Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Reasoning, Learning & Decision-Making Track

Relational Graph Convolutional Networks Do Not Learn Sound Rules

Matthew Morris[1] , **David J. Tena Cucala**[1,2] , **Bernardo Cuenca Grau**[1] , **Ian Horrocks**[1]
[1]Department of Compute Science, University of Oxford
[2]Department of Computer Science, Royal Holloway, University of London
{matthew.morris, bernardo.cuenca.grau, ian.horrocks}@cs.ox.ac.uk, david.tenacucala@rhul.ac.uk

Abstract

Graph neural networks (GNNs) are frequently used to predict missing facts in knowledge graphs (KGs). Motivated by the lack of explainability for the outputs of these models, recent work has aimed to explain their predictions using Datalog, a widely used logic-based formalism. However, such work has been restricted to certain subclasses of GNNs. In this paper, we consider one of the most popular GNN architectures for KGs, R-GCN, and we provide two methods to extract rules that explain its predictions and are *sound*, in the sense that each fact derived by the rules is also predicted by the GNN, for any input dataset. Furthermore, we provide a method that can verify that certain classes of Datalog rules are not sound for the R-GCN. In our experiments, we train R-GCNs on KG completion benchmarks, and we are 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 R-GCN models to generalise and about the explainability of their predictions. We further provide two variations to the training paradigm of R-GCN that encourage it to learn sound rules and find a trade-off between model accuracy and the number of learned sound rules.

1 Introduction

Knowledge graphs (KGs) are graph-structured knowledge bases where nodes and edges represent entities and their relationships, respectively (Hogan et al. 2022). KGs can be typically stored as sets of unary and binary facts, and are being exploited in an increasing range of applications (Vrandečić and Krötzsch 2014; Suchanek, Kasneci, and Weikum 2007; Bouchard, Singh, and Trouillon 2015; Hamilton, Ying, and Leskovec 2017).

Knowledge graphs are, however, often incomplete, which has led to the rapid development of the field of KG completion—the task of extending an incomplete KG with all missing facts holding in its (unknown) complete version. KG completion is typically conceptualised as a classification problem, where the aim is to learn a function that, given the incomplete KG and a candidate fact as input, decides whether the latter holds in the completion of the former. A wide range of KG approaches have been proposed in the literature, including embedding-based approaches with distance-based scoring functions (Bordes et al. 2013; Sun et al. 2018), tensor products (Nickel et al. 2011; Yang et al. 2015), box embeddings (Abboud et al. 2020), recurrent neural networks (Sadeghian et al. 2019), differentiable reasoning (Rocktäschel and Riedel 2017; Evans and Grefenstette 2018), and LLMs (Yao, Mao, and Luo 2019; Xie et al. 2022). Amongst all neural approaches to KG completion, however, those based on graph neural networks (GNNs) have received special attention (Ioannidis, Marques, and Giannakis 2019; Liu et al. 2021; Pflueger, Tena Cucala, and Kostylev 2022). These include R-GCN (Schlichtkrull et al. 2018) and its extensions (Tian et al. 2020; Cai et al. 2019; Vashishth et al. 2019; Yu et al. 2021; Shang et al. 2019; Liu et al. 2021), where the basic R-GCN model remains a common baseline for evaluating against or as part of a larger system (Gutteridge et al. 2023; Liu et al. 2023; Li et al. 2023; Tang et al. 2024; Zhang et al. 2023; Lin et al. 2023).

Although these embedding and neural-based approaches to KG completion have proved effective in practice, their predictions are difficult to explain and interpret (Garnelo and Shanahan 2019). This is in contrast to logic-based and neuro-symbolic approaches to KG completion, such as rule learning methods, where the extracted rules can be used to generate rigourous proofs explaining the prediction of any given fact. RuleN (Meilicke et al. 2018) and AnyBURL (Meilicke et al. 2018) heuristically identify Datalog rules from the given data and apply them directly for completing the input graph. RNNLogic (Qu et al. 2020) uses a probabilistic model to select the most promising rules. Other works attempt to extract Datalog rules from trained neural models, including Neural-LP (Yang, Yang, and Cohen 2017), DRUM (Sadeghian et al. 2019), and Neural Theorem Provers (Rocktäschel and Riedel 2017). As shown in (Tena Cucala, Cuenca Grau, and Motik 2022; Wang et al. 2023), however, the extracted Datalog rules are not faithful to the model in the sense that the rules may derive, for some dataset, facts that are not predicted by the model (*unsoundness*) as well as failing to derive predicted facts (*incompleteness*).

