Bridging Max Graph Neural Networks and Datalog with Negation

David J. Tena Cucala[1,2] , **Bernardo Cuenca Grau**[1]
[1]University of Oxford
[2]Royal Holloway, University of London
david.tenacucala@rhul.ac.uk, bernardo.cuenca.grau@cs.ox.ac.uk

Abstract

We consider a general class of data transformations based on Graph Neural Networks (GNNs), which can be used for a wide variety of tasks. An important question in this setting is characterising the expressive power of these transformations in terms of a suitable logic-based language. From a practical perspective, the correspondence of a GNN with a logical theory can be exploited for explaining the model's predictions symbolically. In this paper, we introduce a broad family of GNN-based transformations which can be characterised using Datalog programs with negation-as-failure, which can be computed from the GNNs after training. This generalises existing approaches based on positive programs by enabling the learning of nonmonotonic transformations. We show empirically that these GNNs offer good performance for knowledge graph completion tasks, and that we can efficiently extract programs for explaining individual predictions.

1 Introduction

Numerous tasks over knowledge graphs (KGs) such as KG completion (Lin et al. 2015), node classification (Portisch and Paulheim 2022) and recommendation (Wang et al. 2019) can be conceptualised as transformations of datasets consisting of logical facts. For example, in a recommender system, we can represent user–item interactions and background knowledge in a KG as a set of facts, and the system can then transform this dataset into another dataset containing the recommended user–item interactions.

Graph Neural Networks (GNNs)—models specifically developed for processing graph data (Wu et al. 2023)—are increasingly being used to learn these transformations (Schlichtkrull et al. 2018; Pflueger, Tena Cucala, and Kostylev 2022; Pflueger, Tena Cucala, and Kostylev 2024; Liu et al. 2021; Ioannidis, Marques, and Giannakis 2019; Qu, Bengio, and Tang 2019; Yang, Cohen, and Salakhutdinov 2016; Kipf and Welling 2017; Zhang and Chen 2018; Teru, Denis, and Hamilton 2020). The key advantage of GNNs is that their output is resilient to the renaming of vertices (i.e., applying a GNN to isomorphic graphs produces the same output). Dataset transformations are often expected to exhibit analogous properties; for example, we would expect distinct users interacting with sets of items in exactly the same way to receive the same recommendations.

An important question in such applications is understanding the expressive power of the underpinning GNNs by identifying a logic-based language that can express the same class of dataset transformations. Such correspondences allow for a rigorous understanding and comparison of different families of GNN-based models. Furthermore, they facilitate practical applications such as the formal verification of model properties and the explanation of their predictions. For instance, a property of interest can be expressed as a logical formula, and logical methods can then be used to determine if it is a consequence of the logical theory characterising the model. Additionally, any output fact of the ML-based transformation can be explained by using a symbolic engine to generate a proof showing how the fact logically follows from the theory describing the transformation. These rigorous, yet human-readable explanations can help users understand why the model derived a given output fact and check, for example, whether a given input fact was relevant to its derivation. The ability to explain symbolically each prediction can also be used to ensure that a model complies with algorithmic explainability policies (Tzimas 2023).

In a pioneering study, Barceló et al. (2020) showed that each GNN-induced transformation expressible in first-order logic is equivalent to a concept query of the $\mathcal{ALCQ}$ description logic (Baader et al. 2007). Huang et al. (2023) proved an analogous result for a slightly different class of GNNs, and Grohe (2023) generalised these results by showing that GNN-induced transformations can, under mild restrictions, be captured by the guarded fragment of FOL with counting terms GFO+C. Finally, Benedikt et al. (2024) have shown that transformations realised by GNNs with bounded piecewise linear activation functions correspond to formulas expressed in Presburguer logic, a language that is decidable but not comparable with first-order logic.

In our previous work (Tena Cucala et al. 2022; Tena Cucala et al. 2023), we introduced a family of GNN-based transformations which can be characterised using tree-like Datalog programs and we presented practical algorithms for extracting an equivalent program from any such trained GNN. This enables the application of Datalog engines to obtain a step-by-step logical proof of each fact predicted by the GNN. These transformations were carefully crafted to ensure that they can be described symbolically using rules and, in particular, that they are monotonic under homomorphisms. As a result, GNNs in these transformations are restricted to use max aggregation functions in each layer, matrix weights are required to be nonnegative, and the activation and classification functions must satisfy certain restrictions, such as being monotonically increasing.

In this paper, we lift some of the restrictions in our prior work, such as the requirement that matrix weights need to be nonnegative and the restrictions on activation and classification functions. This leads to a significant increase in expressive power, enabling the learning of nonmonotonic transformations. Our main result is that the transformations induced by these GNNs can be characterised by tree-like Datalog programs with negation-as-failure (Dantsin et al. 2001), which can be computed from the GNN. These programs, however, can be large and difficult to compute. Thus, we propose a practical algorithm for explaining individual predictions, which reduces the search space of relevant rules. Our transformations are "pay-as-you-go": if the classification and activation functions are monotonic, and all matrix weights are nonnegative, we obtain a program from which negation-as-failure can be eliminated, thus obtaining a (positive) Datalog program as in (Tena Cucala et al. 2022). We demonstrate the effectiveness of our transformations by applying them to knowledge graph completion tasks. Our experiments show that our approach delivers competitive performance while at the same time enabling rule extraction for the explanation of individual predictions.

2 Preliminaries

Datalog with Negation We fix a signature consisting of countably infinite, disjoint sets of unary predicates, binary predicates, and constants. 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. A unary atom is an expression of the form $P(t)$, with P a unary predicate and t a term, and a binary atom is of the form $P(t_1, t_2)$ with P a binary predicate and t_1, t_2 terms. A term or an atom is ground if it is variable-free. A fact is a ground atom and a dataset is a finite set of facts. We consider rules of the form (1), with $n, m \geq 0$, where H is an atom called the head of the rule, and each P_i (resp. Q_i) is a unary or binary atom called a positive (resp. negated) body atom.

$$P_1 \wedge \cdots \wedge P_n \wedge \mathsf{not}\ Q_1 \wedge \cdots \wedge \mathsf{not}\ Q_m \rightarrow H \qquad (1)$$

A (Datalog$^\neg$) program is a finite set of rules.

A substitution μ is a mapping of finitely many variables to constants. For α a variable, atom, or conjunction thereof, $\alpha\mu$ is the result of replacing in α each variable x in the domain of μ with $\mu(x)$. For a rule r of the form (1), the operator T_r maps a dataset D to the dataset that contains $H\mu$ for each substitution μ mapping the variables of r to constants in D such that $P_i\mu \in D$ for each $1 \leq i \leq n$, and $Q_i\mu \notin D$ for each $1 \leq i \leq m$. For $\mathcal{P}$ a program, $T_\mathcal{P}(D) = \bigcup_{r \in \mathcal{P}} T_r(D)$. Furthermore, for a nonnegative integer n, $T^n_\mathcal{P}(D)$ is defined inductively as follows: $T^0_\mathcal{P}(D) = D$, and $T^n_\mathcal{P}(D) = T_\mathcal{P}(T^{n-1}_\mathcal{P}(D))$, for $n \geq 1$.

A homomorphism from dataset D to dataset D' is a mapping h of constants to constants defined at least on the set of constants of D and satisfying $h(D) \subseteq D'$, where $h(D)$ is the dataset obtained by replacing each constant a in D with $h(a)$ and removing duplicate facts. If T is an operator mapping datasets to datasets over the same constants, then T is monotonic under homomorphisms if, for each D and D' in its domain, each homomorphism from D to D' is also a homomorphism from $T(D)$ to $T(D')$.

Graph Neural Networks We consider real-valued vectors and matrices. For $\mathbf{v}$ a vector and $i \geq 1$, $(\mathbf{v})_i$ is the i-th element of $\mathbf{v}$. Similarly, for $\mathbf{A}$ a matrix and $i, j \geq 1$, $(\mathbf{A})_{i,j}$ is the element in row i, column j of $\mathbf{A}$. A vector is null if all its elements are 0. We apply scalar functions to vectors element-wise; for example, for $\mathbf{v}_1, \ldots, \mathbf{v}_n$ vectors of equal dimension, $\max\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ is the vector whose i-th element is $\max\{(\mathbf{v}_1)_i, \ldots, (\mathbf{v}_n)_i\}$.

For Col a finite nonempty set of colours and $\delta \in \mathbb{N}$ a dimension, a (Col, δ)-graph is a tuple $\mathcal{G} = \langle \mathcal{V}, \{\mathcal{E}^c\}_{c \in \mathsf{Col}}, \lambda \rangle$ where $\mathcal{V}$ is a finite set of vertices; $\mathcal{E}^c \subseteq \mathcal{V} \times \mathcal{V}$ is a set of (directed) edges for each colour $c \in \mathsf{Col}$; and labelling λ assigns to each $v \in \mathcal{V}$ a feature vector $\mathbf{v}$ of dimension δ. Graph $\mathcal{G}$ is Boolean if $(\mathbf{v})_i \in \{0, 1\}$ for each $v \in \mathcal{V}$ and $1 \leq i \leq \delta$. A vertex u is a c-neighbour of a vertex v in a (Col, δ)-graph if $\langle v, u \rangle \in \mathcal{E}^c$. We consider a class of GNNs that use max as aggregation function. A (Col, δ)-max graph neural network (*GNN*) $\mathcal{N}$ with $L \geq 1$ layers is a tuple

$$\langle\ \{\mathbf{A}_\ell\}_{1 \leq \ell \leq L},\ \{\mathbf{B}^c_\ell\}^{c \in \mathsf{Col}}_{1 \leq \ell \leq L},\ \{\mathbf{b}_\ell\}_{1 \leq \ell \leq L},\ \sigma,\ \mathsf{cls}\ \rangle, \qquad (2)$$

where, for each $1 \leq \ell \leq L$ and $c \in \mathsf{Col}$, $\mathbf{A}_\ell$ and $\mathbf{B}^c_\ell$ are matrices of dimension $\delta_\ell \times \delta_{\ell-1}$ with $\delta_0 = \delta_L = \delta$, $\mathbf{b}_\ell$ is a vector over $\mathbb{R}$ of dimension δ_ℓ, $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is an activation function, and $\mathsf{cls} : \mathbb{R} \rightarrow \{0, 1\}$ is a classification function.

Applying (Col, δ)-GNN $\mathcal{N}$ to (Col, δ)-graph $\mathcal{G}$ induces the sequence $\lambda_0, \ldots, \lambda_L$ of vertex labelling functions such that $\lambda_0 = \lambda$ and, for each $1 \leq \ell \leq L$, λ_ℓ assigns to each $v \in \mathcal{V}$ the vector $\mathbf{v}_\ell$ given by

$$\sigma\Big(\mathbf{A}_\ell \mathbf{v}_{\ell-1} + \sum_{c \in \mathsf{Col}} \mathbf{B}^c_\ell \max\{\mathbf{w}_{\ell-1} \mid \langle v, w \rangle \in \mathcal{E}^c\} + \mathbf{b}_\ell\Big), \qquad (3)$$

where $\max\{\}$ is the null vector of dimension δ_ℓ. The result $\mathcal{N}(\mathcal{G})$ of applying $\mathcal{N}$ to $\mathcal{G}$ is the Boolean (Col, δ)-graph with the same vertices and edges as $\mathcal{G}$, but where each vertex $v \in \mathcal{V}$ is labelled by $\mathsf{cls}(\mathbf{v}_L)$.

