I refer to the datasets obtained by enriching FB15K-237 and WN18RR by FB-P and WN-P, respectively. For each created dataset, in each training epoch of a model, I randomly select 10% of the enriched training dataset to be used as targets and the rest as inputs to the model. Furthermore, I consider an

additional instance of FB15K-237 and WN18RR where I again apply the hierarchy pattern, but I include a much larger number of consequences in the enriched training dataset; I refer to these datasets as FB-superhier and WN-superhier, as a shorthand for *super-hierarchy*. This yields a setting where it should be simple for a model to learn the rules applied in the dataset's creation.

Finally, I use LogInfer to augment FB15K-237 and WN18RR using multiple rule patterns at the same time; in particular, I combine monotonic with non-monotonic rule patterns. This allows me to test the ability of my rule extraction methods to recover the sound monotonic rules that were used to generate the enriched dataset, in the presence of facts derived by non-monotonic rules. For example, WN-hier_nmhier refers to a LogInfer dataset where WN18RR is extended using the hierarchy and non-monotonic hierarchy rule patterns. When creating a mixed dataset, I reserve a number of predicates to use in the heads of the one pattern, and the rest of the predicates to use in the heads of the other.

Negative examples are generated for LogInfer using predicate corruption: i.e. for each positive example $P(a, b) \in D$, I sample a predicate Q at random to obtain a negative example $Q(a, b)$ such that it does not appear in the training, validation, or test set. I avoid using constant corruption to produce negative examples because this would inflate the performance of my models. The reason for this is that the link prediction encoding by Tena Cucala et al. [254] can only predict binary facts for constant pairs occurring together in input facts, and hence would trivially (and correctly) classify almost all negative examples. When generating negative examples for a hier_nmhier dataset, I sample $T(a, b)$ such that $\{R(a, b), S(b, c)\} \subseteq D$ (see NM-Hierarchy in Table 2.1), to penalize the model for learning the rule pattern $R(x, y) \rightarrow T(x, y)$ instead of the non-monotonic one.

Importantly, all of these datasets are *inductive*, meaning that the validation and testing sets contain constants not seen during training. Thus, machine learning methods that learn embeddings for constants cannot be used with them. The alternative to being inductive is being *transductive*, where all relations and constants used for testing are seen during training.

3
Literature Review

Contents

3.1 Non-GNN Approaches to Learning on KGs 32
3.2 Graph Neural Networks 35
3.2.1 Important Graph Neural Networks 35
3.2.2 Knowledge Graph Encodings 38
3.3 Expressivity of GNNs 40
3.3.1 Graph Separation Power 40
3.3.2 Approximation Ability 42
3.3.3 Substructure Counting 43
3.3.4 Other Non-Logic-Based Methods 44
3.3.5 Formal Logic 45
3.4 Explainability of GNNs 49
3.4.1 Post-Hoc GNN Explainability 50
3.4.2 Self-Explainable GNNs 53
3.4.3 Logic-Based Self-Explainable GNNs 55

In this chapter, I provide a summary of the literature relevant to the work of my thesis. Papers that are very closely related to a later chapter will be further discussed within the chapter itself. I first summarize various non-GNN approaches to learning on KGs in Section 3.1, some of which I will make use of in Chapter 5 in conjunction with GNNs. Then in Section 3.2, I discuss a number of different important GNN models. Finally, in Section 3.3, I summarize important works that analyse the expressivity of GNNs within different paradigms, and in Section 3.4 other works

which provide explainability for GNNs. Sections 3.3 and 3.4 are particularly relevant for giving context to the work in my thesis, since I deal with GNN expressivity in Chapters 4, 5 and 7 and GNN explainability in Chapters 5 to 7.

3.1 Non-GNN Approaches to Learning on KGs

In this section, I provide an overview of various methods for learning on KGs, not including graph neural networks, which will be covered in Section 3.2. The majority of modern methods for performing knowledge graph completion rely on neural networks, or learning vector embeddings using gradient descent. Collectively, these approaches are often referred to as *subsymbolic* methods [125]. However, other techniques rely on logical systems, rule-based inference, and statistical patterns to infer new facts in KGs. Approaches based on formal logic and rules are collectively referred to as *symbolic* methods.

Symbolic Methods A prominent subset of symbolic methods for learning on KGs is Inductive Logic Programming (ILP) [62], which encompasses a variety of methods that learn logic programs to explain the training data given to the model [4, 61, 126, 195, 221, 245, 320]. Others symbolic approaches include rule-mining methods such as AMIE [91], AMIE+ [90], AMIE 3 [142], and AnyBURL [179]. Finally, approaches based on Statistical Relational Learning [98] use FOL rules with uncertainty, which include MLNs [165, 200, 222] and PSL [16, 41, 213].

Scoring Functions A common approach to subsymbolic learning on KGs is to score binary facts in isolation using learned embeddings. I refer to this subfield of methods as "scoring functions", although they are sometimes called "knowledge graph embedding" [281]. Different KG scoring function models are broadly distinguished from one another by their choices of constant embeddings, relation embeddings, scoring functions, and regularization / embedding constraints. Scoring function models typically only operate in transductive settings, since they require learned embeddings for constants. On the other hand, many GNN models do not

learn embeddings for the constants, and thus can be used in inductive settings. In this chapter, I use the term "scoring functions" to refer to this more general class of models, rather than the specific formalization that I use in Chapters 2 and 5 with GNNs.

"Distance-based" models use scoring functions that compute distances between constant embeddings, typically after a translation has been performed, conditional upon the relation embedding. Such models include the pioneering TransE [32], along with its precursors, unstructured model (UM) [31] and structured embedding (SE) [30], and its successors, which are proposed to address various expressivity issues: TransH [285], TransR [157], TransD [127], TransSparse [128], TransM [84], ManifoldE [296], TransF [86], and TransA [295]. Other distance-based models include RotatE [249], HAKE [333], and TransW [169]. KG2E [114] and TransG [297] represent constants and relations as random vectors sampled from multivariate Gaussian distributions. BoxE [1] encodes relations as axis-aligned hyper-rectangles and constants as points. Abboud et al. [1] provide an analysis of the types of relations and inference patterns that several scoring function models are able to represent: BoxE emerges as one of the most expressive. In Chapter 5, I show how these results can be leveraged when using scoring functions with a GNN for link prediction.

"Semantic matching" models use scoring functions that rely on computing similarity between representations. These include RESCAL [197] and its extensions: TATEC [94], DistMult [303], HolE [199], ANALOGY [158], and TuckER [17]. Other semantic matching models use non-real vector spaces for embeddings, such as ComplEx [261], QuatE (also known as Quaternion) [330], and DualE [47]. Some models use neural networks to compare embeddings, such as SME [34], NAM [159], NTN [240], MLP [77]. In Chapter 5, I use some semantic matching models with GNNs for link prediction.

Random Walks and Deep Learning Several models use random walks over the graph to learn embeddings, including PRA [144], CPRA [280], DeepWalk [208], Node2vec [105], and LINE [250]. The operation of these models can be explained

by implicit factorization of the graph Laplacian matrix [54]. Thus, linked to these random walk methods, several approaches explicitly use matrix decomposition to learn embeddings, such as SDNE [275], NetMF [214] (which unifies DeepWalk, LINE, and Node2vec), NetSMF [215], ProNE [324], and HOPE [56].

A Convolutional Neural Network (CNN) [106, 146] is a neural architecture that is typically used in applications pertaining to image processing. However, it has also found use for KG completion, in models like ConvE [76], ConvKB [69], HypER [18], InteractE [265], and ConEx [75].

Since KGs can be represented as sequences of facts, neural networks that can handle sequence data, such as RNNs and LSTMs [97, 116], have been used in approaches to KG completion. These methods include DSKG [107] and PRCTA [147], which uses a mixture of path-based reasoning and sequence processing to perform KG completion. DRUM [227] uses bidirectional RNNs for learning logical rules from KGs.

Other approaches to KG learning define differentiable versions of reasoning algorithms and use them to train embeddings. Prominent examples of this include NTPs [224], CTPs [184], δILP [83], and Neural Logic Programming [60, 304].

Large Language Models Large Language Models (LLMs) are at the forefront of contemporary AI breakthroughs, with impressive results being obtained across many natural language processing tasks. There is a plethora of work combining LLMs and KGs, which Pan et al. [204] provide a thorough overview of.

One of the ways in which LLMs have been used to augment KGs is to perform KG completion. These methods fall into two broad categories: LLMs as encoders and LLMs as generators. The first approach uses LLMs to generate embeddings for facts and constants / relations, which are then used downstream by scoring function KG completion methods, such as TransE [32]. These methods include KG-Bert [307], MTL-KGC [136], PKGC [168], LASS [239], MEM-KGC [59], OpenWorld KGC [58], StAR [274], SimKGC [279], CSPromp-KG [51], and LP-BERT [149].

The second approach is to use LLMs as sequence-to-sequence generators, where they receive a query fact $R(h, ?)$ and then output the text representing the missing constant. Among these methods are GenKGC [299], KGT5 [233], KG-S2S [50], and AutoKG [336]. Overall, LLMs as encoders have proven more successful for KG completion than LLMs as generators, since the scoring function can be fine-tuned in isolation, while the parameters of the LLM are frozen.

3.2 Graph Neural Networks

In this section, I cover some GNNs that are foundational to the field, and some other important GNNs from contemporary work. I also note that GNNs can be applied to different encodings of KGs, and cover some non-canonical encodings that have found use in the literature.

3.2.1 Important Graph Neural Networks

There are far too many GNN architectures to fully describe, so I instead cover some GNNs that are foundational to the field, along with some other important modern GNNs. With respect to a unified view of GNNs, Battaglia et al. [21] define a general framework referred to as "graph networks", which subsumes most GNN architectures, and some non-GNN architectures as well. The framework is defined for graphs that contain node, edge, and global features, but parts of it can be removed to yield simpler architectures that fall within the specification.

Graph Convolutional Networks GCNs [80, 138] are one of the earliest and simplest GNN architectures. They use mean aggregation for f^{aggr} and a linear transformation for f^{update}. Kipf and Welling [138] use symmetric normalization (symmetric dividing by the degree of the nodes) when defining their final single GCN layer. Note that the definition of the term "GCN" used in this thesis (see Section 2.3.2) is slightly different, as it also accounts for edges of varying colour and different aggregation functions besides mean.

R-GCN [234] is one of the first attempts to use GNNs for KG completion. The model applies an adapted GCN to the canonical encoding of the KG. Since GCN does not take edge attributes into account, R-GCN aggregates the edges of each different colour separately.

To perform link prediction, the model uses R-GCN as an encoder to generate embeddings for each node. These embeddings are then scored using DistMult [303]. The scores are used to evaluate the truth of binary facts: either by applying a threshold and classifying any fact with a score above the threshold as true, or by ranking candidate completions of binary facts by how high the score of the fact is.

Other GCN-based Networks RA-GCN [259] uses the same graph representation as R-GCN, but proposes a small modification to the GCN layer used, experimentally showing minor improvements. Their approaches to constant classification and link prediction are the same as for R-GCN. The intuition behind the method is that nodes with almost identical features are likely to represent classes of similar constants: during the update step of a model layer, each group of such nodes connected to the node being updated is collected into a single virtual constant, which is used for the feature update.

Cai et al. [45] note that R-GCN does not learn embeddings for relations and thus cannot be used for link prediction without a scoring function decoder, as well as pointing out the large number of parameters needed in the model, since every relation is represented by a different parameter matrix. To solve these issues, they propose TransGCN, which extends R-GCN by using distance-based scoring functions in their GCN layers.

KE-GCN [313] extend the concepts behind TransGCN to work with any scoring function. However, they also reintroduce a parameter matrix for each relation. KE-GCN subsumes CompGCN [264]. KE-GCN also trivially subsumes R-GCN [234]. It subsumes W-GCN [238] under similar conditions. W-GCN is part of a larger model, SACN [238], which has an additional step of using a convolutional network as a decoder, after W-GCN has been applied. Other examples of GCN-based GNNs

include CayleyNet [148], DualGCN [338], DCNN [11], GraphSage [112], LGCN [92], FastGCN [53], GIN [300], and ClusterGCN [57].

Instead of setting initial node features based on facts in the KG, some GNNs learn initial embeddings $\mathbf{v}_0, \mathbf{u}_0, ...$ for the nodes in the graph – however, this only works in the transductive setting, as nodes not seen during training will have no learned embedding. As an alternative approach, Hamaguchi et al. [111] use a GCN to compute embeddings for nodes that were not seen during training, using existing embeddings for neighbouring nodes that appeared in training.

Graph Attention Networks As discussed in Section 2.3.2, in this thesis I consider GCNs with different aggregation functions, specifically sum, max, mean, and max-sum aggregation. However, I do not consider *attention*, another prominent method for aggregation. Graph Attention Networks (GATs) [268] extends GCNs by using *attention coefficients* to control how much of each message passed from the neighbouring nodes contributes to the sum. KB-GAT [196], RAGAT [163], MRGAT [67], and DisenKGAT [291] are all examples of models that perform KG completion by applying GATs to the canonical graph encoding.

Besides attention, other less common methods for performing aggregation include using LSTMs [262] and other learnable aggregators [171, 203].

Transformer-Based GNNs GATs perform attention aggregation within the neighbourhood of each node. In comparison, transformers [267], primarily used for natural language processing, use attention over entire sequences. Graphormer [309] extends transformers to work on graphs, meaning that each node receives information from every other node in the graph via attention aggregation, instead of just from its own neighbourhood. The graph structure is still preserved by adding node degree information to node features, using the shortest path distance between nodes when computing the attention coefficient, and incorporating edge features / colours into attention coefficients. Graphormer proved effective for molecular graphs, but struggled to scale on larger graphs due to expensive computations.

Nodeformer [292] aims to solve the scalability problems with Graphormer and allow the method to work with large graphs. This is done by using a kernelized attention mechanism, which is a technique from transformers for processing long sequences efficiently, by approximating attention coefficients through dot products instead of matrix multiplications. M-Graphormer [49] extends Graphormer by having multiple attention channels, which improves model performance on graphs with more complex relational structures.

Query-Dependent GNNs There are a number of GNN models whose operations depend on a query: i.e. the models are used for link prediction, and their operation is conditional upon the particular pair of nodes being considered. GraIL [258] extracts sub-graphs by considering all the paths between the query nodes, then applies GNNs to these sub-graphs. The underlying GNN is similar to R-GCN, but augmented with attention coefficients. Teru et al. [258] also prove that the operation of a GraIL model can represent a subset of first-order logic.

Similar to GraIL, DPMPN [301] constructs query-dependent subgraphs: two GNNs are applied, one to the whole graph and one to the subgraph. In the SEAL model [327], a subgraph is obtained from a query pair of nodes by extracting the subgraph within a certain number of hops from the pair. Nodes of the subgraph are then given positional labels based on their distances to the query nodes. Finally, a GNN that produces an output for the whole graph (i.e. a graph-level task) predicts a binary label: whether the query nodes are linked or not.

3.2.2 Knowledge Graph Encodings

Whilst many GNNs operate on the canonical graph encoding of KGs, others use non-standard encodings. A prominent example of this is INDIGO [161], which provides a new way to encode a KG into a graph that can be operated on by a GNN, conditional upon the candidate binary facts being evaluated for truth. As a result of this encoding, INDIGO can function in inductive settings. It also allows the model predictions to be read out straight from the GNN outputs, instead of

needing a learnable decoder or scoring function. MGNNs [254] use a similar graph encoding (as described in Section 2.4.3), with two primary differences. First, edges also receive one of four different colours depending on the nodes being connected. Second, the encoding does not consider candidate facts.

The k-GNNs proposed by Morris et al. [186] also operate on a graph encoding, which consists of k-tuples of nodes. Cai et al. [44] define Line GNNs, where the KG is encoded as a line graph: each node in the line graph corresponds to an edge in the original graph, and two nodes in the encoding are connected if their corresponding edges in the input graph shared a common endpoint.

Wang and Chen [276] propose a graph encoding where extra nodes are appended between adjacent constants, using auxiliary binary predicates to connect them. They prove that when the graph encoding is used alongside MPNNs with global readout, the model can capture any logical classifier in the logic $\mathcal{FOC}_2$, which is not the case for MPNNs with the canonical encoding.

Gao and Ribeiro [93] consider temporal graphs, encoding each temporal graph as a single static (standard) graph. They achieve this by using recurrent neural networks to combine node and edge features from the temporal graph, which is represented as a sequence of static graphs. They then use a standard MPNN to process the resulting static graph. They prove that this system is more expressive that another commonly used strategy for representing temporal graphs.

The operation of Graphormer [309] can be seen as applying a model to a graph encoding, which is a fully connected version of the graph where node features contain encodings of structural information. Grötschla et al. [104] benchmark different positional encodings that can be added to the node features and then used with GNNs, including Laplacian positional encodings [82], SignNet [155], ESLapPE [277], GCKN [182], RRWP [170], RWSE [82], and Weisfeiler Lehman -based positional encodings [81].

3.3 Expressivity of GNNs

Whilst graph neural networks are a widely-used and effective machine learning architecture for graphs, there are many open questions regarding their expressivity. In this section, I first provide an overview of prominent methods for examining the expressivity of GNNs, particularly in their abilities to distinguish between different non-isomorphic graphs, approximate different functions, count substructures, and other methods. I also cover a method of describing expressivity that is particularly relevant to the work of my thesis: using formal logic. In Chapter 4, I consider the expressivity of various extensions of MPNNs in terms of their ability to compute node-labelling functions that go beyond equivariance, and in Chapters 5 and 7, I describe the expressivity of different GCNs using formal logic.

In this section, I use the term MPNN to refer to MPNNs specifically without global readout f^{read}, unless stated otherwise.

3.3.1 Graph Separation Power

One of the earliest methods for identifying the expressive power of GNNs is to describe which non-isomorphic graphs GNNs are able to distinguish. In this line of work, GNNs that produce an output for the whole graph are used, and a GNN is said to be able to distinguish two graphs if it can produce different outputs for the graphs. The Weisfeiler Leman graph isomorphism test [287] has proven especially useful for describing which graphs GNNs are able to distinguish: it has come to be known as the "1-WL algorithm", and operates by iteratively colouring the graph. In Chapter 4, I define and use a simple extension of the 1-WL algorithm to describe the expressivity of my proposed models. Note that due to the correspondence between the Weisfeiler Leman test and fragments of counting logic [43], these approaches to expressivity are closely related to the methods based on logical expressivity I present in Section 3.3.5.

Morris et al. [186], Xu et al. [300] proved that MPNNs are upper-bounded by 1-WL when it comes to distinguishing graphs. Formally, they prove that if 1-WL yields the same output for two graphs, then any MPNN will also yield the same

output for the graphs. Xu et al. [300] define the Graph Isomorphism Network layer (GIN) and prove that it has the same expressive power as 1-WL: note that other MPNNs are upper-bounded by 1-WL, but may not be as expressive as it.

Going Beyond 1-WL To go beyond the limits of 1-WL, Morris et al. [186] defined k-GNNs, which learn features over tuples of k nodes. These models are bounded in expressivity by the k-WL test, which is a generalization of the 1-WL test that operates on k-tuples of nodes. However, there exist k-GNNs that can distinguish graphs that the $(k-1)$-WL test cannot distinguish. k-GNNs allow one to break the expressivity barrier of 1-WL, but they have serious computational costs that make them difficult to use in practice. To address this, Morris et al. [188] propose a more efficient version of k-GNNs that pools information from a smaller neighbourhood.

As another approach, Maron et al. [176] prove that "2-order invariant graph networks", GNNs which are built from stacked equivariant layers, are as powerful as the 3-WL algorithm and show their use in practice. Vignac et al. [271] go beyond 1-WL using unique node identifiers. Bouritsas et al. [37] increase the expressive power of GNNs by augmenting node features with information about which subgraphs in the graph each node belongs to and what role it has in each subgraph. If particular sets of subgraphs are checked, the resulting model has expressivity beyond 1-WL. Bevilacqua et al. [28] propose representing a graph as a set of its subgraphs, since whilst two graphs may not be distinguishable by 1-WL, some of their subgraphs may be. They explore four different methods for sampling subgraphs and prove that their model can distinguish graphs that are indistinguishable by 3-WL. Finally, Papp and Wattenhofer [205] consider four different GNN augmentation strategies (e.g. higher-order GNNs and appending substructure counts) and compare their expressivity by providing examples of graphs that can be distinguished by one of the augmented GNNs, but not by another.

Extensions to WL Several extensions to the WL algorithm have been proposed. Souza et al. [243] extend WL to the temporal setting, proposing a 1-WL test for temporal graphs. They prove that injective versions of a popular temporal GNN,

message passing temporal graph networks [226], are equivalent to the temporal 1-WL test. Huang et al. [122] consider the task of link prediction and define a relational WL algorithm to describe the expressivity of such models. In particular, they consider "conditional MPNNs", which encode representations of nodes conditional on a query (two nodes and a relation) and yield representations for pairs of nodes. They prove that conditional MPNNs can have the same expressive power as the relational WL algorithm.

Morris et al. [189] review and summarize works which have investigated the expressive power of GNNs with respect to their distinguishing power, particularly in relation to the WL test. Finally, Huang et al. [123] consider Knowledge Graph Foundation Models (KGFMs), which is a GNN architecture that is inductive on both constants and relations. They prove their expressivity by means of their separation power; however, instead of considering their ability to distinguish pairs of non-isomorphic graphs, they consider their ability to distinguish collections of graph motifs around candidate links.

3.3.2 Approximation Ability

Rather than considering their distinguishing power, other works have investigated GNNs in terms of their ability to approximate various functions. In Chapter 4, I prove the ability of different extensions of MPNNs (with global readout) to approximate various node-labelling functions.

Similar to universal approximation theorems for general neural networks [64], Maron et al. [177] consider neural networks that are restricted to being invariant and prove that they can approximate any invariant function on a bounded number of nodes, provided that they use high-order tensors: this is a non-uniform expressivity result. They also prove that certain invariant functions *require* higher-order tensors to be approximated. GNNs (used for graph tasks) are a special case of this that use first-order tensors (vectors), since they typically use vector node features. Keriven and Peyré [134] provide an alternative proof of the results of Maron et al. [177], as

well as extending the ideas to equivariant neural networks, which subsume most GNNs that are used for node-level tasks.

Azizian and Lelarge [12] consider a variety of GNN architectures and prove approximation results for them, particularly MPNNs, k-LGNNs [175], and k-FGNNs [176]. They find that FGNNs have the best approximation power among these three architectures when using k-order tensors. They also provide universal approximation results for k-FGNNs. However, many of the above universality results depend on constructions of GNNs that are intractable in practice [12].

As another approach, Abboud et al. [2] consider GNNs augmented by appending random noise to their initial node features. Invariance / equivariance is broken in this case, but the GNNs retain equivariance in expectation. Most importantly, such GNNs provide universal approximation for invariant functions (in the non-uniform setting, i.e. for any function on a set of graphs that are upper-bounded in the number of vertices they have). This augmentation does not add significant computational overhead like the above architectures do. They consider different distributions for sampling the random noise: normal, uniform, Xavier normal, and Xavier uniform [101], as well as experimenting with different amounts of random noise. Their experiments find that the method increases the expressive power of MPNNs in practice.

3.3.3 Substructure Counting

Several works describe the expressive power of GNN models in terms of their ability to count substructures in the graph. One of the earliest examples of this is from Chen et al. [55], who consider GNNs in terms of their ability to count subgraphs and induced subgraphs. In particular, they prove that MPNNs cannot count induced subgraphs for connected patterns of three or more nodes, that they can count star-shaped subgraphs, and that k-WL cannot count induced path subgraphs of sufficiently high length. They propose a new GNN architecture for counting subgraphs and induced subgraphs. Papp and Wattenhofer [205] consider four different augmentations to GNNs and describe their expressivity in terms of

their ability to count cliques and cycles in the graph. For example, they prove that GNNs augmented with subgraph counts up to size k [37] can count k-cliques and k-cycles in the graph, but cannot count $(k+1)$-cycles.

Zhang et al. [322] abstract the idea of substructure counting to counting *homomorphisms*: they describe the expressivity of a GNN model by a set of all substructures that can be encoded by the model under homomorphism. They then use this definition to describe the expressivity of several classes of GNN models: MPNNs [99], subgraph GNNs [28], local GNNs [188], and folklore-type GNNs [176]. For example, the homomorphism expressivity of MPNNs is the set of forests. In closely related work, Jin et al. [129] describe the expressivity of models in terms of their abilities to express *graph motif parameters*. A graph motif parameter is a function that maps graphs into rational numbers and can be expressed as a finite linear combination of homomorphism counts into the graph.

3.3.4 Other Non-Logic-Based Methods

Before I discuss expressivity from the perspective of formal logic, I will briefly cover a couple of other approaches that have been used for GNN expressivity. Sato [229] discuss some approaches to GNN expressivity that depend on comparing them to "distributed local algorithms". For example, the MB(1) distributed algorithm has a number of connected nodes sending messages to each other simultaneously, where each node broadcasts the same message to each adjacent node. Sato et al. [230] show that MPNNs correspond to MB(1). Loukas [164] also points out the connection between MPNNs and local distributed algorithms, proves that MPNNs with unique node IDs achieve universal approximation power, and proves that MPNNs can lose significant expressive power when their depth (number of layers) and width (size of latent node features) are restricted.

In more contemporary work, Zhang et al. [321] consider GNN expressivity from the perspective of graph biconnectivity [10]. Biconnectivity includes a number of topics, including cut vertices, cut edges, biconnected components, and block cut trees. Zhang et al. [321] consider different GNN architectures in terms of their

ability to solve different biconnectivity problems: distinguishing whether a graph has cut vertices / edges, identifying all cut vertices or edges, and distinguishing graphs with non-isomorphic block cut trees. They prove that many GNN models, including MPNNs [99] and subgraph counting GNNs [37] cannot solve any of the above biconnectivity problems. They also find that the ESAN model [28] can identify both cut vertices and cut edges, and use those insights to provide a method for designing efficient GNNs that can solve biconnectivity problems.

Finally, as another approach, Kemper et al. [133] relax the 1-WL test to a version that models the possibility of information being lost during message passing. They use this to define "message passing complexity" (MPC), a continuous measure of how difficult a graph function is to learn. Intuitively, if there is a high probability of retaining the information needed to represent the function, then the task should have low complexity (and vice versa). They evaluate MPC in practice, showing that it better corresponds to empirical performance of MPNNs than other classical expressivity measures such as 1-WL.

3.3.5 Formal Logic

Finally, a prominent subset of methods approaches the question of GNN expressivity from the perspective of formal logic. In Table 3.1, I provide a summary of all such papers that I consider in this section. The foundational work is that of Barceló et al. [19], who consider first-order logical classifiers over graphs. I already presented a description of their approach to expressivity in Section 2.4.5 based on logical datasets. An MPNN (that performs binary node classification) is said to *capture* a logical classifier if the output of the MPNN and logical classifier coincide on every node from every graph. Note that this is a definition of *uniform* expressivity.

Foundational Results Barceló et al. [19] consider graded modal logic (GML) – equivalent to $\mathcal{ALCQ}$ – a fragment of FOL that permits only negation, conjunction, and guarded existential quantification with counting.

Barceló et al. [19] also consider FOC^2, a fragment of FOL that only allows formulas with two variables and allows counting quantifiers. They prove several foundational results, firstly that MPNNs (as a reminder, without global readout f^{read}) can capture any graded modal logic classifier. They also prove that any MPNN that is captured by a logical classifier can be captured by a graded modal logic classifier. This shows that MPNNs have the same first-order logical expressivity as graded modal logic; however, it does not show that every MPNN can be captured by graded model logic, in fact there are some MPNNs that cannot be captured by any FOL classifier. They also prove that any FOC^2 classifier can be captured by an MPNN (including global readout f^{read}). Hauke and Wałęga [113] answer an open question posed by Barceló et al. [19] by proving that the logical expressivity of MPNNs (with global readout) strictly exceeds that of FOC^2: i.e. that there exists an MPNN (with global readout) equivalent to a FOL classifier such that there is no equivalent FOC^2 classifier.

Expressivity with Extensions of FOL Grohe [103] considers uniform and non-uniform cases. He proves that there is a direct correspondence between graph queries that can be computed by certain bounded depth MPNNs and those that can be defined in the guarded logical fragment GFO+C, which uses FOL with counting and built-in relations. Nunn et al. [201] present an extension of modal logic, $K^{\#}$, which combines counting modalities and linear constraints. They then show that every $K^{\#}$ classifier has an MPNN that captures it, and vice-versa that every MPNN has a $K^{\#}$ classifier that it captures: this is possible since the logic goes beyond FOL.

Benedikt et al. [24] use Presburger logics [23] to answer a number of expressivity questions about GNNs, which are logics that have the ability to perform arithmetic on the number of elements satisfying a query. They consider the expressivity of some GNNs beyond FOL, explore expressivity when using directed graphs, and finally compare unbounded and bounded activation functions. For example, they show that sum-GNNs have expressivity beyond that of graded modal logic.

Variations in GNN Architecture Other papers provide a more fine-grained analysis of variations in GNN architectures. Barceló et al. [20] analyse the language MPLang [96], a declarative language using arithmetic operators which corresponds to the computation of MPNNs. They focus particularly on how model expressivity changes based on the choice of activation function. They find that linear models without activation functions are less expressive than modal logic, that all eventually constant activation functions (e.g. sigmoid) have the same expressive power as one another, and that ReLU activation functions have strictly more expressive power than any eventually constant activation function. Tena Cucala and Cuenca Grau [253] consider max-GCNs and prove that they can be expressed as the composition of a series of logical programs expressed in the language of Datalog with negation-as-failure.

Finally, Schönherr and Lutz [235] consider mean-GNNs in the uniform and non-uniform settings. In the uniform setting, they prove that any continuous mean-GNN (that is equivalent to a FOL classifier) has an equivalent logical classifier expressible in alternation-free modal logic (AFML). In Chapter 7, I also consider some questions around the logical expressivity of mean-GNNs, albeit a more restricted version than the models considered by Schönherr and Lutz [235].

Restricted GNNs Some works consider various restricted definitions of GNNs. Khalife [135], for example, consider the expressivity of GNNs with rational activation functions (i.e. fractions of polynomials). Cuenca Grau et al. [63] consider GNNs with bounded aggregation, i.e. aggregation functions which only consider a bounded number of each repeating element in the multiset of neighbourhood features during aggregation. The aggregation function max-k-sum, for example, is k-bounded. They prove that MPNNs with 1-bounded aggregation functions correspond to modal logic (equivalently, to $\mathcal{ALC}$), and that MPNNs with bounded aggregation functions correspond to graded modal logic (equivalently, to $\mathcal{ALCQ}$). They also prove similar expressivity results for MPNNs with bounded aggregation functions and global readout.

Paper	GNN Architecture	Logics
[19]	MPNNs	GML and FOC^2
[113]	MPNNs (with readout)	FOC^2
[103]	Bounded depth MPNNs	GFO+C
[201]	MPNN	$K^{\#}$
[20]	MPLang / MPNNs, activation functions	Modal logics
[24]	sum-GNNs, (un)bounded activations	Presburger logics
[235]	mean-GNNs	AFML, ML, GML
[253]	max-GCNs	Datalog + negation
[135]	GNNs with rational activations	GML
[63]	GNNs with bounded aggregation	ML, GML
[254]	non-negative max-GCNs	Datalog
[256]	non-negative max-sum-GCNs	Datalog + inequalities
[241]	HE-GNNs	Graded hybrid logic
[242]	max/sum-MPNNs, unique node IDs	GML, OI-FOL, FOL+C
[209]	RecGNNs, GSGNNs	FOL/GML + fixpoint
[29]	Halting recurrent GNNs	μGML (recursion)
[228]	Temporal GNNs	Product logics

Table 3.1: A summary of each of the works on GNN logical expressivity discussed in this literature review, in terms of which GNN architectures and logics are considered. ML, RML, and GML respectively denote modal, ratio modal, and graded modal logics. OI-FOL and FOL+C respectively denote order-invariant FOL and FOL with counting.

Tena Cucala et al. [254] and Tena Cucala et al. [256] consider GCNs with non-negative weights and max and max-sum aggregation, respectively. Recall that such GCNs are referred to as "monotonic". In the former case, monotonic max-GCNs are found to have equivalent Datalog programs of tree-like rules, and in the latter, monotonic max-sum-GCNs have equivalent programs using Datalog with inequalities. The style of logical expressivity I consider is inspired by the above two works, and in Chapter 5, I extend the expressivity results of Tena Cucala et al. [256] to GCNs that use scoring functions for link prediction.

More Expressive GNNs Soeteman and ten Cate [241] define a more expressive GNN architecture, Hierarchical Ego Graph Neural Networks (HE-GNNs), which operate by alternating between node individualization and message passing steps. To characterize the model expressivity, they use graded hybrid logic, which extends

GML with a variable binder. They find that HE-GNNs correspond to graded hybrid logic in terms of their ability to distinguish graphs. In a similar line of work, Soeteman et al. [242] study the expressive power of MPNNs that have input graphs where each node has a real-valued feature that uniquely identifies it. They consider both max and sum aggregation. They provide lower and upper bounds on expressivity using variants of GML, order-invariant FOL, and FOL with counting.

Non-Standard GNNs Finally, several works consider the logical expressivity of non-standard GNNs. Pflueger et al. [209] use two different GNN architectures which do not have a fixed number of layers. Recurrent GNNs (RecGNNs) apply message passing layers until node features become stable. Graph-size GNNs (GSGNNs) determine the number of message passing layers based on the size of the input graph. They use the logic MMFP to characterize these models, which is FOL extended with a fixpoint operator: they define MMFP(GML$^-$) to be GML with inverse roles and the fixpoint operator. Similar to the result of Barceló et al. [19], they prove that a node classifier is captured by both a RecGNN and a local MMFP formula if it is captured by a MMFP(GML$^-$) formula. They prove the same result for GSGNNs.

Bollen et al. [29] also consider recurrent GNNs, but propose a new halting mechanism for them that ceases message passing once a function is satisfied for every node. They use the logic μGML [141], which is GML extended with recursion, to analyse the model expressivity. They prove that every μGML formula can be captured by their halting recurrent GNNs. Finally, Sälzer et al. [228] consider various temporal GNNs, such as recursive TGNNs [312] and time-and-graph TGNNs [93], and analyse their logical expressivity using product logics that combine temporal logics and modal logics.

3.4 Explainability of GNNs

Graph neural networks are black boxes, and their reliance on graph structure, node features, and deep neural network architectures make their predictions difficult to understand and interpret. Various methods have been proposed to

yield explanations for GNNs. *Local* explanations cover single predictions, whereas *global* explanations seek to explain the model's behaviour in general [236, 244]. The types of GNN explanations can be separated into the following: (1) which input features, nodes, edges, or subgraphs contributed most to the prediction, (2) what minimal changes would need to be made to the input graph to change the prediction, and (3) logical rules.

In this section, I describe the two primary approaches to generating explanations for GNNs: post-hoc GNN explainability and Self-Explainable GNNs. A prominent subset of both post-hoc and self-explainable methods that I cover are those based on formal logic, which are of particular relevance to my thesis. In Chapters 5 to 7, I extract explanations in the form of logical rules for different GCN models.

3.4.1 Post-Hoc GNN Explainability

This section covers methods that extract explanations after the model has been trained.

Learned Explanations GNNExplainer [311] is one of the first methods proposed for generating local post-hoc explanations for GNNs. The model learns masks for edges and node features by maximizing the mutual information between the GNN prediction on the masked graph and the GNN prediction on the original input graph. It regularizes for mask sparsity to encourage the explanations to be sparse.

PGExplainer [167] extend the approach to be more global by using a deep neural network: whilst GNNExplainer is trained separately for each instance needing explanation, PGExplainer is trained on multiple instances simultaneously. It too provides local explanations in the form of subgraph structures, albeit with probabilities assigned to each edge.

Perturbation-based Methods Other approaches measure the contribution of node / edges / subgraphs by testing how structural changes modify the model's output, where greater changes in output correspond to more important contributions. These all provide local explanations in the form of the subgraphs. SubGraphX

[317] evaluate the importance of different subgraphs using Shapley values [140]. They explore the space of subgraphs using Monte Carlo tree search. However, the computational cost of the approach can be high for large graphs. GraphSVX [79] solve this by approximating the Shapley values, using local perturbations and training a model to estimate marginal contributions. GStarX [328] further improve efficiency by considering only star-shaped subgraphs centred on important nodes.

Common Latent Feature Patterns Some methods use commonly occurring latent features within a GNN to obtain explanations. DISCERN [270] identifies sets of important feature values in hidden GNN layers that discriminate between classes for the GNN predictions. They use these values to obtain the parts of the input graph that are most important to the prediction, yielding local explanations. INSIDE-SHAP [130] extend this by computing Shapley values for the important feature values to quantify their contribution to the model predictions. Using these, they produce both local and global explanations in the form of arbitrary subgraph structures.

Other Non-Logic-Based Approaches PaGE-Link [329] provides a novel form of local GNN explanations: paths connecting node pairs. The method is specifically applied to GNNs that are used for link prediction: given two nodes that are predicted by the GNN to have a link, they first prune unimportant sections of the graph, then learn an edge mask to extract the most important paths between the two nodes for the prediction. On the other hand, GNNXEMPLAR [7] provides global explanations for GNN predictions by identifying important (exemplar) nodes and using their neighbourhoods. However, their neighbourhoods are not given directly as explanations; rather, they are given to LLMs which derive natural language explanations.

Several works [109, 124, 166], provide counterfactual explanations. Such methods identify minimal changes to the input graph that result in the model prediction changing. In this case, the explanation is the perturbed input graph. These explanations are particularly helpful in scenarios such as identifying loan eligibility, where feedback from the model can be used to adjust behaviour. For example, if

the model predicts that you can adjust part of the input to change the prediction from "approve-loan" to "reject-loan", then the loan applicant can use this to improve their eligibility.

Formal Logic Finally, I discuss several approaches to post-hoc GNN explainability that are based on formal logic. These methods bridge the gap between symbolic reasoning and machine learning, and are of particular relevance to the work of my thesis.

One of the earliest attempts in this direction is GLGExplainer [13]. It is built on top of other methods that produce local subgraph explanations: it takes such explanations as inputs and learns to combine them into logical formulas (using conjunction and disjunction) over different graph structures. This yields global explanations for output classes. GraphTrail [6] also produces global explanations, but removes the need for other local explainers by automatically identifying subgraph-level concepts using Shapley values. They provide logical rules as explanations, using conjunction, disjunction, negation, and XOR over the subgraph-level concepts. The rules are learned using symbolic regression [172], which obtains rules directly from the computation trees in the GNN message passing.

The work of Nunn et al. [201], which was already discussed in Section 3.3, identifies the logical expressivity of MPNNs (without global readout) using the logic $\mathrm{K}^{\#}$. Given an MPNN, they provide a construction of a $\mathrm{K}^{\#}$ formula such that the MPNN captures the formula. The construction can be done in polynomial time in terms of the size of the MPNN. In theory, this provides a global logical explanation for the MPNN; however, Nunn et al. [201] do not demonstrate that this is feasible in practice, or that the obtained explanation is at all useful. The $\mathrm{K}^{\#}$ formula's usefulness as an explanation is particularly in question due to the non-standard nature of the logic used.

Finally, Pluska et al. [210] define an extension of decision trees [39] and train them to approximate the operations of GNNs. The decision tree then serves as a global explanation, which can be represented by formulas from the modal logic $\mathcal{EMLC}$.

3.4.2 Self-Explainable GNNs

Unlike post-hoc explainability methods, Self-Explainable GNNs are designed to be interpretable as part of their model architecture and learning procedure. This has been approached in several different ways.

Information Bottlenecks First, there are classes of models that learn representations that compress the input, to encourage the model to only retain information relevant to the task. For GNNs, this means identifying subgraphs and features that are minimally sufficient for prediction. GIB [315] includes a learnable masking distribution over nodes, edges, and features that is used to identify important subgraphs. It also uses a regularization term to encourage learning compact and connected structures. VGIB [316] addresses issues that GIB has with unstable and computationally expensive training. It does this using a variational formulation [137]. Nodes that are still predictive despite the injection of noise into their features are useful as explanations.

VIB-GSL [248] extends VGIB by also considering uncertainty in the graph structure itself: VGIB only considers a fixed graph structure. GSATs [183] use GATs (with unnormalized attention) and inject stochasticity into the attention weights: the attention weights are used to randomly decide whether an edge is dropped during message passing. The learned attention scores operate as explanations, as they highlight subgraphs that contribute most to the model's predictions.

Prototypes Another prominent approach is to make predictions based on comparisons between the input graph / node and its similarity to one or more "prototypes", which are representative examples that the model has learned. Such models typically learn a finite set of prototypes which each encode a subgraph or feature, and classify an input graph / node based on which prototypes the input is most similar to. In this approach, the set of prototypes serve as a global explanation.

Prototype-based methods were first proposed by Dai and Wang [65], who learn subgraph embeddings as prototypes. During prediction, each node is encoded

into an embedding summarizing its local neighbourhood, which is then compared against the prototype embeddings. Local explanations are given by which prototype embeddings were most similar. ProtGNN [332] deals with graph classification, where each class is given a trainable set of prototype embeddings. Furthermore, each prototype embedding is associated with the nearest training subgraph, which is identified by Monte Carlo tree search. This means that prototypes (and thus explanations) can be represented as subgraphs.

PiGNN [216] improves the efficiency of ProtGNN by instead defining prototypes at the node level. PGIB [237] combines the prototype approach with that of GIB [315], which was discussed above. Finally, instead of learning prototypes in a latent embedding space, PxGNN [66] learns a set of prototype graphs directly using a graph generator. During training, the model jointly optimizes a set of prototype graphs created by a parameterized generator, and a function which measures similarity between an input graph and each prototype.

Concept-Based Other models aim to provide interpretability by learning higher-level abstractions that correspond to concepts understandable by humans. This can be achieved in different ways. One method is to enforce that the model represents predefined concepts in its latent space: then one can analyse which concepts are activated during model inference, and use them as explanations. Examples of such methods include GCW [212] and GNAN [22]. Another method is to learn structural patterns directly from the data, such as in KerGNN [85] and TIF [305]. These approaches typically have a model structure with 3 components: an encoder that maps the input graph into a vector of concept activations, a relevance scorer that assigns weights to each concept, and a predictor that aggregates concept scores to generate model predictions. Finally, TOEE [308] uses a prototype-style approach by pre-training the GNN on a synthetic set of graph motifs designed to provide universal concepts. During model inference, an input graph is encoded based on its similarity to each of the motif-based concepts, and classification is performed using the resulting vector.

3.4.3 Logic-Based Self-Explainable GNNs

Finally, there are a variety of approaches to Self-Explainable GNNs that are based on formal logic. DC-GNNs [14], for example, combine a Self-Explainable GNN (they provide several options) with an interpretable rule-based model. The rule-based model is a typically a sparse linear model over node features. A DC-GNN's output is then defined by a small MLP that adaptively combines the output of the GNN with the output of the interpretable model. The resulting explanations are hybrid: the GNN provides subgraphs, while the interpretable model provides feature-level logical rules (provided there sufficient sparsity in the model).

Another set of approaches is to constrain the GNN architecture, specifically to only having non-negative weights. This results in model architectures that correspond to logical fragments and enables the extraction of explanatory rules. Importantly, all the following approaches obtain rules that are provably sound for the model, rather than the other approaches to GNN explainability that do not provide such guarantees. LogiX-GIN [217] uses GCN layers with sum aggregation and threshold-based aggregation functions. This leads to each layer corresponding to GML rules, which can be directly extracted from the model. Composing them provides a global explanation for the GNN.

Tena Cucala et al. [254] and Tena Cucala et al. [256] consider GCNs with non-negative weights, monotonically increasing scoring functions, and max and sum aggregation respectively. They prove that the resulting GCNs are *monotonic under homomorphisms* (only under injective homomorphisms in the latter case). This property means that the models correspond to the immediate consequences of tree-like Datalog rules (without and with inequalities, respectively). Both provide the means for checking whether a rule is sound for the model, which can be used to generate local explanations. Both also provide constructions for equivalent programs for such GCNs; however, Tena Cucala et al. [254] only show this for max-GCNs up to a depth of 2, and the construction for sum-GCNs by Tena Cucala et al. [256] is intractable in practice. If they could be constructed efficiently, such programs would serve as provably sound global explanations.

Finally, Wang et al. [284] also use sum-GCNs with non-negative weights, but with ClippedReLU activation functions. They show how Datalog rules can be injected into the model weights to guide the model to learn the rules: to achieve this, they regularize the parameters of their network to encourage the parameters corresponding to injected rules to increase. Furthermore, they show how sound rules can be read off directly from the model weights. However, this is only shown in the case of a single-layer or two-layer GCN, whereas the approaches of Tena Cucala et al. [254, 256] extend to GCNs of arbitrary depth.

4

Orbit-Equivariant Graph Neural Networks

Contents

4.1 Introduction 57
4.2 Orbit-Equivariance 60
4.3 Non-Equivariant MPNN Architectures 65
4.4 Experiments 71
4.4.1 Datasets 71
4.4.2 Loss 73
4.4.3 Methodology 74
4.4.4 Results 75
4.5 Discussion 78

4.1 Introduction

Message Passing Neural Networks [99, 102, 294] leverage local message-passing operations that are invariant to permutations so that the resulting output does not fundamentally depend on how the input graph is reordered: this is referred to as permutation equivariance / invariance. This structural inductive bias enables MPNNs to effectively and efficiently learn functions combining local and global graph labels [21, 42].

It is well known that MPNNs are upper-bounded in expressivity by the Weisfeiler-Leman (1-WL) graph colouring algorithm and are thus limited in the classes of

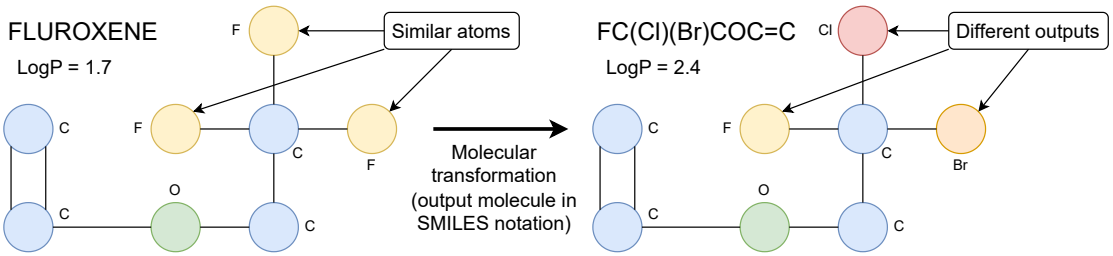


Figure 4.1: Non-equivariant molecular transformation that increases lipophilicity (LogP), where the nodes (atoms) are labelled with the atom type, without positional information, and the edges represent chemical bonds.

non-isomorphic graphs they can distinguish [186, 300]. Different models have been proposed to address this and other limitations, including appending unique IDs [73, 164] or random features [2, 231] to node labels, random dropouts [206], performing message passing over tuples of nodes [188], augmenting node features with subgraph information [37], and augmenting node features or message passing using graph-distance features [151]. Although significant advancements have been achieved, all these models are still designed with the primary objective of enhancing the expressiveness of MPNNs in the context of permutation equivariant or permutation invariant[1] functions.

As pointed out by Morris et al. [191], even MPNNs with universal (non-uniform) expressivity for equivariant functions cannot solve certain types of *symmetry breaking* graph problems. Equivariant functions cannot yield different outputs for *similar* nodes (i.e. structurally identical nodes with the same label). This presents problems in multiagent scenarios [191] where similar agents in a communication graph might need to take different actions to solve a task (e.g., drones that need to split up to find a target). Similarly, if MPNNs are used to transform molecules so as to optimize a molecular property, equivariant functions cannot transform similar atoms in different ways. An example of such a property is lipophilicity: the ability of a chemical compound to dissolve in fats, oils, and similar solvents. This is an important factor in drug design [8, 40, 286], as high lipophilicity often leads to undesirable effects [119, 202]. Figure 4.1 illustrates how optimizing the lipophilicity

[1]An equivalent notion to permutation equivariance, but for functions that yield an output for the whole graph, instead of each node.

of a fluroxene molecule requires two out of three similar fluorine atoms to be replaced with chlorine and bromine atoms. As noted by Lawrence et al. [145], symmetry breaking is also useful in generative modelling of atomic systems [232, 298] and molecular editing [160].

To tackle symmetry-breaking problems, I investigate functions that generate a multiset of output labels for each equivalence class, called an *orbit*, formed by the node similarity relation; for instance, in Figure 4.1, the orbit of three fluorine atoms is replaced by a multiset of three new atoms. The assignment of labels to specific nodes should be considered arbitrary because all nodes are fundamentally identical within the same orbit. Thus, a more permissive property than equivariance is necessary, which I define as *orbit-equivariance*. This property stipulates that irrespective of how the input graph is reordered, the multiset of outputs for each distinct orbit should remain identical. This approach enables one to address symmetry-breaking problems while preserving the valuable inductive bias that "the manner in which the graph is rearranged should not impact the outcome".

My Contributions The contributions presented in this chapter are as follows. In Section 4.2, I define orbit-equivariance and place it in the hierarchy of node-labelling functions as strictly in-between equivariant and universal node-labelling functions. I introduce a property called `max-orbit`, which allows for further refinement of orbit-equivariant functions. In Section 4.3, I establish a connection between the 1-WL graph isomorphism test and the identification of orbits, using it to assess the capacity of MPNNs to differentiate between the orbits within a graph. I exploit these insights to examine four distinct MPNN-based neural architectures, two of them novel, which are capable of learning non-equivariant functions and examine the theoretical expressivity of each architecture. In Section 4.4, I define a custom loss function designed to facilitate the training of orbit-equivariant neural networks. To evaluate the performance of my models, I conduct empirical assessments on two novel datasets: Bioisostere and Alchemy-Max-Orbit. The former stems from a real-world scenario involving the optimization of drug effectiveness. Neither

of these tasks can be fully addressed using equivariant functions, but both can be tackled with the aid of an orbit-equivariant function. Full proofs for all my technical results are given in Appendix A.1.

This chapter is based on a paper published in ICLR 2024 [192].

Background For ease of notation in this chapter, we consider graphs G with integer nodes: $V(G) = \{1, 2, ..., n\}$. Furthermore, we consider only graphs without edge colours; however, I anticipate that these definitions and results will extend seamlessly to graphs with coloured edges and/or global labels. Likewise, for ease of presentation, we consider only undirected graphs in the definitions, proofs, and examples.

Concepts that will be particularly relevant for this chapter include orbits and graph automorphisms (Section 2.2.3), node-labelling functions and equivariance (Section 2.2.4), and MPNNs (Section 2.3.1).

4.2 Orbit-Equivariance

In this section, I first prove that any equivariant function cannot distinguish similar nodes. To address this issue, I provide a relaxation of equivariance, called *orbit-equivariance*, which retains important inductive biases whilst allowing for differing outputs for similar nodes. I then position orbit-equivariant functions in relation to equivariant functions and node-labelling functions in general. Next, I define an additional property to count the number of distinct values in the output of a node-labelling function for each graph orbit. I use this to provide a more fine-grained characterization of orbit-equivariant functions. Finally, I also prove that, given an orbit-equivariant function, if one imposes the restriction of only having a single value in the output of each orbit, then one obtains an equivariant function.

Motivating and Defining Orbit-Equivariance As discussed in Section 4.1, equivariance may be too restrictive in applications where optimizing some graph-level property (such as lipophilicity) requires similar nodes to be differentiated.

Indeed, equivariant functions will always produce the same output for nodes within the same graph orbit.

Proposition 3. *Let f be an equivariant node-labelling function and let G be a labelled graph in its domain. If $v, w \in V(G)$ are similar, then $f(G)_v = f(G)_w$.*

Proof. Define σ to be a permutation that maps $v \to w$, and maps other nodes such that σ is an automorphism of G. Such a permutation exists since v and w are similar. Since σ is an automorphism, $\sigma \cdot G = G$. So we prove:

$$\begin{aligned} f(G)_v &= (\sigma^{-1} \cdot \sigma \cdot f(G))_v && \text{(identity permutation)} \\ &= (\sigma^{-1} \cdot f(\sigma \cdot G))_v && \text{(equivariance)} \\ &= f(\sigma \cdot G)_w && (\sigma^{-1} \text{ maps } w \to v) \\ &= f(G)_w && (\sigma \text{ is an automorphism}) \end{aligned}$$

□

I next introduce *orbit-equivariance* as a relaxation of equivariance that retains important structural inductive biases [21, 42] while extending the range of problems that can be solved. This is accomplished by mandating that, regardless of the permutation applied to the input graph, the multiset of outputs associated with each distinct graph orbit should not change.

Definition 4. *A node-labelling function f on domain $\mathcal{G}$ closed under permutation is **orbit-equivariant** if, for all labelled graphs $G \in \mathcal{G}$, permutations σ on $V(G)$, and orbits $r \in R(G)$, it holds that $\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{f(G)_v \mid v \in r\}\!\}$.*

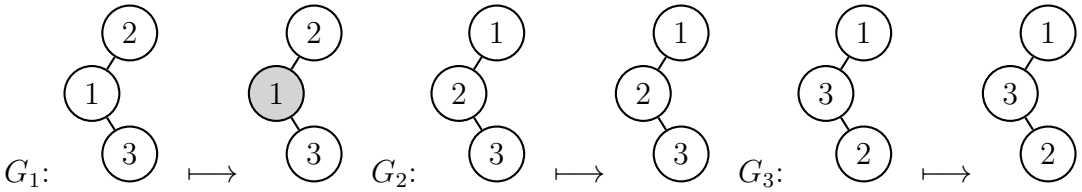
Characterizing Orbit-Equivariant Functions I now position orbit-equivariance within the hierarchy of node-labelling functions, by proving the following proposition. It shows that the class of orbit-equivariant functions lies strictly in-between all node-labelling functions and equivariant functions.

Proposition 5. *All equivariant functions are orbit-equivariant, but not vice-versa. There exist node-labelling functions which are not orbit-equivariant.*

Proof. In the following examples, we define graphs with the vertex set $V(G) = \{1, 2, 3\}$. Node colours represent their Boolean labels, where *white* represents 0 and *grey* represents 1. We begin by demonstrating that not all node-labelling functions are orbit-equivariant.

There exist node-labelling functions which are not orbit-equivariant.

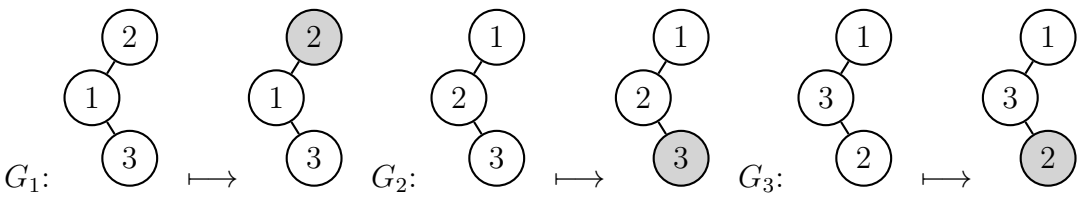
Consider a node-labelling function f that exhibits the following behaviour for distinct graphs G_1, G_2, G_3 and assigns 0 to all node labels of each other graph.



This function is not orbit-equivariant. Consider the following permutation, $\sigma = \{(1,2),(2,1),(3,3)\}$, for which $\sigma \cdot G_1 = G_2$, and consider orbit $r = \{1\}$ of G_1. Then $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{f(G_2)_2\}\!\} = \{\!\{0\}\!\}$ and $\{\!\{f(G_1)_v \mid v \in r\}\!\} = \{\!\{f(G_1)_1\}\!\} = \{\!\{1\}\!\}$. So there exists an orbit $r \in R(G_1)$ and permutation σ such that $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r\}\!\} \neq \{\!\{f(G_1)_v \mid v \in r\}\!\}$.

There exist orbit-equivariant functions which are not equivariant.

For an example of an orbit-equivariant function that is not equivariant, consider the node-labelling function f that maps input graphs G_1, G_2, G_3 as shown and assigns 0 to all node labels of each other graph.



We first show that orbit-equivariance holds for G_1. There are two orbits of G_1: $r_1 := \{1\}$ and $r_2 = \{2, 3\}$. Consider the permutation $\sigma = \{(1,2),(2,1),(3,3)\}$.

Note that $\sigma \cdot G_1 = G_2$ and $\sigma \cdot G_2 = G_1$. For r_1, we have $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r_1\}\!\} = \{\!\{f(G_1)_v \mid v \in r_1\}\!\} = \{\!\{0\}\!\}$, and for r_2, we have $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r_2\}\!\} = \{\!\{f(G_1)_v \mid v \in r_2\}\!\} = \{\!\{0, 1\}\!\}$. A similar argument holds for the other graphs and permutations. Thus, f is orbit-equivariant.

However, it is not equivariant. For the same permutation σ, $f(\sigma \cdot G_1) = f(G_2)$, which is a graph with the respective node labels of $(0, 0, 1)$. However, $f(G_1)$ is a graph with the respective node labels of $(0, 1, 0)$, so $\sigma \cdot f(G_1)$ is a graph with the respective node labels of $(1, 0, 0)$. So $f(\sigma \cdot G_1) \neq \sigma \cdot f(G_1)$. Furthermore, we can show that each equivariant function is orbit-equivariant, resulting in the following proposition.

It only remains to prove the following claim:

All equivariant functions are orbit-equivariant.

Let a node-labelling function f on domain $\mathcal{G}$ be equivariant. Then for all labelled graphs G and permutations σ on $V(G)$, $f(\sigma \cdot G) = \sigma \cdot f(G)$. To prove that f is orbit-equivariant, choose arbitrary labelled graph $G \in \mathcal{G}$, permutation σ on $V(G)$, and let $r \in R(G)$ be any orbit of G. Then,

$$\begin{aligned}
\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} &= \{\!\{(\sigma^{-1} \cdot f(\sigma \cdot G))_v \mid v \in r\}\!\} && \text{(permute vector)} \\
&= \{\!\{(\sigma^{-1} \cdot \sigma \cdot f(G))_v \mid v \in r\}\!\} && \text{(equivariance)} \\
&= \{\!\{f(G)_v \mid v \in r\}\!\} && \text{(inverse permutation)}
\end{aligned}$$

Thus, f is orbit-equivariant.

□

Counting Distinct Values in the Orbits I proceed to establish a taxonomy of orbit-equivariant functions, categorizing them according to the count of distinct values in the output for each orbit. This refined definition offers a more fine-grained characterization of orbit-equivariant functions, facilitating the identification of functions that might be empirically challenging to learn and aiding in the development of tools, particularly neural networks, for approximating such functions.

Definition 6. *For a node-labelling function f on domain $\mathcal{G}$, we define* `max-orbit`(f) *to be the maximum across all $G \in \mathcal{G}$ and orbits $r \in R(G)$ of the cardinality of the set $\{f(G)_v \mid v \in r\}$. If this is unbounded, we define* `max-orbit`$(f) := \infty$.

For any equivariant function f, `max-orbit`$(f) = 1$ as a direct consequence of Proposition 3. For any $m \in \mathbb{Z}^+$, there is an orbit-equivariant function f such that `max-orbit`$(f) = m$, since there are graphs (e.g. cycle graphs) with arbitrarily large orbits.

Proposition 7. *If f is orbit-equivariant and* `max-orbit`$(f) = 1$*, then f is equivariant.*

Proof. Let G be a labelled graph in the domain $\mathcal{G}$ of f and σ a permutation on G. RTP: $f(\sigma \cdot G) = \sigma \cdot f(G)$.

Since `max-orbit`$(f) = 1$, the maximum across all orbits $r \in R(G)$ of the number of unique values in $\{\!\{f(G)_v \mid v \in r\}\!\}$ is 1. Thus, $\forall r \in R(G),\ \forall v, w \in r,\ f(G)_v = f(G)_w$.

Since f is orbit-equivariant:

$$\forall r \in R(G),\ \{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{f(G)_v \mid v \in r\}\!\}$$

$$\Rightarrow\ \forall \text{ orbits } r \in R(G),\ \forall v \in r,\ f(\sigma \cdot G)_{\sigma(v)} = f(G)_v \qquad \text{(RHS multiset all equal)}$$

$$\Rightarrow\ \forall v \in V(G),\ f(\sigma \cdot G)_{\sigma(v)} = f(G)_v \qquad \text{(all orbits = all nodes)}$$

$$\Rightarrow\ \forall v \in V(G),\ (\sigma^{-1} \cdot f(\sigma \cdot G))_v = f(G)_v \qquad \text{(permute vector)}$$

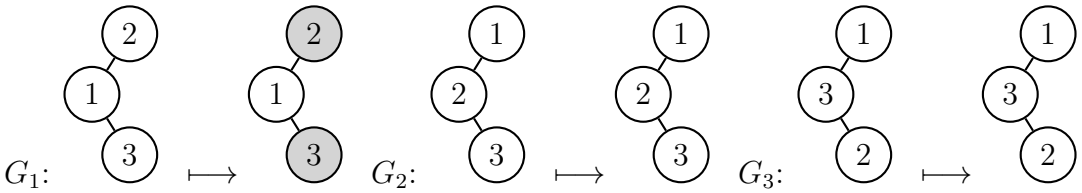
$$\Rightarrow\ \forall v \in V(G),\ (\sigma \cdot \sigma^{-1} \cdot f(\sigma \cdot G))_v = (\sigma \cdot f(G))_v$$

$$\Rightarrow\ \forall v \in V(G),\ f(\sigma \cdot G)_v = (\sigma \cdot f(G))_v \qquad \text{(inverse permutation)}$$

$$\Rightarrow\ f(\sigma \cdot G) = \sigma \cdot f(G)$$

□

The proposition above illustrates that equivariance is a restricted version of orbit-equivariance: when one requires that the output for each orbit only contains a single unique value. Recall that this restriction is exactly the property of equivariance that we wanted to relax. The example below of a function f shows why $\texttt{max-orbit}(f) = 1$ alone is not sufficient to obtain equivariance. Function f maps input graphs G_1, G_2, G_3 as shown, and assigns 0 to all node labels of each other graph.



Certainly $\texttt{max-orbit}(f) = 1$, but the function is not equivariant, since for permutation $\sigma = \{(1,2),(2,1),(3,3)\}$, $f(\sigma \cdot G_1) = f(G_2)$ is a graph with the respective node labels of $(0,0,0)$. However, $f(G_1)$ is a graph with the respective node labels of $(0,1,1)$, so $\sigma \cdot f(G_1)$ is a graph with the respective node labels of $(1,0,1)$. Thus, $f(\sigma \cdot G_1) \neq \sigma \cdot f(G_1)$.

4.3 Non-Equivariant MPNN Architectures

In this section, I discuss four MPNN-based neural architectures for solving orbit-equivariant problems which extend standard equivariant MPNNs in a way that is independent of the specific details of the MPNN. For each of these extensions, I provide a classification of their expressivity and in Appendix A.3.1 I analyse their computational complexity.

Orbit-1-WL Before delving into the details of these architectures, I first argue that the 1-WL graph colouring algorithm [189, 287] for isomorphism testing can also be used to compute graph orbits. This is achieved by computing iterative recolourings in the usual way, and then using the final stable colourings to partition the nodes into their orbits (instead of comparing them to another histogram of colours as in the standard isomorphism testing use case). When 1-WL is used in this manner, we refer to it as *orbit-1-WL*. Theorems 1 and 2 of Morris et al. [186] prove

that MPNNs and 1-WL respectively model each other's computation, which implies that MPNNs can distinguish between orbits if and only if orbit-1-WL can. That is: for nodes $v, w \in V(G)$, there exists an MPNN f such that $f(G)_v \neq f(G)_w$ if and only if the final colours of v and w are different when orbit-1-WL is applied to G.

Orbit-1-WL is sound but not complete: there exists a graph G and nodes $v, w \in G$ such that orbit-1-WL assigns the same colourings to v and w, but v and w are not similar. Examples of such graphs are shown in Appendix A.2.3. The following theorem is proved in Appendix A.1.4.

Theorem 8. *If orbit-1-WL assigns different colourings to $u, v \in V(G)$, then u, v are not similar (i.e. orbit-1-WL is sound).*

Analogously to 1-WL [186], there is a strict hierarchy of orbit-k-WL algorithms such that for all $k \geq 3$ there are graphs for which orbit-k-WL correctly identifies the orbits but orbit-$(k-1)$-WL does not. This is proved in Appendix A.2.4.

Model Architectures We are now ready to discuss and analyse four distinct MPNN-based neural architectures that are able to capture non-equivariant functions. The first two have been proposed in prior work. As they are presented, all the models require that the nodes in the input graph are ordered (recall that in this chapter, we assumed that nodes are integers): however, the definitions and corresponding results extend to graphs where the nodes cannot be ordered.

(1) **Unique-ID-MPNN** [73, 164, 191]: this extension deterministically appends a unique ID to the label of each node before the MPNN is applied to the input graph.

(2) **RNI-MPNN** [2, 231]: some independently sampled random features are appended to the label of each node before the MPNN is applied to the input graph; max pooling is used for the global readout f^{read} in each MPNN layer. This yields a probabilistic model.

The remaining two architectures are novel and can be viewed as instantiations of the general approach illustrated in Figure 4.2, where we first apply a standard MPNN to the input graph, then compute the orbits using orbit-1-WL, and finally devise a mechanism for assigning labels to nodes within the identified orbits.

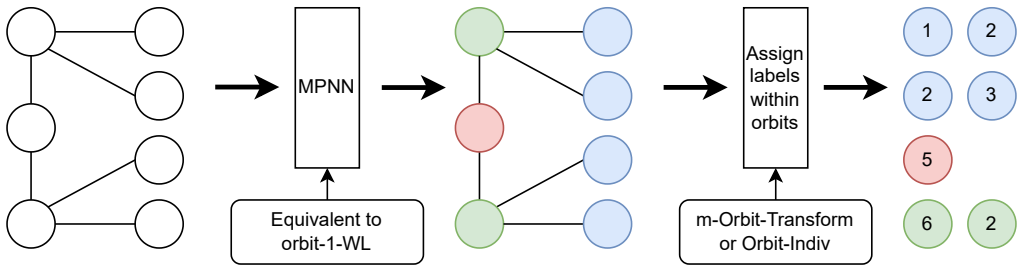


Figure 4.2: Architecture for constructing orbit-equivariant MPNNs, where different output labels (e.g. $1, 2, 3, 5, 6$) can be assigned within each graph orbit.

(3) **Orbit-Indiv-MPNN**: in this extension, the MPNN is first applied to the input graph G as usual to yield $f_{\text{MPNN}}(G)$. Orbit-1-WL is subsequently used to identify the orbits[2] of G; then, for each orbit, a unique ID from $\{1, ..., m\}$ is appended to each of the node labels, where m is the orbit size, yielding an updated labelled graph $f_{\text{orbit_indiv}}(f_{\text{MPNN}}(G)))$. A single MLP is finally applied to every extended node label in parallel and hence the overall function is defined as $f_{\text{mlp}}(f_{\text{orbit_indiv}}(f_{\text{MPNN}}(G)))$.

(4) m**-Orbit-Transform-MPNN**: this is a more advanced extension, and I illustrate its formal description with an example depicted in Figure 4.3; in my description, I use (*number) to refer to markers in the example. An MPNN (*1) is first applied to the input G to yield $f_{\text{MPNN}}(G)$ and the orbits are computed using orbit-1-WL; in the example, I demonstrate the model primarily using the orbit highlighted in blue. Each vector $f_{\text{MPNN}}(G)_v$ has $m \cdot (o+1)$ components, where o is the number of final output channels (often categories) and $m \geq 2$ is a parameter of the model; as a result, we can define m parts $f_{\text{MPNN}}(G)_v^1, ..., f_{\text{MPNN}}(G)_v^m$ in the output of $f_{\text{MPNN}}(G)_v$, each of them of size $(o+1)$ (*2). Then we apply $f_{\text{transform}}$ to $f_{\text{MPNN}}(G)$, which is defined as follows.

1. Iterate through each $v \in V(G)$, and set $G_v := f_{\text{MPNN}}(G)_v^1[1:o]$, where $[1:o]$ denotes the vector slice from 1 to o (inclusive). This is a vector of size o (*3).

[2]Since orbit-1-WL is not complete, these WL-orbits might not match the true orbits of the graph. However, computing the orbits of certain large graphs is intractable, and the models need to run efficiently to be trained on many examples: thus, I use orbit-1-WL, since it is efficient and the expressivity of the underlying MPNN is already limited by orbit-1-WL.

2. Iterate through each orbit $r \in R(G)$ (*4) and through each part $p \in \{2, 3, ..., \min(m, |r|)\}$ (*5). Select the p-th node v (*6) in orbit r, where r is sorted by ascending order of node index. Then, if $\text{argmax}(f_{\text{MPNN}}(G)_v^p) \neq o$, set $G_v := f_{\text{MPNN}}(G)_v^p[1 : o]$ (*7).

This yields a final model output of $f_{\text{transform}}(f_{\text{MPNN}}(G))$ (*8). Since f_{MPNN} is equivariant, for each orbit $r \in R(G)$ and for all nodes $v, w \in r$, $f_{\text{MPNN}}(G)_v = f_{\text{MPNN}}(G)_w$. Thus, the transformed output for an entire orbit r is essentially contained in the output of $f_{\text{MPNN}}(G)_v$ for any $v \in r$. Intuitively, the first part $f_{\text{MPNN}}(G)_v^1$ (*9) contains the default value for the orbit and each remaining part value $f_{\text{MPNN}}(G)_v^p$ (*10) represents that the label of a single node v in the orbit r should be swapped from the default to the given value. If the final channel of the part (with index o) is the largest (*11), this represents that the default should not be swapped out for the value in this part. The motivation for this is that it aids the model in learning functions which are "nearly equivariant": i.e. those in which within most orbits, most of the target labels within each orbit are the same.

Expressivity Results The following theorem establishes the expressivity of the above-described models with respect to equivariance and orbit-equivariance. It also proves the ability of Unique-ID-MPNNs and RNI-MPNNs to approximate different functions and the ability of Orbit-Indiv-MPNNs and m-Orbit-Transform-MPNNs to represent functions with different `max-orbit` values. The theorem is proved in Appendix A.1.5.

To formally define approximation, consider G_n, the set of graphs with $\leq n$ nodes (without node labels). Let f be a node-labelling function and $\mathcal{M}$ a GNN that both assign real-valued labels to each node in graphs from G_n. Then we say that $\mathcal{M}$ ϵ-approximates f if for all $G \in G_n$ and $v \in V(G)$, $|\mathcal{M}(G)_v - f(G)_v| \leq \epsilon$. If for a class of GNN models, $\forall \epsilon > 0$ there exists a GNN $\mathcal{M}$ in the class that ϵ-approximates f, then we say that the class can approximate f.