As a result, there is increasing interest in the development of neural KG completion methods whose predictions can be faithfully characterised by means of rule-based reasoning. As shown in (Tena Cucala, Cuenca Grau, and Motik 2022; Wang et al. 2023), the Neural-LP and DRUM approaches can be adapted to ensure soundness and completeness of the generated rules with respect to the model. The class

of *monotonic GNNs* (Tena Cucala et al. 2021), which use max aggregation, restrict all 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. In turn, *monotonic max-sum GNNs* (Tena Cucala et al. 2023), which relax the requirements on the aggregation function to encompass both max and sum aggregation, can be faithfully characterised by means of tree-like Datalog programs with inequalities in the body of rules. Both variants of monotonic GNNs, however, require model weights to be non-negative, which is crucial to ensure that their application to datasets is monotonic under homomorphisms—a key property of (negation-free) Datalog reasoning. The monotonicity requirement, however, is not applicable to popular GNN-based models such as R-GCN.

Our Contribution In this paper, we consider *sum-GNNs*, which use sum as aggregation function and which do not impose restrictions on model weights. These GNNs can be seen both as an extension to the GNNs in (Tena Cucala et al. 2023) with sum aggregation but without the monotonicity requirement, as well as a variant of the R-GCN model. The functions learnt by sum-GNNs may be non-monotonic: predicted facts may be invalidated by adding new facts to the input KG. As a result, these GNNs cannot, in general, be faithfully captured by (negation-free) Datalog programs. Our aim in this paper is to identify a subset of the output channels (i.e. features) of the GNN which exhibit monotonic behaviour, and for which sound Datalog rules may be extracted. The ability to extract sound rules is important as it allows us to explain model predictions associated to the identified output channels. Furthermore, we provide means for identifying *unbounded* output channels for which no sound Datalog rule exists, hence suggesting that these channels inherently exhibit non-monotonic behaviour.

We conducted experiments on the benchmark datasets by (Teru, Denis, and Hamilton 2020) and also use the rule-based LogInfer evaluation framework described in (Liu et al. 2023), which we extended to include a mixture of monotonic and non-monotonic rules. Our experiments show that even under ideal scenarios, without restrictions on the training process, all the channels in the trained GNN are unbounded (even for monotonic LogInfer benchmarks), which implies that there are no sound Datalog rules for the model. We then consider two adjustments 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, we observe that an increasing number of output channels exhibit monotonic behaviour and we obtain more sound rules, although the model accuracy diminishes. Hence, there is a trade-off between model performance and rule extraction.

2 Background

Datalog We fix a signature of countably infinite, disjoint sets of predicates and constants, where each predicate is associated with a non-negative integer arity. We also consider a countably infinite set of variables disjoint with the sets of predicates and constants. 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 arity n. A literal is an atom or any inequality $t_1 \not\approx t_2$. 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. A (Datalog) rule is an expression of the form

$$B_1 \wedge ... \wedge B_n \rightarrow H, \tag{1}$$

where $B_1, ..., B_n$ are its body literals and H is its head atom. We use the standard safety requirements for rules: every variable that appears in a rule must occur in a body atom. Furthermore, to avoid vacuous 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 ground atom 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. The immediate consequence operator T_r for a rule r of form (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 $i \in \{1, \dots, n\}$. For a program $\mathcal{P}$, $T_\mathcal{P}(D) = \bigcup_{r\in\mathcal{P}} T_r(D)$.

Graphs We consider real-valued vectors and matrices. 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 j of $\mathbf{A}$. A function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is *monotonically increasing* if $x < y$ implies $\sigma(x) \leq \sigma(y)$. We apply functions to vectors element-wise.

For a finite set Col of colours and $\delta \in \mathbb{N}$, a (Col, δ)-graph G is a tuple $\langle V, \{E^c\}_{c\in\text{Col}}, \lambda\rangle$ where V is a finite vertex set, each $E^c \subseteq V \times V$ is a set of directed edges, and λ assigns to each $v \in V$ a vector of dimension δ. When λ is clear from the context, we abbreviate the labelling $\lambda(v)$ as $\mathbf{v}$. 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 $i \in \{1, ..., \delta\}$.