GNN-based Transformations To apply a max GNN to a dataset, we must first encode the dataset into a suitable graph, and then decode the GNN's output back into a dataset. The canonical scheme (Tena Cucala et al. 2023) provides a direct encoding, where constants map into vertices, edges describe binary facts, and unary facts are encoded as elements of the feature vectors labelling vertices. Let $\delta \in \mathbb{N}$ and Col a finite, nonempty set of colours. Consider a signature containing δ unary predicates $U_1, \ldots, U_\delta$ and a binary predicate E^c for each $c \in \mathsf{Col}$. A (Col, δ)-dataset is a dataset over predicates $\{U_i\}^\delta_{i=1}$ and $\{E^c\}_{c \in \mathsf{Col}}$.

The canonical encoding $\mathsf{enc}(D)$ of a (Col, δ)-dataset D is the Boolean (Col, δ)-graph $\langle \mathcal{V}, \{\mathcal{E}^c\}_{c \in \mathsf{Col}}, \lambda \rangle$ where $\mathcal{V}$ consists of vertex v^a for each constant a occurring in D, an edge $\langle v^a, v^b \rangle \in \mathcal{E}^c$ for each fact $E^c(a, b) \in D$, and where $(\mathbf{v}^a)_i = 1$ if and only if $U_i(a) \in D$. The canonical decoding $\mathsf{dec}(\mathcal{G})$ of a Boolean (Col, δ)-graph $\mathcal{G} = \langle \mathcal{V}, \{\mathcal{E}^c\}_{c \in \mathsf{Col}}, \lambda \rangle$ is the dataset that contains the fact $E^c(a, b)$ for each

$\langle v^a, v^b\rangle \in \mathcal{E}^c$ and $c \in \mathsf{Col}$, and the fact $U_i(a)$ for each $v^a \in \mathcal{V}$ and $1 \leq i \leq \delta$ such that $(\mathbf{v}^a)_i = 1$.

A max (Col, δ)-GNN $\mathcal{N}$ induces the canonical transformation $T_{\mathcal{N}}$ mapping each (Col, δ)-dataset D to the (Col, δ)-dataset $T_{\mathcal{N}}(D) = \mathsf{dec}(\mathcal{N}(\mathsf{enc}(D)))$.

3 From Max-GNNs to Rules

In this section, we show that the transformation induced by a max-GNN with L layers can be captured by a Datalog$^{\neg}$ program that can be constructed algorithmically from the GNN.

In Section 3.1 we show that, although the number of possible input graphs to the transformation is unbounded, the number of possible real values that can occur in feature vectors throughout the application of a given GNN is actually finite. Hence, thanks to the restriction to max aggregation, there is only a finite number of different vectors that can occur during GNN application. In Section 3.2 we exploit this result to formalise the construction of the equivalent Datalog$^{\neg}$ program. This program introduces auxiliary predicates to represent the possible numeric feature vectors and can be seen as the union of $L + 2$ subprograms, one for each layer of the GNN, plus two programs representing the canonical encoding and decoding, respectively. The latter two subprograms mention predicates occurring in the data, while the remaining L subprograms mention only the relevant auxiliary predicates.

For the remainder of this section, let us assume a fixed, but arbitrary max (Col, δ)-GNN $\mathcal{N}$ of the form (2) with L layers and dimensions $\delta_0, \delta_1, \ldots, \delta_L$.

3.1 Characterising Possible Feature Vectors

For each layer $0 \leq \ell \leq L$ and position $1 \leq i \leq \delta_\ell$ in $\mathcal{N}$, we introduce a set $\mathcal{X}_{\ell,i}$ containing each real number that can occur in the ith position of a feature vector computed in layer ℓ when applying $\mathcal{N}$ to an arbitrary Boolean (Col, δ)-graph. The number of input graphs to $\mathcal{N}$ is unbounded; however, the sets $\mathcal{X}_{\ell,i}$ are actually finite, as a consequence of restricting the aggregation to $\max$.

Definition 1. *For each* $1 \leq i \leq \delta_0$, $\mathcal{X}_{0,i} = \{0, 1\}$. *For each* $1 \leq \ell \leq L$ *and* $1 \leq i \leq \delta_\ell$, $\mathcal{X}_{\ell,i}$ *is the set containing* 0 *and each of the real values defined by the following expression:*

$$\mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}}) = (\sigma(\mathbf{A}_\ell \mathbf{x} + \sum_{c\in\mathsf{Col}} \mathbf{B}^c_\ell\, \mathbf{y}^c + \mathbf{b}_\ell))_i \quad (4)$$

where $\mathbf{x}$ *ranges over all vectors of dimension* $\delta_{\ell-1}$ *such that each* $(\mathbf{x})_j \in \mathcal{X}_{\ell-1,j}$, *and* $(\mathbf{y}^c)_{c\in\mathsf{Col}}$ *ranges over* $|\mathsf{Col}|$*-tuples of vectors of dimension* $\delta_{\ell-1}$ *such that each* $(\mathbf{y}^c)_j \in \mathcal{X}_{\ell-1,j}$.

The following lemma ensures that these sets carry the intended meaning.

Lemma 2. *For each* $0 \leq \ell \leq L$ *and* $1 \leq i \leq \delta_\ell$, *the set* $\mathcal{X}_{\ell,i}$ *is finite. Moreover, for each Boolean* (Col, δ)*-graph* $\mathcal{G}$ *and vertex* v *of* $\mathcal{G}$, *it holds that* $(\mathbf{v}_\ell)_i \in \mathcal{X}_{\ell,i}$, *where* $\mathbf{v}_\ell$ *is the* ℓ*th layer feature vector for* v *obtained when applying* $\mathcal{N}$ *to* $\mathcal{G}$.

Proof. We proceed by induction on ℓ. For the base case, let $\ell = 0$ and consider an arbitrary $1 \leq i \leq \delta$. Set $\mathcal{X}_{0,i}$ is finite by definition. Furthermore, for $\mathcal{G}$ an arbitrary Boolean (Col, δ)-graph and v an arbitrary vertex of it, $(\mathbf{v}_0)_i$ is either 0 or 1, and so it is in $\mathcal{X}_{0,i}$. For the induction step, consider $1 \leq \ell \leq L$ and $1 \leq i \leq \delta_\ell$. We first show that $\mathcal{X}_{\ell,i}$ is finite. Since each $\mathcal{X}_{\ell-1,j}$ is finite by induction hypothesis, there exist only a finite number of vectors $\mathbf{x}$ and $\{\mathbf{y}^c\}_{c\in\mathsf{Col}}$ such that each $(\mathbf{x})_j$ and $(\mathbf{y}^c)_j$ are in $\mathcal{X}_{\ell-1,j}$. Moreover, each element of $\mathcal{X}_{\ell,i}$ is equal to $\mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}})$ for some such $\mathbf{x}$ and $\{\mathbf{y}^c\}_{c\in\mathsf{Col}}$ by definition, and so the number of elements in $\mathcal{X}_{\ell,i}$ is finite. Now, let $\mathcal{G}$ be an arbitrary (Col, δ)-graph and let v be an arbitrary vertex in it. The value $(\mathbf{v}_\ell)_i$ is defined by the ith element of equation (3). Let $\mathbf{x} = \mathbf{v}_{\ell-1}$ and, for each $c \in \mathsf{Col}$, let $\mathbf{y}^c = \max\{\mathbf{w}_{\ell-1} \mid \langle v, w\rangle \in \mathcal{E}^c\}$. The induction hypothesis and the inclusion of 0 in $\mathcal{X}_{\ell-1,j}$ ensure that the jth element in each of these vectors is in $\mathcal{X}_{\ell-1,j}$, and so $\mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}}) \in \mathcal{X}_{\ell,i}$. Comparing equations (3) and (4) shows that $\mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}}) = (\mathbf{v}_\ell)_i$ and so $(\mathbf{v}_\ell)_i \in \mathcal{X}_{\ell,i}$ as required. $\square$

3.2 Construction of the Equivalent Program

In this section, we describe the construction of a Datalog$^{\neg}$ program $\mathcal{P}_{\mathcal{N}}$ such that applying the transformation $T^{L+2}_{\mathcal{P}_{\mathcal{N}}}$ to an arbitrary (Col, δ)-dataset D yields the same result as applying $T_{\mathcal{N}}$ to D. In other words, if we successively apply to D the immediate consequence operator $T_{\mathcal{P}_{\mathcal{N}}}$ a total of $L + 2$ times, then the resulting dataset coincides with that obtained by applying $T_{\mathcal{N}}$ once to D. Intuitively, the first application of $T_{\mathcal{P}_{\mathcal{N}}}$ captures the canonical encoding; each of the L subsequent applications captures the application of a layer of $\mathcal{N}$, and the last application captures simultaneously $\mathcal{N}$'s classification function and the canonical decoder.

Program $\mathcal{P}_{\mathcal{N}}$ introduces auxiliary predicates of the form $U_{\ell,i,\alpha}$, where ℓ ranges from 0 to L, i ranges from 1 to δ_ℓ, and α is an element of $\mathcal{X}_{\ell,i}$. These predicates are used to represent the feature vectors obtained through the application of $\mathcal{N}$. For example, if $T_{\mathcal{P}_{\mathcal{N}}}$ derives $U_{\ell,i,\alpha}(a)$ after one or more applications to a (Col, δ)-dataset D, this means that when $\mathcal{N}$ is applied to the canonical encoding of D, the ith element of the feature vector computed in layer ℓ for vertex v^a will take value α. Program $\mathcal{P}_{\mathcal{N}}$ also introduces fresh predicates of the form $V_{c,\ell,i,\alpha}$, where c is a colour in Col, and ℓ, i, and α range over the same values as in $U_{\ell,i,\alpha}$. Intuitively, if $T_{\mathcal{P}_{\mathcal{N}}}$ derives the fact $V_{c,\ell,i,\alpha}(a)$ after one or more applications to a (Col, δ)-dataset D, this indicates that the vertex v^a has a c-neighbour vertex w such that when $\mathcal{N}$ is applied to the canonical encoding of D, the jth element of the vector computed by $\mathcal{N}$ in layer ℓ for w is α. Finally, program $\mathcal{P}_{\mathcal{N}}$ introduces an auxiliary binary predicate W^c for each $c \in \mathsf{Col}$, which has the same meaning as E^c. Using this predicate instead of E^c helps ensure that $T^{L+2}_{\mathcal{P}_{\mathcal{N}}}$ will not derive spurious facts over auxiliary predicates.

Program $\mathcal{P}_{\mathcal{N}}$ is the union of $L + 2$ programs $\mathcal{P}_{\mathcal{N},0}, \ldots, \mathcal{P}_{\mathcal{N},L+1}$. Rule bodies in $\mathcal{P}_{\mathcal{N},0}$ and rule heads in $\mathcal{P}_{\mathcal{N},L+1}$ mention only predicates in the data signature. Furthermore, for $1 \leq \ell \leq L + 1$, predicates in rule bodies in $\mathcal{P}_{\mathcal{N},\ell}$ appear only in rule heads in $\mathcal{P}_{\mathcal{N},\ell-1}$. Thus, the application of $T^{L+2}_{\mathcal{P}_{\mathcal{N}}}$ to a (Col, δ)-dataset D can be understood in $L + 2$ stages where, in the ℓth stage, only rules from $\mathcal{P}_{\mathcal{N},\ell-1}$ will fire.

The first program, $\mathcal{P}_{\mathcal{N},0}$, captures the canonical encoding.