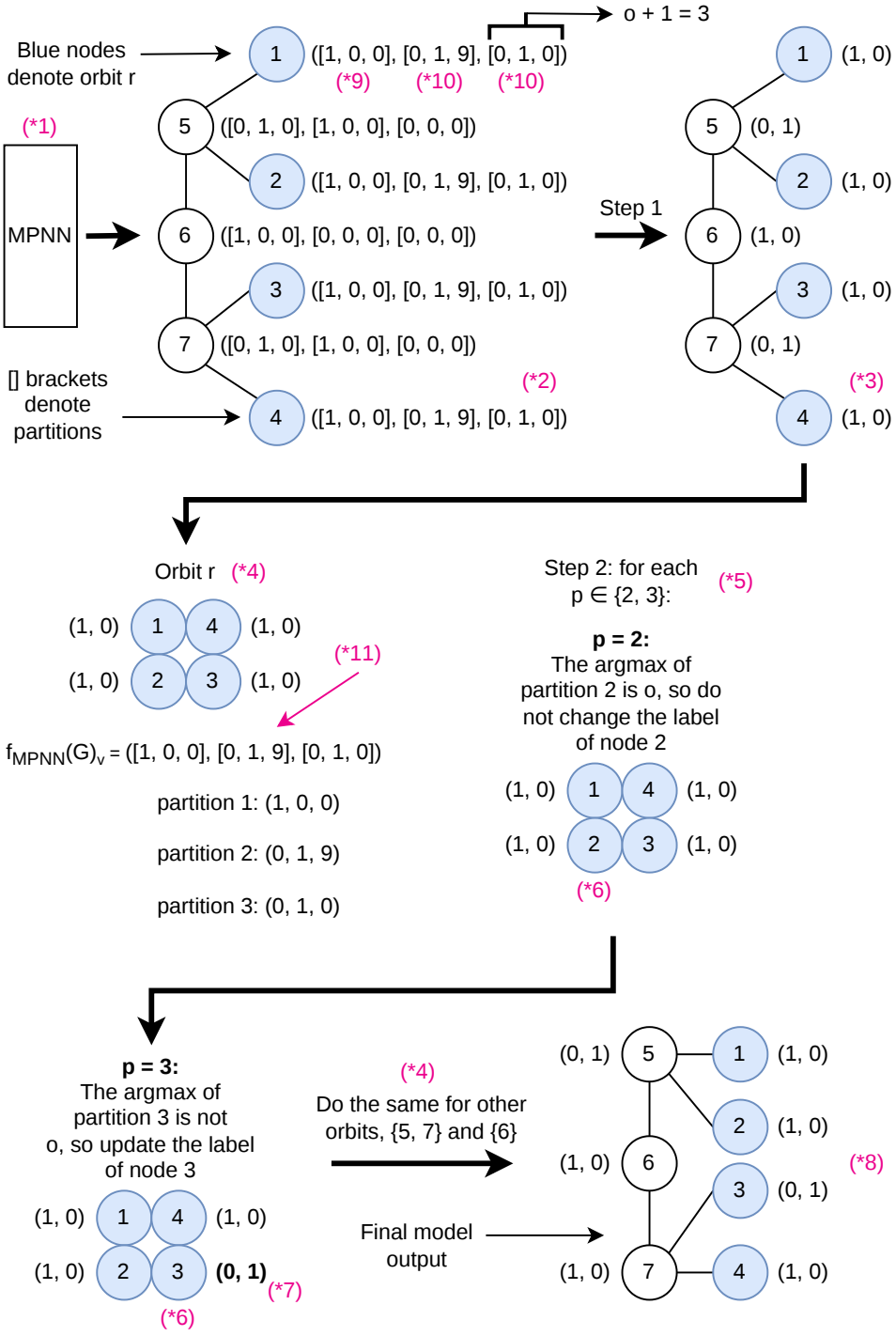


Figure 4.3: The transform operation of a 3-Orbit-Transform-MPNN with $o = 2$.

Since RNI-MPNNs are functions that include randomness, we cannot discuss them simply approximating functions. Instead, we need to refer to them approximating functions with a given probability; to do so, we extend the definitions of Abboud et al. [2] to node-labelling functions. Let f be a node-labelling function as before and let X be a randomized node-labelling function that associates with every graph $G \in G_n$ and node $v \in V(G)$ a random variable $X(G)_v$. Then X is an (ϵ, δ)-*approximation* of f if for all $G \in G_n$ and $v \in V(G)$ it holds that $\Pr(|f(G)_v - X(G)_v| \leq \epsilon) \geq 1 - \delta$. Note that an RNI-MPNN $\mathcal{M}$ computes such functions X. If X is computed by $\mathcal{M}$, we say that $\mathcal{M}$ (ϵ, δ)-*approximates* f. If for all $\epsilon, \delta > 0$, there is an RNI-MPNN that (ϵ, δ)-*approximates* f, then we abbreviate this by saying that the RNI-MPNN "approximates f with probability arbitrarily close to 1".

Theorem 9. *The expressivity of the above models is as follows, where for a class of models X, the claim "X is not equivariant" means that "there exists $f \in X$ such that f is not equivariant":*

1. *Unique-ID-MPNNs are not orbit-equivariant and, for any fixed n, can approximate any node-labelling function f that assigns a real value to each node, defined on the set G_n of graphs with $\leq n$ nodes.*

2. *RNI-MPNNs are equivariant in expectation and, for any fixed n, can approximate (with probability arbitrarily close to 1) any equivariant function f that assigns a real value to each node, defined on the set G_n of graphs with $\leq n$ nodes [2, 191]. They can approximate some functions that are non-equivariant and orbit-equivariant with probability arbitrarily close to 1, but there exist RNI-MPNNs which, with probability arbitrarily close to 1, are not orbit-equivariant.*

3. *Orbit-Indiv-MPNNs are not equivariant but are orbit-equivariant on graphs whose orbits are distinguishable by orbit-1-WL. For any $m \in \mathbb{Z}^+$, there exist Orbit-Indiv-MPNNs f with* `max-orbit`$(f) > m$.

4. *For any $m \geq 2$, m-Orbit-Transform-MPNNs f are not equivariant but are orbit-equivariant on graphs whose orbits are distinguishable by orbit-1-WL. They have* $\texttt{max-orbit}(f) \leq m$ *and there exist m-Orbit-Transform-MPNNs f with* $\texttt{max-orbit}(f) = m$.

In summary, all the above extensions to MPNNs go beyond equivariance and should be able to solve some orbit-equivariant tasks. However, Unique-ID-MPNNs and RNI-MPNNs are not strictly orbit-equivariant, whereas Orbit-Indiv-MPNNs and m-Orbit-Transform-MPNNs are. This provides the latter models with a stronger inductive bias for solving orbit-equivariant problems. Note that the universal approximation results stated in points (1) and (2) pertain to non-uniform expressivity, as they are only on sets of graphs up to a fixed number of nodes. Points (3) and (4) show that Orbit-Indiv-MPNNs are more expressive than m-Orbit-Transform-MPNNs for any m with respect to their ability to represent functions with different `max-orbit` values. In my experiments in the next section, I will test the performance of these models in practice on datasets with orbit-equivariant tasks.

4.4 Experiments

In this section, I describe the novel datasets used in my experiments, which are created from molecular datasets. I also define a novel loss function for training orbit-equivariant models. Next, I describe the setup of my experiments and my methodology. Finally, I present the results of my experiments and analyse the findings, establishing that the novel Orbit-Indiv-GCNs consistently exhibit the best performance.

4.4.1 Datasets

I propose a new dataset (**Bioisostere**) capturing a real-world use case and a new suite of synthetic datasets (**Alchemy-Max-Orbit**) designed to test the ability of models to learn functions with different max-orbit values. Although synthetic, Alchemy-Max-Orbit datasets consist of graphs obtained from real-world organic molecule data.

The task in **Bioisostere** is, given an input molecule as a graph[3], output a label for each node which represents either keeping the atom or swapping it out for some other particular atom, in such a way that the final resulting molecule has minimal lipophilicity; as discussed in Section 4.1, this is an important factor in drug design. Solving this task requires a non-equivariant function when an atom being swapped out comes from an orbit of size ≥ 2. To construct this dataset, I retrieved 961 small drug molecules from ChEMBL [181] and then used MMPDB [219] to compute many different bioisosteres for each molecule that each differ by at most one atom from the original molecule. Using RDKit [220], I computed the lipophilicity of each bioisostere and selected the one that yielded the lowest value as the target for each molecule. Of these 961 molecules, only 588 actually require some swapping out of an atom to achieve minimal lipophilicity, and only 156 require a non-equivariant function. The function f required to solve the task is orbit-equivariant with $\texttt{max-orbit}(f) = 2$.

An **Alchemy-Max-Orbit-**m dataset is constructed using Alchemy [52, 187], a large dataset of organic molecules. All graphs without an orbit of size $\geq m$ are removed, and the dataset is then extended by augmenting the existing graphs, if it contains fewer than 1000 graphs. I.e. new graphs are added to the dataset using two graph generators that derive new graphs by making small changes to graphs already in the dataset. Details on this process are in Appendix A.3.2.

The task in the dataset is to identify the largest orbit(s) in the graph and, for each largest orbit, distribute the labels $T := \{\!\{1, 1, ..., 1, 1, 2, ..., m\}\!\}$ at random without replacement among the orbit's nodes, such that $|T|$ coincides with the number of nodes in the orbit. Every node not part of a largest orbit should be assigned the label of 0. Formally, T is defined by $T := A \cup \{\!\{1, 2, ..., m\}\!\}$, where $A = \{\!\{1, ..., 1\}\!\}$ has $|A| = |r| - m$ and $|r|$ is the number of nodes in the orbit. Equivariant functions such as MPNNs cannot solve any of the examples in the dataset. The function f required to solve the task is orbit-equivariant with $\texttt{max-orbit}(f) = m$.

[3]In this graph, nodes represent atoms, the node labels are the atom type, and edges represent chemical bonds.

4.4.2 Loss

I focus on categorical outputs in my experiments, so I adopt cross-entropy loss as my baseline. For the loss to be representative of the model performance for orbit-equivariant problems, however, I need to account for the fact that I now support a multiset of outputs for each orbit, and that treating multisets as vectors may lead to suboptimal performance. I.e., an orbit-equivariant model can be seen as yielding a multiset of outputs for each graph orbit – thus, to properly assess the model accuracy using a loss function, it is not sufficient to merely compare the GNN output with the ground truth using cross-entropy loss.

Welleck et al. [288] investigate loss functions for multiset prediction, but purely in the context of the multiset being a collection of sequential decisions made by a policy. Similarly to the method of Probst [211], I use a deterministic algorithm to greedily pair up outputs and targets and then compute cross-entropy between them. Specifically, for each orbit, I sort the model outputs and target outputs by fixing some a-priori order upon the output categories, and then pair outputs and targets based on where they appear in the sorted lists. Computationally, this is $O(n \log n)$, so computing the loss is feasible even for large orbits. I refer to this method as *orbit-sorting cross-entropy loss*.

For the m-Orbit-Transform-MPNN models, loss is computed before the transform takes place, since the transform operation is not parameterized and does not require training. To obtain targets for the loss, I invert the transform operation on the ground truth outputs for the training set. This means that standard cross-entropy can be used, since the model is equivariant before the transform is applied. Full details on my novel loss function are given in Appendix A.3.4. I note that there are many other ways ([337], for example) in which this problem could be approached, and leave it as future work. I focus on evaluating different models in my experiments, rather than different loss functions.

4.4.3 Methodology

I perform my experiments on the Bioisostere, Alchemy-Max-Orbit-2, and Alchemy-Max-Orbit-6 datasets. To evaluate model performance, accuracy is measured and reported in 3 ways: the average proportion of nodes for which the correct output was predicted (node accuracy), the average proportion of orbits which have entirely correct predictions (orbit accuracy), and the proportion of graphs which were entirely solved (graph accuracy). Since we are dealing with orbit-equivariant functions, accuracy is calculated by collecting the orbits of the graph and computing the multiset intersections of model and target outputs for each orbit.

To define these metrics formally, consider an input graph G, model output $\mathcal{M}(G)$, and ground truth output $t(G)$. Then we define:

$$\begin{aligned}
\texttt{node-accuracy}(G, \mathcal{M}, t) &= \frac{\sum_{r \in R(G)} |\{\!\{\mathcal{M}(G)_v : v \in r\}\!\} \cap \{\!\{t(G)_v : v \in r\}\!\}|}{|V(G)|} \\
\texttt{orbit-accuracy}(G, \mathcal{M}, t) &= \frac{|\{r \in R(G) : \{\!\{\mathcal{M}(G)_v : v \in r\}\!\} = \{\!\{t(G)_v : v \in r\}\!\}\}|}{|R(G)|} \\
\texttt{graph-accuracy}(G, \mathcal{M}, t) &= 1 \text{ if } \forall r \in R(G),\ \{\!\{\mathcal{M}(G)_v : v \in r\}\!\} = \{\!\{t(G)_v : v \in r\}\!\} \\
&= 0 \text{ otherwise}
\end{aligned}$$

For each of these metrics, I report the average across all graphs G in the dataset. I use mean-GCNs [138] as the baseline MPNN to be augmented, since they are simple, effective, and still competitive in contemporary work [89, 120, 150, 218]. For each experiment, I train and test the following models: Unique-ID-GCN, RNI-GCN, Orbit-Indiv-GCN, and m-Orbit-Transform-GCN. Pure GCNs are equivariant and thus cannot achieve a better graph accuracy than 0 on the Alchemy-Max-Orbit datasets. However, they can still solve some subset of the examples in Bioisostere, so I include it as a baseline to see how an equivariant model performs.

Every experiment is run 10 times, with different seeds, and 10% of the data is randomly left out of training to be used as a test dataset. I report the mean and standard deviation across all seeds. I run each experiment with standard

Dataset	Model	Graph Accuracy	Orbit Accuracy	Node Accuracy
Bioisostere (cross-entropy loss)	GCN	52.4 ± 6.37	92.9 ± 1.14	94.4 ± 0.79
	Unique-ID-GCN	66.1 ± 5.13	94.5 ± 0.97	95.6 ± 0.62
	RNI-GCN	63.6 ± 4.29	93.9 ± 0.86	95.1 ± 0.76
	Orbit-Indiv-GCN	$\mathbf{69.9 \pm 4.68}$	$\mathbf{95.4 \pm 0.63}$	$\mathbf{96.3 \pm 0.49}$
	2-Orbit-Transform	57.1 ± 6.43	93 ± 0.99	94.1 ± 0.79
Alchemy-Max-Orbit-2 (orbit-sorting cross-entropy)	Unique-ID-GCN	20 ± 4.57	79.9 ± 2.01	77 ± 1.52
	RNI-GCN	0 ± 0	74.5 ± 1.7	75.2 ± 1.66
	Orbit-Indiv-GCN	$\mathbf{51.9 \pm 4.38}$	$\mathbf{87.5 \pm 1.78}$	$\mathbf{90.6 \pm 1.12}$
	2-Orbit-Transform	47.9 ± 6.45	86.8 ± 1.53	85.1 ± 1.66
Alchemy-Max-Orbit-6 (orbit-sorting cross-entropy)	Unique-ID-GCN	66.8 ± 7.15	84.8 ± 2.97	95.4 ± 1.07
	RNI-GCN	44.9 ± 7.19	78.5 ± 3.39	91.4 ± 1.47
	Orbit-Indiv-GCN	$\mathbf{83.4 \pm 4.22}$	$\mathbf{88.9 \pm 2.71}$	$\mathbf{97.1 \pm 1.46}$
	6-Orbit-Transform	10.6 ± 4.14	71.2 ± 2.47	87.6 ± 1.08

Table 4.1: Mean and standard deviation of final model accuracy percentage on the test datasets.

cross-entropy loss and with orbit-sorting cross-entropy loss. Full hyperparameters for my experiments can be found in Appendix A.3.3.

4.4.4 Results

My main results are shown in Table 4.1 and full results are shown in Appendix A.3.5. In the plots of Appendix A.3.5 and figs. 4.4 and 4.5, "max_orbit_gcn" refers to m-Orbit-Transform-GCN. Since I found that slightly better results were obtained for Bioisostere when standard cross-entropy is used, those results are shown, with orbit-sorting cross-entropy used for the other two datasets. Across all 3 datasets and metrics, Orbit-Indiv-GCNs consistently exhibited the best performance. GCNs exhibited the worst performance on Bioisostere and would have achieved an accuracy of 0 on the Alchemy-Max-Orbit datasets, highlighting the need for models that go beyond equivariance. Unique-ID-GCNs consistently outperformed RNI-GCNs, which aligns with the experiments of Morris et al. [191] on non-equivariant problems. It is also interesting to note that, whilst RNI-GCNs achieved the 3rd highest test accuracy on Bioisostere, they had the highest accuracy on the training set, suggesting that the model is more inclined to overfit.

On the Alchemy-Max-Orbit-2 dataset, 2-Orbit-Transform-GCNs achieved comparable performance to Orbit-Indiv-GCNs, whilst the other two models performed

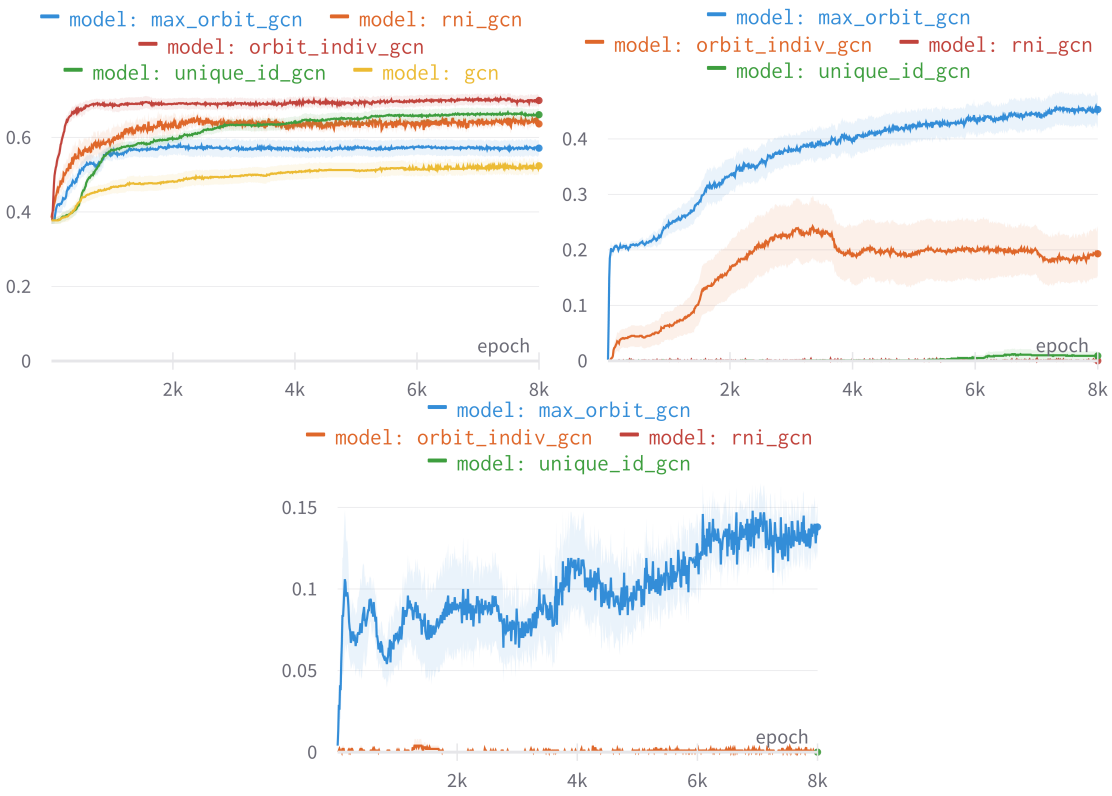


Figure 4.4: Graph accuracy with standard error on the test datasets across all models using cross-entropy loss: Bioisostere (left), Alchemy-Max-Orbit-2 (centre), and Alchemy-Max-Orbit-6 (right).

poorly. This suggests that the transformation model can be useful on certain tasks, when the max-orbit of the function is low. In contrast, on the Alchemy-Max-Orbit-6 dataset, 6-Orbit-Transform-GCNs had by far the worst performance among all the models, demonstrating that the transformation becomes infeasible for large max-orbit sizes. This is expected, since the number of output channels required for the transformation is linear in the max-orbit. Overall, whilst Alchemy-Max-Orbit-2 has smaller max-orbits than Alchemy-Max-Orbit-6, the models tend to perform better on the latter dataset because there are fewer graphs in Alchemy that possess a maximum orbit of 6 than those that possess a maximum orbit of 2, meaning that similar structures appear repeatedly when the Max-Orbit-6 graphs are augmented to extend the dataset.

Model performance on the test datasets during training is shown in Figure 4.4 for the experiments using cross-entropy loss, and in Figure 4.5 for those using

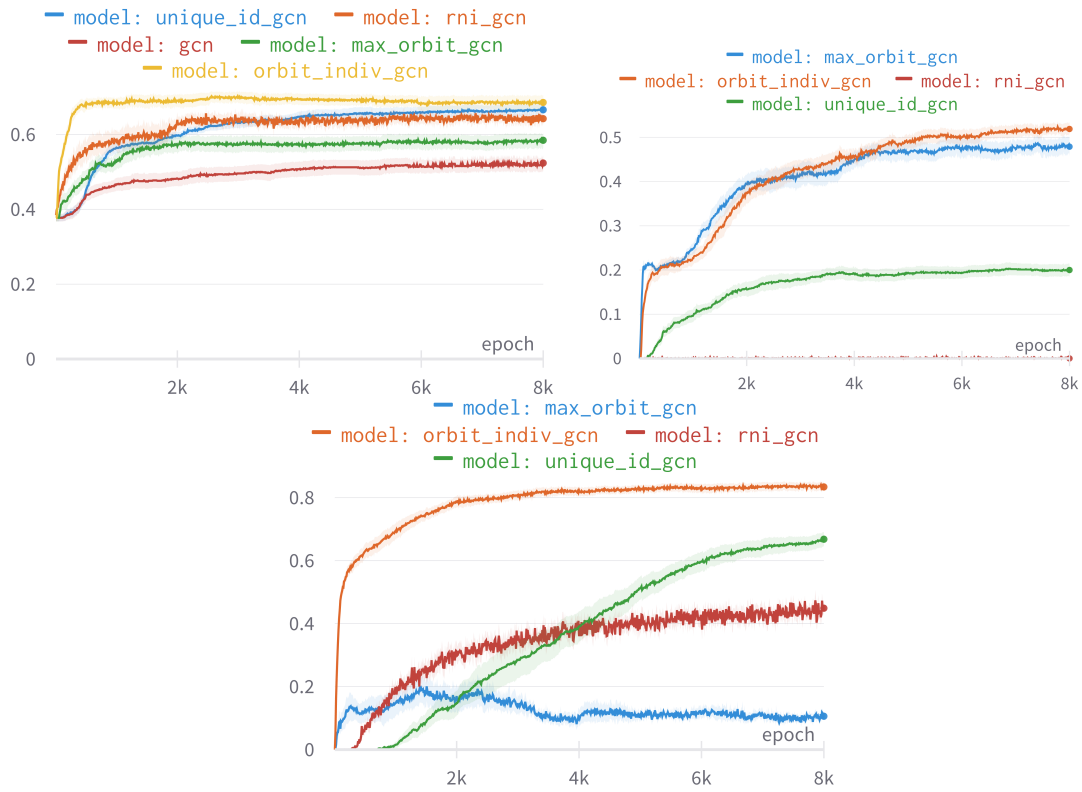


Figure 4.5: Graph accuracy with standard error on the test datasets across all models using orbit-sorting cross-entropy: Bioisostere (left), Alchemy-Max-Orbit-2 (centre), and Alchemy-Max-Orbit-6 (right).

orbit-sorting cross-entropy loss. In addition to Orbit-Indiv-GCNs performing the best out of all the models, they also converge by far the fastest. I attribute this to their strong structural inductive bias: they are exactly orbit-equivariant whilst not deviating much from the provenly effective architecture of GCNs. Abboud et al. [2], Morris et al. [191] found that RNI-GCNs can be slow to converge, but on Bioisostere, we find them to converge comparatively swiftly. A comparison between Figure 4.4 and Figure 4.5 highlights the importance of using orbit-sorting cross-entropy loss. The performance of Orbit-Transform-GCNs is mostly unaffected, since they are trained as equivariant models: this is an advantage unique to Orbit-Transform-GCNs. The models also do not show improvement on Bioisostere when using orbit-sorting cross-entropy, which is probably because each node from an orbit that is to be swapped out is chosen deterministically to be the one with the lowest index. This is not the case for the Max-Orbit datasets.

4.5 Discussion

In this chapter, I showed the limits of equivariant functions when it comes to producing different outputs for similar nodes. I defined a less restrictive property, orbit-equivariance, to address this issue while still retaining a strong structural inductive bias, and showed where it lies in the hierarchy of node-labelling functions. I also provided a brief taxonomy of orbit-equivariant functions using a further property, `max-orbit`. I presented four different ways to extend MPNNs to solve non-equivariant problems and analyse their expressivity. I evaluated these models on both real-world and synthetic datasets, finding that one of my novel models (Orbit-Indiv-MPNN) exhibited the best performance in all scenarios. I also found that the use of a specialist loss function can be necessary for the models to train.

Future Work The novelty of orbit-equivariance as a property provides considerable scope for future work in this area. First, the idea of orbit-equivariance can be generalized to other data structures besides graphs [42] and related to node embeddings [246]. Extensions to MPNNs have also proven to be useful for aiding in designing optimal bioisosteres, but this is a complex task and deserves further investigation. Moreover, the establishment of orbit-equivariance as a formal property could lead to the identification of other problems that require non-equivariant and orbit-equivariant models to solve. Finally, there may be ways to design better, specially tailored orbit-equivariant neural networks to solve such problems.

Limitations My work considers only graphs where the nodes are not individualized; this does not account for application domains where the nodes have noisy features that are almost identical, but not precisely. My theorems focus on property satisfaction and there are open questions as to how expressive the new proposed models are. The scope of my experiments is restricted to datasets where non-equivariant functions are required. Finally, the Bioisostere dataset does not capture all aspects of actual drug design, since I only optimize for a single molecular property instead of many, and do not consider biological effects.

5

Expressivity and Explanations for GCNs with Scoring Functions

Contents

5.1 Introduction 79
5.2 Sound Rules 82
5.2.1 Monotonically Increasing Scoring Functions 82
5.2.2 Checking Soundness of Rules 84
5.2.3 Leveraging Scoring Function Results 85
5.3 Equivalent Programs 88
5.3.1 Monotonic Max-GCNs 90
5.3.2 Monotonic Max-Sum-GCNs 92
5.4 Experiments 95
5.4.1 Experiment Setup 95
5.4.2 Results 98
5.5 Discussion 100

5.1 Introduction

In Chapter 4, I showed how MPNNs could be augmented to bring their expressivity beyond equivariant functions into the class of orbit-equivariant functions. Going forward, I will consider a popular instantiation of MPNNs, GCNs (as described in Section 2.3.2), which will allow me to conduct a more fine-grained analysis of their

expressivity within the class of equivariant functions. Furthermore, this will allow me to prove results that obtain sound logical explanations for their predictions.

A limitation of GCN models is that applying them to a KG produces real vector embeddings for the graph's vertices but leaves the edges untouched. Therefore, to predict new edges (the task known as link prediction), many approaches use a decoder based on a *scoring function* [281]—a learnable mapping of binary predicates and pairs of embeddings to a real-valued score—which can be easily applied to the embeddings of each pair of vertices in the graph. The score can then be compared against a threshold to decide whether a link should be predicted. This is a very popular approach for link prediction with GNNs [326], including the pioneering R-GCN [234], which uses DistMult [303] as a decoder.

Despite their effectiveness, the predictions of models using GNNs and scoring functions cannot easily be explained, verified, and interpreted [95]. Furthermore, their expressivity is not clearly understood yet either. In important prior work that has already been discussed, Tena Cucala et al. [254] consider monotonic max-GCNs and show that they can be faithfully characterized by programs of tree-like Datalog rules. Similarly, monotonic max-sum-GCNs [256], can be faithfully characterized by means of tree-like Datalog programs with inequalities. These results provide sound logical rules as explanations, as well as upper-bounding the logical expressivity of such GCNs. However, these models do not use scoring functions for link prediction; instead, they rely on an encoding-decoding strategy [254] that can predict new edges only between constants already linked in the input graph.

When it comes to GNNs and scoring functions, Huang et al. [122] characterize the expressivity of various GNN and scoring function models using a relational Weisfeiler-Leman algorithm. They also extend the results of Barceló et al. [19] to GNNs that perform link prediction relative to a fixed query relation and source vertex, showing that they can only express logical classifiers from a specific fragment of first-order logic. However, their work does not provide a means for obtaining sound rules from the GNNs. Thus, given the advantages and popularity of GNNs

with scoring function decoders, there is an important open question as to how sound rules may be obtained from them, as well as their logical expressivity.

My Contributions In this chapter, I extend the work of Tena Cucala et al. [256] to monotonic GCNs with scoring functions. I bound the logical expressivity of monotonic max-sum-GCNs with monotonic scoring functions from above using Datalog rules with inequality, and I provide a method to extract sound rules from these models to explain their predictions. In particular, in Section 5.2, I show how many popular scoring functions can be made *monotonically increasing*, provide a means for checking the soundness of Datalog rules that can be used to explain predictions of such models, and prove how existing results about which rule patterns are captured by different scoring functions affect the logical expressivity of the models. In Section 5.3, I also define procedures for obtaining finite equivalent programs of *tree-like* Datalog rules for (1) any monotonic GCN with max aggregation and monotonically increasing scoring function, and (2) any monotonic GCN with max-sum aggregation and a *non-negative bilinear* scoring function. In Section 5.4, I conduct experiments on benchmark link prediction datasets and also use LogInfer. I find that performance either drops slightly or increases when restricting the model to be monotonic, and that many sound rules can be extracted. Therefore, my models maintain performance comparable to standard models while offering the added benefit of explainability.

This chapter is based on a paper published in KR 2025 [194].

Background In comparison to Section 2.3.2, in this chapter we define GCNs without the classification function $\texttt{cls}_t$, such that the output of $\mathcal{M}$ is a graph with the same vertices and edges as the input graph, but where each vertex is labelled by $\mathbf{v}_L$. Note that this definition is the same as that of Tena Cucala et al. [256], except without the classification function. In particular, we consider GCNs with max-sum aggregation functions, which generalize both max and sum aggregation.

I use GCNs with families of scoring functions (Section 2.3.3). For example, recall that RESCAL is the family of scoring functions of the form $f(R, \mathbf{h}, \mathbf{t}) = \mathbf{h}^\top \mathbf{M}_R \mathbf{t}$,

where $\mathbf{M}_R$ is a real matrix of dimension $d_f \times d_f$. As described in Section 2.4.3, a GCN and scoring function can be used to realize a transformation $T_{\mathcal{M},f}$ from datasets to datasets. Soundness, completeness, and equivalence of Datalog rules/programs (Section 2.1.1) are defined in the same way as for standard GCNs: for example, a program or rule α is sound for a combination $(\mathcal{M}, f)$ of a GCN $\mathcal{M}$ and a scoring function f if $T_\alpha(D) \subseteq T_{\mathcal{M},f}(D)$ for each dataset D.

5.2 Sound Rules

In this section, I propose adaptions to several existing families of scoring functions to make them amenable to rule extraction when used with a monotonic max-sum-GCN—specifically, I make them *monotonically increasing*. Furthermore, I provide a means for checking whether a Datalog rule is sound for a monotonic max-sum-GCN and monotonically increasing scoring function. Finally, I show how existing results about which rule patterns are captured by various scoring function families affect the soundness of rules when they are used with a GCN.

5.2.1 Monotonically Increasing Scoring Functions

Recall that, for any Datalog program $\mathcal{P}$ (possibly containing inequalities), the operator $T_\mathcal{P}$ is *monotonic under injective homomorphisms*. Thus, Tena Cucala et al. [256] show that the ability to extract Datalog rules from plain monotonic max-sum-GCNs crucially depends on the fact that these models are also monotonic under injective homomorphisms, meaning that they are agnostic to the particular constants used, and if the model is applied to an arbitrary dataset, then the output values will never decrease as new facts are added to the input. I identify a property of scoring functions which ensures that combining such a function with a monotonic max-sum-GCN produces a transformation that is still monotonic in the same way.

Definition 10. *A scoring function f is* monotonically increasing *if for all relations R and vectors $\mathbf{h}, \mathbf{h}', \mathbf{t}, \mathbf{t}' \in \mathbb{R}^{d_f}$ such that $\mathbf{h}[i] \leq \mathbf{h}'[i]$ and $\mathbf{t}[i] \leq \mathbf{t}'[i]$ $\forall i \in \{1, ..., d_f\}$, we have that $f(R, \mathbf{h}, \mathbf{t}) \leq f(R, \mathbf{h}', \mathbf{t}')$.*

This property guarantees that a scoring function will never decrease its output for a pair of embeddings $\mathbf{h}$ and $\mathbf{t}$ if the values of those embeddings increase. We now consider various scoring function families from the literature and show that some are never monotonically increasing, whereas others can be restricted to be monotonically increasing.

There are a variety of "distance-based models" used for scoring binary facts. For example, TransE [32] has the scoring function $f(R, \mathbf{h}, \mathbf{t}) = -||\mathbf{h} + \mathbf{r}_R - \mathbf{t}||_2$, where each $\mathbf{r}_R$ is a vector of size d_f representing a fixed embedding for relation R. Since this involves a sum in which $\mathbf{h}$ and $\mathbf{t}$ have different signs, there is no collection of vectors $\{\mathbf{r}_R, \ldots\}$ such that the corresponding scoring function is monotonically increasing. This problem occurs in all the distance-based models, including UM [31], TransH [285], TransR [157], TransD [127], TransSparse [128], TransM [84], ManifoldE [296], TransF [86], TransA [295], RotatE [249], and HAKE [333]. A similar issue occurs with some non-distance-based models including ComplEx [261], QuatE [330], and DualE [47], when the computation is considered in terms of real vectors, and is explained in Appendix B.1.1.

In contrast, the following scoring function families rely on combinations of multiplication, addition, and non-linear functions; they can be modified by restricting their main parameters—usually expressed as vectors, matrices, tensors, or neural network weights—to contain only non-negative values, and by making all non-linear functions in them monotonically increasing and non-negative (e.g., ReLU).

For example, when restricting RESCAL [197] such that each $\mathbf{M}_R$ contains only non-negative numbers, f is monotonically increasing. This approach yields monotonically increasing scoring functions for the following other families: DistMult [303], ANALOGY [158], HolE [199], TATEC [94], TuckER [17], SimplE [131], SME [34], NTN [240], SLM [240], MLP [77], NAM [159], and ConvE [76]. These families are fully described in Appendix B.1.2 and Appendix B.1.3. This list is non-exhaustive – there may be other scoring function families that can also be modified in this way or others to yield monotonically increasing scoring functions.

5.2.2 Checking Soundness of Rules

I provide a means for checking whether a given Datalog rule is sound for a monotonic max-sum-GCN and monotonically increasing scoring function. First, I establish that using a monotonic max-sum-GCN and monotonically increasing scoring function yields a monotonic (under dataset inclusion) dataset transformation.

Lemma 11. *Let $\mathcal{M}$ be a monotonic max-sum-GCN and f a monotonically increasing scoring function. Then for all datasets D, D' such that $D \subseteq D'$, $T_{\mathcal{M},f}(D) \subseteq T_{\mathcal{M},f}(D')$.*

The lemma is proved in Appendix B.2.2 and relies on vertex labels yielded by the GCN not decreasing when facts are added to the input dataset, and outputs of f not decreasing as its input vectors increase. In addition to this, max-sum-GCNs and the canonical encoding are agnostic to the particular constants used in the input dataset: they depend only on the structure. These two results show that monotonic max-sum-GCNs with monotonically increasing scoring functions are monotonic under injective homomorphisms.

I now provide the proposition that allows one to check for sound rules, the proof of which is given in Appendix B.2.3 and relies on the above results, as well the fact that any dataset satisfying the body of r contains one of the following datasets D_μ (up to constant renaming). This is an analogous result to that of Tena Cucala et al. [256, Lemma 12].

Proposition 12. *Let $\mathcal{M}$ be a monotonic max-sum-GCN and f a monotonically increasing scoring function. Let r be a Datalog rule with a binary head atom H, a (possibly empty) set A of body atoms, and a (possibly empty) set I of body inequalities. For each variable x in r, let a_x, b_x be distinct constants uniquely associated with x. Then, r is sound for $(\mathcal{M}, f)$ if and only $H\mu \in T_{\mathcal{M},f}(D_\mu)$ for each substitution μ mapping the variables of r to constants in the set $\{a_x \mid x \text{ is a variable of } r\}$ such that $\mu(x) \neq \mu(y)$ for each inequality $x \not\approx y \in I$, where D_μ is defined by $D_\mu = A\mu \cup \{R(b_x, \mu(x)) \mid x \text{ is a variable occurring in } H \text{ but not in } A\}$ where R is an arbitrary but fixed binary predicate.*

The inclusion of each $R(b_x, \mu(x))$ in D_μ accounts for unsafe rules. A rule can be checked for soundness by enumerating each above-defined substitution μ and computing $T_{\mathcal{M},f}(D_\mu)$; then the rule is sound if and only if $H\mu \in T_{\mathcal{M},f}(D_\mu)$ for each such substitution. For example, to check the soundness of rule $r : P_1(x, y) \wedge U_1(x) \rightarrow P_2(x, y)$, we have four possible substitutions. The first two are:

1. $\mu_1 = \{x \mapsto a_x, y \mapsto a_x\}$
   $D_{\mu_1} = \{U_1(a_x), P_1(a_x, a_x)\}$
2. $\mu_2 = \{x \mapsto a_x, y \mapsto a_y\}$
   $D_{\mu_2} = \{U_1(a_x), P_1(a_x, a_y)\}$

and the others are symmetric to them. If $H\mu_i \in T_{\mathcal{M},f}(D_{\mu_i})$ for each substitution, then the soundness of r follows, since any dataset satisfying the body of r contains one of the above datasets, up to constant renaming.

5.2.3 Leveraging Scoring Function Results

In this section, I consider GCNs (not necessarily monotonic, or even max-sum) and exploit known results [1, 249] about the ability of a scoring function family to model different types of rules (referred to as "patterns"), to bound the logical expressivity of the GCN and scoring function. Pavlović and Sallinger [207] formally define the capturing of rule patterns. Intuitively, a scoring function family captures a pattern *exactly* if there exists a scoring function in the family such that, for all constant embeddings, if the body of the pattern is satisfied then the head of the pattern is satisfied. To formalize this using our definition of scoring functions, we first define a rule pattern.

Definition 13. *A* rule pattern *is a safe Datalog rule containing only binary predicates*[1] *from a set of meta predicates* $\mathcal{M} = \{M_1, M_2, ...\}$*, disjoint from the signature. A rule* r conforms *to a rule pattern* ρ *if it can be obtained from* ρ *by replacing each distinct meta binary predicate with a different binary predicate from the signature.*

[1]Since scoring functions can only predict binary facts, and patterns in the existing rule capturing literature do not use inequalities, we omit unary predicates and inequalities from the rules.

We can now formalize the capturing of rules. In addition to defining "capturing exactly", we define "capturing universally", which generalizes cases such as DistMult being symmetric.

Definition 14. *Let f be a scoring function, C a set of constants, and $e : C \to \mathbb{R}^{d_f}$ an embedding function. Then the tuple (f, C, e)* captures *a rule $B \to H$ if for every substitution μ such that $\mathcal{F}_f(C, e) \models B\mu$, we have $\mathcal{F}_f(C, e) \models H\mu$.*

A family of scoring functions F universally captures *a rule pattern ρ if for all $f \in F$, constants C, $e : C \to \mathbb{R}^{d_f}$, and any rule r conforming to ρ, we have (f, C, e) captures r.*

A family of scoring functions F exactly captures *a rule pattern ρ if for any rule r conforming to ρ, there exists $f \in F$ such that for all constants C and $e : C \to \mathbb{R}^{d_f}$, (f, C, e) captures r.*

Note that capturing universally specifies a restriction that the scoring function always satisfies. Below, I provide a sample of some pattern capturing results from the literature. Consider the following rule patterns:

1. Hierarchy: $M_1(x, y) \to M_2(x, y)$
2. Symmetry: $M_1(x, y) \to M_1(y, x)$
3. Intersection: $M_1(x, y) \wedge M_2(x, y) \to M_3(x, y)$
4. Inversion: $M_1(x, y) \leftrightarrow M_2(y, x)$
5. Composition: $M_1(x, y) \wedge M_2(y, z) \to M_3(x, z)$

Then, for example, DistMult universally captures symmetry and exactly captures hierarchy [1]. TransE exactly captures inversion, composition, and intersection.

We now analyse the effect that a scoring function tuple capturing a rule has when used with a GCN. We find that when a rule is captured across all constants and embeddings, certain rules being sound for the GCN and scoring function implies that there are other rules (not necessarily logically entailed by them) which must also be sound. This limits the class of equivalent programs for the model,

and thus its logical expressivity. This is formalized in the following proposition, which is proved in Appendix B.2.4.

Proposition 15. *Let f be a scoring function and $B \to H$ a safe rule such that for all constants C and $e : C \to \mathbb{R}^{d_f}$, (f, C, e) captures $B \to H$. Let $\mathcal{M}$ be a GCN. Assume that for each atom B_i of B, there is an inequality-free[2] rule $A_i \to B_i$ that is sound for $(\mathcal{M}, f)$, where each A_i is a conjunction of atoms. Then the rule $\bigwedge_i A_i \to H$ is sound for $(\mathcal{M}, f)$.*

As a direct consequence of this, notice that when using a scoring function family that universally captures a pattern, the result applies to any rule conforming to the pattern and scoring function in the family.

Corollary 16. *Let F be a scoring function family that universally captures a rule pattern ρ, $B \to H$ a rule conforming to ρ, $\mathcal{M}$ a GCN, and $f \in F$. Assume that for each atom B_i of B, there is an inequality-free rule $A_i \to B_i$ sound for $(\mathcal{M}, f)$. Then the rule $\bigwedge_i A_i \to H$ is sound for $(\mathcal{M}, f)$.*

For example, let F be a scoring function family that universally captures symmetry. Then for any GCN $\mathcal{M}$ and scoring function $f \in F$, if a rule $A \to P_1(x, y)$ is sound for $(\mathcal{M}, f)$, so is $A \to P_1(y, x)$.

A similar result applies to patterns captured exactly, except that it shows that there *exists* a scoring function from the family such that when certain rules are sound, others must also be sound, regardless of the parameters of the GCN.

Corollary 17. *Let F be a scoring function family that exactly captures a rule pattern ρ, and $B \to H$ a rule conforming to ρ. Then there exists $f \in F$ such that: for each GCN $\mathcal{M}$ and inequality-free rules $A_i \to B_i$ sound for $(\mathcal{M}, f)$ for each B_i in B, the rule $\bigwedge_i A_i \to H$ is sound for $(\mathcal{M}, f)$.*

Note that the above $f \in F$ is the same that witnesses the capturing of the rule in the definition of capturing exactly. For example, let F be a scoring function family that exactly captures intersection. Then, there exists $f \in F$ such that for

[2]This requirement is not strictly necessary, but it simplifies the proof.

any GCN $\mathcal{M}$, if rules $A_1 \rightarrow P_1(x, y)$ and $A_2 \rightarrow P_2(x, y)$ are sound for $(\mathcal{M}, f)$, then so is $A_1 \wedge A_2 \rightarrow P_3(x, y)$.

As an application of the results in this section, Corollaries 16 and 17 allow one to use rules already known to be sound to derive new sound rules. This avoids both searching for the new rules and using Proposition 12 to verify their soundness.

5.3 Equivalent Programs

In this section, I provide a means to obtain finite equivalent Datalog programs from GCNs and scoring functions: I do this for arbitrary monotonic max-GCNs with monotonically increasing scoring functions, and for monotonic max-sum-GCNs with a particular subclass of monotonically increasing scoring functions. This provides an upper bound on the expressive power of such models. To define the rule space, I first provide the definition from Tena Cucala et al. [256, Definition 11] of a (p, o)-tree-like formula φ_x for variable x (with a small adaption to allow for tree-like formulas both with and without inequalities).

Definition 18. *For each root variable x:*

1. *$\top$ is tree-like for x;*
2. *for each unary predicate U, $U(x)$ is tree-like for x;*
3. *for all formulas φ_1, φ_2 that share no variables and are tree-like for x, $\varphi_1 \wedge \varphi_2$ is tree-like for x;*
4. *for each variable x, binary predicate R, and tree-like formulas $\varphi_1, ..., \varphi_n$ for distinct variables $y_1, ..., y_n$ where no φ_i contains x and no φ_i and φ_j with $i \neq j$ share a variable, the following formulas are tree-like for x:*

$$\bigwedge_{i=1}^{n} \Big(R(x, y_i) \wedge \varphi_i \Big) \tag{5.1}$$

$$\bigwedge_{i=1}^{n} \Big(R(x, y_i) \wedge \varphi_i \Big) \wedge \bigwedge_{1 \leq i \leq j \leq n}^{n} y_i \not\approx y_j \tag{5.2}$$

Let φ be a tree-like formula and let x be a variable in φ. The fan-out *of x in φ is the number of distinct variables y_i for which $R(x, y_i)$ is a conjunct of φ. The* depth *of x is the maximal n for which there exist variables $x_0, ..., x_n$ and predicates $R_1, ..., R_n$ such that $x_n = x$ and $R(x_{i-1}, x_i)$ is a conjunct of φ for each $1 \leq i \leq n$. The* depth *of φ is the maximum depth of a variable in φ.*

For p and o natural numbers, a tree-like formula φ is (p, o)-tree-like if, for each variable x in φ, the depth i of x is at most p and the fan-out of x is at most $o(p-i)$.

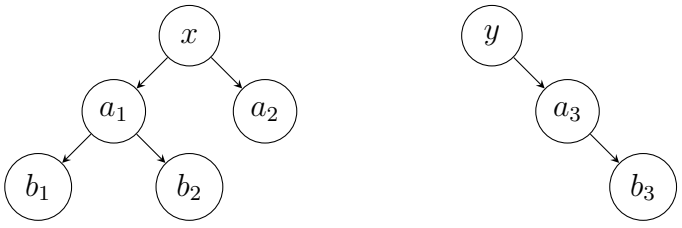
Using this, we define a *tree-like* rule as follows. Note that this differs from the definition of a tree-like rule given in Tena Cucala et al. [256, Definition 11], since here we have two formulas in the body that share no variables, and a binary predicate in the head of the rule.

Definition 19. *For $p, o \in \mathbb{N}$, a Datalog rule is* (p, o)-tree-like *if it is of the form $\varphi_x \wedge \varphi_y \rightarrow R(x, y)$, where φ_x, φ_y share no variables, φ_x is (p, o)-tree-like for x, and φ_y is (p, o)-tree-like for y.*

For example, the following rule is $(2, 2)$-tree-like:

$$R(x, a_1) \wedge R(x, a_2) \wedge a_1 \not\approx a_2 \wedge R(a_1, b_1) \wedge R(a_1, b_2) \wedge b_1 \not\approx b_2 \\ \wedge R(y, a_3) \wedge R(a_3, b_3) \rightarrow R(x, y).$$

A diagram representing the connections between the variables in the body is shown below, where it can be seen that the depth and the fan-out of x are both 2, and that the depth of y is 2 and the fan-out of y is 1:



For an example of a rule that is not tree-like, consider the following:

$$R(x, a_1) \wedge R(x, a_2) \wedge R(a_1, x) \wedge R(a_1, b_2) \wedge R(y, a_3) \wedge R(a_3, b_2) \wedge a_3 \not\approx b_2 \rightarrow R(x, y).$$

This is not tree-like because φ_x, φ_y share a variable b_2, because the inequality $a_3 \not\approx b_2$ relates two variables that are not permitted under the definition, and because the subtree of $R(x, a_1)$ contains an x (in the atom $R(a_1, x)$).

5.3.1 Monotonic Max-GCNs

I now provide a theorem describing a Datalog program equivalent to a given monotonic max-GCNs and monotonically increasing scoring function.

Theorem 20. *Let $\mathcal{M}$ be a monotonic max-GCN and f a monotonically increasing scoring function. Let $\mathcal{P}_\mathcal{M}$ be the Datalog program containing, up to variable renaming, each $(L, |Col| \cdot \delta_\mathcal{M})$-tree-like rule without inequalities that is sound for $(\mathcal{M}, f)$, where $\delta_\mathcal{M} = max(\delta_0, ..., \delta_L)$. Then $T_{\mathcal{M},f}$ and $\mathcal{P}_\mathcal{M}$ are equivalent.*

The full proof is given in Appendix B.2.7. Although the form of the rules extracted is similar to that for plain monotonic max-sum-GCNs, the proof of the theorem differs from that of Tena Cucala et al. [256, Theorem 13] due to (1) the inclusion of two variables x, y (with corresponding vertices and constants) in the base case and (2) the use of the monotonicity of the scoring function.

Proof sketch. We show that $T_{\mathcal{M},f}(D) = T_{P_\mathcal{M}}(D)$ holds for every dataset D.

$T_{P_\mathcal{M}}(D) \subseteq T_{\mathcal{M},f}(D)$ is trivial. So let α be an arbitrary binary fact in $T_{\mathcal{M},f}(D)$. Observe that α can be of the form $R(s, t)$, for distinct constants s, t, or of the form $R(s, s)$, for a constant s. In this sketch, we will consider only the $R(s, t)$ case – the other is similar. We construct a $(L, |\text{Col}| \cdot \delta_\mathcal{M})$-tree-like rule without inequalities r, such that $\alpha \in T_r(D)$ and r is sound for $(\mathcal{M}, f)$. Together, these will imply that $\alpha \in T_{P_\mathcal{M}}(D)$, as required for the proof.

Let $G = (V, E, \lambda)$ be the canonical encoding of D and let $\lambda_0, ..., \lambda_L$ the vertex labelling functions arising from the application of $\mathcal{M}$ to G. We inductively construct Γ, a conjunction of two formulas, one of which is tree-like for x and the other for y.

During the inductive construction, we also define a substitution ν from the variables of Γ to the set of constants in D, and a graph U (without vertex labels) where each vertex in U is of the form u^x for x a variable and each edge has a colour in Col. We assign to each vertex in U a *level* between 0 and L.

For the base case, we introduce fresh variables x, y, define $\nu(x) = s, \nu(y) = t$, introduce vertices u^x and u^y, and extend Γ with $U(x)$ for each $U(s) \in D$ (likewise for y for t). For the induction step, consider $1 \leq \ell \leq L$ and assume that all vertices of level greater than or equal to ℓ have been already defined. We then consider each vertex of the form u^x of level ℓ. Let $t = \nu(x)$. For each colour $c \in \text{Col}$, each layer $1 \leq \ell' < \ell$, and each dimension $j \in \{1, ..., \delta_{\ell'-1}\}$ for which there exists some $(v^t, w) \in E^c$, let w be a maximum c-coloured neighbour of v^t in feature vector index j at layer ℓ' (when applying $\mathcal{M}$ to G). Note that w must be of the form v^s for some constant s in D.

We then introduce a fresh variable y and define $\nu(y) = s$. We introduce a vertex u^y of level $\ell - 1$ and an edge $E^c(u^x, u^y)$ to U. Finally, we append to Γ the conjunction $E^c(x, y) \ \wedge \ \bigwedge_{U(s) \in D} U(y)$, where U stands in for arbitrary unary predicates. This completes the inductive construction. We then let $H = R(x, y)$, and define our tree-like rule to be $\Gamma \to H$. The construction of ν means $D \models \Gamma\nu$. So we have $H\nu \in T_r(D)$, with $H\nu = \alpha$, so $\alpha \in T_r(D)$, as required.

We now show that r is sound for $(\mathcal{M}, f)$. Let D' be an arbitrary dataset and α' an arbitrary ground atom such that $\alpha' \in T_r(D')$. Then there exists a substitution ν' such that $D' \models \Gamma\nu'$. Let the graph $G = (V', E', \lambda')$ be the canonical encoding of D' and $\lambda'_0, ..., \lambda'_L$ the functions labelling the vertices of G' when $\mathcal{M}$ is applied to it.

The following statement is proved by induction, which relies on the monotonicity of $\mathcal{M}$: for each $0 \leq \ell \leq L$ and each vertex u^x of U whose level is at least ℓ, we have $\mathbf{v}_{\lambda_\ell}[i] \leq \mathbf{p}_{\lambda'_\ell}[i]$ for each $i \in \{1, ..., \delta_\ell\}$, where $v = v^{\nu(x)}$ and $p = v^{\nu'(x)}$.

Note α' has the form $R(s', t')$ with $s' = \nu'(x)$ and $t' = \nu'(y)$. Now let $v = v^s, w = v^t, p = v^{s'}, q = v^{t'}$. Then $R(s, t) \in T_{\mathcal{M},f}(D)$ implies $f(R, \mathbf{v}_{\lambda_L}, \mathbf{w}_{\lambda_L}) \geq t_f$. The above property ensures that $\mathbf{v}_{\lambda_L}[i] \leq \mathbf{p}_{\lambda'_L}[i]$ and $\mathbf{w}_{\lambda_L}[i] \leq \mathbf{q}_{\lambda'_L}[i]$ for all $i \in \{1, ..., \delta_L\}$.

Thus, since f is monotonically increasing, we have $f(R, \mathbf{v}_{\lambda_L}, \mathbf{w}_{\lambda_L}) \leq f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L})$. So $f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L}) \geq t_f$, which implies that $R(s', t') \in T_{\mathcal{M},f}(D')$. □

The program $\mathcal{P}_{\mathcal{M}}$ is computable in principle by enumerating all rules in the finite class of tree-like rules mentioned in the theorem, and then filtering out rules that do not pass the soundness check in Proposition 12. This brute-force procedure can be optimized in practice: for example, by excluding all rules that subsume a known sound rule, since they are redundant; or by exploiting results about the expressiveness of scoring functions to avoid unnecessary soundness checks, as discussed at the end of Section 5.2. However, the size of the finite class described in Theorem 20 is exponential in terms of L (the number of GCN layers), |Col| (the number of binary predicates in the signature), and $\delta_{\mathcal{M}}$ (the maximum number of hidden features in a layer of the GCN). Thus, in practice, it is unlikely that the entire equivalent program can be recovered.

Note that as a consequence of this theorem, rules conforming to common rule patterns (e.g. symmetry, inversion, hierarchy, composition, intersection, triangle, diamond [162], fork, and cup) can only be sound if they are subsumed by a sound tree-like rule. For example, if a composition rule of the form $R_1(x, z) \wedge R_2(z, y) \rightarrow R_3(x, y)$ is sound for a monotonic max-GCN and a monotonic scoring function, there must exist another sound tree-like rule that subsumes it, such as $R_1(x, z_1) \wedge R_2(z_2, y) \rightarrow R_3(x, y)$.

5.3.2 Monotonic Max-Sum-GCNs

When seeking to obtain equivalent programs for the larger class of monotonic max-sum-GCNs instead of max-GCNs, we can no longer consider all monotonically increasing scoring functions. This is because there are a potentially unbounded number of neighbours that contribute to the computation during the execution of the max-sum-GCN, so if we tried to follow the same approach as for monotonic max-GCNs, we would have to consider an infinite number of tree-like rules. To address this, I prove that an argument of the same type as that of Tena Cucala et al. [256] for max-sum-GCNs can also be made when a scoring function is considered.

I show that the number of neighbours considered in each max-sum aggregation step can be restricted to a finite number without changing the output of the GCN and scoring function on any dataset.

This requires, however, an additional restriction on the scoring function: namely, it must be *non-negative bilinear*. This property ensures that there is a scalar value α such that once a vertex feature output by a GCN is greater than it, increasing any input to the scoring function will not result in a change of the output of $T_{\mathcal{M},f}$. Note, however, that the restriction is only a sufficient one, and there is an open question as to how much it can be relaxed whilst still obtaining equivalent programs.

Definition 21. *A* bilinear scoring function f *is one such that for all relations* R *and vectors* $\mathbf{h}, \mathbf{t} \in \mathbb{R}^{d_f}$*, we have* $f(R, \mathbf{h}, \mathbf{t}) = \mathbf{h}^\top \mathbf{M}_R \mathbf{t}$*, where* $\mathbf{M}_R \in \mathbb{R}^{d_f \times d_f}$ *is a matrix conditioned on* R*. We say that* f *is* non-negative *if each* $\mathbf{M}_R$ *contains only non-negative entries.*

Notice that RESCAL, DistMult, and ANALOGY are bilinear scoring functions. Also, TuckER can be rewritten as one, as shown in Appendix B.1.2. In addition to restricting the scoring function, we also add the restriction that the activation functions $\{\sigma_\ell\}_{1 \leq \ell \leq L}$ of $\mathcal{M}$ must be unbounded.

The essence of my approach is to compute, for each layer ℓ of the GCN, a non-negative integer $\mathcal{C}_\ell$ (called a *capacity* by Tena Cucala et al. [256]) such that replacing the number k_ℓ in the aggregation function by $\mathcal{C}_\ell$ does not change the output of the model on any dataset. I compute the capacities $\mathcal{C}_\ell$ using a more complex variant of the algorithm of Tena Cucala et al. [256, Algorithm 1], where the role of their GCN classification threshold is replaced by α, defined as follows. In the following, the set $\mathcal{X}_{\ell,i}$ consists of all real numbers that can occur in label vector index i at layer ℓ, when a max-sum-GCN $\mathcal{M}$ is applied to a dataset; these sets can be computed as shown in Tena Cucala et al. [256, Definition 7].

Definition 22. *Consider a monotonic max-sum-GCN* $\mathcal{M}$ *and a non-negative bilinear scoring function* f*. For each binary relation* R *in the signature, where* $\mathbf{M}_R$ *is its relation matrix in* f*, we define:* $\alpha_R := 1$ *if all elements of* $\mathbf{M}_R$ *are* 0 *or* $\bigcup_i \mathcal{X}_{L,i} = \{0\}$.

Otherwise, we define α_R *to be the least natural number such that* $\alpha_R \cdot w \cdot \epsilon \geq t_f$*, where* w *is the least non-zero element of* $\mathbf{M}_R$ *and* ϵ *is the least non-zero element of* $\bigcup_i \mathcal{X}_{L,i}$*. Then, we define* $\alpha := max\{\alpha_R \mid$ *for each binary relation* R *in the signature*$\}$*.*

The following theorem then establishes that replacing the aggregation number k_ℓ by the capacity $\mathcal{C}_\ell$ leaves the output of the GCN and scoring function unchanged *on any dataset.*

Theorem 23. *Let* $\mathcal{M}$ *be a monotonic max-sum-GCN with unbounded activation functions and* f *a non-negative bilinear scoring function. Let* $\mathcal{M}'$ *be the GCN obtained from* $\mathcal{M}$ *by replacing* k_ℓ *with the capacity* $\mathcal{C}_\ell$ *for each* $\ell \in \{1, ..., L\}$*. Then for each dataset* D*, it holds that* $T_{\mathcal{M},f}(D) = T_{\mathcal{M}',f}(D)$*.*

The proof is given in Appendix B.2.9. Whilst the proofs for other theorems in this section use similar techniques to Tena Cucala et al. [256], the proof of this theorem is novel and depends on the structure of non-negative bilinear scoring functions, as well as using several results proved by Tena Cucala et al. [256]. It considers three different cases in relation to non-zero entries of the scoring function relation matrix $\mathbf{M}_R$ and GCN output comparisons to α, where the final case itself considers five subcases.

Using the above theorem, I prove the main result of this section, which is analogous to Theorem 20. The following theorem shows that an equivalent program can be constructed for any monotonic max-sum-GCN $\mathcal{M}$ and non-negative bilinear scoring function f using rules taken from the finite space of all $(L, |\text{Col}| \cdot \delta_\mathcal{M} \cdot \mathcal{C}_{\mathcal{M},f})$-tree-like rules, where $\mathcal{C}_{\mathcal{M},f} = \max\{\mathcal{C}_1, ..., \mathcal{C}_L\}$. The proof is given in Appendix B.2.10.

Theorem 24. *Let* $\mathcal{M}$ *be a monotonic max-sum-GCN with unbounded activation functions and* f *a non-negative bilinear scoring function. Let* $\mathcal{P}_\mathcal{M}$ *be the Datalog program containing, up to variable renaming, each* $(L, |Col| \cdot \delta_\mathcal{M} \cdot \mathcal{C}_{\mathcal{M},f})$*-tree-like rule that is sound for* $(\mathcal{M}, f)$*, where* $\delta_\mathcal{M} = max(\delta_0, ..., \delta_L)$*. Then* $T_{\mathcal{M},f}$ *and* $\mathcal{P}_\mathcal{M}$ *are equivalent.*

As before, the size of the finite class of tree-like rules is exponential in terms of L (the number of GCN layers), |Col| (the number of binary predicates in the

signature), and $\delta_{\mathcal{M}}$ (the maximum number of hidden features in a layer of the GCN). However, it is also exponential in the capacity $\mathcal{C}_{\mathcal{M},f}$, which may be very large in practice. Thus, recovering the equivalent program in practice is not feasible. Nonetheless, one can still find sound tree-like rules as explanations, which I will explore in my experiments in the next section.

5.4 Experiments

In this section, I describe the experiments run in this chapter. The experiments test max-sum-GCN and scoring function performance when restricted to being monotonic, as well as extracting sound rules from the monotonic models. First, I describe the setup of my experiments. Next, I outline how I extract sound rules. Finally, I present and analyse my results, finding that, in general, monotonic models are comparable to or better than their standard counterparts. I also find that the models have many sound rules, and often recover the monotonic rule patterns that are injected into some of the datasets.

5.4.1 Experiment Setup

For the model architecture, I fix a hidden dimension of 50, 2 layers, and ReLU activation functions. I use max-GCNs and sum-GCNs, as well as experimenting with 4 different varieties of scoring function families: RESCAL, DistMult, TuckER, and NAM. I train each model for 8000 epochs, except for ones that use TuckER, which I train for 4000, given my computational constraints and how much slower the TuckER models are. For all trained models, I compute standard classification metrics, as defined in Section 2.5.2. To select the scoring function threshold, I evaluate the model on the validation set across candidate thresholds and select the one which maximizes accuracy. The candidate thresholds are the set of all scores produced on the validation input set.

Each training epoch, for each positive target fact, 10 negative facts are generated by randomly corrupting the binary predicate. These facts are filtered to ensure they do not contain any false negatives (i.e. facts that appear in the training

Dataset	Decoder	Model	%Acc	%Prec	%Rec	%F1	AUPRC	Loss	%SO	#1B	#2B
WN-hier	DistMult	Standard	89.87	89.1	90.86	89.97	0.9423	0.11	-	-	-
		Monotonic	85.05	79.18	95.52	86.43	0.809	1.41	52	171	26532
	RESCAL	Standard	92.69	**90.79**	95.02	92.86	**0.9597**	0.09	-	-	-
		Monotonic	88.59	85.3	93.26	89.1	0.863	1.23	**60**	104	17212
	NAM	Standard	92.1	90.22	94.46	92.28	0.9481	0.09	-	-	-
		Monotonic	67.87	68.33	65	64.77	0.6577	1.64	48	141	19244
	TuckER	Standard	**93**	90.51	**96.08**	**93.21**	0.9542	0.09	-	-	-
		Monotonic	87.5	82.91	94.88	88.39	0.8206	1.36	56	89	14498
WN-sym	DistMult	Standard	**96.6**	**95.79**	97.5	**96.63**	**0.9929**	0.09	-	-	-
		Monotonic	92.34	86.92	**99.68**	92.86	0.9561	1.32	88	81	12845
	RESCAL	Standard	96.1	94.71	97.66	96.16	0.9889	0.07	-	-	-
		Monotonic	94.27	89.72	100	94.58	0.9355	1.12	**92**	42	7537
	NAM	Standard	95.72	94.58	97.02	95.78	0.9897	0.06	-	-	-
		Monotonic	72.16	76.71	67.3	68.82	0.6775	1.79	40	88	12457
	TuckER	Standard	96.24	95.03	97.58	96.29	0.9888	0.07	-	-	-
		Monotonic	91.31	85.97	98.84	91.95	0.8892	1.27	80	81	12825
WN-cup_nmhier	DistMult	Standard	74.98	78.44	68.88	73.35	0.8139	0.17	-	-	-
		Monotonic	69.69	71.21	66.45	68.61	0.6953	1.77	40	143	21366
	RESCAL	Standard	**77.98**	77.38	79.1	**78.22**	0.8152	0.15	-	-	-
		Monotonic	73.21	75.91	67.99	71.72	0.7058	1.61	40	31	5006
	NAM	Standard	76.72	74.56	**81.34**	77.72	**0.833**	0.14	-	-	-
		Monotonic	59.74	59.04	63.1	57.47	0.5697	2.41	**45**	354	50014
	TuckER	Standard	77.46	77.61	77.17	77.38	0.8144	0.15	-	-	-
		Monotonic	71.51	**80.15**	57.27	66.78	0.664	1.68	40	22	3772
fb237v1	DistMult	Standard	55.55	53.08	97.1	68.62	0.9611	0.02	-	-	-
		Monotonic	81.85	79.48	86	82.59	0.6884	0.80	-	35366	-
	RESCAL	Standard	58.8	55.38	96.5	70.23	**0.9625**	0.01	-	-	-
		Monotonic	**91**	**96.72**	84.9	**90.41**	0.6114	0.70	-	4797	-
	NAM	Standard	60.5	56.44	96.8	71.17	0.9552	0.01	-	-	-
		Monotonic	64	58.66	97	73.03	0.9557	0.07	-	155348	-
	TuckER	Standard	54.35	52.31	**98.9**	68.42	0.9597	0.01	-	-	-
		Monotonic	90.5	92.07	88.7	90.34	0.6305	-	-	8511	-
WN18RRv1	DistMult	Standard	89.76	83.13	**100**	90.75	0.9337	0.02	-	-	-
		Monotonic	91.52	85.49	**100**	92.18	0.7922	0.79	-	97	11995
	RESCAL	Standard	91.52	85.61	99.88	92.19	0.9306	0.01	-	-	-
		Monotonic	**95.82**	**92.38**	99.88	**95.98**	0.7878	0.77	-	52	6488
	NAM	Standard	78.61	70.96	**100**	82.75	0.9287	0.01	-	-	-
		Monotonic	74.73	66.51	99.88	79.83	0.9251	0.10	-	202	20874
	TuckER	Standard	91.76	85.94	**100**	92.41	**0.9402**	0.01	-	-	-
		Monotonic	92.73	88.3	98.91	93.23	0.7787	0.81	-	65	8134
nellv1	DistMult	Standard	57.65	54.51	92.71	68.56	0.9271	0.09	-	-	-
		Monotonic	65.53	59.24	**100**	74.39	0.9081	1.35	-	631	125094
	RESCAL	Standard	58.94	55.01	99.06	70.72	0.9076	0.04	-	-	-
		Monotonic	70.59	**72.23**	66.59	69.05	0.5372	1.05	-	144	30154
	NAM	Standard	65.53	61.05	86.82	71.15	0.9127	0.03	-	-	-
		Monotonic	53.24	51.69	**100**	68.15	**0.929**	0.44	-	970	176836
	TuckER	Standard	55.53	53.04	99.06	69.06	0.9103	0.04	-	-	-
		Monotonic	**76.82**	68.57	99.29	**81.1**	0.6893	1.21	-	265	53418

Table 5.1: Results for max-GCNs. Loss is from the final epoch on the training set. AUPRC is from the validation set. Other metrics are computed on the test set. %SO is the percentage of LogInfer rules that are sound for the model. #1B and #2B are the number of sound rules with one and two body atoms respectively.

Decoder	Model	%Acc	AUPRC	Loss	%SO	#1B	#2B
DistMult	Standard	76.49	0.9262	0.08	-	-	-
	Monotonic	79.79	0.8046	1.26	45	6581	42225
RESCAL	Standard	78.9	0.93	0.06	-	-	-
	Monotonic	**85.23**	0.7461	1.12	61	1045	19527
NAM	Standard	75.2	0.9243	**0.06**	-	-	-
	Monotonic	66.39	0.8104	1.08	**68**	26758	59898
TuckER	Standard	79.12	**0.9306**	0.06	-	-	-
	Monotonic	82.64	0.7385	1.23	61	1599	25003

Table 5.2: Results for max and sum-GCNs, aggregated across all datasets. Loss is from the final epoch on the training set. AUPRC is from the validation set. Accuracy is computed on the test set. %SO is the percentage of LogInfer rules that are sound for the model. #1B and #2B are the number of sound rules with one and two body atoms respectively.

set). I originally corrupted the constants in the positive fact, but found that predicate corruption leads to significantly better performance by the baseline models on standard benchmarks [258]. Models are trained using binary cross entropy with logits (BCE) loss and the Adam optimizer with a standard learning rate of 0.001 and weight decay of $5e^{-4}$.

I train models without restrictions as baselines (denoted by "Standard"), as well as monotonic models (denoted by "Monotonic") as described in Section 2.5.1. When clamping weights, I multiply by 50 the term in the BCE loss function corresponding to the positive examples: without this, I found there to be insufficient positive signal for the training, leading to consistently positive gradients and thus weights that only tended to 0 as training progressed; the value of 50 was arrived at by hyperparameter tuning on the standard benchmark datasets collectively, and then fixed for all experiments. I run each experiment across five different random seeds.

Datasets I use three standard benchmarks, WN18RRv1, fb237v1, and nellv1 [258], as well as the LogInfer [162] datasets of WN-hier, WN-sym, and WN-cup_nmhier, as described in Section 2.5.3. Importantly, these benchmarks are inductive, meaning that the validation and testing sets contain constants not seen during training, so approaches where embeddings are learned for each constant do not work for

them (such as standard scoring function approaches). Given that the datasets have no unary predicates, I introduce a dummy unary predicate that holds for every constant in the dataset, yielding the same initial embedding for every node in the encoding. More details on this and other approaches to initial node features in the absence of unary predicates can be found in Appendix B.1.4.

Rule Extraction On all datasets, I iterate over each Datalog rule in the signature with up to two body atoms and a binary head predicate, and count the number of sound rules, using Proposition 12 to check soundness. On the fb237v1 dataset, I only check rules with one body atom, since the large number of predicates means that searching the space of rules with two body atoms intractable. I also check if the injected rules in the LogInfer datasets are sound.

5.4.2 Results

The experimental results for max-GCNs are provided in Table 5.1. For each dataset and metric, I highlight in bold the best value achieved by a model variant for that metric. In Table 5.2, I provide results for max and sum-GCNs, aggregated across all datasets. My findings show that restricting GCNs and scoring functions to be monotonic does not adversely affect model performance in most cases (except for models that use NAM). In fact, in some scenarios, the restrictions cause a significant increase in performance. Similar conclusions can be drawn from my results for sum-GCNs as for max-GCNs; for completeness, full results for monotonic sum-GCNs are given in Table B.1, of Appendix B.3.