Graph Neural Networks We consider GNNs with sum aggregation. A (Col,δ)-sum graph neural network (sum-GNN) $\mathcal{N}$ with $L \geq 1$ layers is a tuple

$$\langle\, \{\mathbf{A}_\ell\}_{1\leq\ell\leq L},\ \{\mathbf{B}^c_\ell\}_{c\in\text{Col},1\leq\ell\leq L},\ \{\mathbf{b}_\ell\}_{1\leq\ell\leq L},\ \sigma_\ell,\ \texttt{cls}_t\,\rangle, \tag{2}$$

where, for each $\ell \in \{1, \dots, L\}$ and $c \in$ Col, matrices $\mathbf{A}_\ell$ and $\mathbf{B}^c_\ell$ are of dimension $\delta_\ell \times \delta_{\ell-1}$ with $\delta_0 = \delta_L = \delta$, $\mathbf{b}_\ell$ is a vector of dimension δ_ℓ, $\sigma_\ell : \mathbb{R} \rightarrow \mathbb{R}^+ \cup \{0\}$ is a monotonically increasing activation function with non-negative range, and $\texttt{cls}_t : \mathbb{R} \rightarrow \{0, 1\}$ for threshold $t \in \mathbb{R}$ is a step classification function such that $\texttt{cls}_t(x) = 1$ if $x \geq t$ and $\texttt{cls}_t(x) = 0$ otherwise.

Applying $\mathcal{N}$ to a (Col, δ)-graph induces a sequence of labels $\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_L$ for each vertex v in the graph as follows. First, $\mathbf{v}_0$ is the initial labelling of the input graph; then, for each $1 \leq \ell \leq L$, $\mathbf{v}_\ell$ is defined by the following expression:

$$\mathbf{v}_\ell = \sigma_\ell(\mathbf{b}_\ell + \mathbf{A}_\ell\mathbf{v}_{\ell-1} + \sum_{c\in\text{Col}} \mathbf{B}^c_\ell \sum_{(v,u)\in E^c} \mathbf{u}_{\ell-1}) \tag{3}$$

The output of $\mathcal{N}$ is a (Col,δ)-graph with the same vertices and edges as the input graph, but where each vertex is la-

belled by $\texttt{cls}_t(\mathbf{v}_L)$. For layer $\ell \in \{0, ..., L\}$ of $\mathcal{N}$, each $i \in \{1, ..., \delta_\ell\}$ is referred to as a *channel*.

The R-GCN model (Schlichtkrull et al. 2018) can be seen as a sum-GNN variant with ReLU activations and zero biases in all layers. The definition in (Schlichtkrull et al. 2018) includes a normalisation 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). Sum-GNNs capture the R-GCN model under the assumption that the normalisation parameters are data-independent and hence can be fixed after training; we further assume for simplicity that they are set to 1.

Dataset Transformations Through sum-GNNs A sum-GNN $\mathcal{N}$ can be used to realise a transformation $T_\mathcal{N}$ from datasets to datasets over a given finite signature (Tena Cucala et al. 2023). To this end, the input dataset must be first encoded into a graph that can be directly processed by the sum-GNN, and the graph resulting from the sum-GNN application must be subsequently decoded back into an output dataset. Several encoding/decoding schemes have been proposed in the literature. We adopt the so-called *canonical scheme*, which is a straightforward way of converting datasets to coloured graphs. In particular, colours in graphs correspond to binary predicates in the signature and channels of feature vectors in the input and output layers of the sum-GNN to unary predicates. For each $p \in \{1, \dots, \delta\}$, we denote the unary predicate corresponding to channel 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 $p \in \{1, \dots, \delta\}$. The decoder $\texttt{dec}$ is the inverse of the encoder. The *canonical* dataset transformation induced by a sum-GNN $\mathcal{N}$ is then defined as: $T_\mathcal{N}(D) = \texttt{dec}(\mathcal{N}(\texttt{enc}(D)))$. We abbreviate $\mathcal{N}(\texttt{enc}(D))$ by $\mathcal{N}(D)$.

Soundness and Completeness A Datalog program or rule α is sound for a sum-GNN $\mathcal{N}$ if $T_\alpha(D) \subseteq T_\mathcal{N}(D)$ for each dataset D. Conversely, α is complete for $\mathcal{N}$ if $T_\mathcal{N}(D) \subseteq T_\alpha(D)$ for each dataset D. Finally, we say that α is equivalent to $\mathcal{N}$ if it is both sound and complete for $\mathcal{N}$. The following proposition establishes that the containment relation that defines soundness for a rule or program still holds when the operators are composed a finite number of times.

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

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 α involving a binary predicate and constants from D, the task is to predict whether $\alpha \in D^*$. To perform link prediction with a sum-GNN $\mathcal{N}$, we use the (non-canonical) encoding and decoding from (Tena Cucala et al. 2021), 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 channels of the input and output layers. This allows us to derive new binary facts (which the canonical transformation cannot do). As shown in Section 3.2 of (Tena Cucala et al. 2023), 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 $T_\mathcal{N}$ to the end-to-end transformation $\mathcal{P}_\texttt{dec}(T_\mathcal{N}(\mathcal{P}_\texttt{enc}(D)))$.