Definition 3. *$\mathcal{P}_{\mathcal{N},0}$ is the Datalog$^\neg$ program that contains the following rules instantiated for each two positions $i, j \leq \delta_0$ and each colour $c \in \mathsf{Col}$:*

$$U_i(x) \to U_{0,i,1}(x), \tag{5}$$
$$U_j(x) \wedge \textsf{not}\, U_i(x) \to U_{0,i,0}(x), \tag{6}$$
$$E^c(x,y) \wedge \textsf{not}\, U_i(x) \to U_{0,i,0}(x), \tag{7}$$
$$E^c(y,x) \wedge \textsf{not}\, U_i(x) \to U_{0,i,0}(x), \tag{8}$$
$$E^c(x,y) \wedge U_i(y) \to V_{c,0,i,1}(x), \tag{9}$$
$$E^c(x,y) \to W^c(x,y). \tag{10}$$

Rules (6), (7), and (8) together encode the rule $\textsf{not}\, U_i(x) \to U_{0,i,0}(x)$, but their positive body atoms ensure that they will only be fired during the first application of $\mathcal{P}_{\mathcal{N}}$.

Programs $\mathcal{P}_{\mathcal{N},1}, \ldots, \mathcal{P}_{\mathcal{N},L}$, capture the behaviour of equation (2) for layers $1, \ldots, L$ of the GNN, respectively.

Definition 4. *For each $1 \leq \ell \leq L$, $\mathcal{P}_{\mathcal{N},\ell}$ is the Datalog$^\neg$ program consisting of the following rules instantiated for each $i \leq \delta_\ell$, each vector $\mathbf{x}$ of dimension $\delta_{\ell-1}$ with $(\mathbf{x})_j \in \mathcal{X}_{\ell-1,j}$, each $|\mathsf{Col}|$-tuple $(\mathbf{y}^c)_{c\in\mathsf{Col}}$ of vectors of dimension $\delta_{\ell-1}$ where $(\mathbf{y}^c)_j \in \mathcal{X}_{\ell-1,j}$, and each $c' \in \mathsf{Col}$:*

$$\bigwedge_{1\leq j \leq \delta_{\ell-1}} U_{\ell-1,j,(\mathbf{x})_j}(x) \wedge \bigwedge_{\substack{c\in\mathsf{Col}\\ 1\leq j\leq\delta_{\ell-1}}} \Gamma_{c,j} \to U_{\ell,i,\alpha}(x), \tag{11}$$

$$W^{c'}(y,x) \wedge \bigwedge_{1\leq j \leq \delta_{\ell-1}} U_{\ell-1,j,(\mathbf{x})_j}(x) \wedge \bigwedge_{\substack{c\in\mathsf{Col}\\ 1\leq j\leq\delta_{\ell-1}}} \Gamma_{c,j} \to V_{c',\ell,i,\alpha}(y), \tag{12}$$

$$W^{c'}(x,y) \to W^{c'}(x,y), \tag{13}$$

where $\alpha = \mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}})$ and $\Gamma_{c,j}$ is the conjunction

$$W^c(x, y_{c,j}) \wedge U_{\ell-1,j,(\mathbf{y}^c)_j}(y_{c,j}) \wedge \bigwedge_{\substack{\beta\in\mathcal{X}_{\ell-1,j}\\ s.t.\ \beta>(\mathbf{y}^c)_j}} \textsf{not}\, V_{c,\ell-1,j,\beta}(x), \tag{14}$$

and also, each rule obtained by replacing, for one or more c such that $\mathbf{y}^c$ is null, the conjunction $\bigwedge_{j=1}^{\delta_{\ell-1}} \Gamma_{c,j}$ in rule (11) *or* (12) *by the negated body atom $\textsf{not}\, W^c(x, y_c)$.*

We explain the intuition of these rules for the case $\ell = 1$, which is analogous to the other cases. The instances of rule (11) in program $\mathcal{P}_{\mathcal{N},1}$ encode the behaviour of the first layer of $\mathcal{N}$ on the canonical encoding of (Col, δ)-datasets. To see this, consider an arbitrary (Col, δ)-dataset D and a constant a mentioned in it. By equation (2), if $\mathbf{x}$ is the initial feature vector for v^a in the canonical encoding of D, and for each colour $c \in \mathsf{Col}$, $\mathbf{y}^c$ is the result of the max aggregation of the initial feature vectors for c-neighbours of v^a, then the ith element of feature vector for v^a in layer 1 is equal to $\mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}})$. This behaviour is precisely captured by the instances of rule (11) for i, $\mathbf{x}$, and $\mathbf{y}^c$. To see this, observe that the body of the rule contains an atom $U_{0,j,1}(x)$ if the jth component of $\mathbf{x}$ is 1, and otherwise it contains an atom $U_{0,j,0}(x)$. However, rules (5), (6) , (7), and (8) ensure that the input contains $U_{0,j,1}(a)$ (resp. $U_{0,j,0}(a)$) if and only if $U_j(a) \in D$ (resp. $U_j(a) \notin D$). Therefore, the first conjunction in the rule matches a if and only if the initial feature vector (via the canonical encoding) of v^a is precisely $\mathbf{x}$. Furthermore, for each colour c and position j, the body of rule (11) includes a conjunction $\Gamma_{c,j}$ ensuring that, if the rule applies for constant a, then $(\mathbf{y}^c)_j$ has the intended meaning as the position j of the max aggregation of feature vectors of c-neighbours of v^a. For example, if $(\mathbf{y}^c)_j = 0$, $\Gamma_{c,j}$ includes $W^c(x, y_{c,j}) \wedge U_{0,j,0}(y_{c,j})$ to represent that there is a neighbour contributing with this value, and furthermore it includes a negated atom $\textsf{not}\, V_{c,0,j,1}(x)$ to ensure that there exists no c-neighbour of v^a with a 1 in its j-th position. If $\mathbf{y}^c$ is null, it might be the case that v^a has no c-neighbours; for this, we have a copy of rule (11) where the conjunction of all $\Gamma_{c,j}$ for $1 \leq j \leq \delta_{\ell-1}$ is replaced by the atom $\textsf{not}\, W^c(x, y_c)$. When the rule fires, it derives $U_{1,i,\alpha}(a)$ which, as we have seen, represents that the ith component of the feature vector for v^a in layer 1 has value $\alpha = \mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}})$.

Rule (12) includes all the atoms in the body of rule (11) plus an atom of the form $W^{c'}(y, x)$. Intuitively, it simulates in a single step an application of rule (11) followed by an application of the rule $W^{c'}(y,x) \wedge U_{1,i,\alpha}(x) \to V_{c',1,i,\alpha}(y)$, and so it defines the value of the auxiliary predicates $V_{c,1,i,\alpha}$ for layer 1. These predicates might appear in negated body atoms in the next subprogram, $\mathcal{P}_{\mathcal{N},2}$ Finally, rule (13) ensures that the information about connectivity between constants is preserved.

The last subprogram $\mathcal{P}_{\mathcal{N},L+1}$ simulates the application of $\mathcal{N}$'s classification function cls and the canonical decoder.

Definition 5. *Program $\mathcal{P}_{\mathcal{N},L+1}$ is the Datalog$^\neg$ program containing the following rule instantiated for each $i \leq \delta_L$, each $\alpha \in \mathcal{X}_{L,i}$ such that $\mathsf{cls}(\alpha) = 1$, and each $c \in \mathsf{Col}$:*

$$U_{L,i,\alpha}(x) \to U_i(x) \tag{15}$$
$$W^c(x,y) \to E^c(x,y) \tag{16}$$

Finally, we define $\mathcal{P}_{\mathcal{N}}$ as $\bigcup_{\ell=0}^{L+1} \mathcal{P}_{\mathcal{N},\ell}$.

The following lemma confirms the correctness of our interpretation of the facts over auxiliary predicates derived in each stage of the application of $T_{\mathcal{P}_{\mathcal{N}}}^{L+1}$ to D.

Lemma 6. *For each (Col, δ)-dataset D and constant a in it, each layer $0 \leq \ell \leq L$, $i \leq \delta_\ell$, $\alpha \in \mathcal{X}_{\ell,i}$, and $c \in \mathsf{Col}$, it holds that: $U_{\ell,i,\alpha}(a) \in T_{\mathcal{P}_{\mathcal{N}}}^{\ell+1}(D)$ if and only if $(\mathbf{v}_\ell^a)_i = \alpha$; furthermore, $V_{c,\ell,i,\alpha}(a) \in T_{\mathcal{P}_{\mathcal{N}}}^{\ell+1}(D)$ if and only if there exists $\langle a, b\rangle \in \mathcal{E}^c$ such that $(\mathbf{v}_\ell^b)_i = \alpha$; finally, if $\ell \geq 1$, $W^c(a,b) \in T_{\mathcal{P}_{\mathcal{N}}}^{\ell}(D)$ if and only if $E^c(a,b) \in D$.*

Proof. We use induction over ℓ to prove all three claims in the lemma simultaneously. For the base case ($\ell = 0$), all claims follow from the rules of $\mathcal{P}_{\mathcal{N},0}$ and the fact that the canonical encoding ensures $(\mathbf{v}_0^b)_i = 1$ iff $U_i(b) \in D$ for any constant b in D, and $\langle a, b\rangle \in \mathcal{E}^c$ if and only if $E^c(a,b) \in D$. For the inductive step, consider $1 \leq \ell \leq L$; we prove the first claim by showing the implication in each direction.

($\Rightarrow$). Assume $U_{\ell,i,\alpha}(a) \in T_{\mathcal{P}_{\mathcal{N}}}^{\ell+1}(D)$. The definition of $\mathcal{P}_{\mathcal{N}}$ ensures that there exists a rule $r \in \mathcal{P}_{\mathcal{N},\ell}$ of the form (11) introduced for $\delta_{\ell-1}$-dimensional Boolean vectors $\mathbf{x}$ and $(\mathbf{y}^c)_{c\in\mathsf{Col}}$ with jth elements in $\mathcal{X}_{\ell-1,j}$ such that $\alpha = \mathsf{Val}(\ell, i, \mathbf{x}, \mathbf{y}^c)$, and possibly with some $\bigwedge_{j=1}^{\delta_{\ell-1}} \Gamma_{c,j}$

replaced by $\mathsf{not}\ W^c(x, y_c)$, and a substitution ν such that $P\nu \in T^{\ell}_{\mathcal{P}_\mathcal{N}}(D)$ (resp. $Q\nu \notin T^{\ell}_{\mathcal{P}_\mathcal{N}}(D)$) for each positive (resp. negated) body atom P (resp. Q) in r. Applying the induction hypothesis to these facts $P\nu$ and $Q\nu$ ensures that $\mathbf{x}$ is the feature vector in layer $\ell - 1$ for $v^{x\nu}$, and for each $c \in \mathsf{Col}$, $\mathbf{y}^c$ is the result of max aggregation over c-neighbours in layer $\ell - 1$ of the application of $\mathcal{N}$ to D. Hence, $(\mathbf{v}^a_\ell)_i = \alpha$, as required.

