CQD-SHAP: Explainable Complex Query Answering via Shapley Values

Parsa Abbasi
Paderborn University
Paderborn, Germany
parsa.abbasi@uni-paderborn.de

Stefan Heindorf
Paderborn University
Paderborn, Germany
heindorf@uni-paderborn.de

Abstract

Complex query answering (CQA) goes beyond the well-studied link prediction task by addressing more sophisticated queries that require multi-hop reasoning over incomplete knowledge graphs (KGs). Research on neural and neurosymbolic CQA methods is still an emerging field. Almost all of these methods can be regarded as black-box models, which may raise concerns about user trust. Although neurosymbolic approaches like CQD are slightly more interpretable, allowing intermediate results to be tracked, the importance of different parts of the query remains unexplained. In this paper, we propose CQD-SHAP, a novel framework that computes the contribution of each query part to the ranking of a specific answer. This contribution explains the value of leveraging a neural predictor that can infer new knowledge from an incomplete KG, rather than a symbolic approach relying solely on existing facts in the KG. CQD-SHAP is formulated based on Shapley values from cooperative game theory and satisfies all the fundamental Shapley axioms. Automated evaluation of these explanations in terms of necessary and sufficient explanations, and comparisons with various baselines, shows the effectiveness of this approach for most query types.

CCS Concepts

• **Computing methodologies** → *Knowledge representation and reasoning*.

Keywords

Explainable Query Answering, Shapley Value, Complex Query Answering, Neurosymbolic Query Answering, Knowledge Graph

1 Introduction

Knowledge graphs (KGs) are a rich resource for retrieving fact-based answers to queries. However, real-world KGs are often incomplete. For example, in Freebase [4], among more than 3 million persons, only 25% have a recorded nationality and only 6% have information about their parents [31]. Similarly, in Wikidata [28] only 50% of the artists have a recorded place of birth [34]. Consequently, retrieving knowledge solely by traversing the graph, a procedure we refer to as *symbolic execution*, is prone to missing relevant results.

Although there has been extensive research on the link prediction task [29], these methods focus on answering simple one-hop queries, i.e., inferring a missing link between pairs of entities. In practice, however, the queries of real interest are usually more complex, often combining conditions through logical operators (e.g., conjunction or disjunction) and requiring multi-hop reasoning. Towards this end, a number of neural and neurosymbolic approaches have recently been proposed to answer complex queries on incomplete KGs [22]. One of the first and most influential among them is CQD [1], a neurosymbolic model, which decomposes a query into atomic link predictions, hereafter called *neural execution*, and combines the results through fuzzy logic.

Despite their success, neural query answering approaches often lack transparency, and it is not always clear why a particular answer is produced. While some explanation methods have been proposed for one-hop link prediction models [6, 18, 25, 35], they are not directly applicable to complex queries. Moreover, such methods typically focus on identifying *important triples* in the KG for a given prediction. In contrast, complex queries may require different types of evidence across different parts of the query, calling for novel definitions of explanation. In this work, we focus on the constituent parts of a query, which we refer to as *atoms*, and aim to explain the *importance of each atom* contributing to the ranking of an answer.

We hypothesize that quantifying the contribution of query atoms is useful in different ways: (1) it allows users to investigate the neural model's influence on each query atom, increasing their trust in the results, (2) it provides novel insights into the behavior of neural models and reveals potential weaknesses, supporting comparison and guiding future improvements, (3) it highlights the query parts where the neural model's inference of missing knowledge strongly affects an answer's ranking, helping domain experts identify and correct gaps in the KG or data scientists debug the model or queries.

The practical relevance of atom-level explanations is best demonstrated through an example. Consider the query: "Which drugs are prescribed for diabetes *and* cause kidney toxicity?" Suppose `Insulin` appears among the top-ranked predictions in the output of the neural model. Our proposed explanation approach can help the user understand why the neural reasoner produces this ranking. Specifically, it reveals whether the ranking is driven primarily by the fact that `Insulin` is prescribed for diabetes in the KG, or by the (incorrect) association that it causes kidney toxicity. For instance, the method might assign a contribution score of 10 to the first atom ("prescribed for diabetes") and 450 to the second atom ("causes kidney toxicity"), indicating that the latter contributed much more to achieving the high rank. Such explanations enable the user to critically examine that part of the query and its associated subgraph, in order to assess whether the system is relying on misleading correlations, behaving incorrectly, or potentially uncovering a valid but previously overlooked insight.