3 Partitioning the Channels of a GNN

In this section, we first provide two approaches for identifying channels in a sum-GNN that exhibit monotonic behaviour, which will later allow us 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. 2023) for monotonic GNNs. Furthermore, we provide an approach for identifying channels for which no sound Datalog rule exists. Our techniques are data-independent and rely on direct analysis of the dependencies between feature vector components via the parameters of the model.

3.1 Safe Channels

In this section, we introduce the notion of a safe channel of a sum-GNN $\mathcal{N}$. 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 $\mathcal{N}$ to D is affected only by non-negative values in the weight matrices of the GNN.

For instance, consider a simple sum-GNN 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 $\mathbf{B}^c_\ell$ 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 generalises this intuition.

Definition 2. *Let $\mathcal{N}$ be a sum-GNN as in Equation* (2)*. 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 the i-th row of each matrix $\mathbf{A}_\ell$ and $\{(\mathbf{B}^c_\ell)\}_{c \in Col}$ 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 $\ell - 1$. Otherwise, i is* unsafe*.*

We can now show that safe channels in a GNN 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.

Lemma 3. *Let $\mathcal{N}$ be a sum-GNN as in Equation* (2)*, 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 $\mathcal{N}$ to D and D' respectively. If channel i in $\mathcal{N}$ 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 (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 2, 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 $(\mathbf{B}^c_\ell \sum_{(v,u)\in (E^c)'} \mathbf{u}'_{\ell-1})[i]$ 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

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

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.

□

We conclude this section by showing that the identification of safe channels allows for the extraction of sound rules from a trained sum-GNN model. In particular, given a candidate Datalog rule, it is possible to algorithmically verify its soundness using the approach of (Tena Cucala et al. 2023). The soundness check for a rule r of the form (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^ν_r, the dataset consisting of each fact $B_i\nu$ such that B_i is a body atom in r, we check whether the GNN 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.

Proposition 4. *Let $\mathcal{N}$ be a sum-GNN as in Equation* (2)*, and let r be a rule of the form* (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 $\mathcal{N}$ is safe at layer L. Then r is sound for $T_\mathcal{N}$ if and only if $H\nu \in T_\mathcal{N}(D^\nu_r)$ 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 $T_r(D) \subseteq T_\mathcal{N}(D)$. To this end, we consider an arbitrary fact in $T_r(D)$ and show that it is also contained in $T_\mathcal{N}(D)$. 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 $D^\nu_r \models B_i\nu$ because $D \models B_i\mu$ and σ is injective. Therefore, by the assumption of the proposition, $H\nu \in T_\mathcal{N}(D^\nu_r)$. Now, observe that the result of applying $T_\mathcal{N}$ to a dataset does not depend on the identity of the constants, but only on the structure of the dataset; therefore, $T_\mathcal{N}$ is invariant under one-to-one mappings of constants, and since σ is one such map, $H\nu \in T_\mathcal{N}(D^\nu_r)$ implies $H\mu \in T_\mathcal{N}(D^\mu_r)$. Now, let a be the single constant in $H\mu$. Since $D^r_\mu \subseteq D$ by definition of μ, and channel p is safe at layer L, we can apply Lemma 3 to conclude that $\mathbf{v}'_L[p] \leq \mathbf{v}_L[p]$, for v the vertex corresponding to a in $\mathtt{enc}(D^\mu_r)$, and $\mathbf{v}'_L$ and $\mathbf{v}_L$ the feature vectors in layer L computed for v by $\mathcal{N}$ on D^μ_r and D, respectively. But $H\mu \in T_\mathcal{N}(D^\mu_r)$ 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 $H\mu \in T_\mathcal{N}(D)$, as we wanted to show.

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

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, we provide a more involved analysis of the dependencies between channels and model parameters which yields a more fine-grained channel classification. In particular, we identify two new classes of channels that behave monotonically, such that their union contains all safe channels, but may also contain unsafe channels.

3.2 Stable and Increasing Channels

In this section, we provide a classification of the channels of the GNN depending on their behaviour under updates involving the addition of new facts to an input dataset. In-

tuitively, *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 categorised as *undetermined*. Consider again a sum-GNN where matrix $\mathbf{A}_\ell$ for layer ℓ is given below and each of the matrices $\mathbf{B}^c_\ell$ 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 formalises these intuitions and extends the analysis to the matrices $\mathbf{B}^c_\ell$ involved in neighbourhood aggregation.

