



# GNNQ: A Neuro-Symbolic Approach to Query Answering over Incomplete Knowledge Graphs

Maximilian Pflueger<sup>1</sup>, David J. Tena Cucala<sup>1</sup>, and Egor V. Kostylev<sup>2</sup>

<sup>1</sup> University of Oxford, Oxford OX1 2JD, UK

{maximilian.pflueger,david.tena.cucala}@cs.ox.ac.uk

<sup>2</sup> University of Oslo, Problemveien 7, 0315 Oslo, Norway  
egork@ifi.uio.no

**Abstract.** Real-world knowledge graphs (KGs) are usually incomplete—that is, miss some facts representing valid information. So, when applied to such KGs, standard symbolic query engines fail to produce answers that are expected but not logically entailed by the KGs. To overcome this issue, state-of-the-art ML-based approaches first embed KGs and queries into a low-dimensional vector space, and then produce query answers based on the proximity of the candidate entity and the query embeddings in the embedding space. This allows embedding-based approaches to obtain expected answers that are not logically entailed. However, embedding-based approaches are not applicable in the inductive setting, where KG entities (i.e., constants) seen at runtime may differ from those seen during training. In this paper, we propose a novel neuro-symbolic approach to query answering over incomplete KGs applicable in the inductive setting. Our approach first symbolically augments the input KG with facts representing parts of the KG that match query fragments, and then applies a generalisation of the Relational Graph Convolutional Networks (RGCNs) to the augmented KG to produce the predicted query answers. We formally prove that, under reasonable assumptions, our approach can capture an approach based on vanilla RGCNs (and no KG augmentation) using a (often substantially) smaller number of layers. Finally, we empirically validate our theoretical findings by evaluating an implementation of our approach against the RGCN baseline on several dedicated benchmarks.

**Keywords:** Query answering · Knowledge graphs · Graph neural networks · Neuro-symbolic AI

## 1 Introduction

*Knowledge graphs* (KGs) are databases where information is represented as a collection of *entities* and *relations* between them [13], or, equivalently, as a set of (function-free) first-order facts. *Query answering* is a fundamental reasoning

task on KGs, which requires identifying all (tuples of) entities in a KG that satisfy a specific formal expression, called a *query*. For example, (conjunctive) query  $q(x) = \exists y_1, y_2. \text{almaMater}(x, y_1) \wedge \text{professorAt}(y_1, y_2)$  finds, in a KG, all the universities that are the alma maters of persons working as professors.

Queries can be answered over KGs using symbolic logic-based engines, such as SPARQL and Cypher [16]. This approach, however, is challenged by the problem that many real-life KGs are *incomplete*, in the sense that there are true facts missing in the KG that may be relevant for answering a particular query. For example, if a KG contains the fact  $\text{professorAt}(\text{edith}, \text{berkeley})$ , representing that Edith is a professor at UC Berkeley, but it is missing the fact  $\text{almaMater}(\text{melbourne}, \text{edith})$ , representing that the University of Melbourne is the alma mater of Edith, then *melbourne* will not be returned as an answer for the above query, even though this answer may be expected by the user.

*Query Embedding* (QE) approaches have been proposed as a way to overcome this limitation [4, 9, 11, 17, 18, 20]. QE approaches embed KGs and monadic conjunctive queries jointly in a low dimensional vector space, and then they evaluate the likelihood of candidate answers according to their distance to the query embedding in the embedding space. These methods can produce answers that may be of interest to the user, even if they correspond to parts of the KG that only partially match the query. However, to the best of our knowledge, existing QE approaches are only applicable in the *transductive* setting, where trained models can only process KGs that mention only entities seen during training. An increasing number of applications, however, require an *inductive* setting [10, 14, 23, 25], where unseen entities are also allowed.

*Relational Graph Convolutional Networks* (RGCNs) [19] are a class of graph neural networks (GNNs) which take as input directed labelled multigraphs—in particular, graphs with nodes connected by coloured edges and annotated with real-valued feature vectors. When applied to such a multigraph, an RGCN updates, in each layer, the feature vector of each node by combining, by means of learned parameters, the node’s feature vector in the previous layer with the previous-layer vectors of the node’s neighbours. If the vector in the final layer is a single Boolean value, then the RGCN can be seen as a (binary) node classifier. RGCNs can be used to answer monadic queries on a KG: first, encode the KG as a directed multigraph with a node for each entity in the KG; then, run a trained RGCN on the multigraph to predict whether each entity is an answer to the query or not (similar approaches have been used for the related problem of KG completion [10, 14, 22–24]). This method has three properties making it suitable for answering queries on incomplete KGs in an inductive setting.

1. *Inductive Capabilities*. RGCNs do not use entity-specific parameters, so they can be applied to KGs mentioning entities not seen during training.
2. *Expressivity*. Recent theoretical analysis of RGCNs [5] shows that, for every monadic tree-shaped conjunctive query, there exists an RGCN that exactly *captures* this query—that is, for each KG, the answers provided by the RGCN on the KG are the same as the real query answers over the KG.

3. *Noise Tolerance*. Similarly to other ML approaches, RGCNs can produce relevant query answers even if such answers do not have exact matches in the input KG (e.g., due to missing information).

A key limitation of using RGCNs for query answering over KGs, however, is that, in order to recognise a part of the KG relevant to a query answer, any RGCN requires at least as many layers as the length of the longest (simple) path in the query to an answer variable. Empirical results have shown, however, that GNNs with many layers often fail to learn long-range dependencies and suffer from several problems, such as over-smoothing [12]. This problem persists even if the input KGs have no missing information.

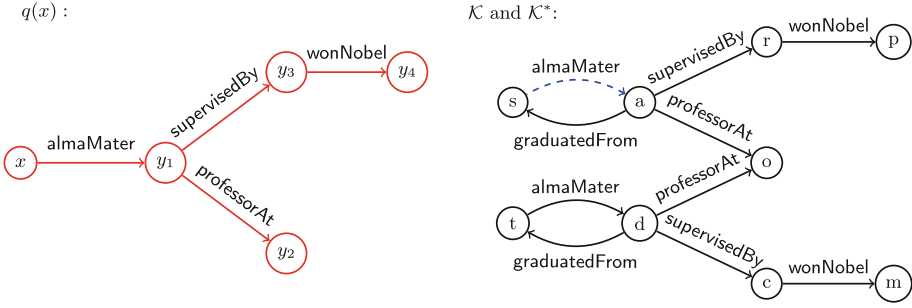
To address this limitation, we propose in this paper a novel neuro-symbolic approach to inductive query answering over incomplete KGs. Our approach first augments an input KG using a set of logical (i.e., symbolic) rules extracted from the query. The application of a rule to a KG adds new facts that represent (complete) parts of the KG matching connected query fragments. Then the approach encodes the augmented KG as a coloured *hypergraph*, and processes this hypergraph using a novel neural architecture called *Hyper-Relational Graph Convolutional Network (HRGCN)*, which generalises vanilla RGCNs to be applicable to coloured hypergraphs. We then provide a proof that, under mild and reasonable assumptions, our approach can emulate the baseline approach that relies on vanilla RGCNs (without KG augmentation) using significantly less layers. Finally, we present an implementation of our approach in a system called GNNQ and evaluate it on nine novel benchmarks for inductive query answering over incomplete KGs against a baseline without augmentation. Our results show that instances of GNNQ can be effectively trained and deployed in practice; moreover, they outperform the baselines, even if the latter use more layers.