Some general patterns emerge. First, loss is always lower for the standard model than its monotonic counterpart. This is due to the restriction on model weights preventing gradient descent from optimizing the parameters to the same extent. On the other hand, the monotonicity may help to prevent overfitting on the training set. Likewise, AUPRC is consistently higher on the standard model than the monotonic version. Used as an indication for model performance on the validation set, this suggests that the standard models may be overfitting on the

validation facts and struggling to generalize to the test set, in comparison to the monotonic models. Finally, we see that models using NAM as a scoring function consistently struggled when restricted to being monotonic: other scoring functions are thus more suitable when monotonicity is desired.

I obtained a number of sound rules for every monotonic model (columns #1B and #2B in Table 5.1), showing the efficacy of my rule-checking methodology. On the LogInfer-WN datasets, there are 605 possible rules with one body atom and 90508 with two. On fb237v1, there are 162000 possible rules with one body atom. On WN18RRv1, there are 405 possible rules with one body atom and 49572 with two. On nellv1, there are 980 possible rules with one body atom and 186592 with two. Notice that, for example, nearly all possible rules are sound when using monotonic NAM on nellv1, which corresponds to the perfect model recall and poor precision. I show some randomly sampled sound rules in Table B.2 and Table B.3 of Appendix B.3.

On the LogInfer datasets, I find consistently small drops in performance ($\approx 5\%$ less accuracy) when monotonic models are used. However, the restriction enables one to check sound rules for the model: we find many sound rules with one or two body atoms, and also that around half (or sometimes more) of the injected LogInfer rules are sound for the model. On WN-cup_nmhier, the monotonic models showed their ability to still learn and recover some of the injected monotonic rules, even in the presence of some injected rules that are explicitly non-monotonic.

On fb237v1, we see significantly better performance by monotonic models over their standard counterparts. The same is seen on nellv1. On WN18RRv1, the monotonic models again outperform the standard ones, but by smaller margins. These results are somewhat surprising, but encouraging, since one would expect monotonic models to have a greater advantage on the LogInfer datasets, where the underlying patterns in the data are explicitly monotonic. Finally, I note that the monotonic model performance on fb237v1 and WN18RRv1 is significantly better than those of MGNNs [254], highlighting the benefits of using a scoring function as a decoder instead of their dataset encoding-decoding scheme, since the recall of

their models is upper-bounded by the number of positive test pairs of constants that also appear in the input dataset.

5.5 Discussion

In this chapter, I showed how scoring functions can be made monotonically increasing, how sound Datalog rules can be found for monotonic GCNs with monotonic scoring functions, and showed how existing results about scoring functions capturing rule patterns impact the expressivity of GCN models that use those scoring functions. I also provided ways to obtain an equivalent program for any monotonic GCN with max aggregation and monotonically increasing scoring function, and any monotonic GCN with max-sum aggregation and a scoring function that is non-negative bilinear. My experiments showed that, in practice, the performance of GCNs with scoring functions does not drop substantially when applying the restrictions that make them monotonic, and in some cases increase the model performance. I also showed that, in practice, many sound rules can be recovered from these monotonic models, which can be used to explain their predictions.

Limitations and Future Work A limitation of this work is that I only consider non-negative bilinear scoring functions when obtaining equivalent programs for max-sum-GCNs: the approach I describe may also work for other classes of monotonically increasing scoring functions. For future work, there may be more advanced training paradigms that would work better for the monotonic models considered in this chapter.

6

Relational Graph Convolutional Networks Do Not Learn Sound Rules

Contents

6.1 Introduction 101
6.2 Partitioning the Channels of a GCN 104
6.2.1 Safe Channels 104
6.2.2 Stable and Increasing Channels 108
6.2.3 Unbounded Channels 111
6.3 Experiments 115
6.3.1 Experiment Setup 116
6.3.2 Training Paradigm Variations 117
6.3.3 Results 118
6.4 Discussion 121

6.1 Introduction

In Chapter 5, I considered monotonic GCNs with monotonic scoring functions. Whilst the restriction of having only non-negative weights allows one to recover sound rules for explanations, it is not used in practice for standard and popular GCNs, such as R-GCN [234]. Furthermore, it may be too restrictive in cases where the function required to solve the task is not monotonic.

In this chapter, I consider sum-GCNs, which do not impose restrictions on model

weights. These GCNs can be seen both as an extension to the sum-GCNs in Tena Cucala et al. [256] but without the monotonicity requirement, as well as a variant of R-GCN. The functions learnt by sum-GCNs may be non-monotonic: predicted facts may be invalidated by adding new facts to the input KG. As a result, these GCNs cannot, in general, be faithfully captured by (negation-free) Datalog programs. In fact, they cannot, in general, be captured by FOL programs [19]. However, some subset of their predictions may still be explainable by sound Datalog rules. Thus, prior to the work of this chapter, there was an open question as to whether sound rules can be recovered from sum-GCN models with negative weights.

My Contributions The contributions in this chapter are as follows. In Section 6.2, I identify a subset of the output channels (i.e. features) of the sum-GCN which exhibit monotonic behaviour, and for which sound Datalog rules may be extracted. The ability to extract sound rules is important as it allows one to explain model predictions associated to the identified output channels, despite the non-monotonicity of the model as a whole. Furthermore, I provide means for identifying *unbounded* output channels for which no sound Datalog rule exists, hence suggesting that these channels inherently exhibit non-monotonic behaviour.

As in Chapter 5, I conduct experiments in Section 6.3 on benchmark datasets and also LogInfer. My experiments show that even under ideal scenarios, without restrictions on the training process, all the channels in the trained sum-GCN are unbounded (even for monotonic LogInfer benchmarks), which implies that there are no sound Datalog rules for the model. This stands in stark contrast to the monotonic GCNs from Chapter 5, whose entire operation could be characterized by sound Datalog rules: thus, this chapter serves as a negative counterpart to Chapter 5, as it shows how sound rule extraction collapses when the restriction of non-negative weights is relaxed. To address this, I consider an adjustment to the training process, where weights sufficiently close to zero (as specified by a given threshold) are clamped to zero iteratively during training. As the weight clamping threshold increases, I observe that an increasing number of output channels exhibit monotonic

behaviour, and I obtain more sound rules, although the model accuracy diminishes. Hence, there is a trade-off between model performance and rule extraction.

This chapter is based on a paper published in KR 2024 [193].

Background There are four important distinctions in the theory of this chapter. First, in comparison to Section 2.1.1, we only consider safe Datalog rules: every variable that appears in a rule must occur in a body atom. Next, for convenience, when considering a GCN $\mathcal{M}$, for layer $\ell \in \{0, ..., L\}$ of $\mathcal{M}$, we refer to each $i \in \{1, ..., \delta_\ell\}$ as a *channel*. In this chapter, we consider sum-GCNs in particular, as described in Section 2.3.2. We apply the further restriction that the GCN activation functions must be monotonically increasing with non-negative ranges. I.e. we require that for all $\ell \in \{1, ..., L\}$, σ_ℓ is monotonically increasing with non-negative range (for example, ReLU or Sigmoid).

Finally, note that the R-GCN model [234] can be seen as a sum-GCN variant with ReLU activations and zero biases in all layers. The definition in Schlichtkrull et al. [234] includes a normalization parameter $c_{i,r}$ (cf. their Equation (2)), which can depend on a predicate r and/or vertex i under special consideration when applying the GNN. A dependency on the vertex implies that the values of these parameters are data dependent—that is, they are computed at test time based on the concrete graph over which the GNN is evaluated (e.g., $c_{i,r}$ could be computed as the number of r-neighbours of vertex i). The R-GCN model is a sum-GCN under the assumption that the R-GCN normalization parameters are data independent and hence can be fixed after training; we further assume for simplicity that they are set to 1.

Other concepts that will be particularly relevant for this chapter include sound rules for GCN-based dataset transformations (Section 2.4.2) and the non-canonical encoding and decoding from Tena Cucala et al. [256] that we use for link prediction (Section 2.4.3).

6.2 Partitioning the Channels of a GCN

In this section, I first provide two approaches for identifying channels in a sum-GCN that exhibit monotonic behaviour, which will later allow one to extract sound Datalog rules with head predicates corresponding to these channels. Candidate Datalog rules can be effectively checked for soundness using the approach developed by Tena Cucala et al. [256] for monotonic GCNs. Furthermore, I provide an approach for identifying channels for which no sound Datalog rule exists. My techniques are data independent and rely on direct analysis of the dependencies between feature vector components via the parameters of the model.

6.2.1 Safe Channels

In this section, I introduce the notion of a safe channel of a sum-GCN $\mathcal{M}$. Intuitively, a channel is *safe* if, for any dataset D, the computation of its value for each vertex $v \in \texttt{enc}(D)$ through the application of $\mathcal{M}$ to D is affected only by non-negative values in the weight matrices of the GCN.

For instance, consider a simple sum-GCN where matrix $\mathbf{A}_\ell$ for layer ℓ is given below

$$\mathbf{A}_\ell = \begin{pmatrix} 1 & 0 & 1 & 4 \\ 1 & 0 & 0 & 2 \\ -8 & -1 & 0 & -2 \end{pmatrix}$$

and each of the matrices $\mathbf{B}_\ell^c$ for $c \in \text{Col}$ contain only zeroes. Furthermore, assume that layer $\ell - 1$ has four channels, where the *third* one has been identified as unsafe and all other channels have been identified as safe. Then, for an arbitrary vertex v, the product of $\mathbf{A}_\ell$ and $\mathbf{v}_{\ell-1}$ in the computation of $\mathbf{v}_\ell$ reveals that the second channel in ℓ is safe, because the unsafe component in $\mathbf{v}_{\ell-1}$ is multiplied by zero ($\mathbf{A}_\ell[2,3]$), and the safe components are multiplied by non-negative matrix values. In contrast, the first channel of ℓ is unsafe since its computation involves the product of an unsafe component of $\mathbf{v}_{\ell-1}$ with a non-zero matrix component ($\mathbf{A}_\ell[1,3]$), and the third channel is unsafe due to the product of a component of $\mathbf{v}_{\ell-1}$ with a negative matrix value (e.g. $\mathbf{A}_\ell[3,1]$). The following definition generalizes this intuition, and is inductive over the layers of the sum-GCN.

Definition 25. *Let $\mathcal{M}$ be a sum-GCN as in Section 6.1. All channels $i \in \{1, ..., \delta_0\}$ are* safe *at layer $\ell = 0$. Channel $i \in \{1, \ldots, \delta_\ell\}$ is* safe *at layer $\ell \in \{1, \ldots, L\}$ if*

1. *the i-th row of each matrix $\mathbf{A}_\ell$ and $\{(\mathbf{B}^c_\ell)\}_{c \in Col}$ contains only non-negative values; and*

2. *the j-th element in each such row is zero for each $j \in \{1, \ldots, \delta_{\ell-1}\}$ such that j is unsafe in layer $\ell - 1$.*

Otherwise, i is unsafe.

We can now show that safe channels in a GCN exhibit monotonic behaviour. In particular, the value of a safe channel may only increase (or stay the same) when new facts are added to the input dataset. Note that this monotonicity is monotonicity under dataset inclusion (see Section 2.4.4), rather than the monotonicity under injective homomorphisms considered in Chapter 5. However, it is sufficient for extracting sound rules.

Lemma 26. *Let $\mathcal{M}$ be a sum-GCN as in Section 6.1, let D', D be datasets satisfying $D' \subseteq D$, let $v \in \mathtt{enc}(D')$, and let $\mathbf{v}_\ell$ and $\mathbf{v}'_\ell$ be the vectors associated to v in layer ℓ upon applying $\mathcal{M}$ to D and D' respectively. If channel i in $\mathcal{M}$ is safe at layer ℓ, then $\mathbf{v}'_\ell[i] \leq \mathbf{v}_\ell[i]$.*

Proof. We proceed by induction on ℓ. The base case $\ell = 0$ holds trivially by the definition of the canonical encoding and the fact that $D' \subseteq D$. For the inductive step, assume that the claim holds for $\ell - 1 \geq 0$ and that channel i is safe at layer ℓ. We show that $\mathbf{v}'_\ell[i] \leq \mathbf{v}_\ell[i]$. Consider the computation of $\mathbf{v}_\ell[i]$ and $\mathbf{v}'_\ell[i]$ as per Equation (2.3). The value $(\mathbf{A}_\ell \mathbf{v}_{\ell-1})[i]$ is the sum over all $j \in \{1, ..., \delta_{\ell-1}\}$ of $\mathbf{A}_\ell[i, j]\mathbf{v}_{\ell-1}[j]$. By Definition 25, this sum involves only non-negative values and can be restricted to values of j corresponding to safe channels at layer $\ell-1$. By induction, each such safe j satisfies $\mathbf{v}'_{\ell-1}[j] \leq \mathbf{v}_{\ell-1}[j]$ and hence $\mathbf{A}_\ell[i, j]\mathbf{v}'_{\ell-1}[j] \leq \mathbf{A}_\ell[i, j]\mathbf{v}_{\ell-1}[j]$; thus, $(\mathbf{A}_\ell \mathbf{v}'_{\ell-1})[i] \leq (\mathbf{A}_\ell \mathbf{v}_{\ell-1})[i]$.

Consider now $c \in$ Col and let E^c and $(E^c)'$ be the c-coloured edges in $\mathtt{enc}(D)$ and $\mathtt{enc}(D')$ respectively. The sums involved in the products $(\mathbf{B}^c_\ell \sum_{(v,u) \in E^c} \mathbf{u}_{\ell-1})[i]$

and $(\mathbf{B}_\ell^c \sum_{(v,u)\in(E^c)'} \mathbf{u}'_{\ell-1})[i]$ contain only non-negative values and can similarly be restricted to safe channels at layer $\ell - 1$. By induction, each such safe j satisfies that $\forall u \in \mathtt{enc}(D'),\ \mathbf{u}'_{\ell-1}[j] \leq \mathbf{u}_{\ell-1}[j]$. Furthermore, $(E^c)' \subseteq E^c$ given that $D' \subseteq D$. We conclude that

$$(\sum_{c\in\mathrm{Col}} \mathbf{B}_\ell^c \sum_{(v,u)\in(E^c)'} \mathbf{u}'_{\ell-1})[i] \leq (\sum_{c\in\mathrm{Col}} \mathbf{B}_\ell^c \sum_{(v,u)\in E^c} \mathbf{u}_{\ell-1})[i].$$

By combining the previous inequalities and taking into account that σ_ℓ is monotonically increasing, we conclude that $\mathbf{v}'_\ell[i] \leq \mathbf{v}_\ell[i]$, as required. □

I conclude this section by showing that the identification of safe channels allows for the extraction of sound rules from a trained sum-GCN model. In particular, given a candidate Datalog rule, it is possible to algorithmically verify its soundness using the approach of Tena Cucala et al. [256]. The soundness check for a safe rule r of the form (2.1) involves considering an arbitrary (but fixed) set containing as many constants as variables in r, and considering each substitution ν mapping variables in r to these constants and satisfying the inequalities in r. For D_r^ν, the dataset consisting of each fact $B_i\nu$ such that B_i is a body atom in r, we check whether the GCN predicts the fact $H\nu$, corresponding to the grounding of the head atom. If this holds for each considered ν, then the rule is sound; otherwise, the substitution ν for which it does not hold provides a counter-example to soundness.

The proof of the following proposition differs from that of Tena Cucala et al. [256, Lemma 12] and Proposition 12 from Chapter 5 in the use of channel safety to establish local monotonicity.

Proposition 27. *Let $\mathcal{M}$ be a sum-GCN as in Section 6.1, and let r be a safe rule of the form* (2.1) *where H mentions a unary predicate U_p. Let S be an arbitrary set of as many constants as there are variables in r. Assume channel p in $\mathcal{M}$ is safe at layer L. Then r is sound for $T_\mathcal{M}$ if and only if $H\nu \in T_\mathcal{M}(D_r^\nu)$ for each substitution ν mapping the variables of r to constants in S and such that $D_r^\nu \models B_i\nu$ for each inequality B_i in r.*

Proof. To prove the soundness of r, consider an arbitrary dataset D. We show that $T_r(D) \subseteq T_{\mathcal{M}}(D)$. To this end, we consider an arbitrary fact in $T_r(D)$ and show that it is also contained in $T_{\mathcal{M}}(D)$. By the definition of the immediate consequence operator T_r, this fact is of the form $H\mu$, where μ is a substitution from the variables of r to constants in D satisfying $D \models B_i\mu$ for each body literal B_i of r. Let σ be an arbitrary one-to-one mapping from the co-domain of μ to some subset of S; such a mapping exists because S has as least as many constants as variables in r. Let ν be the composition of μ and σ.

Observe that for each body inequality B_i of r, we have $D_r^\nu \models B_i\nu$ because $D \models B_i\mu$ and σ is injective. Therefore, by the assumption of the proposition, $H\nu \in T_{\mathcal{M}}(D_r^\nu)$. Now, observe that the result of applying $T_{\mathcal{M}}$ to a dataset does not depend on the identity of the constants, but only on the structure of the dataset; therefore, $T_{\mathcal{M}}$ is invariant under one-to-one mappings of constants, and since σ is one such map, $H\nu \in T_{\mathcal{M}}(D_r^\nu)$ implies $H\mu \in T_{\mathcal{M}}(D_r^\mu)$. Now, let a be the single constant in $H\mu$. Since $D_\mu^r \subseteq D$ by definition of μ, and channel p is safe at layer L, we can apply Lemma 26 to conclude that $\mathbf{v}'_L[p] \leq \mathbf{v}_L[p]$, for v the vertex corresponding to a in $\mathtt{enc}(D_r^\mu)$, and $\mathbf{v}'_L$ and $\mathbf{v}_L$ the feature vectors in layer L computed for v by $\mathcal{M}$ on D_r^μ and D, respectively. But $H\mu \in T_{\mathcal{M}}(D_r^\mu)$ implies that $\mathtt{cls}_t(\mathbf{v}'_L[p]) = 1$ and so $\mathbf{v}'_L[p] \geq t$. Hence, $\mathbf{v}_L[p] \geq t$, and so $\mathtt{cls}_t(\mathbf{v}_L[p]) = 1$, which implies that $H\mu \in T_{\mathcal{M}}(D)$, as we wanted to show.

If on the other hand, if $H\nu \notin T_{\mathcal{M}}(D_r^\nu)$ for some substitution ν defined as in the proposition, then $T_r(D_r^\nu) \not\subseteq T_{\mathcal{M}}(D_r^\nu)$, as $H\nu \in T_r(D_r^\nu)$. Thus, r is unsound for $T_{\mathcal{M}}$. □

The identification of safe channels provides a sufficient condition for monotonic behaviour that can be easily computed in practice and enables rule extraction. The fact that a channel is unsafe, however, does not imply that it behaves non-monotonically. In the following section, I provide a more involved analysis of the dependencies between channels and model parameters which yields a more fine-grained channel classification. In particular, I identify two new classes of

channels that behave monotonically, such that their union contains all safe channels, but may also contain unsafe channels.

6.2.2 Stable and Increasing Channels

In this section, I provide a classification of the channels of the GCN depending on their behaviour under updates involving the addition of new facts to an input dataset. Intuitively, *stable* channels are those whose value always remains unaffected by such updates; in turn, *increasing* channels cannot decrease in value, whereas *decreasing* channels cannot increase in value. All remaining channels are categorized as *undetermined*. Consider again a sum-GCN where matrix $\mathbf{A}_\ell$ for layer ℓ is given below and each of the matrices $\mathbf{B}_\ell^c$ for $c \in \text{Col}$ contain only zeroes.

$$\mathbf{A}_\ell = \begin{pmatrix} 1 & 0 & -1 & -2 \\ 2 & 1 & -3 & 0 \\ 1 & 0 & 1 & 2 \\ -3 & 0 & 2 & 0 \end{pmatrix}$$

Furthermore, assume again that layer $\ell-1$ has four channels, where the first has been identified as increasing, the second one as undetermined, the third one as decreasing, and the fourth one as stable. For an arbitrary vertex v, the product of $\mathbf{A}_\ell$ and $\mathbf{v}_{\ell-1}$ in the computation of $\mathbf{v}_\ell$ reveals that the first channel of ℓ is increasing since the undetermined component of $\mathbf{v}_{\ell-1}$ is multiplied by 0, the increasing component by a positive matrix value, and the decreasing component by a negative matrix value. The second channel is undetermined, since it involves the product of an undetermined component with a non-zero matrix value ($\mathbf{A}_\ell[1,1]$). The third channel is also undetermined, since it is a mixture of increasing and decreasing values: "positive times increasing" is increasing, whereas "positive times decreasing" is decreasing, so it cannot be known a-priori whether the sum of these values will increase or decrease. Finally, the fourth channel of ℓ is decreasing since the undetermined component of $\mathbf{v}_{\ell-1}$ is multiplied by 0, and increasing (resp. decreasing) components are multiplied by negative (resp. positive) matrix values.

The following definition formalizes these intuitions and extends the analysis to the matrices $\mathbf{B}_\ell^c$ involved in neighbourhood aggregation.

Definition 28. *Let $\mathcal{M}$ be a sum-GCN as in Equation* (6.1)*. All channels are* increasing *at layer* 0*, A channel $i \in \{1, \ldots, \delta_\ell\}$ is* stable *at layer $\ell \in \{1, ..., L\}$ if both of the following conditions hold for each $j \in \{1, \ldots, \delta_{\ell-1}\}$:*

- $\mathbf{B}^c_\ell[i, j] = 0$ *for each $c \in Col$, and*
- $\mathbf{A}_\ell[i, j] \neq 0$ *implies that j is stable in layer $\ell - 1$.*

It is increasing *(resp.* decreasing*) at layer ℓ if it is not stable and, for each $j \in \{1, \ldots, \delta_{\ell-1}\}$, these conditions hold:*

1. *if j is increasing at layer $\ell - 1$, then $\mathbf{A}_\ell[i, j] \geq 0$ (resp. $\mathbf{A}_\ell[i, j] \leq 0$);*
2. *if j is decreasing at layer $\ell - 1$, then $\mathbf{A}_\ell[i, j] \leq 0$ (resp. $\mathbf{A}_\ell[i, j] \geq 0$) and $\mathbf{B}^c_\ell[i, j] = 0$ for each $c \in Col$;*
3. *if j is undetermined at layer $\ell - 1$, then $\mathbf{A}_\ell[i, j] = 0$ and $\mathbf{B}^c_\ell[i, j] = 0$ for each $c \in Col$; and*
4. *$\mathbf{B}^c_\ell[i, j] \geq 0$ (resp. $\mathbf{B}^c_\ell[i, j] \leq 0$) for each $c \in Col$.*

It is undetermined *at layer ℓ if it is neither stable, increasing, nor decreasing.*

Note that a channel cannot be both increasing and decreasing, because satisfying the conditions for both classes would imply that it is stable, which is incompatible with being increasing or decreasing.

The following lemma shows that the behaviour of the channel types aligns with their intended interpretation, where stable channels are those not affected by the input dataset.

Lemma 29. *Let $\mathcal{M}$ be a sum-GCN of L layers, and let D', D be datasets satisfying $D' \subseteq D$. For each vertex $v \in \texttt{enc}(D')$, layer $\ell \in \{0, \ldots, L\}$, and channel $i \in \{1, \ldots, \delta_\ell\}$, the following hold:*

- *If i is stable at layer ℓ, then $\mathbf{v}'_\ell[i] = \mathbf{v}_\ell[i]$;*
- *If i is increasing at layer ℓ, then $\mathbf{v}'_\ell[i] \leq \mathbf{v}_\ell[i]$, and*

- *If i is decreasing at layer ℓ, then $\mathbf{v}'_\ell[i] \geq \mathbf{v}_\ell[i]$,*

where $\mathbf{v}_\ell$ and $\mathbf{v}'_\ell$ are the vectors induced for v in layer ℓ by applying $\mathcal{M}$ to D and D' respectively.

Proof sketch. The full proof is given in Appendix C.1.1.

We show the claim of the lemma by induction on ℓ. The base case holds because $\mathbf{v}'_0[i] \leq \mathbf{v}_0[i]$ for each $i \in \{1, \ldots, \delta\}$ by definition of `enc`, and all channels are increasing in layer 0. For the inductive step, we prove that it holds at layer ℓ for stable, increasing, and decreasing i.

For stable i, notice that $(\mathbf{A}_\ell \mathbf{v}_{\ell-1})[i]$ is just a sum over channels j that are stable at $\ell - 1$, since the non-stable j's are zeroed out. The induction hypothesis then implies $\mathbf{v}'_{\ell-1}[j] = \mathbf{v}_{\ell-1}[j]$ for each such j. Furthermore, $\mathbf{B}^c_\ell[i, j] = 0$ for every j and c. Hence, $\mathbf{v}'_\ell[i] = \mathbf{v}_\ell[i]$.

For increasing i, consider the four possibilities for j at $\ell - 1$ and conditions (1) - (3) in the definitions. For example, if j is increasing, then from (1) we have $\mathbf{A}_\ell[i, j] \geq 0$ and by our induction hypothesis, $\mathbf{v}'_{\ell-1}[j] \leq \mathbf{v}_{\ell-1}[j]$. In any of these cases, we find that $\mathbf{A}_\ell[i, j]\mathbf{v}'_{\ell-1}[j] \leq \mathbf{A}_\ell[i, j]\mathbf{v}_{\ell-1}[j]$. For the product involving $\mathbf{B}^c_\ell$, if j is decreasing or undetermined at $\ell - 1$ then from (2) and (3) we have that $\mathbf{B}^c_\ell[i, j] = 0$. Otherwise, by the inductive hypothesis we obtain $\mathbf{u}'_{\ell-1}[j] \leq \mathbf{u}_{\ell-1}[j]$ for every node $u \in \mathtt{enc}(D')$. Then since $(E^c)' \subseteq E^c$, the inequality is preserved when summing over neighbours of v. Also, since from (4) we have that $\mathbf{B}^c_\ell[i, j] \geq 0$, we find that for all j,

$$\mathbf{B}^c_\ell[i, j](\sum_{(v,u)\in(E^c)'} \mathbf{u}'_{\ell-1})[j] \leq \mathbf{B}^c_\ell[i, j](\sum_{(v,u)\in E^c} \mathbf{u}_{\ell-1})[j].$$

Therefore, by monotonicity of σ, $\mathbf{v}'_\ell[i] \leq \mathbf{v}_\ell[i]$. For decreasing i, the proof is very similar to that for increasing i. □

Both increasing and stable channels are amenable to rule extraction. In particular, the soundness check in Proposition 27 extends seamlessly to this new setting.

Proposition 30. *Let $\mathcal{M}$ be a sum-GCN as in Section 6.1, and let r be a safe rule of the form* (2.1) *where H mentions a unary predicate U_p. Let S be an arbitrary set of as many constants as there are variables in r. Assume channel p in $\mathcal{M}$ is increasing or stable at layer L. Then r is sound for $T_{\mathcal{M}}$ if and only if $H\nu \in T_{\mathcal{M}}(D_r^\nu)$ for each substitution ν mapping the variables of r to constants in S and such that $D_r^\nu \models B_i\nu$ for each inequality B_i in r.*

On the other hand, if a channel is decreasing or undetermined, it does not behave monotonically, and so it may or may not have sound rules. I conclude this section by relating the classes of channels described here to those of Section 6.2.1, with a full proof given in Appendix C.1.3.

Theorem 31. *Safe channels are increasing or stable. There exist increasing unsafe channels and stable unsafe channels.*

6.2.3 Unbounded Channels

I now show that certain channels *cannot* admit any sound Datalog rules. In Section 6.2.2, I noted that extracting sound Datalog rules for channels that are decreasing or undetermined might be possible. In this section, I identify a subset of such channels, which I refer to as *unbounded*, for which *no* sound Datalog rule may exist. Thus, being unbounded provides a sufficient condition for non-monotonic channel behaviour.

The techniques in this section require that the sum-GCN uses ReLU as the activation function in all but (possibly) the last layer L. Furthermore, they require the co-domain of the activation function in layer L to include a number strictly less than the threshold t of the classification function $\texttt{cls}_t$. This restriction is non-essential and simply excludes GCNs that derive all possible facts regardless of the input dataset, in which case all rules would be sound.

First, I provide a definition of unbounded channels, which checks to see if there exists a Boolean vector (node label in the input graph) and a sequence of edge colours such that the output of a sequence of matrix multiplications and ReLU applications is negative.

Definition 32. *Channel $p \in \{1, \ldots, \delta_L\}$ is* unbounded *at layer L if there exist a Boolean vector $\mathbf{y}_0$ of dimension δ_0, and a sequence $c_1, \ldots, c_L$ of (not necessarily distinct) colours in Col, such that, with $\{\mathbf{y}_\ell\}_{\ell=1}^{L-1}$ the sequence defined inductively as $\mathbf{y}_\ell := ReLU(\mathbf{B}_\ell^{c_\ell}\mathbf{y}_{\ell-1})$ for each $1 \leq \ell \leq L-1$, it holds that $(\mathbf{B}_L^{c_L}\mathbf{y}_{L-1})[p] < 0$.*

Intuitively, the value of an unbounded channel at layer L for a given vertex can always be made smaller than the classification threshold t by extending the input graph in a precise way. This helps us prove that each rule r with head predicate U_p corresponding to an unbounded channel p is not sound for $T_{\mathcal{M}}$. In particular, we first generate the dataset D_r^ν, where ν is a substitution that maps each variable in r to a different constant, and we let $a := \nu(x)$. Then we extend D_r^ν to a dataset D' in a way that ensures that the value of channel p at layer L for the vertex v_a in D' is smaller than the threshold t. Thus, $U_p(a) \notin T_{\mathcal{M}}(D')$, even though $U_p(a)$ is clearly in $T_r(D')$, so dataset D' is a counterexample to the soundness of r.

The following theorem formally states this result. The full proof of the theorem is given in Appendix C.1.4.

Theorem 33. *Let $\mathcal{M}$ be a sum-GCN as in Equation* (6.1)*, where σ_ℓ is ReLU for each $1 \leq \ell \leq L-1$, and the co-domain of σ_L contains a number strictly less than the threshold t of the classification function $\mathtt{cls}_t$. Then, each rule with head predicate U_p corresponding to an unbounded channel p is unsound for $\mathcal{M}$.*

Proof sketch. Consider an unbounded channel p at layer L and a rule r with head $U_p(x)$. Let ν be an arbitrary substitution mapping each variable in r to a different constant. Let $G = \mathtt{enc}(D_r^\nu)$ and let v_a be the vertex corresponding to constant $a := \nu(x)$. Let $\mathbf{y}_0$ and $c_1, \ldots, c_L$ be a Boolean vector and a sequence of colours, respectively, satisfying the condition in Definition 32 for p, and let $\{\mathbf{y}_\ell\}_{\ell=1}^{L-1}$ be the sequence computed for them as in the definition.

For each $d \in \mathbb{N}$, let D_d be the extension of D_r^ν with

1. facts $R^{c_1}(a_j, b_1)$ for $1 \leq j \leq d$, facts $R^{c_\ell}(b_{\ell-1}, b_\ell)$ for $2 \leq \ell \leq L-1$, and fact $R^{c_L}(b_{L-1}, a)$;

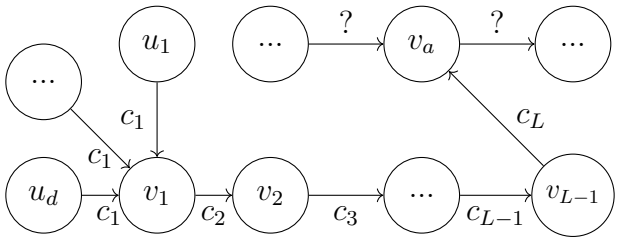


Figure 6.1: Canonical encoding G_d of the dataset D_d that extends D_r^ν in the proof of Theorem 33 for a given $d \in \mathbb{N}$.

2. facts $U_k(a_j)$ for $1 \leq j \leq d$ and $k \leq \delta$ s.t. $\mathbf{y}_0[k] = 1$;

for a_j and b_ℓ distinct constants not in D_r^ν for each j and ℓ. Furthermore, let G_d be the canonical encoding of D_d, where v_ℓ is the vertex corresponding to constant b_ℓ and u_j is the vertex corresponding to constant a_j. The resulting graph G_d is illustrated in Figure 6.1, where edges labelled with question marks represent the canonical encoding of D_r^ν.

We next show that there exists $d^* \in \mathbb{N}$ such that the value of channel p at layer L for vertex v_a in the canonical encoding of D_{d^*} is under the threshold t. To find d^*, we inductively define a number d_ℓ for each $1 \leq \ell \leq L-1$ that satisfies the following condition:

(C1) for all $d \geq d_\ell$, the value of the feature vector computed by $\mathcal{M}$ on G_d in layer ℓ for vertex v_ℓ is $\mathbf{g}_\ell + d \cdot \mathbf{y}_\ell$, where $\mathbf{g}_\ell$ is a fixed vector that does not depend on d.

Each d_ℓ can be defined inductively (assuming $d_0 = 1$) as follows. Condition (C1) ensures that if $d \geq d_{\ell-1}$, the value of $\mathbf{v}_\ell[j]$ computed by equation (2.3) for vertex $v = v_\ell$ in layer ℓ when applying $\mathcal{M}$ to graph G_d is of the form $\mathrm{ReLU}(S_{j,d})$, for $S_{j,d}$ a sum that contains a term of the form $z_j = d \cdot (\mathbf{B}_\ell^{c_\ell} \mathbf{y}_{\ell-1})[j]$ and no other terms depending on d. We define d_ℓ as the smallest integer large enough to ensure that each z_j dominates the sum of all other terms in $S_{j,d}$. Hence, if $d \geq d_\ell$, we have that if z_j is positive, then so is $S_{j,d}$, and ReLU will act on it as the identity; if z_j is negative, then so is $S_{j,d}$, and ReLU will map it to 0, and if $z_j = 0$, then $\mathrm{ReLU}(S_{j,d})$ will be independent of d. Thus, in all three cases, $\mathrm{ReLU}(S_{j,d})$ is of the

form $\mathbf{g}_\ell[j] + d \cdot \mathbf{y}_\ell[j]$, for $\mathbf{g}_\ell[j]$ a number that does not depend on d, and so Condition (C1) holds for d_ℓ as required.

Finally, we consider the instance of equation (2.3) that computes the value of vertex v_a in layer L when applying $\mathcal{M}$ to graph G_d, for $d \geq d_{\ell-1}$. Condition (C1) ensures that the value of $\mathbf{v}_L[p]$ is the result of applying σ_L to a sum that contains a term of the form $d \cdot (\mathbf{B}_L^{c_L} \mathbf{y}_{L-1})[p]$ and no other terms depending on d. The definition of an unbounded channel then ensures that $(\mathbf{B}_L^{c_L} \mathbf{y}_{L-1})[p]$ is negative, and so by choosing d sufficiently large, the aforementioned sum is arbitrarily small towards $-\infty$. But σ_L is monotonic and includes a number less than t in its codomain, which means that there exists a minimum natural number d such that the image by σ_L of the aforementioned sum is under the threshold t; we then let d^* be this number.

Dataset D_{d^*} is therefore a counterexample showing that r is not sound for $\mathcal{M}$. Indeed, when applying $\mathcal{M}$ to G_{d^*}, the value of channel p for vertex v_a in layer L is smaller than the threshold t, and so we have that $U_p(a) \notin T_{\mathcal{M}}(D_{d^*})$. However, $U_p(a) \in T_r(D_{d^*})$, since D_{d^*} contains D_r^ν and all inequalities in the body of r are satisfied because ν maps each variable to a different constant. Therefore, rule r is unsound for $\mathcal{M}$. □

Notice that the proof depends specifically on the use of ReLU as the activation function in the sum-GCN, since it requires that (1) the output of the activation function tends to infinity as the input does, and (2) negative inputs to the activation function result in an output of 0. It is unclear whether these techniques can be adapted to other unbounded activation functions.

In order to check which channels are unbounded in practice, I initialize an empty set S of unbounded channels. I then enumerate all Boolean feature vectors $\mathbf{y}_0$ of dimension δ_0 and all sequences $c_1, \ldots, c_L$ of colours; for each vector and sequence, I compute the vector $\mathbf{B}_L^{c_L} \mathbf{y}_{L-1}$ as shown in Definition 32 and add to S each channel p such that $(\mathbf{B}_L^{c_L} \mathbf{y}_{L-1})[p]$ is negative. The enumeration can be stopped early if all channels are found to be unbounded (i.e. whenever S becomes $\{1, \ldots, \delta\}$).

In most practical cases, however, the dimension δ and number of colours $|\text{Col}|$ and layers L are large enough that it is not practically feasible to enumerate all

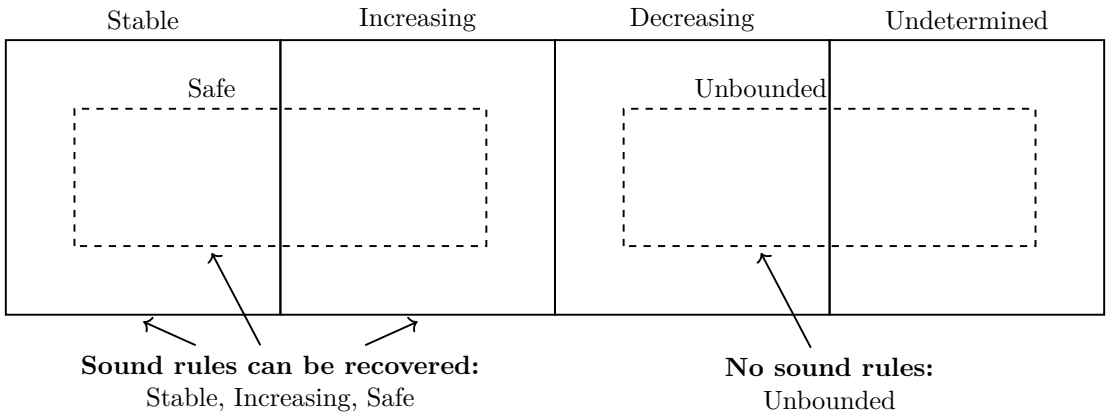


Figure 6.2: A summary of the different channel types introduced in this chapter, their relationships to one another, and whether sound rules can be extracted from them or whether they can be shown to admit no sound rules.

combinations. In those cases, I randomly sample a fixed number of pairs $(\mathbf{y}_0, \{c_\ell\}_{\ell=1}^{L})$ of Boolean feature vectors and sequences of colours. All channels in S after this iteration will be unbounded, but channels not in S may also be unbounded.

To conclude this section, I relate unbounded channels to those defined in Section 6.2.1 and Section 6.2.2. A full proof is given in Appendix C.1.5.

Theorem 34. *Unbounded channels are neither increasing, nor stable, nor safe. There exist, however, decreasing unbounded channels and undetermined unbounded channels.*

In Figure 6.2, I summarize the different channel types introduced in this chapter, their relationships to one another, and whether sound rules can be extracted from them or whether they can be shown to admit no sound rules. Recall that stable, increasing, decreasing, and undetermined channels are all mutually exclusive, and that every channel is classified as one of the four.

6.3 Experiments

In this section, I describe the experiments run in this chapter. The experiments identify safe, stable, increasing, and unbounded channels, extracting sound rules for the stable and increasing channels. First, I discuss my experiment setup and the datasets that I use. Next, I describe different training paradigms to encourage the

sum-GCNs to have stable or increasing channels. Finally, I present and analyse the results. I find that R-GCN never has any sound rules in practice, as all channels are unbounded after training. In general, I find a trade-off between model performance and rule extraction, as training variants that tend to result in higher accuracy also result in a lower number of sound rules.

6.3.1 Experiment Setup

I train sum-GCNs on several link prediction datasets, using the non-canonical dataset transformation described in Section 2.4.3. I use the non-canonical dataset transformation to convert the problems to node-labelling ones, rather than introducing the additional complication of scoring functions, as in Chapter 5. For each dataset, I train a sum-GCN with ReLU activation functions, biases, and two layers (this architecture corresponds to R-GCN with biases). The hidden layer of the GCN has twice as many channels as its input. Moreover, I also train four additional instances of sum-GCNs, using a modified training paradigm to facilitate the learning of sound rules (see Section 6.3.2). For each trained model, I compute standard classification metrics, as defined in Section 2.5.2. I train all my models using binary cross entropy loss and the Adam optimizer with a standard learning rate of 0.001. For each model, I choose the classification threshold by computing the accuracy on the validation set across a range of 108 thresholds between 0 and 1 and selecting the one which maximizes accuracy. I run each experiment across ten different random seeds.

Channel Classification and Rule Extraction For each trained model, I compute which output channels of the model were safe, stable, increasing, and unbounded (using 1000 random samples of pairs of Boolean feature vectors and colour sequences). On all datasets, for each channel p that is stable or increasing, I iterate over each Datalog rule in the signature with up to two body atoms and a head predicate U_p, and count the number of sound rules, using Proposition 30 to check soundness. In benchmarks with a large number of predicates, I only check

rules with one body atom, since searching the space of rules with two body atoms is intractable. I also check if the injected rules in the LogInfer datasets are sound.

Datasets I use three standard benchmarks, WN18RRv1, fb237v1, and nellv1 [258], as well as the LogInfer [162] datasets of WN-hier, WN-sym, WN-superhier, FB-hier, FB-sym, FB-superhier, WN-hier_nmhier, and WN-cup_nmhier, as described in Section 2.5.3. Models are trained for 8000 epochs on WN18RRv1, FB237v1, NELLv1, and the LogInfer-WN datasets, and for 1500 epochs on the LogInfer-FB datasets.

6.3.2 Training Paradigm Variations

In these experiments, I use the term "MGCN" to refer to monotonic sum-GCNs, trained as described in Section 2.5.1. This is the same training procedure used for the monotonic GCNs and scoring functions in Chapter 5. Since the weights are non-negative, every channel is trivially safe and rules can be checked for soundness. However, MGCNs are purely monotonic and are thus unable to learn non-monotonic patterns in the data. When a sum-GCN is trained in the standard way, I refer to it as R-GCN in these experiments. I also use early-stopping in my normal training of R-GCNs and MGCNs variation, ceasing training if the model loss deteriorates for 50 epochs in a row.

To encourage sum-GCNs to exhibit non-monotonic behaviour while simultaneously learning sound monotonic rules, I propose a novel variation to the training routine. My method relies on matrix weight clamping: intuitively, having more zero matrix weights leads to a higher number of safe, stable, and increasing channels. Smaller weights affect the actual computation less, so I clamp those before larger ones. To clamp a sum-GCN $\mathcal{M}$ using a *clamping threshold* $\tau \in \mathbb{R}^+$, I set to 0 each model weight $\mathbf{A}_\ell[i,j]$ such that $|\mathbf{A}_\ell[i,j]| \leq \tau$ (and likewise for each $\mathbf{B}^c_\ell[i,j]$). Given a parameter $X \in [0, 100]$, let τ_X be the minimum absolute value of a matrix weight of $\mathcal{M}$ such that the percentage of output channels of $\mathcal{M}$ that are stable or increasing is strictly greater than X. I train three separate instances of each model corresponding to three values of X: 0, 25, and 50; for a given value of X, after

each epoch, I compute τ_X and clamp the model's matrices using τ_X as a clamping threshold. I refer to the model trained for X as R-X. These models are applied to all datasets except those derived from FB (LogInfer-FB and FB237v1), as it is prohibitively slow to compute which channels are increasing or stable at every epoch due to the high number of predicates used in the dataset.

6.3.3 Results

Monotonic LogInfer Datasets The results on the monotonic LogInfer benchmarks (i.e. those using only monotonic rule patterns) are shown in Table 6.1. I note that, across every experiment, R-GCN never has any sound rules, since every channel is found to be unbounded. These results are very surprising, since the monotonic rule patterns in the LogInfer datasets are very simple. They are especially surprising for super-hierarchy, where most positive examples are consequences of simple rules. This stands in contrast with the near-perfect accuracy that R-GCN obtained on the monotonic LogInfer benchmarks: for example, 99.66% for WN-superhier. This indicates that even in cases where accuracy is very high, R-GCN may not have learned any sound rules, which raises doubts about the usefulness of accuracy as a metric for KG completion models, since its high value does not correspond to sound rules (in this case, which is a very controlled setting).

In Table 6.1, R-GCN consistently achieves a lower loss than MGCN on the training data due to the lack of restrictions. However, it is almost always outperformed by MGCN on the test set. I attribute this to the strong inductive bias that MGCNs have for these particular problems, which require learning monotonic rules. Furthermore, the rules used to augment the datasets are often sound for the MGCN models, which demonstrates that they have effectively learned the relevant completion patterns for these datasets.

Benchmark Datasets The findings of my experiments on the benchmark datasets are shown in Table 6.2. AUPRC is consistently higher for R-GCN on the validation set than for MGCN, but accuracy on the test set is always higher for MGCN. This

Dataset	Model	%Acc	%Prec	%Rec	Loss	%UB	%Stable	%Inc	%SO	%NG	#1B	#2B
WN-hier	R-GCN	**98.87**	**99.32**	98.42	**9172**	100	0	0	0	0	0	0
	MGCN	98.75	97.56	**100**	40366	0	0	100	**100**	0	**31**	**3095**
WN-sym	R-GCN	98.95	99.2	98.69	**13113**	100	0	0	0	0	0	0
	MGCN	**100**	**100**	**100**	35544	0	19.09	80.91	**100**	0	**17**	**1671**
WN-superhier	R-GCN	99.66	**99.7**	99.63	**21331**	100	0	0	0	0	0	0
	MGCN	**99.67**	99.34	**100**	108789	0	10	90	**98**	0	**24**	**2348**
FB-hier	R-GCN	92.21	94.28	89.88	**17209**	100	0	0	0	0	0	0
	MGCN	**97.47**	**98.28**	**96.65**	164345	0	3.08	96.92	**54.05**	45.95	**523**	-
FB-sym	R-GCN	94.03	95.72	92.21	**26397**	100	0	0	0	0	0	0
	MGCN	**98.11**	**96.61**	**99.72**	127030	0	27.22	72.78	**91.4**	8.6	**1929**	-
FB-superhier	R-GCN	98.87	99.25	98.48	**56247**	100	0	0	0	0	0	0
	MGCN	**99.67**	**99.6**	**99.75**	426788	0	2.15	97.85	**51.9**	48.1	**307**	-

Table 6.1: Results for the monotonic LogInfer datasets. Loss is on the training set. %UB, %Stable, and %Inc are the percentages of unbounded, stable, and increasing channels, respectively. %SO is the percentage of LogInfer rules that are sound for the GCN and %NG is the percentage of monotonic LogInfer rules that are not sound for the GCN since there is some grounding of the body which does not entail the head. #1B and #2B are the number of sound rules with one and two body atoms respectively.

Dataset	Model	%Acc	%Prec	%Rec	AUPRC	Loss	%UB	%Stable	%Inc	%Safe	#1B	#2B
FB237v1	R-GCN	68.4	**99.86**	36.85	0.1407	**24**	100	0	0	0	0	0
	MGCN	**74.95**	82.89	**64.05**	**0.1456**	765	0	69.44	30.56	100	**13640**	-
NELLv1	R-GCN	60.4	47.96	22.35	**0.08**	**687**	100	0	0	0	0	0
	R-0	61.24	83.18	24.71	0.0769	1342	92.86	2.14	5	6.43	0	10
	R-25	67.35	76.62	36.35	0.0766	2095	71.43	9.29	19.29	27.96	6	799
	R-50	71.53	**91.99**	46.59	0.0761	3148	40.71	18.57	38.57	56.43	9	1055
	MGCN	**91.94**	86.17	**100**	0.0737	2549	0	14.29	85.71	100	**26**	**3300**
WN18RRv1	R-GCN	94	96.9	90.97	**0.1197**	**941**	100	0	0	0	0	0
	R-0	94.52	**98.36**	90.55	0.1176	1109	87.78	10	1.11	11.11	0	16
	R-25	94.09	96.82	91.45	0.1157	1544	61.11	28.89	4.44	33.33	3	293
	R-50	95.15	98.32	91.88	0.1163	1513	38.89	40	15.56	55.56	3	222
	MGCN	**95.18**	97.52	**92.73**	0.114	2292	0	44.44	55.56	100	**7**	**681**

Table 6.2: Results for the benchmark datasets. AUPRC is on the validation set and Loss on the training set. %UB, %Stable, %Inc, and %Safe are the percentages of unbounded, stable, increasing, and safe channels, respectively. #1B and #2B are the number of sound rules with one and two body atoms respectively.

is because, for most constant pairs used in the validation set targets, there exists no fact in the input dataset where these constants appear together. The dataset transformation I use, however, can only predict a fact if the constants in it appear together in a fact of the input dataset. This means that there are few data points with which to choose a sensible threshold on the validation set. It is also possible that R-GCN is overfitting on the validation set. Nevertheless, the results illustrate how, as X increases in model R-X, the ratios of stable/increasing/safe channels and the number of sound rules increase, whilst the AUPRC generally deteriorates.

Dataset	Model	%Acc	%Prec	%Rec	%UB	%Stable	%Inc	%Safe	%SO	%NG	%NB	#1B	#2B
WN-hier_nmhier	R-GCN	**89.0**	**87.65**	**90.88**	100	0	0	0	0	0	100	0	0
	R-0	79.48	78.18	82.33	90.91	0	9.09	8.18	0	0	100	5	513
	R-25	71.13	75.03	66.17	72.73	0	27.27	25.45	8	2	90	10	947
	R-50	69.59	74.63	61.94	45.45	4.55	50	51.82	30	10	60	**36**	**3589**
	MGCN	66.87	74.37	56.38	0	0	100	100	**52**	48	0	31	3042
WN-cup_nmhier	R-GCN	**83.32**	**83.36**	83.47	100	0	0	0	0	0	100	0	0
	R-0	77.22	77.3	78.38	90.91	0.91	8.18	9.09	0	0	100	3	320
	R-25	71.16	69.6	76.32	72.73	5.46	21.82	26.36	10	10	80	20	2023
	R-50	66.8	67.54	66.88	44.55	7.27	47.27	49.09	34	16	50	23	2249
	MGCN	62.74	59.25	**85.27**	0	4.55	95.45	100	**50**	50	0	**50**	**4905**

Table 6.3: Results for the mixed LogInfer-WN datasets. %UB, %Stable, %Inc, and %Safe are the percentages of unbounded, stable, increasing, and safe channels, respectively. %SO is the percentage of monotonic LogInfer rules that are sound for the GCN, %NG is the percentage of rules that are not sound for the GCN since there is some grounding of the body which does not entail the head, and %NB the percentage that are not sound since their heads correspond to unbounded channels. #1B and #2B are the number of sound rules with one and two body atoms respectively.

Mixed LogInfer Datasets The findings on the mixed LogInfer-WN datasets, that is, those that use both monotonic and non-monotonic rule patterns, are shown in Table 6.3 and demonstrate a clear tendency: as the number of channels required to be increasing/stable goes up (from R-GCN, to R-X, to MGCN), accuracy decreases, whilst the number of sound rules and percentage of sound injected LogInfer rules increase. This demonstrates a trade-off between interpretability and empirical performance. As in the case of purely monotonic datasets, R-GCN achieves consistently superior accuracy, but has no sound rules, since every channel is unbounded. Notice that on WN-cup_nmhier, MGCN achieves high recall but low precision. This indicates that the model is learning versions of the non-monotonic rules without the negation, and thus achieving a high recall but getting penalized by false positives, since it is predicting facts that are not derived by the non-monotonic rules. I observe that the MGCN solution does not scale to every situation, as it can lead to poor performance. Given that many real-world datasets may also have non-monotonic patterns, MGCNs could be too restrictive.

Channel Classification As can be seen across all results, the classification of channels into unbounded, stable, and increasing almost always adds up to 100%. This demonstrates empirically that, whilst my theoretical analysis allows for the

existence of undetermined or decreasing channels that are not unbounded, virtually no such channels are found when using my proposed training paradigms, and so my proposed channel classification allows one to fully determine whether each channel shows monotonic or non-monotonic behaviour. Furthermore, across all experiments on the LogInfer datasets, for every rule used to inject facts, my rule-checking procedures were always able to classify the rule as sound or provably not sound. Finally, the results show that the total number of channels found to be either stable or increasing is often larger than the number of channels shown to be safe. This can be seen by computing the sum of the %Stable and %Inc columns and comparing it to %Safe. Although the difference is not large, this validates the need for the more complex classification of channels into stable/increasing/decreasing/undetermined, instead of the simpler classification into safe/unsafe.

6.4 Discussion

In this chapter, I provided two ways to extract sound rules from sum-GCNs and a procedure to prove that certain rules are not sound for a sum-GCN. My methods rely on classifying the output channels of the sum-GCN. I found that, in my experiments, R-GCN, a specific variant of sum-GCN, provably has no sound rules when trained in the standard way, even when using ideal datasets. I considered two alternatives to train sum-GCNs, the first of which clamps all negative weights to zero and results in entirely monotonic sum-GCNs, yielding good performance and many sensible sound rules on datasets with monotonic patterns, but poor performance on datasets with a mixture of monotonic and non-monotonic patterns. My second alternative clamps small negative weights to zero, and yields a trade-off between accuracy and the number of sound rules. I found that, in practice, almost every channel of the sum-GCN is classified in a manner that allows one to either extract sound rules or prove that there are no associated sound rules.

Limitations My analysis only considers GCNs with sum aggregation and monotonically increasing activation functions with non-negative range: it is unclear how sound rules can be extracted from models that use mean aggregation or GELU activation functions. Furthermore, my techniques to prove that certain channels cannot admit sound Datalog rules depend on the GCN using ReLU activation functions. Finally, my methods do not guarantee that every output channel will be characterized such that one can show that it either has no sound rules or it can be checked for sound rules.

Future Work For future work, this approach could be applied to other GNN architectures and the rule extraction extended to non-monotonic logics. Furthermore, the negative weight clamping procedure could be replaced by a soft penalty in the loss function for having negative weights. As an alternative approach, one could define relaxed definitions of soundness and examine whether rules are approximately sound, even if they are provably not formally sound.

7

Sound Logical Explanations for Mean Aggregation GCNs

Contents

7.1 Introduction 123
7.2 Sound Rules and Explanations 126
7.2.1 Relationship of MAGCNs with FOL 126
7.2.2 Which Monotonic Rules can be Sound for MAGCNs . . 128
7.2.3 Explaining MAGCN Predictions 132
7.3 Experiments 135
7.3.1 Experiment Setup 135
7.3.2 Results 137
7.4 Discussion 140

7.1 Introduction

Whilst Chapter 5 considered purely monotonic GCNs, in Chapter 6 I relaxed the monotonicity requirement by considering sum-GCNs with negative weights. In this chapter, I consider a different variation of GCNs that led to them being non-monotonic: the choice of aggregation function. Although any function mapping multisets of real numbers to real numbers can be used for the aggregation function in a GCN, standard options include mean, sum, max, and attention [268]. The

results of Tena Cucala et al. [254, 256] and Chapter 5 only apply to GCNs with non-negative weights and max or sum aggregation functions, since these functions exhibit monotonic behaviour when using only non-negative values. However, this is not true for mean aggregation.

In practice, mean aggregation often emerges as the default choice [112, 138, 225, 234, 294] due to its simplicity, stability during training, and good empirical performance. With regard to the expressivity of mean-GCNs, Rosenbluth et al. [225] study mean, sum, and max aggregation in terms of their ability to approximate one another. In a comprehensive paper, Grohe [103] shows that GNNs with piecewise linear activation functions (e.g. ReLU) and dyadic rational parameters[1] can be approximated with arbitrary precision by formulas from the logic GFO+C.[2] This result includes mean-GNNs: in fact, the approximation error arises precisely because dyadic rationals cannot simulate the division involved in mean aggregation. However, to my knowledge, there are no other prior works analysing the logical expressivity of or attempting to extract sound rules from GCNs that use mean aggregation (mean-GCNs). Given the popularity and usefulness of mean aggregation, this is an important area of study.

My Contributions To obtain explanatory sound rules for mean-GCNs, I consider mean-GCNs with non-negative weights (MAGCNs). This isolates the source of the non-monotonicity to the aggregation function (since negative weights can also be a source of non-monotonicity, as in Chapter 6), enabling a simpler analysis. In Tena Cucala et al. [254] and Chapter 6, it was found that restricting GCNs to non-negative weights did not greatly impact performance when using max or sum aggregation, respectively.

In this chapter, I provide two main theoretical results. First, in Section 7.2, I prove the exact class of monotonic $\mathcal{ALCQ}$ rules that can be sound for MAGCNs, which turns out to be very limited. I also provide a means for checking if such rules

[1] A dyadic rational is a number that can be expressed as a fraction whose numerator is an integer and whose denominator is a power of two.

[2] This is the guarded fragment of FOL with only two variables, with the inclusion of counting terms. Note that the counting terms mean that this logic is beyond FOL.

are sound. Second, in light of the fact that some MAGCNs do not have equivalent FOL programs, I instead provide a restricted fragment of FOL that contains sound rules that can be used to explain any prediction of a MAGCN. This chapter is similar to Chapter 6, in that only a subset of the model predictions can be explained by sound monotonic rules, whereas the monotonic GCNs considered in Chapter 5 can be fully characterized by sound monotonic (Datalog) rules.

In my experiments in Section 7.3, I show that mean-GCNs still perform well on benchmark datasets when restricted to having non-negative weights. I also demonstrate that they recover a variety of sound monotonic rules on the datasets with injected monotonic rules, as well as providing examples of the generated rules that explain predicted facts of the GCN. Despite good model performance on the test set, I find that MAGCNs still learn some sound rules which do not make sense in the real-world setting the dataset is capturing, showing the importance of explainability. In comparison to monotonic max-GCNs or sum-GCNs, my experiments show no clear advantage to using MAGCNs: performance is comparable between different aggregation functions for monotonic GCNs (the non-monotonicity of MAGCNs appears to not provide an advantage in practice), and monotonic max-GCNs or sum-GCNs can be fully characterized by simple sound rules, rather than just a subset of the predictions of MAGCNs.

This chapter is based on a paper published in NeurIPS 2025 [190].

Background In this chapter, I consider in particular the class of *monotonic mean-GCNs* (MAGCNs), using the definitions provided in Section 2.3.2. Besides the restriction of non-negative weights, this is the same model used in R-GCN [234] (where the normalization constant is set as described in their paper, to be the number of c-coloured neighbours of the node). Instead of Datalog, I consider first-order logic (Section 2.1.2) and description logics (Section 2.1.3) for extracting sound rules, since mean-GCNs are non-monotonic (even when they have non-negative weights and monotonically increasing activation functions).

As in Chapter 6, other concepts that will be particularly relevant for this chapter include the non-canonical encoding and decoding from Tena Cucala et al. [256] that I use for link prediction (Section 2.4.3), and sound rules for GCN-based dataset transformations (Section 2.4.2). Note that these definitions apply to both FOL and description logic rules/programs.

7.2 Sound Rules and Explanations

In this section, I prove a series of theoretical results for MAGCNs, to identify which monotonic rules can be sound for them and obtain sound explanatory rules. The restriction of non-negative weights isolates the source of the non-monotonicity to the aggregation function, making analysis easier; however, MAGCNs still possess the defining feature of mean-GCNs in their choice of aggregation function. This restriction is the same approach that was used effectively for GCNs with sum (see Chapter 6) or max aggregation [254].

7.2.1 Relationship of MAGCNs with FOL

First, I show that there exist MAGCNs with no equivalent FOL programs. Intuitively, this follows from FOL's inability to express numerical comparisons.

Proposition 35. *There exists a MAGCN $\mathcal{M}$ such that for any dataset D and constant $a \in \texttt{con}(D)$, $U(a) \in T_{\mathcal{M}}(D)$ if and only if at least half of the neighbours b of a in D are such that $U(b) \in D$, where U is a unary predicate. This logical function cannot be defined in FOL.*

Proof. Consider a MAGCN $\mathcal{M}$ over a signature with only one unary predicate (U) and only one binary predicate (P), corresponding to colour c in the canonical encoding. Let $\mathcal{M}$ have $L = 1$ (one layer), with $\mathbf{A}_1 = (0)$, $\mathbf{b}_1 = (0)$, and $\mathbf{B}_1^c = (1)$. Finally, let σ_1 be the identity function and $t_{\mathcal{M}} = 0.5$.

Consider a dataset D and constant a in D. Then $\mathbf{v}_L^a$, the labelling of v^a after $\mathcal{M}$ is applied to $\texttt{enc}(D)$, depends only on the number and type of the direct neighbours of v^a. Each neighbour v^b either has $U(b) \in D$ or $U(b) \notin D$. We end up with

$$\mathbf{v}_L^a[1] = \frac{|\{v^b \mid (v^a, v^b) \in E^c \text{ and } \mathbf{v}_L^b[1] = 1\}|}{|N_c(v^a)|} = \frac{|\{b \mid P(a,b) \in D \text{ and } U(b) \in D\}|}{|\{b \mid P(a,b) \in D\}|}$$

Now $\mathbf{v}_L^a[1] \geq 0.5 = t_{\mathcal{M}}$ if and only if $|\{b \mid P(a,b) \in D \text{ and } U(b) \in D\}| \geq |\{b \mid P(a,b) \in D \text{ and } U(b) \notin D\}|$. I.e. $U(a) \in T_{\mathcal{M}}(D)$ if and only at least half of the neighbours b of a in D are such that $U(b) \in D$.

However, this logical function cannot be expressed in FOL. To prove this, we consider Libkin [154], who defines the majority predicate $\text{MAJ}(\varphi, \psi)$, which tests if all constants satisfying φ contains at least half of the constants satisfying ψ. We find that, for every dataset D and constant $a \in \texttt{con}(D)$, we have $(D, a) \models \text{MAJ}(R(a,x) \wedge U(x), R(a,x))$ if and only if "at least half of the neighbours b of a in D are such that $U(b) \in D$" if and only if $U(a) \in T_{\mathcal{M}}(D)$.

However, Libkin [154, Corollary 13.17] proves that MAJ queries cannot be defined in an extension of FOL, from which it trivially follows that they cannot be defined in FOL. Thus, $\mathcal{M}$ is equivalent to a logical function that cannot be defined in FOL. This can also be shown using the zero-one law for first-order logic [100], which intuitively says that as the size of a random dataset (equivalently, a random graph) tends to infinity, then any property expressible in FOL becomes either almost surely true or almost surely false. Thus, since the probability of MAJ holding is 0.5 as the size tends to infinity, it cannot be expressed in FOL. □

This means that the task of trying to find equivalent FOL programs for them, such as in Chapter 5, is impossible. Instead, I aim to extract sound rules from MAGCNs, to explain their predictions. Thus, I consider in particular a class of "monotonic" rules encompassing many common inference patterns, and identify the restricted fragment that can be sound for MAGCNs. Finally, I define a rule language using a fragment of FOL that, for any mean-GCN prediction, contains a sound rule that entails the prediction. The restriction to considering only FOL is a limitation of this study, and considering FOL with the majority predicate MAJ could be interesting future work.

7.2.2 Which Monotonic Rules can be Sound for MAGCNs

Since equivalent programs cannot be found in some cases, I instead first show which simple monotonic rules can be sound for MAGCNs. Monotonic rules appear commonly in practice [1, 162, 249], are often easily human-readable, and are easier to use for extracting sound rules. Recall the definition of a monotonic rule, Definition 2, which I restate below:

Definition. *A FOL / $\mathcal{ALCQ}$ rule (or program) r is* monotonic *(under dataset inclusion) if for all datasets D, D' such that $D \subseteq D'$, $T_r(D) \subseteq T_r(D')$.*

At this point, it is worth briefly distinguishing between various notions of monotonicity. Recall that a GCN is said to be monotonic if all its weights are non-negative and each activation function is monotonically increasing with non-negative range (see Section 2.3.2). Thus, whilst MAGCNs are non-monotonic in their behaviour, I still refer to them as "monotonic" GCNs. Second, note that FOL is *monotonic under entailment*; however, it is not monotonic under injective homomorphisms or dataset inclusion, which are the notions of logical monotonicity I use in my thesis. Third, whilst a FOL / $\mathcal{ALCQ}$ rule (or program) can be sound for a GCN, the notion of soundness is purely based on model-checking, and not on closure (due to the potential non-monotonicity of the rule or program). Finally, we can also discuss the monotonicity of $\mathcal{ALCQ}$ concepts using model-checking semantics, which aligns with monotonicity under dataset inclusion. Going forward in this chapter, when discussing the monotonicity of a rule or concept, I am referring to monotonicity under dataset inclusion.

Monotonic Description Logics Description logics, such as $\mathcal{ALCQ}$ [15], are a natural choice for providing a language for sound rules due to their widespread use in data modelling and their theoretical relationship to GCNs [19]. However, some $\mathcal{ALCQ}$ concepts are inherently non-monotonic: for example, if $(D, c) \models \forall P.A$ for some atomic concept A, binary predicate P, dataset D, and constant $c \in \texttt{con}(D)$, then for $D' = D \cup \{P(c, d)\}$ with a fresh constant d, $(D', c) \not\models \forall P.A(c)$.

$\mathcal{EL}$ [15] is a more restricted description logic than $\mathcal{ALCQ}$, where concepts use only intersection and existential quantification—any $\mathcal{EL}$ rule is thus monotonic. I define $\mathcal{ELUQ}$, a restricted fragment of $\mathcal{ALCQ}$ that subsumes $\mathcal{EL}$, as the language containing concepts defined by

$$C ::= \top \mid A \mid C \sqcap D \mid C \sqcup D \mid \exists P.C \mid \geq_n P.C,$$

where C, D are $\mathcal{ELUQ}$ concepts, P is a binary predicate, A is an atomic concept, and n a positive integer. Note that any concept $\exists P.C$ can be written equivalently as $\geq_1 P.C$. An $\mathcal{ELUQ}$ rule has the form $C \sqsubseteq A$. This language is similar[3] to that of Datalog with inequalities considered in Chapters 5 and 6, and by Tena Cucala et al. [256] for GCNs with sum aggregation and non-negative weights, given constraints on the inequalities and the addition of disjunction. All $\mathcal{ELUQ}$ rules are monotonic; furthermore, Wałęga and Cuenca Grau [273] prove that $\mathcal{ELUQ}$ is the maximal fragment of $\mathcal{ALCQ}$ that is preserved under injective homomorphisms (i.e. the maximal monotonic fragment). Since any $\mathcal{ELUQ}$ rule is monotonic, Proposition 1 shows that the containment relation that defines soundness for an $\mathcal{ELUQ}$ rule or program still holds when the operators are composed a finite number of times.

Sound Monotonic Rules We find that for any $\mathcal{ELUQ}$ rule r that is sound for a MAGCN $\mathcal{M}$, there exists a set of rules of a very simple form that (1) are each sound for $\mathcal{M}$ and (2) collectively subsume r. This is formalized in the following theorem.

Theorem 36. *Let $\mathcal{M}$ be a MAGCN and r an $\mathcal{ELUQ}$ rule that is sound for $\mathcal{M}$. Then there exists a finite set of rules R such that:*

1. *Each $r' \in R$ has the form $\exists P_1.\top \sqcap ... \sqcap \exists P_j.\top \sqcap A_1 \sqcap ... \sqcap A_k \sqcap \top \sqsubseteq A_{k+1}$, where each P_i is a binary predicate, A_i a unary predicate, and $j, k \in \mathbb{N}_0$.*
2. *Each $r' \in R$ is sound for $\mathcal{M}$.*
3. *R subsumes r.*

[3]Whether $\mathcal{ELUQ}$ is equivalent to Datalog with inequalities is currently an open question.

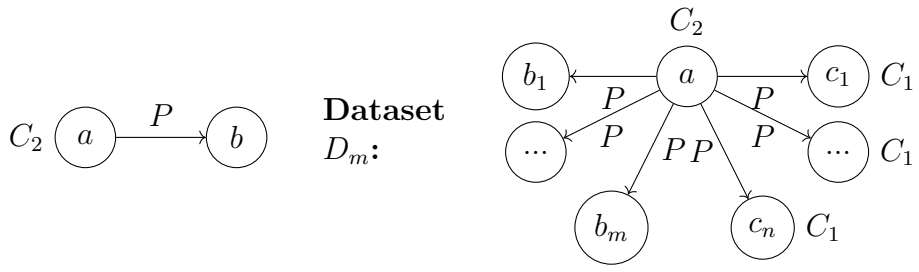
The proof provides a construction of R. For example, given an $\mathcal{ELUQ}$ rule $r : \geq_3 P_1.(A_1 \sqcup A_2) \sqcap \exists P_2.(\exists P_1.A_3) \sqcup A_4 \sqsubseteq A_5$ that is sound for $\mathcal{M}$, we can construct rules $r_1 : \exists P_1.\top \sqcap \exists P_2.\top \sqsubseteq A_5$ and $r_2 : A_4 \sqsubseteq A_5$ such that both r_1 and r_2 are sound for $\mathcal{M}$ and $\{r_1, r_2\}$ subsumes r.

As a consequence of this, when checking for sound $\mathcal{ELUQ}$ rules it suffices to check for rules of the above restricted form. In addition, whilst the space of all $\mathcal{ELUQ}$ rules is infinite, the restricted fragment I provide is finite, meaning that all sound $\mathcal{ELUQ}$ rules can be covered by instead iterating over a finite number of restricted rules. Finally, this raises concerns for the monotonic FOL expressivity of MAGCNs, since any sound $\mathcal{ELUQ}$ rule is ultimately subsumed by a set of much simpler sound rules.

Proof sketch. The full proof of Theorem 36 is given in Appendix D.1.1. A different r' is defined for each concept in the outer disjunction of the body of r. To see the crucial step of the proof, consider some rule r without disjunction in the outer concept, and let $r_1 := r$. Then r_1 is of the form $\geq_n P.C_1 \sqcap C_2 \sqsubseteq A$, where P is a binary predicate, C_1 an $\mathcal{ELUQ}$ concept, C_2 an $\mathcal{ELUQ}$ concept without disjunction, and n a positive integer.

We inductively define a sequence $r_2, ..., r_k$ of $\mathcal{ELUQ}$ rules that such for all $i \in \{1, ..., k-1\}$, r_{i+1} is sound for $\mathcal{M}$ and r_{i+1} subsumes r_i. Given r_i of the form $\geq_n P.C_1 \sqcap C_2 \sqsubseteq A$, we define r_{i+1} as the rule $\exists P.\top \sqcap C_2 \sqsubseteq A$. Trivially, r_{i+1} subsumes r_i.

Now consider the following datasets D_1 and D_m (parametrized by $m \in \mathbb{N}$). In D_1, a concept C_2 annotating a constant a denotes that there exists other facts in the dataset such that $(D_1, a) \models C_2$ (similarly for D_m).



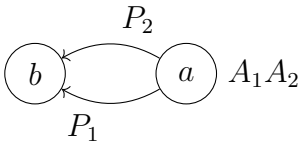
Note that D_1 is an instantiation of the body of r_{i+1}. The soundness of r_{i+1} depends on the fact that as $m \to \infty$, the computation of $\mathcal{M}$ on D_m at constant a tends to that of $\mathcal{M}$ on D_1 (and that $A(a) \in T_{r_i}(D_m)$).