In this paper, we propose *CQD-SHAP*, a novel approach to explaining the results of the CQD model using Shapley values [26], which are grounded in cooperative game theory. We define a Shapley game over query atoms and compute a Shapley value for each atom, thereby quantifying its contribution to the ranking of a target

answer. Since the game is defined in terms of retrieving the result of an atom either symbolically or neurally, the Shapley value of each atom can be interpreted as the extent to which executing that atom with a neural model affects the ranking of the answer of interest.

For instance, in the previous example, if the Shapley value of the second atom ("causes kidney toxicity") is 450, this indicates that, averaged over all possible execution choices of the other atom, executing this atom neurally improves the ranking of the answer `Insulin` by 450 positions. Shapley values are supported by a mathematically rigorous foundation, and any well-defined Shapley game inherits useful formal properties. One such property is *efficiency*, which provides additional value for explainability. In our framework, efficiency means that the sum of Shapley values across all atoms in a query equals the overall improvement achieved by executing the entire query neurally, compared to executing it symbolically (i.e., by traversing the KG alone).

We summarize our main contributions as follows:

- To the best of our knowledge, this is the first approach to explaining the results of neural complex query answering models using Shapley values.
- We propose a formal definition of explanation for the complex query answering task at the level of query atoms.
- We introduce *CQD-SHAP*, which quantifies the contribution of using a neural model for each query atom to the ranking of a target answer.
- We conduct extensive evaluations of the *CQD-SHAP* explanations in terms of *necessary* and *sufficient* explanations, through quantitative experiments on two standard benchmarks, FB15k-237 and NELL995, showing that our method yields meaningful and interpretable explanations.

All resources used in this work, including code, data, and pretrained models, are publicly available in our GitHub repository.[1]

2 Related Work

Neural Query Answering. Unlike symbolic query answering methods, such as SPARQL, which can only retrieve incomplete sets of answers directly reachable within the existing graph, neural methods attempt to infer missing information using neural networks, thereby recovering additional *hard answers* that symbolic methods cannot attain. Following the categorization suggested in [22], *Neural Query Answering* methods execute all parts of a query in the latent space. Some neural approaches, such as GQE [14], embed queries, entities, and relations as vectors and model logical operations as geometric transformations. However, other works, such as MPQE [9], do not employ any explicit execution of logical operations and instead process the entire query graph simultaneously.

Neurosymbolic Query Answering. As the name suggests, these methods lie between neural and symbolic methodologies. They execute query atoms (projections) using a neural approach, but simulate logical operations when combining neural results to enable greater interpretability. There are different ways to mimic logical operations; for example, Query2Box [23] leverages geometric operations, whereas CQD [1] and its variants [2, 10, 19, 30] employ fuzzy logic. In this work, we focus on neurosymbolic approaches and specifically adopt the design principles underlying CQD.

Explainable Link Prediction. Complex query answering methods, and in particular neurosymbolic approaches such as CQD [1], can be viewed as extensions of the well-studied link prediction problem. Recent advances in Graph Neural Networks (GNNs) have yielded strong performance across graph-based tasks, including link prediction, but their black-box nature makes predictions difficult to interpret. Since many real-world applications require both high performance and transparency, various explanation methods have been proposed for link prediction problem in recent years. For example, Kelpie [25] explains why a given tail entity o is top-ranked for a query $(s, p, ?)$ by identifying training facts that are *necessary* or *sufficient* for the prediction. CRIAGE [20] identifies influential facts by applying adversarial perturbations to the KG and observing their impact on the prediction. Other works, such as Power-Link [6] and PaGE-Link [35], instead focus on path-based explanations, finding influential paths whose intermediate nodes and edges strongly impact a predicted link. However, all these methods primarily address single-projection link prediction, whereas complex queries require multiple projections whose contributions may interact to produce the final answer. In our work, we address complex queries that require multiple projections, where the results of individual projections may depend on each other to produce the final answer. We do not assume that every projection is executed by a link prediction model; instead, we measure the importance of each projection by assessing how much replacing its neural execution via a link prediction model with a symbolic execution (i.e., a simple graph look-up) affects the final result. Our explanation approach considers different paths leading to the final answer set, which makes it partially related to path-based explainable link prediction methods. Moreover, since our work builds on CQD, the path to each final answer remains explicit and accessible to the end user.