## 2 Preliminaries

In this paper, we rely on a standard formalisation of knowledge graphs (and related concepts) in first-order logic.

Let us consider disjoint countable sets of *predicates*, *constants*, and *variables*, where each predicate is assigned a natural number called *arity*. A  $k$ -ary *atom*, with  $k \in \mathbb{N}$ , is an expression of the form  $P(\bar{t})$ , where  $P$  is a  $k$ -ary predicate and  $\bar{t} = t_1, \dots, t_k$  is a  $k$ -tuple of constants and variables. A *fact* is a variable-free atom. A *dataset* is a finite set of facts. A *knowledge graph (KG)* is a dataset containing only unary and binary facts. So, entities in a KG are represented by constants, while classes of entities and relations between them are represented by unary and binary facts, respectively. Let  $\text{Const}(\mathcal{D})$  and  $\text{Pred}(\mathcal{D})$  denote the constants and predicates mentioned in a dataset  $\mathcal{D}$ , respectively.

A *conjunctive query (CQ)* with (a tuple of) *answer variables*  $\bar{x}$ , is a formula  $q(\bar{x}) = \exists \bar{y}. \phi(\bar{x}, \bar{y})$ , where the *body*  $\phi(\bar{x}, \bar{y})$  is a conjunction of atoms over variables  $\bar{x}, \bar{y}$ . A tuple  $\bar{a}$  of constants is an *answer* to  $q(\bar{x})$  over a dataset  $\mathcal{D}$  if there is a *homomorphism* from  $q(\bar{a})$  to  $\mathcal{D}$ —that is, an assignment of constants to  $\bar{y}$  such that each atom in  $\phi(\bar{a}, h(\bar{y}))$  is in  $\mathcal{D}$ . Let  $q[\mathcal{D}]$  denote the set of all answers to



**Fig. 1.** Representation of tree-CQ  $q(x)$  and KG  $\mathcal{K}$  with completion  $\mathcal{K}^*$  from Example 1: the single completion fact is drawn as a dashed line, constants are shown by first letter.

$q(\bar{x})$  over  $\mathcal{D}$ . In this paper, we concentrate on *tree-shaped* CQs (*tree-CQs*)—that is, constant-free CQs over unary and binary predicates with one answer variable such that the primal pseudograph of the CQ’s body is a tree; here, the *primal pseudograph* of a conjunction of atoms is the undirected pseudograph whose nodes are the variables of the conjunction and which has an edge between (not necessary distinct)  $z_1$  and  $z_2$  for each binary atom  $R(z_1, z_2)$  in the conjunction. We call a primal pseudograph *primal tree* if it is a tree. The *height* of a tree-CQ is the height of its primal tree with the answer variable as the root.

### 3 Inductive Query Answering over Incomplete KGs

We are interested in the problem of finding the answers to a given (known in advance) tree-CQ over KGs that may be incomplete—that is, missing (relevant) information. In particular, we assume that each KG has a *completion*—that is, a larger (or identical) KG that may include additional facts, which are ‘missing’ in the original KG. We consider the setting where all the constants in the completion facts are already mentioned in the original KG. However, we assume that the function that maps a KG to its completion is unknown; instead, only partial knowledge about this function is provided to a system in the form of examples, each of which consists of a KG, a constant, and a Boolean value, which tells whether the constant is an answer to the tree-CQ over the completion of the KG. Finally, our setting is *inductive* [10, 14, 23, 25], which means that there exists a finite, known-in-advance set of predicates used in all KGs, their completions, and the tree-CQ, but the constants in different KGs may be different.

We are now ready to formalise the ML task of inductive tree-CQ answering over incomplete KGs, which we call the *IQA task* for brevity.

**Definition 1.** *Given a finite set  $\text{Pred}$  of unary and binary predicates, and a tree-CQ  $q(x)$  that uses only predicates from  $\text{Pred}$ , let us assume a hidden completion function  $\cdot^*$  mapping each KG  $\mathcal{K}$  with  $\text{Pred}(\mathcal{K}) \subseteq \text{Pred}$  to another KG*

$\mathcal{K}^*$  with  $\text{Pred}(\mathcal{K}^*) \subseteq \text{Pred}$ , called the completion of  $\mathcal{K}$ , such that  $\mathcal{K} \subseteq \mathcal{K}^*$  and  $\text{Const}(\mathcal{K}^*) = \text{Const}(\mathcal{K})$ . Then, the IQA task is to learn a function  $g_q$  mapping each KG  $\mathcal{K}$  with  $\text{Pred}(\mathcal{K}) \subseteq \text{Pred}$  to the set  $q[\mathcal{K}^*]$  of answers to  $q(x)$  over  $\mathcal{K}^*$ .

*Example 1.* Let  $q(x)$  be the tree-CQ

$$\exists y_1, y_2, y_3, y_4. \text{almaMater}(x, y_1) \wedge \text{professorAt}(y_1, y_2) \wedge \\ \text{supervisedBy}(y_1, y_3) \wedge \text{wonNobel}(y_3, y_4),$$

which asks for all universities that are the *alma maters* of professors who were supervised by Nobel Prize winners, and let  $\mathcal{K}$  be the KG

$$\{\text{supervisedBy}(\text{alice}, \text{roger}), \text{supervisedBy}(\text{daniel}, \text{carol}), \\ \text{wonNobel}(\text{roger}, \text{physics}), \text{wonNobel}(\text{carol}, \text{medicine}), \\ \text{professorAt}(\text{alice}, \text{oxford}), \text{professorAt}(\text{daniel}, \text{oxford}), \\ \text{graduatedFrom}(\text{alice}, \text{shanghai}), \text{graduatedFrom}(\text{daniel}, \text{toronto}), \\ \text{almaMater}(\text{toronto}, \text{daniel})\}$$

with  $\mathcal{K}^* = \mathcal{K} \cup \{\text{almaMater}(\text{shanghai}, \text{alice})\}$  (see Fig. 1). The desired function  $g_q$  for  $q(x)$  should return the set  $\{\text{shanghai}, \text{toronto}\}$  of answers when applied to  $\mathcal{K}$ , because both *toronto* and *shanghai* are answers to  $q(x)$  over  $\mathcal{K}^*$ . Note, however, that *shanghai* is not an answer to  $q(x)$  over  $\mathcal{K}$ , since the fact  $\text{almaMater}(\text{shanghai}, \text{alice})$  is missing from  $\mathcal{K}$ .