Since r_1 contains a finite number of $\exists$ or $\geq_n$ operators and each r_{i+1} has one fewer $\exists$ or $\geq_n$ operator than r_i, this inductive construction is guaranteed to terminate for some k. We include r_k in R. Once this has been done for every $r_1 \in R'$, we have R subsumes r. □

Checking for Monotonic Rule Soundness It remains to be shown how one can verify the soundness of $\mathcal{ELUQ}$ rules. Since we only need to consider rules of the form given in Theorem 36, there are exactly $\delta \cdot 2^{\delta \cdot |\text{Col}|}$ possible rules to check. For rules r_1, r_2, when considering the sets of body concepts B_{r_1} and B_{r_2}, if $B_{r_1} \subseteq B_{r_2}$ and r_1 is sound, then r_2 is sound: I use this to optimize the procedure. The following proposition allows one to check the soundness of $\mathcal{ELUQ}$ rules.

Proposition 37. *Let $\mathcal{M}$ be a MAGCN and $r : \exists P_1.\top \sqcap ... \sqcap \exists P_j.\top \sqcap A_1 \sqcap ... \sqcap A_k \sqsubseteq A_{k+1}$ be a rule of the form given in Theorem 36. We define the dataset $D_{base} := \{A_1(a), ..., A_k(a), P_1(a,b), ..., P_j(a,b)\}$ for constants $a \neq b$. Then r is sound for $\mathcal{M}$ if and only if $A_{k+1}(a) \in T_{\mathcal{M}}(D_{base})$.*

For example, to check if a rule $r : \exists P_1.\top \sqcap \exists P_2.\top \sqcap A_1 \sqcap A_2 \sqsubseteq A_3$ is sound, it suffices to compute the output of $\mathcal{M}$ on the following dataset $D_{\text{base}} = \{A_1(a), A_2(a), P_1(a,b), P_2(a,b)\}$, which can be represented as follows:



Then r is sound for $\mathcal{M}$ if and only if $A_3(a) \in T_{\mathcal{M}}(D_{\text{base}})$. Note that this method also works if the body of the rule r is empty, by simply checking if $A_{k+1}(a) \in T_{\mathcal{M}}(D_\emptyset)$, where $D_\emptyset = \{P(b,a)\}$ for an arbitrary binary predicate P.

Proof sketch. The full proof of Proposition 37 is given in Appendix D.1.2. It relies on the fact that $(D_{\text{base}}, a) \models B$, where B is the body of r. Furthermore, no extension

to D_{base} will decrease the output at v^a, and any dataset satisfying B is an extension of D_{base} (up to constant relabelling, unmerging the neighbours of v^a, and letting $b = a$). Note that none of these merges / unmerges will decrease the output at v^a and that MAGCNs are agnostic to the particular constants used in the input dataset. □

Consequences for Link Prediction As in Chapter 6, to perform link prediction, I combine the canonical encoding with the dataset encoding-decoding scheme defined by Tena Cucala et al. [256]. As described in Section 2.4.3, nodes correspond to pairs of constants and node labels can then encode binary facts. Graph edges are used only to connect nodes that have constants in common; and so the presence of an edge is independent of whether any specific fact holds in the dataset.

As noted in Section 2.4.3, the encoding and decoding can be described by a set of rules, each expressed in a simple extension of Datalog. Given a sound rule for the canonical transformation, we can combine it (via unfolding) with the rules representing the encoder and decoder to obtain a Datalog rule that is sound for the entire link prediction transformation. This observation, combined with Theorem 36, ensures that each monotonic rule sound for the entire link prediction transformation is subsumed by rules of the form $R_1(x, y) \wedge ... \wedge R_{m-1}(x, y) \rightarrow R_m(x, y)$, where $R_1, ..., R_m$ are binary predicates from the signature. This is obtained by unfolding a rule of the form given in Theorem 36, since the unary predicates A_i are unfolded into the binary predicates they represent, and the existentials $\exists P_j.\top$ can safely be removed since the edge presence is independent of any specific fact.

7.2.3 Explaining MAGCN Predictions

Given the non-monotonic nature of mean aggregation, a MAGCN may make predictions that cannot be explained by any of the above sound $\mathcal{ELUQ}$ rules. In this section, I provide a family of rules Ω (a fragment of FOL) to explain any prediction of a MAGCN. More precisely, for any MAGCN $\mathcal{M}$, dataset D, and predicted fact $A(a) \in T_{\mathcal{M}}(D)$, there exists a rule $r \in \Omega$ such that $A(a) \in T_r(D)$ and

r is sound for $\mathcal{M}$. To achieve this, I define a procedure to derive the rule $r \in \Omega$, given $\mathcal{M}$, D, and $A(a)$. The family Ω aims to be as limited as possible, so that rules can be kept smaller and more general than simply instantiating the entire input dataset as the body of a rule. Nevertheless, the rules are much less useful as explanations than rules of the form given in Theorem 36.

Logic Language Needed We first define the family of rules using $\mathcal{ALCQ}$ syntax, with the addition of a new operator, rather than defining it directly in FOL. The operator is $\exists_n$ ("exists n unique"): for example: $\exists_3 P.(A_1, A_2, \top)$. Its semantics are defined as follows. For concepts $C_1, ..., C_n$, binary predicate P, dataset D, and constant $c \in D$, we define $(D, c) \models \exists_n P.(C_1, ..., C_n)$ if and only if there exist distinct constants $c_1, ..., c_n \in \texttt{con}(D)$ such that $(D, c_1) \models C_1, ..., (D, c_n) \models C_n$ and $P(c, c_1), ..., P(c, c_n) \in D$. Written in first-order logic, it is defined as $(D, c) \models \exists_n P.(C_1, ..., C_n) \equiv$

$$(D, c) \models \exists x_1 ... \exists x_n (x_i \neq x_j \text{ for all } i, j \in \{1, ..., n\} \text{ such that } i < j)$$
$$\wedge C_1(x_1) \wedge ... \wedge C_n(x_n) \wedge P(x, x_1) \wedge ... \wedge P(x, x_n),$$

for free variable x. We then define the family of rules Ω as follows. An Ω-concept C is defined by $C ::= \top \mid A \mid C_1 \sqcap C_2 \mid \exists_n P.(C_1, ..., C_n) \sqcap \leq_m P.\top$, where A is any atomic concept, n any positive integer, $m \geq n$ an integer, $C_1, ..., C_n$ any Ω-concepts, and P any binary predicate. A rule in Ω has the form $C \sqsubseteq A$, where C is any Ω-concept and A any atomic concept.

Note that Ω is not a monotonic fragment of FOL, which was necessary since MAGCNs are themselves non-monotonic.

Explanatory Rule Let $\mathcal{M}$ be a MAGCN, D a dataset, and $A(a) \in T_{\mathcal{M}}(D)$ a prediction of the MAGCN. We define a rule $r \in \Omega$ by $C_L^a \sqsubseteq A$, where the only part of $\mathcal{M}$ that the rule depends on is L, which is what guarantees that the rule will be finite. The body concept C_L^a is defined as follows:

Definition 38. *For a dataset D, $c \in \texttt{con}(D)$, $\ell \in \mathbb{N}_0$, we inductively define a concept C_ℓ^c by first defining $E_\ell^c := \top \sqcap A_1 \sqcap ... \sqcap A_k$, where $A_1, ..., A_k$ are all atomic concepts A_i such that $A_i(c) \in D$. Then, if $\ell = 0$, we define $C_\ell^c := E_\ell^c$.*

Otherwise, for each binary predicate P, let $c_1, ..., c_n$ be all the constants $c_i \in \texttt{con}(D)$ such that $P(c, c_i) \in D$. Then, if $n > 0$, let $F_\ell^{c,P} := \exists_n P.(C_{\ell-1}^{c_1}, ..., C_{\ell-1}^{c_n}) \sqcap \leq_n P.\top$. Note that each $C_{\ell-1}^{c_i}$ is defined inductively.

Then we define $C_\ell^c := E_\ell^c \sqcap F_\ell^{c,P_1} \sqcap ... \sqcap F_\ell^{c,P_m}$, where $P_1, ..., P_m$ are all the binary predicates in the signature for which $F_\ell^{c,P}$ is defined.

The following theorem then shows that the rule both explains the prediction (by producing the same fact on the dataset as the MAGCN) and is sound for the MAGCN.

Theorem 39. *For any MAGCN $\mathcal{M}$, dataset D_a, and fact $A(a) \in T_\mathcal{M}(D_a)$, the rule $r : C_L^a \sqsubseteq A$ (dependent on D_a) is sound for $\mathcal{M}$, and $A(a) \in T_r(D_a)$.*

Consider $D_a = \{A_1(a), P_1(a, b_1), P_1(a, b_2), P_2(b_3, a), A_2(b_2), P_2(b_2, c), P_3(c, d)\}$ and 2-layer GCN $\mathcal{M}$, for example, such that $A_3(a) \in T_\mathcal{M}(D)$. Then, using Theorem 39, we construct the rule $r : C_2^a \sqsubseteq A_3$, where C_2^a is

$$\top \sqcap A_1 \sqcap \exists_2 P_1.(\top, \top \sqcap A_2 \sqcap \exists_1 P_2.(\top) \sqcap \leq_1 P_2.\top) \sqcap \leq_2 P_1.\top.$$

Notice that $(D_a, a) \models C_2^a$, so we have $A_3(a) \in T_r(D)$. Furthermore, $A_3(a) \in T_\mathcal{M}(D)$ is a sufficient witness for the soundness of r, as we will prove below.

Proof sketch. The full proof of Theorem 39 is given in Appendix D.1.4.

Notice that $(D_a, a) \models C_L^a$ follows from the inductive construction of C_L^a, from which we obtain $A(a) \in T_r(D_a)$, i.e. that r produces the same fact on the dataset as the MAGCN. It remains to be shown that r is sound for $\mathcal{M}$. Let D_b be a dataset: to show soundness, we prove that $T_r(D_b) \subseteq T_\mathcal{M}(D_b)$, so let $A(b) \in T_r(D_b)$.

First, we define a dataset D_L^a to be the L-hop neighbourhood of the constant a in D_a (analogously to how L-hop neighbourhoods are defined for nodes in graphs). Then we show that $A(a) \in T_\mathcal{M}(D_L^a)$, since $A(a) \in T_\mathcal{M}(D_a)$ — this follows from the definition of D_L^a as the L-hop neighbourhood. Finally, we prove $A(b) \in T_\mathcal{M}(D_b)$,

since $A(a) \in T_{\mathcal{M}}(D_L^a)$ — this follows from the fact that any differences between D_b and D_L^a cannot yield a lower MAGCN output for v^b in comparison to v^a, plus a mapping between constants of D_b and D_L^a. □

7.3 Experiments

In this section, I describe the experiments run in this chapter. The experiments test standard GCN and monotonic GCN performance for different aggregation functions, finding that monotonic GCNs often have comparable performance to their standard counterparts. I introduce the datasets, explain how sound monotonic rules are extracted from MAGCNs, and also how explanatory rules are generated. Finally, I present and analyse the results, finding a number of sound rules, some of which are sensible and others of which do not make sense in the real-world context of the datasets.

7.3.1 Experiment Setup

I train GCNs across the link prediction datasets of WN18RRv1, FB237v1, NELLv1, WN-hier, WN-sym, and WN-hier_nmhier, and the node classification dataset of LUBM (see Section 2.5.3). Note that WN-hier, WN-sym, and LUBM are generated using monotonic rule systems, WN-hier_nmhier is generated using a mixture of monotonic and non-monotonic rules, and that WN18RRv1, FB237v1, NELLv1 are real-world benchmarks. Thus, running experiments with WN-hier, WN-sym, LUBM, and WN-hier_nmhier tests whether MAGCNs can recover sound monotonic rules in monotonic (or partially monotonic) settings.

I show that sound rules and explanations can be found for MAGCNs in practice, and that the restriction of monotonicity does not significantly decrease performance. For the model architecture, I fix a hidden dimension of twice the input dimension, 2 GCN layers, ReLU after the first layer, and sigmoid after the second layer.

Recall that the GCN definition given in Section 2.3.2 describes GCNs aggregating in the reverse direction of the edges. For my experiments, I again follow the standard approach and aggregate in the direction of the edges. Thus, when presenting

a rule, I write each binary predicate as its inverse. For example, "advisor" is written as "advisorOf".

I use GCNs with max, sum, and mean aggregation. I train each model for 8000 epochs, stopping training early if loss does not improve for 50 epochs. For all trained models, I compute standard classification metrics, as defined in Section 2.5.2. For each model, I choose the classification threshold by computing the accuracy on the validation set across a range of 108 thresholds between 0 and 1, selecting the one which maximizes accuracy (this is the same approach that was taken in Chapter 6). I train all my models using binary cross entropy loss and the Adam optimizer with a learning rate of 0.001. I train models without restrictions as baselines (denoted by "Standard"), as well as models with non-negative weights (denoted by "Non-Neg") as described in Section 2.5.1. I use "Non-Neg" instead of "Monotonic" to avoid confusion, since the "monotonic mean-GCNs" are technically non-monotonic due to the use of mean aggregation. I run each experiment across five different random seeds. Experiments were run using 34 days of total compute time.

Rule Extraction and Explanations On all datasets, I use Proposition 37 to check for monotonic rules of the form given in Theorem 36. This procedure is described in Algorithm 2 of Appendix D.2.1. Given the differing number of unary and binary predicates in each dataset and the exponential growth of the rule space, I check all rules up to a differing number of body concepts: 4 for LUBM, all 15 for WN-based-datasets, 1 for FB237v1, and 2 for NELLv1. This yields a total possible number of sound rules: 383670 for LUBM, all 360448 for WN18RRv1 and the LogInfer datasets, 56169 for FB237v1, and 216600 for NELLv1. When a rule is a subsumed by another with fewer body atoms, it is not counted.

I also compute sound explanatory rules of the form given in Theorem 39 for all true positive predictions made on the test set of LUBM. Given that the explanatory rules are often large, I improve the process by first checking if a sound rule of the form given in Theorem 36 derives the fact on the input dataset, and return the rule

Dataset	Agg	Weights	%Acc	%Prec	%Rec	F1
LUBM	Mean	Standard	97.1 ± 0.0	96.9 ± 0.0	97.2 ± 0.0	97.1 ± 0.0
		Non-Neg	91.5 ± 0.0	87.8 ± 0.0	96.4 ± 0.0	91.9 ± 0.0
WN-hier	Mean	Standard	99.5 ± 0.0	99.5 ± 0.0	99.5 ± 0.0	99.5 ± 0.0
		Non-Neg	99.8 ± 0.0	99.7 ± 0.0	99.9 ± 0.0	99.8 ± 0.0
WN-sym	Mean	Standard	99.4 ± 0.0	99.5 ± 0.0	99.3 ± 0.0	99.4 ± 0.0
		Non-Neg	100 ± 0.0	100 ± 0.0	100 ± 0.0	100 ± 0.0
WN-hier_nmhier	Mean	Standard	86.2 ± 0.0	84.7 ± 0.0	88.4 ± 0.0	86.5 ± 0.0
		Non-Neg	71.6 ± 0.0	80.1 ± 0.1	58.8 ± 0.1	67.2 ± 0.0
FB237v1	Mean	Standard	68.7 ± 0.0	95.4 ± 0.0	39.3 ± 0.0	55.7 ± 0.0
		Non-Neg	71.8 ± 0.0	75.4 ± 0.0	64.8 ± 0.0	69.7 ± 0.0
WN18RRv1	Mean	Standard	93.7 ± 0.0	98.5 ± 0.0	88.8 ± 0.0	93.4 ± 0.0
		Non-Neg	95.5 ± 0.0	98.1 ± 0.0	92.7 ± 0.0	95.3 ± 0.0
NELLv1	Mean	Standard	75.2 ± 0.1	93.8 ± 0.0	53.4 ± 0.2	65.7 ± 0.2
		Non-Neg	93.4 ± 0.0	88.8 ± 0.0	99.4 ± 0.0	93.8 ± 0.0

Table 7.1: Results for mean-GCNs with standard / non-negative weights, across all datasets. Metrics are computed on the test set and shown with a 95% CI.

Dataset	Tot	Un	Bin	Mix	0	1	2	3	4	5	6	7	8
LUBM	11.6	1.4	9.8	0.4	2	9.6	1.4	0.6	0				
WN-hier	22.6	15.6	0	7	0	2.6	5.4	6.6	1.4	3.8	2.2	0.4	0.2
WN-sym	0.4	0	0	0.4	0	0	0	0	0.4	0	0	0	0
WN-hier_nmhier	53.6	4.6	0	49	0	0	1.8	16.2	17.8	12.6	4.8	0	0.4
FB237v1	136	136	0	0	29	136							
WN18RRv1	0	0	0	0	0	0	0	0	0	0	0	0	0
NELLv1	1	1	0	0	0	0	1						

Table 7.2: Counts of monotonic rules of the form given in Theorem 36. Tot, Un, Bin, Mix are the counts of the total number of rules, rules with only unary atoms, rules with only binary atoms, and rules with a mix of atom arities. Each remaining column i counts the number of rules with i body concepts.

as an explanation if it does. If no such rule exists, I use rules from Theorem 39. Other strategies for improving the rule quality are discussed in Appendix D.2.2.

7.3.2 Results

Mean-GCN performance on the test set is shown in Table 7.1. For completeness, results across all aggregation functions are given in Tables D.1 to D.3 of Appendix D.3. We see similar behaviour for mean-GCNs as for sum and max-GCNs when they are restricted to non-negative weights, which matches the behaviour seen in Chapter 6

for sum-GCNs and in Tena Cucala et al. [254] for max-GCNs. On LUBM, the restriction yields a small decrease in performance. For the monotonic LogInfer datasets, there is a slight increase in performance, and for the non-monotonic one, a substantial decrease, since the dataset explicitly penalizes monotonic model behaviour. Finally, on the benchmark datasets, the restriction improves performance, sometimes substantially. This demonstrates that the restriction of mean-GCNs to MAGCNs can be done in practice without sacrificing significant model performance, while enabling the extraction of sound rules. I hypothesize that the occasional performance increase seen when restricting the models to non-negative weights is due to (1) it regularizing the model and (2) the dataset patterns not requiring negative weights to be captured.

Sound Monotonic Rules In Table 7.2, I present counts of the number of sound monotonic rules that were obtained on each dataset. On LUBM, WN-hier, WN-hier_nmhier, and FB237v1, I found a number of sound rules with a varying number of body concepts. On average, I found 2 sound rules on LUBM that have empty bodies, and 29 on FB237v1; these rules are clearly absurd, but are what the GCN has learned, highlighting the need for sound rules so that such issues in the models can be exposed.

Further to this, I note that almost no sound monotonic rules were obtained on WN-sym, WN18RRv1, or NELLv1. This implies that the MAGCN learned functions that are explicitly non-monotonic. For WN-sym, which was populated with the consequences of monotonic rules, this behaviour is due to the very restricted form that sound monotonic rules can have when used with the link prediction encoding of Tena Cucala et al. [256], as discussed in Section 7.2.2. In Chapter 6, on the other hand, I found that all the rules used to create WN-sym were sound when using monotonic sum-GCNs. I show some sample sound monotonic rules in Table 7.3, with more given in Table D.4 of Appendix D.3.

Dataset	Examples of Sound Rules
LUBM	$\top \sqsubseteq$ Publication $\exists$advisorOf.$\top \sqsubseteq$ AssociateProfessor Department $\sqcap$ UndergraduateStudent $\sqcap \exists$advisorOf.$\top \sqsubseteq$ FullProfessor
WN-hier	_member_of_domain_usage(X, Y) $\rightarrow$ _member_meronym(X, Y)
WN-sym	_has_part(X, Y) $\wedge$ _similar_to(X, Y) $\rightarrow$ _derivationally_related_form(X, Y)
WN-hier_nmhier	_hypernym$(X, Y) \wedge$ _synset_domain_topic_of$(X, Y) \rightarrow$ _also_see(X, Y)
FB237v1	$\top \rightarrow$ /music/instrument/instrumentalists(X, Y) /base/biblioness/bibs_location/state(X, Y) $\rightarrow$ /film/film/genre(X, Y)
NELLv1	organizationterminatedperson(X, Y) $\wedge$ topmemberoforganization(X, Y) $\rightarrow$ organizationhiredperson(X, Y)

Table 7.3: Randomly sampled sound monotonic rules of the form given in Theorem 36.

Fact	Publication(http://www.Department10.University0.edu/FullProfessor5/Publication3)
Rule	$\top \sqsubseteq$ Publication
Fact	ResearchAssistant(http://www.Department1.University0.edu/GraduateStudent14)
Rule	GraduateStudent $\sqsubseteq$ ResearchAssistant
Fact	GraduateStudent(http://www.Department12.University0.edu/GraduateStudent48)
Rule	ResearchAssistant $\sqsubseteq$ GraduateStudent
Fact	GraduateStudent(http://www.Department3.University0.edu/GraduateStudent44)
Rule	ResearchAssistant $\sqcap \exists$authorOfPublication.$\top \sqsubseteq$ GraduateStudent
Fact	Course(http://www.Department12.University0.edu/Course1)
Rule	$\exists$courseTakenBy.$\top \sqcap \exists$hasTeacher.$\top \sqsubseteq$ Course
Fact	AssistantProfessor(http://www.Department13.University0.edu/AssistantProfessor10)
Rule	$\exists$advisorOf.$\top \sqsubseteq$ AssistantProfessor
Fact	University(http://www.University772.edu)
Rule	$\exists$grantedUndergraduateDegreeTo.$\top \sqsubseteq$ University

Table 7.4: Randomly sampled explanatory rules that have successfully been reduced in size, shown with the corresponding facts produced by the MAGCN on LUBM.

Explanations In Table D.5 of Appendix D.3, I give samples of explanatory rules of the form given in Theorem 39. On LUBM, these explanatory rules had 25 body concepts on average. When using the improved procedure of first checking for explanatory rules of the form given in Theorem 36, the rules averaged only 11 body concepts. On average, out of 1990 true positive predictions, the process only failed 22 times to explain the predicted fact using a rule from Theorem 36. This shows that in the monotonic setting of LUBM, MAGCNs can almost fully characterize the predictions of the model with sound monotonic rules. These rules are small and general, and are thus very useful for explanations in comparison with rules of the form given in Theorem 39.

In Table 7.4, I give samples of such explanatory rules, which are useful for providing insights into the data, as well as exposing issues with the trained MAGCN. For example, the explanatory rule $\top \sqsubseteq$ Publication exposes a strange pattern learned by the MAGCN, which arises as a consequence of publications in LUBM having no incoming edges. Likewise, it is sensible that the MAGCN has learned that research assistants are graduate students, but it has also learned that all graduate students are research assistants (which is not the case in the setting the dataset is capturing). Finally, note that the fourth rule in the table is subsumed by the third rule — this is because they come from two different MAGCNs, which have learned different sound rules. Table D.6 of Appendix D.3 provides a longer list of explanatory rules that have been reduced in size.

7.4 Discussion

I considered monotonic mean-GCNs (MAGCNs), showing first that they can express logical functions that go beyond FOL. I proved which $\mathcal{ELUQ}$ rules (a class of monotonic $\mathcal{ALCQ}$ rules) can be sound for MAGCNs, which turned out to be very limited. The resulting rule space is finite, which allowed me to define a procedure to check for all sound $\mathcal{ELUQ}$ rules. I also provided a restricted fragment of FOL, from which sound rules can be constructed to explain any prediction of a MAGCN.

In my experiments, I found that mean-GCNs still perform well on benchmark datasets when restricted to having non-negative weights. I also found a variety of sound monotonic rules on half the datasets, with provably almost no sound monotonic rules on the other half. This raise questions about how mean-GCNs can be encouraged to recover the underlying rules in the datasets, and suggests that max- or sum-GCNs may be preferable when provable soundness is required. It also shows the importance of rule extraction techniques, given the good empirical performance of MAGCNs on the test datasets whilst having sound rules that do not make sense in the real-world context of the datasets (for example, that everything is a publication). Finally, I computed explanatory rules for predictions made by MAGCNs on the LUBM dataset.

In comparison to monotonic max-GCNs or sum-GCNs, I find no clear advantage to using MAGCNs: model accuracy is comparable when using different aggregation functions for monotonic GCNs, and monotonic max-GCNs or sum-GCNs can be fully characterized by sound monotonic rules, rather than only a subset of the MAGCN predictions.

Related Work Further works also provide theoretical analyses of GNNs with mean aggregation. Vasileiou et al. [266] provide a generalization bound for mean-GNNs. Adam-Day et al. [3] show that mean-GNNs used for graph classification tend to a constant function as the sizes of the input graphs tend to infinity (under certain random graph models). Their approach of tending the size to infinity is similar to my technique used in the proof of Theorem 36. In contemporaneous work, Schönherr and Lutz [235] identify the FOL expressivity of mean-GNNs in the uniform (standard) and non-uniform (only on graphs up to a bounded number of nodes) settings. In the uniform setting, they prove that any continuous mean-GNN (that is equivalent to a FOL classifier) has an equivalent logical classifier expressible in alternation-free modal logic (AFML). However, they do not provide a construction of the AFML classifier, a demonstration that such a construction

would be obtainable in practice, or a means to recover sound rules if the mean-GNN is not equivalent to a FOL classifier.

With respect to extracting $\mathcal{ELUQ}$ rules as explanations, Köhler and Heindorf [139] only consider explanations in $\mathcal{EL}$, which is a fragment of $\mathcal{ELUQ}$. Furthermore, their extracted rules are not provably sound.

Limitations My work is limited in several ways. First, recall that one is able to check the infinite space of $\mathcal{ELUQ}$ rules for soundness by instead checking finitely many rules of the form given in Theorem 36; however, the size of this finite space grows exponentially in the number of predicates, making the search intractable as the number of predicates increases (as seen on FB237v1, for example). Furthermore, I only consider the $\leq$ operator for the mean-GCN step classification function $\texttt{cls}_t$ due to its standard use in prior work [254, 256] and the fact that using $<$ yields an entirely different set of theoretical results (for example, rules with a body of $\exists P.A$ could be sound for such a mean-GCN, without $\exists P.\top$ being sound). Finally, restriction to considering only FOL is a limitation of this study, and considering FOL with the majority predicate MAJ could be interesting future work.

8 Conclusions

Contents

8.1 Summary **143**
8.2 Future Work **146**

In this chapter, I conclude the thesis. First, in Section 8.1, I summarize my novel contributions and draw some overall conclusions. Then, in Section 8.2, I discuss some potential directions for future work.

8.1 Summary

In this thesis, I aimed to address the expressivity and the explainability of MPNNs when they are applied to KGs. In particular, I considered their expressivity in relation to graph functions that go beyond equivariance, as well as their logical expressivity, and I provided sound logical explanations for various GCN models using Datalog, FOL, and description logics.

Tailored extensions to MPNNs proved crucial when learning non-equivariant graph functions, showing the importance of analysing expressivity. With respect to logical explanations, I found that monotonic GCNs still yield the same explainability benefits when combined with scoring functions. On the other hand, when relaxing

the restrictions that make GCNs monotonic (specifically, having non-negative weights and using a max or sum aggregation function), I found that GCNs often provably have no sound Datalog rules in practice, and that the classes of monotonic rules that can be sound for them are very limited.

In Chapter 4, I demonstrated the inherent limitations of equivariant functions regarding their capacity to distinguish between similar nodes. To mitigate these constraints while maintaining a robust structural inductive bias, I introduced the concept of orbit-equivariance and established its theoretical position within the hierarchy of node-labelling functions. Furthermore, I developed a taxonomy of orbit-equivariant functions characterized by the `max-orbit` property. I proposed four distinct extensions to MPNNs designed to solve non-equivariant tasks and conducted a formal analysis of their expressive power in relation to orbit-equivariance and `max-orbit`. Empirical evaluations on both synthetic and real-world datasets revealed that one of the proposed architectures, Orbit-Indiv-MPNN, consistently achieved superior performance. Finally, the results indicated that the use of a specialized loss function is often a prerequisite for learning on orbit-equivariant tasks.

In Chapter 5, I considered monotonic GCNs with scoring functions, to provide a more fine-grained analysis of their expressivity and to obtain sound logical explanations. I showed how scoring functions may be rendered monotonically increasing, established how sound Datalog rules can be identified for monotonic GCNs utilizing such functions, and illustrated how prior findings concerning the ability of scoring functions to represent rule patterns influence the expressivity of the associated GCN models. Furthermore, I extended prior work to derive equivalent programs for any monotonic GCN employing max aggregation with a monotonically increasing scoring function, as well as for those utilizing max-sum aggregation with a non-negative bilinear scoring function. My empirical results indicated that the practical performance of GCNs does not decline significantly when the restrictions for monotonicity are imposed (and may occasionally improve) while also confirming that numerous sound rules can be recovered from these models to provide explanations for their predictions.

In Chapter 6, I relaxed the constraints on monotonic GCNs by considering sum-GCNs with negative weights. I provided two ways to extract sound rules from sum-GCNs and a procedure to prove that certain rules are not sound for a sum-GCN. My approach centres on the classification of the output channels of the sum-GCN; using this, I observed that R-GCN (a particular sum-GCN variant) possesses no provably sound rules when trained using standard techniques, even upon ideal datasets. To address this, I evaluated two training alternatives: the first involves clamping all negative weights to zero to produce entirely monotonic sum-GCNs, which achieved good performance and generated numerous sensible sound rules on datasets with monotonic patterns, but poor performance on datasets containing some non-monotonic patterns. My second alternative, which involves clamping only small negative weights to zero, facilitates a trade-off between predictive accuracy and the number of recoverable sound rules. Ultimately, my empirical findings indicate that nearly every channel of the sum-GCN can be classified in a way that either enables the extraction of sound rules or confirms their absence.

Finally, in Chapter 7, I considered mean-GCNs, whereas Chapters 5 and 6 only considered GCNs with max and sum aggregation. Specifically, I examined monotonic mean-GCNs (MAGCNs), initially demonstrating their capacity to represent logical functions beyond FOL. I identified the specific $\mathcal{ELUQ}$ rules (the maximal monotonic subset of $\mathcal{ALCQ}$) that can be sound for MAGCNs, finding this class to be very limited; however, because the resulting rule space is finite, I was able to define a procedure to verify all sound $\mathcal{ELUQ}$ rules. Furthermore, I established a restricted FOL fragment containing sound rules to explain any MAGCN prediction. My empirical findings indicate that mean-GCNs retain high performance on benchmark datasets despite non-negative weight restrictions. However, while I extracted various sound monotonic rules from half of the datasets, others provably had none. These results, alongside the recovery of several incoherent sound rules, suggest that max- or sum-GCNs may be superior when provable soundness is required. This discrepancy between strong predictive accuracy and the learning of rules that do not make sense in the real-world context of the datasets (such as the universal classification

of constants as publications) highlights the necessity of rule extraction techniques. Lastly, I computed explanatory rules for positive predictions made by MAGCNs on the LUBM dataset. In my experiments, I found no distinct advantage to MAGCNs over monotonic max- or sum-GCNs, as predictive accuracy remained comparable across different aggregation functions, whereas monotonic max- and sum-GCNs offer the benefit of being fully characterized by sound monotonic rules rather than a mere subset of predictions.

Overall, I find model expressivity to be an important consideration, as in Chapter 4 I show the limits of standard MPNNs, and in Chapters 5 and 7 I respectively show some expressivity limits of monotonic GCNs with scoring functions and MAGCNs. These expressivity results are important since they show inductive biases of the models (which may be useful in different settings), as well as proving the limits of what MPNNs and GCNs can express.

With respect to provably sound logical explanations, my methods demonstrate that these can be recovered in settings beyond those considered by Tena Cucala et al. [254, 256]. However, both the relaxation of the non-negative weights restriction in Chapter 6 and the use of mean aggregation in Chapter 7 come with significant downsides, in that sound monotonic rules can no longer be recovered for all predictions. This shows that explainability via sound rules often comes at the expense of restricting the GCN, and that having fewer restrictions tends to result in a diminished capacity for explainability.

8.2 Future Work

The findings of my thesis have raised many other questions that could be pursued in future work, which I will summarize in this section.

First, the identification of orbit-equivariance as a structural property offers significant potential for subsequent research within this field. To begin, one can generalize the concept of orbit-equivariance to alternative data architectures beyond graphs [42]. Although extensions to MPNNs have demonstrated utility in the designing of optimal bioisosteres, the inherent complexity of this application

requires deeper inquiry, including biological effects. Furthermore, formalizing orbit-equivariance as a distinct property may facilitate the recognition of applications that necessitate non-equivariant or orbit-equivariant frameworks. There are also opportunities to design more sophisticated, specialized orbit-equivariant neural networks. Whilst my theoretical results prove the satisfaction of properties, there are still open expressivity questions about which orbit-equivariant functions different non-equivariant models are able to represent.

With respect to my results in Chapter 5 pertaining to monotonic GCNs with scoring functions, the equivalent programs I construct for monotonic max-sum-GCNs are only for those that use non-negative bilinear scoring functions. However, being non-negative bilinear is a sufficient (but not necessary) condition for obtaining equivalent programs. Thus, there is an open question as to what properties of a scoring function are strictly necessary. Finally, the monotonic GCN and scoring function models needed extra signal from positive facts during training; this is worth exploring further, and potentially designing alternative training regimes to facilitate learning.

My theoretical results in Chapter 6 that extract sound rules only consider GCNs with sum aggregation and monotonically increasing activation functions with non-negative range. It is unclear how sound rules can be recovered from models that use other activation functions, such as GELU. Furthermore, my techniques to prove that certain channels cannot admit sound Datalog rules depend on the GCN specifically using ReLU activation functions. The existing proof cannot be easily adapted to work with other activation functions, as it depends on four exclusive properties of ReLU, specifically that it is unbounded, has non-negative range, is linear, and zeroes any input that is negative. Finally, my methods do not guarantee that every output channel will be characterized such that one can show that it either has no sound rules or it can be checked for sound rules: such a result would be very useful. In regard to model training, the negative weight clamping procedure could be substituted for a soft penalty in the loss function for having negative weights.

In the theory of Chapter 7, I only consider the $\leq$ operator for the mean-GCN step classification function $\texttt{cls}_t$. In future work, one could consider the consequences for MAGCNs when using $<$ in the classification function. This would lead to a different set of theoretical results, since the proofs around which $\mathcal{ELUQ}$ rules can be sound for MAGCNs rely specifically on using $\leq$. For example, rules with a body of $\exists P.A$ could be sound for such a mean-GCN, without $\exists P.\top$ being sound: this is not the case when using $\leq$, as proven in the chapter.

Across my thesis, when it comes to searching the space of sound rules and recovering equivalent programs, there is a frequent gap between the potential to obtain sound rules and whether the search is feasible in practice. For example, in Chapter 7, despite being able to check the infinite space of $\mathcal{ELUQ}$ rules for soundness by instead checking finitely many simple rules, the size of this finite space becomes intractable as the number of predicates increases. This is because, in general (as seen across Chapters 5 to 7), the number of possible sound rules is exponential in the number of predicates in the signature.

As an alternative approach to recovering sound rules (as done in Chapters 5 to 7), one could define relaxed notions of soundness and examine whether rules are approximately sound, even if they are provably not formally sound. This could provide a middle ground between the provable soundness considered in this thesis, and the unsound explanations considered by other works. For example, recall that a rule or program α is sound for a GCN $\mathcal{M}$ if $T_\alpha(D) \subseteq T_\mathcal{M}(D)$ for each dataset D. This strict inclusion could be relaxed to a "soundness error", which measures how close the inclusion is to holding across a distribution of possible datasets.

Finally, statements about GNN expressivity only provide information about what functions a model could represent; however, an important question in machine learning is what functions a model can actually *learn*. In my thesis, I only answer this question using empirical studies; future work could aim to establish whether MPNNs can learn orbit-equivariant functions, and whether monotonic GCNs can learn different logical functions.

Appendices

A

Appendices for Chapter 4

Contents

A.1 Proofs 151
A.1.1 Proposition 3 151
A.1.2 Proposition 5 151
A.1.3 Proposition 7 153
A.1.4 Theorem 8 154
A.1.5 Theorem 9 156
A.2 Orbit-1-WL 170
A.2.1 Definition 170
A.2.2 Successful Examples 170
A.2.3 Counterexamples for Completeness 171
A.2.4 Orbit-k-WL 173
A.3 Experiments 174
A.3.1 Computational Complexity 174
A.3.2 Datasets 174
A.3.3 Hyperparameters 177
A.3.4 Loss 178
A.3.5 Full Results 181

A.1 Proofs

A.1.1 Proposition 3

Proposition. *Let f be an equivariant node-labelling function and let G be a labelled graph in its domain. If $v, w \in V(G)$ are similar, then $f(G)_v = f(G)_w$.*

Proof. Define σ to be a permutation that maps $v \to w$, and maps other nodes such that σ is an automorphism of G. Such a permutation exists since v and w are similar. Since σ is an automorphism, $\sigma \cdot G = G$. So we prove:

$$\begin{aligned} f(G)_v &= (\sigma^{-1} \cdot \sigma \cdot f(G))_v && \text{(identity permutation)} \\ &= (\sigma^{-1} \cdot f(\sigma \cdot G))_v && \text{(equivariance)} \\ &= f(\sigma \cdot G)_w && (\sigma^{-1} \text{ maps } w \to v) \\ &= f(G)_w && (\sigma \text{ is an automorphism}) \end{aligned}$$

□

A.1.2 Proposition 5

Proposition. *All equivariant functions are orbit-equivariant, but not vice-versa. There exist node-labelling functions which are not orbit-equivariant.*

Proof. We prove the three claims of this proposition separately.

All equivariant functions are orbit-equivariant.

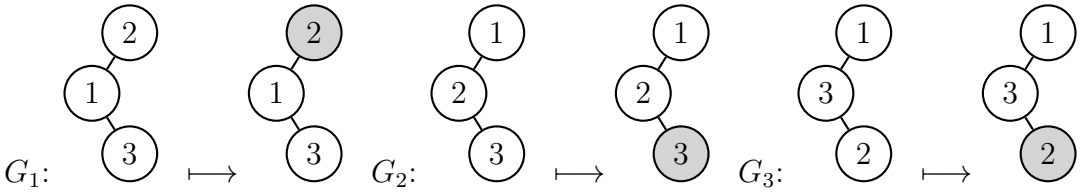
Let a node-labelling function f on domain $\mathcal{G}$ be equivariant. Then for all labelled graphs G and permutations σ on $V(G)$, $f(\sigma \cdot G) = \sigma \cdot f(G)$. To prove that f is orbit-equivariant, choose arbitrary labelled graph $G \in \mathcal{G}$, permutation σ on $V(G)$, and let $r \in R(G)$ be any orbit of G. Then,

$$\begin{aligned} \{\!\{ f(\sigma \cdot G)_{\sigma(v)} \mid v \in r \}\!\} &= \{\!\{ (\sigma^{-1} \cdot f(\sigma \cdot G))_v \mid v \in r \}\!\} && \text{(permute vector)} \\ &= \{\!\{ (\sigma^{-1} \cdot \sigma \cdot f(G))_v \mid v \in r \}\!\} && \text{(equivariance)} \\ &= \{\!\{ f(G)_v \mid v \in r \}\!\} && \text{(inverse permutation)} \end{aligned}$$

Thus, f is orbit-equivariant.

There are orbit-equivariant functions which are not equivariant.

For an example of an orbit-equivariant function that is not equivariant, consider the node-labelling function f, defined on the set of all labelled graphs, that maps input graphs G_1, G_2, G_3 as shown, and assigns 0 to all node labels of each other graph.



This function is orbit-equivariant. There are two orbits of G_1, $r_1 := \{1\}, r_2 = \{2, 3\}$. Consider the permutation $\sigma = \{(1, 2), (2, 1), (3, 3)\}$. Note that $\sigma \cdot G_1 = G_2$ and $\sigma \cdot G_2 = G_1$. For r_1, we have $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r_1\}\!\} = \{\!\{f(G_1)_v \mid v \in r_1\}\!\} = \{\!\{0\}\!\}$, and for r_2, we have $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r_2\}\!\} = \{\!\{f(G_1)_v \mid v \in r_2\}\!\} = \{\!\{0, 1\}\!\}$.

Any permutation on a graph in $D = \{G_1, G_2, G_3\}$ yields a graph in $\mathcal{G}$, and the only graphs that permute to a graph in $\mathcal{G}$ are the graphs in $\mathcal{G}$. In other words, $\mathcal{G}$ and $\mathcal{G}_{\text{all}} \setminus \mathcal{G}$ are both closed under permutation, where $\mathcal{G}_{\text{all}}$ is the set of all labelled graphs. Each graph $G \in \mathcal{G}$ has two orbits r_1, r_2, of size 1 and 2 respectively, with $\{\!\{f(G)_v \mid v \in r_1\}\!\} = \{\!\{0\}\!\}$ and $\{\!\{f(G)_v \mid v \in r_2\}\!\} = \{\!\{0, 1\}\!\}$. For $G_1, r_1 = \{1\}, r_2 = \{2, 3\}$, for $G_2, r_1 = \{2\}, r_2 = \{1, 3\}$, and for $G_3, r_1 = \{3\}, r_2 = \{1, 2\}$.

Thus, following the same unfolding of the definition as above, we see that $\forall G \in D$, $\forall$ permutations σ on $V(G)$, for the corresponding orbit r_1 of G, we have $\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r_1\}\!\} = \{\!\{f(G)_v \mid v \in r_1\}\!\} = \{\!\{0\}\!\}$, and for r_2, we have $\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r_2\}\!\} = \{\!\{f(G)_v \mid v \in r_2\}\!\} = \{\!\{0, 1\}\!\}$.

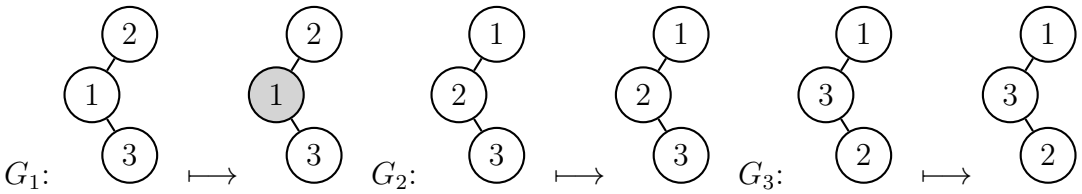
Furthermore, each graph $G \in \mathcal{G}_{\text{all}} \setminus \mathcal{G}$ has $f(G)$ mapping all node labels to 0, and for any permutation σ on $V(G)$, $\sigma \cdot G \in \mathcal{G}_{\text{all}} \setminus \mathcal{G}$, so $f(\sigma \cdot G)$ is also a graph with all node labels equal to 0. Thus, $\forall G \in \mathcal{G}_{\text{all}} \setminus D$, $\forall$ permutations σ on $V(G)$, $\forall r \in R(G)$, we have $\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{f(G)_v \mid v \in r\}\!\} = \{\!\{0\}\!\}^{|r|}$.

Thus, f is orbit-equivariant, since the definition holds $\forall G \in \mathcal{G}$ and $\forall G \in \mathcal{G}_{\text{all}} \setminus \mathcal{G}$.

However, f is not equivariant. For the same permutation $\sigma = \{(1,2),(2,1),(3,3)\}$, $f(\sigma \cdot G_1) = f(G_2)$, which is a graph with the respective node labels of $(0,0,1)$. However, $f(G_1)$ is a graph with the respective node labels of $(0,1,0)$, so $\sigma \cdot f(G_1)$ is a graph with the respective node labels of $(1,0,0)$. So $f(\sigma \cdot G_1) \neq \sigma \cdot f(G_1)$.

There exist node-labelling functions which are not orbit-equivariant.

Consider a node-labelling function f that exhibits the following behaviour for distinct graphs G_1, G_2, G_3 and assigns 0 to all node labels of each other graph.



This function is not orbit-equivariant. Consider the permutation

$$\sigma = \{(1,2),(2,1),(3,3)\},$$

for which $\sigma \cdot G_1 = G_2$, and consider orbit $r = \{1\}$ of G_1. Then $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{f(G_2)_2\}\!\} = \{\!\{0\}\!\}$ and $\{\!\{f(G_1)_v \mid v \in r\}\!\} = \{\!\{f(G_1)_1\}\!\} = \{\!\{1\}\!\}$. So there exists an orbit $r \in R(G_1)$ and permutation σ such that $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r\}\!\} \neq \{\!\{f(G_1)_v \mid v \in r\}\!\}$.

□

A.1.3 Proposition 7

Proposition. *If f is orbit-equivariant and* `max-orbit`$(f) = 1$*, then f is equivariant.*

Proof. Let G be a labelled graph in the domain $\mathcal{G}$ of f and σ a permutation on G. RTP: $f(\sigma \cdot G) = \sigma \cdot f(G)$.

Since `max-orbit`$(f) = 1$, the maximum across all orbits $r \in R(G)$ of the number of unique values in $\{\!\{f(G)_v \mid v \in r\}\!\}$ is 1. Thus, $\forall r \in R(G),\ \forall v, w \in r,\ f(G)_v = f(G)_w$.

Since f is orbit-equivariant:

$$\forall r \in R(G),\ \{\!\{ f(\sigma \cdot G)_{\sigma(v)} \mid v \in r \}\!\} = \{\!\{ f(G)_v \mid v \in r \}\!\}$$

$$\Rightarrow\ \forall \text{ orbits } r \in R(G),\ \forall v \in r,\ f(\sigma \cdot G)_{\sigma(v)} = f(G)_v \qquad \text{(RHS multiset all equal)}$$

$$\Rightarrow\ \forall v \in V(G),\ f(\sigma \cdot G)_{\sigma(v)} = f(G)_v \qquad \text{(all orbits = all nodes)}$$

$$\Rightarrow\ \forall v \in V(G),\ (\sigma^{-1} \cdot f(\sigma \cdot G))_v = f(G)_v \qquad \text{(permute vector)}$$

$$\Rightarrow\ \forall v \in V(G),\ (\sigma \cdot \sigma^{-1} \cdot f(\sigma \cdot G))_v = (\sigma \cdot f(G))_v$$

$$\Rightarrow\ \forall v \in V(G),\ f(\sigma \cdot G)_v = (\sigma \cdot f(G))_v \qquad \text{(inverse permutation)}$$

$$\Rightarrow\ f(\sigma \cdot G) = \sigma \cdot f(G)$$

□

A.1.4 Theorem 8

Theorem. *If orbit-1-WL assigns different colourings to $u, v \in V(G)$, then u, v are not similar.*

Proof. We prove this by induction on n, the number of iterations the orbit-1-WL algorithm has taken. The orbit-1-WL algorithm is given formally in Appendix A.2.1. The stated theorem thus becomes: "if orbit-1-WL assigns different colourings to nodes $u, v \in G$ after n iterations, then they are not similar".

Base case: n=0

If u, v have different colourings after no iterations of orbit-1-WL, then they have different labels in the original graph. This implies that u and v are not similar, since any automorphism must be label-preserving.

Inductive step: assume true for n

Let $C_n : V(G) \to C$ be the colouring of G after n iterations of orbit-1-WL and similarly C_{n+1} after $n+1$ iterations. We have nodes $u, v \in G$ s.t. $C_{n+1}(u) \neq C_{n+1}(v)$, and aim to prove that u and v are not similar.

Case 1: $C_n(u) \neq C_n(v)$

Then as assumed in the inductive step, u and v are not similar.

Case 2: $C_n(u) = C_n(v)$

Let $N(v) := \{w \in V(G) \mid (v, w) \in E(G)\}$ and $N(u) := \{w \in V(G) \mid (u, w) \in E(G)\}$ be the respective neighbourhoods of v, w in G.

$C_{n+1}(u)$ is computed by $C_{n+1}(u) := h(C_n(u), \{C_n(w) \mid w \in N(u)\})$, where h is the WL hash function. But then since $C_n(u) = C_n(v)$ and $C_{n+1}(u) \neq C_{n+1}(v)$, $\{C_n(w) \mid w \in N(u)\} \neq \{C_n(w) \mid w \in N(v)\}$.

If $|N(u)| \neq |N(v)|$, then u and v are trivially not similar, since any automorphism must be edge-preserving. So assume that $|N(u)| = |N(v)|$. The above set inequality thus implies that there does not exist a bijection $\pi : N(u) \to N(v)$ such that $\forall w \in N(u),\ C_n(w) = C_n(\pi(w))$.

Assume to the contrary that u and v are similar.

Then there exists an automorphism $\sigma : G \to G$ such that $\sigma(u) = v$. This implies that $\forall w_1 \in N(u)\ \exists w_2 \in N(v)$ such that $\sigma(w_1) = w_2$, otherwise σ is not edge-preserving. So we have $|N(u)|$ pairs (w_1, w_2) of similar nodes, where each $w_1 \in N(u),\ w_2 \in N(v)$. Define $V_\sigma := \{(w_1, w_2) \mid w_1 \in N(u),\ w_2 := \sigma(w_1)\}$ to be the set containing those pairs.

If for some $(w_1, w_2) \in V_\sigma$, $C_n(w_1) \neq C_n(w_2)$, then w_1 and w_2 are not similar by assumption in the inductive step, which creates a contradiction.

So instead assume that $\forall (w_1, w_2) \in V_\sigma,\ C_n(w_1) = C_n(w_2)$. This implies that $\forall w \in N(u),\ C_n(w) = C_n(\sigma(w))$. Thus, when restricted to $N(u)$, σ induces a bijection $\pi : N(u) \to N(v)$ such that $\forall w \in N(u),\ C_n(w) = C_n(\pi(w))$. But this is a contradiction, since it was shown above that no such bijection can exist.

Conclusion

Thus, either way, by contradiction, we can conclude that u and v are not similar. This completes the inductive step, as we have shown that the hypothesis holds for C_{n+1}. Thus, the theorem is true $\forall n$ by induction.

□

A.1.5 Theorem 9

Theorem. *The expressivity of the models considered in this chapter is as follows, where for a class of models X, the claim "X is not equivariant" means that "there exists $f \in X$ such that f is not equivariant":*

1. *Unique-ID-MPNNs are not orbit-equivariant and, for any fixed n, can approximate any node-labelling function f that assigns a real value to each node, defined on the set G_n of graphs with $\leq n$ nodes.*

2. *RNI-MPNNs are equivariant in expectation and, for any fixed n, can approximate (with probability arbitrarily close to 1) any equivariant function f that assigns a real value to each node, defined on the set G_n of graphs with $\leq n$ nodes [2, 191]. They can approximate some functions that are non-equivariant and orbit-equivariant with probability arbitrarily close to 1, but there exist RNI-MPNNs which, with probability arbitrarily close to 1, are not orbit-equivariant.*

3. *Orbit-Indiv-MPNNs are not equivariant but are orbit-equivariant on graphs whose orbits are distinguishable by orbit-1-WL. For any $m \in \mathbb{Z}^+$, there exist Orbit-Indiv-MPNNs f with $\texttt{max-orbit}(f) > m$.*

4. *For any $m \geq 2$, m-Orbit-Transform-MPNNs f are not equivariant but are orbit-equivariant on graphs whose orbits are distinguishable by orbit-1-WL. They have $\texttt{max-orbit}(f) \leq m$ and there exist m-Orbit-Transform-MPNNs f with $\texttt{max-orbit}(f) = m$.*

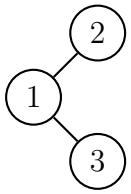
Proof. The theorem follows from the four following Lemmas (Lemma 40, Lemma 41, Lemma 42, and Lemma 43), each of which proves the expressivity of one of the models in the Theorem. □

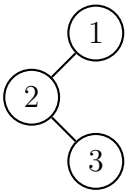
Lemma 40. *Unique-ID-MPNNs are not orbit-equivariant and, for any fixed n, can approximate any node-labelling function f that assigns a real value to each node, defined on the set G_n of graphs with $\leq n$ nodes.*

Proof. This proof has two sections, one for each of its claims.

Unique-ID-MPNNs are not orbit-equivariant.

The following is a trivial example of a Unique-ID-MPNN f that is not orbit-equivariant. Let f be a Unique-ID-MPNN defined by $f(G) = f_{\text{MPNN}}(f_{\text{unique}}(G))$, where $f_{\text{unique}}(G)_v := (v, G_v)$ deterministically appends a unique label (the unique label is v itself, since it is already an integer) to each node v. Define f_{MPNN} as an MPNN with a single layer, where f_1^{aggr} and f_1^{read} are arbitrary and $f_1^{\text{update}}(\mathbf{v}_0, _, _) := \mathbf{v}_0$. The MPNN does not do any actual message-passing, and simply keeps the original label. Then, consider the below graphs G_1, G_2, where the nodes have empty labels.

G_1: 

G_2: 

Consider the permutation $\sigma = \{(1,2),(2,1),(3,3)\}$, for which $\sigma \cdot G_1 = G_2$, and consider the orbit $r = \{1\}$ of G_1. Then $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{f(G_2)_2\}\!\} = \{\!\{2\}\!\}$ and $\{\!\{f(G_1)_v \mid v \in r\}\!\} = \{\!\{f(G_1)_1\}\!\} = \{\!\{1\}\!\}$. So there exists an orbit $r \in R(G_1)$ and permutation σ such that $\{\!\{f(\sigma \cdot G_1)_{\sigma(v)} \mid v \in r\}\!\} \neq \{\!\{f(G_1)_v \mid v \in r\}\!\}$. Hence, the Unique-ID-MPNN f is not orbit-equivariant.

Unique-ID-MPNNs can, for any fixed n, approximate any node-labelling function f that assigns a real value to each node, defined on the set G_n of graphs with $\leq n$ nodes.

We aim to prove that, for any node-labelling function defined on G_n (that produces a real value for each node), there exists a Unique-ID-MPNN that approximates f with arbitrary precision ϵ. For this proof, we rely on Theorem 4 of Morris et al. [191], which is stated as follows:

Let $n \geq 1$ and $f : G_n \to \mathbb{R}^n$ be equivariant. Then $\forall \epsilon > 0$, there exists a GNN with unique node IDs that ϵ-approximates f.

Note that their definition of a "GNN with unique node IDs" is identical to Unique-ID-MPNN. G_n is defined as the set of all graphs with at most n nodes, so n can be made arbitrarily large so as to include any particular graph. They also use a codomain of $\mathbb{R}$ for each output node feature, but note that the result extends to real vectors of arbitrary size. In their proof, they show that there exists a Unique-ID-MPNN $g(f_{\text{unique}}(G))$ that with arbitrary precision yields a final label of $\mathbf{v}_L = (\mathsf{unique}_v, r_v)$ for each node v, where r_v uniquely identifies the graph G in G_n (up to isomorphism) and unique_v is the original unique ID given to v.

However, this only identifies the graph up to isomorphism. So, we first prove that for any graphs G, H with $V(G) = V(H)$, $f_{\text{unique}}(G) \equiv f_{\text{unique}}(H)$ if and only if $G = H$. If $G = H$, then trivially $f_{\text{unique}}(G) = f_{\text{unique}}(H)$ and thus $f_{\text{unique}}(G) \equiv f_{\text{unique}}(H)$. If instead $f_{\text{unique}}(G) \equiv f_{\text{unique}}(H)$, then there exists a permutation σ on $V(G)$ such that $\sigma \cdot f_{\text{unique}}(G) = f_{\text{unique}}(H)$. The permutation maps each $u \in V(G)$ (with unique ID unique_u) to the vertex $u' \in V(H)$ with the same unique ID, since the permutation is label-preserving. But since f_{unique} is deterministic, $u = u'$. Thus, $\forall v, w \in V(G),\ (v, w) \in E(G) \iff (\sigma(v), \sigma(w)) \in E(H) \iff (v, w) \in E(H)$, and $G_v = H_w$. So $G = H$. This proves that identifying $f_{\text{unique}}(G)$ up to isomorphism is equivalent to identifying G up to equality.

Thus, define an MPNN f_{copy} with a single layer that simply copies the unique label: f_1^{aggr} and f_1^{read} are arbitrary and

$$f_1^{\text{update}}((\mathsf{unique}_v, G_v), _, _) := (\mathsf{unique}_v, \mathsf{unique}_v, G_v).$$

The MPNN will take the output of f_{unique} as input and does not do any actual message-passing: it simply copies the unique ID portion of the original label. But then the Unique-ID-MPNN $g(f_{\text{copy}}(f_{\text{unique}}))$ yields a final label of $\mathbf{v}_L = (\mathsf{unique}_v, r_v)$ for each node v, where r_v uniquely identifies the graph $f_{\text{unique}}(G)$ up to isomorphism and unique_v is the original unique ID given to v. But since r_v identifies $f_{\text{unique}}(G)$ up to isomorphism, it identifies G up to equality, as proven above.

But then we can define a mapping f^{node} such that for each labelled graph G, $v \in G$, $f^{\text{node}}(\mathsf{unique}_v, r_v) = f(G)_v$. This mapping can be approximated by an MLP with arbitrary precision and appended onto the Unique-ID-MPNN $g(f_{\text{copy}}(f_{\text{unique}}))$ as a final update layer. This yields a Unique-ID-MPNN that approximates f with arbitrary precision. □

Lemma 41. *RNI-MPNNs are equivariant in expectation and, for any fixed n, can approximate (with probability arbitrarily close to 1) any equivariant function f that assigns a real value to each node, defined on the set G_n of graphs with $\leq n$ nodes [2, 191]. They can approximate some functions that are non-equivariant and orbit-equivariant with probability arbitrarily close to 1, but there exist RNI-MPNNs which, with probability arbitrarily close to 1, are not orbit-equivariant.*

Proof. This proof has three sections, one for each of its claims.

RNI-MPNNs are equivariant in expectation and, for any fixed n, can approximate (with probability arbitrarily close to 1) any equivariant function f that assigns a real value to each node, defined on the set G_n of graphs with $\leq n$ nodes.

Note that since RNI-MPNNs are functions that include randomness, we can no longer discuss them simply approximating functions. Instead, we need to refer to them approximating functions with a given probability.

To formally define "with probability arbitrarily close to 1": Abboud et al. [2] say that a randomized function X that associates with every graph $G \in G_n$ a random variable $X(G)$ is an (ϵ, δ)-*approximation* of f if for all $G \in G_n$ it holds that $\Pr(|f(G) - X(G)| \leq \epsilon) \geq 1 - \delta$. Note that an RNI-MPNN $\mathcal{M}$ computes such functions X. If X is computed by $\mathcal{M}$, we say that $\mathcal{M}$ (ϵ, δ)-*approximates* f. If for all $\epsilon, \delta > 0$, there is an RNI-MPNN that (ϵ, δ)-*approximates* f, then we abbreviate this by saying that the RNI-MPNN "approximates f with probability arbitrarily close to 1".

We extend this to node-labelling functions as follows. Let X be a randomized node-labelling function that associates with every graph $G \in G_n$ and node $v \in V(G)$

a random variable $X(G)_v$. Then X is an (ϵ, δ)-*approximation* of f if for all $G \in G_n$ and $v \in V(G)$ it holds that $\Pr(|f(G)_v - X(G)_v| \leq \epsilon) \geq 1 - \delta$. Note that an RNI-MPNN $\mathcal{M}$ computes such functions X. If X is computed by $\mathcal{M}$, we say that $\mathcal{M}$ (ϵ, δ)-*approximates* f. If for all $\epsilon, \delta > 0$, there is an RNI-MPNN that (ϵ, δ)-*approximates* f, then we abbreviate this by saying that the RNI-MPNN "approximates f with probability arbitrarily close to 1".

Abboud et al. [2] prove that RNI-MPNNs are equivariant in expectation and that they can approximate any invariant graph function with probability arbitrarily close to 1 (in their Theorem 1). Morris et al. [191] extend the universality results of Abboud et al. [2] and prove that they can approximate any equivariant node-labelling function with probability arbitrarily close to 1 (in their Theorem 2).

RNI-MPNNs can approximate some functions that are non-equivariant and orbit-equivariant with probability arbitrarily close to 1.

We consider an example of a non-equivariant and orbit-equivariant function that RNI-MPNNs can approximate with probability arbitrarily close to 1. This is done to demonstrate their ability to go beyond equivariance and into orbit-equivariance, but it remains an open question as to whether RNI-MPNNs can approximate any orbit-equivariant function.

Define a node-labelling function f as follows. For a labelled graph G, $f(G)_v := r_{G,v}$, where $r_{G,v}$ is some value that uniquely identifies the orbit of v in G up to isomorphism. Formally: $r_{G,v} = r_{H,w} \iff$ there exists an isomorphism $\alpha : G \to H$ (the graphs containing nodes v and w, respectively) such that $\alpha(v) = w$. Thus, for any permutation σ, $r_{G,v} = r_{\sigma \cdot G, \sigma(v)}$, since σ is an isomorphism from G to $\sigma \cdot G$.

This function f is equivariant, since for all labelled graphs G, all permutations σ on $V(G)$ and all nodes $v \in V(G)$, we have $f(\sigma \cdot G)_v = r_{\sigma \cdot G, v} = r_{G, \sigma^{-1}(v)} = f(G)_{\sigma^{-1}(v)} = (\sigma \cdot f(G))_v$. Since f is equivariant, Theorem 2 of Morris et al. [191] proves that it can be approximated with probability arbitrarily close to 1 by an RNI-MPNN g.

Now consider another node-labelling function f_{copy}, defined on the graph that has had random noise appended, such that $f_{\text{copy}}(G)_v = (\eta_v, r_{G,v})$. This is the same as f, except it also copies the random noise η_v given to node v in the input graph G. f_{copy} is also equivariant, as a trivial consequence of f being equivariant, and can be approximated by extending an RNI-MPNN g to copy the random noise for each node.

Now fix some graph H and node $w \in V(H)$ such that the orbit of w has size at least 2. Then consider a node-labelling function $f_{\max}$, defined on the output of f_{copy}, such that $f_{\max}(G)_v := 1$ if $r_{G,v} = r_{H,w}$ and η_v is the maximum of η_u for all $u \in V(G)$ such that $r_{G,u} = r_{H,w}$. Otherwise, $f_{\max}(G)_v := 0$.

The node-labelling function h defined by $h(G) := f_{\max}(f_{\text{copy}}(G))$ can be approximated by extending the previously defined RNI-MPNN g. Append one final layer to the RNI-MPNN, defined by $f^{\text{read}}(\{\!\{(\eta_u, r_{G,u}) \mid u \in V(G)\}\!\} := \max(\{\!\{\eta_u \mid u \in V(G) \text{ and } r_{G,u} = r_{H,w}\}\!\})$. This function is permutation-invariant and so is a valid choice for f^{read}. f^{update} is defined by $f^{\text{update}}((\eta_v, r_{G,v}), _, \eta_u) := 1$ if $r_{G,v} = r_{H,w}$ and $\eta_v = \eta_u$, and $f^{\text{update}}((\eta_v, r_{G,v}), _, \eta_u) := 0$ otherwise. The resulting RNI-MPNN approximates h with probability arbitrarily close to 1, since a unique value in $\{\!\{\eta_u \mid u \in V(G) \text{ and } r_{G,u} = r_{H,w}\}\!\}$ is maximal with probability arbitrarily close to 1, because the random values are sampled from a continuous interval.

It now remains to be shown that the node-labelling function h is orbit-equivariant. To prove that h is orbit-equivariant, let G be a labelled graph, σ be a permutation on G, and $r \in R(G)$ an orbit. Recall that H was fixed above, to be some graph with node $w \in V(H)$ such that the orbit of w has size at least 2.

Case 1: $r_{\sigma \cdot G, \sigma(u)} = r_{H,w}$ for some node $u \in r$. Then:

$$\{\!\{h(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{f_{\max}(f_{\text{copy}}(\sigma \cdot G))_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{1\}\!\} \cup \{\!\{0\}\!\}^{|r|-1}$$

Also, since $r_{G,u} = r_{\sigma \cdot G, \sigma(u)}$ (proven when f is defined), we can derive that $r_{G,u} = r_{H,w}$. Thus:

$$\{\!\{h(G)_v \mid v \in r\}\!\} = \{\!\{f_{\max}(f_{\text{copy}}(G))_v \mid v \in r\}\!\} = \{\!\{1\}\!\} \cup \{\!\{0\}\!\}^{|r|-1}$$

Case 2: $r_{\sigma\cdot G,\sigma(u)} \neq r_{H,w}$ for every node $u \in r$. Then:

$$\{\!\{h(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{f_{\max}(f_{\text{copy}}(\sigma \cdot G))_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{0\}\!\}^{|r|}$$

Again, since $r_{G,u} = r_{\sigma\cdot G,\sigma(u)}$ for every $u \in V(G)$, we can derive that $r_{G,u} \neq r_{H,w}$ for every $u \in r$. Thus:

$$\{\!\{h(G)_v \mid v \in r\}\!\} = \{\!\{f_{\max}(f_{\text{copy}}(G))_v \mid v \in r\}\!\} = \{\!\{0\}\!\}^{|r|}$$

Hence, in either case, $\{\!\{h(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{h(G)_v \mid v \in r\}\!\}$, so h is orbit-equivariant. However, it is non-equivariant, since $\{\!\{h(G)_v \mid v \in r\}\!\} = \{\!\{1\}\!\} \cup \{\!\{0\}\!\}^{|r|-1}$ when $G = H$ and $w \in r$ (note that $|r| \geq 2$ by the definition of w), which shows that the function produces different values for similar nodes, violating the property of equivariance proved in Proposition 3.

There exist RNI-MPNNs which, with probability arbitrarily close to 1, are not orbit-equivariant.

Consider an RNI-MPNN f with a single layer. The readout function f_1^{read} is defined by $f_1^{\text{read}}(\{\!\{(\eta_w, \mathbf{w}_0) \mid w \in V(G)\}\!\}) := \max(\{\!\{\eta_w \mid w \in V(G)\}\!\})$, where η_w is the random noise given to node w. The update function f_1^{update}, is defined by $f_1^{\text{update}}((\eta_v, \mathbf{v}_0), _, \eta_{\max}) := 1$ if $\eta_v = \eta_{\max}$, and 0 otherwise, where $\eta_{\max}$ is the output of f_1^{read}.

Thus, for a graph G with n nodes, $f(G)_w = 1$ for some $w \in V(G)$ (chosen uniformly at random), and $f(G)_u = 0$ for all $u \in V(G)$ such that $u \neq w$.

As n becomes large for graphs with small orbit sizes, the probability that $f(G)_w = 1$ and $f(G)_u = 1$ for any $w, u \in V(G)$ such that w, u are in the same orbit of G, across two separate applications of f, becomes arbitrarily small.

Thus also, the probability that for all labelled graphs G, permutations σ on $V(G)$, and orbits $r \in R(G)$, it holds that $\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} = \{\!\{f(G)_v \mid v \in r\}\!\}$

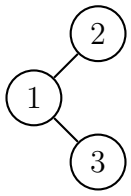
becomes arbitrarily small. So the RNI-MPNN f is not orbit-equivariant with probability arbitrarily close to 1. □

Lemma 42. *Orbit-Indiv-MPNNs are not equivariant but are orbit-equivariant on graphs whose orbits are distinguishable by orbit-1-WL. For any $m \in \mathbb{Z}^+$, there exist Orbit-Indiv-MPNNs f with* $\texttt{max-orbit}(f) > m$.

Proof. This proof has four sections, one for each of its claims and one demonstrating the consequences of having an input graph whose orbits are not distinguishable by orbit-1-WL.

Orbit-Indiv-MPNNs are not equivariant.

We consider a simple example of an Orbit-Indiv-MPNN f that is not equivariant, defined on a graph G by $f(G) := f_{\text{mlp}}(f_{\text{orbit_indiv}}(f_{\text{MPNN}}(G)))$. f_{MPNN} is defined by $f_{\text{MPNN}}(G)_v := 0$. Define f_{mlp} to be the identity function. Let G be the below graph.



Then $f(G)$ assigns the labels $(1,0),(1,0),(2,0)$ to each of the respective nodes. Nodes 2 and 3 are similar in G. But $f(G)_2 = (1,0)$ and $f(G)_2 = (2,0)$, breaking the property of equivariance proved in Proposition 3. Thus, Orbit-Indiv-MPNN f is not equivariant.

For any $m \in \mathbb{Z}^+$, there exist Orbit-Indiv-MPNNs f with $\texttt{max-orbit}(f) > m$.

Let $m \in \mathbb{Z}^+$ and $\mathcal{G} = \{C_{m+1}\}$ be the set containing the single-cycle graph with $m+1$ nodes. Let f be an Orbit-Indiv-MPNN defined on $\mathcal{G}$ by $f(G) := f_{\text{mlp}}(f_{\text{orbit_indiv}}(f_{\text{MPNN}}(G)))$. f_{MPNN} is defined by $f_{\text{MPNN}}(G)_v := 0$. Define f_{mlp} to be the identity function.

Then for $G = C_{m+1} \in \mathcal{G}$, orbit-1-WL correctly identifies all nodes as belonging to the same orbit, so $f(G) = ((1,0),(2,0),...,(m+1,0))$. This has $m+1$ different outputs within the same orbit. But $C_{m+1} \in \mathcal{G}$, so $\texttt{max-orbit}(f) = m+1 > m$.

Orbit-Indiv-MPNNs are orbit-equivariant on graphs whose orbits are distinguishable by orbit-1-WL.

Let f be an Orbit-Indiv-MPNN defined by $f(G) := f_{\text{mlp}}(f_{\text{orbit_indiv}}(f_{\text{MPNN}}(G)))$ and $\mathcal{G}$ a set of all graphs whose orbits are distinguishable by orbit-1-WL. Then since $\forall r \in R(G),\ \forall v, w \in r,\ f_{\text{MPNN}}(G)_v = f_{\text{MPNN}}(G)_w$, we have the following for all labelled graphs $G \in \mathcal{G}$, permutations σ on $V(G)$, and orbits $r \in R(G)$:

$$\{\!\{f(G)_v \mid v \in r\}\!\} = \{\!\{f_{\text{mlp}}((u, f_{\text{MPNN}}(G)_v)) \mid v \in r,\ u \in \{1, 2, ..., |r|\}\}\!\}.$$

Also, $\forall r \in R(G),\ \forall v, w \in r,\ f_{\text{MPNN}}(\sigma \cdot G)_{\sigma(v)} = f_{\text{MPNN}}(\sigma \cdot G)_{\sigma(w)}$, since v, w similar in G implies that $\sigma(v), \sigma(w)$ are similar in $\sigma \cdot G$. So we have, for all labelled graphs G, permutations σ on $V(G)$, and orbits $r \in R(G)$:

$$\begin{aligned}
&\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} \\
=&\{\!\{f_{\text{mlp}}((u, f_{\text{MPNN}}(\sigma \cdot G)_{\sigma(v)})) \mid v \in r,\ u \in \{1, 2, ..., |r|\}\}\!\} \\
&\qquad \text{(since } f_{\text{mlp}} \text{ is element-wise)} \\
=&\{\!\{f_{\text{mlp}}((u, (\sigma \cdot f_{\text{MPNN}}(G))_{\sigma(v)})) \mid v \in r,\ u \in \{1, 2, ..., |r|\}\}\!\} \\
&\qquad \text{(since } f_{\text{MPNN}} \text{ is equivariant)} \\
=&\{\!\{f_{\text{mlp}}((u, f_{\text{MPNN}}(G)_v)) \mid v \in r,\ u \in \{1, 2, ..., |r|\}\}\!\} \\
=&\{\!\{f(G)_v \mid v \in r\}\!\} \\
&\qquad \text{(by definition)}
\end{aligned}$$

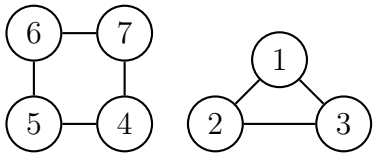
Thus, $\{\!\{f(G)_v \mid v \in r\}\!\} = \{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\}$, so Orbit-Indiv-MPNN f is orbit-equivariant.