($\Leftarrow$). Assume $(\mathbf{v}^a_\ell)_i = \alpha$ for some constant a and value $\alpha \in \mathcal{X}_{\ell,i}$. Let $\mathbf{x}$ be the feature vector in layer $\ell - 1$ for vertex v^a and $\mathbf{y}^c$ the result of the max aggregation over c-neighbours of vectors in layer $\ell - 1$. By Lemma 2, the jth element in each of these vectors is in $\mathcal{X}_{\ell-1,j}$, and so there is a corresponding rule r of the form (11) in $\mathcal{P}_\mathcal{N}$ with $\bigwedge_{j=1}^{\delta_{\ell-1}} \Gamma_{c,j}$ replaced by $\mathsf{not}\ W^c(x, y_c)$ for each $c \in \mathsf{Col}$ such that v^a has no c-neighbours. Let ν be the substitution that maps x to a and each variable $y_{c,j}$ in r to the constant $b_{c,j}$ corresponding to the c-neighbour of $v^{x\nu}$ which contributes with the maximum jth value to the max aggregation in layer $\ell - 1$ over colour c. The induction hypothesis ensures that ν grounds all positive body atoms of r in $T^{\ell}_{\mathcal{P}_\mathcal{N}}(D)$. Suppose now, for the sake of contradiction, that Q is a negated body atom in this rule such that $Q\nu \in T^{\ell}_{\mathcal{P}_\mathcal{N}}(D)$. If $Q = W^c(x, y_c)$, then our induction hypothesis implies that v^a has a c-neighbour, contradicting our choice of r. Otherwise, there exists a c-neighbour of $v^{x\nu}$ with an element in the jth position that is strictly greater than $(\mathbf{y}^c)_j$, which contradicts our definition of $\mathbf{y}^c$. Finally, the head of rule r is of form $U_{\ell,i,\alpha}(x)$; hence, $U_{\ell,i,\alpha}(a) \in T^{\ell+1}_{\mathcal{P}_\mathcal{N}}(D)$ as required.

The proof for the second claim in the lemma is analogous to that of the first, and the third follows directly from the induction hypothesis and rules (10) and (13). $\square$

The following theorem relies on Lemma 6 to ensure that $\mathcal{P}_\mathcal{N}$ faithfully captures $\mathcal{N}$'s behaviour: for a dataset D, the result of applying the immediate consequence operator $T_{\mathcal{P}_\mathcal{N}}$ to D once per layer of the GNN, plus two additional applications (for encoding and decoding, respectively) is equivalent to applying the GNN-based transformation to D.

Theorem 7. *For each* (Col, δ)*-dataset* D*, we have that* $T_\mathcal{N}(D) = T^{L+2}_{\mathcal{P}_\mathcal{N}}(D)$*.*

Proof. A simple induction on ℓ shows that $T^{L+2}_{\mathcal{P}_\mathcal{N}}(D)$ may only contain facts $U_i(a)$ for some $i \leq \delta$ or $E^c(a, b)$ for some $c \in \mathsf{Col}$ and constants a, b in D. The definition of $T_\mathcal{N}$ ensures that $T_\mathcal{N}(D)$ also contains only facts of these forms. Thus, we show that each fact of one of these forms is in $T_\mathcal{N}(D)$ if and only if it is in $T^{L+2}_{\mathcal{P}_\mathcal{N}}(D)$.

For each $E^c(a, b)$ with $c \in \mathsf{Col}$ and a, b constants in D, rule (16) in $\mathcal{P}_\mathcal{N}$ and the third claim in Lemma 6 ensure that $E^c(a, b) \in T^{L+2}_{\mathcal{P}_\mathcal{N}}(D)$ iff $E^c(a, b) \in D$, from which the claim follows since $T_\mathcal{N}$ does not alter binary facts.

For each $i \leq \delta$, constant a in D, and $\alpha \in \mathcal{X}_{L,i}$, we have that by Lemma 6, $U_{L,i,\alpha}(a) \in T^{L+1}_{\mathcal{P}_\mathcal{N}}(D)$ iff $(\mathbf{v}^a_L)_i = \alpha$. This and Lemma 2 together ensure that $U_{L,i,(\mathbf{v}^a_L)_i}(a) \in T^{L+1}_{\mathcal{P}_\mathcal{N}}(D)$ and that there is no other fact of the form $U_{L,i,\alpha'}(a)$ in $T^{L+1}_{\mathcal{P}_\mathcal{N}}(D)$ for $\alpha' \in \mathcal{X}_{L,i}$. Then, the rules in $\mathcal{P}_\mathcal{N}$ with predicate U_i in the head, which are of form (15), ensure that $U_i(a) \in T^{L+2}_{\mathcal{P}_\mathcal{N}}(D)$ iff $\mathsf{cls}((\mathbf{v}^a_L)_i) = 1$, which in turn holds iff $U_i(a) \in T_\mathcal{N}(D)$ by definition of the canonical decoding. Hence, $U_i(a) \in T^{L+2}_{\mathcal{P}_\mathcal{N}}(D)$ iff $U_i(a) \in T_\mathcal{N}(D)$. $\square$

Each rule in $\mathcal{P}_\mathcal{N}$ with a unary predicate in the head is tree-shaped. Consider the primal graph of the rule, obtained by introducing a vertex for each variable and an (undirected) edge between vertices for variables occurring together in a binary body atom. Looking at rules (5) through (15), it is easy to see that the primal graph is a tree rooted in the variable in the head of the rule. The height of this tree is at most 2 and each vertex in the tree participates in at most $|\mathsf{Col}| \cdot \max(\delta_0, \dots, \delta_L)$ edges.

4 Monotonic Max GNNs

In this section, we consider the family of monotonic max GNNs proposed in (Tena Cucala et al. 2023).

Definition 8. *A max* (Col, δ)*-GNN is* monotonic *if the values of all its matrices* $\mathbf{A}_\ell$ *and* $\mathbf{B}^c_\ell$ *are non-negative, its activation function* σ *is unbounded above, has a non-negative range, and is monotonically increasing, and if its classification function* cls *is a step function with a threshold* $t \in \mathbb{R}$ *such that* $\mathsf{cls}(x) = 1$ *if* $x \geq t$*, and otherwise* $\mathsf{cls}(x) = 0$*, for each* $x \in \mathbb{R}$*.*

For the remainder of this entire section, let us consider a fixed, but arbitrary monotonic max (Col, δ)-GNN $\mathcal{N}$.

It was shown in (Tena Cucala et al. 2023) that the operator $T_\mathcal{N}$ is monotonic under homomorphisms and can be captured by a Datalog program without negation. Although the Datalog$^\neg$ program $\mathcal{P}_\mathcal{N}$ constructed from $\mathcal{N}$ as in Section 3.2 still contains negated body atoms, multiple applications of a non-monotonic Datalog$^\neg$ program can result in an operator that is monotonic under homomorphisms. For example, for a program $\mathcal{P}$ consisting of the following rules:

$$\begin{aligned} P(x) &\to Q_1(x), & Q_1(x) &\to R(x) \\ \mathsf{not}\ P(x) &\to Q_2(x), & Q_2(x) &\to R(x), \end{aligned}$$

operator $T^2_\mathcal{P}$ is monotonic under homomorphisms, even though $T_\mathcal{P}$ is not. This is indeed the case in our setting: if $\mathcal{N}$ is monotonic, then by Theorem 7, operator $T^{L+2}_{\mathcal{P}_\mathcal{N}}$ is monotonic under homomorphisms despite the fact that $\mathcal{P}_\mathcal{N}$ contains negation and the immediate consequence operator $T_{\mathcal{P}_\mathcal{N}}$ is nonmonotonic.

In what follows, we show that, by simply removing all negated atoms from rule bodies in $\mathcal{P}_\mathcal{N}$, the resulting Datalog program $\mathcal{P}^+_\mathcal{N}$ satisfies the statement in Theorem 7—that is, $T^{L+2}_{\mathcal{P}^+_\mathcal{N}}(D) = T_\mathcal{N}(D)$ for any (Col, δ)-dataset D. Furthermore, $\mathcal{P}^+_\mathcal{N}$ can be unfolded into a program $\mathcal{P}'_\mathcal{N}$ that does not contain auxiliary predicates and which is equivalent to that obtained in (Tena Cucala et al. 2023).

4.1 Eliminating Negation

Consider the program $\mathcal{P}^+_\mathcal{N}$ obtained from $\mathcal{P}_\mathcal{N}$ by removing all negated body atoms, and let D be a (Col, δ)-dataset.

When applied to D, the set of facts derived from the application of $\mathcal{P}_\mathcal{N}$ is (often strictly) contained in those derived from the application of $\mathcal{P}^+_\mathcal{N}$. Indeed, by eliminating body atoms, we have dropped some preconditions required for the application of the affected rules. We next show, however, that both programs yield exactly the same facts when applied $L+2$ times—that is, the number of applications required to simulate $T_\mathcal{N}$—to D. This is so despite the fact that, during the first $L+1$ applications, $\mathcal{P}^+_\mathcal{N}$ may derive additional facts.

As discussed in Section 3.2, rules (11) in $\mathcal{P}_\mathcal{N}$ simulate the application of layer ℓ of $\mathcal{N}$ to D. In particular, the rule matches a set of facts representing a vertex v^a with feature vector $\mathbf{x}$ in layer $\ell-1$ and elements of feature vectors of c-neighbours of v^a in layer $\ell-1$ that contribute to the result $\mathbf{y}^c$ of the max aggregation for each colour c. The rule derives $U_{\ell,i,\alpha}(a)$, where $\alpha = \mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}})$ coincides with $(\mathbf{v}^a_\ell)_i$, the value of the ith element of the feature vector for v^a computed in layer ℓ by $\mathcal{N}$. The negated body atoms, in particular, ensure that $\mathbf{y}^c$ is the result of max aggregation of all c-neighbours, by negating the existence of any additional c-neighbour with a feature vector whose jth element is greater than $(\mathbf{y}^c)_j$—or negating the existence of c-neighbours altogether. Once negated body atoms have been removed, $\mathbf{y}^c$ no longer represents the result of the max aggregation; instead, the rule would now match any set of facts that represent a vertex v^a with feature vector $\mathbf{x}$ in layer $\ell-1$ and (possibly) an arbitrary c-neighbour whose feature vector in layer $\ell-1$ has a jth element equal to $(\mathbf{y}^c)_j$ for each colour c—not necessarily a neighbour that contributes to the max aggregation.

Since $\mathcal{N}$ is monotonic, however, all matrix weights are non-negative and the activation function is monotonic. Then, value $\alpha = \mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}})$ in $U_{\ell,i,\alpha}(a)$ is now at most that of $(\mathbf{v}^a_\ell)_i$. This is the key property ensuring that $\mathcal{P}^+_\mathcal{N}$ and $\mathcal{P}_\mathcal{N}$ yield the same facts after $L+2$ rounds of application to D. To see this, suppose that $\mathcal{P}^+_\mathcal{N}$ uses a fact $U_{L,i,\alpha}(a)$ to derive fact $U_i(a)$ in the last round (i.e. using rule (15)). Since $\mathcal{N}$ is monotonic, its classification function is a monotonic step function with a threshold, and so we know that α is greater than this threshold. But then, since $\alpha \le (\mathbf{v}^a_L)_i$, we have that $(\mathbf{v}^a_L)_i$ also exceeds the threshold and so $T_\mathcal{N}$ also derives $U_i(a)$ on D; by Theorem 7, so does $T^{L+2}_{\mathcal{P}_\mathcal{N}}$. Hence, all facts derived by $T^{L+2}_{\mathcal{P}^+_\mathcal{N}}$ on D are also derived by $T^{L+2}_{\mathcal{P}_\mathcal{N}}$.