Definition 5. *Let $\mathcal{N}$ be a sum-GNN as in Equation* (2)*. All channels are* increasing *at layer 0, A channel $i \in \{1,\dots,\delta_\ell\}$ is* stable *at layer $\ell \in \{1,...,L\}$ if both of the following conditions hold for each $j \in \{1,\dots,\delta_{\ell-1}\}$:*

- $\mathbf{B}^c_\ell[i,j] = 0$ *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 $j \in \{1,\dots,\delta_{\ell-1}\}$, these conditions hold:*

1. *if j is increasing in $\ell-1$, then $\mathbf{A}_\ell[i,j] \geq 0$ (resp. $\mathbf{A}_\ell[i,j] \leq 0$);*
2. *if j is decreasing in $\ell-1$, 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 in $\ell-1$, 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.

Lemma 6. *Let $\mathcal{N}$ be a sum-GNN of L layers, and let D', D be datasets satisfying $D' \subseteq D$. For each vertex $v \in \texttt{enc}(D')$, layer $\ell \in \{0,\dots,L\}$, and channel $i \in \{1,\dots,\delta_\ell\}$, 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 $\mathcal{N}$ to D and D' respectively.

Proof sketch. The full proof is given in **??**.

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 $i \in \{1,\dots,\delta\}$ by definition of $\texttt{enc}$, 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 $u \in \texttt{enc}(D')$. 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 4 extends seamlessly to this new setting.

Proposition 7. *Let $\mathcal{N}$ be a sum-GNN as in Equation* (2)*, and let r be a rule of the form* (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 $\mathcal{N}$ is increasing or stable at layer L. Then r is sound for $T_\mathcal{N}$ if and only if $H\nu \in T_\mathcal{N}(D^\nu_r)$ 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.*

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. We conclude this section by relating the classes of channels described here to those of Section 3.1, with a full proof given in **??**.

Theorem 8. *Safe channels are increasing or stable. There exist increasing unsafe channels and stable unsafe channels.*

3.3 Unbounded Channels

In Section 3.2, we noted that extracting sound Datalog rules for channels that are decreasing or undetermined might be possible. In this section, we identify a subset of such channels, which we 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-GNN uses ReLU as the activation function in all but (possibly) the last layer L. Furthermore, we 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 $\mathtt{cls}_t$. This restriction is non-essential and simply excludes GNNs that derive all possible facts regardless of the input dataset, in which case all rules would be sound.

Definition 9. *Channel $p \in \{1, \dots, \delta_L\}$ is* unbounded *at layer L if there exist a Boolean vector $\mathbf{y}_0$ of dimension δ_0, and a sequence $c_1, \dots, c_L$ of (not necessarily distinct) colours in Col, such that, with $\{\mathbf{y}_\ell\}_{\ell=1}^{L-1}$ the sequence defined inductively as $\mathbf{y}_\ell := \mathit{ReLU}(\mathbf{B}_\ell^{c_\ell}\mathbf{y}_{\ell-1})$ 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 $T_{\mathcal{N}}$. 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, $U_p(a) \notin T_{\mathcal{N}}(D')$, 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 **??**.

Theorem 10. *Let $\mathcal{N}$ be a sum-GNN as in Equation* (2)*, 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 $\mathcal{N}$.*

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 $c_1, \dots, c_L$ be a Boolean vector and a sequence of colours, respectively, satisfying the condition in Definition 9 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 $R^{c_L}(b_{L-1}, a)$;
2. facts $U_k(a_j)$ for $1 \leq j \leq d$ and $k \leq \delta$ s.t. $\mathbf{y}_0[k] = 1$;

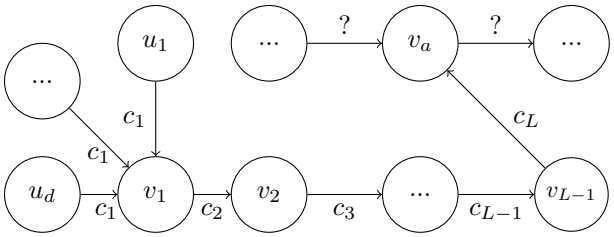

Figure 1: Canonical encoding G_d of the dataset D_d that extends D_r^ν in the proof of Theorem 10 for a given $d \in \mathbb{N}$.