Explainable Query Answering. To the best of our knowledge, there have not been any direct attempts to explain complex query answering over KGs. However, there is considerable research on query explanation in the context of traditional databases. For example, Gathani et al. [12] surveys various techniques that can aid in explaining and debugging database queries, and our approach is related to the "Why and Why Not" [7] strategy introduced in that work. Similar to our study, some works such as [11] and [16] adopt the Shapley value methodology for explanation. However, these works are not specifically focused on explaining the query itself; rather, they quantify the contribution of database facts or tuples to the query results. Furthermore, both methods focus on aggregation or Boolean queries, which have numerical outputs such as "Is there any route from any airport in Germany to any airport in Iran with at most one connection?"—where the answer is simply Yes (1) or No (0). While the aforementioned studies are based on databases, there is another line of research like [3] addressing knowledge bases; however, their objective remains to quantify the contribution of facts, not to explain the query.

Shapley Values for Ranking and Retrieval. Although there is a lack of research on using Shapley values to explain complex queries, it is possible to approach this problem through the perspective

[1] https://github.com/ds-jrg/CQD-SHAP

of ranking and retrieval, as both domains output an ordered list of entities ranked by their relevance to a query. Shapley values, and particularly SHAP [17], were originally introduced for models that output a single score. Therefore, applying them to explain the outcome of a ranking model is not straightforward and requires a novel definition of a value function capable of producing a single score from a ranked list. In recent years, this challenge has been the focus of several studies. For instance, Heuss et al. [15] introduced an extension of Shapley value, named RankingSHAP, which generates the feature contributions for ordered lists. However, their proposed method does not satisfy some of the most fundamental properties of Shapley values, such as *efficiency* and *monotonicity*, as correctly demonstrated in Chowdhury et al. [8].

Chowdhury et al. [8] proposed another method called *RankSHAP* by introducing a generalized ranking value function (referred to as GREM) and discussing the properties required to satisfy the four Shapley value axioms in ranking. However, both of the aforementioned methodologies focus on explaining feature contributions for the entire ranking. In contrast, Pliatsika et al. [21] focus on explaining a single ranked outcome and propose a different method for quantifying the payoff with respect to the ranking of a particular target entity, referred to as the *Quantity of Interest (QoI)*. Furthermore, they demonstrate that these proposed QoIs satisfy all the fundamental axioms. We follow this approach and use the same QoI definitions.

3 Background

We briefly introduce the task of complex query answering and fundamentals of Shapley values, before introducing our approach in the next section.

3.1 Complex Query Answering

Knowledge Graph. A triple-based knowledge graph $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ comprises a set of triples $\langle s, p, o \rangle$, where each triple represents a factual statement involving a relation $p \in \mathcal{R}$ between the subject entity $s \in \mathcal{E}$ and the object entity $o \in \mathcal{E}$. Here, $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively.

Queries. Following Arakelyan et al. [1], queries over such a KG can be expressed in first-order logic (FOL). For this, each triple can be represented as an atomic formula $a = p(s, o)$. By combining such atomic formulas using logical operators such as conjunction ($\wedge$) and existential quantification ($\exists$), one can construct a class of FOL queries known as *conjunctive queries*. Following Arakelyan et al. [1], we focus on Existential Positive First-Order (EPFO) queries, which additionally include disjunction ($\vee$), and we transform such queries into Disjunctive Normal Form (DNF). The formal definition of a DNF query Q is provided in Equation (1). In this formulation, $V_1, \ldots, V_m$ denote variables that are bound to nodes through existential quantification and each $e \in \mathcal{E}$ represents a fixed, constant input entity, also called *anchor* node. The variable V_A in this equation is the target of the query, and we denote the answer set of the query Q by $\mathcal{E}_A = [\![Q[V_A]]\!]$.