## 4 Neuro-Symbolic Approach to the IQA Task

In this section, we describe our approach for solving the IQA task. For the remainder of this section, let us fix a (possibly empty) set  $\text{Pred}_1 = \{A_1, \dots, A_m\}$  of unary predicates, a finite set  $\text{Pred}_2$  of binary predicates, and a tree-CQ  $q(x) = \exists \bar{y}. \phi(x, \bar{y})$  over predicates in  $\text{Pred}_1 \cup \text{Pred}_2$ . For technical reasons, we assume that the variables  $x, \bar{y}$  are ordered following a breadth-first traverse of the primal tree of  $\phi(x, \bar{y})$ . This assumption is without loss of generality, since given an arbitrary tree-CQ, we can always construct a semantically equivalent query that satisfies our requirement by reordering  $\bar{y}$ . Finally, for each  $R \in \text{Pred}_2$ , we consider a fresh binary predicate  $\bar{R}$ , which we call the *inverse* of  $R$ , and we let  $\text{Pred}_2^+$  denote the set  $\text{Pred}_2 \cup \{\bar{R} \mid R \in \text{Pred}_2\}$ .

Our approach is divided in three steps. In the first step, described in Sect. 4.1, the input KG is augmented with new facts that will assist our ML model in recognising parts of the input KG that match selected query fragments. In the second step, described in Sect. 4.2, our approach encodes the augmented KG into a data structure suitable for our ML model, namely, a coloured labelled (multi-)hypergraph, where nodes correspond to constants in the KG and edges to non-unary atoms. In the third and final step, described in Sect. 4.3, the approach processes the coloured hypergraph by means of a generalisation of RGCNs. The output of this process is a Boolean value for each node in the hypergraph, representing whether the constant associated to this node is predicted as an answer to  $q(x)$  over the completion of the input KG or not.

## 4.1 Augmentation of Knowledge Graphs

As discussed in the introduction, vanilla RGCNs with Boolean outputs can be used to solve the IQA task by first encoding the input KG as a directed multi-graph and then applying a trained RGCN to the encoding. Such RGCNs, however, may require a large number of layers to adequately capture the target function. However, training RGCNs with many layers is expensive; moreover, the resulting models may have poor performance due to over-smoothing [12]. To address these issues, our procedure first augments the input KG with facts representing (complete) parts of the KG matching query fragments. As we prove in Sect. 5, this allows us to solve the IQA task using significantly less layers.

The KG augmentation relies on a set of logical rules, which correspond to fragments of the tree-CQ. These rules are applied to the KG to infer new facts, which are added to the KG. To formalise this step, we need some terminology.

A (projection-free) *rule* is an expression of the form  $H(\bar{z}) \leftarrow \psi(\bar{z})$ , where the *head*  $H(\bar{z})$  is an atom over a  $|\bar{z}|$ -ary predicate  $H$ , and the *body*  $\psi(\bar{z})$  is a conjunction of atoms using variables  $\bar{z}$  (i.e., each variable in  $\bar{z}$  appears in at least one atom in  $\psi(\bar{z})$ , and there are no other variables in these atoms). The *application* of a set  $\mathcal{R}$  of rules to a dataset  $\mathcal{D}$  is a dataset  $\mathcal{R}(\mathcal{D})$  that extends  $\mathcal{D}$  with each fact  $H(\bar{a})$  such that there is a rule  $H(\bar{z}) \leftarrow \psi(\bar{z})$  in  $\mathcal{R}$  with every fact in  $\psi(\bar{a})$  belonging to  $\mathcal{D}$ . Note that in what follows we will only apply a rule to datasets that do not mention the head predicate of the rule.

Next, we associate a set  $\mathcal{R}_q$  of rules to our fixed tree-CQ  $q(x)$ . Specifically, we define  $\mathcal{R}_q$  as the set of all the rules  $H(\bar{z}) \leftarrow \psi(\bar{z})$ , where  $\psi(\bar{z})$  is a sub-conjunction of  $\phi(x, \bar{y})$  with the same order of variables in  $\bar{z}$  as their order in  $x, \bar{y}$ , such that

- the primal pseudograph of  $\psi(\bar{z})$  is connected (and hence it is a tree) and
- the height of this tree is at least 2,

and where  $H$  is a fresh  $|\bar{z}|$ -ary predicate uniquely associated to  $\psi(\bar{z})$ . Subsequently, we use  $\text{Pred}_q$  to denote the set of head predicates of the rules in  $\mathcal{R}_q$ . Note that, by our assumptions on the order of variables, the first variable in  $\bar{z}$  will always be the one closest to  $x$  in the primal tree of  $\phi(x, \bar{y})$  rooted at  $x$ . Moreover, the assumptions ensure that  $\mathcal{R}_q$  does not contain rules with the same body and head predicate, but different heads; this eliminates redundancy by preventing augmentation with multiple facts identifying the same sub-KGs.

As discussed in Sect. 6.1, in our experiments we observe that it is often better not to use all rules in  $\mathcal{R}_q$  in the augmentation step. We believe that there are two main reasons for this: first, increasing the number of augmentation facts appears to have diminishing returns, since different facts can represent similar parts of the input KG (satisfying similar query fragments); second, having a large number of augmentation facts mentioning the same constant can produce problems similar to over-smoothing. Therefore, we consider KG augmentations with full  $\mathcal{R}_q$  and augmentations with subsets of  $\mathcal{R}_q$ .

**Definition 2.** *The partial augmentation of a KG  $\mathcal{K}$  over  $\text{Pred}_1 \cup \text{Pred}_2$  for the tree-CQ  $q(x)$  with respect to rules  $\mathcal{R}'_q \subseteq \mathcal{R}_q$  is the dataset  $\mathcal{R}'_q(\mathcal{K})$ . The (full) augmentation of  $\mathcal{K}$  is the partial augmentation with respect to all  $\mathcal{R}_q$ .*

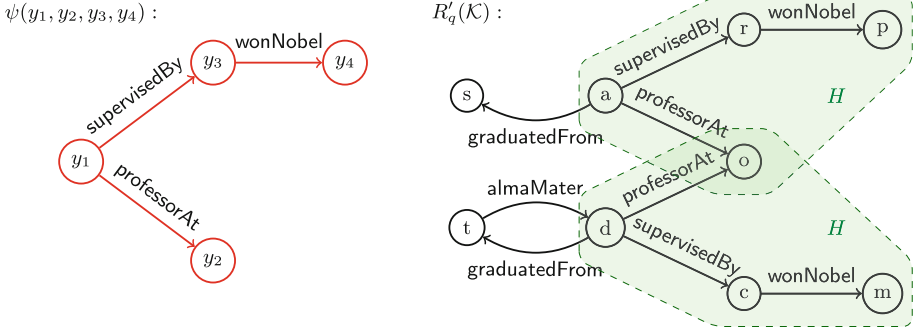


Fig. 2. Representation of  $\psi$ ,  $\mathcal{K}$ , and augmentation  $\mathcal{R}'_q(\mathcal{K})$  from Example 2

Example 2. Consider KG  $\mathcal{K}$ , tree-CQ  $q(x)$  in Example 1, and the sub-conjunction

$$\psi(y_1, y_2, y_3, y_4) = \text{professorAt}(y_1, y_2) \wedge \text{supervisedBy}(y_1, y_3) \wedge \text{wonNobel}(y_3, y_4),$$

of the body of  $q(x)$  (see Fig. 2); its primal pseudograph is a tree of height 3. So,  $\mathcal{R}_q$  contains rule  $r = H(y_1, y_2, y_3, y_4) \leftarrow \psi(y_1, y_2, y_3, y_4)$  for a fresh predicate  $H$ , and the partial augmentation of  $\mathcal{K}$  for  $q(x)$  with respect to  $\mathcal{R}'_q = \{r\}$  is  $\mathcal{K} \cup \{H(\text{alice}, \text{oxford}, \text{roger}, \text{physics}), H(\text{daniel}, \text{oxford}, \text{carol}, \text{medicine})\}$ .