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 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 $\mathcal{N}$ 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 (3) for vertex $v = v_\ell$ in layer ℓ when applying $\mathcal{N}$ 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 (3) that computes the value of vertex v_a in layer L when applying $\mathcal{N}$ 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 $\mathcal{N}$. Indeed, when applying $\mathcal{N}$ 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 $U_p(a) \notin T_{\mathcal{N}}(D_{d^*})$. 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 $\mathcal{N}$. □

In order to check which channels are unbounded in practice, we initialise an empty set S of unbounded channels. We 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, we compute the vector $\mathbf{B}_L^{c_L}\mathbf{y}_{L-1}$ as shown in Definition 9 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 $|\mathrm{Col}|$ and layers L are large enough that it is not practically feasible to enumerate all combinations. In those cases, we 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, we relate unbounded channels to those defined in Section 3.1 and Section 3.2. A full proof is given in **??**.

Theorem 11. *Unbounded channels are neither increasing, nor stable, nor safe. There exist, however, decreasing unbounded channels and undetermined unbounded channels.*

4 Experiments

We train sum-GNNs on several link prediction datasets, using the dataset transformation described in Section 2. For each dataset, we train a sum-GNN with ReLU activation functions, biases, and two layers (this architecture corresponds to R-GCN with biases). The hidden layer of the GNN has twice as many channels as its input. Moreover, we also train four additional instances of sum-GNNs, using a modified training paradigm to facilitate the learning of sound rules (see Section 4.2). For each trained model, we compute standard classification metrics, such as precision, recall, accuracy, and F1 score, and area under the precision-recall curve (AUPRC).

We train all our models using binary cross entropy loss for training and the Adam optimizer with a standard learning rate of 0.001. For each model, we 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 maximises accuracy. We run each experiment across 10 different random seeds and present the aggregated metrics. Experiments are run using PyTorch Geometric, with a CPU on a Linux server.

Channel Classification and Rule Extraction For each trained model, we 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, we 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 7 to check soundness. In benchmarks with a large number of predicates, we only check rules with one body atom, since searching the space of rules with two body atoms is intractable. For datasets created with LogInfer (Liu et al. 2023), which are obtained by enriching a pre-existing dataset with the consequences of a known set of Datalog rules, we also check if these rules were sound for the model.

4.1 Datasets

We use three benchmarks provided by (Teru, Denis, and Hamilton 2020): WN18RRv1, FB237v1, and NELLv1; each of these benchmarks provides datasets for training, validation, and testing, as well as negative examples.

We also use LogInfer (Liu et al. 2023), 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. We apply the LogInfer framework to datasets FB15K-237 (Toutanova and Chen 2015) and WN18RR (Dettmers et al. 2018). We consider the very simple rule patterns *hierarchy* and *symmetry* defined in (Liu et al. 2023); we 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 1. For each pattern P, we refer to the datasets obtained by enriching FB15K-237 and WN18RR by FB-P and WN-P, respectively. For each dataset and pattern, we randomly select 10% of the enriched training dataset to be used as targets and the rest as inputs to the model. Furthermore, we consider an additional instance of FB15K-237 and WN18RR where we again apply the hierarchy pattern, but we include a much larger number of consequences in the enriched training dataset; we refer to these datasets as FB-superhier and WN-superhier, as a shorthand for *super-hierarchy*. The purpose of this is to create a dataset where it should be as easy as possible for a model to learn the rules applied in the dataset's creation. Finally, we use LogInfer to augment FB15K-237 and WN18RR using multiple rule patterns at the same time; in particular, we combine monotonic with non-monotonic rule patterns. This allows us to test the ability of our 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 WN is extended using the hierarchy and non-monotonic hierarchy rule patterns. When creating a mixed dataset, we 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 $P(a, b) \in D$, we 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. We avoid using constant corruption to produce negative examples because this would inflate the performance of our models. The reason for this is that the encoding by (Tena Cucala et al. 2021) that we use in our transformation can only predict binary

	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	$R(x,y) \wedge \neg S(y,z) \rightarrow T(x,y)$

Table 1: LogInfer inference patterns used in this paper.

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, we sample $T(a,b)$ such that $\{R(a,b), S(b,c)\} \subseteq D$, to penalise the model for learning the rule pattern $R(x,y) \rightarrow T(x,y)$ instead of the non-monotonic one. We train for 8000 epochs on WN18RRv1, FB237v1, NELLv1, and the LogInfer-WN datasets, and for 1500 epochs on the LogInfer-FB datasets.

4.2 Training Paradigm Variations

We use the term "MGCN" to refer to R-GCN, but with the negative weights clamped to zero during training, after every optimizer step. In this way, every channel is trivially safe and rules can be checked for soundness. This is the same training method used by (Tena Cucala et al. 2021), but on a different GNN. We also use early-stopping in our normal training of R-GCNs and MGCNs variation, ceasing training if the model loss deteriorates for 50 epochs in a row. MGCNs are purely monotonic and are thus unable to learn non-monotonic patterns in the data.