The expressivity of Orbit-Indiv-MPNNs for distinguishing nodes into different orbits is limited by orbit-1-WL.

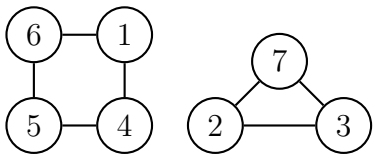
First, note that f_{MPNN} has the same expressive power as orbit-1-WL for distinguishing nodes into different orbits and that both f_{MPNN} and orbit-1-WL are explicitly used in computing the output of an Orbit-Indiv-MPNN. Thus, Orbit-Indiv-MPNNs are limited by the expressivity of orbit-1-WL. We will briefly unpack the consequences of this here.

The contrapositive of Theorem 8 is: if $v, w \in V(G)$ are similar, then orbit-1-WL assigns the same colourings to u, v. Thus, the following statement (used in the above proof that Orbit-Indiv-MPNNs are orbit-equivariant) still holds true, despite the limited expressive power of f_{MPNN}: $\forall r \in R(G),\ v, w \in r,\ f_{\text{MPNN}}(G)_v = f_{\text{MPNN}}(G)_w$.

In the above proof, it was assumed that for every orbit $r \in R(G)$, orbit-1-WL correctly identifies the nodes in r and thus that $f_{\text{orbit_indiv}}$ distributes the labels $\{1, 2, ..., |r|\}$ among the nodes of the orbit. However, consider the following graph for which orbit-1-WL fails to identify the orbits:



The actual orbits are $r_1 = \{1, 2, 3\},\ r_2 = \{4, 5, 6, 7\}$, but orbit-1-WL incorrectly identifies them as $r_3 = \{1, 2, 3, 4, 5, 6, 7\}$. Consider Orbit-Indiv-MPNN f, defined on a graph G by $f(G) := f_{\text{mlp}}(f_{\text{orbit_indiv}}(f_{\text{MPNN}}(G)))$. f_{MPNN} is defined by $f_{\text{MPNN}}(G)_v := 0$. Define f_{mlp} to be the identity function. Then $f(G)$ produces the following node labels: $(1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0)$. Let σ be the permutation $\{(1, 7), (7, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6)\}$. Then $\sigma \cdot G$ is the following graph:



Then $\{\!\{f(G)_v \mid v \in r_1\}\!\} = \{\!\{(1, 0), (2, 0), (3, 0)\}\!\}$, but since $f_{\text{orbit_indiv}}$ uses the orbit-1-WL orbits, $\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r_1\}\!\} = \{\!\{(7, 0), (2, 0), (3, 0)\}\!\}$. So f is not

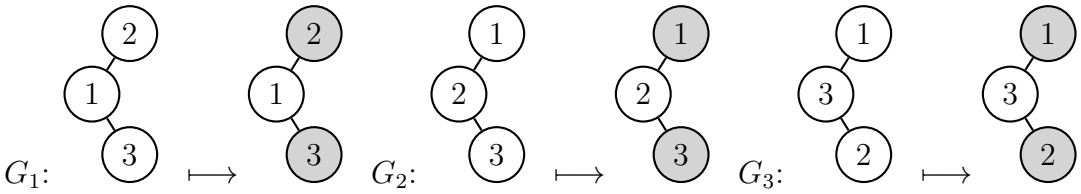
orbit-equivariant for this graph, due to the expressivity of Orbit-Indiv-MPNNs for distinguishing nodes into different orbits being limited by orbit-1-WL. The same problem occurs on any input graph where the orbits cannot be correctly identified by orbit-1-WL. This theoretical issue can be alleviated by using a more expressive (and likely slower) orbit-identification algorithm than orbit-1-WL, or by only operating on graphs whose nodes can be distinguished by orbit-1-WL. □

Lemma 43. *For any $m \geq 2$, m-Orbit-Transform-MPNNs f are not equivariant but are orbit-equivariant on graphs whose orbits are distinguishable by orbit-1-WL. They have* `max-orbit`$(f) \leq m$ *and there exist m-Orbit-Transform-MPNNs f that satisfy* `max-orbit`$(f) = m$.

Proof. This proof has four sections, one for each of its claims and one demonstrating the consequences of having an input graph whose orbits are not distinguishable by orbit-1-WL.

m-Orbit-Transform-MPNNs are not equivariant.

We consider a simple example of a 2-Orbit-Transform-MPNN f that is not equivariant, defined on a graph G by $f(G) := f_{\text{transform}}(f_{\text{MPNN}}(G))$. In this case, $o := 1$ (the number of final output channels). f_{MPNN} is defined as stated below for input graphs G_1, G_2, G_3, and maps all the node labels in each other graph G to $((0, 99), (0, 99))$. In the diagram below, the colours of output nodes represent their features, with *white* $= ((0, 99), (0, 99))$ and *grey* $= ((1, -99), (2, -99))$. The large final channel (99) represents not swapping from the default (i.e. if the final channel is the largest, then the default should not be swapped out for the value in this partition).



Then $f(G_1)$ produces respective node labels of $0, 1, 2$. Nodes 2 and 3 are similar in G_1. But $f(G_1)_2 = 1$ and $f(G_1)_3 = 2$, breaking the property of equivariance

proved in Proposition 3. Thus, 2-Orbit-Transform-MPNN f is not equivariant. A similar example for any $m > 2$ shows that m-Orbit-Transform-MPNNs are not equivariant.

m-Orbit-Transform-MPNNs f have $\texttt{max-orbit}(f) \leq m$ and there exist m-Orbit-Transform-MPNNs f with $\texttt{max-orbit}(f) = m$.

Since we aim to convey the `max-orbit` of an entire class of models instead of just a single function, we prove that there exist specific m-Orbit-Transform-MPNNs f that have $\texttt{max-orbit}(f) = m$ and that there are no m-Orbit-Transform-MPNNs f with $\texttt{max-orbit}(f) > m$. Note that there do exist trivial m-Orbit-Transform-MPNNs f such that $\texttt{max-orbit}(f) < m$.

We prove this in two parts. First, we prove that for any m-Orbit-Transform-MPNN f, $\texttt{max-orbit}(f) \leq m$. This follows from there being exactly m partitions in the output of $f_{\text{MPNN}}(G)$. For each graph G and orbit $r \in R(G)$, step (1) of $f_{\text{transform}}$ gives a single label to all nodes in the orbit. Then step (2) adds a new label to the orbit at most $\min(m, |r|) - 1$ times, meaning there are at most m different labels in the orbit after step (2) is finished.

Second, we give an example of an m-Orbit-Transform-MPNN f that has $\texttt{max-orbit}(f) = m$. f_{MPNN} is defined by $f(G)_v = ((1, -99), (2, -99), ..., (m, -99))$ for each node $v \in V(G)$. Consider the cycle graph $H := C_m$ (a single cycle of m nodes). It has a single orbit, $r = \{1, 2, ..., m\}$, which is identifiable by orbit-1-WL. Thus, $f(H)$ assigns the respective values of $1, 2, ..., m$ to the node labels, so there are m unique values in $\{\!\{f(H)_v \mid v \in r\}\!\}$.

m-Orbit-Transform-MPNNs are orbit-equivariant on graphs whose orbits are distinguishable by orbit-1-WL.

Let f be an m-Orbit-Transform-MPNN defined by $f(G) := f_{\text{transform}}(f_{\text{MPNN}}(G))$ and $\mathcal{G}$ a set of all graphs whose orbits are distinguishable by orbit-1-WL. Then since $\forall r \in R(G),\ \forall v, w \in r,\ f_{\text{MPNN}}(G)_v = f_{\text{MPNN}}(G)_w$, we have, for all labelled graphs $G \in \mathcal{G}$, permutations σ on $V(G)$, and orbits $r \in R(G)$:

$$\{\!\{f(G)_v \mid v \in r\}\!\} = f_{\text{orbit_transform}}(f_{\text{MPNN}}(G)_w) \quad \text{(for any } w \in r)$$

where $f_{\text{orbit_transform}}$ is a function that yields the multiset of outputs of $f_{\text{transform}}$ for the orbit containing w. These outputs are all set to a single value by step (1) of $f_{\text{transform}}$, and are then changed by each iteration of step (2). Also, $\forall r \in R(G),\ \forall v, w \in r,\ f_{\text{MPNN}}(\sigma \cdot G)_{\sigma(v)} = f_{\text{MPNN}}(\sigma \cdot G)_{\sigma(w)}$, since v, w similar in G implies that $\sigma(v), \sigma(w)$ are similar in $\sigma \cdot G$. So we have, for all labelled graphs G, permutations σ on $V(G)$, and orbits $r \in R(G)$:

$$\begin{aligned}
\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\} &= f_{\text{orbit_transform}}(f_{\text{MPNN}}(\sigma \cdot G)_{\sigma(w)}) \\
&\qquad \text{(for any } w \in r) \\
&= f_{\text{orbit_transform}}((\sigma \cdot f_{\text{MPNN}}(G))_{\sigma(w)}) \\
&\qquad \text{(since } f_{\text{MPNN}} \text{ is equivariant)} \\
&= f_{\text{orbit_transform}}(f_{\text{MPNN}}(G)_w)
\end{aligned}$$

Thus, $\{\!\{f(G)_v \mid v \in r\}\!\} = \{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r\}\!\}$, so m-Orbit-Transform-MPNN f is orbit-equivariant.

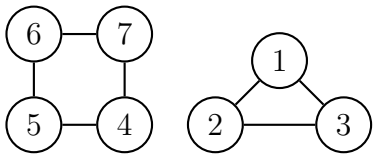
The expressivity of m-Orbit-Transform-MPNNs for distinguishing nodes into different orbits is limited by orbit-1-WL.

First, note that f_{MPNN} has the same expressive power as orbit-1-WL for distinguishing nodes into different orbits and that both f_{MPNN} and orbit-1-WL are explicitly used in computing the output of an m-Orbit-Transform-MPNN. Thus, m-Orbit-Transform-MPNNs are limited by the expressivity of orbit-1-WL. We will briefly unpack the consequences of this here.

The contrapositive of Theorem 8 is: if $v, w \in V(G)$ are similar, then orbit-1-WL assigns the same colourings to u, v. Thus, the following statement (used in the above proof that m-Orbit-Transform-MPNNs are orbit-equivariant) still holds true,

despite the limited expressive power of f_{MPNN}: $\forall r \in R(G),\ \forall v, w \in r,\ f_{\text{MPNN}}(G)_v = f_{\text{MPNN}}(G)_w$.

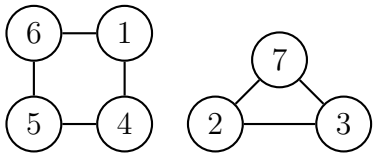
In the above proof, it was assumed that for every orbit $r \in R(G)$, orbit-1-WL correctly identifies the nodes in r and thus that $f_{\text{orbit_transform}}$ yields the multiset of outputs of $f_{\text{transform}}$ for the orbit containing w. However, consider the following graph for which orbit-1-WL fails to identify the orbits:



The actual orbits are $r_1 = \{1, 2, 3\},\ r_2 = \{4, 5, 6, 7\}$, but orbit-1-WL incorrectly identifies them as $r_3 = \{1, 2, 3, 4, 5, 6, 7\}$. Consider 7-Orbit-Transform-MPNN f, defined on a graph G by $f(G) := f_{\text{transform}}(f_{\text{MPNN}}(G))$. f_{MPNN} is defined by $f_{\text{MPNN}}(G)_v := ((1, -99), (2, -99), (3, -99), (4, -99), (5, -99), (6, -99), (7, -99))$ for every node $v \in V(G)$. Then $f(G)$ produces the following respective labels for each node: $1, 2, 3, 4, 5, 6, 7$. Let σ be the permutation

$$\{(1, 7), (7, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6)\}.$$

Then $\sigma \cdot G$ is the following graph:



Then $\{\!\{f(G)_v \mid v \in r_1\}\!\} = \{\!\{1, 2, 3\}\!\}$, but since $f_{\text{transform}}$ uses the orbit-1-WL orbits, $\{\!\{f(\sigma \cdot G)_{\sigma(v)} \mid v \in r_1\}\!\} = \{\!\{7, 2, 3\}\!\}$. So f is not orbit-equivariant for this graph, due to the expressivity of m-Orbit-Transform-MPNNs for distinguishing nodes into different orbits being limited by orbit-1-WL. The same problem occurs on any input graph where the orbits cannot be correctly identified by orbit-1-WL. This theoretical issue can be alleviated by using a more expressive (and likely slower) orbit-identification algorithm than orbit-1-WL, or by only operating on graphs whose nodes can be distinguished by orbit-1-WL. □

A.2 Orbit-1-WL

A.2.1 Definition

The 1-WL algorithm [287] can be formalized algorithmically as follows:

1. Initialize $C_0 : V(G) \to C$ by $C_0(v) = c_1$ for each node $v \in V(G)$, where C is a codomain of colours and $c_1 \in C$. C_0 is a node colouring function at iteration 0 of the algorithm.

2. At each iteration n of the algorithm (starting from $n = 0$), we define a new colouring function C_{n+1}. For each node $v \in V(G)$, set $C_{n+1}(v) := h(C_n(v), \{\!\{C_n(w) \mid (v, w) \in E(G)\}\!\})$, where h is a hash function.

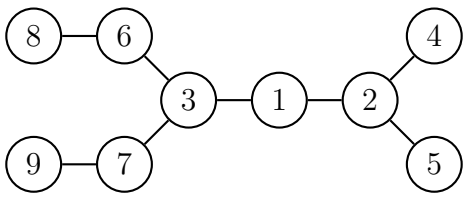
3. At each iteration n of the algorithm, also compute a partition P_{n+1} of $V(G)$, where for any nodes $v, w \in V(G)$, $v, w \in Q$ for any $Q \in P$ if and only if $C_{n+1}(v) = C_{n+1}(w)$. In other words, the nodes are partitioned by their colours.

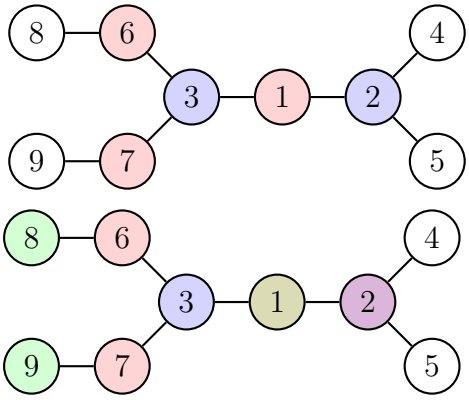
4. If at any point, $P_{n+1} = P_n$, the algorithm halts and returns $h(\{\!\{C_n(w) \mid w \in V(G)\}\!\})$.

Orbit-1-WL operates in the exact same way, except that when the algorithm halts, it returns P_n instead of $h(\{\!\{C_n(w) \mid w \in V(G)\}\!\})$. P_n represents a partition of nodes into their orbits by orbit-1-WL.

A.2.2 Successful Examples

As an example, here is orbit-1-WL identifying the orbits of a graph:





This yields orbits of $\{\{1\},\{2\},\{3\},\{4,5\},\{6,7\},\{8,9\}\}$, so orbit-1-WL has correctly identified the orbits. As another example, consider the following two graphs, G_1 and G_2:

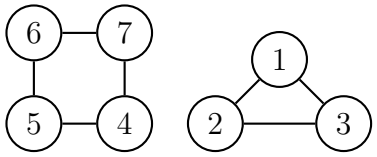


Despite 1-WL being unable to distinguish between these two graphs, orbit-1-WL correctly identifies the orbits for both (all nodes are in the same orbit). This illustrates why it is important to distinguish between 1-WL and orbit-1-WL, since there are cases where 1-WL fails and orbit-1-WL succeeds, and conversely there are cases where orbit-1-WL fails and 1-WL succeeds (as will be shown in a later section).

A.2.3 Counterexamples for Completeness

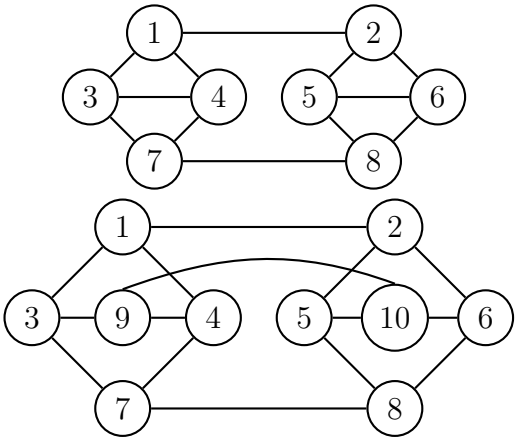
Theorem 8 proves the soundness of orbit-1-WL. However, orbit-1-WL is not complete. That is: there exists a graph G and nodes $v, w \in G$ such that orbit-1-WL assigns the same colourings to v and w, but v and w are not similar. I show several such graphs here.

In the following example, orbit-1-WL colours all the nodes the same (since the graph is 2-regular), so it identifies all nodes as being in the same orbit. However, the true orbits are $\{\{1,2,3\},\{4,5,6,7\}\}$.

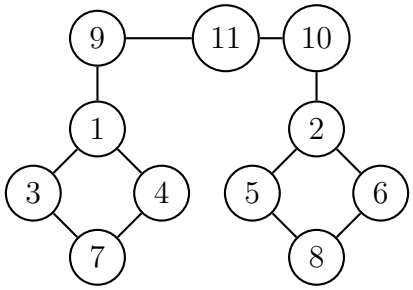


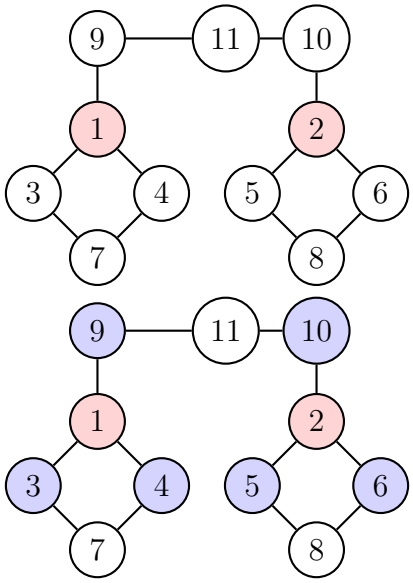
The above example can be scaled up into infinitely many other 2-regular graphs for which the same problem arises, yielding infinitely many counterexamples. It is worth noting as well that if a graph G contains a graph H as a maximally connected sub-graph (i.e. a collection of components of G), such that the orbits of H are incorrectly identified by orbit-1-WL, then the orbits of G are also incorrectly identified by orbit-1-WL. This yields another way to generate infinitely many counterexamples.

Here follows two other non-trivial 3-regular graphs (one planar and one non-planar) for which orbit-1-WL fails. Orbit-1-WL colours all nodes the same since the graphs are 3-regular. The first has actual orbits of $\{\{1, 2, 7, 8\}, \{3, 4, 5, 6\}\}$. The second has actual orbits of $\{\{1, 2, 7, 8, 9, 10\}, \{3, 4, 5, 6\}\}$.



Here follows a counterexample (found in the Alchemy dataset [52]) which is not a regular graph. It is coloured by orbit-1-WL as shown.





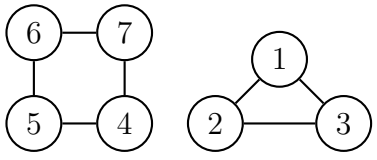
However, the true orbits of the above graph are

$$\{\{1,2\},\{3,4,5,6\},\{7,8\},\{9,10\},\{11\}\}$$

Out of the 202579 graphs in Alchemy [52], orbit-1-WL only predicts the incorrect orbits for 5. I calculated this by comparing the true orbits and orbit-1-WL orbits for every graph in the dataset.

A.2.4 Orbit-k-WL

Similar to classic WL [186], there is a strict hierarchy of orbit-WL algorithms: orbit-1-WL, orbit-3-WL, orbit-4-WL, ..., such that for all $k \geq 3$ there are graphs for which orbit-k-WL will correctly identify the orbits but orbit-$(k-1)$-WL will not. The below example shows a construction that proves this:



Neither orbit-1-WL nor orbit-2-WL correctly identifies the orbits of the graph, but orbit-3-WL does. For any $k \geq 3$, the graph consisting of a $(k+1)$-cycle and a k-cycle has its orbits correctly identified by orbit-k-WL but not by orbit-$(k-1)$-WL.

Dataset		**Bio**	**Alchemy-2**	**Alchemy-6**
Graphs		961	1000	1000
Input Features		43	6	6
Possible Targets per Node		44	2	6
Nodes	Average	10.3	10	12.1
	Min	2	10	9
	Max	15	10	22
Edges	Average	10.1	10.4	12.5
	Min	1	9	8
	Max	17	12	22

Table A.1: Properties of the new datasets proposed in the chapter, where "Bio" denotes "Bioisostere", "Alchemy-2" denotes Alchemy-Max-Orbit-2, and "Alchemy-6" denotes Alchemy-Max-Orbit-6.

A.3 Experiments

A.3.1 Computational Complexity

With respect to the computational complexity of my proposed models, Unique-ID-MPNNs and RNI-MPNNs have no extra overhead, besides yielding a larger input vector for each label in the MPNN. Orbit-Indiv-MPNNs use 1-WL, which is worst-case $O(n)$, where n is the number of nodes in the graph. Appending unique IDs and the application of the final MLP are also $O(n)$. So overall, the additional complexity of Orbit-Indiv-MPNNs is $O(n)$.

Likewise, for m-Orbit-Transform-MPNNs, 1-WL is worst-case $O(n)$. The output size of the MPNN is larger than before: it is now $m(o+1)$ instead of o, which adds additional computation to the MPNN. Apart from this, there is no additional computational cost during training. When making predictions, the transform is applied, which is worst-case $O(n)$. So overall, the additional complexity of m-Orbit-Transform-MPNNs when making predictions is $O(n)$.

A.3.2 Datasets

In Table A.1, I present a summary of the new datasets I propose in this chapter. Full details are provided in the following sections.

A.3.2.1 Bioisostere

The task in **Bioisostere** is, given an input molecule as a graph, output a label for each node which represents either keeping the atom or swapping it out for some other particular atom, in such a way that the final resulting molecule has minimal lipophilicity; as discussed in Section 4.1, this is an important factor in drug design. Solving this task requires a non-equivariant function when an atom being swapped out comes from an orbit of size ≥ 2.

To construct this dataset, I first retrieved 961 small drug molecules from ChEMBL [181]. The version of ChEMBL I used is CHEMBL32[1] and the content is licensed under the CC Attribution-ShareAlike 3.0 Unported license. I filtered to only include drug molecules[2]. From there, I filtered by "Type" to include only those with the type "Small molecule", and further filtered by "Molecular Weight" to only include molecules with a molecular weight ≤ 199. This yielded a CSV file of molecules, which was cleaned up into `chembl.smi` a file of SMILES code and molecule name pairs.

I then used MMPDB[3] [219] to compute many different bioisosteres for each molecule that each differ by at most one atom from the original molecule.

```
mmpdb fragment chembl.smi -o chembl.fragments --num-cuts 1
```

was used to compute the fragments of all the molecules from CHEMBL that can be created by cutting a single non-ring bond.

```
mmpdb index chembl.fragments -o chembl.mmpdb
```

was then used to group fragments with the same R-groups into a database. For each molecule with SMILE code "input_smile",

```
mmpdb transform --smiles input_smile chembl.mmpdb
```

was used to compute a list of bioisosteres of the molecule. For each of those bioisosteres (and the original molecule), I used RDKit [220] to compute the

[1] `http://doi.org/10.6019/CHEMBL.database.32`, accessed 27 February 2026
[2] `https://www.ebi.ac.uk/chembl/g/#browse/drugs`, accessed January 2021
[3] `https://github.com/rdkit/mmpdb`, accessed January 2021

lipophilicity and selected the one that yielded the lowest value as the target for each molecule. Both MMPDB and RDKit are licensed under the BSD 3-Clause Licence.

Of these molecules, only 588 actually require some swapping out of an atom to achieve minimal lipophilicity, and only 156 require a non-equivariant function. All 961 molecules are used in the dataset. The function f required to solve the task is orbit-equivariant with $\texttt{max-orbit}(f) = 2$.

A.3.2.2 Alchemy-Max-Orbit

An **Alchemy-Max-Orbit-m** dataset is constructed using Alchemy [52, 187], a large dataset of organic molecules. The dataset constructor is given m as a hyperparameter. First, it removes all duplicate graphs from Alchemy using 1-WL as a heuristic. Next, all graphs without an orbit of size $\geq m$ are removed, where the orbits are computed using orbit-1-WL.[4] If the number of graphs remaining is larger than the desired size of the constructed dataset (1000 for this chapter), then graphs are dropped from the end of the dataset to make the dataset the desired size.

If the number of graphs remaining is instead smaller than the desired size of the constructed dataset, then new graphs are added to the dataset using two graph generators. Both generators make small changes to graphs already in the dataset, broadly preserving the structures whilst adding new non-isomorphic graphs. The constructor continues to alternate between applying each generator, with the process immediately halting when the dataset has attained the desired size. Each generator is applied to every graph that is in the dataset so far. However, before any new graph is added to the dataset, it is checked using its 1-WL hash to ensure that a graph isomorphic to it is not already in the dataset. For each graph given to it, the first generator takes every orbit of the graph, and varies all the node features within the orbit in the same way. The second generator takes every orbit

[4] I elected to use orbit-1-WL to compute the orbits when creating the Alchemy-Max-Orbit datasets, because some graphs in the dataset were too large to efficiently compute all automorphisms and thus the orbits. For consistency, I elected to compute all orbits using orbit-1-WL, rather than exactly computing the orbits of the graphs that were small enough and using orbit-1-WL for the others.

Group	Parameter	Value	Description
Model	gnn_layers	4	Number of message-passing layers in the MPNN
	gnn_hidden_size	40	Size of each hidden layer in the MPNN
	rni_channels	10	Number of channels of random noise to append
	use_cpu	True	Force use of the CPU over CUDA
Evaluation	train_eval_interval	10	Eval on train set interval in epochs
	test_eval_interval	10	Eval on test set interval in epochs
	loss_log_interval	10	How often to log the model loss in epochs
Dataset	train_on_entire_dataset	False	Whether to train on the entire dataset
	train_split	0.9	Ratio of dataset to use for the train set
	bioisostere_only_equivariant	False	Filter out non-equivariant examples from Bioisostere
	shuffle_targets_in_max_orbit	True	Shuffle targets within orbits of Alchemy-Max-Orbit
	shuffle_dataset	True	Shuffle dataset before splitting into train / test sets
Training	learning_rate	0.0001	Adam optimizer learning rate
	weight_decay	5e-4	Adam optimizer weight decay
	n_epochs	8000	Number of epochs (full training set) to run

Table A.2: Fixed model parameters for all experiments

of the input graph and, for every node v within the orbit, appends a new node w to the graph that is connected to only v.

The task in the dataset is to identify the largest orbit(s) in the graph (where the orbits are computed by orbit-1-WL) and, for each largest orbit r, distribute the labels $T := \{\!\{1, 1, ..., 1, 1, 2, ..., m\}\!\}$ at random without replacement among the orbit's nodes, where $|T| = |r|$. Thus, T contains the multiset $\{\!\{1, 2, ..., m\}\!\}$ and is padded with 1s to ensure that there are the same number of labels in it as nodes in the orbit. Every node not part of a largest orbit should be assigned the label of 0. Equivariant functions such as MPNNs cannot solve any of the examples in the dataset. The function f required to solve the task is orbit-equivariant with $\texttt{max-orbit}(f) = m$.

A.3.3 Hyperparameters

Fixed hyperparameters across all experiments are shown in Table A.2. The fixed hyperparameters were chosen by preliminary optimization experiments of GCN on Bioisostere. Hyperparameters for the experiments on Bioisostere are shown in Table A.3. Hyperparameters for the experiments on Alchemy-Max-Orbit-2 are shown in Table A.4. Hyperparameters for the experiments on Alchemy-Max-Orbit-6 are shown in Table A.5.

Parameter	Value	Description
dataset	bioisostere	Dataset
loss	[cross_entropy, orbit_sorting_cross_entropy]	Loss function
model	[gcn, unique_id_gcn, rni_gcn, orbit_indiv_gcn, max_orbit_gcn]	Model (max_orbit = Orbit-Transform-MPNN)
model_max_orbit	2	Max orbit for Orbit-Transform-MPNN
seed	[1,2,3,4,5,6,7,8,9,10]	Seed for all random generators

Table A.3: Model parameters for Bioisostere experiments

Parameter	Value	Description
dataset	alchemy	Base dataset
max_orbit_alchemy	2	Max-orbit size (m) in Alchemy-Max-Orbit-m
loss	[cross_entropy, orbit_sorting_cross_entropy]	Loss function
model	[unique_id_gcn, rni_gcn, orbit_indiv_gcn, max_orbit_gcn]	Model (max_orbit = Orbit-Transform-MPNN)
model_max_orbit	2	Max orbit for Orbit-Transform-MPNN
seed	[1,2,3,4,5,6,7,8,9,10]	Seed for all random generators

Table A.4: Model parameters for Alchemy-Max-Orbit-2 experiments

A.3.4 Loss

Here follows a full explanation of *orbit-sorting cross-entropy loss*. Let f be a learnable node-labelling function and D a dataset of graphs. Let t be a "target function" that yields a category for each node in the input graph. For each orbit r of the graph G (computed by orbit-1-WL), collect the model outputs $(f(G)_v \mid v \in r)$ and the target outputs $(t(G)_v \mid v \in r)$ for the orbit, both as tuples. The category predictions of the model are $(\text{arg_max}(f(G)_v) \mid v \in r)$, since each output channel of $f(G)_v$ corresponds to a category. Fix some a-priori order on the output categories and define a function `sort` that takes in a tuple of categories c and returns the permutation σ that, when applied to the indices of the tuple, yields the tuple

Parameter	Value	Description
dataset	alchemy	Base dataset
max_orbit_alchemy	6	Max-orbit size (m) in Alchemy-Max-Orbit-m
loss	[cross_entropy, orbit_sorting_cross_entropy]	Loss function
model	[unique_id_gcn, rni_gcn, orbit_indiv_gcn, max_orbit_gcn]	Model (max_orbit = Orbit-Transform-MPNN)
model_max_orbit	6	Max orbit for Orbit-Transform-MPNN
seed	[1,2,3,4,5,6,7,8,9,10]	Seed for all random generators

Table A.5: Model parameters for Alchemy-Max-Orbit-6 experiments

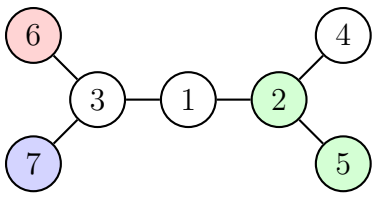
sorted by category. For example, given the tuple $c = (8, 2, 4, 7, 3)$ of categories, $\texttt{sort}(c) = \sigma = \{(1, 5), (2, 1), (3, 3), (4, 4), (5, 2)\}$, and $\sigma \cdot c = (2, 3, 4, 7, 8)$.

For the entire graph, initially set $T := t(G)$. Then, for each orbit r (computed by orbit-1-WL), compute $\sigma_f := \texttt{sort}((\text{arg_max}(f(G)_v) \mid v \in r))$ and $\sigma_t := \texttt{sort}((t(G)_v \mid v \in r))$, the permutations that sort the model category predictions and the targets respectively. Next, compute a re-ordered target T^r for the orbit, with $T^r := \sigma_f^{-1} \cdot (\sigma_t \cdot (t(G)_v \mid v \in r))$. Then $T[r]$ is set to T^r, where $T[r]$ accesses the indices of T using the nodes in r.

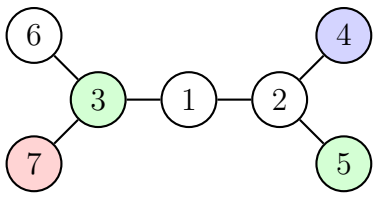
Once all orbits have been iterated through, we return the cross-entropy between the new targets T and the model outputs $(f(G)_v \mid v \in r)$. Note that cross-entropy is computed between T and the raw output of f, instead of using $(\text{arg_max}(f(G)_v) \mid v \in r)$ (the categorical predictions of f), even though the categorical predictions were used for the sorting process.

To better illustrate the mechanisms of orbit-sorting cross-entropy loss, here follows an example. For an input graph G, $f(G)$ (converted to categorical outputs with arg_max, for ease of notation) and $t(G)$ are shown below, where colours represent node features. White = 0, red = 1, green = 2, and blue = 3.

$f(G)$:



$t(G)$:



The orbits of this graph are $r_1 = \{1\}, r_2 = \{2, 3\}, r_3 = \{4, 5, 6, 7\}$, as computed correctly by orbit-1-WL. Set $T := t(G) = (0, 0, 2, 3, 2, 0, 1)$ initially. r_1 has size 1, so no re-ordering is possible. For r_2, $(\text{arg_max}(f(G)_v) \mid v \in r_2) = (2, 0)$, so $\sigma_f = \{(1, 2), (2, 1)\}$. Also, $(t(G)_v \mid v \in r_2) = (0, 2)$, so $\sigma_t = \{(1, 1), (2, 2)\}$. Thus, $T^{r_2} := \sigma_f^{-1} \cdot (\sigma_t \cdot (t(G)_v \mid v \in r_2)) = \sigma_f^{-1} \cdot (\sigma_t \cdot (0, 2)) = (2, 0)$. $T[r_2]$ is then set to T^{r_2}, so $T = (0, \mathbf{2}, \mathbf{0}, 3, 2, 0, 1)$ (the affected part of the tuple is highlighted in bold).

For r_3, $(\text{arg_max}(f(G)_v) \mid v \in r_3) = (0, 2, 1, 3)$, so

$$\sigma_f = \{(1, 1), (2, 3), (3, 2), (4, 4)\}.$$

Also, $(t(G)_v \mid v \in r_3) = (3, 2, 0, 1)$, so

$$\sigma_t = \{(1, 4), (2, 3), (3, 1), (4, 2)\}.$$

Thus, $T^{r_3} := \sigma_f^{-1} \cdot (\sigma_t \cdot (t(G)_v \mid v \in r_3)) = \sigma_f^{-1} \cdot (\sigma_t \cdot (3, 2, 0, 1)) = (0, 2, 1, 3)$. $T[r_3]$ is then set to T^{r_3}, so $T = (0, 2, 0, \mathbf{0}, \mathbf{2}, \mathbf{1}, \mathbf{3})$. Cross-entropy is then computed between the new targets T and the model outputs $(f(G)_v \mid v \in r)$, which have the respective arg_maxes of $(0, 2, 0, 0, 2, 1, 3)$.

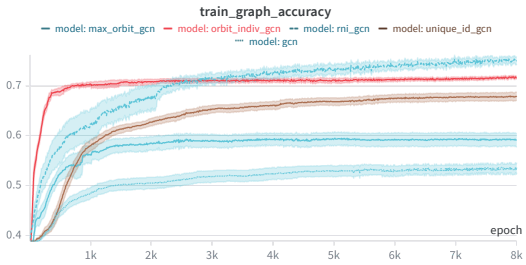
For the m-Orbit-Transform-MPNN models, loss is computed before the transform takes place. This method works for this model in particular since the transform component of the function does not need to be trained. Thus, standard cross-entropy can be used, since the model is equivariant before the transform is applied.

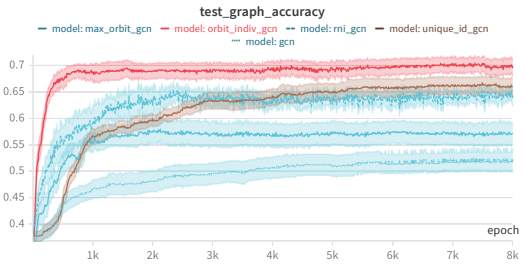
A.3.5 Full Results

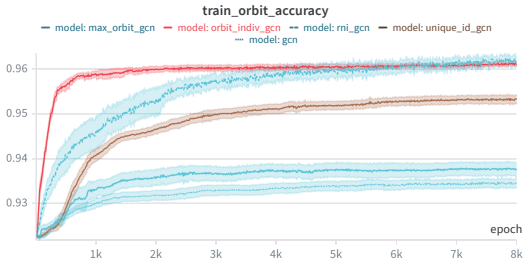
All experiments were run in a distributed manner on an internal university cluster, on exclusively CPU nodes. In my final experiments, there were 260 individual runs. A total of 88 days of compute time was used for my final experiments. Across all of my experiments, including failed runs and preliminary experiments, a total of 219 days of compute time was used. Full results for my experiments across each of the datasets are shown in the following sections. The mean is shown along with standard error.

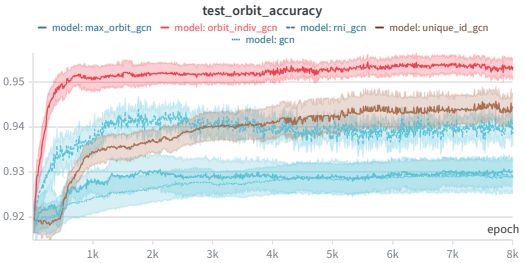
A.3.5.1 Bioisostere

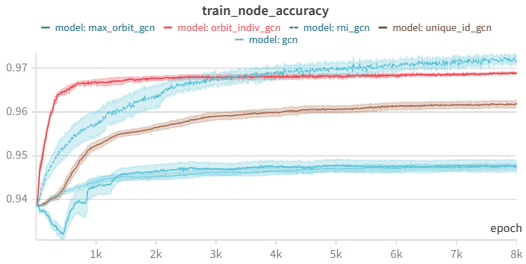
Cross-Entropy Loss

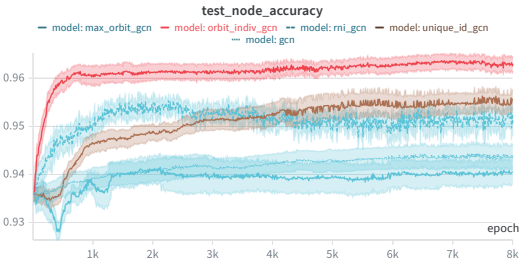


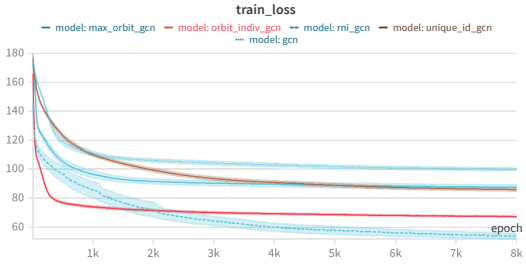




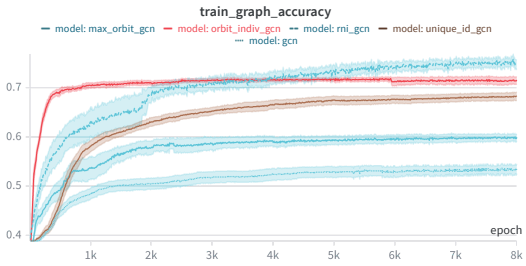


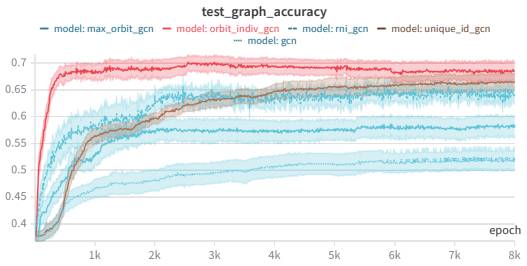


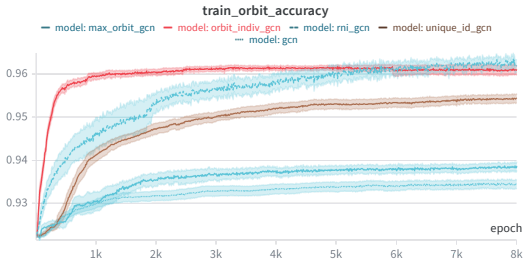


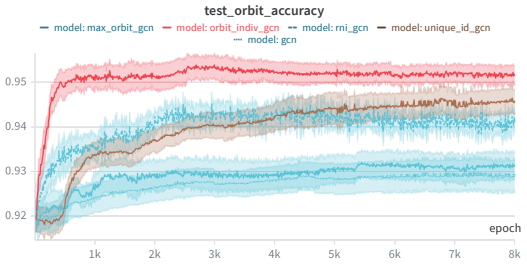


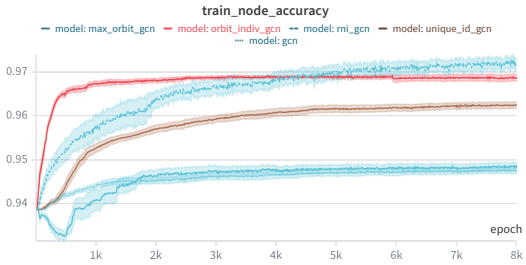
Orbit-Sorting Cross-Entropy Loss

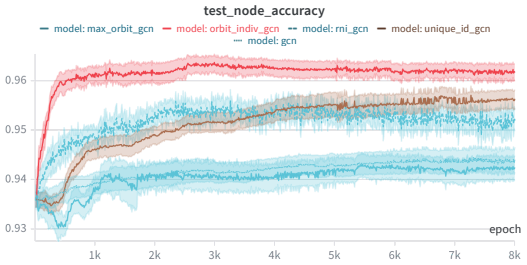


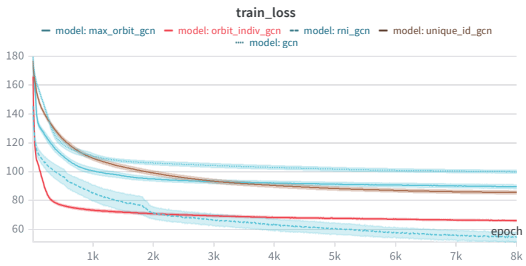






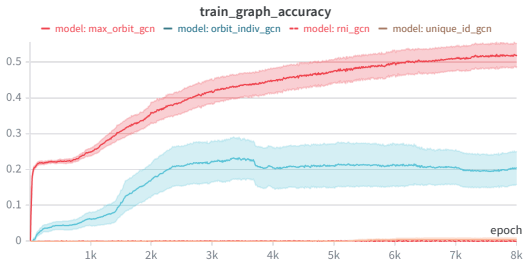


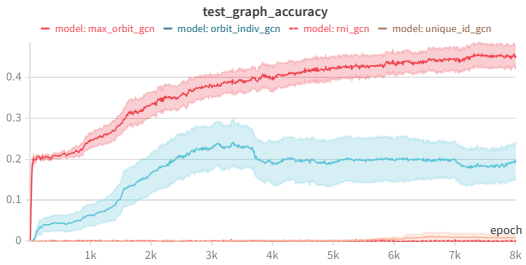


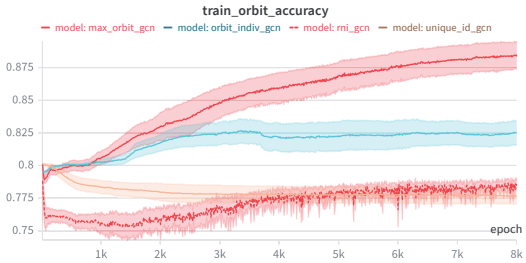


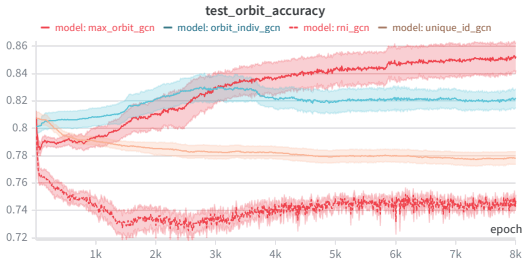
A.3.5.2 Alchemy-Max-Orbit-2

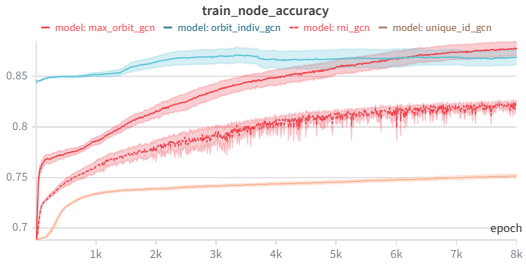
Cross-Entropy Loss

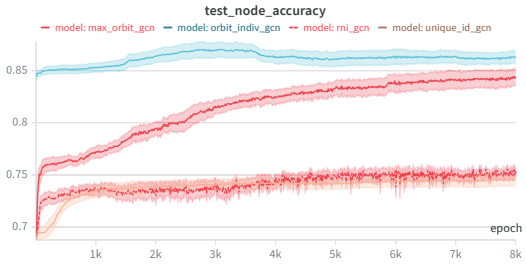


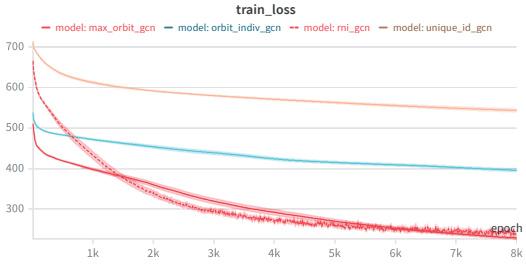




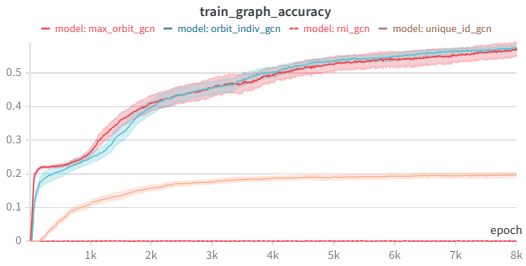


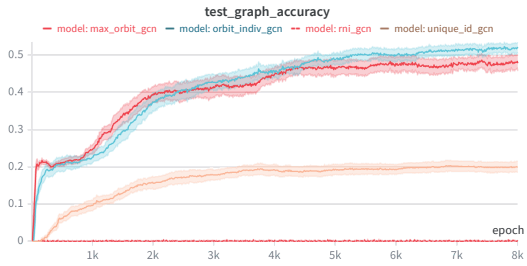


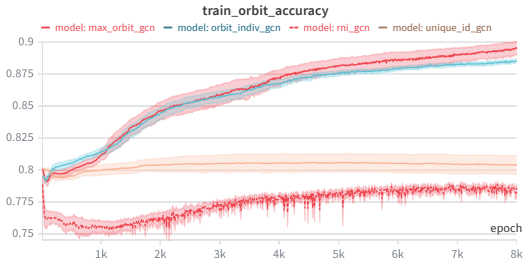


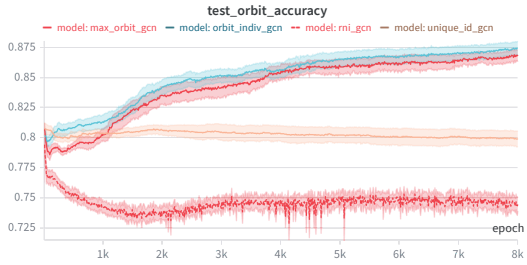


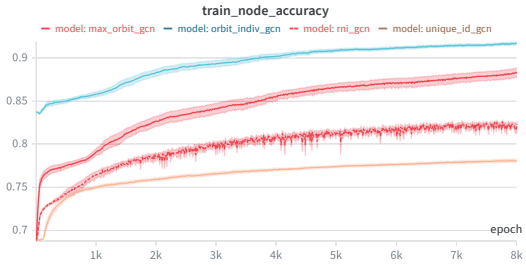
Orbit-Sorting Cross-Entropy Loss

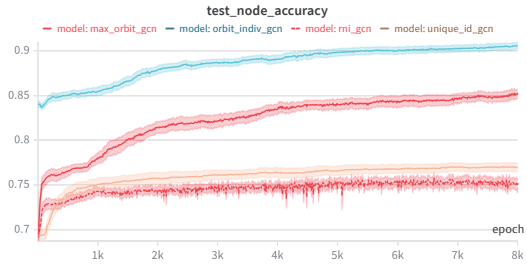


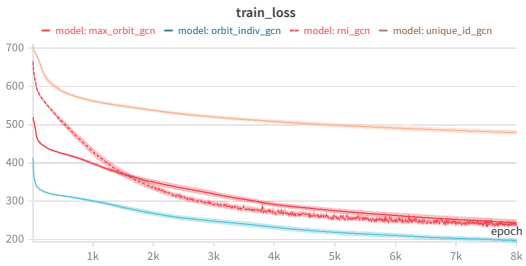






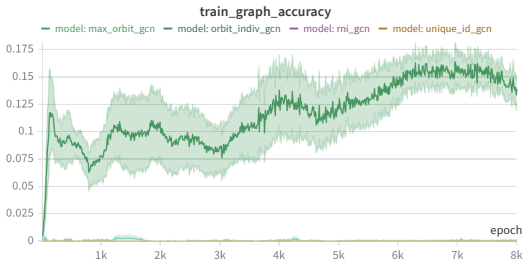


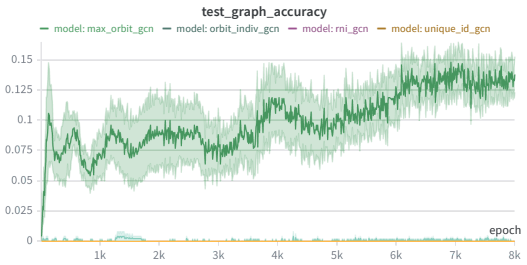


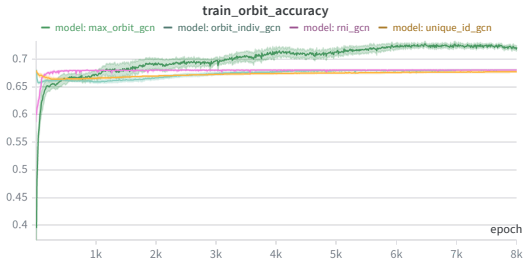


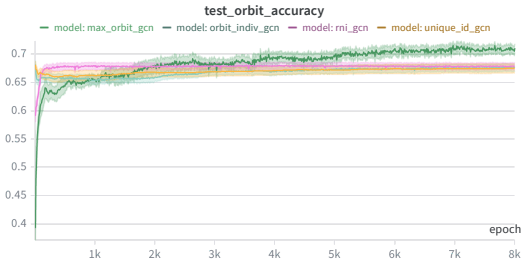
A.3.5.3 Alchemy-Max-Orbit-6

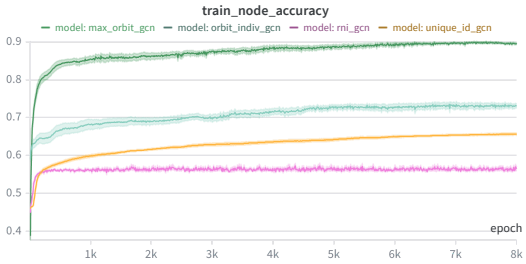
Cross-Entropy Loss



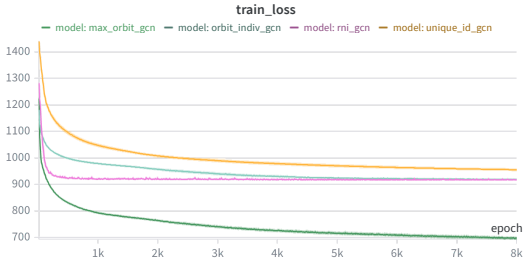




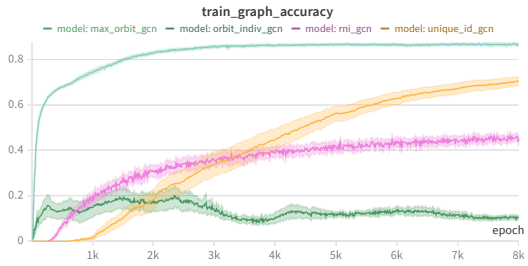


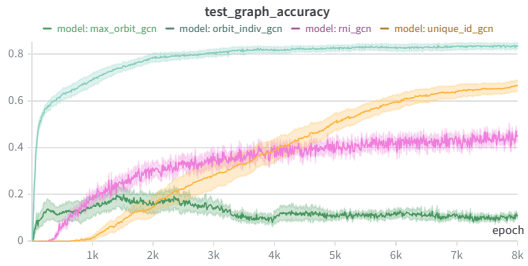


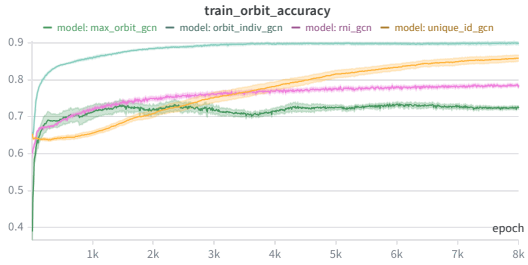


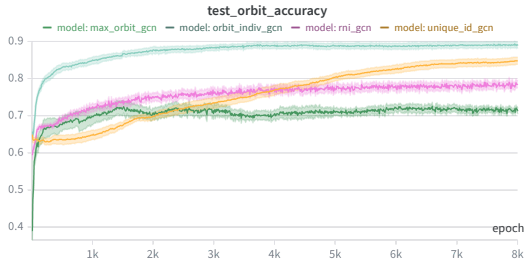


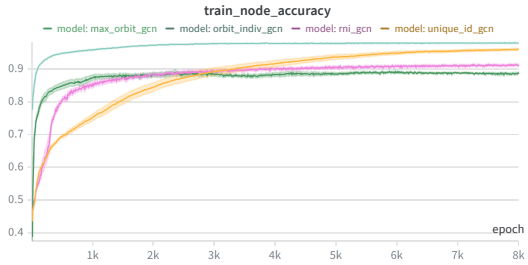
Orbit-Sorting Cross-Entropy Loss

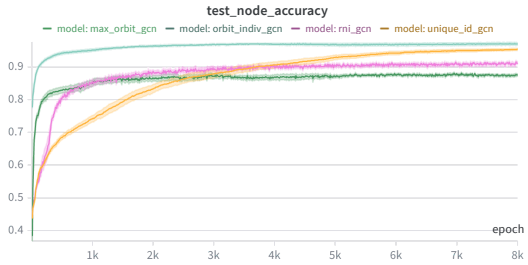


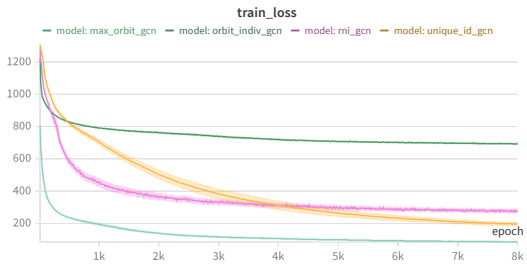












B

Appendices for Chapter 5

Contents

B.1 Theory 189
B.1.1 Non Monotonically Increasing Scoring Functions 189
B.1.2 Monotonically Increasing Semantic Matching Models . . 189
B.1.3 Monotonically Increasing Neural Network Models 191
B.1.4 Initial Node Features without Unary Predicates 192
B.2 Full Proofs 194
B.2.1 Lemma 45 194
B.2.2 Lemma 11 194
B.2.3 Proposition 12 195
B.2.4 Proposition 15 197
B.2.5 Corollary 16 198
B.2.6 Corollary 17 198
B.2.7 Theorem 20 199
B.2.8 Capacity Setup 205
B.2.9 Theorem 23 207
B.2.10 Theorem 24 209
B.3 Full Experiment Results 217

B.1 Theory

B.1.1 Non Monotonically Increasing Scoring Functions

ComplEx is defined as follows:

$$f(R, \mathbf{h}, \mathbf{t}) = \mathrm{Re}(\mathbf{h}^\top \mathbf{M}_R \bar{\mathbf{t}}),$$

where each $\mathbf{M}_R$ is a diagonal $d_f \times d_f$ matrix, $\mathbf{h}, \mathbf{t}, \mathbf{M}_R$ contain complex numbers, Re denotes the real part of a complex number, and $\bar{\mathbf{t}}$ denotes the complex conjugate. Splitting these vectors / matrices into their real and imaginary components, we obtain $\mathbf{h} = \mathbf{h}^{\mathrm{im}} \cdot i + \mathbf{h}^{\mathrm{re}},\ \mathbf{M}_R = \mathbf{M}_R^{\mathrm{im}} \cdot i + \mathbf{M}_R^{\mathrm{re}},\ \mathbf{t} = \mathbf{t}^{\mathrm{im}} \cdot i + \mathbf{t}^{\mathrm{re}}$, and thus:

$$f(R, \mathbf{h}, \mathbf{t}) = \mathbf{h}^{\mathrm{re}}\mathbf{M}_R^{\mathrm{re}}\mathbf{t}^{\mathrm{re}} + \mathbf{h}^{\mathrm{im}}\mathbf{M}_R^{\mathrm{re}}\mathbf{t}^{\mathrm{im}} + \mathbf{h}^{\mathrm{re}}\mathbf{M}_R^{\mathrm{im}}\mathbf{t}^{\mathrm{im}} - \mathbf{h}^{\mathrm{im}}\mathbf{M}_R^{\mathrm{im}}\mathbf{t}^{\mathrm{re}}.$$

To ensure monotonicity, we have to enforce one of $\mathbf{h}^{\mathrm{im}}, \mathbf{M}_R^{\mathrm{im}}, \mathbf{t}^{\mathrm{re}} = 0$, otherwise we will have the same situation as for the distance-based models, where we have vectors with different signs. Any one of these being zero eliminates two of the terms of the above equation, so for now assume only $\mathbf{M}_R^{\mathrm{im}} = 0$, i.e. that the relation vector has only real numbers. Then we obtain:

$$f(R, \mathbf{h}, \mathbf{t}) = \mathbf{h}^{\mathrm{re}}\mathbf{M}_R^{\mathrm{re}}\mathbf{t}^{\mathrm{re}} + \mathbf{h}^{\mathrm{im}}\mathbf{M}_R^{\mathrm{re}}\mathbf{t}^{\mathrm{im}}.$$

This is a severely limited version of ComplEx, so I do not use it. A similar problem occurs for QuatE and DualE, which use quaternions instead of the complex numbers.

B.1.2 Monotonically Increasing Semantic Matching Models

When restricting the main parameters of the following— expressed as vectors, matrices, or tensors—to contain only non-negative values, we obtain monotonically increasing scoring functions.

RESCAL is defined by

$$f(R, \mathbf{h}, \mathbf{t}) = \mathbf{h}^\top \mathbf{M}_R \mathbf{t},$$

where $\mathbf{M}_R \in \mathbb{R}^{d_f \times d_f}$. When every $\mathbf{M}_R$ is diagonal, we obtain DistMult. When we enforce normality ($\mathbf{M}_R\mathbf{M}_R^\top = \mathbf{M}_R^\top\mathbf{M}_R$) and commutativity ($\mathbf{M}_R\mathbf{M}_{R'} = \mathbf{M}_{R'}\mathbf{M}_R$) for all relations R, R', we obtain ANALOGY. TATEC is defined by:

$$f(R, \mathbf{h}, \mathbf{t}) = \mathbf{h}^\top \mathbf{M}_R \mathbf{t} + \mathbf{h}^\top \mathbf{r}_R + \mathbf{t}^\top \mathbf{r}_R + \mathbf{h}^\top \mathbf{D}\mathbf{t},$$

where $\mathbf{r}_R \in \mathbb{R}^{d_f}, \mathbf{M}_R \in \mathbb{R}^{d_f \times d_f}, \mathbf{D} \in \mathbb{R}^{d_f \times d_f}$. HolE uses circular correlation operation to first combine the constant representations, and is defined as follows:

$$\begin{aligned}(\mathbf{h} \star \mathbf{t})[i] &= \sum_{j=1}^{d_f} \mathbf{h}[j] \cdot \mathbf{t}[(j+i) \bmod d_f] \\ f(R, \mathbf{h}, \mathbf{t}) &= \mathbf{r}_R^\top (\mathbf{h} \star \mathbf{t}),\end{aligned}$$

where $\mathbf{h} \star \mathbf{t}$ produces a vector of dimension d_f and $\mathbf{r}_R \in \mathbb{R}^{d_f}$. TuckER is defined by:

$$\begin{aligned}f(R, \mathbf{h}, \mathbf{t}) &= \mathbf{W} \times_1 \mathbf{h} \times_2 \mathbf{r}_R \times_3 \mathbf{t} \\ &= \sum_{1 \le i,k \le d_f, 1 \le j \le d_r} \mathbf{W}[i,j,k]\mathbf{h}[i]\mathbf{r}_R[j]\mathbf{t}[k],\end{aligned}$$

where $\times_n$ denotes the n-mode tensor product, $d_r \in \mathbb{N}$, $\mathbf{W} \in \mathbb{R}^{d_f \times d_r \times d_f}$, and $\mathbf{r}_R \in \mathbb{R}^{d_r}$. Note that TuckER can be rewritten as a bilinear model: $f(R, \mathbf{h}, \mathbf{t}) = \mathbf{h}\mathbf{M}_R\mathbf{t} = \sum_{i=1}^{d_f}\sum_{k=1}^{d_f} \mathbf{h}[i]\mathbf{M}_R[i,k]\mathbf{t}[k]$. To do so, we define $\mathbf{M}_R \in \mathbb{R}^{d_f \times d_f}$ by $\mathbf{M}_R[i,k] = \sum_{j=1}^{d_r} \mathbf{W}[i,j,k]\mathbf{r}_R[j]$. Then:

$$
\begin{aligned}
f(R, \mathbf{h}, \mathbf{t}) &= \sum_{1 \leq i,k \leq d_f, 1 \leq j \leq d_r} \mathbf{W}[i,j,k]\mathbf{h}[i]\mathbf{r}_R[j]\mathbf{t}[k] \\
&= \sum_{i=1}^{d_f} \sum_{k=1}^{d_f} \sum_{j=1}^{d_r} \mathbf{W}[i,j,k]\mathbf{h}[i]\mathbf{r}_R[j]\mathbf{t}[k] \\
&= \sum_{i=1}^{d_f} \sum_{k=1}^{d_f} \mathbf{h}[i]\mathbf{t}[k] \sum_{j=1}^{d_r} \mathbf{W}[i,j,k]\mathbf{r}_R[j] \\
&= \sum_{i=1}^{d_f} \sum_{k=1}^{d_f} \mathbf{h}[i]\mathbf{t}[k]\mathbf{M}_R[i,k] \\
&= \sum_{i=1}^{d_f} \sum_{k=1}^{d_f} \mathbf{h}[i]\mathbf{M}_R[i,k]\mathbf{t}[k] \\
&= \mathbf{h}\mathbf{M}_R\mathbf{t}
\end{aligned}
$$

SimplE [131] is defined as follows:

$$f(R, \mathbf{h}, \mathbf{t}) = \frac{1}{2}(\langle \mathbf{h}, \mathbf{r}_R, \mathbf{t} \rangle + \langle \mathbf{h}, \mathbf{r}_R^{-1}, \mathbf{t} \rangle),$$

where $\langle \mathbf{h}, \mathbf{r}_R, \mathbf{t} \rangle = \sum_{i=1}^{d_f} \mathbf{h}[i] \cdot \mathbf{r}_R[i] \cdot \mathbf{t}[i]$ and each relation R has two corresponding vectors, $\mathbf{r}_R, \mathbf{r}_R^{-1} \in \mathbb{R}^{d_f}$.

B.1.3 Monotonically Increasing Neural Network Models

We now consider some neural network models. When restricting their main parameters— expressed as vectors, matrices, tensors, or neural network weights—to contain only non-negative values, and by making all non-linear functions in them monotonically increasing and non-negative (e.g., ReLU), we obtain monotonically increasing scoring functions.

SME is defined by:

$$f(R, \mathbf{h}, \mathbf{t}) = g_u(\mathbf{h}, \mathbf{r}_R)^\top g_v(\mathbf{t}, \mathbf{r}_R)^\top,$$

where g_u, g_v are neural networks, $k \in \mathbb{N}$, and each $\mathbf{r}_R \in \mathbb{R}^k$. NTN is defined by:

$$f(R, \mathbf{h}, \mathbf{t}) = \mathbf{r}_R^\top \, \sigma(\mathbf{h}^\top \mathbf{T}_R \mathbf{t} + \mathbf{M}_R^1 \mathbf{h} + \mathbf{M}_R^2 \mathbf{t} + \mathbf{b}_R),$$

where σ is a non-linear activation function, $k \in \mathbb{N}$, each $\mathbf{T}_R \in \mathbb{R}^{k \times d_f \times d_f}$, $\mathbf{M}_R^1, \mathbf{M}_R^2 \in \mathbb{R}^{k \times d_f}$, and $\mathbf{r}_R, \mathbf{b}_R \in \mathbb{R}^k$. When every $\mathbf{T}_R$ and $\mathbf{b}_R$ is set to $\mathbf{0}$, it yields the model SLM. MLP is defined as follows:

$$f(R, \mathbf{h}, \mathbf{t}) = \mathbf{w}^\top \sigma(\mathbf{M}^1 \mathbf{h} + \mathbf{M}^2 \mathbf{r}_R + \mathbf{M}^3 \mathbf{t})$$

where σ is a non-linear activation function, $k \in \mathbb{N}$, $\mathbf{M}^1, \mathbf{M}^2, \mathbf{M}^3 \in \mathbb{R}^{k \times d_f}$, $\mathbf{w} \in \mathbb{R}^k$, and each $\mathbf{r}_R \in \mathbb{R}^{d_f}$.

NAM is defined as follows:

$$f(R, \mathbf{h}, \mathbf{t}) = \mathbf{t}^\top \cdot N(\mathbf{h}, \mathbf{r}_R),$$

where N is a deep neural network and each $\mathbf{r}_R \in \mathbb{R}^{d_f}$. ConvE [76] is defined as follows:

$$f(R, \mathbf{h}, \mathbf{t}) = \sigma(\text{vec}(f([\bar{\mathbf{h}}; \bar{\mathbf{r}}] \star \omega))\mathbf{W})\mathbf{t},$$

where $\mathbf{r}_R$ is a vector, $\bar{\mathbf{h}}, \bar{\mathbf{r}}$ denote a 2D reshaping of $\mathbf{h}, \mathbf{r}$ respectively (i.e. a vector with dimension $k_1 \cdot k_2$ is turned into a matrix of dimension $k_1 \times k_2$), $[_;_]$ denotes concatenation, $\star\, \omega$ denotes a convolution with filters ω, vec flattens an input tensor into a vector, σ is a non-linear activation function, and $\mathbf{W}$ is a matrix. HypER [18] is defined as follows, where the notation is the same as above, except additionally vec^{-1} turns an input vector into a matrix and $\mathbf{H}$ is a matrix:

$$f(R, \mathbf{h}, \mathbf{t}) = \sigma(\text{vec}(\mathbf{h} \star \text{vec}^{-1}(\mathbf{w}_R \mathbf{H}))\mathbf{W})\mathbf{t}.$$

B.1.4 Initial Node Features without Unary Predicates

In scenarios where the KG completion task is transductive, i.e. where the test set contains only constants that are seen during training, the dominant approach is to learn an embedding for each constant. However, in inductive settings such as those of Teru et al. [258] and Liu et al. [162], this approach does not work. Furthermore, in these particular datasets (as well as others), there are no unary predicates, only

binary predicates. Thus, since the straightforward canonical encoding method uses unary predicates to obtain initial features for each node, an alternative is needed in such scenarios. A simple approach, which I adopt since it does not change the model expressivity, is to have a single dummy unary predicate in the signature. In this way, every node will have the same initial feature.

Other approaches to this problem include that of Teru et al. [258], where each node w in the sub-graph around nodes u and v is labelled with the tuple $(d(w,u), d(w,v))$, where $d(w,u)$ denotes the shortest distance between w and u in the sub-graph. Hamaguchi et al. [111] use a GCN to compute embeddings for constants that were not seen during training, using existing embeddings for neighbouring constants that appeared in training. Schlichtkrull et al. [234] state that "the input to the first layer can be chosen as a unique one-hot vector for each node in the graph if no other features are present". Berg et al. [27] also use unique one-hot vectors for every node in the graph. Finally, Zhang and Chen [327, Section 4.1] give an initial feature to each node based on the structure of an enclosing sub-graph.

B.2 Full Proofs

B.2.1 Lemma 45

I introduce a definition and lemma that will aid in the proof of Proposition 12.

Definition 44. *An* isomorphism *from a dataset D to D' is an injective mapping h of constants to constants that is defined on $\mathtt{con}(D)$ and satisfies $h(D) = D'$, where $h(D)$ is the dataset obtained by replacing each fact $U(s) \in D$ with $U(h(s))$ and each fact of the form $R(s,t)$ with $R(h(s), h(t))$.*

Lemma 45. *Let $\mathcal{M}$ be a max-sum-GCN and f a scoring function. Then for all datasets D, D', each isomorphism from D to D' is an isomorphism from $T_{\mathcal{M},f}(D)$ to $T_{\mathcal{M},f}(D')$.*

Proof. Let h be an isomorphism from D to D'. This induces a bijective mapping π between the vertices of $\mathtt{enc}(D)$ and $\mathtt{enc}(D')$.

Let vertices $u, v \in \mathtt{enc}(D)$ and $u' := \pi(u), v' := \pi(v) \in \mathtt{enc}(D')$. Define $\mathbf{u}_L$ (and similarly $\mathbf{v}_L, \mathbf{u}'_L, \mathbf{v}'_L$) to be the final embedding of v after $\mathcal{M}$ has been applied.

Since the result of applying $\mathcal{M}$ depends only on the structure of the graph and not the specific names of the vertices, we have that $\mathbf{u}_L = \mathbf{u}'_L$ and $\mathbf{v}_L = \mathbf{v}'_L$, and thus that for any binary relation R, $f(R, \mathbf{u}_L, \mathbf{v}_L) = f(R, \mathbf{u}'_L, \mathbf{v}'_L)$.