### 4.2 Encoding of Knowledge Graphs

We now describe our encoding of datasets into directed (multi-)hypergraphs where hyperedges are coloured and nodes are labelled by real-valued vectors. Specifically, our encoding introduces a hypergraph node for each constant in the input dataset; then, each fact of arity greater than 1 is encoded into a hyperedge of the colour corresponding to the fact’s predicate, and each fact of arity 1 is encoded as a component of the feature vector labelling the corresponding node. Furthermore, for each binary fact in the original dataset with a predicate  $R$ , the encoding introduces, besides the  $R$ -coloured edge, an  $\bar{R}$ -coloured edge in the reverse direction; such edges will ensure that our ML model propagates information in both directions whenever a binary fact connects two constants.

**Definition 3.** Given a finite set  $\text{Col}$  of colours with fixed arities greater than 1 and a dimension  $\delta \in \mathbb{N}$ , a  $(\text{Col}, \delta)$ -hypergraph  $G$  is a triple  $(\mathcal{V}, \mathcal{E}, \lambda)$  where  $\mathcal{V}$  is a finite set of nodes,  $\mathcal{E}$  is a set of directed hyperedges of the form  $(v, c, (u_1, \dots, u_k))$  with  $c \in \text{Col}$  of arity  $k + 1$ ,  $\{v, u_1, \dots, u_k\} \subseteq \mathcal{V}$ , and  $\lambda$  is a labelling function that assigns a vector  $\lambda(v) \in \mathbb{R}^\delta$  to every  $v \in \mathcal{V}$ . Hypergraph  $G$  is Boolean if  $\delta = 1$  and  $\lambda(v) \in \{0, 1\}$  for every  $v \in \mathcal{V}$ .

Given a  $(\text{Col}, \delta)$ -hypergraph  $G = (\mathcal{V}, \mathcal{E}, \lambda)$ , we denote, for brevity, the vector  $\lambda(v)$  for a node  $v$  with  $\mathbf{v}$ , and we refer to its  $i^{\text{th}}$  element as  $(\mathbf{v})_i$ . Furthermore,



for each  $v \in \mathcal{V}$  and  $c \in \text{Col}$ , we define the  $c$ -neighbourhood  $N_G^c(v)$  of  $v$  in  $G$  as the set  $\{(u_1, \dots, u_k) \mid (v, c, (u_1, \dots, u_k)) \in \mathcal{E}\}$ .

Now that we have our target graph structure, we can define our encoding function, which maps datasets (including augmented KGs) to hypergraphs.

**Definition 4.** *The encoding of a dataset  $\mathcal{D}$  over predicates  $\text{Pred}_1 \cup \text{Pred}_2 \cup \text{Pred}_q$  is the  $(\text{Col}, \delta)$ -hypergraph  $G_{\mathcal{D}} = (\mathcal{V}, \mathcal{E}, \lambda)$  such that*

- $\text{Col} = \text{Pred}_2^+ \cup \text{Pred}_q$  (i.e., the binary predicates, their inverses, and the head predicates of  $\mathcal{R}_q$ ),
- $\delta = m + 1$  (where  $m$  is the number  $|\text{Pred}_1|$  of unary predicates),
- $\mathcal{V}$  is the set of constants in  $\mathcal{D}$ ,
- $\mathcal{E}$  includes  $(a, R, (b_1, \dots, b_k))$  for every  $R(a, b_1, \dots, b_k) \in \mathcal{D}$  with  $k > 1$ ,
- $\mathcal{E}$  includes  $(b, \bar{R}, (a))$  for every  $R(a, b) \in \mathcal{D}$  with  $R \in \text{Pred}_2$ , and
- $\lambda$  is the labelling that assigns, to each  $a \in \mathcal{V}$ , the vector  $\mathbf{a} \in \mathbb{R}^\delta$  such that  $(\mathbf{a})_i = 1$  if  $A_i(a) \in \mathcal{D}$  or  $i = m + 1$ , and  $(\mathbf{a})_i = 0$  otherwise.

Note that the  $(m + 1)^{\text{th}}$  element of each vector  $\mathbf{a}$  is always 1; this element is needed to cover the case  $m = 0$ —that is, when there are no unary predicates.

### 4.3 Hyper-Relational Graph Convolutional Networks

We now introduce a generalised version of the RGCN [19] architecture that can process  $(\text{Col}, \delta)$ -hypergraphs; we call this generalisation *Hyper-Relational Graph Convolutional Network (HRGCN)*. Our approach uses a HRGCN to process the hypergraphs that are encodings of augmented KGs.

**Definition 5.** *Given a finite set  $\text{Col}$  of colours with fixed arities and  $\delta \in \mathbb{N}$ , a  $(\text{Col}, \delta)$ -HRGCN  $\mathfrak{S}$  with  $L \geq 1$  layers and dimensions  $(\delta_0, \dots, \delta_L)$ , for  $\delta_0 = \delta$ , is*

$$(\{\text{Aggr}^\ell\}_{\ell=1}^L, \{\text{Comb}^\ell\}_{\ell=1}^L, \text{Cls}),$$

where

- each aggregation function  $\text{Aggr}^\ell$ ,  $1 \leq \ell \leq L$ , maps a multiset of tuples of the form  $(c, \mathbf{u}_1, \dots, \mathbf{u}_k)$  with  $c \in \text{Col}$  and each  $\mathbf{u}_i$  in  $\mathbb{R}^{\delta^{\ell-1}}$  to a vector in  $\mathbb{R}^{\delta^{\ell-1}}$ ;
- each combination function  $\text{Comb}^\ell$ ,  $1 \leq \ell \leq L$ , maps two vectors in  $\mathbb{R}^{\delta^{\ell-1}}$  to a vector in  $\mathbb{R}^{\delta^\ell}$ ;
- classification function  $\text{Cls}$  maps a vector in  $\mathbb{R}^{\delta^L}$  to a value in  $\{0, 1\}$ .