Theorem 9. *Let $\mathcal{P}^+_\mathcal{N}$ be the program obtained from $\mathcal{P}_\mathcal{N}$ by removing all negated body atoms from bodies of rules. If $\mathcal{N}$ is a monotonic max (Col,δ)-GNN, then, $T_\mathcal{N}(D) = T^{L+2}_{\mathcal{P}^+_\mathcal{N}}(D)$ for each (Col,δ)-dataset D.*

Proof. The inclusion $T_\mathcal{N}(D) \subseteq T^{L+2}_{\mathcal{P}^+_\mathcal{N}}(D)$ follows from Theorem 7. To prove the converse, we show inductively on ℓ that, for each $0 \le \ell \le L$, $i \le \delta_\ell$, constant a in D, and $\alpha \in \mathcal{X}_{\ell,i}$, if $U_{\ell,i,\alpha}(a) \in T^{\ell+1}_{\mathcal{P}^+_\mathcal{N}}(D)$, then $(\mathbf{v}^a_\ell)_i \ge \alpha$, and if $\ell \ge 1$, $W^c(a,b) \in T^{\ell}_{\mathcal{P}^+_\mathcal{N}}(D)$ iff $E^c(a,b) \in D$.

In the base case ($\ell = 0$), each $\mathcal{X}_{0,i}$ is equal to $\{0,1\}$, and each $(\mathbf{v}^a_0)_i$ can also take values only from $\{0,1\}$. Thus, for $\alpha = 0$, $(\mathbf{v}^a_0)_i \ge 0$ holds immediately. For $\alpha = 1$, $U_{0,i,1}(a) \in T_{\mathcal{P}^+_\mathcal{N}}(D)$ implies $U_i(a) \in D$ and so $(\mathbf{v}^a_0)_i = 1$. The second claim follows as in the proof of Lemma 6.

For the induction step ($1 \le \ell \le L$), we assume that $U_{\ell,i,\alpha}(a) \in T^{\ell+1}_{\mathcal{P}^+_\mathcal{N}}(D)$ and show that $(\mathbf{v}^a_\ell)_i \ge \alpha$. From the definition of $\mathcal{P}^+_\mathcal{N}$, there is a rule $r \in \mathcal{P}_\mathcal{N}$ and a substitution ν mapping x to a such that for each positive body atom P in r, we have $P\nu \in T^{\ell}_{\mathcal{P}^+_\mathcal{N}}(D)$. This rule is in $\mathcal{P}_{\mathcal{N},\ell}$ and of the form (11). Thus, $U_{\ell-1,j,(\mathbf{x})_j}(a) \in T^{\ell}_{\mathcal{P}^+_\mathcal{N}}(D)$ for each j. Since $(\mathbf{x})_j \in \mathcal{X}_{\ell-1,j}$, the induction hypothesis implies that $(\mathbf{v}^a_{\ell-1})_j \ge (\mathbf{x})_j$. By an analogous argument, for each $c \in \mathsf{Col}$ we obtain that if r mentions $y_{c,j}$, then $W^c(a, b_{c,j}) \in T^{\ell}_{\mathcal{P}^+_\mathcal{N}}(D)$, for $b_{c,j} = \nu(y_{c,j})$, so by the induction hypothesis it follows that $E^c(a, b_{c,j}) \in D$ and so $\langle v^a, v^{b_{c,j}}\rangle \in \mathcal{E}^c$, and $(\mathbf{v}^{b_{c,j}}_{\ell-1})_j \ge (\mathbf{y}^c)_j$. Vectors $\mathbf{x}$ and $\mathbf{y}^c$ have no negative elements because σ has a nonnegative range. Applying this observation to equation (3), substituting the inequalities derived so far, and using the facts that weights of $\mathcal{N}$ are nonnegative and σ is monotonically increasing, we obtain $(\mathbf{v}^a_\ell)_i \ge \mathsf{Val}(\ell, i, \mathbf{x}, (\mathbf{y}^c)_{c\in\mathsf{Col}}) = \alpha$, as required.

To complete the proof, consider $U_i(a) \in T^{L+2}_{\mathcal{P}^+_\mathcal{N}}(D)$. No rule other than those of the form (15) in $\mathcal{P}_\mathcal{N}$ has predicate U_i in the head, and thus $U_{L,i,\alpha}(a) \in T^{L+1}_{\mathcal{P}^+_\mathcal{N}}(D)$ for some α such that $\mathsf{cls}(\alpha) = 1$. The claim shown above implies $(\mathbf{v}^a_L)_i \ge \alpha$, and so, since cls is monotonically increasing, $\mathsf{cls}((\mathbf{v}^a_L)_i) = 1$; hence, $U_i(a) \in T_\mathcal{N}(D)$. Similarly, for a binary $E^c(a,b) \in T^{L+2}_{\mathcal{P}^+_\mathcal{N}}(D)$ rule (16) ensures that $W^c(a,b) \in T^{L+1}_{\mathcal{P}^+_\mathcal{N}}(D)$, and by the claim shown by induction we have $E^c(a,b) \in D$, and so $E^c(a,b) \in T_\mathcal{N}(D)$, since GNNs do not affect binary facts. Thus, $T^{L+2}_{\mathcal{P}^+_\mathcal{N}}(D) \subseteq T_\mathcal{N}(D)$. $\square$

4.2 Unfolding the Negation-free Program

We conclude this section by describing the unfolding of $\mathcal{P}^+_\mathcal{N}$ into a program $\mathcal{P}'_\mathcal{N}$ without auxiliary predicates and such that its application to a (Col,δ)-dataset is equivalent to a single application of $T_\mathcal{N}$. To this end, we exploit standard techniques based on hyperresolution (Robinson 1974).

Let r be a Datalog (i.e. negation-free) rule $B_1(z_1) \wedge \cdots \wedge B_n(z_n) \to H$. For each $1 \le i \le n$, let r_i be a Datalog rule $\Gamma_i \to B_i(x_i)$, and assume w.l.o.g. that no rules share variables. A hyperresolution step with main premise r and side premises $r_1, \ldots, r_n$ yields $\Gamma_1\{x_1 \mapsto z_1\} \wedge \cdots \wedge \Gamma_n\{x_n \mapsto z_n\} \to H$. The unfolding procedure initialises a set of rules $\mathcal{S}$ to $\mathcal{P}^+_{\mathcal{N},L+1}$ and eagerly performs hyperresolution steps with a rule in $\mathcal{S}$ as the main premise and rules in $\bigcup_{\ell=0}^{L} \mathcal{P}^+_{\mathcal{N},\ell}$ as side premises, adding the resulting inferences to $\mathcal{S}$ and renaming away shared variables, until no more rules can be added to $\mathcal{S}$. The output program $\mathcal{P}'_\mathcal{N}$ is the subset of $\mathcal{S}$ mentioning only predicates in the data signature $\{U_i\}_{i\le\delta} \cup \{E^c\}_{c\in\mathsf{Col}}$.

Termination relies on the restriction that rules of $\mathcal{P}^+_{\mathcal{N},L+1}$ are never used as side premises. Indeed, each hyperresolution step produces a rule with body atoms of the form $U_{\ell-1,j,\beta}(z')$ from a main premise with body atoms of the form $U_{\ell,i,\alpha}(z)$; thus, after $L+2$ consecutive eager hyperresolution steps, all body predicates of the resulting rule are contained in the data signature, and since no allowed side premise mentions any of these predicates in the head, no more hyperresolution steps are possible.

Theorem 10. *If $\mathcal{N}$ is a monotonic max (Col, δ)-GNN, then $T_{\mathcal{N}}(D) = T_{\mathcal{P}'_{\mathcal{N}}}(D)$ for each (Col, δ)-dataset D.*

Proof. We show that $T_{\mathcal{P}'_{\mathcal{N}}}(D) = T^{L+2}_{\mathcal{P}^+_{\mathcal{N}}}(D)$ which, together with Theorem 9 implies our claim. We first prove that each fact $\alpha \in T^{L+2}_{\mathcal{P}^+_{\mathcal{N}}}(D)$ is also in $T_{\mathcal{P}'_{\mathcal{N}}}(D)$. Let $D_{L+2} = \{\alpha\}$ and $\mathcal{S}$ the saturated rule set produced by hyperresolution. For each $0 \leq \ell \leq L+1$, we inductively construct (in decreasing order of ℓ) a dataset D_ℓ and rule r_ℓ such that: (1) $D_\ell \subseteq T^{\ell}_{\mathcal{P}^+_{\mathcal{N}}}(D)$; (2) $\alpha \in T_{r_\ell}(D_\ell)$, and (3) $r_\ell \in \mathcal{S}$. Consider one such ℓ and assume that $D_{\ell'}$ is defined for each $L+2 \geq \ell' > \ell$, satisfying all three conditions. Condition 1 ensures that each fact in $D_{\ell+1}$ can be derived from a rule in $\mathcal{P}^+_{\mathcal{N}}$ and facts in $T^{\ell}_{\mathcal{P}^+_{\mathcal{N}}}(D)$; let $\mathcal{Q}_\ell$ be the set of all such rules. By the form of atoms in $D_{\ell+1}$, we have that $\mathcal{Q}_\ell \subseteq \mathcal{P}^+_{\mathcal{N},\ell}$. We define D_ℓ as the support for these rules in $T^{\ell}_{\mathcal{P}^+_{\mathcal{N}}}(D)$, and so Condition 1 holds. If $\ell = L+1$, then $D_{\ell+1} = \{\alpha\}$ and so clearly $\mathcal{Q}_{\ell+1}$ consists of a single rule from $\mathcal{P}^+_{\mathcal{N},L+1}$, which we choose as r_ℓ; Condition 2 then holds trivially. Condition 3 also holds due to our initialisation of $\mathcal{S}$. If $\ell < L+1$, then we define r_ℓ as the result of a hyperresolution step using $r_{\ell+1}$ as main premise and rules in $\mathcal{Q}_\ell$ as side premises; this step is possible because, by Condition 2 in the induction hypothesis, there is a substitution ν grounding each body atom P of $r_{\ell+1}$ in $D_{\ell+1}$, and so for each body atom of $r_{\ell+1}$ we use the rule $r_{P\nu}$ in $\mathcal{Q}_\ell$ that derives $P\nu$ from D_ℓ using a substitution $\mu_{P\nu}$; such rule exists since we have already shown that Condition 1 is satisfied. Taking the union ν and all $\mu_{P\nu}$ yields a substitution that grounds the body of r_ℓ in D_ℓ and makes its head equal to α, and so Condition 2 holds. Furthermore, $r_\ell \in \mathcal{S}$ since so is $r_{\ell+1}$ and we performed a hyperresolution step with $\mathcal{Q}_\ell \subseteq \mathcal{P}^+_{\mathcal{N},\ell}$, so Condition 3 is verified. This concludes the inductive construction. We now have that $r_0 \in \mathcal{S}$ by Condition 3; furthermore, $D_0 \subseteq D$ by Condition 1 and so r_0 mentions only predicates in the data signature, so $r_0 \in \mathcal{P}'_{\mathcal{N}}$. Finally, Condition 2 guarantees that $\alpha \in T_{r_0}(D_0)$, and so $\alpha \in T_{r_0}(D)$, as T_{r_0} is monotonic under homomorphisms; hence, $\alpha \in T_{\mathcal{P}'_{\mathcal{N}}}(D)$.