To encourage R-GCNs to exhibit non-monotonic behaviour while simultaneously learning sound monotonic rules, we propose a novel variation to the training routine. Our 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 we clamp those before larger ones. To clamp a sum-GNN $\mathcal{N}$ using a *clamping threshold* $\tau \in \mathbb{R}^+$, we 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 $\mathcal{N}$ such that the percentage of output channels of $\mathcal{N}$ that are stable or increasing is strictly greater than X. We 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, we compute τ_X and clamp the model's matrices using τ_X as a clamping threshold. We refer to the model trained for X as R-X. We apply these models 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.

4.3 Results

Monotonic LogInfer Datasets The results on the monotonic LogInfer benchmarks are shown in Table 2. We note that, across every experiment, R-GCN provably never has any sound rules, since every channel is found to be unbounded. These results are very surprising, since our monotonic rule patterns 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 proves 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 (in this case) does not correspond to sound rules.

In Table 2, 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. We 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 our experiments on the benchmark datasets are shown in Table 3. 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 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. Our dataset transformation, 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. 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.

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 4 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 provably 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 penalised by false positives, since it is predicting facts that are not derived by the non-monotonic rules. We 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.

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 2: 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 GNN and %NG is the percentage of monotonic LogInfer rules that are not sound for the GNN 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 3: 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.

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 4: 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 GNN, %NG is the percentage of rules that are not sound for the GNN 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.

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 our theoretical analysis allows for the existence of undetermined or decreasing channels that are not unbounded, virtually no such channels are found when using our proposed training paradigms, and so our proposed channel classification allows us 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, our 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 big, this validates the need for the more complex classification of channels into stable/increasing/decreasing/undetermined, instead of the simpler classification into safe/unsafe.

5 Conclusion

In this paper, we provided two ways to extract sound rules from sum-GNNs and a procedure to prove that certain rules are not sound for a sum-GNN. Our methods rely on classifying the output channels of the sum-GNN. We found that, in our experiments, R-GCN, a specific instance of sum-GNN, provably has no sound rules when trained in the standard way, even when using ideal datasets. We provided two alternatives to train R-GCN, the first of which clamps all negative weights to zero and results in R-GCN being entirely monotonic, 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. Our second alternative yields a trade-off between accuracy and the number of sound rules. We found that, in practice, almost every channel of the sum-GNN is classified in a manner that allows us to either extract sound rules or prove that there are no associated sound rules.

The limitations of this work are as follows. First, our analysis only considers GNNs with sum aggregation and monotonically increasing activation functions: it is unclear how sound rules can be extracted from models that use mean aggregation or GELU activation functions. Furthermore, our methods do not guarantee that every output channel will be characterised such that we can show that it either has no sound rules or it can be checked for sound rules.

For future work, we aim to consider other GNN architectures, extend our rule extraction to non-monotonic logics, provide relaxed definitions of soundness, and consider R-GCN with a scoring function (such as DistMult) as a decoder, instead of using a dataset transformation to perform link prediction.

Acknowledgments

Matthew Morris is funded by an EPSRC scholarship (CS2122_EPSRC_1339049). This work was also supported by the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project 237889), Samsung Research UK, the EPSRC projects AnaLOG (EP/P025943/1), OASIS (EP/S032347/1), UKFIRES (EP/S019111/1) and ConCur (EP/V050869/1). The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work http://dx.doi.org/10.5281/zenodo.22558.

References

Abboud, R.; Ceylan, İ. İ.; Lukasiewicz, T.; and Salvatori, T. 2020. Boxe: A box embedding model for knowledge base completion. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26.

Bouchard, G.; Singh, S.; and Trouillon, T. 2015. On approximate reasoning capabilities of low-rank vector spaces. In *2015 AAAI Spring Symposium Series*.

Cai, L.; Yan, B.; Mai, G.; Janowicz, K.; and Zhu, R. 2019. Transgcn: Coupling transformation assumptions with graph convolutional networks for link prediction. In *Proceedings of the 10th international conference on knowledge capture*, 131–138.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Evans, R., and Grefenstette, E. 2018. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research* 61:1–64.

Garnelo, M., and Shanahan, M. 2019. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences* 29:17–23.

Gutteridge, B.; Dong, X.; Bronstein, M. M.; and Di Giovanni, F. 2023. Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, 12252–12267. PMLR.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30.