Given a  $(\text{Col}, \delta)$ -hypergraph  $G = (\mathcal{V}, \mathcal{E}, \lambda)$ , HRGCN  $\mathfrak{S}$  induces the sequence  $\lambda^0, \dots, \lambda^L$  of labellings such that  $\lambda^0 = \lambda$ , and, for each  $\ell \in \{1, \dots, L\}$  and  $v \in \mathcal{V}$ , the value of  $\lambda^\ell(v) = \mathbf{v}^\ell$  is defined as

$$\mathbf{v}^\ell = \text{Comb}^\ell \left( \mathbf{v}^{\ell-1}, \text{Aggr}^\ell \left( \{(c, \mathbf{u}_1^{\ell-1}, \dots, \mathbf{u}_k^{\ell-1}) \mid (u_1, \dots, u_k) \in N_G^c(v), c \in \text{Col}\} \right) \right).$$

The result  $\mathfrak{S}(G)$  of applying  $\mathfrak{S}$  to  $G$  is the  $(\text{Col}, \delta)$ -hypergraph  $(\mathcal{V}, \mathcal{E}, \lambda_{\text{bool}})$ , where  $\lambda_{\text{bool}}$  is the labelling of every node  $v \in \mathcal{V}$  by  $\text{Cls}(\mathbf{v}^L)$ . Subsequently,  $\mathfrak{S}(G, v)$  denotes  $\mathbf{v}^L$  and  $\mathfrak{S}_{\text{true}}[G]$  denotes the set of all  $v \in \mathcal{V}$  with  $\lambda_{\text{bool}}(v) = 1$ .

Then, a  $(\text{Col}, \delta)$ -RGCN is a  $(\text{Col}, \delta)$ -HRGCN for  $\text{Col}$  with no colours of arity greater than 2 (this is essentially the standard definition of RGCNs [19]).



## 5 Advantages of Knowledge Graph Augmentation

As discussed in the introduction, the main motivation for KG augmentation is to help ML models easily recognise parts of the input KG that match complete connected fragments of the query. In this section, we present a theorem that makes this conjecture precise. To this end, we assume a natural and broad class of completion functions, which arguably captures those one may expect to find in practice. Then, we will show that, for each (big enough) tree-CQ  $q$ , if there is an instance of our approach capturing the goal function  $g_q$  without using KG augmentation, then there exists an instance of the approach that also captures  $g_q$  using (full) KG augmentation, but whose HRGCN has significantly less layers.

**Definition 6.** A completion function  $\cdot^*$  over a set of predicates  $\text{Pred}$  is

- monotonic under homomorphisms if for every KGs  $\mathcal{K}_1$  and  $\mathcal{K}_2$  over  $\text{Pred}$  and each homomorphism  $h$  from  $\mathcal{K}_1$  to  $\mathcal{K}_2$ ,  $h$  is also a homomorphism from  $\mathcal{K}_1^*$  to  $\mathcal{K}_2^*$ , where a homomorphism  $h$  from  $\mathcal{K}_1$  to  $\mathcal{K}_2$  is a mapping from  $\text{Const}(\mathcal{K}_1)$  to constants such that  $h(\mathcal{K}_1) \subseteq \mathcal{K}_2$ ;
- $s$ -local, for  $s \in \mathbb{N}$ , if for every KG  $\mathcal{K}$  over  $\text{Pred}$  and every fact  $\alpha \in \mathcal{K}^*$  there is  $\mathcal{K}_\alpha \subseteq \mathcal{K}$  such that  $\alpha \in \mathcal{K}_\alpha^*$  and  $\mathcal{K}_\alpha$  contains an undirected path (through constants and binary facts) from each constant in  $\mathcal{K}_\alpha$  to each constant in  $\alpha$  of length at most  $s$ ;
- $k$ -incomplete for a tree-CQ  $q(x)$  if for each KG  $\mathcal{K}$  over  $\text{Pred}$  and each answer  $a \in q[\mathcal{K}^*]$  there is  $\mathcal{K}_a$  such that  $\mathcal{K} \subseteq \mathcal{K}_a \subseteq \mathcal{K}^*$ ,  $a \in q[\mathcal{K}_a]$ , and  $|\mathcal{K}_a \setminus \mathcal{K}| \leq k$ .

The intuition under these notions is as follows. Monotonicity under homomorphisms requires that every fact in the completion of a KG should also appear (in a suitable form) in the completion of any KG that has the same structure as the original KG. Locality reflects the intuition that every fact in the completion is a consequence of a small neighbourhood of the fact in the original KG. Finally, incompleteness for a query means that, for every answer to the query, only a small number of facts can be missing in any ‘witness’ of it—that is, any part of the KG completion (fully) matching the query. We will now state our main result; its proof can be found in the supplemental material.

**Theorem 1.** Let  $\text{Pred}_1$  and  $\text{Pred}_2$  be finite sets of unary and binary predicates, respectively, and let  $\text{Pred}_2^+ = \text{Pred}_2 \cup \{\bar{R} \mid R \in \text{Pred}_2\}$  and  $\text{Pred} = \text{Pred}_1 \cup \text{Pred}_2$ . Let  $q(x)$  be a tree-CQ of height  $h$  over  $\text{Pred}$  and  $\cdot^*$  be a completion function over  $\text{Pred}$  that is monotonic under homomorphisms,  $s$ -local, and  $k$ -incomplete for  $q(x)$ . If there is an  $L$ -layer  $(\text{Pred}_2^+, \delta)$ -RGCN  $\mathfrak{R}$ , for  $\delta \in \mathbb{N}$ , such that  $\mathfrak{R}_{\text{true}}[G_{\mathcal{K}}] = q[\mathcal{K}^*]$  for each KG  $\mathcal{K}$  over  $\text{Pred}$ , then there is a  $(k(s+1)+1)$ -layer  $(\text{Pred}_2^+ \cup \text{Pred}_q, \delta)$ -HRGCN  $\mathfrak{S}$  such that  $\mathfrak{S}_{\text{true}}[G_{\mathcal{R}_q(\mathcal{K})}] = q[\mathcal{K}^*]$  for each KG  $\mathcal{K}$  over  $\text{Pred}$ .

We emphasise that many completion functions that one may find in practice will have small values of  $s$  and  $k$ , thus making  $k(s+1)+1$  significantly smaller than  $L$ . Therefore (for large enough  $L$ ) KG augmentation allows us to reduce the number of layers that an HRGCN instance in our approach requires to capture the goal function  $g_q$ —that is, to capture query  $q$  on incomplete KGs.