Thus, for any fact $R(s,t) \in D$, we must also have $R(h(s), h(t)) \in D'$. □

B.2.2 Lemma 11

Lemma. *Let $\mathcal{M}$ be a monotonic max-sum-GCN and f a monotonically increasing scoring function. Then for all datasets D, D' such that $D \subseteq D'$, $T_{\mathcal{M},f}(D) \subseteq T_{\mathcal{M},f}(D')$.*

Proof. Let D, D' be datasets such that $D \subseteq D'$ and E, E' their respective edges in the canonical encoding. For $v \in \mathtt{enc}(D)$, let $\mathbf{v}_\ell$ be the vector labelling of v after ℓ layers of $\mathcal{M}$ applied to D. Similarly define $\mathbf{v}'_\ell$ to be the labelling after ℓ layers of $\mathcal{M}$ applied to D'.

We prove by induction on $0 \leq \ell \leq L$ that $\mathbf{v}_\ell[i] \leq \mathbf{v}'_\ell[i]$ for each vertex $v \in \texttt{enc}(D)$ and $1 \leq i \leq \delta_\ell$. The base case follows from the canonical encoding and the fact that $D \subseteq D'$.

For the inductive step, consider a layer ℓ and $1 \leq i \leq \delta_\ell$. The canonical encoding and $D \subseteq D'$ implies that $E^c \subseteq E'^c$. For every $u \in \texttt{enc}(D)$ and $1 \leq j \leq \delta_{\ell-1}$, the inductive hypothesis gives us that $\mathbf{u}_{\ell-1}[j] \leq \mathbf{u}'_{\ell-1}[j]$. Together, these imply that for every $c \in \text{Col}$

$$\begin{aligned} &\text{max-}k_\ell\text{-sum}(\{\!\{\mathbf{u}_{\ell-1}[j] \mid (v,u) \in E^c\}\!\}) \\ \leq\ &\text{max-}k_\ell\text{-sum}(\{\!\{\mathbf{u}'_{\ell-1}[j] \mid (v,u) \in E'^c\}\!\}). \end{aligned}$$

Also, since all elements of $\mathbf{A}_\ell$ and each $\mathbf{B}^c_\ell$ are non-negative and σ_ℓ is monotonically increasing, we have $\mathbf{v}_\ell[i] \leq \mathbf{v}'_\ell[i]$.

Thus, as proven by induction, $\mathbf{v}_L[i] \leq \mathbf{v}'_L[i]$ for each vertex $v \in \texttt{enc}(D)$ and $1 \leq i \leq \delta_\ell$, and so $\mathbf{v}_L[i] \leq \mathbf{v}'_L[i]$ for all $i \in \{1, ..., \delta_L\}$.

Now let $R(s,t) \in T_{\mathcal{M},f}(D)$ and let $u, v \in \texttt{enc}(D)$ be the corresponding nodes for s, t. Then $f(R, \mathbf{u}_L, \mathbf{v}_L) \geq t_f$. But as proven above, we then have $\mathbf{u}_L[i] \leq \mathbf{u}'_L[i]$ and $\mathbf{v}_L[i] \leq \mathbf{v}'_L[i]$ for all $i \in \{1, ..., \delta_L\}$.

Now since f is monotonically increasing, we have $f(R, \mathbf{u}_L, \mathbf{v}_L) \leq f(R, \mathbf{u}'_L, \mathbf{v}'_L)$, and so $f(R, \mathbf{u}'_L, \mathbf{v}'_L) \geq t_f$. So $R(s,t) \in T_{\mathcal{M},f}(D')$, as required. □

B.2.3 Proposition 12

Proposition. *Let $\mathcal{M}$ be a monotonic max-sum-GCN and f a monotonically increasing scoring function. Let r be a Datalog rule with a binary head atom H, a (possibly empty) set A of body atoms, and a (possibly empty) set I of body inequalities. For each variable x in r, let a_x, b_x be distinct constants uniquely associated with x. Then, r is sound for $(\mathcal{M}, f)$ if and only $H\mu \in T_{\mathcal{M},f}(D_\mu)$ for each substitution μ mapping the variables of r to constants in the set $\{a_x \mid x \text{ is a variable of } r\}$ such that $\mu(x) \neq \mu(y)$ for each inequality $x \not\approx y \in I$, where D_μ is defined by*

$D_\mu = A\mu \cup \{R(b_x, \mu(x)) \mid x$ *is a variable occurring in* H *but not in* $A\}$ *where* R *is an arbitrary but fixed binary predicate.*

Proof. Let $\mathcal{M}$, r, H, A, and I be as specified in the proposition, and likewise a_x, b_x for each variable x in r. We prove the claim in both directions.

Forward Direction ($\Longrightarrow$)

Assume r is sound for $(\mathcal{M}, f)$ and consider arbitrary μ, D_μ as specified in the proposition. Note that μ is defined on the variables of r. Whichever constants μ maps to from variables in the body of r appear in D_μ by virtue of them being in $A\mu$. For any variable x in H but not in A, $\mu(x)$ will also appear in D_μ by its definition.

Thus, μ is a substitution mapping from the variables of r to the constants in D_μ. So by the properties of Datalog, we have $H\mu \in T_r(D_\mu)$. Then since r is sound, we obtain $H\mu \in T_{\mathcal{M}}(D_\mu)$, as required.

Backward Direction ($\Longleftarrow$)

Assume that $H\mu \in T_{\mathcal{M}}(D_\mu)$ for each μ as defined in the proposition. To prove the soundness of r, consider an arbitrary dataset D: we prove that $T_r(D) \subseteq T_{\mathcal{M}}(D)$. Consider an arbitrary fact in $T_r(D)$. It will have the form $H\nu$ for some substitution ν from the variables of r to the constants in D such that $D \models B_i\nu$ for each literal B_i in the body of r (which may be an atom from A or inequality from I).

We construct a mapping h from constants in the range of ν to the set:

$$\{a_x \mid x \text{ is a variable of } r\}.$$

For each constant d in the range of ν, define $h(d) = a_x$, where x is a variable of r such that $\nu(x) = d$ (if there are multiple such variables x, simply choose one). Notice that h is an injective mapping, since if $h(d_1) = h(d_2) = a_x$, then $d_1 = \nu(x) = d_2$, so $d_1 = d_2$.

We now let μ be the substitution $\mu(x) = h(\nu(x))$ for each variable x of r, and let D_μ be defined as in the proposition. Note that for all inequalities $B_i \in I$, $D \models B_i\nu$

and h being injective mean that $D_\mu \models B_i\mu$, so μ satisfies the constraints on its definition given in the proposition. Thus, $H\mu \in T_{\mathcal{M},f}(D_\mu)$.

Define $D_\nu = A\nu \cup \{R(b_x, \nu(x)) \mid x \text{ is a variable occurring in } H \text{ but not in } A\}$ and $D' = D \cup \{R(b_x, \nu(x)) \mid x \text{ is a variable occurring in } H \text{ but not in } A\}$, where R is as defined in the proposition. Then $h(D_\nu) = D_\mu$, so h is an isomorphism from D_ν to D_μ. By Lemma 45, h is thus an isomorphism from $T_{\mathcal{M},f}(D_\nu)$ to $T_{\mathcal{M},f}(D_\mu)$, and $h(T_{\mathcal{M},f}(D_\nu)) = T_{\mathcal{M},f}(D_\mu)$.

Since we have $H\mu \in T_{\mathcal{M},f}(D_\mu)$, we must then have $h^{-1}(H\mu) \in T_{\mathcal{M},f}(D_\nu)$, so $H\nu \in T_{\mathcal{M},f}(D_\nu)$. But $A\nu \subseteq D$, so $D_\nu \subseteq D'$. Then by Lemma 11, $T_{\mathcal{M},f}(D_\nu) \subseteq T_{\mathcal{M},f}(D')$, so $H\nu \in T_{\mathcal{M},f}(D')$.

Finally, note that the possible extra (binary) facts in D' that are not in D connect constants not in D to constants in D, but not the other way round; thus, the direction of GCN message propagation (reverse of the edge direction) ensures that $T_{\mathcal{M},f}(D')$ and $T_{\mathcal{M},f}(D)$ coincide on facts derived over the constants of D. Since all constants in the range of ν are in D, we have $H\nu \in T_{\mathcal{M},f}(D)$, as required.

□

B.2.4 Proposition 15

Proposition. *Let f be a scoring function and $B \to H$ a safe rule such that for all constants C and $e : C \to \mathbb{R}^{d_f}$, (f, C, e) captures $B \to H$. Let $\mathcal{M}$ be a GCN. Assume that for each atom B_i of B, there is an inequality-free rule $A_i \to B_i$ that is sound for $(\mathcal{M}, f)$, where each A_i is a conjunction of atoms. Then the rule $\bigwedge_i A_i \to H$ is sound for $(\mathcal{M}, f)$.*

Proof. Assume that for all atoms B_i in B, rule $r_2^i = A_i \to B_i$ is sound for $(\mathcal{M}, f)$ and let $r_3 = \bigwedge_i A_i \to H$. We prove that for all datasets D, $T_{r_3}(D) \subseteq T_{\mathcal{M},f}(D)$. To this end, consider a fact in $T_{r_3}(D)$. It will have the form $H\nu$ for some substitution ν from the variables of r_3 to the constants of D such that $D \models \bigwedge_i A_i\nu$. But then for each i, we have $D \models A_i\nu$, and thus $B_i\nu \in T_{r_2^i}(D)$.

We now define $C := \texttt{con}(D)$ and $e : C \to \mathbb{R}^{d_f}$ by $e(c) = \mathbf{v}_L^c$, the final embedding of the vertex corresponding to the constant c after $\mathcal{M}$ is applied to D. We likewise define $\mathbf{v}_L^a, \mathbf{v}_L^b$ for constants a, b.

Now consider an arbitrary atom B_i in B. Note that $B_i\nu = R_1(a, b)$, for some constants a, b in the range of ν and binary predicate R_1. From the soundness of r_2^i and $B_i\nu \in T_{r_2^i}(D)$, we thus obtain $R_1(a, b) \in T_{\mathcal{M},f}(D)$.

Then since $R_1(a, b) \in T_{\mathcal{M},f}(D)$, we have $f(R_1, \mathbf{v}_L^a, \mathbf{v}_L^b) \geq t_f$, which implies $f(R_1, e(a), e(b)) \geq t_f$, and thus $R_1(a, b) \in \mathcal{F}_f(C, e)$.

Thus, for every fact $B_i\nu$ in $B\nu$, we have $\mathcal{F}_f(C, e) \models B_i\nu$. So by definition, $\mathcal{F}_f(C, e) \models B\nu$. But as assumed in the lemma, (f, C, e) captures $B \to H$. So we obtain that $\mathcal{F}_f(C, e) \models H\nu$.

Note that $H\nu = R_2(c, d)$, for some constants c, d in the range of ν and binary predicate R_2. Going through the computation again, this means that $\mathcal{F}_f(C, e) \models R_2(c, d)$, so $f(R_2, e(c), e(d)) \geq t_f$, thus $f(R_2, \mathbf{v}_L^c, \mathbf{v}_L^d) \geq t_f$, which implies $R_2(c, d) \in T_{\mathcal{M},f}(D)$. □

B.2.5 Corollary 16

Corollary. *Let F be a scoring function family that universally captures a rule pattern ρ, $B \to H$ a rule conforming to ρ, $\mathcal{M}$ a GCN, and $f \in F$. Assume that for each atom B_i of B, there is an inequality-free rule $A_i \to B_i$ sound for $(\mathcal{M}, f)$. Then the rule $\bigwedge_i A_i \to H$ is sound for $(\mathcal{M}, f)$.*

Proof. F universally captures ρ. So for all constants C and $e : C \to \mathbb{R}^{d_f}$, we have (f, C, e) captures $B \to H$, which is a rule conforming to ρ. Then from Proposition 15, we have that $\bigwedge_i A_i \to H$ is sound for $(\mathcal{M}, f)$. □

B.2.6 Corollary 17

Corollary. *Let F be a scoring function family that exactly captures a rule pattern ρ, and $B \to H$ a rule conforming to ρ. Then there exists $f \in F$ such that: for each GCN $\mathcal{M}$ and inequality-free rules $A_i \to B_i$ sound for $(\mathcal{M}, f)$ for each B_i in B, the rule $\bigwedge_i A_i \to H$ is sound for $(\mathcal{M}, f)$.*

Proof. F exactly captures ρ, so there exists $f \in F$ such that for all constants C and $e : C \to \mathbb{R}^{d_f}$, (f, C, e) captures $B \to H$, a rule conforming to ρ. So let $\mathcal{M}$ be a GCN with $A_i \to B_i$ sound for $(\mathcal{M}, f)$ for all atoms B_i of B (where each A_i is a conjunction of atoms). Then from Proposition 15, we have that $\bigwedge_i A_i \to H$ is sound for $(\mathcal{M}, f)$. □

B.2.7 Theorem 20

The proof of this theorem uses the same techniques as in Tena Cucala et al. [256, Theorem 13], but differs due to (1) the inclusion of two variables x, y (with corresponding vertices and constants) in the base case, (2) the necessity of having two cases throughout the proof, to cover whether the binary fact corresponding to the head is of the form $R(s, t)$ for distinct constants s, t or of the form $R(s, s)$, and (3) the use of the monotonicity of the scoring function. To make the technical contributions of this proof clearer, I use footnotes to mark points at which the proof diverges from that of [256, Theorem 13].

Theorem. *Let $\mathcal{M}$ be a monotonic max-GCN and f a monotonically increasing scoring function. Let $\mathcal{P}_\mathcal{M}$ be the Datalog program containing, up to variable renaming, each $(L, |Col| \cdot \delta_\mathcal{M})$-tree-like rule without inequalities that is sound for $(\mathcal{M}, f)$, where $\delta_\mathcal{M} = max(\delta_0, ..., \delta_L)$. Then $T_{\mathcal{M},f}$ and $\mathcal{P}_\mathcal{M}$ are equivalent.*

Proof. We show that $T_{\mathcal{M},f}(D) = T_{P_\mathcal{M}}(D)$ holds for every dataset D. By definition of $P_\mathcal{M}$, each rule in $P_\mathcal{M}$ is sound for $(\mathcal{M}, f)$, so we have $T_r(D) \subseteq T_{\mathcal{M},f}(D)$ for every $r \in P_\mathcal{M}$, and thus $T_{P_\mathcal{M}}(D) \subseteq T_{\mathcal{M},f}(D)$. So we prove $T_{\mathcal{M},f}(D) \subseteq T_{P_\mathcal{M}}(D)$.

Let α be an arbitrary binary fact in $T_{\mathcal{M},f}(D)$. Observe that α can be of the form $R(s, t)$, for distinct constants s, t, or of the form $R(s, s)$, for a constant s. We will denote these two cases as **(Case 1)** and **(Case 2)**, respectively. We will prove both cases simultaneously, making a separate argument when each case needs to be treated differently. [1]

[1] Tena Cucala et al. [256, Theorem 13] considers only one case.

We construct a $(L, |\text{Col}| \cdot \delta_{\mathcal{M}})$-tree-like rule without inequalities r, such that $\alpha \in T_r(D)$ and r is sound for $(\mathcal{M}, f)$. Together, these will imply that $\alpha \in T_{P_{\mathcal{M}}}(D)$, as required for the proof.

Let $G = (V, E, \lambda)$ be the canonical encoding of D and let $\lambda_0, ..., \lambda_L$ the vertex labelling functions arising from the application of $\mathcal{M}$ to G.

Setup of Construction

We now construct a binary atom H, a conjunction Γ, a substitution ν from the variables of Γ to the set of constants in D, and a graph U (without vertex labels) where each vertex in U is of the form u^x for x a variable and each edge has a colour in Col. We also construct mappings $M_{c,\ell,j} : U \to V$ for each $c \in \text{Col}$, $1 \leq \ell \leq L$, and $1 \leq j \leq \delta_{\ell-1}$. Intuitively, $M_{c,\ell,j}(u^x)$ represents the c-coloured neighbour of v_t that contributes to the result of the max aggregation in vector index j at layer ℓ (where v_t is the vertex corresponding to the constant $\nu(x)$). We initialize Γ as the empty conjunction, and we initialize ν and each $M_{c,\ell,j}$ as empty mappings.

We will assign to each vertex in U a *level* between 0 and L. We will identify a single vertex from U as the *first root* vertex. Furthermore, if α is of the form $R(s, t)$, then we identify another distinct vertex from U as the *second root* vertex. [2]

In the remainder of the proof, we use letters s, t for constants in D, letters x, y for variables, letters v, w for the vertices in V, and (possibly indexed) letter u for the vertices in U. Our construction is by induction from level L down to level 1. The base case defines one or two vertices of level L. Then, for each $1 \leq \ell \leq L$, the induction step considers the vertices of level ℓ and defines new vertices of level $\ell - 1$.

Construction: Base Case

(Case 1) If α is of the form $R(s, t)$ for a binary predicate R, then V contains vertices v_s, v_t. We introduce fresh variables x, y, and define $\nu(x) = s, \nu(y) = t$; we define $H = R(x, y)$; we introduce vertices u^x, u^y of level L; and we make u^x the first

[2] In this, the proof differs from that of Tena Cucala et al. [256, Theorem 13], which only considers a single root vertex.

root vertex and u^y the second root vertex. Finally, we extend Γ with atom $U(x)$ for each $U(s) \in D$, and also extend Γ with atom $U(y)$ for each $U(t) \in D$. [3]

(Case 2) If on the other hand, α is of the form $R(s, s)$, then V contains a vertex v^s. We introduce fresh variables x, y, and define $\nu(x) = s, \nu(y) = s$; we define $H = R(x, y)$; we introduce vertices u^x, u^y of level L; and we make u^x the first root vertex and u^y the second root vertex. Finally, we extend Γ with atoms $U(x)$ and $U(y)$ for each $U(s) \in D$.

Construction: Induction Step

For the induction step, consider $1 \leq \ell \leq L$ and assume that all vertices of level greater than or equal to ℓ have been already defined. We then consider each vertex of the form u^x of level ℓ. Let $t = \nu(x)$. For each colour $c \in \mathrm{Col}$, each layer $1 \leq \ell' < \ell$, and each dimension $j \in \{1, ..., \delta_{\ell'-1}\}$ for which there exists some $(v^t, w) \in E^c$, let

$$M_{c,\ell',j}(u^x) = w, \text{for } (v^t, w) \in E^c \text{ and } \mathbf{w}_{\lambda_{\ell'}}[j] = \max(\{\!\{\mathbf{w}_{\lambda_{\ell'}}[j] \mid (v^t, w) \in E^c\}\!\})$$

Intuitively, w picks out the maximum c-coloured neighbour of v^t in feature vector index j at layer ℓ'. Note that there may be multiple such w that satisfy the above equation: any of them may be chosen. Note that such a w must exist, since we assumed that there exists some $(v^t, w) \in E^c$. Also, w must be of the form v^s for some constant s in D.

We then introduce a fresh variable y and define $\nu(y) = s$. We introduce a vertex u^y of level $\ell - 1$ and an edge $E^c(u^x, u^y)$ to U. Finally, we append to Γ the conjunction

$$E^c(x, y) \ \wedge \bigwedge_{U(s) \in D} U(y), \tag{B.1}$$

where $v^s = w = M_{c,\ell',j}(u^x)$ and U stands in for arbitrary unary predicates. This step adds at most $|\mathrm{Col}| \cdot \delta_{\ell'-1} \cdot \ell$ new successors of u^x. This completes the inductive construction.

[3] Again, Tena Cucala et al. [256, Theorem 13] considers only one case.

Rule Shape and Immediate Consequences

At this point in the proof, Γ consists of a conjunction of two formulas, one of which is $(L, |\text{Col}| \cdot \delta_{\mathcal{M}})$-tree-like for x and the other of which is $(L, |\text{Col}| \cdot \delta_{\mathcal{M}})$-tree-like for y (where x, y were defined in the base case). We also have $H = R(x, y)$, so $\Gamma \to H$ is a $(L, |\text{Col}| \cdot \delta_{\mathcal{M}})$-tree-like rule. We define this rule to be r. [4]

Furthermore, the construction of ν means $D \models \Gamma\nu$. So we have $H\nu \in T_r(D)$, with $H\nu = \alpha$, so $\alpha \in T_r(D)$, as required.

Note that if s, t are such that they are not mentioned in any fact in D, then Γ is empty, and r will then have the form $\top \to R(x, y)$. Under our definitions, r is still a $(L, |\text{Col}| \cdot \delta_{\mathcal{M}})$-tree-like rule in this case, since $\top$ is tree-like for x and y. This applies similarly if only one of s, t is not mentioned in any fact in D.

Rule Soundness

To complete the proof, we show that r is sound for $(\mathcal{M}, f)$. For this purpose, let D' be an arbitrary dataset and α' an arbitrary ground atom such that $\alpha' \in T_r(D')$. Then there exists a substitution ν' such that $D' \models \Gamma\nu'$.

Let the graph $G = (V', E', \lambda')$ be the canonical encoding of D' and $\lambda'_0, ..., \lambda'_L$ the functions labelling the vertices of G' when $\mathcal{M}$ is applied to it. We will use letters p, q, q' to denote vertices of V'.

We prove the following statement by induction, which relies on the monotonicity of $\mathcal{M}$: for each $0 \leq \ell \leq L$ and each vertex u^x of U whose level is at least ℓ, we have $\mathbf{v}_{\lambda_\ell}[i] \leq \mathbf{p}_{\lambda'_\ell}[i]$ for each $i \in \{1, ..., \delta_\ell\}$, where $v = v^{\nu(x)}$ and $p = v^{\nu'(x)}$. A similar statement was proved in Tena Cucala et al. [256, Theorem 13] in the same manner.

Monotonicity: Base Case

For the base case, $\ell = 0$, consider an arbitrary $1 \leq i \leq \delta_0$ and $u^x \in U$. Let $v = v^{\nu(x)}$ and $p = v^{\nu'(x)}$. Since $\ell = 0$, values $\mathbf{v}_{\lambda_0}[i], \mathbf{p}_{\lambda'_0}[i]$ come from the canonical encoding, so $\mathbf{v}_{\lambda_0}[i] \in \{0, 1\}$ and $\mathbf{p}_{\lambda'_0}[i] \in \{0, 1\}$.

[4] In Tena Cucala et al. [256, Theorem 13], Γ is a single tree-like formula, rather than a conjunction.

Thus, we prove that $\mathbf{v}_{\lambda_0}[i] = 1$ implies $\mathbf{p}_{\lambda'_0}[i] = 1$. By definition of the canonical encoding, $\mathbf{v}_{\lambda_0}[i] = 1$ implies $U_i(x\nu) \in D$. From the construction of Γ, we have $U_i(x) \in \Gamma$. Also, by definition of ν', we have $D' \models \Gamma\nu'$. So $U_i(x\nu') \in D'$ and $\mathbf{p}_{\lambda'_0}[i] = 1$, as required.

Monotonicity: Induction Step

For the induction step, assume that the property holds for some $\ell - 1$. Consider an arbitrary vertex $u^x \in U$ with level at least ℓ, $c \in \text{Col}$, and $i \in \{1, ..., \delta_i\}$. Let $v = v^{\nu(x)}$ and $p = v^{\nu'(x)}$. Then $\mathbf{v}_{\lambda_\ell}[i], \mathbf{p}_{\lambda'_\ell}[i]$ are computed as follows:

$$\mathbf{v}_{\lambda_\ell}[i] = \sigma_\ell\Bigg(\mathbf{b}_\ell[i] + \sum_{j=1}^{\delta_{\ell-1}} \mathbf{A}_\ell[i,j]\mathbf{v}_{\lambda_{\ell-1}}[j] + \tag{B.2}$$

$$\sum_{c\in\text{Col}} \sum_{j=1}^{\delta_{\ell-1}} \mathbf{B}_\ell[i,j]\max\{\!\{\mathbf{w}_{\lambda_{\ell-1}}[j] \mid (v,w) \in E^c\}\!\}\Bigg) \tag{B.3}$$

$$\mathbf{p}_{\lambda'_\ell}[i] = \sigma_\ell\Bigg(\mathbf{b}_\ell[i] + \sum_{j=1}^{\delta_{\ell-1}} \mathbf{A}_\ell[i,j]\mathbf{p}_{\lambda'_{\ell-1}}[j] + \tag{B.4}$$

$$\sum_{c\in\text{Col}} \sum_{j=1}^{\delta_{\ell-1}} \mathbf{B}_\ell[i,j]\max\{\!\{\mathbf{q}'_{\lambda'_{\ell-1}}[j] \mid (p,q') \in E'^c\}\!\}\Bigg) \tag{B.5}$$

The induction assumption ensures $\mathbf{v}_{\lambda_{\ell-1}}[j] \leq \mathbf{p}_{\lambda'_{\ell-1}}[j]$ for each $1 \leq j \leq \delta_{\ell-1}$. Also, for each $c \in \text{Col}$ and $1 \leq j \leq \delta_{\ell-1}$, we have $\max\{\!\{\mathbf{w}_{\lambda_{\ell-1}}[j] \mid (v,w) \in E^c\}\!\} = \mathbf{w}_{\lambda_{\ell-1}}[j]$, where $w = M_{c,\ell-1,j}(u^x)$.

If $w = M_{c,\ell-1,j}(u^x)$ is undefined, then there does not exist a $(v^t, w) \in E^c$, and so $\max\{\!\{\mathbf{w}_{\lambda_{\ell-1}}[j] \mid (v,w) \in E^c\}\!\} = 0$. Then $\mathbf{w}_{\lambda_{\ell-1}}[j] \leq \max\{\!\{\mathbf{q}'_{\lambda'_{\ell-1}}[j] \mid (p,q') \in E'^c\}\!\}$ holds trivially, since node values are non-negative. So instead assume that $w = M_{c,\ell-1,j}(u^x)$ is defined.

Recall that w has the form v^s, where s is a constant in D. Furthermore, by construction of U, there is a vertex u^y in U of level $\ell - 1$ such that $\nu(y) = s$ and $E^c(u^x, u^y)$ is in U.

Furthermore, Γ contains the atom $E^c(x, y)$. We then have $E^c(\nu'(x), \nu'(y)) \in D'$. Thus, $q = v^{\nu'(y)}$ is a c-neighbour of $v^{\nu'(x)}$ in G'. The induction assumption means that

for $w = v^{\nu(y)}$ and $q = v^{\nu'(y)}$, we have $\mathbf{w}_{\lambda_{\ell-1}}[j] \leq \mathbf{q}_{\lambda'_{\ell-1}}[j]$. So $\max\{\!\{\mathbf{w}_{\lambda_{\ell-1}}[j] \mid (v,w) \in E^c\}\!\} \leq \max\{\!\{\mathbf{q}'_{\lambda'_{\ell-1}}[j] \mid (p,q') \in E'^c\}\!\}$.

Thus, by the computations in Equation (B.2) and Equation (B.4), and since all elements of $\mathbf{A}_\ell$ and each $\mathbf{B}^c_\ell$ are non-negative and σ_ℓ is monotonically increasing, we have $\mathbf{v}_{\lambda_\ell}[i] \leq \mathbf{p}_{\lambda'_\ell}[i]$, as required.

Conclusion

Recall that $\alpha' \in T_r(D')$, so α' has the form $R(s',t')$ with $s' = \nu'(x)$ and $t' = \nu'(y)$. [5]

(Case 1) if α has the form $R(s,t)$, then $R(s,t) \in T_{\mathcal{M},f}(D)$ for $s = \nu(x)$ and $t = \nu(y)$, since $\alpha = R(s,t)$ is the original binary fact we assumed that the model produces from D.

Now let $v = v^s, w = v^t, p = v^{s'}, q = v^{t'}$. Then $R(s,t) \in T_{\mathcal{M},f}(D)$ implies $f(R, \mathbf{v}_{\lambda_L}, \mathbf{w}_{\lambda_L}) \geq t_f$.

The above property ensures that $\mathbf{v}_{\lambda_L}[i] \leq \mathbf{p}_{\lambda'_L}[i]$ and $\mathbf{w}_{\lambda_L}[i] \leq \mathbf{q}_{\lambda'_L}[i]$ for all $i \in \{1, ..., \delta_L\}$. Thus, since f is monotonically increasing, we have $f(R, \mathbf{v}_{\lambda_L}, \mathbf{w}_{\lambda_L}) \leq f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L})$.

So $f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L}) \geq t_f$, which implies that $R(s',t') \in T_{\mathcal{M},f}(D')$, as required to show the soundness of the rule.

(Case 2) if on the other hand, α has the form $R(s,s)$, then $R(s,s) \in T_{\mathcal{M},f}(D)$ for $s = \nu(x)$, since $\alpha = R(s,s)$ is the original binary fact we assumed that the model produces from D.

Now let $v = v^s, p = v^{s'}, q = v^{t'}$. Then $R(s,s) \in T_{\mathcal{M},f}(D)$ implies that $f(R, \mathbf{v}_{\lambda_L}, \mathbf{v}_{\lambda_L}) \geq t_f$. Recall that $\nu(x) = \nu(y) = s$, so we have $v = v^{\nu(x)} = v^{\nu(y)}$, $p = v^{\nu'(x)}$, and $q = v^{\nu'(y)}$.

Thus, the above property ensures that $\mathbf{v}_{\lambda_L}[i] \leq \mathbf{p}_{\lambda'_L}[i]$ and $\mathbf{v}_{\lambda_L}[i] \leq \mathbf{q}_{\lambda'_L}[i]$ for all $i \in \{1, ..., \delta_L\}$. Thus, since f is monotonically increasing, we have $f(R, \mathbf{v}_{\lambda_L}, \mathbf{v}_{\lambda_L}) \leq f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L})$.

[5] Here, the proof once again diverges from that of Tena Cucala et al. [256, Theorem 13], since we again need to consider multiple cases, as well as exploit the monotonicity of the scoring function.

So $f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L}) \geq t_f$, which implies that $R(s', t') \in T_{\mathcal{M},f}(D')$, as required to show the soundness of the rule.

□

B.2.8 Capacity Setup

For the following proofs, I cite from the paper "From Monotonic Graph Neural Networks to Datalog and Back: Expressive Power and Practical Applications" [257], which has been accepted for publication but not yet been published. The work is an extension of Tena Cucala et al. [256] which also addresses minor technical issues in the original paper. The proofs in this section, up to and including Theorem 24, are adapted from and use the same techniques as [256, 257]. However, the proof of Theorem 23 uses novel techniques, and depends on the structure of non-negative bilinear scoring functions.

To start, we adopt the definition of the sets $\mathcal{X}_{\ell,i}$ as given in [257, Definition 14]. We then use the same style of "capacity" argument made in [256, 257], which allows one to limit the number of neighbours considered in each aggregation step of a monotonic max-sum-GCN. We compute capacity in Algorithm 1, which is adapted from [257, Algorithm 3] by changing only the first line.

Algorithm 1: Computing Capacity

Input : A monotonic max-sum-GCN $\mathcal{M}$ and a non-negative bilinear scoring function f

Let $\alpha_L := \alpha$, as defined in Definition 22
for ℓ *from* L *down to* 1 **do**
  if *all elements of* $\mathbf{A}_\ell$ *and all* $\mathbf{B}^c_\ell$ *are* 0 *or* $\bigcup_i \mathcal{X}_{\ell-1,i} = \{0\}$ **then**
    $\mathcal{C}_\ell := \mathcal{C}_{\ell-1} := \ldots := \mathcal{C}_1 := 0$
    return
  $w_\ell :=$ the least non-zero element of $\mathbf{A}_\ell$ and all $\mathbf{B}^c_\ell$
  $\epsilon_\ell :=$ the least non-zero element of $\bigcup_i \mathcal{X}_{\ell-1,i}$
  $\beta_\ell :=$ the least natural number such that $\sigma_\ell(\beta_\ell) \geq \alpha_\ell$
  $b_\ell :=$ the least element of $\mathbf{b}_\ell$
  $\mathcal{C}_\ell := \min(k_\ell, \max(0, \lceil \frac{\beta_\ell - b_\ell}{w_\ell \cdot \epsilon_\ell} \rceil))$
  $\alpha_{\ell-1} := \frac{\beta_\ell - b_\ell}{w_\ell}$

We then define the capacity of a max-sum-GCN $\mathcal{M}$ and bilinear scoring function f, as follows.

Definition 46. *The* capacity $\mathcal{C}_\ell$ *of each layer* ℓ *of* $\mathcal{M}$ *is defined in Algorithm 1. The* capacity *of* $(\mathcal{M}, f)$ *is defined as* $\mathcal{C}_{\mathcal{M},f} = max\{\mathcal{C}_1, ..., \mathcal{C}_L\}$.

Algorithm 1 makes reference to a new definition I provide, Definition 22, which yields a value α and is presented again below.

Definition. *Consider a monotonic max-sum-GCN* $\mathcal{M}$ *and a non-negative bilinear scoring function* f. *For each binary relation* R *in the signature, where* $\mathbf{M}_R$ *is its relation matrix in* f, *we define:* $\alpha_R := 1$ *if all elements of* $\mathbf{M}_R$ *are* 0 *or* $\bigcup_i \mathcal{X}_{L,i} = \{0\}$. *Otherwise, we define* $\alpha_R :=$ *the least natural number such that* $\alpha_R \cdot w \cdot \epsilon \geq t_f$, *where* w *is the least non-zero element of* $\mathbf{M}_R$ *and* ϵ *is the least non-zero element of* $\bigcup_i \mathcal{X}_{L,i}$. *Then, we define* $\alpha := max\{\alpha_R \mid$ *for each binary relation* R *in the signature*$\}$.

Intuitively, α represents a value such that once a vertex feature (output of a GCN) is already greater than it, increasing it further (by adding facts to the input dataset) cannot change the output of $T_{\mathcal{M},f}$. In Tena Cucala et al. [256, 257], this role was filled by the classification threshold $t \in \mathbb{R}$ in $\texttt{cls}_t$.

Note that [257, Algorithm 4] can be used as defined to compute the smallest elements of each $\mathcal{X}_{\ell,i}$, which justifies the use of the term "algorithm" for Algorithm 1 and shows that α from Definition 22 can be computed. [257, Theorem 19] proves that [257, Algorithm 4] is correct and terminates on all inputs.

During the proof of [257, Theorem 18], the following claim is proved, pertaining to max-sum-GCNs whose aggregation has been restricted based on their capacity.

Proposition 47. *Let* D *be a dataset,* $\mathcal{M}$ *a max-sum-GCN with unbounded activation functions, and* $\mathcal{M}'$ *the GCN obtained from* $\mathcal{M}$ *by replacing* k_ℓ *with the capacity* $\mathcal{C}_\ell$ *for each* $\ell \in \{1, ..., L\}$. *For vertex* $v \in \texttt{enc}(D)$ *and layer* ℓ, *we define* $\mathbf{v}_\ell, \mathbf{v}'_\ell$ *to be the labelling of the vertices after layer* ℓ *by applying* $\mathcal{M}, \mathcal{N}'$ *to* $\texttt{enc}(D)$, *respectively. Let* ℓ_{st} *be the largest* $\ell \in \{1, ..., L\}$ *such that either all elements of* $\mathbf{A}_\ell$ *and all* $\mathbf{B}^c_\ell$ *are* 0, *or* $\bigcup_i \mathcal{X}_{\ell-1,i} = \{0\}$. *If such an* ℓ *does not exist, let* $\ell_{st} = 0$.

Then for each vertex $v \in \texttt{enc}(D)$*, layer* $\ell \in \{\ell_{st}, ..., L\}$*, and position* $i \in \{1, ..., \delta_\ell\}$*, either* $\mathbf{v}_\ell[i] = \mathbf{v}'_\ell[i]$ *or* $\mathbf{v}_\ell[i] > \mathbf{v}'_\ell[i] \geq \alpha_\ell$*.*

Notice that in [257, Algorithm 3], α_L is defined in terms of the classifier of $\mathcal{M}$, and nowhere in the proof [257] of Proposition 47 is the classifier referred to. Thus, for the purposes of the proposition, the definition of α_L is arbitrary, and the result can be applied to Algorithm 1. Notice also that since $L \geq \ell_{\text{st}}$ always holds, the result can be applied to each $\mathbf{v}_L, \mathbf{v}'_L$.

Using this result, we prove that the number of neighbours used in each aggregation step can be upper-bounded without changing the consequences on any dataset.

B.2.9 Theorem 23

Note that whilst this theorem proves a similar result to Tena Cucala et al. [256, Theorem 10], the proof of this theorem relies on different techniques and is entirely novel, besides from the use of Proposition 47. The following proof exploits the structure of non-negative bilinear scoring functions.

Theorem. *Let* $\mathcal{M}$ *be a monotonic max-sum-GCN with unbounded activation functions and* f *a non-negative bilinear scoring function. Let* $\mathcal{M}'$ *be the GCN obtained from* $\mathcal{M}$ *by replacing* k_ℓ *with the capacity* $\mathcal{C}_\ell$ *for each* $\ell \in \{1, ..., L\}$*. Then for each dataset* D*, it holds that* $T_{\mathcal{M},f}(D) = T_{\mathcal{M}',f}(D)$*.*

Proof. Let D be a dataset, $a, b \in D$ be constants (where possibly $a = b$ or $a \neq b$), and R a binary predicate. Let $v^a, v^b \in \texttt{enc}(D)$ be the vertices corresponding to a, b respectively. For a vertex $v \in \texttt{enc}(D)$, let $\mathbf{v}_L, \mathbf{v}'_L$ be the labelling of the vertices after layer L by applying $\mathcal{M}, \mathcal{N}'$ to $\texttt{enc}(D)$, respectively.

To prove the required equality, we show that $R(a, b) \in T_{\mathcal{M},f}(D) \iff R(a, b) \in T_{\mathcal{M}',f}(D)$. This is equivalent to proving that $f(R, \mathbf{v}^a_L, \mathbf{v}^b_L) \geq t_f \iff f(R, \mathbf{v}^{a\prime}_L, \mathbf{v}^{b\prime}_L) \geq t_f$, which in turn is equivalent to $\mathbf{v}^a_L \mathbf{M}_R \mathbf{v}^b_L \geq t_f \iff \mathbf{v}^{a\prime}_L \mathbf{M}_R \mathbf{v}^{b\prime}_L \geq t_f$, where $\mathbf{M}_R$ is the relation matrix of R in f.

Backward Direction ($\Longleftarrow$)

As proven at the start of the proof of [257, Theorem 18], $\forall i \in \{1, ..., \delta_L\}$ it holds that $\mathbf{v}_L^a[i] \geq \mathbf{v}_L^{a\prime}[i]$ and $\mathbf{v}_L^b[i] \geq \mathbf{v}_L^{b\prime}[i]$. The proof of this follows trivially from the definition of max-sum aggregation.

Thus, from the monotonicity of f, we have that $f(R, \mathbf{v}_L^a, \mathbf{v}_L^b) \geq f(R, \mathbf{v}_L^{a\prime}, \mathbf{v}_L^{b\prime})$. So if $f(R, \mathbf{v}_L^{a\prime}, \mathbf{v}_L^{b\prime}) \geq t_f$, then also $f(R, \mathbf{v}_L^a, \mathbf{v}_L^b) \geq t_f$.

Forward Direction ($\implies$)

Assume that $\mathbf{v}_L^a \mathbf{M}_R \mathbf{v}_L^b \geq t_f$. First, consider the definition of α_R in Definition 22. If all elements of $\mathbf{M}_R$ are 0 or $\bigcup_i \mathcal{X}_{L,i} = \{0\}$, then $\mathbf{v}_L^a \mathbf{M}_R \mathbf{v}_L^b = 0 = \mathbf{v}_L^{a\prime} \mathbf{M}_R \mathbf{v}_L^{b\prime}$, and thus $\mathbf{v}_L^{a\prime} \mathbf{M}_R \mathbf{v}_L^{b\prime} \geq t_f$ as required.

So instead assume the opposite, in which case we define w, ϵ, α_R as in Definition 22. Recall that $f(R, \mathbf{v}_L^a, \mathbf{v}_L^b), f(R, \mathbf{v}_L^{a\prime}, \mathbf{v}_L^{b\prime})$ are computed as follows:

$$\mathbf{v}_L^a \mathbf{M}_R \mathbf{v}_L^b = \sum_{i=1}^{\delta_L} \sum_{j=1}^{\delta_L} \mathbf{v}_L^a[i] \mathbf{M}_R[i,j] \mathbf{v}_L^b[j] \tag{B.6}$$

$$\mathbf{v}_L^{a\prime} \mathbf{M}_R \mathbf{v}_L^{b\prime} = \sum_{i=1}^{\delta_L} \sum_{j=1}^{\delta_L} \mathbf{v}_L^{a\prime}[i] \mathbf{M}_R[i,j] \mathbf{v}_L^{b\prime}[j] \tag{B.7}$$

We now consider 3 cases:

Case (1): $\exists i, j \in \{1, ..., \delta_L\}$ such that $\mathbf{v}_L^{a\prime}[i] \geq \alpha$ and $\mathbf{M}_R[i,j] \mathbf{v}_L^{b\prime}[j] > 0$.

Then $\alpha \geq \alpha_R$, so $\mathbf{v}_L^{a\prime}[i] \geq \alpha_R$. Also since $\mathbf{M}_R[i,j] > 0$, we have $\mathbf{M}_R[i,j] \geq w$, the least non-zero element of $\mathbf{M}_R$. And since $\mathbf{v}_L^{b\prime}[j] > 0$, we have $\mathbf{v}_L^{b\prime}[j] \geq \epsilon$, the least non-zero element of $\bigcup_i \mathcal{X}_{L,i}$. Now from the definition of α_R, we have $\alpha_R \cdot w \cdot \epsilon \geq t_f$. Subbing into this equation, we obtain $\mathbf{v}_L^{a\prime}[i] \mathbf{M}_R[i,j] \mathbf{v}_L^{b\prime}[j] \geq t_f$, which implies $\mathbf{v}_L^{a\prime} \mathbf{M}_R \mathbf{v}_L^{b\prime} \geq t_f$, as required.

Case (2): $\exists i, j \in \{1, ..., \delta_L\}$ such that $\mathbf{v}_L^{b\prime}[j] \geq \alpha$ and $\mathbf{v}_L^{a\prime}[i] \mathbf{M}_R[i,j] > 0$.

This case is very similar to the first. $\alpha \geq \alpha_R$, so $\mathbf{v}_L^{b\prime}[j] \geq \alpha_R$. Also since $\mathbf{M}_R[i,j] > 0$, we have $\mathbf{M}_R[i,j] \geq w$, the least non-zero element of $\mathbf{M}_R$. And since $\mathbf{v}_L^{a\prime}[i] > 0$, we have $\mathbf{v}_L^{a\prime}[i] \geq \epsilon$, the least non-zero element of $\bigcup_i \mathcal{X}_{L,i}$. Now from the definition of α_R, we have $\alpha_R \cdot w \cdot \epsilon \geq t_f$. Subbing into this equation, we obtain $\mathbf{v}_L^{b\prime}[j] \mathbf{M}_R[i,j] \mathbf{v}_L^{a\prime}[i] \geq t_f$, which implies $\mathbf{v}_L^{a\prime} \mathbf{M}_R \mathbf{v}_L^{b\prime} \geq t_f$, as required.

Case (3): $\forall i, j \in \{1, ..., \delta_L\}$, we have ($\mathbf{v}_L^a[i] = \mathbf{v}_L^{a\prime}[i]$ or $\mathbf{M}_R[i,j]\mathbf{v}_L^{b\prime}[j] = 0$) and ($\mathbf{v}_L^b[j] = \mathbf{v}_L^{b\prime}[j]$ or $\mathbf{v}_L^{a\prime}[i]\mathbf{M}_R[i,j] = 0$).

To see that the above covers all remaining options, recall that, from Proposition 47, for all $v \in \texttt{enc}(D), \forall i \in \{1, ..., \delta_L\}$, either $\mathbf{v}_L[i] = \mathbf{v}'_L[i]$ or $\mathbf{v}'_L[i] \geq \alpha$.

The conditions in the case imply that $\forall i, j \in \{1, ..., \delta_L\}$, either ($\mathbf{v}_L^a[i] = \mathbf{v}_L^{a\prime}[i]$ and $\mathbf{v}_L^b[j] = \mathbf{v}_L^{b\prime}[j]$) or $\mathbf{v}_L^{a\prime}[i]\mathbf{M}_R[i,j]\mathbf{v}_L^{b\prime}[j] = 0$. Now let $i, j \in \{1, ..., \delta_L\}$. We consider a variety of subcases, proving that, in any of them, the following equality holds:

$$\mathbf{v}_L^a[i]\mathbf{M}_R[i,j]\mathbf{v}_L^b[j] = \mathbf{v}_L^{a\prime}[i]\mathbf{M}_R[i,j]\mathbf{v}_L^{b\prime}[j]. \tag{B.8}$$

If $\mathbf{v}_L^a[i] = \mathbf{v}_L^{a\prime}[i]$ and $\mathbf{v}_L^b[j] = \mathbf{v}_L^{b\prime}[j]$, Equation (B.8) holds. So instead consider $\mathbf{v}_L^{a\prime}[i]\mathbf{M}_R[i,j]\mathbf{v}_L^{b\prime}[j] = 0$, which is necessitated in the alternative. At least one of these terms must be zero.

If $\mathbf{M}_R[i,j] = 0$, Equation (B.8) trivially holds. Likewise, if $\mathbf{v}_L^a[i] = 0$ or $\mathbf{v}_L^b[j] = 0$, Equation (B.8) trivially holds, so assume to the contrary that both are non-zero. Now if $\mathbf{v}_L^{a\prime}[i] = 0$, then $\mathbf{v}_L^{a\prime}[i] \neq \mathbf{v}_L^a[i]$, so we conclude from Equation (B.8) that $\mathbf{v}_L^{a\prime}[i] \geq \alpha$. This is a contradiction, since $\alpha > 0$. We reach a similar contradiction if $\mathbf{v}_L^{a\prime}[i] = 0$.

Thus, $\forall i, j \in \{1, ..., \delta_L\}$, Equation (B.8) holds. From this it trivially follows that $\mathbf{v}_L^a\mathbf{M}_R\mathbf{v}_L^b = \mathbf{v}_L^{a\prime}\mathbf{M}_R\mathbf{v}_L^{b\prime}$. Then since $\mathbf{v}_L^a\mathbf{M}_R\mathbf{v}_L^b \geq t_f$, so also $\mathbf{v}_L^{a\prime}\mathbf{M}_R\mathbf{v}_L^{b\prime} \geq t_f$, as required.

□

B.2.10 Theorem 24

We can finally prove the main result for monotonic max-sum-GCNs and non-negative bilinear scoring functions, which relies on Theorem 23 and is similar to Theorem 20. The theorem shows that an equivalent program can be constructed for any monotonic max-sum-GCN $\mathcal{M}$ and non-negative bilinear scoring function f using rules taken from the finite space of all $(L, |\text{Col}| \cdot \delta_{\mathcal{M}} \cdot \mathcal{C}_{\mathcal{M},f})$-tree-like rules.

The proof of this theorem uses the same techniques as in Tena Cucala et al. [256, Theorem 13], but differs in the same way that Theorem 20 does: due to (1) the inclusion of two variables x, y (with corresponding vertices and constants) in the base case, (2) the necessity of having two cases throughout the proof, to cover whether the binary fact corresponding to the head is of the form $R(s,t)$ for distinct constants s,t or of the form $R(s,s)$, and (3) the use of the monotonicity of the scoring function. Whilst this proof is similar to that of Theorem 20, I include it for the sake of completeness. I will again use footnotes to highlight where the proof diverges from that of Tena Cucala et al. [256, Theorem 13].

Theorem 48. *Let $\mathcal{M}$ be a monotonic max-sum-GCN with unbounded activation functions and f a non-negative bilinear scoring function. Let $\mathcal{P}_{\mathcal{M}}$ be the Datalog program containing, up to variable renaming, each $(L, |Col| \cdot \delta_{\mathcal{M}} \cdot \mathcal{C}_{\mathcal{M},f})$-tree-like rule that is sound for $(\mathcal{M}, f)$, where $\delta_{\mathcal{M}} = max(\delta_0, ..., \delta_L)$. Then $T_{\mathcal{M},f}$ and $\mathcal{P}_{\mathcal{M}}$ are equivalent.*

Proof. We show that $T_{\mathcal{M},f}(D) = T_{P_{\mathcal{M}}}(D)$ holds for every dataset D. By definition of $P_{\mathcal{M}}$, each rule in $P_{\mathcal{M}}$ is sound for $(\mathcal{M}, f)$, so we have $T_r(D) \subseteq T_{\mathcal{M},f}(D)$ for every $r \in P_{\mathcal{M}}$, and thus $T_{P_{\mathcal{M}}}(D) \subseteq T_{\mathcal{M},f}(D)$.

So we let D be a dataset and prove $T_{\mathcal{M},f}(D) \subseteq T_{P_{\mathcal{M}}}(D)$. First, let $\mathcal{M}'$ be the GCN obtained from $\mathcal{M}$ by replacing k_ℓ with the capacity $\mathcal{C}_\ell$ for each $\ell \in \{1, ..., L\}$. Then from Theorem 23, it holds that $T_{\mathcal{M},f}(D) = T_{\mathcal{M}',f}(D)$, so it suffices to show that $T_{\mathcal{N}',f}(D) \subseteq T_{P_{\mathcal{M}}}(D)$.

Let α be an arbitrary binary fact in $T_{\mathcal{N}',f}(D)$. Observe that α can be of the form $R(s,t)$, for distinct constants s,t, or of the form $R(s,s)$, for a constant s. We will denote these two cases as **(Case 1)** and **(Case 2)**, respectively. We will prove both cases simultaneously, making a separate argument when each case needs to be treated differently. [6]

We construct a $(L, |\text{Col}| \cdot \delta_{\mathcal{M}} \cdot \mathcal{C}_{\mathcal{M},f})$-tree-like rule r (potentially with inequalities), such that $\alpha \in T_r(D)$ and r is sound for $(\mathcal{M}', f)$. Since from Theorem 23, $T_{\mathcal{M},f}(D_1) =$

[6] Tena Cucala et al. [256, Theorem 13] considers only one case.

$T_{\mathcal{M}',f}(D_1)$ for every dataset D_1, r being sound for $(\mathcal{M}', f)$ implies that r is sound for $(\mathcal{M}, f)$. So, together, these will imply that $\alpha \in T_{P_{\mathcal{M}}}(D)$, as required for the proof.

Let $G = (V, E, \lambda)$ be the canonical encoding of D and let $\lambda_0, ..., \lambda_L$ the vertex labelling functions arising from the application of $\mathcal{M}'$ to G.

Setup of Construction

We now construct a binary atom H, a conjunction Γ, a substitution ν from the variables of Γ to the set of constants in D, and a graph U (without vertex labels) where each vertex in U is of the form u^x for x a variable and each edge has a colour in Col. We also construct mappings $M_{c,\ell,j} : U \to 2^V$ for each $c \in \text{Col}$, $1 \leq \ell \leq L$, and $1 \leq j \leq \delta_{\ell-1}$. Intuitively, $M_{c,\ell,j}(u^x)$ represents the c-coloured neighbours of v^t that contribute to the result of the max-sum aggregation in vector index j at layer ℓ (where v^t is the vertex corresponding to the constant $\nu(x)$). We initialize Γ as the empty conjunction, and we initialize ν and each $M_{c,\ell,j}$ as empty mappings.

We will assign to each vertex in U a *level* between 0 and L. We will identify a single vertex from U as the *first root* vertex. Furthermore, if α is of the form $R(s,t)$, then we identify another distinct vertex from U as the *second root* vertex. [7]

In the remainder of the proof, we use letters s, t for constants in D, letters x, y for variables, letters v, w for the vertices in V, and (possibly indexed) letter u for the vertices in U. Our construction is by induction from level L down to level 1. The base case defines one or two vertices of level L. Then, for each $1 \leq \ell \leq L$, the induction step considers the vertices of level ℓ and defines new vertices of level $\ell - 1$.

Construction: Base Case

(Case 1) If α is of the form $R(s,t)$ for a binary predicate R, then V contains vertices v^s, v^t. We introduce fresh variables x, y, and define $\nu(x) = s, \nu(y) = t$; we define $H = R(x, y)$; we introduce vertices u^x, u^y of level L; and we make u^x the first

[7] Tena Cucala et al. [256, Theorem 13] has only a single root vertex.

root vertex and u^y the second root vertex. Finally, we extend Γ with atom $U(x)$ for each $U(s) \in D$, and also extend Γ with atom $U(y)$ for each $U(t) \in D$. [8]

(Case 2) If on the other hand, α is of the form $R(s, s)$, then V contains a vertex v^s. We introduce fresh variables x, y, and define $\nu(x) = s, \nu(y) = s$; we define $H = R(x, y)$; we introduce vertices u^x, u^y of level L; and we make u^x the first root vertex and u^y the second root vertex. Finally, we extend Γ with atoms $U(x)$ and $U(y)$ for each $U(s) \in D$.

Construction: Induction Step

For the induction step, consider $1 \leq \ell \leq L$ and assume that all vertices of level greater than or equal to ℓ have been already defined. We then consider each vertex of the form u^x of level ℓ. Let $t = \nu(x)$. For each colour $c \in \text{Col}$, each layer $1 \leq \ell' < \ell$, and each dimension $j \in \{1, ..., \delta_{\ell'-1}\}$, let

$$M_{c,\ell',j}(u^x) = \{w \mid (v^t, w) \in E^c \text{ and } \mathbf{w}_{\lambda_{\ell'}}[j] \tag{B.9}$$

$$\text{contributes to max-}\mathcal{C}_{\ell'}\text{-sum}(\{\!\{\mathbf{w}_{\lambda_{\ell'}}[j] \mid (v^t, w) \in E^c\}\!\})\}. \tag{B.10}$$

Intuitively, each w picks out a c-coloured neighbour of v^t that is among the $\mathcal{C}_{\ell'}$ largest in feature vector index j at layer ℓ'. Note that at least one such set exists; however, it may not be unique, in which case any set satisfying Equation (B.9) may be chosen. Also, each vertex of $M_{c,\ell',j}(u^x)$ is of the form v^{s_n} for some constant s_n in D, where $s_n \neq s_m$ for all $1 \leq n < m \leq |M_{c,\ell',j}(u^x)|$.

We then introduce a fresh variable y_n and define $\nu(y_n) = s_n$. We introduce a vertex u^{y_n} of level $\ell - 1$ and an edge $E^c(u^x, u^{y_n})$ to U. Finally, we append to Γ the conjunction

$$\bigwedge_{n=1}^{|W|} \left(E^c(x, y_n) \ \wedge \bigwedge_{U(s_n) \in D} U(y_n) \right) \tag{B.11}$$

$$\wedge \bigwedge_{1 \leq n < m \leq |W|} y_n \not\approx y_m, \tag{B.12}$$

[8] Again, Tena Cucala et al. [256, Theorem 13] considers only one case and a single root vertex.

where $W = M_{c,\ell',j}(u^x)$ and U stands in for arbitrary unary predicates. Since $M_{c,\ell',j}(u^x)$ contains at most $\mathcal{C}_{\ell'}$ vertices, this step adds at most $|\text{Col}| \cdot \delta_{\ell'-1} \cdot \mathcal{C}_{\ell'} \cdot \ell$ new successors of u^x. This completes the inductive construction.

Rule Shape and Immediate Consequences

At this point in the proof, Γ consists of a conjunction of two formulas, one of which is $(L, |\text{Col}| \cdot \delta_{\mathcal{M}} \cdot \mathcal{C}_{\mathcal{M},f})$-tree-like for x and the other of which is $(L, |\text{Col}| \cdot \delta_{\mathcal{M}} \cdot \mathcal{C}_{\mathcal{M},f})$-tree-like for y (where x, y were defined in the base case). We also have $H = R(x, y)$, so $\Gamma \to H$ is a $(L, |\text{Col}| \cdot \delta_{\mathcal{M}} \cdot \mathcal{C}_{\mathcal{M},f})$-tree-like rule. We define this rule to be r. [9]

Furthermore, the construction of ν means $D \models \Gamma\nu$. So we have $H\nu \in T_r(D)$, with $H\nu = \alpha$, so $\alpha \in T_r(D)$, as required.

Note that if s, t are such that they are not mentioned in any fact in D, then Γ is empty, and r will then have the form $\top \to R(x, y)$. Under our definitions, r is still a $(L, |\text{Col}| \cdot \delta_{\mathcal{M}} \cdot \mathcal{C}_{\mathcal{M},f})$-tree-like rule in this case, since $\top$ is tree-like for x and y. This applies similarly if only one of s, t is not mentioned in any fact in D.

Rule Soundness

To complete the proof, we show that r is sound for $(\mathcal{M}', f)$. For this purpose, let D' be an arbitrary dataset and α' an arbitrary ground atom such that $\alpha' \in T_r(D')$. Then there exists a substitution ν' such that $D' \models \Gamma\nu'$.

Let the graph $G = (V', E', \lambda')$ be the canonical encoding of D' and $\lambda'_0, ..., \lambda'_L$ the functions labelling the vertices of G' when $\mathcal{M}'$ is applied to it. We will use letters p, q, q' to denote vertices of V'.

We prove the following statement by induction, which relies on the monotonicity of $\mathcal{M}'$: for each $0 \leq \ell \leq L$ and each vertex u^x of U whose level is at least ℓ, we have $\mathbf{v}_{\lambda_\ell}[i] \leq \mathbf{p}_{\lambda'_\ell}[i]$ for each $i \in \{1, ..., \delta_\ell\}$, where $v = v^{\nu(x)}$ and $p = v^{\nu'(x)}$.

Monotonicity: Base Case

[9] In Tena Cucala et al. [256, Theorem 13], Γ is only a single formula, rather than a conjunction.

For the base case, $\ell = 0$, consider an arbitrary $1 \leq i \leq \delta_0$ and $u^x \in U$. Let $v = v^{\nu(x)}$ and $p = v^{\nu'(x)}$. Since $\ell = 0$, values $\mathbf{v}_{\lambda_0}[i], \mathbf{p}_{\lambda'_0}[i]$ come from the canonical encoding, so $\mathbf{v}_{\lambda_0}[i] \in \{0, 1\}$ and $\mathbf{p}_{\lambda'_0}[i] \in \{0, 1\}$.

Thus, we prove that $\mathbf{v}_{\lambda_0}[i] = 1$ implies $\mathbf{p}_{\lambda'_0}[i] = 1$. By definition of the canonical encoding, $\mathbf{v}_{\lambda_0}[i] = 1$ implies $U_i(x\nu) \in D$. From the construction of Γ, we have $U_i(x) \in \Gamma$. Also, by definition of ν', we have $D' \models \Gamma\nu'$. So $U_i(x\nu') \in D'$ and $\mathbf{p}_{\lambda'_0}[i] = 1$, as required.

Monotonicity: Induction Step

For the induction step, assume that the property holds for some $\ell - 1$. Consider an arbitrary vertex $u^x \in U$ with level at least ℓ, $c \in \text{Col}$, and $i \in \{1, ..., \delta_i\}$. Let $v = v^{\nu(x)}$ and $p = v^{\nu'(x)}$. Then $\mathbf{v}_{\lambda_\ell}[i], \mathbf{p}_{\lambda'_\ell}[i]$ are computed as follows:

$$\mathbf{v}_{\lambda_\ell}[i] = \sigma_\ell\Bigg(\mathbf{b}_\ell[i] + \sum_{j=1}^{\delta_{\ell-1}} \mathbf{A}_\ell[i,j]\mathbf{v}_{\lambda_{\ell-1}}[j] + \sum_{c\in\text{Col}} \tag{B.13}$$

$$\sum_{j=1}^{\delta_{\ell-1}} \mathbf{B}_\ell[i,j]\text{max-}\mathcal{C}_\ell\text{-sum}\{\!\{\mathbf{w}_{\lambda_{\ell-1}}[j] \mid (v,w) \in E^c\}\!\}\Bigg) \tag{B.14}$$

$$\mathbf{p}_{\lambda'_\ell}[i] = \sigma_\ell\Bigg(\mathbf{b}_\ell[i] + \sum_{j=1}^{\delta_{\ell-1}} \mathbf{A}_\ell[i,j]\mathbf{p}_{\lambda'_{\ell-1}}[j] + \sum_{c\in\text{Col}} \tag{B.15}$$

$$\sum_{j=1}^{\delta_{\ell-1}} \mathbf{B}_\ell[i,j]\text{max-}\mathcal{C}_\ell\text{-sum}\{\!\{\mathbf{q}'_{\lambda'_{\ell-1}}[j] \mid (p,q') \in E'^c\}\!\}\Bigg) \tag{B.16}$$

The induction assumption ensures $\mathbf{v}_{\lambda_{\ell-1}}[j] \leq \mathbf{p}_{\lambda'_{\ell-1}}[j]$ for each $1 \leq j \leq \delta_{\ell-1}$. Also, for each $c \in \text{Col}$ and $1 \leq j \leq \delta_{\ell-1}$, we have $\text{max-}\mathcal{C}_\ell\text{-sum}\{\!\{\mathbf{w}_{\lambda_{\ell-1}}[j] \mid (v,w) \in E^c\}\!\} = \sum_{w\in W} \mathbf{w}_{\lambda_{\ell-1}}[j]$, where $W = M_{c,\ell-1,j}(u^x)$.

Recall that W has the form $\{v^{s_1}, ..., v^{s_{|W|}}\}$, where $s_1, ..., s_{|W|}$ are constants in D Furthermore, by construction of U, there are $|W|$ distinct vertices $u^{y_1}, ..., u^{y_{|W|}}$ in U of level $\ell - 1$ such that $\nu(y_n) = s_n$ and $E^c(u^x, u^{y_n})$ is in U for each $1 \leq n \leq |W|$.

Furthermore, Γ contains the atoms $E^c(x, y_1), ..., E^c(x, y_{|W|})$ and inequalities $y_n \not\approx y_m$ for $1 \leq n < m \leq |W|$. We then have $E^c(\nu'(x), \nu'(y_n)) \in D'$ and

$\nu'(y_n) \neq \nu'(y_m)$ for $1 \leq n < m \leq |W|$. Thus, $W' = \{v^{\nu'(y_1)}, ..., v^{\nu'(y_{|W|})}\}$ is a set of $|W|$ distinct c-neighbours of $v^{\nu'(x)}$ in G'.

The induction assumption means that for $w = v^{\nu(y_n)}$ and $q = v^{\nu'(y_n)}$, we have $\mathbf{w}_{\lambda_{\ell-1}}[j] \leq \mathbf{q}_{\lambda'_{\ell-1}}[j]$. So $\sum_{w \in W} \mathbf{w}_{\lambda_{\ell-1}}[j] \leq \sum_{q \in W'} \mathbf{q}_{\lambda'_{\ell-1}}[j]$.

Thus, by the computations in Equation (B.13) and Equation (B.15), and since all elements of $\mathbf{A}_\ell$ and each $\mathbf{B}_\ell^c$ are non-negative and σ_ℓ is monotonically increasing, we have $\mathbf{v}_{\lambda_\ell}[i] \leq \mathbf{p}_{\lambda'_\ell}[i]$, as required.

Conclusion

Recall that $\alpha' \in T_r(D')$, so α' has the form $R(s', t')$ with $s' = \nu'(x)$ and $t' = \nu'(y)$. [10]

(Case 1) if α has the form $R(s, t)$, then $R(s, t) \in T_{\mathcal{M}',f}(D)$ for $s = \nu(x)$ and $t = \nu(y)$, since $\alpha = R(s, t)$ is the original binary fact we assumed that the model produces from D.

Now let $v = v^s, w = v^t, p = v^{s'}, q = v^{t'}$. Then $R(s, t) \in T_{\mathcal{M}',f}(D)$ implies $f(R, \mathbf{v}_{\lambda_L}, \mathbf{w}_{\lambda_L}) \geq t_f$.

The above property ensures that $\mathbf{v}_{\lambda_L}[i] \leq \mathbf{p}_{\lambda'_L}[i]$ and $\mathbf{w}_{\lambda_L}[i] \leq \mathbf{q}_{\lambda'_L}[i]$ for all $i \in \{1, ..., \delta_L\}$. Thus, since f is monotonically increasing, we have $f(R, \mathbf{v}_{\lambda_L}, \mathbf{w}_{\lambda_L}) \leq f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L})$.

So $f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L}) \geq t_f$, which implies that $R(s', t') \in T_{\mathcal{M}',f}(D')$, as required to show the soundness of the rule.

(Case 2) if on the other hand, α has the form $R(s, s)$, then $R(s, s) \in T_{\mathcal{M}',f}(D)$ for $s = \nu(x)$, since $\alpha = R(s, s)$ is the original binary fact we assumed that the model produces from D.

Now let $v = v^s, p = v^{s'}, q = v^{t'}$. Then $R(s, s) \in T_{\mathcal{M}',f}(D)$ implies that $f(R, \mathbf{v}_{\lambda_L}, \mathbf{v}_{\lambda_L}) \geq t_f$. Recall that $\nu(x) = \nu(y) = s$, so we have $v = v^{\nu(x)} = v^{\nu(y)}$, $p = v^{\nu'(x)}$, and $q = v^{\nu'(y)}$.

[10] Here, the proof once again diverges from that of Tena Cucala et al. [256, Theorem 13], since we again need to consider multiple cases, as well as exploit the monotonicity of the scoring function.

Thus, the above property ensures that $\mathbf{v}_{\lambda_L}[i] \leq \mathbf{p}_{\lambda'_L}[i]$ and $\mathbf{v}_{\lambda_L}[i] \leq \mathbf{q}_{\lambda'_L}[i]$ for all $i \in \{1, ..., \delta_L\}$. Thus, since f is monotonically increasing, we have $f(R, \mathbf{v}_{\lambda_L}, \mathbf{v}_{\lambda_L}) \leq f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L})$.

So $f(R, \mathbf{p}_{\lambda'_L}, \mathbf{q}_{\lambda'_L}) \geq t_f$, which implies that $R(s', t') \in T_{\mathcal{M}',f}(D')$, as required to show the soundness of the rule. □

B.3 Full Experiment Results

Experimental findings for monotonic sum-GCNs are given in Table B.1. Randomly sampled sound rules, with one or two body atoms, from monotonic max-GCNs using a RESCAL decoder are shown in Table B.2 and Table B.3.

Dataset	Decoder	Model	%Acc	%Prec	%Rec	%F1	AUPRC	Loss	%SO	#1B	#2B
WN-hier	DistMult	Standard	90.04	89.37	90.9	90.12	0.9284	0.11	-	-	-
		Monotonic	84.81	77.85	97.3	86.5	0.8464	1.48	76	186	28644
	RESCAL	Standard	92.51	90.34	95.28	92.71	0.9493	0.09	-	-	-
		Monotonic	86.18	79.4	97.72	87.61	0.827	1.34	72	150	24000
	NAM	Standard	91.64	89.73	94.06	91.83	0.9325	0.08	-	-	-
		Monotonic	78.4	72.66	93.68	81.54	0.7961	1.42	92	306	43699
	TuckER	Standard	92.11	90.74	93.84	92.25	0.9462	0.09	-	-	-
		Monotonic	79.84	71.66	99.42	83.21	0.749	1.50	92	235	36357
WN-sym	DistMult	Standard	96.7	96.08	97.38	96.72	0.9944	0.08	-	-	-
		Monotonic	91.62	93.39	89.58	91.45	0.9596	1.31	12	21	3458
	RESCAL	Standard	96.09	95.52	96.72	96.11	0.9928	0.06	-	-	-
		Monotonic	93.11	88.73	98.76	93.48	0.9483	1.18	80	72	11287
	NAM	Standard	94.98	93.65	96.52	95.05	0.9877	0.06	-	-	-
		Monotonic	84.44	78.83	94.4	85.85	0.8621	0.94	88	223	29578
	TuckER	Standard	95.92	95.68	96.2	95.93	0.9919	0.06	-	-	-
		Monotonic	88.02	82.94	96	88.92	0.8934	1.31	80	96	15197
WN-cup_nmhier	DistMult	Standard	75.06	73.84	78.09	75.74	0.8234	0.17	-	-	-
		Monotonic	63.87	61.76	72.77	66.81	0.6252	1.88	0	182	27048
	RESCAL	Standard	78.93	76.55	83.42	79.83	0.8365	0.14	-	-	-
		Monotonic	61.98	60.26	70.21	64.77	0.6171	1.82	20	187	28678
	NAM	Standard	78	74.09	86.33	79.67	0.823	0.14	-	-	-
		Monotonic	60.02	57.78	85.96	68.39	0.5501	2.02	100	487	72270
	TuckER	Standard	79.07	75.29	86.6	80.53	0.84	0.14	-	-	-
		Monotonic	59.2	60.05	59.31	58.41	0.5808	1.84	20	140	21367
fb237v1	DistMult	Standard	60.5	56.4	93.5	70.32	0.9579	0.02	-	-	-
		Monotonic	79.65	76.3	86.1	80.89	0.6902	0.81	-	41201	-
	RESCAL	Standard	52.9	51.51	98.9	67.74	0.962	0.02	-	-	-
		Monotonic	89.85	94.7	84.5	89.28	0.6106	0.72	-	6608	-
	NAM	Standard	58.45	55.18	92.9	69.15	0.9651	0.01	-	-	-
		Monotonic	56.85	53.88	97.6	69.39	0.9652	0.32	-	161875	-
	TuckER	Standard	59.65	55.58	96.6	70.55	0.9534	0.01	-	-	-
		Monotonic	89.5	90.64	88.2	89.38	0.629	0.86	-	9233	-
WN18RRv1	DistMult	Standard	72.91	65.47	99.52	78.83	0.8995	0.01	-	-	-
		Monotonic	91.21	85.05	100	91.92	0.7914	0.79	-	97	11929
	RESCAL	Standard	84.67	79.14	98.79	87.36	0.9175	0.01	-	-	-
		Monotonic	94.97	92.06	98.42	95.14	0.7878	0.77	-	50	6143
	NAM	Standard	64.67	59.68	100	74.43	0.898	0.01	-	-	-
		Monotonic	77.58	69.12	100	81.72	0.9198	0.12	-	201	21137
	TuckER	Standard	93.33	89.11	98.79	93.69	0.9285	0.01	-	-	-
		Monotonic	94.73	92.03	97.94	94.89	0.7861	0.79	-	51	6309
nellv1	DistMult	Standard	58.24	60.2	44.94	51.27	0.9392	0.09	-	-	-
		Monotonic	60.35	55.77	100	71.61	0.8937	1.43	-	794	153341
	RESCAL	Standard	65.65	67.7	63.76	65.02	0.9368	0.03	-	-	-
		Monotonic	83.06	74.77	100	85.54	0.7208	1.15	-	305	58784
	NAM	Standard	45.53	44.48	16.94	22.81	0.918	0.03	-	-	-
		Monotonic	48.12	48.41	60.24	53.65	0.9292	1.65	-	862	146403
	TuckER	Standard	61.88	66.46	51.53	57.17	0.9392	0.03	-	-	-
		Monotonic	70	62.66	100	77	0.7511	1.30	-	395	78152

Table B.1: Results for sum-GCNs. Loss is from the final epoch on the training set. AUPRC is from the validation set. Other metrics are computed on the test set. %SO is the percentage of LogInfer rules that are sound for the model. #1B and #2B are the number of sound rules with one and two body atoms respectively.