Hogan, A.; Blomqvist, E.; Cochez, M.; d'Amato, C.; de Melo, G.; Gutierrez, C.; Kirrane, S.; Gayo, J. E. L.; Navigli, R.; Neumaier, S.; Ngomo, A. N.; Polleres, A.; Rashid, S. M.; Rula, A.; Schmelzeisen, L.; Sequeda, J. F.; Staab, S.; and Zimmermann, A. 2022. Knowledge graphs. *ACM Comput. Surv.* 54(4):71:1–71:37.

Ioannidis, V. N.; Marques, A. G.; and Giannakis, G. B. 2019. A recurrent graph neural network for multi-relational

data. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8157–8161. IEEE.

Li, W.; Zhu, L.; Mao, R.; and Cambria, E. 2023. Skier: A symbolic knowledge integrated model for conversational emotion recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 13121–13129.

Lin, Q.; Mao, R.; Liu, J.; Xu, F.; and Cambria, E. 2023. Fusing topology contexts and logical rules in language models for knowledge graph completion. *Information Fusion* 90:253–264.

Liu, S.; Grau, B.; Horrocks, I.; and Kostylev, E. 2021. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. *Advances in Neural Information Processing Systems* 34:2034–2045.

Liu, S.; Grau, B. C.; Horrocks, I.; and Kostylev, E. V. 2023. Revisiting inferential benchmarks for knowledge graph completion. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, 461–471.

Meilicke, C.; Fink, M.; Wang, Y.; Ruffinelli, D.; Gemulla, R.; and Stuckenschmidt, H. 2018. Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In *The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part I 17*, 3–20. Springer.

Nickel, M.; Tresp, V.; Kriegel, H.-P.; et al. 2011. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, 3104482–3104584.

Pflueger, M.; Tena Cucala, D. J.; and Kostylev, E. V. 2022. Gnnq: A neuro-symbolic approach to query answering over incomplete knowledge graphs. In *International Semantic Web Conference*, 481–497. Springer.

Qu, M.; Chen, J.; Xhonneux, L.-P.; Bengio, Y.; and Tang, J. 2020. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. In *International Conference on Learning Representations*.

Rocktäschel, T., and Riedel, S. 2017. End-to-end differentiable proving. *Advances in neural information processing systems* 30.

Sadeghian, A.; Armandpour, M.; Ding, P.; and Wang, D. Z. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems* 32.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, 593–607. Springer.

Shang, C.; Tang, Y.; Huang, J.; Bi, J.; He, X.; and Zhou, B. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 3060–3067.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, 697–706.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2018. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Tang, J.; Hua, F.; Gao, Z.; Zhao, P.; and Li, J. 2024. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. *Advances in Neural Information Processing Systems* 36.

Tena Cucala, D.; Cuenca Grau, B.; Kostylev, E. V.; and Motik, B. 2021. Explainable gnn-based models over knowledge graphs. In *International Conference on Learning Representations*.

Tena Cucala, D.; Cuenca Grau, B.; Motik, B.; and Kostylev, E. V. 2023. On the Correspondence Between Monotonic Max-Sum GNNs and Datalog. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning*, 658–667.

Tena Cucala, D. J.; Cuenca Grau, B.; and Motik, B. 2022. Faithful approaches to rule learning. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, 484–493.

Teru, K.; Denis, E.; and Hamilton, W. 2020. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, 9448–9457. PMLR.

Tian, A.; Zhang, C.; Rang, M.; Yang, X.; and Zhan, Z. 2020. Ra-gcn: Relational aggregation graph convolutional network for knowledge graph completion. In *Proceedings of the 2020 12th international conference on machine learning and computing*, 580–586.

Toutanova, K., and Chen, D. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, 57–66.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. 2019. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

Vrandečić, D., and Krötzsch, M. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10):78–85.

Wang, X.; Tena Cucala, D.; Cuenca Grau, B.; and Horrocks, I. 2023. Faithful rule extraction for differentiable rule learning models. In *The Twelfth International Conference on Learning Representations*.

Xie, X.; Zhang, N.; Li, Z.; Deng, S.; Chen, H.; Xiong, F.; Chen, M.; and Chen, H. 2022. From discrimination to generation: Knowledge graph completion with generative transformer. In *Companion Proceedings of the Web Conference 2022*, 162–165.

Yang, B.; Yih, S. W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems* 30.

Yao, L.; Mao, C.; and Luo, Y. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Yu, D.; Yang, Y.; Zhang, R.; and Wu, Y. 2021. Knowledge embedding based graph convolutional network. In *Proceedings of the Web Conference 2021*, 1619–1628.

Zhang, M.; Xia, Y.; Liu, Q.; Wu, S.; and Wang, L. 2023. 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)*, 12617–12631.

For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.