**Table 1.** Benchmark statistics, where  $\|q\|$  and  $h(q)$  are the number of atoms and height of the tree-CQ, and ‘pos./neg.’ stands for ‘number of positive / negative examples’

Benchmark	Pred	$\ q\  / h(q)$	train: pos./neg	test: pos./neg.
WATDIV-Q1	158	8 / 4	2114 / 699699	1085 / 349877
WATDIV-Q2	158	8 / 3	3258 / 698396	1769 / 349119
WATDIV-Q3	158	8 / 3	1520 / 700276	798 / 350165
WATDIV-Q4	158	10 / 4	2397 / 698986	1226 / 349546
WATDIV-Q5	158	10 / 4	6338 / 693988	2866 / 347570
WATDIV-Q6	158	10 / 4	7545 / 692439	3744 / 346290
FB15k237-Q1	237	7 / 4	1185 / 1180	395 / 395
FB15k237-Q2	237	7 / 4	650 / 660	220 / 220
FB15k237-Q3	237	5 / 4	860 / 870	290 / 290

## 6 Implementation and Evaluation

We have implemented our approach to the IQA task over incomplete KGs using Python 3.8.10, RDFLIB 6.1.1, and PyTorch 1.11.0 in a system called GNNQ. We then evaluated several instances  $\text{GNNQ}_L$  of GNNQ using KG augmentation, parametrised by the number  $L$  of layers of the underlying HRGCN, on a number of benchmarks. To the best of our knowledge, no existing system can solve the IQA task (in particular, can deal simultaneously with KG incompleteness, complex queries, and the inductive setting); thus, we compared the instances  $\text{GNNQ}_L$  against instances  $\text{GNNQ}_L^-$  of GNNQ that do not use KG augmentation, which we treat as baselines. Our experiments show that the  $\text{GNNQ}_L$  instances significantly outperform the  $\text{GNNQ}_L^-$  instances, even if the RGCNs underlying the latter use more layers. Thus, we conclude that KG augmentation can provide a significant advantage in solving the IQA task in practice. All experiments were performed on a machine equipped with an Intel<sup>®</sup> Core<sup>™</sup> i9-10900K CPU, 64GB of RAM, running Ubuntu 20.04.4, and a Nvidia GeForce RTX 3090 GPU.

### 6.1 Benchmarks

The existing benchmarks for query answering on KGs used in the QE literature [4, 9, 11, 17, 18, 20] are designed for the transductive setting, so we cannot use them for an informative comparison of systems addressing the IQA task. Thus, in order to evaluate GNNQ instances, we have designed nine novel IQA benchmarks. Six of these, called WATDIV-Q $i$ , for  $i \in \{1, \dots, 6\}$ , are based on synthetic KGs generated with the WatDiv framework [3], and the remaining three, called FB15k237-Q $i$ , for  $i \in \{1, 2, 3\}$ , are based on subgraphs of FB15k-237 [6], a real-life KG commonly used in benchmarks for evaluation of KG completion and QE systems. Each of our benchmarks provides the following:

- a set  $\text{Pred}$  of unary and binary predicates and a tree-CQ over  $\text{Pred}$ ;
- sets of examples for training (including validation) and testing; each example is of the form  $(\mathcal{K}, a, \text{Ans})$  where  $\mathcal{K}$  is a KG over  $\text{Pred}$ ,  $a$  is a constant, and  $\text{Ans} \in \{0, 1\}$  is the ground-truth answer.

The benchmarks are constructed so that the ground-truth answer of an example is 1 if and only if  $a \in q[\mathcal{K}^*]$ , where  $\cdot^*$  is a hidden completion function over  $\text{Pred}$ , which is not given as part of the benchmark. For all our benchmarks,  $\cdot^*$  is defined by appropriately constructed Datalog rules [2]; such an approach allows us to capture structural dependencies of KGs, which are best-fitted for the inductive setting [23]. Table 1 summarises the statistics of our nine benchmarks. Further details about the selection of queries, completion functions, and examples for each benchmark are provided in the supplemental material.

## 6.2 GNNQ Implementation

Using a set of predicates  $\text{Pred}$  and a tree-CQ  $q(x)$  as parameters, each  $GNNQ_L$  processes a KG  $\mathcal{K}$  over  $\text{Pred}$  and a candidate constant  $a \in \text{Const}(\mathcal{K})$  by performing the following steps, implementing (and specifying) our approach.

*Step 1.* Each  $GNNQ_L$  computes a partial augmentation  $\mathcal{R}'_q(\mathcal{K})$  of  $\mathcal{K}$  with respect to some subset  $\mathcal{R}'_q \subseteq \mathcal{R}_q$  specified as follows: for the FB15k237- $Q_i$  benchmarks, we take  $\mathcal{R}'_q = \mathcal{R}_q$ ; in contrast, for the WATDIV- $Q_i$  benchmarks, we take  $\mathcal{R}'_q$  as the subset of all rules in  $\mathcal{R}_q$  with at most 4 variables. We selected such  $\mathcal{R}'_q$  because, on the one hand, the FB15k237- $Q_i$  benchmarks are relatively irregular, so we expect that even with full augmentation only a relatively small number of augmentation facts will be generated; on the other hand, the WATDIV- $Q_i$  benchmarks are highly regular, which suggests that performance may be hampered if we perform full augmentation, as this will derive many similar facts, which may cause problems analogous to over-smoothing. Each  $GNNQ_L$  then encodes  $\mathcal{R}'_q(\mathcal{K})$  as a  $(\text{Col}, \delta)$ -hypergraph  $G_{\mathcal{R}'_q(\mathcal{K})}$  with appropriate  $\text{Col}$  and  $\delta$  (see Sect. 4.2).

*Step 2.* Each  $GNNQ_L$  applies, to  $G_{\mathcal{R}'_q(\mathcal{K})}$ , a  $(\text{Col}, \delta)$ -HRGCN  $\mathfrak{S}$  with  $L$  layers, dimensions  $(\delta^0, \dots, \delta^L)$  such that  $\delta^0 = \delta$  and  $\delta^L = 1$ , and the following components. Functions  $\text{Aggr}^\ell$  and  $\text{Comb}^\ell$  for each layer  $\ell \in \{1, \dots, L\}$  of  $\mathfrak{S}$  are defined so that the feature vector of each node  $v$  is updated as

$$\mathbf{v}^\ell = \sigma^\ell \left( \mathbf{C}^\ell \mathbf{v}^{\ell-1} + \sum_{c \in \text{Col}} \sum_{(u_1, \dots, u_{k_c}) \in N_{G_{\mathcal{R}'_q(\mathcal{K})}}^c(v)} \frac{\mathbf{A}_c^\ell [\mathbf{u}_1^{\ell-1}, \dots, \mathbf{u}_{k_c}^{\ell-1}]}{|N_{G_{\mathcal{R}'_q(\mathcal{K})}}^c(v)|} + \mathbf{b}^\ell \right),$$

where  $\sigma^\ell$  is a element-wise leaky ReLU for each  $\ell \in \{1, \dots, L-1\}$  and the element-wise sigmoid function if  $\ell = L$ ; where every  $\mathbf{C}^\ell$  and  $\mathbf{A}_c^\ell$ , for each colour  $c \in \text{Col}$ , are (learnable) real-valued matrices of dimension  $\delta^\ell \times \delta^{\ell-1}$  and  $\delta^\ell \times (k_c \delta^{\ell-1})$ , respectively, for  $k_c + 1$  the arity of  $c$ , and each  $\mathbf{b}^\ell$  is a (learnable) real-valued *bias* vector of dimension  $\delta^\ell$ ; and where  $[\mathbf{u}_1^\ell, \dots, \mathbf{u}_{k_c}^\ell]$  is the vector obtained by concatenating  $\mathbf{u}_1^\ell, \dots, \mathbf{u}_{k_c}^\ell$ . The classification function maps  $x \in \mathbb{R}$