Dataset	Sample Sound Rules
WN-hier	_similar_to(x,y) `implies` _hypernym(x,y) _has_part(x,y) `and` _member_meronym(z,a) `implies` _hypernym(x,a) _verb_group(x,y) `implies` _derivationally_related_form(y,y) _hypernym(y,x) `and` _similar_to(z,y) `implies` _hypernym(x,z) _has_part(x,y) `implies` _hypernym(y,x) _member_of_domain_usage(y,x) `and` _similar_to(y,z) `implies` _member_meronym(y,x) _synset_domain_topic_of(x,y) `implies` _hypernym(x,y) _instance_hypernym(y,x) `and` _synset_domain_topic_of(z,y) `implies` _has_part(y,z)
WN-sym	_synset_domain_topic_of(x,y) `implies` _synset_domain_topic_of(y,x) _hypernym(y,x) `and` _similar_to(z,y) `implies` _hypernym(x,z) _member_of_domain_usage(x,y) `implies` _hypernym(x,y) _has_part(x,y) `and` _member_meronym(z,a) `implies` _hypernym(x,a) _instance_hypernym(x,y) `implies` _instance_hypernym(y,y) _also_see(y,x) `and` _also_see(z,y) `implies` _also_see(y,y) _similar_to(x,x) `implies` _similar_to(x,x) _hypernym(x,y) `and` _verb_group(z,a) `implies` _hypernym(a,y)
WN-cup_nmhier	_derivationally_related_form(x,y) `implies` _derivationally_related_form(y,x) _has_part(x,y) `and` _member_meronym(z,a) `implies` _hypernym(x,a) _member_of_domain_usage(x,y) `implies` _member_of_domain_usage(y,y) _member_of_domain_region(y,y) `and` _member_of_domain_region(z,y) `implies` _hypernym(y,y) _instance_hypernym(x,y) `implies` _has_part(y,y) _hypernym(x,y) `and` _instance_hypernym(y,z) `implies` _instance_hypernym(x,z) _member_meronym(x,y) `implies` _hypernym(x,y) _member_of_domain_region(x,y) `and` _hypernym(y,z) `implies` _has_part(y,y)

Table B.2: Randomly sampled sound rules from the LogInfer datasets, with monotonic max-GCNs using a RESCAL decoder.

Dataset	Sample Sound Rules
fb237v1	/sports/sports_position/players./sports/sports_team_roster/team(x,y) `implies` /people/person/places_lived./people/place_lived/location(x,x) /education/educational_degree/people_with_this_degree./education/education/institution(x,y) `implies` /location/location/contains(y,x) /film/film/release_date_s./film/film_regional_release_date/film_release_region(x,y) `implies` award/award_nominee/award_nominations./award/award_nomination/ award_nominee(x,y) /location/country/official_language(x,y) `implies` /location/hud_county_place/county(y,y) /travel/travel_destination/how_to_get_here./travel/transportation/mode_of_transportation(x,y) `implies` /location/ hud_county_place/county(y,x)
WN18RRv1	_also_see(x,y) `implies` _derivationally_related_form(x,x) _also_see(y,x) `and` _hypernym(y,y) `implies` _also_see(x,x) _member_meronym(x,y) `implies` _derivationally_related_form(x,y) _verb_group(x,y) `and` _has_part(z,y) `implies` _derivationally_related_form(y,x) _synset_domain_topic_of(x,y) `implies` _derivationally_related_form(y,x) _similar_to(x,y) `and` _has_part(z,a) `implies` _derivationally_related_form(y,x) _derivationally_related_form(x,y) `and` _verb_group(y,y) `implies` _hypernym(y,y) _hypernym(x,y) `implies` _derivationally_related_form(y,y)
nellv1	concept:organizationhiredperson(x,y) `implies` concept:organizationhiredperson(y,y) concept:agentbelongstoorganization(y,x) `and` concept:organizationterminatedperson(z,y) `implies` concept:worksfor(x,z) concept:subpartoforganization(x,x) `implies` concept:acquired(x,x) concept:televisionstationaffiliatedwith(y,x) `and` concept:organizationhiredperson(z,y) `implies` concept:organizationhiredperson(x,y) concept:organizationterminatedperson(x,y) `implies` concept:topmemberoforganization(y,y) concept:headquarteredin(x,y) `and` concept:organizationhiredperson(y,z) `implies` concept:headquarteredin(z,y) concept:subpartoforganization(x,y) `and` concept:subpartoforganization(z,y) `implies` concept:worksfor(x,z)

Table B.3: Randomly sampled sound rules from the benchmark datasets, with monotonic max-GCNs using a RESCAL decoder.

C

Appendices for Chapter 6

Contents

C.1 Full Proofs 222
C.1.1 Lemma 29 222
C.1.2 Proposition 30 224
C.1.3 Theorem 31 225
C.1.4 Theorem 33 228
C.1.5 Theorem 34 233
C.2 More Details About Experiments 235

C.1 Full Proofs

C.1.1 Lemma 29

Lemma. *Let $\mathcal{M}$ be a sum-GCN of L layers, and let D', D be datasets satisfying $D' \subseteq D$. For each vertex $v \in \texttt{enc}(D')$, layer $\ell \in \{0, \ldots, L\}$, and channel $i \in \{1, \ldots, \delta_\ell\}$, the following hold:*

- *If i is stable at layer ℓ, then $\mathbf{v}'_\ell[i] = \mathbf{v}_\ell[i]$;*
- *If i is increasing at layer ℓ, then $\mathbf{v}'_\ell[i] \leq \mathbf{v}_\ell[i]$, and*
- *If i is decreasing at layer ℓ, then $\mathbf{v}'_\ell[i] \geq \mathbf{v}_\ell[i]$,*

where $\mathbf{v}_\ell$ and $\mathbf{v}'_\ell$ are the vectors induced for v in layer ℓ by applying $\mathcal{M}$ to D and D' respectively.

Proof. We show the claim of the lemma by induction on ℓ. The base case $\ell = 0$ holds trivially by the definition of the canonical encoding, the fact that $D' \subseteq D$, and all channels being increasing at layer $\ell = 0$.

For the inductive step, we assume that the claim holds for $\ell - 1 \geq 0$. We prove that it holds at layer ℓ in three parts. Defining $E^c, (E^c)'$ to be the edges of $\texttt{enc}(D), \texttt{enc}(D')$ respectively, we have that $(E^c)' \subseteq E^c$.

Stable Channels

Assume that channel i is stable at layer ℓ. We show that $\mathbf{v}'_\ell[i] = \mathbf{v}_\ell[i]$. Consider the computation of $\mathbf{v}_\ell[i]$ and $\mathbf{v}'_\ell[i]$ as per Equation (2.3). The value $(\mathbf{A}_\ell \mathbf{v}_{\ell-1})[i]$ is the sum over all $j \in \{1, ..., \delta_{\ell-1}\}$ of $\mathbf{A}_\ell[i, j]\mathbf{v}_{\ell-1}[j]$. But since i is stable, each such j that is unstable at layer $\ell - 1$ has $\mathbf{A}_\ell[i, j] = 0$.

Thus, $(\mathbf{A}_\ell \mathbf{v}_{\ell-1})[i]$ is equal to the sum over each channel j that is stable at layer $\ell - 1$. By induction, each such stable j satisfies $\mathbf{v}'_{\ell-1}[j] = \mathbf{v}_{\ell-1}[j]$. Thus, $\mathbf{A}_\ell[i, j]\mathbf{v}'_{\ell-1}[j] = \mathbf{A}_\ell[i, j]\mathbf{v}_{\ell-1}[j]$, and therefore $(\mathbf{A}_\ell \mathbf{v}'_{\ell-1})[i] = (\mathbf{A}_\ell \mathbf{v}_{\ell-1})[i]$.

Furthermore, for all j (both stable and unstable), $\mathbf{B}^c_\ell[i, j] = 0$ for each colour $c \in \text{Col}$. Thus, we can derive that both $(\sum_{c \in \text{Col}} \mathbf{B}^c_\ell \sum_{(v,u) \in (E^c)'} \mathbf{u}'_{\ell-1})[i] = 0$ and $(\sum_{c \in \text{Col}} \mathbf{B}^c_\ell \sum_{(v,u) \in E^c} \mathbf{u}_{\ell-1})[i] = 0$.

This means that $\mathbf{v}'_\ell[i] = \mathbf{v}_\ell[i]$, as required.

Increasing Channels

Assume that channel i is increasing at layer ℓ. We show that $\mathbf{v}'_\ell[i] \leq \mathbf{v}_\ell[i]$. As before, consider the computation of $\mathbf{v}_\ell[i]$ and $\mathbf{v}'_\ell[i]$ as per Equation (2.3). The value $(\mathbf{A}_\ell \mathbf{v}_{\ell-1})[i]$ is the sum over all $j \in \{1, ..., \delta_{\ell-1}\}$ of $\mathbf{A}_\ell[i,j]\mathbf{v}_{\ell-1}[j]$.

Now consider the four possibilities for j at $\ell - 1$ and conditions (1) - (3) in the definitions. If j is stable, then by induction, $v'_{\ell-1}[j] = v_{\ell-1}[j]$. If j is increasing, then from (1) we have $\mathbf{A}_\ell[i,j] \geq 0$ and by induction $v'_{\ell-1}[j] \leq v_{\ell-1}[j]$. If j is decreasing, then from (2) we have $\mathbf{A}_\ell[i,j] \leq 0$ and again by induction $v'_{\ell-1}[j] \geq v_{\ell-1}[j]$. Finally, if j is undetermined, then from (3) we obtain $v'_{\ell-1}[j] = 0 = v_{\ell-1}[j]$.

In any of these cases, we find that $\mathbf{A}_\ell[i,j]\mathbf{v}'_{\ell-1}[j] \leq \mathbf{A}_\ell[i,j]\mathbf{v}_{\ell-1}[j]$, and thus we can conclude that $(\mathbf{A}_\ell \mathbf{v}'_{\ell-1})[i] \leq (\mathbf{A}_\ell \mathbf{v}_{\ell-1})[i]$.

Now let $c \in \text{Col}$ and consider the matrix product involving $\mathbf{B}^c_\ell$, which is a sum over all $j \in \{1, ..., \delta_{\ell-1}\}$. If j is decreasing or undetermined at $\ell - 1$ then from (2) and (3) we have that $\mathbf{B}^c_\ell[i,j] = 0$. Then trivially

$$\mathbf{B}^c_\ell[i,j](\sum_{(v,u)\in(E^c)'} \mathbf{u}'_{\ell-1})[j] \leq \mathbf{B}^c_\ell[i,j](\sum_{(v,u)\in E^c} \mathbf{u}_{\ell-1})[j].$$

On the other hand, if $\mathbf{B}^c_\ell[i,j] \neq 0$, then by elimination j must be increasing or stable at $\ell - 1$, which by induction means that $\mathbf{u}'_{\ell-1}[j] \leq \mathbf{u}_{\ell-1}[j]$ for every node $u \in \texttt{enc}(D')$.

Then since $(E^c)' \subseteq E^c$, we can conclude that

$$(\sum_{(v,u)\in(E^c)'} \mathbf{u}'_{\ell-1})[j] \leq (\sum_{(v,u)\in E^c} \mathbf{u}_{\ell-1})[j].$$

Note that both of these sums are positive. Also, since from (4) we have that $\mathbf{B}^c_\ell[i,j] \geq 0$, we find that for all j,

$$\mathbf{B}^c_\ell[i,j](\sum_{(v,u)\in(E^c)'} \mathbf{u}'_{\ell-1})[j] \leq \mathbf{B}^c_\ell[i,j](\sum_{(v,u)\in E^c} \mathbf{u}_{\ell-1})[j].$$

$$(\mathbf{B}^c_\ell \sum_{(v,u)\in(E^c)'} \mathbf{u}'_{\ell-1})[i] \leq (\mathbf{B}^c_\ell \sum_{(v,u)\in E^c} \mathbf{u}_{\ell-1})[i].$$

and thus that

$$(\sum_{c\in\mathrm{Col}} \mathbf{B}^c_\ell \sum_{(v,u)\in(E^c)'} \mathbf{u}'_{\ell-1})[i] \leq (\sum_{c\in\mathrm{Col}} \mathbf{B}^c_\ell \sum_{(v,u)\in E^c} \mathbf{u}_{\ell-1})[i].$$

Bringing the above together, along with the fact that σ_ℓ is monotonically increasing, we obtain $\mathbf{v}'_\ell[i] \leq \mathbf{v}_\ell[i]$.

Decreasing Channels

This argument is very similar to the one for increasing channels, as the definitions are almost symmetric. If channel i is decreasing at layer ℓ, we obtain that $\mathbf{v}'_\ell[i] \leq \mathbf{v}_\ell[i]$.

This concludes the proof by induction. □

C.1.2 Proposition 30

Proposition. *Let $\mathcal{M}$ be a sum-GCN as in Section 6.1, and let r be a safe rule of the form* (2.1) *where H mentions a unary predicate U_p. Let S be an arbitrary set of as many constants as there are variables in r. Assume channel p in $\mathcal{M}$ is increasing or stable at layer L. Then r is sound for $T_\mathcal{M}$ if and only if $H\nu \in T_\mathcal{M}(D^\nu_r)$ for each substitution ν mapping the variables of r to constants in S and such that $D^\nu_r \models B_i\nu$ for each inequality B_i in r.*

Proof. To prove the soundness of r, consider an arbitrary dataset D. We show that $T_r(D) \subseteq T_\mathcal{M}(D)$. To this end, we consider an arbitrary fact in $T_r(D)$ and show that it is also contained in $T_\mathcal{M}(D)$. By the definition of the immediate consequence operator T_r, this fact is of the form $H\mu$, where μ is a substitution from the variables of r to constants in D satisfying $D \models B_i\mu$ for each body literal B_i of r. Let σ be an arbitrary one-to-one mapping from the co-domain of μ to some subset of S; such a mapping exists because S has at least as many constants as variables in r. Let ν be the composition of μ and σ.

Observe that for each body inequality B_i of r, we have $D_r^\nu \models B_i\nu$ because $D \models B_i\mu$ and σ is injective. Therefore, by the assumption of the proposition, $H\nu \in T_\mathcal{M}(D_r^\nu)$. Now, observe that the result of applying $T_\mathcal{M}$ to a dataset does not depend on the identity of the constants, but only on the structure of the dataset; therefore, $T_\mathcal{M}$ is invariant under one-to-one mappings of constants, and so $H\nu \in T_\mathcal{M}(D_r^\nu)$ implies $H\mu \in T_\mathcal{M}(D_r^\mu)$. Now, let a be the single constant in $H\mu$. Since $D_\mu^r \subseteq D$ by definition of μ, and channel p is increasing or stable at layer L, we can apply Lemma 29 to conclude that $\mathbf{v}'_L[p] \leq \mathbf{v}_L[p]$, for v the node corresponding to a in $\texttt{enc}(D_\mu^r)$. But $H\mu \in T_\mathcal{M}(B\mu)$ implies that $\texttt{cls}_t(\mathbf{v}'_L[p]) = 1$ and so $\mathbf{v}'_L[p] \geq t$. Hence, $\mathbf{v}_L[p] \geq t$, and so $\texttt{cls}_t(\mathbf{v}_L[p]) = 1$, which implies that $H\mu \in T_\mathcal{M}(D)$, as we wanted to show.

If on the other hand, $H\nu \notin T_\mathcal{M}(D_r^\nu)$ for some ν, then $T_r(D_r^\nu) \not\subseteq T_\mathcal{M}(D_r^\nu)$, since $H\nu \in T_r(D_r^\nu)$. Thus, r is not sound for $T_\mathcal{M}$. □

C.1.3 Theorem 31

Theorem. *Safe channels are increasing or stable. Increasing channels are not decreasing. There exist increasing unsafe channels and stable unsafe channels.*

Proof. Consider a sum-GCN $\mathcal{M}$ as in Section 6.1. We prove the Theorem in three parts.

Safe channels are increasing or stable

We prove this by induction on $\ell \in \{0, ..., L\}$: we show that if channel $i \in \{1, ..., \delta_\ell\}$ is safe at layer $\ell \in \{0, ..., L\}$, then it is increasing or stable at layer ℓ.

Base case: $\ell = 0$. This is trivial, since all channels are increasing at layer 0.

For the inductive step, consider some layer $\ell \in \{1, ..., L\}$ such that channel $i \in \{1, ..., \delta_\ell\}$ is safe at layer ℓ. Then the i-th row of each matrix $\mathbf{A}_\ell$ and $\{(\mathbf{B}_\ell^c)\}_{c\in\text{Col}}$ contains only non-negative values and, additionally, the j-th element in each such row is zero for each $j \in \{1, \ldots, \delta_{\ell-1}\}$ such that j is unsafe in layer $\ell - 1$.

We prove that if i is not stable, then it is increasing. To this end, we need to show that conditions (1) - (4) in Definition 28 are satisfied for each $j \in \{1, ..., \delta_{\ell-1}\}$.

Condition (4) holds for any j, since the i-th row of each matrix $\{(\mathbf{B}_\ell^c)\}_{c\in\text{Col}}$ contains only non-negative values.

If j is safe at layer $\ell-1$ then by the inductive hypothesis it is increasing or stable at layer $\ell-1$. So conditions (2) and (3) hold vacuously. Condition (1) holds since its consequent holds.

If on the other hand, j is unsafe at layer $\ell-1$, then we have $\mathbf{A}_\ell[i,j]=0$ and $\mathbf{B}_\ell^c[i,j]=0$ for each $c\in\text{Col}$. So conditions (1) - (3) are satisfied by virtue of their consequents holding.

Increasing channels are not decreasing

Consider a channel $i\in\{1,...,\delta_\ell\}$ at layer $\ell\in\{1,...,L\}$. Assume to the contrary that i is increasing and decreasing at layer ℓ.

Then from Definition 28, i is not stable. Furthermore, for each $j\in\{1,...,\delta_{\ell-1}\}$, if j is not stable at layer $\ell-1$, then it is increasing, decreasing, or undetermined. Thus, from conditions (1) - (3) in the definition, we have that for all non-stable $j\in\{1,...,\delta_{\ell-1}\}$, $\mathbf{A}_\ell[i,j]=0$, since i is both increasing and decreasing.

Finally, from condition (4), we have that $\mathbf{B}_\ell^c[i,j]=0$ for each $c\in\text{Col}$ and each $j\in\{1,...,\delta_{\ell-1}\}$. This satisfies the conditions for stability in Definition 28, implying that channel i is stable at layer ℓ, which is a contradiction.

There exist increasing unsafe channels and stable unsafe channels

For simplicity, we provide examples of sum-GCNs with such unsafe channels using 2-layer GCNs with $\delta_0=\delta_1=\delta_2=2$ and $|\text{Col}|=1$. However, the examples can be extended to sum-GCNs with an arbitrary number of layers, arbitrary δ, and arbitrary number of colours.

Let $c\in\text{Col}$ be the unique colour in Col. First, to show that there exist unsafe channels that are increasing, consider a 2-layer sum-GCN with the following matrices. Let $\mathbf{B}_1^c=\mathbf{B}_2^c$ be 2×2 matrices filled with zeroes. Define $\mathbf{A}_1$ and $\mathbf{A}_2$ as follows:

$$\mathbf{A}_1=\begin{pmatrix}-1 & -1\\ -1 & -1\end{pmatrix},\qquad \mathbf{A}_2=\begin{pmatrix}-1 & -1\\ -1 & -1\end{pmatrix}$$

Trivially, no channels are safe. Also, both channels are decreasing at layer 1, with all negative values in $\mathbf{A}_2$. Thus condition (2) of Definition 28 is satisfied, and both channels are increasing at layer 2.

Next, to show that there exist unsafe channels that are stable, consider a 2-layer sum-GCN with the following matrices. Let $\mathbf{B}_1^c = \mathbf{B}_2^c$ be 2×2 matrices filled with zeroes. Define $\mathbf{A}_1$ and $\mathbf{A}_2$ as follows:

$$\mathbf{A}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \qquad \mathbf{A}_2 = \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix}$$

Both channels are safe at layer 1 and both are unsafe at layer 2, since $\mathbf{A}_2$ only contains non-zero values. Also, both channels are stable at layer 1, with all zero values in $\mathbf{B}_1^c = \mathbf{B}_2^c$. Thus, the conditions for stability in Definition 28 are satisfied, and both channels are stable at layer 2.

The above examples can be extended to an arbitrary number of layers $L \geq 2$, dimensions $\delta_0, ..., \delta_L$, and colour set Col by using the following scheme:

1. For each $\ell \in \{1, ..., L\}$ and $c \in \text{Col}$, let $\mathbf{B}_\ell^c$ a matrix of zeroes.

2. Let every entry in the matrix $\mathbf{A}_L$ be -1.

3. For each $\ell \in \{1, ..., L-1\}$, define the matrix $\mathbf{A}_\ell$ as follows:

   (a) To obtain the case where all channels are both unsafe and increasing at L, let every entry in the matrix $\mathbf{A}_\ell$ be -1, as in the example given above.

   (b) To obtain the case where all channels are both unsafe and stable at L, let every entry in the matrix $\mathbf{A}_\ell$ be 0, also as in the example given above.

□

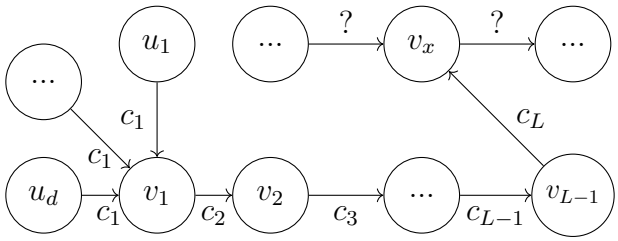


Figure C.1: Succinct representation of the encoding of dataset D_d in the family of datasets used in our counterexample.

C.1.4 Theorem 33

Theorem. *Let $\mathcal{M}$ be a sum-GCN as in Equation* (6.1)*, where σ_ℓ is ReLU for each $1 \leq \ell \leq L-1$, and the co-domain of σ_L contains a number strictly less than the threshold t of the classification function $\texttt{cls}_t$. Then, each rule with head predicate U_p corresponding to an unbounded channel p is unsound for $\mathcal{M}$.*

Proof. Assume to the contrary that there exists some rule r with head $U_p(x)$ that is sound for $T_{\mathcal{M}}$. Let ν be an arbitrary variable substitution grounding the body of r. Since p is unbounded, there exist colours $c_1, c_2, ..., c_L$ and a vector $\mathbf{y}_0$ such that $\mathbf{y}_L[p] < 0$. Let $\mathbf{y}_1, ..., \mathbf{y}_{L-1}$ be defined as in Definition 32, and $\mathbf{y}_L = \mathbf{B}_L^{c_L}\mathbf{y}_{L-1}$.

For each $\ell \in \{0, \ldots, L-1\}$, let $\mathbf{z}_\ell = \mathbf{B}_{\ell+1}^{c_{\ell+1}}\mathbf{y}_\ell$. Note that $\mathbf{y}_{\ell+1} = \mathrm{ReLU}(\mathbf{z}_\ell)$ for $\ell \in \{0, \ldots, L-2\}$.

Dataset Construction

Furthermore, for each $d \in \mathbb{N} \cup \{0\}$, let D_d be the dataset defined as the extension of D_ν^r with:

- $R^{c_1}(a_i, f_1)$ for $i \in \{1, \ldots, d\}$,
- $R^{c_\ell}(f_{\ell-1}, f_\ell)$

for $\ell \in \{2, \ldots, L\}$, with $f_L = a = \nu(x)$, (recall, we are assuming that we derive $U_p(a)$ for some channel p and constant a). To keep our notation simple, for $d \in \mathbb{N} \cup \{0\}$ and $\ell \in \{0, \ldots, L\}$, and constant f_i, let v_i be the node in $\texttt{end}(D)$ associated with f_i and likewise u_i the node associated with a_i. Let v_x be the node associated with a.

Let $\lambda_\ell^d(v_i)$ be the vector assigned to v_i in layer ℓ when applying $\mathcal{M}$ to the encoding of D_d. Furthermore, for each $\ell \in \{1, \ldots, L-1\}$, let $\mathbf{t}_\ell = \lambda_{\ell-1}^0(v_\ell)$, that is, the labelling assigned to v_ℓ in layer $\ell - 1$ when applying $\mathcal{M}$ to the encoding of D_0. Observe that for each $d \geq 1$, v_ℓ is exactly ℓ steps away from each u_i, and so $\lambda_{\ell-1}^d(v_\ell)$ does not depend on d, that is, $\lambda_{\ell-1}^d(v_\ell) = \mathbf{t}_\ell$ for each $d \in \mathbb{N} \cup \{0\}$.

Claim to be Proved by Induction

We next define inductively over $\ell \in \{1, \ldots, L-1\}$ a sequence of natural numbers $d_1 \leq d_2 \leq \cdots \leq d_{L-1}$, and simultaneously we prove the following property for d_ℓ:

(Claim) There exists a vector $\mathbf{g}_\ell$ of dimension δ_ℓ such that, for each $d \geq d_\ell$, it holds that

$$\lambda_\ell^d(v_\ell) = \mathbf{g}_\ell + d \cdot \mathbf{y}_\ell.$$

Note that $\mathbf{g}_\ell$ is fixed before quantifying over d, and so it is independent of d.

To provide an intuitive summary about what we are using each of our variables to track:

- Each $\mathbf{y}_\ell$ is a vector from Definition 32;
- $\mathbf{z}_\ell$ refers to the value of $\mathbf{y}_{\ell+1}$ before the ReLU is applied;
- v_i refers to the node in $\texttt{enc}(D_d)$ corresponding to f_i;
- $\lambda_\ell^d(v_i)$ refers to the label of v_i in the graph $\texttt{enc}(D_d)$ after ℓ layers of $\mathcal{M}$;
- $\mathbf{t}_\ell = \lambda_{\ell-1}^0(v_\ell)$ is the labelling assigned to v_ℓ in layer $\ell - 1$ when applying $\mathcal{M}$ to the encoding of D_0; and
- $\mathbf{g}_\ell$ refers to the non-d component of $\lambda_\ell^d(v_\ell)$.

Base Case

For the base case, we define d_1 as the smallest natural number such that for each $i \in \{1, \ldots, \delta_1\}$ with $\mathbf{z}_0[i] \neq 0$, it holds that

$$d_1 \cdot |\mathbf{z}_0[i]| > |\mathbf{b}_1[i] + (\mathbf{A}_1\mathbf{t}_1)[i]|.$$

Clearly, such number always exists, since $\mathbf{z}_0[i] \neq 0$ implies $|\mathbf{z}_0[i]| > 0$.

Now we prove the claim for $\ell = 1$. First, we have that for each d, $\lambda_1^d(v_1) = \text{ReLU}(\mathbf{b}_1 + \mathbf{A}_1\mathbf{t}_1 + d \cdot \mathbf{z}_0)$. Note that $\mathbf{t}_1$ has all elements equal to 0 (since $\lambda_0^0(v_1)$ does by definition), so we ignore this term. Now, consider an arbitrary $i \in \{1, \ldots, \delta_1\}$. We define $\mathbf{g}_1[i]$ attending to three possible cases:

- If $\mathbf{z}_0[i]$ is negative, we let $\mathbf{g}_1[i] = 0$. To see that the claim holds, consider an arbitrary $d \geq d_1$. Since $\mathbf{z}_0[i]$ is negative, $\mathbf{z}_0[i] \neq 0$; then, the definition of d_1 ensures that $\mathbf{b}_1[i] + d_1 \cdot \mathbf{z}_0[i] < 0$. Since $d \geq d_1$, we then have $\mathbf{b}_1[i] + d \cdot \mathbf{z}_0[i] < 0$ and so $\lambda_1^d(v_1) = 0$. However, $\mathbf{y}_1[i] = \text{ReLU}(\mathbf{z}_0[i]) = 0$. Thus, the claim holds.

- If $\mathbf{z}_0[i] = 0$, we let $\mathbf{g}[i] = \text{ReLU}(\mathbf{b}_1[i])$. To prove the claim, assume again $d \geq d_1$. Since $\mathbf{z}_0[i] = 0$, $\lambda_1^d(v_1)[i] = \text{ReLU}(\mathbf{b}_\ell[i])$, so the claim holds.

- If $\mathbf{z}_0[i]$ is positive, then $\mathbf{g}_1[i] = \mathbf{b}_1[i]$. To show the claim, consider $d \geq d_1$. Since $\mathbf{z}_0[i]$ is positive, $\mathbf{z}_0[i] \neq 0$, and the definition of d_1 ensures that $\mathbf{b}_1[i] + d_1 \cdot \mathbf{z}_0[i] > 0$, but since $d \geq d_1$, we have $\mathbf{b}_1[i] + d \cdot \mathbf{z}_0[i] > 0$, and so $\lambda_1^d(v_1)[i] = \mathbf{b}_1[i] + d \cdot \mathbf{z}_0[i]$. Furthermore, $\mathbf{z}_0[i]$ being positive implies $\mathbf{y}_1[i] = \mathbf{z}_0[i]$, and so the claim holds.

Inductive Step

Consider $\ell \in \{2, \ldots, L-1\}$; suppose that $d_{\ell-1}$ is defined and the corresponding claim holds, that is, there exists $\mathbf{g}_{\ell-1}$ such that for each $d \geq d_{\ell-1}$, it holds that

$$\lambda_{\ell-1}^d(v_{\ell-1}) = \mathbf{g}_{\ell-1} + d_{\ell-1} \cdot \mathbf{y}_{\ell-1}.$$

We now define d_ℓ as the smallest natural number greater than or equal to $d_{\ell-1}$ such that for each $i \in \{1, \ldots, \delta_\ell\}$ with $\mathbf{z}_\ell[i] \neq 0$, it holds that

$$d_\ell \cdot |\mathbf{z}_{\ell-1}[i]| > |\mathbf{b}_\ell[i] + (\mathbf{A}_\ell\mathbf{t}_\ell)[i] + (\mathbf{B}_\ell^{c_\ell}\mathbf{g}_{\ell-1})[i]|.$$

Clearly, such a number always exists, since $\mathbf{z}_{\ell-1}[i] \neq 0$ implies $|\mathbf{z}_{\ell-1}[i]| > 0$.

We next prove the corresponding claim, that is, there exists $\mathbf{g}_\ell$ such that for arbitrary $d \geq d_\ell$,

$$\lambda_\ell^d(v_\ell) = \mathbf{g}_\ell + d \cdot \mathbf{y}_\ell,$$

for a fixed $\mathbf{g}_\ell$. To define it first note that the vertices $u_1, ..., u_d$ are exactly ℓ steps away from v_ℓ, so such vertices do not affect the computation of $\lambda_{\ell-1}^d(v_\ell)$; using this observation and the induction hypothesis (since $d_\ell \geq d_{\ell-1}$) we have that for any $d \geq d_\ell$,

$$\lambda_\ell^d(v_\ell) = \mathrm{ReLU}(\mathbf{b}_\ell + \mathbf{A}_\ell \mathbf{t}_\ell + \mathbf{B}_\ell^{c_\ell}(\mathbf{g}_{\ell-1} + d \cdot \mathbf{y}_{\ell-1}),$$

which, using distribution of matrix product over vectors, can be rewritten as

$$\lambda_\ell^d(v_\ell) = \mathrm{ReLU}(\mathbf{b}_\ell + \mathbf{A}_\ell \mathbf{t}_\ell + \mathbf{B}_\ell^{c_\ell}\mathbf{g}_{\ell-1} + d \cdot \mathbf{z}_{\ell-1}).$$

We next consider an arbitrary $i \in \{1, \ldots, \delta_\ell\}$ and define $\mathbf{g}_\ell[i]$, again attending to the sign of $\mathbf{z}_{\ell-1}[i]$.

- If $\mathbf{z}_{\ell-1}[i]$ is negative, then $\mathbf{g}_\ell[i] = 0$. To show the claim let $d \geq d_\ell$. Then $\mathbf{b}_\ell[i] + (\mathbf{A}_\ell \mathbf{t}_\ell)[i] + (\mathbf{B}_\ell^{c_\ell}\mathbf{g}_{\ell-1})[i] + d_\ell \cdot \mathbf{z}_{\ell-1}[i] < 0$, by definition of d_ℓ; however, since $d \geq d_\ell$, we have $\mathbf{b}_\ell[i] + (\mathbf{A}_\ell \mathbf{t}_\ell)[i] + (\mathbf{B}_\ell^{c_\ell}\mathbf{g}_{\ell-1})[i] + d \cdot \mathbf{z}_{\ell-1}[i] < 0$, and so $\lambda_\ell^d(v_\ell)[i] = 0$. However, $\mathbf{y}_\ell[i] = \mathrm{ReLU}(\mathbf{z}_{\ell-1}[i]) = 0$. Thus, the claim holds.

- If $\mathbf{z}_{\ell-1}[i] = 0$, we let $\mathbf{g}[i] = \mathrm{ReLU}(\mathbf{b}_\ell[i] + (\mathbf{A}_\ell \mathbf{t}_\ell)[i] + (\mathbf{B}_\ell^{c_\ell}\mathbf{g}_{\ell-1})[i])$. To prove the claim, assume again $d \geq d_\ell$. Since $\mathbf{z}_{\ell-1}[i] = 0 = \mathbf{y}_\ell[i]$, we have $\lambda_\ell^d(v_\ell)[i] = \mathbf{g}_\ell[i]$, and the claim holds.

- If $\mathbf{z}_{\ell-1}[i]$ is positive, then $\mathbf{g}_\ell[i] = \mathbf{b}_\ell[i] + (\mathbf{A}_\ell \mathbf{t}_\ell)[i] + (\mathbf{B}_\ell^{c_\ell}\mathbf{g}_{\ell-1})[i]$. To prove the claim, consider $d \geq d_\ell$. The definition of d_ℓ ensures that $\mathbf{b}_\ell[i] + (\mathbf{A}_\ell \mathbf{t}_\ell)[i] + (\mathbf{B}_\ell^{c_\ell}\mathbf{g}_{\ell-1})[i] + d_\ell \cdot \mathbf{z}_{\ell-1}[i] > 0$, but since $d \geq d_\ell$, we have $\mathbf{b}_\ell[i] + (\mathbf{A}_\ell \mathbf{t}_\ell)[i] + (\mathbf{B}_\ell^{c_\ell}\mathbf{g}_{\ell-1})[i] + d \cdot \mathbf{z}_{\ell-1}[i] > 0$, and so $\lambda_\ell^d(v_\ell)[i] = \mathbf{b}_\ell[i] + (\mathbf{A}_\ell \mathbf{t}_\ell)[i] + (\mathbf{B}_\ell^{c_\ell}\mathbf{g}_{\ell-1})[i] + d \cdot \mathbf{z}_{\ell-1}[i] > 0$. Furthermore, $\mathbf{z}_{\ell-1}[i]$ being positive implies $\mathbf{y}_\ell[i] = \mathbf{z}_{\ell-1}[i]$, and so the claim holds.

This concludes the proof by induction.

Final d_L

From the proof by induction, there exists $\mathbf{g}_{L-1}$ such that for each $d \geq d_{L-1}$, it holds that

$$\lambda^d_{L-1}(v_{L-1}) = \mathbf{g}_{L-1} + d_{L-1} \cdot \mathbf{y}_{L-1}.$$

Recall that since channel p is unbounded, we have $\mathbf{y}_L[p] < 0$. Also, there exists $\tau \in \mathbb{R}$ such that $\sigma_L(\tau) < t$, where t refers to the threshold of the classifier $\texttt{cls}_t$ of $\mathcal{M}$. We now define d_L as the smallest natural number greater than or equal to d_{L-1} such that

$$d_L \cdot \mathbf{y}_L[p] < -|(\mathbf{b}_L + \mathbf{A}_L\mathbf{t}_L + \mathbf{B}^{c_L}_L\mathbf{g}_{L-1} + \mathbf{x}_L)[p]| + \tau,$$

where $\mathbf{x}_L$ refers to the value passed from the other neighbours of v_x (not including v_{L-1}). Note that this value does not depend on d.

Clearly, such a number always exists, since $\mathbf{y}_L[p] < 0$. Then $(d_L \cdot \mathbf{y}_L + \mathbf{b}_L + \mathbf{A}_L\mathbf{t}_L + \mathbf{B}^{c_L}_L\mathbf{g}_{L-1} + \mathbf{x}_L)[p] < \tau$. Then for any $d \geq d_L$, the above inequality still holds. Also since σ_L is monotonically increasing, we can derive $\sigma_L(d \cdot \mathbf{y}_L + \mathbf{b}_L + \mathbf{A}_L\mathbf{t}_L + \mathbf{B}^{c_L}_L\mathbf{g}_{L-1} + \mathbf{x}_L)[p] < \sigma_L(\tau)$.

But notice that $\lambda^d_L(v_x) = \sigma_L(d \cdot \mathbf{y}_L + \mathbf{b}_L + \mathbf{A}_L\mathbf{t}_L + \mathbf{B}^{c_L}_L\mathbf{g}_{L-1} + \mathbf{x}_L)$, so $\lambda^d_L(v_x)[p] < \sigma(\tau) < t$.

Conclusion

Fix $d = d_L$. We now have our counterexample dataset D_d. Since $\lambda^d_L(v_x)[p] < t$, $U_p(a) \notin T_{\mathcal{M}}(D_d)$. But since $D^r_\nu \subseteq D_d$, $U_p(a) \in T_r(D_d)$, so $T_r(D_d) \not\subseteq T_{\mathcal{M}}(D_d)$. Thus, r is not sound for $T_{\mathcal{M}}$. So, by contradiction, no rule mentioning U_p in the head is sound for $T_{\mathcal{M}}$.

□

C.1.5 Theorem 34

Theorem. *Unbounded channels are neither increasing, nor stable, nor safe. There exist, however, decreasing unbounded channels and undetermined unbounded channels.*

Proof. Consider a sum-GCN $\mathcal{M}$ as in Section 6.1. We prove the Theorem in two parts.

Unbounded channels are neither increasing, stable, nor safe

Consider a channel $i \in \{1, ..., \delta_L\}$ at layer L. Assume to the contrary that i is unbounded, and that it is also increasing or stable at layer L.

Let $\mathbf{y}_1, ..., \mathbf{y}_{L-1}$ be the sequence arising from Definition 32. Since i is unbounded, we have that $(\mathbf{B}_L^{c_L}\mathbf{y}_{L-1})[i] < 0$. But since $\mathbf{y}_{L-1}[j] \geq 0$ for each $j \in \{1, ..., \delta_{L-1}\}$ (due to the application of ReLU), we have $\mathbf{B}_L^{c_L}[k] < 0$ for some $k \in \{1, ..., \delta_L\}$.

From Definition 28, this implies that i is not stable at layer L. Furthermore, from condition (4) of Definition 28, i is not increasing at layer L. This is a contradiction and thus proves that an unbounded channel cannot be increasing nor stable.

Finally, from Theorem 31, safe channels are increasing or stable. Thus, an unbounded channel also cannot be safe, proving the claim.

There exist decreasing unbounded channels and undetermined unbounded channels

For simplicity, we provide examples of sum-GCNs with such unbounded channels using 2-layer GCNs with $\delta_0 = \delta_1 = \delta_2 = 2$ and $|\text{Col}| = 1$. However, the examples can be extended to sum-GCNs with an arbitrary number of layers, arbitrary δ, and arbitrary number of colours.

Let $c \in \text{Col}$ be the unique colour in Col. First, to show that there exist unbounded channels that are decreasing, consider a 2-layer sum-GCN with the following matrices. Let $\mathbf{A}_1 = \mathbf{A}_2$ be 2×2 matrices filled with zeroes. Define $\mathbf{B}_1^c$ and $\mathbf{B}_2^c$ as follows:

$$\mathbf{B}_1^c = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{B}_2^c = \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix}$$

Then for $\mathbf{y}_0 = (1,\ 1)^\top$, $\mathbf{y}_1 := \mathrm{ReLU}(\mathbf{B}_1^c \mathbf{y}_0) = (1,\ 1)^\top$ and $\mathbf{B}_2^c \mathbf{y}_1 = (-1,\ 0)^\top$, so channel 1 is unbounded. Also, all channels are increasing at layer 1, so channel 1 is decreasing at layer 2, providing the required example.

Next, to show that there exist unbounded channels that are undetermined, again consider a 2-layer sum-GCN with the following matrices. Let $\mathbf{A}_1 = \mathbf{A}_2$ be 2×2 matrices filled with zeroes. Define $\mathbf{B}_1^c$ and $\mathbf{B}_2^c$ as follows:

$$\mathbf{B}_1^c = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{B}_2^c = \begin{pmatrix} -2 & 1 \\ 0 & 0 \end{pmatrix}$$

Then for $\mathbf{y}_0 = (1,\ 1)^\top$, $\mathbf{y}_1 := \mathrm{ReLU}(\mathbf{B}_1^c \mathbf{y}_0) = (1,\ 1)^\top$ and $\mathbf{B}_2^c \mathbf{y}_1 = (-1,\ 0)^\top$, so channel 1 is unbounded. Also, all channels are increasing at layer 1, so channel 1 is undetermined at layer 2, providing the required example.

The above examples can be extended to an arbitrary number of layers L, dimensions $\delta_0, ..., \delta_L$, and colour set Col by using the following scheme:

1. Pick some colour $c_0 \in \mathrm{Col}$.
2. For each $\ell \in \{1, ..., L\}$ and $c \in \mathrm{Col}$ such that $c \neq c_0$, let $\mathbf{B}_\ell^c$ and $\mathbf{A}_\ell$ be matrices of zeroes.
3. For each $\ell \in \{1, ..., L-1\}$, let $\mathbf{B}_\ell^{c_0}$ be the identity matrix.
4. Let $\mathbf{B}_L^{c_0}$ consist entirely of zeroes, except for $\mathbf{B}_L^{c_0}[0,0]$ and $\mathbf{B}_L^{c_0}[0,1]$.
   (a) To obtain the case where channel 1 is both unbounded and decreasing, let $\mathbf{B}_L^{c_0}[0,0] = -1$ and $\mathbf{B}_L^{c_0}[0,1] = 0$, as in the example given above.
   (b) To obtain the case where channel 1 is both unbounded and undetermined, let $\mathbf{B}_L^{c_0}[0,0] = -2$ and $\mathbf{B}_L^{c_0}[0,1] = 1$, also as above.

□

Dataset	k_1	k_2	#M	#NM
FB-superhier	5	50000	237	0
WN-superhier	200	50000	237	0
WN-hier_nmhier	5, 20	2000, 2000	1	10
WN-cup_nmhier	5, 20	2000, 2000	1	10
FB-hier_nmhier	50, 200	200, 200	21	216
FB-cup_nmhier	50, 200	200, 200	21	216

Table C.1: Generation parameters for the new LogInfer datasets. Commas separate values where a dataset uses two rule patterns, and show the respective parameter values for the two patterns. #M and #NM are the number of distinct head predicates reserved for monotonic and non-monotonic rules, respectively.

Dataset	Model	Sample Rule
WN-hier	MGCN	_similar_to(x,y) and _instance_hypernym(y,y) implies _member_meronym(y,y)
WN-sym	MGCN	_has_part(y,x) and _derivationally_related_form(y,y) implies _derivationally_related_form(y,y)
WN-superhier	MGCN	_synset_domain_topic_of(y,y) and _member_of_domain_usage(y,z) implies _hypernym(y,y)
FB-hier	MGCN	/celebrity/dated./dated/participant(x,y) implies /celebrity/dated./dated/participant(y,x)
FB-sym	MGCN	/influence/influence_node/influenced_by(x,x) implies /influence/influence_node/influenced_by(x,x)
FB-superhier	MGCN	/winners./award_honor/award_winner(x,y) implies celebrity_friends./celebrities/friendship/friend(x,y)
FB237v1	MGCN	olympic_games/sports(x,y) implies /aareas/schema/administrative_area/administrative_parent(x,y)
NELLv1	R-0	headquarteredin(y,y) and acquired(y,y) implies acquired(y,y)
	R-25	acquired(y,y) and organizationheadquarteredincity(y,z) implies topmemberoforganization(y,y)
	R-50	worksfor(x,x) implies organizationhiredperson(x,x)
	MGCN	agentcontrols(y,y) and subpartoforganization(y,z) implies agentcollaborateswithagent(y,y)
WN18RRv1	R-0	_also_see(x,y) implies _also_see(y,x)
	R-25	_verb_group(x,y) and _has_part(z,y) implies _derivationally_related_form(x,y)
	R-50	_has_part(y,y) and _also_see(y,z) implies _also_see(z,y)
	MGCN	_derivationally_related_form(x,y) implies _derivationally_related_form(y,x)
WN-hier_nmhier	R-0	_synset_domain_topic_of(x,y) implies _instance_hypernym(x,y)
	R-25	_synset_domain_topic_of(x,y) and _also_see(y,y) implies _verb_group(x,y)
	R-50	_member_meronym(x,x) implies _hypernym(x,x)
	MGCN	_hypernym(x,y) and _has_part(z,y) implies _derivationally_related_form(z,y)
WN-cup_nmhier	R-0	_synset_domain_topic_of(y,y) and _derivationally_related_form(y,y) implies _has_part(y,y)
	R-25	_member_of_domain_region(x,x) implies _derivationally_related_form(x,x)
	R-50	_similar_to(y,x) and _also_see(y,y) implies _instance_hypernym(y,x)
	MGCN	_also_see(x,y) and _member_of_domain_region(y,z) implies _also_see(y,x)

Table C.2: Sample sound rules for the models that had some sound rules, where some predicate strings have been shortened to preserve space.

C.2 More Details About Experiments

Instructions for how to run my code and reproduce my results can be found in the "README.md" file in my code.

The parameters I use to generate all my novel LogInfer datasets are given in Table C.1.

In Table C.2, I provide a sample sound rule for each dataset/model combination that had some sound rules.

D

Appendices for Chapter 7

Contents

D.1 Proofs 237
D.1.1 Theorem 36 237
D.1.2 Proposition 37 240
D.1.3 Lemma 49 242
D.1.4 Theorem 39 243
D.2 Algorithms 248
D.2.1 Checking Soundness of ELUQ rules 248
D.2.2 Refining Explanations 248
D.3 Full Results 250

D.1 Proofs

D.1.1 Theorem 36

Theorem. *Let $\mathcal{M}$ be a MAGCN and r an $\mathcal{ELUQ}$ rule that is sound for $\mathcal{M}$. Then there exists a finite set of rules R such that:*

1. *Each $r' \in R$ has the form $\exists P_1.\top \sqcap ... \sqcap \exists P_j.\top \sqcap A_1 \sqcap ... \sqcap A_k \sqcap \top \sqsubseteq A_{k+1}$, where each P_i is a binary predicate, A_i a unary predicate, and $j, k \in \mathbb{N}_0$.*
2. *Each $r' \in R$ is sound for $\mathcal{M}$.*
3. *R subsumes r.*

Proof. First, notice that r is of the form $C^1 \sqcup ... \sqcup C^m \sqsubseteq A$ for $\mathcal{ELUQ}$ concepts $C^1, ..., C^m$ containing no disjunction in the outer scope (i.e. disjunction is only nested within existential quantification), $m \in \mathbb{N}$, and atomic concept A. Note that r not containing any disjunctions in the outer scope is covered by $m = 1$.

Now, let $R' := \{C^1 \sqsubseteq A, ..., C^m \sqsubseteq A\}$. By the operation of rule application, $T_{R'}(D) = T_r(D)$ for all datasets D. Also, each rule in R' is sound for $\mathcal{M}$, as a consequence of r being sound for $\mathcal{M}$. We will construct R by including in it a rule for each rule in R'.

Base Case

Consider each $r_1 \in R'$. If r_1 contains no $\exists$ or $\geq_n$ operators, then it is already trivially of the form specified in the theorem, so let $r_k = r_1$ and include it in R. Also, as stated above, r_1 is sound for $\mathcal{M}$. Otherwise, we have that r_1 is of the form $\geq_n P.C_1 \sqcap C_2 \sqsubseteq A$, where P is a binary predicate, C_1 an $\mathcal{ELUQ}$ concept, C_2 an $\mathcal{ELUQ}$ concept without disjunction, and n a positive integer.

We now inductively define a sequence $r_2, ..., r_k$ of $\mathcal{ELUQ}$ rules that such for all $i \in \{1, ..., k-1\}$,

1. r_{i+1} is sound for $\mathcal{M}$
2. r_{i+1} subsumes r_i

Inductive Step

For this purpose, consider the last defined rule r_i: we define r_{i+1} and prove the above properties.

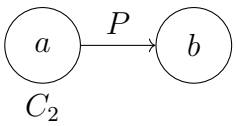
If r_i contains no $\exists$ or $\geq_n$ operators, then let $k := i$ and cease the inductive construction. As assumed in the inductive hypothesis, r_i is sound for $\mathcal{M}$. Otherwise, we have that r_i is of the form $\geq_n P.C_1 \sqcap C_2 \sqsubseteq A$, where P is a binary predicate, C_1 an $\mathcal{ELUQ}$ concept, C_2 an $\mathcal{ELUQ}$ concept without disjunction, and n a positive integer.

Now let r_{i+1} be the rule $\exists P.\top \sqcap C_2 \sqsubseteq A$. Trivially, r_{i+1} subsumes r_i, since they have the same head and whenever the body of r_{i+1} is satisfied, so too is the body of r_i. It remains to prove that r_{i+1} is sound for $\mathcal{M}$. For this purpose, assume to the contrary that r_{i+1} is not sound for $\mathcal{M}$.

If for all datasets D_1 such that $P(a, b) \in D_1$ and $(D_1, a) \models C_2$ for some constant b, we have $A(a) \in T_{\mathcal{M}}(D_1)$, then r_{i+1} would be sound for $\mathcal{M}$. This follows from these conditions covering all datasets that satisfy the body of r_{i+1}.

Likewise, if for all datasets D_1 such that $P(a, b) \in D_1$, $(D_1, a) \models C_2$, $b \neq a$, and b is not mentioned in any other fact of D_1, we have $A(a) \in T_{\mathcal{M}}(D_1)$, then r_{i+1} would be sound for $\mathcal{M}$. This is because neither mentioning b in another fact of D_1 nor having $a = b$ will decrease the output of $\mathcal{M}(D_1)$ for node v^a,

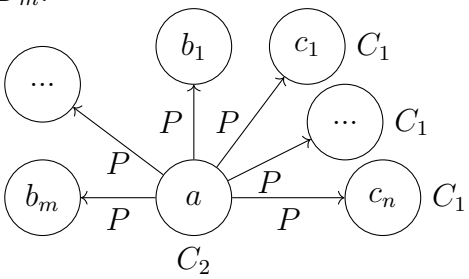
So instead there exists some dataset D_1 such that $P(a, b) \in D_1$, $(D_1, a) \models C_2$, and $A(a) \notin T_{\mathcal{M}}(D_1)$ for some constants $a \neq b$, where b is not mentioned in any fact of D_1 besides $P(a, b)$. Here follows a representation of D_1:



Now consider $\mathbf{v}_L^a$, the output for node v^a when $\mathcal{M}$ is applied to $\texttt{enc}(D_1)$, and let $1 \leq p \leq \delta$ be in the index of the unary predicate A in the canonical encoding. Since $A(a) \notin T_{\mathcal{M}}(D_1)$, we must have $\mathbf{v}_L^a[p] < t_{\mathcal{M}}$. Let $t := \frac{t_{\mathcal{M}} - \mathbf{v}_L^a[p]}{2}$.

We now define a dataset D_m, parameterized by a non-negative integer m. We define $D_m = D_1 \setminus \{P(a, b)\} \cup \{P(a, c_1), ..., P(a, c_n)\} \cup \{P(a, b_1), ..., P(a, b_m)\}$ for

distinct constants $c_1, ..., c_n, b_1, ..., b_m$ (all also distinct from a) and further require that $(D_m, c_1) \models C_1, ..., (D_m, c_n) \models C_1$, by including other facts necessary for the satisfaction to hold. We also require that $b_1, ..., b_m$ are not mentioned in any other facts of D_m. These further requirements may be satisfied in several different ways, but any of them will suffice for the definition of D_m. Here follows a representation of D_m:



Notice that $(D_m, a) \models C_2$ and $(D_m, a) \models \geq_n P.C_1$, so $A(a) \in T_{r_i}(D_m)$. But from the soundness of r_i, we also have $A(a) \in T_{\mathcal{M}}(D_m)$. However, as $m \to \infty$, the computation in node v^a when $\mathcal{M}$ is applied to $\texttt{enc}(D_m)$ tends to that of $\mathcal{M}$ being applied to $\texttt{enc}(D_1)$ — this is consequence of the definition of MAGCNs. Thus, there exists some m such that when considering $\mathbf{v}_L^a$, the output of $\mathcal{M}$ applied to $\texttt{enc}(D_m)$, we have $\mathbf{v}_L^a[p] < t$ (note that this requires the use of continuous activation functions, otherwise it is possible that $\mathbf{v}_L^a[p] \geq t$ for every m).

But since $t < t_{\mathcal{M}}$, we have $\mathbf{v}_L^a[p] < t_{\mathcal{M}}$. And since $A(a) \in T_{\mathcal{M}}(D_m)$, we have $\mathbf{v}_L^a[p] \geq t_{\mathcal{M}}$. This is a contradiction, so r_{i+1} is sound for $\mathcal{M}$.

End of Inductive Construction

Since r_1 contains a finite number of $\exists$ or $\geq_n$ operators and each r_{i+1} has one fewer $\exists$ or $\geq_n$ operator than r_i, this inductive construction is guaranteed to terminate for some k.

We thus have a sequence $r_1, r_2, ..., r_k$ of $\mathcal{ELUQ}$ rules such that for all $i \in \{1, ..., k-1\}$, r_{i+1} subsumes r_i. Since r_{i+1} subsumes r_i, for all datasets D we have $T_{r_i}(D) \subseteq T_{r_{i+1}}(D)$. So $T_{r_1}(D) \subseteq T_{r_2}(D) \subseteq ... \subseteq T_{r_k}(D)$, and thus $T_{r_1}(D) \subseteq T_{r_k}(D)$. Also, notice that r_k contains no existential quantifiers or disjunction, so it is of the form specified in the theorem. Furthermore, r_k is sound for $\mathcal{M}$, as was shown above. We include r_k in R.

Conclusion

We now have a set R of rules of the form specified in the theorem, such that each is sound for $\mathcal{M}$. It remains to be shown that R subsumes r.

Let D be a dataset. Recall that $T_r(D) = T_{R'}(D)$. But for each $r_1 \in R'$, there exists $r_k \in R$ such that $T_{r_1}(D) \subseteq T_{r_k}(D)$. Thus, $\bigcup_{r_1 \in R'} T_{r_1}(D) \subseteq \bigcup_{r_k \in R} T_{r_k}(D)$, and so $T_{R'}(D) \subseteq T_R(D)$. Hence, we have $T_r(D) \subseteq T_R(D)$, so R subsumes r, as required.

□

D.1.2 Proposition 37

Proposition. *Let $\mathcal{M}$ be a MAGCN and $r : \exists P_1.\top \sqcap ... \sqcap \exists P_j.\top \sqcap A_1 \sqcap ... \sqcap A_k \sqsubseteq A_{k+1}$ be a rule of the form given in Theorem 36. We define the dataset $D_{base} := \{A_1(a), ..., A_k(a), P_1(a,b), ..., P_j(a,b)\}$ for constants $a \neq b$. Then r is sound for $\mathcal{M}$ if and only if $A_{k+1}(a) \in T_{\mathcal{M}}(D_{base})$.*

Proof. We prove both direction of the if and only if. First, assume r is sound for $\mathcal{M}$. Notice that $(D_{\text{base}}, a) \models B$, where B denotes the body concepts of r, so we have $A_{k+1}(a) \in T_r(D_{\text{base}})$. But since r is sound for $\mathcal{M}$, $T_r(D_{\text{base}}) \subseteq T_{\mathcal{M}}(D_{\text{base}})$ and thus $A_{k+1}(a) \in T_{\mathcal{M}}(D_{\text{base}})$ as required.

Now assume that $A_{k+1}(a) \in T_{\mathcal{M}}(D_{\text{base}})$. We prove that r is sound for $\mathcal{M}$. For this purpose, let D be an arbitrary dataset: we prove that $T_r(D) \subseteq T_{\mathcal{M}}(D)$. So let $A_{k+1}(c) \in T_r(D)$.

Extending The Base Dataset

Consider the set D_{splits}, consisting of every dataset

$$\{A_1(a), ..., A_k(a), P_1(a, b_1), ..., P_j(a, b_j)\}$$

for (not necessarily mutually distinct) constants $b_1, ..., b_j$ that are distinct to a. Then for each such $D_1 \in D_{\text{splits}}$, the computation of $\mathcal{M}(D_1)$ is identical to that of $\mathcal{M}(D_{\text{base}})$. So $A_{k+1}(a) \in T_{\mathcal{M}}(D_1)$.

Now consider the set D_{merges}, consisting of every dataset

$$\{A_1(a), ..., A_k(a), P_1(a, b_1), ..., P_j(a, b_j)\}$$

for (not necessarily mutually distinct) constants $b_1, ..., b_j$ that are not necessarily distinct to a. Each $D_2 \in D_{\text{merges}}$ either has a distinct to all $b_1, ..., b_j$, or it does not. If it does, then $D_2 \in D_{\text{splits}}$, in which case $A_{k+1}(a) \in T_{\mathcal{M}}(D_2)$ Otherwise, D_2 has a corresponding $D_1 \in D_{\text{splits}}$ where each b_i such that $b_i = a$ is swapped out for another constant $b_i' \neq a$.

Consider $\mathbf{v}_L^a$ and $\mathbf{v}_L^{a\prime}$, the outputs for node v^a when $\mathcal{M}$ is applied to D_1 and D_2 respectively. Notice that, by the operations of MAGCNs, for all $p \in \{1, ..., \delta\}$ we have $\mathbf{v}_L^a[p] \leq \mathbf{v}_L^{a\prime}[p]$. This is because merging each of the neighbours b_i' of a into $b_i = a$ will never decrease the output of the MAGCN.

Now let p be the index of A_{k+1} in the canonical encoding. Then since $\mathbf{v}_L^a[p] \leq \mathbf{v}_L^{a\prime}[p]$ and $A_{k+1}(a) \in T_{\mathcal{M}}(D_1)$, we have $t_{\mathcal{M}} \leq \mathbf{v}_L^a[p] \leq \mathbf{v}_L^{a\prime}[p]$, and thus $A_{k+1}(a) \in T_{\mathcal{M}}(D_2)$. So for every $D_2 \in D_{\text{merges}}$, we have $A_{k+1}(a) \in T_{\mathcal{M}}(D_2)$.

Conclusion

We now prove that $A_{k+1}(c) \in T_{\mathcal{M}}(D)$. Since $A_{k+1}(c) \in T_r(D)$, there exists (not necessarily distinct) constants $d_1, ..., d_j$ in D such that

$$D' := \{A_1(c), ..., A_k(c), P_1(c, d_1), ..., P_j(c, d_j)\} \subseteq D.$$

There are potentially multiple such constants d_i that could be used for $D' \subseteq D$. For each $i \in \{1, ..., j\}$, if there exists constant $d_i \neq a$ such that $P_i(c, d_i) \in D$, then choose this d_i to be in D'.

But then, up to the relabelling of the constant $c \mapsto a$, we have $D' \in D_{\text{merges}}$, so $A_{k+1}(c) \in T_{\mathcal{M}}(D')$, since MAGCNs are agnostic to the specific constants used.

D is a superset of D'. Consider $v^c \in \texttt{enc}(D')$ and the application of $\mathcal{M}$ to D' and D, resulting in $\mathbf{v}_L^{c\prime}$ and $\mathbf{v}_L^c$, respectively. Any unary facts in D that are not in D' will preserve $\mathbf{v}_L^{c\prime} \leq \mathbf{v}_L^c$, from the definition of MAGCNs and the canonical encoding. The only binary facts in D that are not in D' that could decrease the

computation of $\mathbf{v}_L^c{}'$ (leading to $\mathbf{v}_L^c{}'[p] > \mathbf{v}_L^c[q]$ for some index q) are of the form $P_i(c, d_i)$ for $c \neq d_i$, where D' contains some fact $P_i(c, c)$. However, as was stated in the definition of D', if D contains a fact $P_i(c, d_i)$ with $c \neq d_i$, then D' will not contain $P_i(c, c)$. So, no facts added when extending D' to D will decrease the computation of $\mathbf{v}_L^c{}'$, and we obtain $\mathbf{v}_L^c{}' \leq \mathbf{v}_L^c$.

Now let p be the index of A_{k+1} in the canonical encoding. Then since $A_{k+1}(c) \in T_{\mathcal{M}}(D')$, we have $t_{\mathcal{M}} \leq \mathbf{v}_L^c{}'[p] \leq \mathbf{v}_L^c[p]$, and thus $A_{k+1}(c) \in T_{\mathcal{M}}(D)$, as required.

□

D.1.3 Lemma 49

I provide a lemma that is used in the proof of Theorem 39.

Lemma 49. *Let $\mathcal{M}$ be a MAGCN, D_a a dataset, and $a \in \mathtt{con}(D_a)$ a constant. For any $\ell \in \mathbb{N}_0$ and $c \in \mathtt{con}(D_a)$, we have $(D_a, c) \models C_\ell^c$, where C_ℓ^c is dependent on D_a.*

Proof. We prove this by induction on ℓ.

Base Case

For $\ell = 0$ and some $c \in \mathtt{con}(D_a)$, we have $C_0^c = \top \sqcap A_1 \sqcap ... \sqcap A_k$, where $A_1, ..., A_k$ are all atomic concepts A_i such that $A_i(c) \in D_a$. But then $(D_a, c) \models A_i$ for every A_i in the conjunction of C_0^c, so $(D_a, c) \models C_0^c$.

Inductive Step

Consider some $\ell \in \mathbb{N}$, and assume that the claim holds for $\ell - 1$. Let $c \in \mathtt{con}(D_a)$. We have $C_\ell^c = A_1 \sqcap ... \sqcap A_k \sqcap C'$, where $A_1, ..., A_k$ are all atomic concepts A_i such that $A_i(c) \in D_a$ and C' denotes the remainder of the concept. Similarly to the base case, we obtain $(D_a, c) \models A_1 \sqcap ... \sqcap A_k$.

C' is a conjunction of $\top$ with a concept C_P for each binary predicate P such that there exists $P(c, c_i) \in D_a$. So consider such a P: we prove that $(D_a, c) \models C_P$.

Let $c_1, ..., c_n$ be all the constants $c_i \in \mathtt{con}(D_a)$ such that $P(c, c_i) \in D$. Then we have $C_P := \exists_n P.(C_{\ell-1}^{c_1}, ..., C_{\ell-1}^{c_n}) \sqcap \leq_n P.\top$. Since c has exactly n P-neighbours in D_a, we have $(D_a, c) \models \leq_n P.\top$. By induction, $(D_a, c_i) \models C_{\ell-1}^{c_i}$ for every

$i \in \{1, ..., n\}$. Also, all $c_1, ..., c_n$ are distinct. So from the semantics of $\exists_n$, $(D_a, c) \models \exists_n P.(C^{c_1}_{\ell-1}, ..., C^{c_n}_{\ell-1})$. Thus, we obtain $(D_a, c) \models C_P$.

Now since $(D_a, c) \models C_P$ holds for every binary predicate P, we have $(D_a, c) \models C'$, and thus $(D_a, c) \models C^c_\ell$, as required. □

D.1.4 Theorem 39

Theorem. *For any MAGCN $\mathcal{M}$, dataset D_a, and fact $A(a) \in T_\mathcal{M}(D_a)$, the rule $r : C^a_L \sqsubseteq A$ (dependent on D_a) is sound for $\mathcal{M}$, and $A(a) \in T_r(D_a)$.*

Proof. First, we have $A(a) \in T_r(D_a)$, since A is the head of r and $(D_a, a) \models C^a_L$ follows directly from Lemma 49. It remains to be shown that r is sound for $\mathcal{M}$. Let D_b be a dataset: to show soundness, we prove that $T_r(D_b) \subseteq T_\mathcal{M}(D_b)$. For this purpose, let $A(b) \in T_r(D_b)$ be an arbitrary fact; we prove $A(b) \in T_\mathcal{M}(D_b)$.

Minimal Satisfaction Dataset

We define a dataset D^a_L in an analogous fashion to the definition of C^a_L, to be the L-hop neighbourhood of the constant a in D_a.

Given $c \in \texttt{con}(D_a)$, $\ell \in \mathbb{N}_0$, we inductively define a dataset D^c_ℓ by $D^c_\ell := \{A_i(c) \mid A_i(c) \in D_a\}$. Then, if $\ell = 0$, we are done: D^c_ℓ has been defined. Otherwise, for each binary predicate P, let $c_1, ..., c_n$ be all the constants $c_i \in \texttt{con}(D_a)$ such that $P(c, c_i) \in D_a$. Then, if $n > 0$, extend D^c_ℓ as follows: $D^c_\ell := D^c_\ell \cup D^{c_1}_{\ell-1} \cup D^{c_2}_{\ell-1} \cup ... \cup D^{c_n}_{\ell-1} \cup \{P(c, c_i) \mid P(c, c_i) \in D_a\}$. D^c_ℓ is extended for each binary predicate P. Note that each $D^{c_i}_{\ell-1}$ is defined inductively.

At this point in the proof, we have three datasets to consider:

- D_a — the original input dataset from which a fact $A(a)$ was produced by the MAGCN $T_\mathcal{M}$
- D_b — an arbitrary dataset instantiated for the proof of soundness
- D^a_L — the L-hop neighbourhood of the constant a in D_a

In the remainder of the proof, we show the following in sequence:

(1) $A(a) \in T_{\mathcal{M}}(D_L^a)$, since $A(a) \in T_{\mathcal{M}}(D_a)$ — follows from definition of D_L^a as the L-hop neighbourhood.

(2) $A(b) \in T_{\mathcal{M}}(D_b)$, since $A(a) \in T_{\mathcal{M}}(D_L^a)$ — follows from the fact that any differences between D_b and D_L^a cannot decrease the MAGCN output for v^a / v^b, plus a mapping between constants of D_b and D_L^a.

(1) Prove $A(a) \in T_{\mathcal{M}}(D_L^a)$ Using $A(a) \in T_{\mathcal{M}}(D_a)$

First note that $D_L^a \subseteq D_a$, since only facts from D_a are included in its definition. Consider $\mathbf{v}_L^a, \mathbf{v}_L^{a\prime}$, the output of $\mathcal{M}$ on D_a, D_L^a respectively for the vertex corresponding to a, and let p be the index of A in the canonical encoding. Since $A(a) \in T_{\mathcal{M}}(D_a)$, $\mathbf{v}_L^a[p] \geq t_{\mathcal{M}}$. However, since $\mathcal{M}$ has only L layers, the computation of $\mathbf{v}_L^a$ is affected only by vertices in the L-hop neighbourhood of v^a.

Since D_L^a contains exactly the L-hop neighbourhood of a, we thus have $\mathbf{v}_L^a = \mathbf{v}_L^{a\prime}$. This implies $\mathbf{v}_L^{a\prime}[p] \geq t_{\mathcal{M}}$ and thus $A(a) \in T_{\mathcal{M}}(D_L^a)$.

(2) Prove $A(b) \in T_{\mathcal{M}}(D_b)$ Using $A(a) \in T_{\mathcal{M}}(D_L^a)$

First, we inductively construct a mapping $\pi : \texttt{con}(D_L^a) \to \texttt{con}(D_b)$ from constants to constants. Intuitively, this will track "corresponding" constants from the two datasets in the computation of $\mathcal{M}$. We also assign a "level" to each constant of $\texttt{con}(D_L^a)$ from the set $\{0, ..., L\}$. We perform the induction on ℓ from L to 0, simultaneously proving that for each constant $c \in \texttt{con}(D_L^a)$ of level ℓ, we have:

$$(D_L^a, c) \models C_\ell^c \text{ and } (D_b, \pi(c)) \models C_\ell^c \tag{D.1}$$

Inductive Construction of Constant Mapping

For the base case, we define $\pi(a) = b$, and let the level of a be L. Since $A(b) \in T_r(D_b)$, $(D_b, b) \models C_L^a$, the body of r. So we have $(D_b, \pi(a)) \models C_L^a$ and also trivially $(D_L^a, a) \models C_L^a$.

Now consider each constant $c \in \texttt{con}(D_L^a)$ of level $\ell \geq 1$; we define π for constants of level $\ell - 1$. For each binary predicate P, let $c_1, ..., c_n$ be all the constants $c_i \in \texttt{con}(D_a)$ such that $P(c, c_i) \in D_a$. Let the level of each $c_1, ..., c_n$ be $\ell - 1$.

Let $d := \pi(c)$. By the inductive assumption, $(D_b, \pi(c)) \models C_\ell^c$, so we have that $(D_b, d) \models \exists_n P.(C_{\ell-1}^{c_1}, ..., C_{\ell-1}^{c_n}) \sqcap \leq_n P.\top$. So there exist distinct constants $d_1, ..., d_n \in \texttt{con}(D_b)$ such that $(D_b, d_1) \models C_{\ell-1}^{c_1}, ..., (D_b, d_n) \models C_{\ell-1}^{c_n}$ and $P(d, d_1) \in D_b, ..., P(d, d_n) \in D_b$. We define $\pi(c_1) = d_1, ..., \pi(c_n) = d_n$. Notice that we have that $(D_b, \pi(c_1)) \models C_{\ell-1}^{c_1}, ..., (D_b, \pi(c_n)) \models C_{\ell-1}^{c_n}$, and trivially from the construction of D_L^a that $(D_L^a, c_1) \models C_{\ell-1}^{c_1}, ..., (D_b, c_n) \models C_{\ell-1}^{c_n}$, as required for the induction proof.

Monotonicity

For each constant $c \in \texttt{con}(D_L^a)$ and $d \in \texttt{con}(D_b)$, let $\mathbf{v}_L^c$, $\bar{\mathbf{v}}_L^d$ be the outputs when $\mathcal{M}$ is applied to D_L^a and D_b, respectively. Likewise, let $E^o, \bar{E}^o$ denote the edges and $N_o, \bar{N}_o$ the neighbours in $\texttt{enc}(D_L^a)$ and $\texttt{enc}(D_b)$ for colour o, respectively. We prove the following claim by induction on ℓ from 0 to L.

Claim: For each $\ell \in \{0, ..., L\}$, constant $c \in \texttt{con}(D_L^a)$ of level $\ell' \in \{\ell, ..., L\}$, and $i \in \{1, ..., \delta_\ell\}$, we have $\mathbf{v}_\ell^c[i] \leq \bar{\mathbf{v}}_\ell^{\pi(c)}[i]$.

Base Case

For the base case ($\ell = 0$), let $i \in \{1, ..., \delta_0\}$, let $\ell' \in \{0, ..., L\}$, and consider each constant $c \in \texttt{con}(D_L^a)$ of level ℓ'.