$$
\begin{aligned}
Q[V_A] \triangleq ?V_A : \exists\, V_1, \ldots, V_m \cdot \\
\Big[(a_1^1 \wedge \cdots \wedge a_{n_1}^1) \;\vee\; \cdots \vee\; (a_1^d \wedge \cdots \wedge a_{n_d}^d) \Big] \\
\text{where} \quad a_i^j = \begin{cases} p(e, V), & V \in \{V_A, V_1, \ldots, V_m\}, \\ & e \in \mathcal{E},\ p \in \mathcal{R} \\ p(V, V'), & V \in \{V_A, V_1, \ldots, V_m\}, \\ & V' \in \{V_A, V_1, \ldots, V_m\}, \\ & V \neq V',\ p \in \mathcal{R} \end{cases}
\end{aligned} \tag{1}
$$

Symbolic Query Answering. In a symbolic approach, the query Q is answered by binding V_A to each entity in the KG and determining for which entities $Q[V_A]$ holds true. This approach implicitly assumes that the KG is complete.

Neurosymbolic Query Answering. In practice, knowledge graphs are often incomplete and so is the result of a symbolic approach. Neurosymbolic complex query answering models [22] address this limitation by attempting to infer missing information and produce approximate answer sets. For our evaluation, we assume that we have a complete knowledge graph G_{test}. From this, we randomly sample a subset of its triples, yielding the incomplete knowledge graph $G_{\text{valid}} \subseteq G_{\text{test}}$. We then train an approximate, neurosymbolic query answering model on G_{valid}, use it to answer queries, and compare the answers (by the neurosymbolic approach executed on G_{valid}) to the ground truth answers obtained by executing a symbolic query answering approach on G_{test}. The goal of the neurosymbolic model is to approximate the answers that a symbolic method would return on the complete graph—despite only having access to the incomplete graph during training.

Complex Query Decomposition (CQD). The goal of CQD [1] is to find a mapping from variables to entities that maximizes the score of Q, as shown in Equation (2). For this, CQD computes a prediction score $\rho(\mathbf{e}_s, \mathbf{e}_p, \mathbf{e}_o)$ for each atom in the query, based on the embedding vectors of its subject, predicate, and object, $\mathbf{e}_s, \mathbf{e}_p, \mathbf{e}_o \in \mathbb{R}^\ell$, where ℓ denotes the embedding dimension. CQD then aggregates these scores using continuous generalizations of the logical operators, such as t-norms for conjunctions and t-conorms for disjunctions—denoted here as $\top$ and $\bot$, respectively.

$$
\begin{aligned}
&\underset{V_A, V_1, \ldots, V_m \in \mathcal{E}}{\arg\max} \Big[(a_1^1 \top \cdots \top a_{n_1}^1) \;\bot\; \cdots \bot\; (a_1^d \top \cdots \top a_{n_d}^d) \Big] \\
&\text{where } a_i^j = \begin{cases} \rho(\mathbf{e}_e, \mathbf{e}_p, \mathbf{e}_V), & V \in \{V_A, V_1, \ldots, V_m\}, \\ & e \in \mathcal{E},\ p \in \mathcal{R} \\ \rho(\mathbf{e}_V, \mathbf{e}_p, \mathbf{e}_{V'}), & V \in \{V_A, V_1, \ldots, V_m\}, \\ & V' \in \{V_A, V_1, \ldots, V_m\}, \\ & V \neq V',\ p \in \mathcal{R} \end{cases}
\end{aligned} \tag{2}
$$

Let $z(e_i) = (a_1^1 \top \cdots \top a_{n_1}^1) \;\bot\; \cdots \bot\; (a_1^d \top \cdots \top a_{n_d}^d)$ denote the score assigned by the neurosymbolic CQA model to entity $e_i \in \mathcal{E}_A$ as a candidate answer to the query Q. To compute this score, we substitute $V_A \leftarrow e_i$ and replace the remaining variables with the entities that maximize the final score.