We finally argue that $T_{\mathcal{P}'_{\mathcal{N}}}(D) \subseteq T^{L+2}_{\mathcal{P}'_{\mathcal{N}}}(D)$. Let $r_0 \in \mathcal{P}'_{\mathcal{N}}$ and let $r_1, \ldots, r_{L+2}$ be its hyperresolution proof, with $r_{L+2} \in \mathcal{P}^+_{\mathcal{N},L+1}$, and r_ℓ obtained via a hyperresolution step with $r_{\ell+1}$ as main premise and a set $\mathcal{Q}_\ell \subseteq \mathcal{P}^+_{\mathcal{N},\ell}$ of side premises. We argue that $T_{r_0}(D) \subseteq T_{r_{\ell+1}}(T_{Q_\ell}(\ldots T_{Q_0}(D)))$ by induction on ℓ. The base case holds directly by soundness of hyperresolution. For the induction step, assume that the claim holds for $\ell-1$. By soundness of hyperresolution, $T_{r_\ell}(D') \subseteq T_{r_{\ell+1}}(T_{\mathcal{Q}_\ell}(D'))$ for any dataset D'. Let $D' = T_{\mathcal{Q}_{\ell-1}}(\ldots T_{\mathcal{Q}_0}(D))$. We then have that $T_{r_\ell}(T_{\mathcal{Q}_{\ell-1}}(\ldots T_{\mathcal{Q}_0}(D)) \subseteq T_{r_{\ell+1}}(T_{\mathcal{Q}_\ell}(\ldots T_{\mathcal{Q}_0}(D))$. By applying the induction hypothesis on the left-hand side, we obtain the desired expression. This concludes the inductive argument. Using our expression for $L+1$, we have that $T_{r_0}(D) \subseteq T_{r_{L+2}}(T_{\mathcal{Q}_{L+1}}(\ldots T_{\mathcal{Q}_0}(D)))$. However, recall that $r_{L+2} \in \mathcal{P}^+_{\mathcal{N},L+1}$ and $\mathcal{Q}_\ell \subseteq \mathcal{P}^+_{\mathcal{N},\ell}$ for each ℓ. Thus, monotonicity under homomorphisms ensures $T_{r_0}(D) \subseteq T^{L+2}_{\mathcal{P}'_{\mathcal{N}}}(D)$. Since r_0 is an arbitrary rule of $\mathcal{P}'_{\mathcal{N}}$, we have $T_{\mathcal{P}'_{\mathcal{N}}}(D) \subseteq T^{L+2}_{\mathcal{P}'_{\mathcal{N}}}(D)$, as required. ☐

5 Explanation of Individual Facts

The results of Section 3.2 suggest a method for computing the equivalent program $\mathcal{P}_{\mathcal{N}}$ for a given (Col, δ)-GNN $\mathcal{N}$. First, initialise $\mathcal{P}_{\mathcal{N}}$ as $\mathcal{P}_{\mathcal{N},0}$ and each $\mathcal{X}_{0,i}$ as $\{0, 1\}$; then, for each layer ℓ, enumerate all combinations of vectors $\mathbf{x}$ and $\{\mathbf{y}^c\}_{c\in\mathsf{Col}}$ with jth component in $\mathcal{X}_{\ell-1,j}$, and use them to compute $\mathcal{X}_{\ell,i}$ with equation (4) as well as to enumerate the rules of $\mathcal{P}_{\mathcal{N},\ell}$ and add them to $\mathcal{P}_{\mathcal{N}}$. This approach is, however, impractical for large $\mathcal{X}_{\ell,i}$, in which case $\mathcal{P}_{\mathcal{N}}$ contains a large number of rules which become difficult to enumerate.

We address this practical challenge by presenting an algorithm for directly extracting a (typically small) subset of $\mathcal{P}_{\mathcal{N}}$ that suffices to derive a specific unary fact from the output of $T_{\mathcal{N}}$ for a given dataset. Thus, we can explain predictions of the GNN without having to extract the full equivalent program $\mathcal{P}_{\mathcal{N}}$. We focus on unary facts since $T_{\mathcal{N}}$ neither introduces nor removes binary facts from the input, so it is obvious why each binary fact appears in the output of $T_{\mathcal{N}}$.

Algorithm 1 takes as input a (Col, δ)-GNN $\mathcal{N}$, a dataset D, and a fact $U_i(a) \in T_{\mathcal{N}}(D)$, and yields a program $\mathcal{P}_{\mathcal{N},D,U_i(a)} \subseteq \mathcal{P}_{\mathcal{N}}$ such that $U_i(a) \in T^{L+2}_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$. The first step is to apply $\mathcal{N}$ to the canonical encoding of D while storing all feature vectors generated for each layer and vertex. The output program is initialised in line 2 with the unique rule $U_{L,i,\alpha}(x) \to U_i(x)$ from $\mathcal{P}_{\mathcal{N}}$ that may derive $U_i(a)$. Substitution θ initially maps x to a (line 3) and will be extended to include all variables in $\mathcal{P}_{\mathcal{N},D,U_i(a)}$. Substitution θ' is an auxiliary variable that is used to update θ.

The outer loop iterates through layers ℓ in descending order and each variable y in the domain of θ (line 4). Each iteration extends the program with δ_ℓ rules with different heads representing the elements of the feature vector for $v^{y\theta}$ in layer ℓ but with the same body Γ describing the neighbourhood of $v^{y\theta}$ in layer $\ell-1$. The body Γ is initialised with $\delta_{\ell-1}$ atoms describing the feature vector for $v^{y\theta}$ in layer $\ell-1$ (line 5). Then, for each colour c and position $k \in \{1, \ldots, \delta_{\ell-1}\}$ (line 6), the algorithm searches for a c-neighbour v^b of $v^{y\theta}$ contributing to the kth value of max aggregation over c-neighbours of $v^{y\theta}$ in layer $\ell-1$ (line 8). If found (line 9), then it introduces a fresh variable z mapped to b (line 10), and extends Γ with atoms $W^c(y, z)$ and $U_{\ell-1,k,(\mathbf{v}^b_{\ell-1})_k}(z)$ describing the contribution (line 11). It also adds rule $W^c(y, z) \to W^c(y, z)$ to $\mathcal{P}_{\mathcal{N},D,U_i(a)}$, ensuring that fact $W^c(y\theta, b)$, stating that v^b is a c-neighbour of $v^{y\theta}$, is preserved in each application of $T_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}$ (line 12). Finally, it extends Γ with negated atoms $\mathsf{not}\ V_{c,\ell-1,k,\gamma}(x)$ for each $\gamma \in \mathcal{X}_{\ell-1,k}$ greater than β, ensuring that no other neighbours of $v^{y\theta}$ may contribute with a value γ (line 13) to position k of the max aggregation over colour c (line 14). If no neighbour v^b is found for position $\delta_{\ell-1}$ (and thus, no neighbour was found for any other position), the algorithm extends Γ with negated body atom not $W(y, z)$ for fresh z (line 16). Having defined Γ, the algorithm iterates through positions $j \leq \delta_\ell$ (line 17) and adds a rule with body Γ and head predicate representing the jth value of the vector for $v^{y\theta}$ in ℓ (line 18). Upon completion of the outer loop, the algorithm adds all the rules from $\mathcal{P}_{\mathcal{N},0}$ that define predicates in $\mathcal{P}_{\mathcal{N},D,U_i(a)}$ (line 21) and returns $\mathcal{P}_{\mathcal{N},D,U_i(a)}$.

Algorithm 1 Extracting a subset of $\mathcal{P}_\mathcal{N}$ to explain a fact

```
    Input:  $\mathcal{N}$, a max $(\mathsf{Col}, \delta)$-GNN of $L$ layers;
            $D$, a $(\mathsf{Col}, \delta)$-dataset; $U_i(a)$, a fact in $T_\mathcal{N}(D)$;
 1: compute $\mathcal{N}(\mathsf{enc}(D))$ storing each $\mathbf{v}_\ell$ for each vertex $v$
 2: $\mathcal{P}_{\mathcal{N},D,U_i(a)} := U_{L,i,(\mathbf{v}^a_L)_i}(x) \to U_i(x)$
 3: $\theta(x) := \theta'(x) := a$
 4: for $\ell$ from $L$ to 1 and each $y$ in the domain of $\theta$ do
 5:     $\Gamma := \bigwedge_{k=1}^{\delta_{\ell-1}} U_{\ell-1,k,(\mathbf{v}^{y\theta}_{\ell-1})_k}(y)$
 6:     for $c \in \mathsf{Col}$ do
 7:         for $k \in \{1, \cdots, \delta_{\ell-1}\}$ do
 8:             $b := \mathsf{argmax}\{(\mathbf{v}_{\ell-1})^b_k \mid b : E^c(y\theta, b) \in D\}$
 9:             if $b$ is defined then
10:                 $\theta'(z) := b$ for $z$ fresh
11:                 $\Gamma := \Gamma \wedge W^c(y, z) \wedge U_{\ell-1,k,(\mathbf{v}_{\ell-1})^b_k}(z)$
12:                 add $W^c(y, z) \to W^c(y, z)$ to $\mathcal{P}_{\mathcal{N},D,U_i(a)}$
13:                 for $\gamma \in \mathcal{X}_{\ell-1,k}$ s. t. $\gamma > (\mathbf{v}_{\ell-1})^b_k$ do
14:                     $\Gamma := \Gamma \wedge \mathsf{not}\ V_{c,\ell-1,k,\gamma}(y)$
15:         if $b$ is undefined then
16:             $\Gamma := \Gamma \wedge \mathsf{not}\ W^c(y, z)$ for $z$ fresh
17:     for $j \in \{1, \ldots, \delta_\ell\}$ do
18:         add $\Gamma \to U_{\ell,i,(\mathbf{v}^{y\theta}_\ell)_j}(y)$ to $\mathcal{P}_{\mathcal{N},D,U_i(a)}$
19:     $\theta := \theta'$
20: for $r \in \mathcal{P}_{\mathcal{N},0}$ with head predicate in $\mathcal{P}_{\mathcal{N},D,U_i(a)}$ do
21:     $\mathcal{P}_{\mathcal{N},D,U_i(a)} := \mathcal{P}_{\mathcal{N},D,U_i(a)} \cup \{r\}$
    return $\mathcal{P}_{\mathcal{N},D,U_i(a)}$
```

Theorem 11. *For a max (Col, δ)-GNN $\mathcal{N}$, a dataset D, and a fact $U_i(a) \in T_\mathcal{N}(D)$, Algorithm 1 yields a program $\mathcal{P}_{\mathcal{N},D,U_i(a)} \subseteq \mathcal{P}_\mathcal{N}$ such that $U_i(a) \in T^{L+2}_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$.*

Proof. Lemma 2 and the algorithm's construction ensure that $\mathcal{P}_{\mathcal{N},D,U_i(a)} \subseteq \mathcal{P}_\mathcal{N}$, modulo variable renaming. It remains to show that $U_i(a) \in T^{L+2}_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$.