As proven in Equation (D.1), we have $(D_L^a, c) \models C_{\ell'}^c$ and $(D_b, \pi(c)) \models C_{\ell'}^c$. Recall that $C_{\ell'}^c$ includes $\top \sqcap A_1 \sqcap ... \sqcap A_k$, where $A_1, ..., A_k$ are all atomic concepts A' such that $A'(c) \in D_a$. From the definition of D_L^a, the unary facts in D_L^a that mention c are precisely $A_1(c), ..., A_k(c)$. Then since $(D_b, \pi(c)) \models C_{\ell'}^c$, we have $A_1(\pi(c)), ..., A_k(\pi(c)) \in D_b$.

So if $\mathbf{v}_0^c[i] = 1$, then $A_i(c) \in D_L^a$, where A_i is the unary predicate with index i in the canonical encoding. This implies that $A_i(\pi(c)) \in D_b$, and thus $\bar{\mathbf{v}}_0^{\pi(c)}[i] = 1$. This proves the base case, since $\mathbf{v}_0^c[i] = 1 \implies \bar{\mathbf{v}}_0^{\pi(c)}[i] = 1$ and the definition of the canonical encoding let us conclude that $\mathbf{v}_0^c[i] \leq \bar{\mathbf{v}}_0^{\pi(c)}[i]$.

Inductive Step

For the inductive step, we prove the claim for $\ell \geq 1$; to do so, assume the claim holds for $\ell - 1$. Let $\ell' \in \{\ell, ..., L\}$, let $i \in \{1, ..., \delta_\ell\}$, and consider each constant $c \in \texttt{con}(D_L^a)$ of level ℓ'. We prove that $\mathbf{v}_\ell^c[i] \leq \bar{\mathbf{v}}_\ell^{\pi(c)}[i]$. Consider the computation of $\mathbf{v}_\ell^c[i]$ and $\bar{\mathbf{v}}_\ell^{\pi(c)}[i]$:

$$\mathbf{v}_\ell^c[i] = \sigma_\ell \left(\mathbf{b}_\ell[i] + \sum_{j=1}^{\delta_{\ell-1}} \mathbf{A}_\ell[i,j] \mathbf{v}_{\ell-1}^c[j] + \sum_{o \in \text{Col}} \sum_{j=1}^{\delta_{\ell-1}} \mathbf{B}_\ell[i,j] \ \text{mean}(\{\!\{ \mathbf{u}_{\ell-1}[j] \mid (v^c, u) \in E^o \}\!\}) \right) \tag{D.2}$$

$$\bar{\mathbf{v}}_\ell^{\pi(c)}[i] = \sigma_\ell \left(\mathbf{b}_\ell[i] + \sum_{j=1}^{\delta_{\ell-1}} \mathbf{A}_\ell[i,j] \bar{\mathbf{v}}_{\ell-1}^{\pi(c)}[j] + \sum_{o \in \text{Col}} \sum_{j=1}^{\delta_{\ell-1}} \mathbf{B}_\ell[i,j] \ \text{mean}(\{\!\{ \bar{\mathbf{u}}_{\ell-1}[j] \mid (v^{\pi(c)}, u) \in \bar{E}^o \}\!\}) \right) \tag{D.3}$$

From the induction hypothesis, since c is of level $\ell' \geq \ell - 1$, we have $\mathbf{v}_{\ell-1}^c[j] \leq \bar{\mathbf{v}}_{\ell-1}^{\pi(c)}[j]$. Since all else in the above equations is equal, it remains to be shown that, for all colours $o \in \text{Col}$ and $j \in \{1, ..., \delta_{\ell-1}\}$:

$$\text{mean}(\{\!\{ \mathbf{u}_{\ell-1}[j] \mid (v^c, u) \in E^o \}\!\}) \leq \ \text{mean}(\{\!\{ \bar{\mathbf{u}}_{\ell-1}[j] \mid (v^{\pi(c)}, u) \in \bar{E}^o \}\!\})$$

So let $o \in \text{Col}$, $j \in \{1, ..., \delta_{\ell-1}\}$, and let P be the binary predict corresponding to colour o. Now if there exists no $c_k \in \texttt{con}(D_a)$ such that $P(c, c_k) \in D_a$, then by the definition of E^o as the edges from $\texttt{enc}(D_L^a)$, we have $E^o = \emptyset$ and thus $\text{mean}(\{\!\{ \mathbf{u}_{\ell-1}[j] \mid (v^c, u) \in E^o \}\!\}) = 0$. So the inequality trivially holds, since each $\bar{\mathbf{u}}_{\ell-1}[j]$ is non-negative.

Thus, instead consider all $c_1, ..., c_n \in \texttt{con}(D_a)$ such that $P(c, c_k) \in D_a$. Note that each such c_k has level $\ell' - 1$. Then since $\ell' - 1 \geq \ell - 1$, from the inductive hypothesis, we have $\mathbf{v}_{\ell-1}^{c_k}[j] \leq \bar{\mathbf{v}}_{\ell-1}^{\pi(c_k)}[j]$ for each c_k.

Furthermore, since $c \in \texttt{con}(D_L^a)$ has level ℓ', we obtain $(D_b, \pi(c)) \models C_{\ell'}^c$ from Equation (D.1). Thus, we have $(D_b, \pi(c)) \models \exists_n P.(C_{\ell'-1}^{c_1}, ..., C_{\ell'-1}^{c_n}) \sqcap \ \leq_n P.\top$. So there exist exactly n distinct constants $d_1, ..., d_n \in \texttt{con}(D_b)$ such that $P(\pi(c), d_1) \in$

$D_b, ..., P(\pi(c), d_n) \in D_b$. However, recall from the definition of π, that these constants are precisely $d_1 = \pi(c_1), ..., d_n = \pi(c_n)$.

From the canonical encoding, we thus have a one-to-one correspondence between the sets $\{v^{c_k} \mid (v^c, v^{c_k}) \in E^o\}$ and $\{v^{\pi(c_k)} \mid (v^{\pi(c)}, v^{\pi(c_k)}) \in \bar{E}^o\} = \{v^{d_k} \mid (v^{\pi(c)}, v^{d_k}) \in \bar{E}^o\}$. Also, as stated above, from the inductive hypothesis, we have $\mathbf{v}_{\ell-1}^{c_k}[j] \leq \bar{\mathbf{v}}_{\ell-1}^{\pi(c_k)}[j]$ for each c_k. Combining these, we obtain the inequality $\text{mean}(\{\!\{\mathbf{u}_{\ell-1}[j] \mid (v^c, u) \in E^o\}\!\}) \leq \text{mean}(\{\!\{\bar{\mathbf{u}}_{\ell-1}[j] \mid (v^{\pi(c)}, u) \in \bar{E}^o\}\!\})$. This allows us to conclude $\mathbf{v}_\ell^c[i] \leq \bar{\mathbf{v}}_\ell^{\pi(c)}[i]$, as required.

Conclusion

We have shown that for each $\ell \in \{0, ..., L\}$, constant $c \in \texttt{con}(D_L^a)$ of level $\ell' \in \{\ell, ..., L\}$, and $i \in \{1, ..., \delta_\ell\}$, we have $\mathbf{v}_\ell^c[i] \leq \bar{\mathbf{v}}_\ell^{\pi(c)}[i]$.

Now let p be the index of A in the canonical encoding. We obtain $\mathbf{v}_L^a[p] \leq \bar{\mathbf{v}}_L^{\pi(a)}[p]$, and thus $\mathbf{v}_L^a[p] \leq \bar{\mathbf{v}}_L^b[p]$, since $b = \pi(a)$ and a is of level L. But since $A(a) \in T_{\mathcal{M}}(D_L^a)$, $\mathbf{v}_L^a[p] \geq t_{\mathcal{M}}$, thus $\bar{\mathbf{v}}_L^b[p] \geq t_{\mathcal{M}}$, and finally $A(b) \in T_{\mathcal{M}}(D_b)$. □

D.2 Algorithms

D.2.1 Checking Soundness of ELUQ rules

Algorithm 2: Check the soundness of all rules of the form given in Theorem 36 for MAGCN $\mathcal{M}$

Input : MAGCN $\mathcal{M}$

Output : S: set of sound rules for $\mathcal{M}$ of the form given in Theorem 36 that together subsume every $\mathcal{ELUQ}$ rule that is sound for $\mathcal{M}$

$S \leftarrow \emptyset$ `// result set`

foreach $n \in \{1, ...\delta + |\mathsf{Col}|\}$ **do**

  `// consider rules whose body contains exactly` n `concept atoms`

  foreach *candidate rule* r *of form*

$$r : \exists P_1.\top \sqcap \ldots \sqcap \exists P_j.\top \sqcap A_1 \sqcap \ldots \sqcap A_k \sqsubseteq A_{k+1}$$

  with exactly n *body concepts* **do**

    `// check subsumption by a sound (smaller) rule`

    if $\exists r' \in S$ *with head* A_{k+1} *whose body concepts are contained in those of* r **then**

      $S \leftarrow S \cup \{r\}$

      continue

    `// construct dataset with fresh constants` $a \neq b$

    $D_{\text{base}} \leftarrow \{A_1(a), \ldots, A_k(a),\ P_1(a,b), \ldots, P_j(a,b)\}$

    `// test the soundness of` r`, using Proposition 37`

    if $A_{k+1}(a) \in T_{\mathcal{M}}(D_{base})$ **then**

      $S \leftarrow S \cup \{r\}$

return S

D.2.2 Refining Explanations

There are two primary issues with the rules defined in Theorem 39.

First, the size of the rules can be large in practice, making them difficult to read, and also limiting their ability to generalize to other datasets. The size of the rule can be iteratively refined by pruning facts from D_a and then recomputing C_L^a.

More precisely, for $D'_a := D_a \setminus \{\alpha\}$ for some unary fact $\alpha \in D_a$ such that $A(a) \in T_{\mathcal{M}}(D'_a)$, the rule $r' : C_L^a \sqsubseteq A$ (dependent on D'_a) is still sound for $\mathcal{M}$, by the same argument made in Theorem 39. This can be done iteratively, until we find that $A(a) \notin T_{\mathcal{M}}(D'_a)$. This works similarly for $D'_a := D_a \setminus D_\alpha$, where $D_\alpha = \{P(c, c_1), ..., P(c, c_n)\}$ for some binary predicate P and constants $c, c_1, ..., c_n$,

such that there is no fact $P(c,d) \in D_a$ with $P(c,d) \notin D_\alpha$, and $A(a) \in T_\mathcal{M}(D'_a)$. Note that there are many different choices for α and D_α in each iteration, so the space should be searched to find explanatory rules with minimal size. Only facts α and sets D_α within the L-hop neighbourhood of a should be considered for pruning, as this is what will decrease the number of concepts in the body C^a_L.

The second issue is that the inclusion of the $\leq_n P.\top$ term limits their ability to produce facts on other datasets with higher numbers of neighbours. As a possible solution, one can substitute n for some $m > n$. Consider a dataset D_a and fact $A(a) \in T_\mathcal{M}(D_a)$. Instead of defining $C^c_\ell := \ldots \sqcap \leq_n P.\top$ in Definition 38, we can define $C^c_\ell := \ldots \sqcap \leq_m P.\top$, for some $m \geq n$. Any m can be chosen such that extending D_a with $D^c_\ell := \{P(c,d_1), ..., P(c,d_{m-n})\}$ for fresh constants $d_1, ..., d_{m-n}$ still leads to $A(a) \in T_\mathcal{M}(D_a \cup D^c_\ell)$. If the above is done for multiple constants $c_1, ..., c_k$ and $\ell_1, ..., \ell_k$, it must be ensured that $A(a) \in T_\mathcal{M}(D_a \cup D^{c_1}_{\ell_1} \cup ... \cup D^{c_k}_{\ell_k})$.

The simple approach I take in this work is as follows. Given a predicted fact $A(a)$ on a dataset D, I first identify all atoms $A_1, ..., A_n$ such that $A_i(a) \in D$, and check if the rule $A_1 \sqcap ... \sqcap A_n \sqsubseteq A$ is sound using Proposition 37. If it is not, I then identify all binary predicates $P_1, ..., P_m$ such that $P_i(a,b) \in D$ for any constant b, and check if the rule $A_1 \sqcap ... \sqcap A_n \sqcap \exists P_1.\top \sqcap ... \sqcap \exists P_m.\top \sqsubseteq A$ is sound using Proposition 37. If either of these rules is sound, it explains the prediction $A(a)$ on D.

Dataset	Agg	Weights	%Acc	%Prec	%Rec	F1
LUBM	Mean	Standard	97.1 ± 0.0	96.9 ± 0.0	97.2 ± 0.0	97.1 ± 0.0
		Non-Neg	91.5 ± 0.0	87.8 ± 0.0	96.4 ± 0.0	91.9 ± 0.0
	Max	Standard	95.7 ± 0.0	95.3 ± 0.0	96.3 ± 0.0	95.8 ± 0.0
		Non-Neg	91.6 ± 0.0	88 ± 0.0	96.3 ± 0.0	92 ± 0.0
	Sum	Standard	96.1 ± 0.0	95.9 ± 0.0	96.2 ± 0.0	96.1 ± 0.0
		Non-Neg	91 ± 0.0	88.1 ± 0.0	94.9 ± 0.0	91.4 ± 0.0

Table D.1: Results for GCNs across various aggregation functions and standard / non-negative weights, for the LUBM dataset. Metrics are computed on the test set and shown with a 95% CI.

D.3 Full Results

Full results for mean, sum, and max aggregation are given in Tables D.1 to D.3. Randomly sampled sound monotonic rules of the form given in Theorem 36 are shown in Table D.4. Randomly sampled explanatory rules for predictions on LUBM of the form given in Theorem 39 are shown in Table D.5.

Randomly sampled explanatory rules that have been reduced in size are shown in Table D.6. The final entry in the table is an example of a predicted fact where the size reduction process failed and fell back to using Theorem 39.

On average, out of 1990 true positive predictions, the model only failed 22 times to explain the predicted fact using a rule from Theorem 36. For 4 of my 5 trained MAGCNs, it only failed 10 times each. For the other MAGCN, it failed 72 times. One factor that brought up the average number of body concepts is that some explanations were very large: for example, the rule that explained the prediction Department(http://www.Department2.University0.edu) had 1914 body concepts.

Dataset	Agg	Weights	%Acc	%Prec	%Rec	F1
WN-hier	Mean	Standard	99.5 ± 0.0	99.5 ± 0.0	99.5 ± 0.0	99.5 ± 0.0
		Non-Neg	99.8 ± 0.0	99.7 ± 0.0	99.9 ± 0.0	99.8 ± 0.0
	Max	Standard	99.3 ± 0.0	99.5 ± 0.0	99.1 ± 0.0	99.3 ± 0.0
		Non-Neg	99.9 ± 0.0	99.8 ± 0.0	100 ± 0.0	99.9 ± 0.0
	Sum	Standard	98.9 ± 0.0	99.5 ± 0.0	98.3 ± 0.0	98.9 ± 0.0
		Non-Neg	98.8 ± 0.0	97.6 ± 0.0	100 ± 0.0	98.8 ± 0.0
WN-sym	Mean	Standard	99.4 ± 0.0	99.5 ± 0.0	99.3 ± 0.0	99.4 ± 0.0
		Non-Neg	100 ± 0.0	100 ± 0.0	100 ± 0.0	100 ± 0.0
	Max	Standard	99.3 ± 0.0	99.1 ± 0.0	99.5 ± 0.0	99.3 ± 0.0
		Non-Neg	100 ± 0.0	100 ± 0.0	100 ± 0.0	100 ± 0.0
	Sum	Standard	99.1 ± 0.0	99.3 ± 0.0	98.8 ± 0.0	99.1 ± 0.0
		Non-Neg	100 ± 0.0	100 ± 0.0	100 ± 0.0	100 ± 0.0
WN-hier_nmhier	Mean	Standard	86.2 ± 0.0	84.7 ± 0.0	88.4 ± 0.0	86.5 ± 0.0
		Non-Neg	71.6 ± 0.0	80.1 ± 0.1	58.8 ± 0.1	67.2 ± 0.0
	Max	Standard	86.9 ± 0.0	85.1 ± 0.0	89.5 ± 0.0	87.2 ± 0.0
		Non-Neg	69.5 ± 0.0	70.1 ± 0.0	68.4 ± 0.0	69.2 ± 0.0
	Sum	Standard	89.5 ± 0.0	88.1 ± 0.0	91.4 ± 0.0	89.7 ± 0.0
		Non-Neg	66.9 ± 0.0	73.5 ± 0.1	59.3 ± 0.2	62.7 ± 0.1

Table D.2: Results for GCNs across various aggregation functions and standard / non-negative weights, for the LogInfer datasets. Metrics are computed on the test set and shown with a 95% CI.

Dataset	Agg	Weights	%Acc	%Prec	%Rec	F1
FB237v1	Mean	Standard	68.7 ± 0.0	95.4 ± 0.0	39.3 ± 0.0	55.7 ± 0.0
		Non-Neg	71.8 ± 0.0	75.4 ± 0.0	64.8 ± 0.0	69.7 ± 0.0
	Max	Standard	67 ± 0.0	94.9 ± 0.0	35.9 ± 0.0	52 ± 0.0
		Non-Neg	75.8 ± 0.0	85.9 ± 0.1	63.4 ± 0.0	72.6 ± 0.0
	Sum	Standard	68.4 ± 0.0	99.7 ± 0.0	36.8 ± 0.0	53.8 ± 0.0
		Non-Neg	73.8 ± 0.0	79.8 ± 0.1	64.5 ± 0.0	71.2 ± 0.0
WN18RRv1	Mean	Standard	93.7 ± 0.0	98.5 ± 0.0	88.8 ± 0.0	93.4 ± 0.0
		Non-Neg	95.5 ± 0.0	98.1 ± 0.0	92.7 ± 0.0	95.3 ± 0.0
	Max	Standard	93.6 ± 0.0	97 ± 0.0	90.1 ± 0.0	93.4 ± 0.0
		Non-Neg	95.5 ± 0.0	98.7 ± 0.0	92.1 ± 0.0	95.3 ± 0.0
	Sum	Standard	93.9 ± 0.0	96.1 ± 0.0	91.6 ± 0.0	93.8 ± 0.0
		Non-Neg	94.9 ± 0.0	97 ± 0.0	92.7 ± 0.0	94.8 ± 0.0
NELLv1	Mean	Standard	75.2 ± 0.1	93.8 ± 0.0	53.4 ± 0.2	65.7 ± 0.2
		Non-Neg	93.4 ± 0.0	88.8 ± 0.0	99.4 ± 0.0	93.8 ± 0.0
	Max	Standard	51.2 ± 0.0	18.8 ± 0.4	3.5 ± 0.1	5.9 ± 0.1
		Non-Neg	93.2 ± 0.0	92.4 ± 0.0	94.1 ± 0.0	93.2 ± 0.0
	Sum	Standard	51.8 ± 0.0	18.3 ± 0.4	4.9 ± 0.1	7.8 ± 0.2
		Non-Neg	92 ± 0.0	86.3 ± 0.0	100 ± 0.0	92.6 ± 0.0

Table D.3: Results for GCNs across various aggregation functions and standard / non-negative weights, for the standard benchmark datasets. Metrics are computed on the test set and shown with a 95% CI.

Dataset	Sample Sound Rules
LUBM	$\top \sqsubseteq$ Publication $\top \sqsubseteq$ UndergraduateStudent $\exists$authorOfPublication.$\top \sqsubseteq$ ResearchAssistant $\exists$advisorOf.$\top \sqsubseteq$ AssociateProfessor $\exists$courseTakenBy.$\top \sqcap \exists$hasTeacher.$\top \sqsubseteq$ Course Department $\sqcap$ UndergraduateStudent $\sqcap \exists$advisorOf.$\top \sqsubseteq$ FullProfessor
WN-hier	_member_of_domain_usage(X, Y) $\rightarrow$ _member_meronym(X, Y) _derivationally_related_form(X, Y) $\wedge$ _similar_to$(X, Y) \rightarrow$ _synset_domain_topic_of(X, Y) _also_see(X, Y) $\wedge$ _similar_to(X, Y) $\wedge$ _verb_group$(X, Y) \rightarrow$ _instance_hypernym(X, Y)
WN-sym	_has_part(X, Y) $\wedge$ _similar_to(X, Y) $\rightarrow$ _derivationally_related_form(X, Y)
WN-hier_nmhier	_has_part(X, Y) $\wedge$ _member_meronym(X, Y) $\rightarrow$ _member_of_domain_usage(X, Y) _has_part(X, Y) $\wedge$ _verb_group(X, Y) $\rightarrow$ _member_of_domain_usage(X, Y) _hypernym$(X, Y) \wedge$ _synset_domain_topic_of$(X, Y) \rightarrow$ _also_see(X, Y)
FB237v1	$\top \rightarrow$ /music/instrument/instrumentalists(X, Y) $\top \rightarrow$ /sibling_s./sibling_relationship/sibling(X, Y) /base/biblioness/bibs_location/state(X, Y) $\rightarrow$ /film/film/genre(X, Y) /olympics/olympic_games/sports(X, Y) $\rightarrow$ popstra/celebrity/dated.popstra/dated/participant(X, Y)
NELLv1	organizationterminatedperson(X, Y) $\wedge$ topmemberoforganization(X, Y) $\rightarrow$ organizationhiredperson(X, Y) worksfor$(X, Y) \wedge$ organizationterminatedperson$(X, Y) \rightarrow$ organizationhiredperson(X, Y)

Table D.4: Randomly sampled sound monotonic rules of the form given in Theorem 36, where some predicate strings have been shortened to preserve space.

Fact	Publication(http://www.Department2.University0.edu/AssociateProfessor7/Publication0)
Rule	$\top \sqsubseteq$ Publication
Fact	University(http://www.University370.edu)
Rule	$\top \sqcap \exists_1$grantedUndergraduateDegreeTo.(GraduateStudent $\sqcap \exists_1$authorOfPublication.(Publication) $\sqcap \leq_1$ authorOfPublication.$\top$) $\sqcap \leq_1$ grantedUndergraduateDegreeTo.$\top \sqsubseteq$ University
Fact	GraduateStudent(http://www.Department6.University0.edu/GraduateStudent29)
Rule	$\top \sqcap \exists_4$authorOfPublication.($\top, \top$, Publication, Publication) $\sqcap \leq_4$ authorOfPublication.$\top \sqsubseteq$ GraduateStudent
Fact	TeachingAssistant(http://www.Department10.University0.edu/GraduateStudent46)
Rule	GraduateStudent $\sqcap \exists_3$authorOfPublication.($\top, \top$, Publication) $\sqcap \leq_3$ authorOfPublication.$\top \sqsubseteq$ TeachingAssistant
Fact	GraduateStudent(http://www.Department14.University0.edu/GraduateStudent2)
Rule	ResearchAssistant $\sqcap \exists_5$authorOfPublication.($\top$, Publication, Publication, Publication, Publication) $\sqcap \leq_5$ authorOfPublication.$\top \sqsubseteq$ GraduateStudent
Fact	UndergraduateStudent(http://www.Department5.University0.edu/UndergraduateStudent87)
Rule	$\top \sqsubseteq$ UndergraduateStudent
Fact	TeachingAssistant(http://www.Department6.University0.edu/GraduateStudent48)
Rule	$\top \sqcap \exists_3$authorOfPublication.($\top$, Publication, $\top$) $\sqcap \leq_3$ authorOfPublication.$\top \sqsubseteq$ TeachingAssistant
Fact	GraduateStudent(http://www.Department6.University0.edu/GraduateStudent1)
Rule	$\top \sqcap \exists_1$authorOfPublication.($\top$) $\sqcap \leq_1$ authorOfPublication.$\top \sqsubseteq$ GraduateStudent
Fact	Course(http://www.Department14.University0.edu/Course39)
Rule	$\top \sqcap \exists_{14}$courseTakenBy.($\top$, UndergraduateStudent, UndergraduateStudent, UndergraduateStudent, UndergraduateStudent, UndergraduateStudent, UndergraduateStudent, UndergraduateStudent, UndergraduateStudent, $\top, \top$, UndergraduateStudent, UndergraduateStudent, UndergraduateStudent) $\sqcap \leq_{14}$ courseTakenBy.$\top \sqcap \exists_1$hasTeacher.(AssistantProfessor $\sqcap \exists_7$advisorOf.(UndergraduateStudent, UndergraduateStudent, $\top, \top$, TeachingAssistant $\sqcap$ GraduateStudent, GraduateStudent, $\top$) $\sqcap \leq_7$ advisorOf.$\top \sqcap \exists_5$authorOfPublication.(Publication, Publication, Publication, Publication, Publication) $\sqcap \leq_5$ authorOfPublication.$\top$) $\sqcap \leq_1$ hasTeacher.$\top \sqsubseteq$ Course
Fact	ResearchAssistant(http://www.Department11.University0.edu/GraduateStudent71)
Rule	$\top \sqcap \exists_4$authorOfPublication.(Publication, Publication, Publication, Publication) $\sqcap \leq_4$ authorOfPublication.$\top \sqsubseteq$ ResearchAssistant
Fact	Course(http://www.Department14.University0.edu/Course46)
Rule	$\top \sqcap \exists_9$courseTakenBy.($\top, \top$, UndergraduateStudent, $\top$, UndergraduateStudent, $\top$, UndergraduateStudent, $\top$, UndergraduateStudent) $\sqcap \leq_9$ courseTakenBy.$\top \sqcap \exists_1$hasTeacher.(Lecturer) $\sqcap \leq_1$ hasTeacher.$\top \sqcap \exists_1$hasTeachingAssistant.(TeachingAssistant $\sqcap$ GraduateStudent $\sqcap \exists_5$authorOfPublication.(Publication, $\top$, Publication, Publication, $\top$) $\sqcap \leq_5$ authorOfPublication.$\top$) $\sqcap \leq_1$ hasTeachingAssistant.$\top \sqsubseteq$ Course
Fact	University(http://www.University138.edu)
Rule	$\top \sqcap \exists_2$grantedUndergraduateDegreeTo.(GraduateStudent $\sqcap \exists_1$authorOfPublication.(Publication) $\sqcap \leq_1$ authorOfPublication.$\top$, GraduateStudent $\sqcap \exists_5$authorOfPublication.($\top$, Publication, $\top$, Publication, Publication) $\sqcap \leq_5$ authorOfPublication.$\top$) $\sqcap \leq_2$ grantedUndergraduateDegreeTo.$\top \sqsubseteq$ University

Table D.5: Randomly sampled explanatory rules of the form given in Theorem 39, shown with the corresponding facts produced by the MAGCN.

Fact	Publication(http://www.Department10.University0.edu/FullProfessor5/Publication3)
Rule	$\top \sqsubseteq$ Publication
Fact	UndergraduateStudent(http://www.Department8.University0.edu/UndergraduateStudent170)
Rule	$\top \sqsubseteq$ UndergraduateStudent
Fact	TeachingAssistant(http://www.Department5.University0.edu/GraduateStudent45)
Rule	GraduateStudent $\sqcap$ $\exists$authorOfPublication.$\top \sqsubseteq$ TeachingAssistant
Fact	ResearchAssistant(http://www.Department14.University0.edu/GraduateStudent91)
Rule	GraduateStudent $\sqcap$ $\exists$authorOfPublication.$\top \sqsubseteq$ ResearchAssistant
Fact	ResearchAssistant(http://www.Department1.University0.edu/GraduateStudent14)
Rule	GraduateStudent $\sqsubseteq$ ResearchAssistant
Fact	ResearchAssistant(http://www.Department4.University0.edu/GraduateStudent26)
Rule	$\exists$authorOfPublication.$\top \sqsubseteq$ ResearchAssistant
Fact	GraduateStudent(http://www.Department6.University0.edu/GraduateStudent86)
Rule	$\exists$authorOfPublication.$\top \sqsubseteq$ GraduateStudent
Fact	GraduateStudent(http://www.Department12.University0.edu/GraduateStudent48)
Rule	ResearchAssistant $\sqsubseteq$ GraduateStudent
Fact	GraduateStudent(http://www.Department3.University0.edu/GraduateStudent44)
Rule	ResearchAssistant $\sqcap$ $\exists$authorOfPublication.$\top \sqsubseteq$ GraduateStudent
Fact	GraduateStudent(http://www.Department0.University0.edu/GraduateStudent67)
Rule	TeachingAssistant $\sqcap$ $\exists$authorOfPublication.$\top \sqsubseteq$ GraduateStudent
Fact	GraduateCourse(http://www.Department4.University0.edu/GraduateCourse31)
Rule	$\exists$courseTakenBy.$\top$ $\sqcap$ $\exists$hasTeacher.$\top \sqsubseteq$ GraduateCourse
Fact	Course(http://www.Department12.University0.edu/Course1)
Rule	$\exists$courseTakenBy.$\top$ $\sqcap$ $\exists$hasTeacher.$\top \sqsubseteq$ Course
Fact	AssociateProfessor(http://www.Department7.University0.edu/AssociateProfessor4)
Rule	$\exists$advisorOf.$\top \sqsubseteq$ AssociateProfessor
Fact	AssistantProfessor(http://www.Department13.University0.edu/AssistantProfessor10)
Rule	$\exists$advisorOf.$\top \sqsubseteq$ AssistantProfessor
Fact	FullProfessor(http://www.Department6.University0.edu/FullProfessor3)
Rule	$\exists$advisorOf.$\top \sqsubseteq$ FullProfessor
Fact	University(http://www.University772.edu)
Rule	$\exists$grantedUndergraduateDegreeTo.$\top \sqsubseteq$ University
Fact	University(http://www.University547.edu)
Rule	University $\sqcap$ $\exists$grantedUndergraduateDegreeTo.$\top \sqsubseteq$ University
Fact	University(http://www.University900.edu)
Rule	University $\sqcap$ $\exists_2$grantedDoctoralDegreeTo.($\exists_4$advisorOf.($\top$, GraduateStudent, GraduateStudent $\sqcap$ ResearchAssistant, GraduateStudent $\sqcap$ TeachingAssistant) $\sqcap$ $\leq_4$ advisorOf.$\top$ $\sqcap$ $\exists_{17}$authorOfPublication.($\top$, Publication, Publication, Publication, Publication, Publication, $\top$, $\top$, Publication, Publication, Publication, $\top$, Publication, Publication, Publication, Publication, Publication) $\sqcap$ $\leq_{17}$ authorOfPublication.$\top$, $\exists_{14}$advisorOf.(UndergraduateStudent, UndergraduateStudent, UndergraduateStudent, UndergraduateStudent, $\top$, $\top$, GraduateStudent, GraduateStudent $\sqcap$ ResearchAssistant, GraduateStudent, $\top$, TeachingAssistant $\sqcap$ GraduateStudent, GraduateStudent, GraduateStudent $\sqcap$ TeachingAssistant, GraduateStudent $\sqcap$ TeachingAssistant) $\sqcap$ $\leq_{14}$ advisorOf.$\top$ $\sqcap$ $\exists_9$authorOfPublication.($\top$, Publication, Publication, Publication, Publication, Publication, Publication, $\top$, Publication) $\sqcap$ $\leq_9$ authorOfPublication.$\top$) $\sqcap$ $\leq_2$ grantedDoctoralDegreeTo.$\top$ $\sqcap$ $\exists_1$grantedMastersDegreeTo.(Lecturer) $\sqcap$ $\leq_1$ grantedMastersDegreeTo.$\top \sqsubseteq$ University

Table D.6: Randomly sampled explanatory rules that have been reduced in size, shown with the corresponding facts produced by the MAGCN. The rules are grouped by similar bodies / heads. These rules come from 5 MAGCNs trained with different random seeds.

Bibliography

[1] Ralph Abboud, Ismail Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. BoxE: A box embedding model for knowledge base completion. *Advances in Neural Information Processing Systems*, 33:9649–9661, 2020.

[2] Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2112–2118. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/291. Main Track.

[3] Sam Adam-Day, Michael Benedikt, Ismail Ceylan, and Ben Finkelshtein. Almost surely asymptotically constant graph neural networks. *Advances in Neural Information Processing Systems*, 37:124843–124886, 2024.

[4] John Ahlgren and Shiu Yin Yuen. Efficient program synthesis using constraint satisfaction in inductive logic programming. *The Journal of Machine Learning Research*, 14(1):3649–3682, 2013.

[5] Adem RN Aouichaoui, Fan Fan, Seyed Soheil Mansouri, Jens Abildskov, and Gurkan Sin. Combining group-contribution concept and graph neural networks toward interpretable molecular property models. *Journal of Chemical Information and Modeling*, 63(3):725–744, 2023.

[6] Burouj Armgaan, Manthan Dalmia, Sourav Medya, and Sayan Ranu. Graphtrail: Translating GNN predictions into human-interpretable logical rules. *Advances in Neural Information Processing Systems*, 37:123443–123470, 2024.

[7] Burouj Armgaan, Eshan Jain, Harsh Pandey, Mahesh Chandran, and Sayan Ranu. GNNXemplar: Exemplars to explanations–natural language rules for global GNN interpretability. *NeurIPS*, 2025.

[8] John A Arnott and Sonia Lobo Planey. The influence of lipophilicity in drug discovery and design. *Expert opinion on drug discovery*, 7(10):863–875, 2012.

[9] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[10] Bela Bollob AS. Modern graph theory. volume 184 of graduate texts in mathematics. *Springer, New York NY*, 2:4, 1998.

[11] James Atwood and Don Towsley. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[12] Waïss Azizian and Marc Lelarge. Expressive power of invariant and equivariant graph neural networks. In *ICLR 2021-International Conference on Learning Representations*, 2021.

[13] S Azzolin, A Longa, P Barbiero, P Lio, A Passerini, et al. Global explainability of GNNs via logic combination of learned concepts. In *11th International Conference on Learning Representations, ICLR 2023*. International Conference on Learning Representations, ICLR, 2023.

[14] Steve Azzolin, Sagar Malhotra, Andrea Passerini, and Stefano Teso. Beyond topological self-explainable GNNs: A formal explainability perspective. In *Volume 267: International Conference on Machine Learning, 13-19 July 2025, Vancouver Convention Center, Vancouver, Canada*, 2025.

[15] Franz Baader. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[16] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *Journal of Machine Learning Research*, 18(109):1–67, 2017.

[17] Ivana Balazevic, Carl Allen, and Timothy Hospedales. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

[18] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Hypernetwork knowledge graph embeddings. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28*, pages 553–565. Springer, 2019.

[19] Pablo Barceló, Egor Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan-Pablo Silva. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations (ICLR 2020)*, 2020.

[20] Pablo Barceló, Floris Geerts, Matthias Lanzinger, Klara Pakhomenko, and Jan Van den Bussche. A logical view of GNN-style computation and the role of activation functions. *arXiv preprint arXiv:2512.19332*, 2025.

[21] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[22] Maya Bechler-Speicher, Amir Globerson, and Ran Gilad-Bachrach. The intelligible and effective graph neural additive network. *Advances in Neural Information Processing Systems*, 37:90552–90578, 2024.

[23] Bartosz Bednarczyk, Maja Orłowska, Anna Pacanowska, and Tony Tan. On classical decidable logics extended with percentage quantifiers and arithmetics. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 2021.

[24] M Benedikt, CH Lu, B Motik, and T Tan. Decidability of graph neural networks via logical characterizations. *Leibniz International Proceedings in Informatics Lipics*, 297, 2024.

[25] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[26] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[27] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[28] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022.

[29] Jeroen Bollen, Jan Van den Bussche, Stijn Vansummeren, and Jonni Virtema. Halting recurrent GNNs and the graded mu-calculus. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 22, pages 175–184, 2025.

[30] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of the AAAI conference on artificial intelligence*, volume 25, pages 301–306, 2011.

[31] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *Artificial intelligence and statistics*, pages 127–135. PMLR, 2012.

[32] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[33] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 615–620. Association for Computational Linguistics (ACL), 2014.

[34] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data: Application to word-sense disambiguation. *Machine Learning*, 94:233–259, 2014.

[35] Antoine Bordes, Jason Weston, and Nicolas Usunier. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I 14*, pages 165–180. Springer, 2014.

[36] Guillaume Bouchard, Sameer Singh, and Theo Trouillon. On approximate reasoning capabilities of low-rank vector spaces. In *2015 AAAI Spring Symposium Series*, 2015.

[37] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2022.

[38] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[39] L Breiman, J Friedman, CJ Stone, and RA Olshen. Classification and regression trees, taylor & francis. *Classification and Regression Trees. Taylor & Francis*, 1984.

[40] Fabio Broccatelli, Ignacio Aliagas, and Hao Zheng. Why decreasing lipophilicity alone is often not a reliable strategy for extending IV half-life. *ACS Medicinal Chemistry Letters*, 9(6):522–527, 2018.

[41] Matthias Bröcheler, Lilyana Mihalkova, and Lise Getoor. Probabilistic similarity logic. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 73–82, 2010.

[42] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[43] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4): 389–410, 1992.

[44] Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5103–5113, 2021.

[45] Ling Cai, Bo Yan, Gengchen Mai, Krzysztof Janowicz, and Rui Zhu. TransGCN: Coupling transformation assumptions with graph convolutional networks for link prediction. In *Proceedings of the 10th international conference on knowledge capture*, pages 131–138, 2019.

[46] Sicong Cao, Xiaobing Sun, Xiaoxue Wu, David Lo, Lili Bo, Bin Li, and Wei Liu. Coca: Improving and explaining graph neural network-based vulnerability detection systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13, 2024.

[47] Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. Dual quaternion knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6894–6902, 2021.

[48] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2020.

[49] Xinglong Chang, Jianrong Wang, Mingxiang Wen, Yingkui Wang, and Yuxiao Huang. M-Graphormer: Multi-channel graph transformer for node representation learning. *IEEE Transactions on Big Data*, 11(4):1867–1878, 2024.

[50] Chen Chen, Yufei Wang, Bing Li, and Kwok-Yan Lam. Knowledge is flat: A seq2seq generative framework for various knowledge graph completion. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4005–4017, 2022.

[51] Chen Chen, Yufei Wang, Aixin Sun, Bing Li, and Kwok-Yan Lam. Dipping PLMs sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11489–11503, 2023.

[52] Guangyong Chen, Pengfei Chen, Chang-Yu Hsieh, Chee-Kong Lee, Benben Liao, Renjie Liao, Weiwen Liu, Jiezhong Qiu, Qiming Sun, Jie Tang, et al. Alchemy: A quantum chemistry dataset for benchmarking AI models. *arXiv preprint arXiv:1906.09427*, 2019.

[53] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, 2018.

[54] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. Knowledge graph completion: A review. *Ieee Access*, 8:192435–192456, 2020.

[55] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020.

[56] YANG Cheng, LIU Zhiyuan, TU Cunchao, and SUN Maosong. Network representation learning: an overview. *Scientia sinica informationis*, 47(8): 980–996, 2017.

[57] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266, 2019.

[58] Bonggeun Choi and Youngjoong Ko. Knowledge graph extension with a pre-trained language model via unified learning method. *Knowledge-Based Systems*, 262:110245, 2023.

[59] Bonggeun Choi, Daesik Jang, and Youngjoong Ko. Mem-kgc: Masked entity model for knowledge graph completion with pre-trained language model. *IEEE Access*, 9:132025–132032, 2021.

[60] William W Cohen. Tensorlog: A differentiable deductive database. *arXiv preprint arXiv:1605.06523*, 2016.

[61] Andrew Cropper. Learning logic programs though divide, constrain, and conquer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6446–6453, 2022.

[62] Andrew Cropper and Sebastijan Dumančić. Inductive logic programming at 30: a new introduction. *Journal of Artificial Intelligence Research*, 74: 765–850, 2022.

[63] Bernardo Cuenca Grau, Eva Feng, et al. The correspondence between bounded graph neural networks and fragments of first-order logic. *AAAI*, 2026.

[64] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[65] Enyan Dai and Suhang Wang. Towards self-explainable graph neural network. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 302–311, 2021.

[66] Enyan Dai and Suhang Wang. Towards prototype-based self-explainable graph neural network. *ACM Transactions on Knowledge Discovery from Data*, 19 (2):1–20, 2025.

[67] Guoquan Dai, Xizhao Wang, Xiaoying Zou, Chao Liu, and Si Cen. Mrgat: multi-relational graph attention network for knowledge graph completion. *Neural Networks*, 154:234–245, 2022.

[68] Minyi Dai, Mehmet F Demirel, Yingyu Liang, and Jia-Mian Hu. Graph neural networks for an accurate and interpretable prediction of the properties of polycrystalline materials. *npj Computational Materials*, 7(1):103, 2021.

[69] Tu Dinh Nguyen Dai Quoc Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of NAACL-HLT*, pages 327–333, 2018.

[70] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th international conference on semantic systems*, pages 121–124, 2013.

[71] Abdelkader Dairi, Fouzi Harrou, and Ying Sun. Auto-configured explainable graph neural networks for multi-site pollution prediction. *Applied Soft Computing*, 181:113496, 2025.

[72] Danica Damljanovic and Kalina Bontcheva. Named entity disambiguation using linked data. In *Proceedings of the 9th extended semantic web conference*, pages 231–240, 2012.

[73] George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph neural networks for node disambiguation. *International Joint Conference on Artificial Intelligence*, pages 2126–2132, 2020.

[74] Alex Davies and Nirav Ajmeri. Realistic synthetic social networks with graph neural networks. *arXiv preprint arXiv:2212.07843*, 2022.

[75] Caglar Demir and Axel-Cyrille Ngonga Ngomo. Convolutional complex knowledge graph embeddings. In *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*, pages 409–424. Springer, 2021.

[76] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[77] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610, 2014.

[78] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.

[79] Alexandre Duval and Fragkiskos D Malliaros. GraphSVX: Shapley value explanations for graph neural networks. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 302–318. Springer, 2021.

[80] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[81] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[82] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

[83] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

[84] Miao Fan, Qiang Zhou, Emily Chang, and Fang Zheng. Transition-based knowledge graph embedding with relational mapping properties. In *Proceedings of the 28th Pacific Asia conference on language, information and computing*, pages 328–337, 2014.

[85] Aosong Feng, Chenyu You, Shiqiang Wang, and Leandros Tassiulas. Kergnns: Interpretable graph neural networks with graph kernels. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 6614–6622, 2022.

[86] Jun Feng, Minlie Huang, Mingdong Wang, Mantong Zhou, Yu Hao, and Xiaoyan Zhu. Knowledge graph embedding by flexible translation. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2016.

[87] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017.

[88] Morgan R Frank, Dashun Wang, Manuel Cebrian, and Iyad Rahwan. The evolution of citation graphs in artificial intelligence research. *Nature Machine Intelligence*, 1(2):79–85, 2019.

[89] Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. Understanding and extending subgraph GNNs by rethinking their symmetries. *Advances in Neural Information Processing Systems*, 35:31376–31390, 2022.

[90] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6):707–730, 2015.

[91] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422, 2013.

[92] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1416–1424, 2018.

[93] Jianfei Gao and Bruno Ribeiro. On the equivalence between temporal and static equivariant graph representations. In *International Conference on Machine Learning*, pages 7052–7076. PMLR, 2022.

[94] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. Effective blending of two and three-way interactions for modeling multi-relational data. In *Machine Learning and Knowledge Discovery in Databases: European*

Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I 14, pages 434–449. Springer, 2014.

[95] Marta Garnelo and Murray Shanahan. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29:17–23, 2019.

[96] Floris Geerts, Jasper Steegmans, and Jan Van den Bussche. On the expressive power of message-passing neural networks as global feature map transformers. In *International Symposium on Foundations of Information and Knowledge Systems*, pages 20–34. Springer, 2022.

[97] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[98] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*. MIT press, 2007.

[99] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[100] Yu V Glebskii, Dmitry I Kogan, MI Liogon'kiI, and Vladimir A Talanov. Range and degree of realizability of formulas in the restricted predicate calculus. *Cybernetics*, 5(2):142–154, 1969.

[101] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[102] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[103] Martin Grohe. The descriptive complexity of graph neural networks. *TheoretiCS*, 3, 2024.

[104] Florian Grötschla, Jiaqing Xie, and Roger Wattenhofer. Benchmarking positional encodings for GNNs and graph transformers. *arXiv preprint arXiv:2411.12732*, 2024.

[105] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[106] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[107] Lingbing Guo, Qingheng Zhang, Weiyi Ge, Wei Hu, and Yuzhong Qu. DSKG: A deep sequential model for knowledge graph completion. In *Knowledge Graph and Semantic Computing. Knowledge Computing and Language Understanding: Third China Conference, CCKS 2018, Tianjin, China, August 14–17, 2018, Revised Selected Papers*, pages 65–77. Springer, 2019.

[108] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2-3):158–182, 2005.

[109] Zhimeng Guo, Zongyu Wu, Teng Xiao, Charu Aggarwal, Hui Liu, and Suhang Wang. Counterfactual learning on graphs: A survey. *Machine Intelligence Research*, 22(1):17–59, 2025.

[110] Benjamin Gutteridge, Xiaowen Dong, Michael M Bronstein, and Francesco Di Giovanni. Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, pages 12252–12267. PMLR, 2023.

[111] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1802–1808, 2017.

[112] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[113] Stan P Hauke and Przemysław Andrzej Wałęga. Aggregate-combine-readout GNNs are more expressive than logic C2. *arXiv preprint arXiv:2508.06091*, 2025.

[114] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 623–632, 2015.

[115] Larry Heck, Dilek Hakkani-Tür, and Gokhan Tur. Leveraging knowledge graphs for web-scale unsupervised semantic parsing. In *Proceedings of InterSpeech*, 2013.

[116] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[117] Céline Hocquette, Andreas Niskanen, Matti Järvisalo, and Andrew Cropper. Learning MDL logic programs from noisy data. *arXiv preprint arXiv:2308.09393*, 2023.

[118] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 541–550, 2011.

[119] Andrew L Hopkins, György M Keserü, Paul D Leeson, David C Rees, and Charles H Reynolds. The role of ligand efficiency metrics in drug discovery. *Nature reviews Drug discovery*, 13(2):105–121, 2014.

[120] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. GraphMAE: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022.

[121] Xinyue Hu, Zenan Sun, Yi Nian, Yichen Wang, Yifang Dang, Fang Li, Jingna Feng, Evan Yu, and Cui Tao. Self-explainable graph neural network for alzheimer disease and related dementias risk prediction: Algorithm development and validation study. *JMIR aging*, 7(1):e54748, 2024.

[122] Xingyue Huang, Miguel Romero, Ismail Ceylan, and Pablo Barceló. A theory of link prediction via relational weisfeiler-leman on knowledge graphs. *Advances in Neural Information Processing Systems*, 36:19714–19748, 2023.

[123] Xingyue Huang, Pablo Barcelo, Michael M Bronstein, Ismail Ilkan Ceylan, Mikhail Galkin, Juan L Reutter, and Miguel Romero Orth. How expressive are knowledge graph foundation models? In *International Conference on Machine Learning*, pages 25021–25058. PMLR, 2025.

[124] Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Global counterfactual explainer for graph neural networks. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pages 141–149, 2023.

[125] Eleni Ilkou and Maria Koutraki. Symbolic vs sub-symbolic AI methods: Friends or enemies? In *CIKM (Workshops)*, volume 2699, 2020.

[126] Katsumi Inoue, Andrei Doncescu, and Hidetomo Nabeshima. Completing causal networks by meta-level abduction. *Machine learning*, 91(2):239–277, 2013.

[127] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 687–696, 2015.

[128] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[129] Emily Jin, Michael M Bronstein, Ismail Ilkan Ceylan, and Matthias Lanzinger. Homomorphism counts for graph neural networks: All about that basis. In *International Conference on Machine Learning*, pages 22075–22098. PMLR, 2024.

[130] Ataollah Kamal, Alessio Ragno, Marc Plantevit, and Céline Robardet. Leveraging internal representations of GNNs with shapley values. *Data Mining and Knowledge Discovery*, 39(6):1–41, 2025.

[131] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31, 2018.

[132] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, page 5, 2006.

[133] Niklas Kemper, Tom Wollschläger, and Stephan Günnemann. What expressivity theory misses: Message passing complexity for GNNs. *Advances in neural information processing systems*, 2025.

[134] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[135] Sammy Khalife. The logic of rational graph neural networks. *arXiv preprint arXiv:2310.13139*, 2023.

[136] Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. Multi-task learning for knowledge graph completion with pre-trained language models. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1737–1743, 2020.

[137] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[138] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[139] Dominik Köhler and Stefan Heindorf. Utilizing description logics for global explanations of heterogeneous graph neural networks. *arXiv preprint arXiv:2405.12654*, 2024.

[140] Harold W Kuhn and Albert William Tucker. *Contributions to the Theory of Games*. Princeton University Press, 1953.

[141] Orna Kupferman, Ulrike Sattler, and Moshe Y Vardi. The complexity of the graded mu-calculus. In *International Conference on Automated Deduction*, pages 423–437. Springer, 2002.

[142] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. Fast and exact rule mining with amie 3. In *European Semantic Web Conference*, pages 36–52. Springer, 2020.

[143] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4558–4567, 2018.

[144] Ni Lao and William W Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81:53–67, 2010.

[145] Hannah Lawrence, Vasco Portilheiro, Yan Zhang, and Sékou-Oumar Kaba. Improving equivariant networks with probabilistic symmetry breaking. In *The Thirteenth International Conference on Learning Representations*, 2025.

[146] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[147] Kai Lei, Jin Zhang, Yuexiang Xie, Desi Wen, Daoyuan Chen, Min Yang, and Ying Shen. Path-based reasoning with constrained type attention for knowledge graph completion. *Neural Computing and Applications*, 32:6957–6966, 2020.

[148] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[149] Da Li, Boqing Zhu, Sen Yang, Kele Xu, Ming Yi, Yukai He, and Huaimin Wang. Multi-task pre-training language model for semantic network completion. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 2022.

[150] Michelle M Li, Kexin Huang, and Marinka Zitnik. Graph representation learning in biomedicine and healthcare. *Nature Biomedical Engineering*, pages 1–17, 2022.

[151] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

[152] Wei Li, Luyao Zhu, Rui Mao, and Erik Cambria. SKIER: A symbolic knowledge integrated model for conversational emotion recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13121–13129, 2023.

[153] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. In *ICML*, pages 1–21, 2018.

[154] Leonid Libkin. *Elements of finite model theory*, volume 41. Springer, 2004.

[155] Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

[156] Qika Lin, Rui Mao, Jun Liu, Fangzhi Xu, and Erik Cambria. Fusing topology contexts and logical rules in language models for knowledge graph completion. *Information Fusion*, 90:253–264, 2023.

[157] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[158] Hanxiao Liu, Yuexin Wu, and Yiming Yang. Analogical inference for multi-relational embeddings. In *International conference on machine learning*, pages 2168–2178. PMLR, 2017.

[159] Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*, 2016.

[160] Shengchao Liu, Chengpeng Wang, Jiarui Lu, Weili Nie, Hanchen Wang, Zhuoxinran Li, Bolei Zhou, and Jian Tang. Unsupervised discovery of steerable factors when graph deep generative models are entangled. *Transactions on Machine Learning Research*, 2024.

[161] Shuwen Liu, Bernardo Cuenca Grau, Ian Horrocks, and Egor Kostylev. INDIGO: GNN-based inductive knowledge graph completion using pair-wise encoding. *Advances in Neural Information Processing Systems*, 34:2034–2045, 2021.

[162] Shuwen Liu, Bernardo Cuenca Grau, Ian Horrocks, and Egor V Kostylev. Revisiting inferential benchmarks for knowledge graph completion. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, 2023.

[163] Xiyang Liu, Huobin Tan, Qinghong Chen, and Guangyan Lin. RAGAT: Relation aware graph attention network for knowledge graph completion. *IEEE Access*, 9:20840–20849, 2021.

[164] Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[165] Daniel Lowd and Pedro Domingos. Efficient weight learning for markov logic networks. In *European conference on principles of data mining and knowledge discovery*, pages 200–211. Springer, 2007.

[166] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. CF-GNNExplainer: Counterfactual explanations for graph neural networks. In *International conference on artificial intelligence and statistics*, pages 4499–4511. PMLR, 2022.

[167] 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.

[168] Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. Do pre-trained models benefit knowledge graph completion? a

reliable evaluation and a reasonable approach. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3570–3581, 2022.

[169] Lianbo Ma, Peng Sun, Zhiwei Lin, and Hui Wang. Composing knowledge graph embeddings via word embeddings. *arXiv preprint arXiv:1909.03794*, 2019.

[170] Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, pages 23321–23337. PMLR, 2023.

[171] T Lucas Makinen, Justin Alsing, and Benjamin D Wandelt. Fishnets: Information-optimal, scalable aggregation for sets and graphs. *Monthly Notices of the Royal Astronomical Society*, page 2, 2023.

[172] Nour Makke and Sanjay Chawla. Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review*, 57(1):2, 2024.

[173] D Marcheggiani and I Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP 2017-Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 1506–1515, 2017.

[174] Diego Marcheggiani, Jasmijn Bastings, and Ivan Titov. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of NAACL-HLT*, pages 486–492, 2018.

[175] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *International Conference on Learning Representations*, 2018.

[176] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

[177] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pages 4363–4371. PMLR, 2019.

[178] Sohir Maskey, Raffaele Paolino, Fabian Jogl, Gitta Kutyniok, and Johannes F Lutzeyer. Graph representational learning: When does more expressivity hurt generalization? *arXiv preprint arXiv:2505.11298*, 2025.

[179] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3137–3143, 2019.

[180] Elliott Mendelson. *Introduction to mathematical logic*. Chapman and Hall/CRC, 2009.

[181] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. ChEMBL: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1):D930–D940, 2019.

[182] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

[183] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International conference on machine learning*, pages 15524–15543. PMLR, 2022.

[184] Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. In *International Conference on Machine Learning*, pages 6938–6949. PMLR, 2020.

[185] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *Advances in neural information processing systems*, 30, 2017.

[186] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[187] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[188] Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. *Advances in Neural Information Processing Systems*, 33:21824–21840, 2020.

[189] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far. *Journal of Machine Learning Research*, 24(333):1–59, 2023.

[190] Matthew Morris and Ian Horrocks. Sound logical explanations for mean aggregation graph neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[191] Matthew Morris, Thomas D Barrett, and Arnu Pretorius. Universally expressive communication in multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS) 35*, 2022.

[192] Matthew Morris, Bernardo Cuenca Grau, and Ian Horrocks. Orbit-equivariant graph neural networks. In *International Conference on Learning Representations*, pages 5221–5259, 2024.

[193] Matthew Morris, David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. Relational graph convolutional networks do not learn sound rules. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning*, volume 21, pages 897–908, 2024.

[194] Matthew Morris, David Tena Cucala, and Bernardo Cuenca Grau. Logical expressivity and explanations for monotonic GNNs with scoring functions. In *Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning*, volume 22, pages 477–488, 2025.

[195] Stephen H Muggleton, José Carlos Almeida Santos, and Alireza Tamaddoni-Nezhad. TopLog: ILP using a logic program declarative bias. In *International Conference on Logic Programming*, pages 687–692. Springer, 2008.

[196] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723, 2019.

[197] Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, et al. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 3104482–3104584, 2011.

[198] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280, 2012.

[199] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[200] Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: scaling up statistical inference in markov logic networks using an rdbms. *Proceedings of the VLDB Endowment*, 4(6):373–384, 2011.

[201] Pierre Nunn, Marco Sälzer, François Schwarzentruber, and Nicolas Troquard. A logic for reasoning about aggregate-combine graph neural networks. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 3532–3540, 2024.

[202] R Scott Obach, Franco Lombardo, and Nigel J Waters. Trend analysis of a database of intravenous pharmacokinetic parameters in humans for 670 drug compounds. *Drug Metabolism and Disposition*, 36(7):1385–1405, 2008.

[203] Euan Ong and Petar Veličković. Learnable commutative monoids for graph neural networks. In *Learning on Graphs Conference*, pages 43–1. PMLR, 2022.

[204] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*, 2023.

[205] Pál András Papp and Roger Wattenhofer. A theoretical comparison of graph neural network extensions. In *International Conference on Machine Learning*, pages 17323–17345. PMLR, 2022.

[206] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. DropGNN: Random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:21997–22009, 2021.

[207] Aleksandar Pavlović and Emanuel Sallinger. Expressive: A spatio-functional embedding for knowledge graph completion. In *The Eleventh International Conference on Learning Representations*, 2022.

[208] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[209] Maximilian Pflueger, David Tena Cucala, and Egor V Kostylev. Recurrent graph neural networks and their connections to bisimulation and logic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14608–14616, 2024.

[210] Alexander Pluska, Pascal Welke, Thomas Gärtner, and Sagar Malhotra. Logical distillation of graph neural networks. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning*, pages 920–930, 2024.

[211] Malte Probst. The set autoencoder: Unsupervised representation learning for sets. *openreview*, 2018.

[212] Michela Proietti, Alessio Ragno, Biagio La Rosa, Rino Ragno, and Roberto Capobianco. Explainable AI in drug discovery: self-interpretable graph neural network for molecular property prediction using concept whitening. *Machine Learning*, 113(4):2013–2044, 2024.

[213] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In *International semantic web conference*, pages 542–557. Springer, 2013.

[214] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, PTE, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 459–467, 2018.

[215] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. NetSMF: Large-scale network embedding as sparse matrix factorization. In *The World Wide Web Conference*, pages 1509–1520, 2019.

[216] Alessio Ragno, Biagio La Rosa, and Roberto Capobianco. Prototype-based interpretable graph neural networks. *IEEE Transactions on Artificial Intelligence*, 5(4):1486–1495, 2022.

[217] Alessio Ragno, Marc Plantevit, and Céline Robardet. On logic-based self-explainable graph neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[218] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35: 14501–14515, 2022.

[219] RDKit. Matched molecular pair database generation and analysis. `https://github.com/rdkit/mmpdb`, 2023.

[220] RDKit. Rdkit: Open-source cheminformatics. `https://www.rdkit.org`, 2023.

[221] Tony Ribeiro and Katsumi Inoue. Learning prime implicant conditions from interpretation transition. In *Inductive Logic Programming: 24th International Conference, ILP 2014, Nancy, France, September 14-16, 2014, Revised Selected Papers*, pages 108–125. Springer, 2015.

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

[223] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 74–84, 2013.

[224] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. *Advances in neural information processing systems*, 30, 2017.

[225] Eran Rosenbluth, Jan Toenshoff, and Martin Grohe. Some might say all you need is sum. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 4172–4179, 2023.

[226] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[227] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*, 32, 2019.

[228] Marco Sälzer, Przemysław Andrzej Wałęga, and Martin Lange. The logical expressiveness of temporal GNNs via two-dimensional product logics. *NeurIPS*, 2025.

[229] Ryoma Sato. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020.

[230] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. *Advances in neural information processing systems*, 32, 2019.

[231] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 333–341. SIAM, 2021.

[232] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[233] Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828, 2022.

[234] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

[235] Moritz Schönherr and Carsten Lutz. Logical characterizations of GNNs with mean aggregation. *arXiv preprint arXiv:2507.18145*, 2025.

[236] Gesina Schwalbe and Bettina Finzel. A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. *Data Mining and Knowledge Discovery*, 38(5):3043–3101, 2024.

[237] Sangwoo Seo, Sungwon Kim, and Chanyoung Park. Interpretable prototype-based graph information bottleneck. *Advances in Neural Information Processing Systems*, 36:76737–76748, 2023.

[238] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3060–3067, 2019.

[239] Jianhao Shen, Chenguang Wang, Linyuan Gong, and Dawn Song. Joint language semantic and structure embedding for knowledge graph completion. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1965–1978, 2022.

[240] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26, 2013.

[241] Arie Soeteman and Balder ten Cate. Logical expressiveness of graph neural networks with hierarchical node individualization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[242] Arie Soeteman, Michael Benedikt, Martin Grohe, and Balder ten Cate. How expressive are graph neural networks in the presence of node identifiers? *arXiv preprint arXiv:2601.21882*, 2026.

[243] Amauri Souza, Diego Mesquita, Samuel Kaski, and Vikas Garg. Provably expressive temporal graph networks. *Advances in neural information processing systems*, 35:32257–32269, 2022.

[244] Timo Speith. A review of taxonomies of explainable artificial intelligence (XAI) methods. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pages 2239–2250, 2022.

[245] Ashwin Srinivasan. The aleph manual. *Machine Learning at the Computing Laboratory*, 2001.

[246] Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations. In *International Conference on Learning Representations*, 2019.

[247] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[248] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and Philip S Yu. Graph structure learning with variational information bottleneck. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 4165–4174, 2022.

[249] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2018.

[250] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[251] Jianheng Tang, Fengrui Hua, Ziqi Gao, Peilin Zhao, and Jia Li. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. *Advances in Neural Information Processing Systems*, 36, 2024.

[252] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo. RGCNN: Regularized graph CNN for point cloud segmentation. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 746–754, 2018.

[253] David Tena Cucala and Bernardo Cuenca Grau. Bridging max graph neural networks and datalog with negation. In *Proceedings of the international conference on principles of knowledge representation and reasoning*, volume 21, pages 950–961, 2024.

[254] David Tena Cucala, Bernardo Cuenca Grau, Egor V Kostylev, and Boris Motik. Explainable GNN-based models over knowledge graphs. *ICLR*, 2022.

[255] David Tena Cucala, Bernardo Cuenca Grau, and Boris Motik. Faithful approaches to rule learning. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, pages 484–493, 2022.

[256] David Tena Cucala, Bernardo Cuenca Grau, Boris Motik, and Egor V Kostylev. On the correspondence between monotonic max-sum GNNs and datalog. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, pages 658–667, 2023.

[257] David Tena Cucala, Bernardo Cuenca Grau, Boris Motik, and Egor V. Kostylev. From monotonic graph neural networks to datalog and back: Expressive power and practical applications. *Artificial Intelligence*, 2026. Forthcoming.

[258] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR, 2020.

[259] Anqi Tian, Chunhong Zhang, Miao Rang, Xueying Yang, and Zhiqiang Zhan. RA-GCN: Relational aggregation graph convolutional network for knowledge graph completion. In *Proceedings of the 2020 12th international conference on machine learning and computing*, pages 580–586, 2020.

[260] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66, 2015.

[261] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[262] Ke Tu, Peng Cui, Xiao Wang, Philip S Yu, and Wenwu Zhu. Deep recursive network embedding with regular equivalence. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2357–2366, 2018.

[263] Mathias Niemann Tygesen, Francisco Camara Pereira, and Filipe Rodrigues. Unboxing the graph: Towards interpretable graph neural networks for transport prediction through neural relational inference. *Transportation research part C: emerging technologies*, 146:103946, 2023.

[264] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2019.

[265] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3009–3016, 2020.

[266] Antonis Vasileiou, Ben Finkelshtein, Floris Geerts, Ron Levie, and Christopher Morris. Covered forest: Fine-grained generalization analysis of graph neural

networks. In *Forty-second International Conference on Machine Learning*, 2025.

[267] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[268] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.

[269] Alessio Verdone, Simone Scardapane, and Massimo Panella. Explainable spatio-temporal graph neural networks for multi-site photovoltaic energy production. *Applied Energy*, 353:122151, 2024.

[270] Luca Veyrin-Forrer, Ataollah Kamal, Stefan Duffner, Marc Plantevit, and Céline Robardet. What does my GNN really capture? on exploring internal GNN representations. In *International Joint Conference on Artificial Intelligence 2022*, 2022.

[271] Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. *Advances in neural information processing systems*, 33:14143–14155, 2020.

[272] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[273] Przemysław Andrzej Wałęga and Bernardo Cuenca Grau. Preservation theorems for unravelling-invariant classes: A uniform approach for modal logics and graph neural networks. *arXiv preprint arXiv:2602.01856*, 2026.

[274] Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*, pages 1737–1748, 2021.

[275] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[276] Dingmin Wang and Yeyuan Chen. Calibrate and boost logical expressiveness of GNN over multi-relational and temporal graphs. *Advances in Neural Information Processing Systems*, 36:66692–66726, 2023.

[277] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. *arXiv preprint arXiv:2203.00199*, 2022.

[278] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. RippleNet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 417–426, 2018.

[279] Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. SimKGC: Simple contrastive knowledge graph completion with pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294, 2022.

[280] Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang, and Chin-Yew Lin. Knowledge base completion via coupled path ranking. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1308–1318, 2016.

[281] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[282] Xiaxia Wang, David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. Faithful rule extraction for differentiable rule learning models. In *The Twelfth International Conference on Learning Representations*, 2023.

[283] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[284] Zhe Wang, Suxue Ma, Kewen Wang, and Zhiqiang Zhuang. Rule-guided graph neural networks for explainable knowledge graph reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12784–12791, 2025.

[285] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

[286] Michael J Waring. Lipophilicity in drug discovery. *Expert Opinion on Drug Discovery*, 5(3):235–248, 2010.

[287] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

[288] Sean Welleck, Zixin Yao, Yu Gai, Jialin Mao, Zheng Zhang, and Kyunghyun Cho. Loss functions for multiset prediction. *Advances in Neural Information Processing Systems*, 31, 2018.

[289] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526, 2014.

[290] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, 2013.

[291] Junkang Wu, Wentao Shi, Xuezhi Cao, Jiawei Chen, Wenqiang Lei, Fuzheng Zhang, Wei Wu, and Xiangnan He. DisenKGAT: knowledge graph embedding with disentangled graph attention network. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 2140–2149, 2021.

[292] Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in neural information processing systems*, 35:27387–27401, 2022.

[293] Yulei Wu, Hong-Ning Dai, and Haina Tang. Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet of Things Journal*, 9(12):9214–9231, 2021.

[294] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[295] Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. TransA: An adaptive approach for knowledge graph embedding. *arXiv preprint arXiv:1509.05490*, 2015.

[296] Han Xiao, Minlie Huang, and Xiaoyan Zhu. From one point to a manifold: knowledge graph embedding for precise link prediction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1315–1321, 2016.

[297] Han Xiao, Minlie Huang, and Xiaoyan Zhu. TransG: A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2316–2325, 2016.

[298] Tian Xie, Xiang Fu, Octavian Ganea, Regina Barzilay, and Tommi Jaakkola. Crystal diffusion variational autoencoder for periodic material generation. *Bulletin of the American Physical Society*, 67, 2022.

[299] Xin Xie, Ningyu Zhang, Zhoubo Li, Shumin Deng, Hui Chen, Feiyu Xiong, Mosha Chen, and Huajun Chen. From discrimination to generation: Knowledge graph completion with generative transformer. In *Companion Proceedings of the Web Conference 2022*, pages 162–165, 2022.

[300] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[301] Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In *International Conference on Learning Representations*, 2019.

[302] Bishan Yang and Tom Mitchell. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1436–1446, 2017.

[303] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, 2015.

[304] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30, 2017.

[305] Jie Yang, Yuwen Wang, Kaixuan Chen, Tongya Zheng, Yihe Zhou, Zhenbang Xiao, Ji Cao, Mingli Song, and Shunyu Liu. From GNNs to trees: Multi-granular interpretability for graph neural networks. *ICLR*, 2025.

[306] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[307] Liang Yao, Chengsheng Mao, and Yuan Luo. KG-BERT: BERT for knowledge graph completion. *arXiv preprint arXiv:1909.03193*, 2019.

[308] Jun Yin, Chaozhuo Li, Hao Yan, Jianxun Lian, and Senzhang Wang. Train once and explain everywhere: Pre-training interpretable graph neural networks. *Advances in neural information processing systems*, 36:35277–35299, 2023.

[309] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34: 28877–28888, 2021.

[310] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[311] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[312] Jiaxuan You, Tianyu Du, and Jure Leskovec. Roland: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 2358–2366, 2022.

[313] Donghan Yu, Yiming Yang, Ruohong Zhang, and Yuexin Wu. Knowledge embedding based graph convolutional network. In *Proceedings of the Web Conference 2021*, pages 1619–1628, 2021.

[314] Hang Yu, Weixu Liu, Nengjun Zhu, Pengbo Li, and Xiangfeng Luo. INGFD: An interpretable graph fraud detection model for spam reviews. *IEEE Transactions on Artificial Intelligence*, 5(10):5325–5339, 2024.

[315] Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. *arXiv preprint arXiv:2010.05563*, 2020.

[316] Junchi Yu, Jie Cao, and Ran He. Improving subgraph recognition with variational graph information bottleneck. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19396–19405, 2022.

[317] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning*, pages 12241–12252. PMLR, 2021.

[318] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5782–5799, 2022.

[319] Mohamad Zamini, Hassan Reza, and Minou Rabiei. A review of knowledge graph completion. *Information*, 13(8):396, 2022.

[320] Qiang Zeng, Jignesh M Patel, and David Page. Quickfoil: Scalable inductive logic programming. *Proceedings of the VLDB Endowment*, 8(3):197–208, 2014.

[321] Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of GNNs via graph biconnectivity. *The Eleventh International Conference on Learning Representations*, 2023.

[322] Bohang Zhang, Jingchu Gai, Yiheng Du, Qiwei Ye, Di He, and Liwei Wang. Beyond weisfeiler-lehman: A quantitative framework for gnn expressiveness. *The Twelfth International Conference on Learning Representations*, 2024.

[323] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Yan Yeung. GaAN: Gated attention networks for learning on large and spatiotemporal graphs. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 2018.

[324] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. Prone: Fast and scalable network representation learning. In *IJCAI*, volume 19, pages 4278–4284, 2019.

[325] Mengqi Zhang, Yuwei Xia, Qiang Liu, Shu Wu, and Liang Wang. Learning latent relations for temporal knowledge graph reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12617–12631, 2023.

[326] Muhan Zhang. Graph neural networks: link prediction. *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 195–223, 2022.

[327] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[328] Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. GStarX: Explaining graph neural networks with structure-aware cooperative games. *Advances in neural information processing systems*, 35:19810–19823, 2022.

[329] Shichang Zhang, Jiani Zhang, Xiang Song, Soji Adeshina, Da Zheng, Christos Faloutsos, and Yizhou Sun. Page-link: Path-based graph neural network explanation for heterogeneous link prediction. In *Proceedings of the ACM web conference 2023*, pages 3784–3793, 2023.

[330] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. *Advances in neural information processing systems*, 32, 2019.

[331] Zaixi Zhang and Qi Liu. Learning subpocket prototypes for generalizable structure-based drug design. In *International Conference on Machine Learning*, pages 41382–41398. PMLR, 2023.

[332] 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.

[333] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3065–3072, 2020.

[334] Kaizhong Zheng, Shujian Yu, Liangjun Chen, Lujuan Dang, and Badong Chen. BPI-GNN: Interpretable brain network-based psychiatric diagnosis and subtyping. *NeuroImage*, 292:120594, 2024.

[335] Zhicheng Zheng, Xiance Si, Fangtao Li, Edward Y Chang, and Xiaoyan Zhu. Entity disambiguation with freebase. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 82–89. IEEE, 2012.

[336] Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. LLMs for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *arXiv preprint arXiv:2305.13168*, 2023.

[337] Zhuotun Zhu, Xinggang Wang, Song Bai, Cong Yao, and Xiang Bai. Deep learning representation using autoencoder for 3D shape retrieval. *Neurocomputing*, 204:41–50, 2016.

[338] Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 world wide web conference*, pages 499–508, 2018.