**Table 2.** Results for WATDIV-Q*i* benchmarks in the format precision/recall/AP

	WATDIV-Q1	WATDIV-Q2	WATDIV-Q3	WATDIV-Q4	WATDIV-Q5	WATDIV-Q6
$GNNQ_{h-1}^-$	.648/.646/.678	.655/.722/.661	.652/.731/.729	.660/.590/.653	.733/.724/.780	.555/.614/.578
$GNNQ_h^-$	.742/.707/.771	.819/.852/.881	.680/.787/.784	<b>.791</b> /.700/.798	.696/ <b>.933</b> /.860	.625/.840/.733
$GNNQ_{h+1}^-$	.621/ <b>.856</b> /.750	<b>.919</b> /. <b>920</b> /. <b>969</b>	.742/ <b>.835</b> /.807	.770/.804/ <b>.829</b>	.865/.924/ <b>.925</b>	<b>.852</b> /. <b>815</b> /. <b>877</b>
$GNNQ_{h-1}$	.737/.721/.815	.779/.820/.858	.700/.806/.793	.717/.783/.806	.743/.833/.885	.736/.619/.700
$GNNQ_h$	<b>.806</b> /.772/ <b>.870</b>	.821/.830/.906	<b>.797</b> /.791/ <b>.847</b>	.714/ <b>.839</b> /.827	<b>.876</b> /.852/.924	.763/.705/.784

to 1 if and only if  $x \geq 0.5$ . The feature vector dimensions  $\delta^1 = \dots = \delta^{L-1}$  and the negative slope of the ReLU activations are tuneable hyperparameters.

*Step 3.* The model returns 1 if  $a \in \mathfrak{S}_{\text{true}}[G_{\mathcal{R}'_q}(\mathcal{K})]$  and 0 otherwise.

The baselines  $GNNQ_L^-$  follow the same procedure, except that they skip KG augmentation and use  $\mathcal{K}$  instead of  $\mathcal{R}'_q(\mathcal{K})$ , thus relying on vanilla RGCNs [19].

For each benchmark, we trained and evaluated the  $GNNQ_L$  instances for each  $L \in \{h-1, h\}$  and the  $GNNQ_L^-$  instances for each  $L \in \{h-1, h, h+1\}$ , where  $h$  is the height of the benchmark’s tree-CQ. Before training, we randomly split the benchmark’s training-and-validation set of examples into training and validation sets with ratio 1:1 or 2:1, in case of a WATDIV or a FB15k237 benchmark, respectively. In each training run (on the training set), we trained all model parameters for 250 epochs using the Adam optimiser and a standard binary cross-entropy loss computed using the value of the (1-dimensional) feature vector in the last layer of the model as the prediction value (i.e., without applying the classification function). Each training run is specified by hyperparameters: the learning rate from  $\{.0001, .0006, \dots, .1001\}$ , the negative slope of the leaky-ReLU activation functions from  $\{.001, .006, \dots, .101\}$ , and the latent feature vector dimension from  $\{8, 9, \dots, 64\}$ . We report results for the hyperparameter values maximising the *average precision* on the validation set, which are found by means of 100 training runs using Optuna (MedianPruner) with 5 warm-up runs, 30 warm-up epochs in every run, and step size 25.

### 6.3 Performance Metrics

For each benchmark, we evaluated all the (best of the) trained models over the test set. For each model, we recorded the numbers  $tp, tn, fp, fn$  of true positives, true negatives, false positives, and false negatives, respectively, and report the *precision*  $tp/(tp + fp)$  and *recall*  $tp/(tp + fn)$  metrics. Furthermore, to test the robustness of our models under variations to the threshold used in the classification, we modified each learned model by removing the application of the classification function, so that each modified model returns the real value labelling the node for the candidate constant in the last layer. We then applied the modified models to the test set, and used the outputs to compute the *average precision* (*AP*), which is the area under the precision-recall curve.

**Table 3.** Results for the FB15k237-Qi benchmarks in the format precision/recall/AP

	FB15k237-Q1	FB15k237-Q2	FB15k237-Q3
$GNNQ_{h-1}^-$	.582/ .451/ .554	.569/ .618/ .544	.550/ .648/ .518
$GNNQ_h^-$	.606/ .382/ .581	.603/ .559/ .557	.588/ .621/ .579
$GNNQ_{h+1}^-$	.766/ .597/ .742	.624/ .505/ .580	.556/ .731/ .593
$GNNQ_{h-1}$	.903/ .873/ .958	.641/ .650/ <b>.757</b>	.766/ .769/ .889
$GNNQ_h$	<b>.919/.922/.976</b>	<b>.643/.664/</b> .670	<b>.822/.828/.933</b>

## 6.4 Results

We report the results of our experiments for the WATDIV-Qi and FB15k237-Qi benchmarks in Tables 2 and 3, respectively. As one can see, the  $GNNQ_L$  instances outperform the  $GNNQ_L^-$  instances on almost all benchmarks, when comparing instances whose HRGCN has the same number of layers. Furthermore, the  $GNNQ_L$  instances with the smallest number of layers outperformed all  $GNNQ^-$  instances by a significant margin on the FB15k237-Qi benchmarks. We attribute this to the fact that the real-world KGs are more noisy than the synthetic ones, and the baselines are more vulnerable to noise since they must learn longer dependencies. These results confirm our hypothesis that augmenting input KGs with facts representing the parts of the KG that satisfy connected query fragments can lead to improved empirical performance in the IQA task.

## 7 Related Work

*KG Completion*, which predicts missing facts in a KG, is a central soft reasoning task on KGs. Existing KG completion approaches can be classified in two categories. *Transductive* KG completion models learn an *embedding* function that maps constants and predicates in a fixed KG to elements of a vector space. At inference time, a missing target fact can then be verified by first applying the embedding function to the predicate and constants used in the target fact, and then applying a fixed *scoring* function to the resulting embeddings [1, 6, 7, 21, 27]. *Inductive* KG completion assumes only a fixed set of predicates, and a trained model can be applied to any KG over these predicates. Many inductive KG completion approaches use GNNs [10, 14, 23, 24], which can reason over the structure of KGs and are therefore inductive by design.

*Query Embedding* (QE) aims to answer monadic queries from various classes over the completion of an arbitrary but fixed KG. Common QE approaches are inspired by embedding-based KG completion methods [4, 9, 11, 17, 18, 20]. To produce query answers that are not logically entailed, such QE models usually jointly learn embedding functions for constants and for queries during training. At inference time, a QE model first embeds the input query using the learnt embedding functions and then scores constants as potential answers based on

the distance of their embeddings to the query embedding. Thus QE approaches aim to answer arbitrary queries over the predicates and constants of a fixed KG. This is orthogonal to our inductive setting, which assumes a fixed query but is applicable to arbitrary KGs (over a predefined set of predicates).