Let θ be the substitution constructed by the algorithm. For each $1 \leq \ell \leq L$, let D_ℓ be the dataset containing the grounding by θ of each positive body atom in a rule added in the iteration of the loop in line 4 for ℓ, and let $D_{L+1} = \{U_{L,i,(\mathbf{v}^a_L)_i}(a)\}$. We now prove by induction that $D_\ell \subseteq T^\ell_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$ for each $1 \leq \ell \leq L + 1$. For the base case, $\ell = 1$, consider an arbitrary $U_{0,k,1}(b) \in D_1$. By line 11, we have that $(\mathbf{v}^b_0)_k = 1$, and hence $U_k(b) \in D$. Line 21 ensures that rule $U_k(x) \to U_{0,k,1}(x)$ is in $\mathcal{P}_{\mathcal{N},D,U_i(a)}$; thus, we have $U_{0,k,1}(b) \in T_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$. For facts of the form $U_{0,k,0}(b) \in D_1$, an analogous argument gives us $(\mathbf{v}^b_0)_k = 0$. Since b appears in some unary or binary fact of D, line 21 ensures that there is a rule of the form (6), (7), or (8) in $\mathcal{P}_{\mathcal{N},D,U_i(a)}$ such that its positive atom is grounded in D mapping x to b, but without grounding the negated $U_k(x)$ in D since this would imply $(\mathbf{v}^b_0)_k = 1$, contradicting our previous claim. Thus, the rule fires and $U_{0,k,0}(b) \in T_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$. The proof for binary facts in D_1 is analogous. For the induction step, consider $2 \leq \ell \leq L + 1$; we assume that $D_{\ell-1} \subseteq T^{\ell-1}_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$ and prove $D_\ell \subseteq T^\ell_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$. Consider an arbitrary unary fact in D_ℓ, which must be of the form $U_{\ell-1,k,\alpha}(b)$ for $\alpha = (\mathbf{v}^b_{\ell-1})_k$, according to lines 2 or 11 (depending on whether $\ell = L + 1$ or $\ell < L + 1$, respectively); furthermore, there must exist some variable y such that $\theta(y) = b$ after the loop for layer ℓ. Thus, variable y is considered in the loop for layer $\ell - 1$, and so the algorithm introduces a rule $\Gamma \to U_{\ell,j,\alpha}(y)$ in layer 18. By definition, dataset $D_{\ell-1}$ contains the grounding via θ of all positive body atoms in Γ; hence, by induction hypothesis, so does $T^{\ell-1}_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$. We now argue by contradiction that θ cannot ground a negated body atom of this rule in $T^{\ell-1}_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$. If it is of the form not $V_{c,\ell-1,k,\gamma}(y)$, then we cannot ground it in $T^{\ell-1}_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$ since this predicate does not appear in a rule head in $\mathcal{P}_{\mathcal{N},D,U_i(a)}$. If it is of the form not $W^c(y, z)$, we have a contradiction as grounding the atom would imply that there is a c-neighbour $v^{z\theta}$ of $v^{y\theta}$, but then the condition in line 15 would not have been triggered, and so not $W^c(y, z)$ would not be in Γ. Hence, rule $\Gamma \to U_{\ell,j,\alpha}(y)$ fires via θ and so $U_{\ell,j,\alpha}(b) \in T^\ell_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$. The proof for binary atoms in D_ℓ is analogous. This concludes the proof by induction. We have proved, in particular, that $U_{L,i,(\mathbf{v}^a_L)_i}(a) \in T^{L+1}_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$, and so $U_i(a) \in T^{L+2}_{\mathcal{P}_{\mathcal{N},D,U_i(a)}}(D)$ by the rule in line 2. ☐

6 Related Work

There has been significant interest in the extraction of logical rules from ML models. Early approaches focused on the extraction of propositional rules from trained feed-forward neural networks (Andrews, Diederich, and Tickle 1995). Neural-LP (Yang, Yang, and Cohen 2017), DRUM (Sadeghian et al. 2019) and related approaches (Liu et al. 2023; Qu et al. 2021; Xiong, Hoang, and Wang 2017; Das et al. 2018; Omran, Wang, and Wang 2018; Zhang et al. 2019) provide means for extracting chain rules of the form $R_1(x_0, x_1) \wedge R_2(x_1, x_2) \wedge \ldots \wedge R_n(x_{n-1}, x_n) \to R(x_1, x_n)$ from different classes of models. A limitation of these approaches, however, is that the extracted program is not equivalent to the model; that is, predictions made by the model differ from the results of rule application. These limitations have been studied in recent work (Tena Cucala, Cuenca Grau, and Motik 2022; Wang et al. 2024).

Rule mining approaches focus on heuristically identifying patterns in data, which are then lifted as rules. These

		Precision				Recall				Accuracy				F1 Score				AUC				Training (s)		
		Max	Sum	Anb	Drm	Max	Sum	Anb	Drm	Max	Sum	Anb	Drm	Max	Sum	Anb	Drm	Max	Sum	Anb	Drm	Max	Sum	Drm
FB15K-237	v1	97.7	98.8	**100**	82.2	**43.0**	41.0	25.6	40.5	**71.0**	70.3	62.9	65.9	**59.7**	58.0	41.1	54.2	65.7	65.5	62.0	**68.6**	2,118	1,006	15,540
	v2	99.5	**100**	**100**	85.3	48.2	**48.7**	47.9	48.5	74.0	**74.3**	74.0	70.0	65.0	**65.5**	64.5	61.9	72.9	75.5	73.8	**76.0**	2,795	1,658	19,080
	v3	99.3	99.4	**100**	85.6	36.2	**71.4**	43.6	45.2	68.0	**85.5**	71.2	68.8	53.0	**83.1**	60.7	59.2	64.4	**87.5**	71.7	73.3	14,064	316	37,776
	v4	98.5	98.6	**100**	88.8	**48.8**	48.3	46.0	41.6	**74.0**	73.8	73.0	68.2	**65.2**	64.8	63.0	56.7	71.4	**75.7**	73.0	73.7	14,101	10,287	86,991
NELL-995	v1	33.3	**100**	97.5	94.7	1.2	17.6	**77.0**	18.0	49.4	58.9	**87.5**	58.5	2.3	30.0	**86.0**	30.3	3.4	59.9	**85.7**	52.6	3,647	79	402
	v2	98.8	99.0	**100**	80.7	36.4	41.4	**53.1**	41.2	67.9	70.5	**76.6**	65.7	53.2	58.4	**69.4**	54.5	57.3	69.4	**76.3**	74.6	707	183	6,354
	v3	56.7	80.0	**100**	85.8	**76.2**	42.1	47.3	45.6	59.0	65.8	**73.7**	69.0	**65.0**	55.1	64.2	59.6	**78.7**	64.3	69.3	75.3	1,155	1,814	36,090
	v4	93.8	97.3	**100**	80.3	**59.7**	46.3	44.3	26.8	**77.9**	72.5	72.2	60.1	**73.0**	62.8	61.4	40.2	**75.2**	72.8	71.6	67.1	583	358	3,990
WN18RR	v1	97.3	97.8	**99.1**	97.9	**86.7**	82.4	58.5	73.4	**92.1**	90.3	79.0	85.9	**91.7**	89.5	73.6	83.4	91.1	90.7	78.8	**92.5**	290	132	312
	v2	**100**	**100**	**100**	96.3	**82.5**	81.7	75.7	69.8	**91.2**	90.9	87.9	83.6	**90.4**	89.9	86.2	80.9	89.2	**90.0**	59.8	87.4	422	575	2,856
	v3	**100**	98.4	99.7	91.3	57.4	58.3	48.4	**59.7**	**78.7**	**78.7**	74.1	77.0	72.9	**73.2**	65.2	72.2	71.6	**73.2**	60.6	85.2	2,381	594	9,984
	v4	98.9	98.6	**99.9**	98.3	81.8	**81.9**	71.6	65.6	**90.5**	90.3	85.8	82.2	**89.5**	89.4	83.4	78.7	87.8	**88.4**	59.0	93.9	209	171	1,638

Table 1: Results for max GNNs (Max), sum GNNs (Sum), AnyBURL (Anb), and DRUM (Drm) in percentages

approaches do not involve training a ML model prior to rule generation. Prominent such approaches include AMIE+ (Galárraga et al. 2015), AnyBURL (Meilicke et al. 2019) as well as (Ahmadi et al. 2020; Gu, Guan, and Missier 2020).

Inductive Logic Programming (ILP) is a traditional approach where rules are generated from data and examples of positive and negative inferences so that the application of the output rules yields all the positive examples and none of the negative ones (Muggleton 1991; Cropper and Dumancic 2022). ILP techniques aiming to satisfy these requirements exactly are, however, sensitive to inconsistencies in the input examples (Cropper and Muggleton 2014; Si et al. 2019; Raghothaman et al. 2020). Other ILP systems achieve robustness to noise by interpreting the ILP task as a binary classification problem and providing a differentiable implementation of deduction (Evans and Grefenstette 2018). These techniques focus on learning expressive rules from small datasets; their application to KGs needs investigation.

Explanation methods for GNNs identify relevant parts of the input graph (Ying et al. 2019; Luo et al. 2020; Lin, Lan, and Li 2021). They are agnostic to the specifics of the model, but do not generate logical proofs. Characterising the expressivity of GNNs has also gained importance, and we have already mentioned many relevant works (Barceló et al. 2020; Huang et al. 2023; Pflueger, Tena Cucala, and Kostylev 2024; Grohe 2023). Morris et al. (2019) showed that GNNs can express certain graph isomorphism tests.

7 Evaluation

We have implemented the max GNN model using Python 3.8.5 and Pytorch Geometric v1.7.1. All experiments have been performed on a MacBook Pro with an M2 processor and 8GB of RAM, running macOS Ventura 13.3.

KG Completion We consider the task of of knowledge graph (KG) completion. When seen as a classification problem, the aim of KG completion is to learn a Boolean function $f(\cdot, \cdot)$ taking as input an incomplete dataset D and a fact α over a fixed set of unary and binary predicates and accepting iff α holds in the (unknown) completion D' of D containing all missing facts. We consider inductive KG completion, where the test data may mention constants not occurring in the training or validation datasets (Teru, Denis, and Hamilton 2020; Liu et al. 2021).

The canonical transformation $T_{\mathcal{N}}$ induced by a max GNN can only derive new unary facts, while KG completion requires also the derivation of binary facts. To address this limitation, we use an alternative encoding/decoding scheme (Liu et al. 2021; Tena Cucala et al. 2022) where binary facts are also encoded in feature vector components and edges in the graph correspond to different types of possible joins between unary and binary atoms. As shown in (Tena Cucala et al. 2023), such scheme can be captured by fixed encoding and decoding programs $\mathcal{P}_{\mathsf{enc}}$ and $\mathcal{P}_{\mathsf{dec}}$ so that the overall transformation is given by $T_{\mathcal{P}_{\mathsf{dec}}}(T_{\mathcal{N}}(T_{\mathcal{P}_{\mathsf{enc}}}(D)))$, which in turn coincides with $T_{\mathcal{P}_{\mathsf{dec}}}(T^{L+2}_{\mathcal{P}_{\mathcal{N}}}(T_{\mathcal{P}_{\mathsf{enc}}}(D)))$ by Theorem 7.

Benchmarks, metrics, and baselines We used the inductive KG completion benchmarks by Teru, Denis, and Hamilton (2020), based on the FB15K-237 (Bordes et al. 2013), NELL-995 (Xiong, Hoang, and Wang 2017), and WN18RR (Dettmers et al. 2018) KGs. Each benchmark provides disjoint datasets for training, validation, and testing, and a method for randomly splitting the test data into the incomplete dataset and a set of missing facts, which act as positive test examples. Additionally, we sampled an equal number of negative examples using the method in (Liu et al. 2021). We used standard classification metrics: precision, recall, accuracy, and F1 score, and area under the precision-recall curve (AUC) with thresholds in the range $[0.01, 0.99]$.