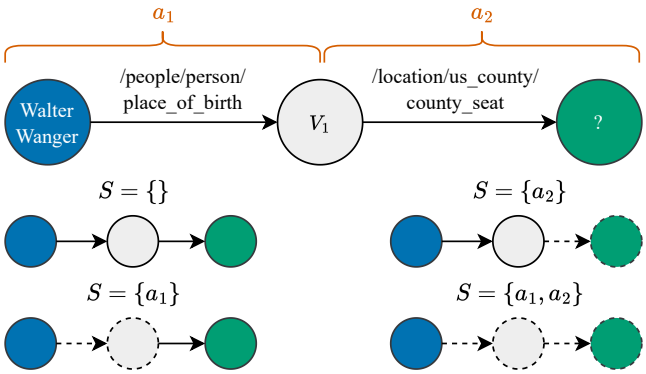

Figure 1: Illustration of all possible partial queries Q_S for a query consisting of two sequential projections ($2p$). We execute the atom a_i using the symbolic approach when $a_i \notin S$ (solid lines), and using the neural approach when $a_i \in S$ (dashed lines).

Then, $z(e_i)$ estimates the likelihood that the entity e_i is an answer to Q in the complete KG. In practice, the model ranks all candidate entities in the graph (or a subset thereof) according to their prediction scores, producing an answer set $\mathcal{E}_A \subseteq \mathcal{E}$. Therefore, for any $e_i, e_j \in \mathcal{E}_A$, if the rank $r_i < r_j$, then e_i is more likely to be an answer to the query than e_j.

3.2 Shapley Values

Shapley values, originally introduced by *Lloyd Shapley* in cooperative game theory [26], are widely used in XAI and popularized by SHAP [17]. They provide a fair and principled way to attribute a model's output to its input features. Formally, let $P = \{1, 2, \ldots, p\}$ be a set of *players*, and let $val : 2^P \to \mathbb{R}$ be a *value function* that assigns a real number to each subset $S \subseteq P$, with $val(\emptyset) = 0$. The *Shapley value* $\phi_i(v)$ for a player $i \in P$ is defined as:

$$\phi_i(val) := \sum_{S \subseteq P \setminus \{i\}} \frac{|S|!\,(|P| - |S| - 1)!}{|P|!} \left[val(S \cup \{i\}) - val(S)\right] \quad (3)$$

This formula computes the average marginal contribution of player i across all possible subsets $S \subseteq P \setminus \{i\}$, weighted by the number of permutations in which i joins the coalition after S—ensuring the properties, efficiency, symmetry, linearity, and null player [32].

4 Explainable Complex Query Answering

Given a query Q which consists of the atoms $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$, each atom can either be answered with a symbolic approach (possibly returning an incomplete or empty answer set) or with a neural approach. Figure 1 provides an illustration of this process, showing all possible partial queries for a simple case of two sequential projections ($2p$). Each path represents a different combination of neural and symbolic executions, with dashed and solid lines indicating the two cases, respectively. Finally, the answer is composed of the intermediate answers to the query atoms. Our goal is to use Shapley values to quantify the importance of employing a neural approach, rather than a symbolic one, for each atom when ranking a specific answer entity. Given a query Q and entity $e_i \in \mathcal{E}_A$, we define the cooperative game as follows:

- The players are the atoms: $P := \mathcal{A}$
- For any coalitions $S \subseteq P$, we define a partial query Q_S where S denotes the set of atoms to be answered neurally and the remaining atoms to be answered symbolically
- The value function $val_{e_i}(Q_S)$ evaluates how good the answer e_i is to the query Q_S

Using this definition of a game, we obtain a Shapley value for each atom of the query with respect to a given target entity. In the following, we define the execution of a partial query Q_S and the value function val in more detail.