*Connection of Logic and GNNs.* The increasing interest in GNNs across different domains has motivated the theoretical analysis of the expressiveness and limitations of GNNs. For example, it is trivial to see that GNNs cannot distinguish between two non-isomorphic  $k$ -regular graphs of the same size with uniform node features. Further analysis connected GNNs to the family of well-known Weisfeiler-Lehman (WL) graph isomorphism tests; in particular, Xu et al. [26] and Morris et al. [15] independently showed that the most expressive GNNs can distinguish the same nodes as the 1-dimensional WL test and hence between the same nodes as formulas in  $\text{FOC}_2$ , the two-variable fragment of the first-order logic with counting quantifiers. Further deep connections between various logics and GNNs have recently followed these works [5, 8, 22], and we anticipate that these results are paving a path for future efficient neuro-symbolic AI approaches to many tasks in data and knowledge management.

## 8 Conclusion and Future Work

In this paper, we presented a novel neuro-symbolic approach to query answering over incomplete KGs. In contrast to existing embedding-based approaches, which assume a fixed KG, our approach is inductive—that is, it only relies on a fixed set of predicates and is thus applicable to arbitrary KGs over these predicates. Our approach proceeds in three phases. First, it uses symbolic rules to augment the input KG with facts representing subgraphs that match connected fragments of the query. Second, it encodes the augmented KG into a hypergraph with vector-labelled nodes. Third, it processes the hypergraph using a Hyper-Relational Graph Convolutional Network (HRGCN), a novel GNN architecture which generalises the well-known RGCN architecture. We then provided a theorem showing that the KG augmentation phase can considerably reduce the number of layers a HRGCN-based system needs to produce correct answers to a query on every KG. Finally, we implemented our approach in the GNNQ system and evaluated it on several novel benchmarks. Our experiments showed that KG augmentation indeed leads to improved empirical performance in the IQA task. The main challenge for future work is extending our approach to support more expressive queries. We shall also investigate the queries and completion functions that can be perfectly captured by our approach and its potential extensions.

**Supplemental Material Statement.** A proof of Theorem 1 as well as details about the creation of the benchmark datasets can be found in the supplementary material. This material, together with the source code of GNNQ, the benchmarks, and the instructions for the reproduction of our experiments are accessible through Github (<https://github.com/KRR-Oxford/GNNQ>).

**Acknowledgements.** This work was supported by Siemens AG, the AIDA project (Alan Turing Institute, EP/N510129/1), the SIRIUS Centre for Scalable Data Access (Research Council of Norway, project number 237889), Samsung Research UK, and the EPSRC projects ConCur (EP/V050869/1), OASIS (EP/S032347/1) and UK FIRES (EP/S019111/1). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

## References

1. Abboud, R., Ceylan, I., Lukasiewicz, T., Salvatori, T.: Boxe: a box embedding model for knowledge base completion. In: *Advances in Neural Information Processing Systems*, vol. 33 (2020)
2. Abiteboul, S., Hull, R., Vianu, V.: *Foundations of Databases*. Addison Wesley, Boston (1995)
3. Aluç, G., Hartig, O., Özsu, M.T., Daudjee, K.: Diversified stress testing of RDF data management systems. In: Mika, P., et al. (eds.) *ISWC 2014. LNCS*, vol. 8796, pp. 197–212. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-11964-9\\_13](https://doi.org/10.1007/978-3-319-11964-9_13)
4. Arakelyan, E., Daza, D., Minervini, P., Cochez, M.: Complex query answering with neural link predictors. In: *International Conference on Learning Representations (2021)*. <https://openreview.net/forum?id=Mos9F9kDwz>
5. Barceló, P., Kostylev, E.V., Monet, M., Pérez, J., Reutter, J., Silva, J.P.: The logical expressiveness of graph neural networks. In: *International Conference on Learning Representations (2019)*
6. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: *Advances in Neural Information Processing Systems*, pp. 2787–2795 (2013)
7. Dettmers, T., Pasquale, M., Pontus, S., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pp. 1811–1818, February 2018 <https://arxiv.org/abs/1707.01476>
8. Grohe, M.: The logic of graph neural networks. In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–17. IEEE (2021)
9. Guu, K., Miller, J., Liang, P.: Traversing knowledge graphs in vector space. arXiv preprint [arXiv:1506.01094](https://arxiv.org/abs/1506.01094) (2015)
10. Hamaguchi, T., Oiwa, H., Shimbo, M., Matsumoto, Y.: Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 1802–1808. AAAI Press (2017)
11. Hamilton, W.L., Bajaj, P., Zitnik, M., Jurafsky, D., Leskovec, J.: Embedding logical queries on knowledge graphs. In: *Advances in Neural Information Processing Systems*, pp. 2026–2037 (2018)
12. Hamilton, W.L.: Graph representation learning. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14(3), pp. 1–159 (2021)
13. Hogan, A., et al.: Knowledge Graphs. *ACM Computing Surveys* **54**(4), 71:1-71:37 (2021)
14. Liu, S., Cuenca Grau, B., Horrocks, I., Kostylev, E.V.: Indigo: GNN-based inductive knowledge graph completion using pair-wise encoding. In: *NeurIPS (2021)*



15. Morris, C., et al.: Weisfeiler and leman go neural: higher-order graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 4602–4609 (2019)
16. Motik, B., Nenov, Y., Piro, R., Horrocks, I., Olteanu, D.: Parallel Materialisation of datalog programs in centralised, main-memory RDF systems. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI 2014), pp. 129–137 (2014)
17. Ren, H., Hu, W., Leskovec, J.: Query2box: reasoning over knowledge graphs in vector space using box embeddings. In: International Conference on Learning Representations (2020)
18. Ren, H., Leskovec, J.: Beta embeddings for multi-hop logical reasoning in knowledge graphs (2020)
19. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-93417-4\\_38](https://doi.org/10.1007/978-3-319-93417-4_38)
20. Sun, H., Arnold, A.O., Bedrax-Weiss, T., Pereira, F., Cohen, W.W.: Faithful embeddings for knowledge base queries. In: Advances in Neural Information Processing Systems, vol. 33 (2020)
21. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. In: International Conference on Learning Representations (2019). <https://openreview.net/forum?id=HkgEQnRqYQ>
22. Tena Cucala, D.J., Cuenca Grau, B., Kostylev, E.V., Motik, B.: Explainable GNN-based models over knowledge graphs. In: International Conference on Learning Representations (2022). <https://openreview.net/forum?id=CrCvGNHAIrz>
23. Teru, K., Denis, E.G., Hamilton, W.L.: Inductive relation prediction by sub-graph reasoning. In: International Conference on Machine Learning, pp. 9448–9457. PMLR (2020)
24. Wang, H., Ren, H., Leskovec, J.: Entity context and relational paths for knowledge graph completion. arXiv preprint [arXiv:2002.06757](https://arxiv.org/abs/2002.06757) (2020)
25. Wang, P., Han, J., Li, C., Pan, R.: Logic attention based neighborhood aggregation for inductive knowledge graph embedding. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, pp. 7152–7159. AAAI Press (2019)
26. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? CoRR abs/1810.00826 (2018), <http://arxiv.org/abs/1810.00826>
27. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases (2015)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