We compared our approach with DRUM and AnyBURL, state-of-the-art approaches for inductive KG completion that enable extraction of chain Datalog rules associated with a confidence score. We also considered a *sum GNN*, defined analogously to a max GNN but using sum aggregation. This is a type of *Relational Convolutional Graph Network (RGCN)* (Schlichtkrull et al. 2018), a widely used model.

Training and evaluation results We trained an instance of each model for each benchmark. We considered GNNs with two layers, with the dimension of the hidden layer twice that of the input layer. The training dataset was first split randomly with a 9:1 ratio into an incomplete dataset and a set of facts that should hold in its completion. We used cross-entropy loss, and we trained our models using Adam optimisation with standard learning rate (0.01) and weight decay (5×10^{-4}), and a maximum of $50,000$ epochs. For DRUM and AnyBURL we used their public code and their default configurations. Training times are shown in Table 1. Times

		# of rules in the program	extraction time (s)
FB15K-237	v1	1,127	23
	v2	1,015	27
	v3	1,054	32
	v4	1,071	37
NELL-995	v1	85	8
	v2	493	14
	v3	747	20
	v4	469	11
WN18RR	v1	54	8
	v2	64	9
	v3	77	10
	v4	48	9

Table 2: Rule extraction for individual facts.

for max GNN and sum GNN were comparable, but considerably shorter than those for DRUM. No times are shown for AnyBURL since this system does not train a model.

Table 1 shows the results of our evaluation. Max GNNs achieve comparable performance to that of other systems, attaining the highest accuracy and F1 score in 7 out of 12 benchmarks; in comparison, AnyBURL and sum GNNs achieved the highest accuracy (resp. F1 score) in 3 benchmarks each (3 and 2 benchmarks, respectively). AUC results were more balanced: each system achieved the best AUC in at least 2 and at most 4 of the benchmarks.

For each benchmark, we attempted to extract the full program $\mathcal{P}_{\mathcal{N}}$ defined as in Section 3.2 for a trained max GNN $\mathcal{N}$. As expected, this proved challenging due to the large number of generated rules. We also evaluated our implementation of Algorithm 1 for explaining individual facts. As shown in Table 2, the number of rules was highest for FB15K-237 benchmarks, and lowest for models trained on WN18RR benchmarks. This matches our expectations since the size of the rule sets generated by Algorithm 1 is proportional to the model dimensions, which in turn depends on the number of predicates in the dataset, and we note that FB15K-237 benchmarks have almost 20 times as many predicates as WN18RR benchmarks. Unlike $\mathcal{P}_{\mathcal{N}}$, programs explaining individual facts can be efficiently extracted.

8 Conclusion

We have shown the equivalence between max GNNs applied to relational data and Datalog programs with negation-as-failure. We have also shown that max GNNs can be trained in practice and achieve good performance in KG completion tasks. For future work, we plan to extend our results to GNNs using sum aggregation and attention mechanisms, and to develop approximation techniques for extracting rule sets with weaker semantic guarantees. A limitation of our approach is that the programs extracted to explain individual facts still contain a large number of rules, which can make them difficult to interpret. To address this, we will aim to develop techniques for extracting more concise explanations using optimised extraction algorithms, more expressive rules, and rules with weaker semantic guarantees.

Acknowledgements

This work was supported by the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project number 237889), and the EPSRC projects ConCur (EP/V050869/1), UK FIRES (EP/S019111/1), and AnaLOG (EP/P025943/1). 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.

References

Ahmadi, N.; Huynh, V.; Meduri, V. V.; Ortona, S.; and Papotti, P. 2020. Mining Expressive Rules in Knowledge Graphs. *ACM J. Data Inf. Qual.* 12(2):8:1–8:27.

Andrews, R.; Diederich, J.; and Tickle, A. B. 1995. Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowl. Based Syst.* 8(6):373–389.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition.

Barceló, P.; Kostylev, E. V.; Monet, M.; Pérez, J.; Reutter, J. L.; and Silva, J. P. 2020. The Logical Expressiveness of Graph Neural Networks. In *ICLR*. OpenReview.net.

Benedikt, M.; Lu, C.; Motik, B.; and Tan, T. 2024. Decidability of Graph Neural Networks via Logical Characterizations. In *51st International Colloquium on Automata, Languages, and Programming*, volume 297, 127:1–127:20.

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Proc. NeurIPS*, 2787–2795.

Cropper, A., and Dumancic, S. 2022. Inductive Logic Programming At 30: A New Introduction. *J. Artif. Intell. Res.* 74:765–850.

Cropper, A., and Muggleton, S. H. 2014. Logical Minimisation of Meta-Rules Within Meta-Interpretive Learning. In *ILP*, volume 9046 of *LNCS*, 62–75. Springer.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and Expressive Power of Logic Programming. *ACM Comput. Surv.* 33(3):374–425.

Das, R.; Dhuliawala, S.; Zaheer, M.; Vilnis, L.; Durugkar, I.; Krishnamurthy, A.; Smola, A.; and McCallum, A. 2018. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. In *Proc. ICLR*. OpenReview.net.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proc. AAAI*, 1811–1818. AAAI Press.

Evans, R., and Grefenstette, E. 2018. Learning Explanatory Rules from Noisy Data. *JAIR* 61:1–64.

Galárraga, L.; Teflioudi, C.; Hose, K.; and Suchanek, F. M. 2015. Fast Rule Mining in Ontological Knowledge Bases with AMIE+. *VLDB J.* 24(6):707–730.

Grohe, M. 2023. The Descriptive Complexity of Graph Neural Networks. In *Proc. LICS*, 1–14. IEEE.

Gu, Y.; Guan, Y.; and Missier, P. 2020. Efficient Rule Learning with Template Saturation for Knowledge Graph Completion. *CoRR* abs/2003.06071.

Huang, X.; Orth, M. A. R.; Ceylan, İ. İ.; and Barceló, P. 2023. A Theory of Link Prediction via Relational Weisfeiler-Leman. *CoRR* abs/2302.02209.

Ioannidis, V. N.; Marques, A. G.; and Giannakis, G. B. 2019. A Recurrent Graph Neural Network for Multi-relational Data. In *Proc. ICASSP*, 8157–8161. IEEE.

Kipf, T. N., and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. ICLR*. OpenReview.net.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proc. AAAI*, 2181–2187. AAAI Press.

Lin, W.; Lan, H.; and Li, B. 2021. Generative Causal Explanations for Graph Neural Networks. In *Proc. ICML*, volume 139, 6666–6679.

Liu, S.; Cuenca Grau, B.; Horrocks, I.; and Kostylev, E. V. 2021. INDIGO: GNN-Based Inductive Knowledge Graph Completion Using Pair-Wise Encoding. In *Proc. NeurIPS*, 2034–2045.

Liu, T.; Lv, Q.; Wang, J.; Yang, S.; and Chen, H. 2023. Learning Rule-Induced Subgraph Representations for Inductive Relation Prediction. In *Advances in Neural Information Processing Systems*, volume 36, 3517–3535. Curran Associates, Inc.

Luo, D.; Cheng, W.; Xu, D.; Yu, W.; Zong, B.; Chen, H.; and Zhang, X. 2020. Parameterized Explainer for Graph Neural Network. In *Proc. NeurIPS*.

Meilicke, C.; Chekol, M. W.; Ruffinelli, D.; and Stuckenschmidt, H. 2019. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *Proc. IJCAI*, 3137–3143. AAAI Press.

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

Muggleton, S. 1991. Inductive Logic Programming. *New Gener. Comput.* 8(4):295–318.

Omran, P. G.; Wang, K.; and Wang, Z. 2018. Scalable Rule Learning via Learning Representation. In *Proc. IJCAI*, 2149–2155.

Pflueger, M.; Tena Cucala, D. J.; and Kostylev, E. V. 2022. GNNQ: A Neuro-Symbolic Approach to Query Answering over Incomplete Knowledge Graphs. In *Proc. ISWC*, volume 13489 of *Lecture Notes in Computer Science*, 481–497. Springer.

Pflueger, M.; Tena Cucala, D.; and Kostylev, E. V. 2024. Recurrent Graph Neural Networks and Their Connections to Bisimulation and Logic. In *Proc. AAAI*. AAAI Press.

Portisch, J., and Paulheim, H. 2022. The DLCC Node Classification Benchmark for Analyzing Knowledge Graph Embeddings. In *Proc. ISWC*, 592–609. Springer International Publishing.

Qu, M.; Bengio, Y.; and Tang, J. 2019. GMNN: Graph Markov Neural Networks. In *Proc. ICML*, volume 97, 5241–5250.

Qu, M.; Chen, J.; Xhonneux, L.-P. A. C.; Bengio, Y.; and Tang, J. 2021. RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs. In *Proc. ICLR*.

Raghothaman, M.; Mendelson, J.; Zhao, D.; Naik, M.; and Scholz, B. 2020. Provenance-Guided Synthesis of Datalog Programs. *Proc. ACM Program. Lang.* 4(62):1–27.

Robinson, J. A. 1974. Automatic Deduction with Hyper-Resolution. *International Journal of Computer Mathematics* 39(1):227–234.

Sadeghian, A.; Armandpour, M.; Ding, P.; and Wang, D. Z. 2019. DRUM: End-To-End differentiable rule mining on knowledge graphs. In *Proc. NeurIPS*, 15321–15331.

Schlichtkrull, M. S.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data with Graph Convolutional Networks. In *Proc. ESWC*, volume 10843 of *LNCS*, 593–607. Springer.

Si, X.; Raghothaman, M.; Heo, K.; and Naik, M. 2019. Synthesizing Datalog Programs using Numerical Relaxation. In *Proc. IJCAI*, 6117–6124. AAAI Press.

Tena Cucala, D.; Cuenca Grau, B.; Kostylev, E. V.; and Motik, B. 2022. Explainable GNN-Based Models over Knowledge Graphs. In *Proc. ICLR*.

Tena Cucala, D.; Cuenca Grau, B.; Motik, B.; and Kostylev, E. V. 2023. On the correspondence between monotonic max-sum gnns and datalog. In *Proc. KR*, 658–667.

Tena Cucala, D.; Cuenca Grau, B.; and Motik, B. 2022. Faithful Approaches to Rule Learning. In *Proc. KR*, 484–493.

Teru, K. K.; Denis, E.; and Hamilton, W. 2020. Inductive Relation Prediction by Subgraph Reasoning. In *Proc. ICML*, volume 119, 9448–9457. PMLR.

Tzimas, T. 2023. Algorithmic transparency and explainability under eu law. *European Public Law* 29(4):385–411.

Wang, X.; He, X.; Cao, Y.; Liu, M.; and Chua, T.-S. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 950–958. Association for Computing Machinery.

Wang, X.; Cucala, D. J. T.; Grau, B. C.; and Horrocks, I. 2024. Faithful Rule Extraction for Differentiable Rule Learning Models. In *Proc. ICLR*.

Wu, L.; Cui, P.; Pei, J.; and Zhao, L. 2023. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer.

Xiong, W.; Hoang, T.; and Wang, W. Y. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Proc. EMNLP*, 564–573. Association for Computational Linguistics.

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proc. ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, 40–48.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *Proc. NeurIPS*, 2319–2328.

Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. In *Proc. NeurIPS*, 9240–9251.

Zhang, M., and Chen, Y. 2018. Link Prediction Based on Graph Neural Networks. In *Proc. NeurIPS*, 5171–5181.

Zhang, W.; Paudel, B.; Wang, L.; Chen, J.; Zhu, H.; Zhang, W.; Bernstein, A.; and Chen, H. 2019. Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning. In *Proc. WWW*, 2366–2377.