For atoms in S, execution is performed using a neural approach (e.g., the link predictor in CQD [1]) which enables inference over incomplete KGs. For atoms not included in S, the answer is computed in a purely symbolic manner, relying solely on the observed facts in the KG.

$$Q_S[V_A] \triangleq$$
$$\underset{V_A, V_1, \ldots, V_m \in \mathcal{E}}{\arg\max} \left[(a_1^1 \top \cdots \top a_{n_1}^1) \;\bot\; \cdots \;\bot\; (a_1^d \top \cdots \top a_{n_d}^d) \right]$$
$$\text{where } a_i^j = \begin{cases} \rho(\mathbf{e}_e, \mathbf{e}_p, \mathbf{e}_V), \text{ or} \\ \rho(\mathbf{e}_V, \mathbf{e}_p, \mathbf{e}_{V'}), & \text{if } a_i^j \in S \\ p(e, V), \text{ or} \\ p(V, V'), & \text{otherwise} \end{cases} \quad (4)$$

To execute such a query, we employ the combinatorial optimization framework introduced in CQD. Query execution begins with atoms that contain an anchor, i.e., $p(e, V)$ or $\rho(\mathbf{e}_e, \mathbf{e}_p, \mathbf{e}_V)$. If the neural approach is selected ($\rho(\mathbf{e}_e, \mathbf{e}_p, \mathbf{e}_V)$), the neural link predictor assigns a score to each entity, representing the likelihood that the entity is connected to e via relation p. As in the original CQD approach, if the atom involves the target variable (V_A), the model produces a score for every entity. If, instead, the atom corresponds to an intermediate step in the beam search, the entities are sorted in descending order by score, and the top-k are returned.

If the symbolic approach is chosen (i.e., $p(e, V)$), we identify all entities connected to e via relation p in the observed graph. By *observed graph*, we mean the validation graph when explaining a test query, and the training graph when explaining a validation query. These entities are assigned a constant score (in our case, 1), while all other entities receive a low score close to 0. If this atom does not contain the target variable V_A, the entities are sorted in descending order, and the top-k candidates are returned; otherwise all results are returned.

This formulation ensures that the importance assigned to each atom reflects both its direct logical contribution and the impact of the specific reasoning mechanism (neural or symbolic) by which the query is executed.

Now that the definition of the game is in hand, the payoff or the value function of a coalition S should be defined. Following Pliatsika et al. [21], we can define different quantities of interest (QoIs) as the outcome of the value function based on the position of answer e_i in the ranking. In this work, we employ the function $\Delta Rank\ QoI$ as defined in the following. Suppose $\mathcal{E}_A$ denotes the answer set

for the partial query Q_S. In addition, let $\mathcal{E}_{\text{easy}}$ denote the set of *easy answers* that can be obtained through symbolic execution on the observed graph, and $\mathcal{E}_{\text{hard}}$ the set of *hard answers* that cannot. Following CQD [1], we adopt the *filtered* setting [5] for computing rankings in order to ensure a fair evaluation. Let $\mathcal{E}_A^{(i)}$ denote the set of candidate answers obtained by excluding all easy answers and all hard answers except the current target hard answer e_i:

$$\mathcal{E}_A^{(i)} = \mathcal{E}_A \setminus \Big(\mathcal{E}_{\text{easy}} \cup \big(\mathcal{E}_{\text{hard}} \setminus \{e_i\}\big)\Big). \tag{5}$$

The rank of an entity e_i is defined as its position in the filtered, ordered list obtained by sorting the entities $\mathcal{E}_A^{(i)}$ in descending order of their score, and is denoted by r_i:

$$r_i = |\{e_j \in \mathcal{E}_A^{(i)} \setminus \{e_i\} : z(e_j) > z(e_i)\}| + 1 \tag{6}$$

The Δ*Rank QoI* is then given as the difference between the rank of e_i in the answer to the partial query Q_S and its rank in the result of the complete symbolic query $Q_{\{\}}$, as formalized in Equation (7). This formulation ensures that the value of the empty coalition, $val_{e_i}(Q_{\{\}})$, is equal to zero, as required in the Shapley framework, thereby guaranteeing that the fundamental axioms are satisfied.

$$val_{e_i}(Q_S) = r_i(Q_{\{\}}) - r_i(Q_S) \tag{7}$$

Finally, to formally measure the marginal contribution of each atom to the ranking of a specific entity e_i, we compute the Shapley value for each atom $a \in \mathcal{A}$ by plugging our definition of a cooperative game into the original Shapley definition, given in Equation (3). The Shapley value then aggregates the expected contribution of atom a over all possible coalitions $S \subseteq \mathcal{A} \setminus \{a\}$:

$$\phi_a(e_i) := \sum_{S \subseteq \mathcal{A} \setminus \{a\}} \frac{|S|!\,(|\mathcal{A}| - |S| - 1)!}{|\mathcal{A}|!}\, \Delta val_{e_i}(S, a) \tag{8}$$

where $\Delta val_{e_i}(S, a) = val_{e_i}(Q_{S \cup \{a\}}) - val_{e_i}(Q_S)$, $|\mathcal{A}|$ denotes the total number of atoms in the query, and $\Delta val_{e_i}(S, a)$ represents the marginal change in the outcome when the atom a is added to the coalition S.

Our defined value functions align with the metrics proposed in Chowdhury et al. [8]. For example, for the ΔRank QoI, we can set the gain function to the linear form, $g(\text{rel}_j) = \text{rel}_j$, and use no discounting, $h(j) = 1$. In this case, the relevance score rel_j can be defined as the difference between the rank of e_j in Q_S and its rank in $Q_{\{\}}$, ensuring that the relevance sensitivity property holds. As proven in that study, such a function satisfies all the main Shapley value axioms. Moreover, since the number of atoms in a complex query is relatively small, no approximation is required, and the exact Shapley values can be computed.

Building upon the Shapley value's *efficiency* axiom, one can derive that the sum of the Shapley values for all atoms (a total of $|\mathcal{A}|$ atoms) with respect to a given answer e_i is equal to the value of the grand coalition, i.e., when all atoms are executed neurally ($Q_{\{a_1,\ldots,a_{|\mathcal{A}|}\}}$). This can be expressed as:

$$\begin{aligned}\phi(e_i) = \sum_{a \in \mathcal{A}} \phi_a(e_i) &= val_{e_i}(Q_{a_1,\ldots,a_{|\mathcal{A}|}}) = \\ &= r_i(Q_{\{\}}) - r_i(Q_{a_1,\ldots,a_{|\mathcal{A}|}})\end{aligned} \tag{9}$$

In other words, the sum of the Shapley values of all atoms corresponds to the difference in the rank of entity e_i when the query is executed entirely with the neurosymbolic approach ($Q_{\{a_1,\ldots,a_{|\mathcal{A}|}\}}$) as compared to purely symbolic execution ($Q_{\{\}}$).

Table 1: KG statistics (top) and query structures (bottom).

Dataset	Nodes	Relations	$\lvert G_{\text{train}}\rvert$	$\lvert G_{\text{valid}}\rvert$	$\lvert G_{\text{test}}\rvert$
FB15k-237	14,505	474	544,230	579,300	620,232
NELL995	63,361	400	456,852	716,472	1,005,086

Type	Logic
$2p$	$?V_2 : \exists V_1 \cdot p_1(e_1, V_1) \wedge p_2(V_1, V_2)$
$2u$	$?V_1 : p_1(e_1, V_1) \vee p_2(e_2, V_1)$
$2i$	$?V_1 : p_1(e_1, V_1) \wedge p_2(e_2, V_1)$
$3i$	$?V_1 : p_1(e_1, V_1) \wedge p_2(e_2, V_1) \wedge p_3(e_3, V_1)$
$3p$	$?V_3 : \exists V_1, V_2 \cdot p_1(e_1, V_1) \wedge p_2(V_1, V_2) \wedge p_3(V_2, V_3)$
$2u1p$	$?V_2 : \exists V_1 \cdot (p_1(e_1, V_1) \vee p_2(e_2, V_1)) \wedge p_3(V_1, V_2)$
$2i1p$	$?V_2 : \exists V_1 \cdot p_1(e_1, V_1) \wedge p_2(e_2, V_1) \wedge p_3(V_1, V_2)$
$1p2i$	$?V_2 : \exists V_1 \cdot p_1(e_1, V_1) \wedge p_2(V_1, V_2) \wedge p_3(e_2, V_2)$

5 Evaluation

After describing our evaluation setup, including performance metrics, evaluation scenarios and baselines (Section 5.1), we provide both quantitative results (Section 5.2) as well as quantitative results of a case study (Section 5.3).

5.1 Evaluation Setup

Datasets. We conduct experiments on the FB15k-237 [27] and NELL995 [33] datasets using the same data splits as Arakelyan et al. [1]. Table 1 (top) shows dataset statistics. Each benchmark contains eight query types, which are defined in Table 1 (bottom), with 5,000 queries per type in the validation/test sets for FB15k-237, and 4,000 queries per type for NELL995.

Neural Query Answering Approaches. We leverage the pre-trained CQD link prediction model without applying any additional fine-tuning. We explicitly reimplemented the combinatorial approach to answer complex queries, as our methodology requires executing some parts of the query using the CQD and handling others via symbolic approach. We validated our reimplementation and obtained results consistent with the original CQD across all query types. We employ a default value of k=10 for all query types, and use the product t-norm $\top_{\text{prod}}(s_1, s_2) = s_1 \cdot s_2$, and the product t-conorm $\perp_{\text{prod}}(s_1, s_2) = (s_1 + s_2) - (s_1 \cdot s_2)$.

Performance Metrics. Following CQD, the last layer of a query always outputs a complete list of scores for each entity in the graph, rather than retaining only the top k entities as is done in the intermediate steps. In our experiments, we adopt Mean Reciprocal Rank (MRR) and Hits@k as evaluation metrics, following their standard definitions [24], to assess how well the model predicts hard answers $e_i \in \mathcal{E}_{\text{hard}}$. These metrics are computed in a filtered setting [5], as described in Section 4. Given a single query Q_S with multiple hard answers $\mathcal{E}_{\text{hard}}$, we compute the MRR as the average inverse rank of each hard answer $e_i \in \mathcal{E}_{\text{hard}}$:

$$\text{MRR} = \frac{1}{|\mathcal{E}_{\text{hard}}|} \sum_{e_i \in \mathcal{E}_{\text{hard}}} \frac{1}{r_i} \tag{10}$$

where r_i denotes the rank of e_i in the filtered answer set, as described in Equation (6).

Hits@k is defined as the proportion of hard answers that are ranked within the top k positions:

$$\text{Hits@}k = \frac{|\{e_i \in \mathcal{E}_{\text{hard}} : r_i \leq k\}|}{|\mathcal{E}_{\text{hard}}|} \tag{11}$$

In our experiments, we report Hits@1, corresponding to the fraction of hard answers placed at rank 1.

Evaluation Scenarios. To evaluate the effectiveness of our explanation method, we extend the notions of necessary and sufficient explanations [25] for link prediction, to the context of explaining atoms in complex queries.

Necessary Explanations: Given a query Q_S and a hard answer $e_i \in \mathcal{E}_{\text{hard}}$ that is ranked as the top answer by the neurosymbolic model (i.e., MRR and Hits@1 are both 1.0), we extract explanations in terms of the query's atoms. An explanation is considered necessary if, upon executing the most important atom in the explanation symbolically, the hard answer e_i drops in the ranking. The effectiveness of a necessary explanation is thus measured by the decrease in MRR and Hits@1 when this atom is replaced by symbolic reasoning. We repeat this process for each hard answer in a single query, computing the change in performance metrics for each, and then average these values to obtain the metrics for the query. Finally, we report the average effectiveness across all queries of a given type.

Sufficient Explanations: As a complementary method, in this scenario, we focus on queries and hard answers that are not ranked at the top when using the symbolic approach alone (i.e., Hits@1 is 0.0 and MRR is not 1.0). The explanation is considered sufficient if executing the most important atom, as indicated by the explanation, using the neural approach leads to an increase in MRR and Hits@1 compared to the baseline where all atoms are executed symbolically. Averaging is performed across hard answers and queries in the same manner as in the necessary scenario.

Explanation Methods. As there are no existing methods to assign importance to atoms in complex queries, we introduce several baselines to enable comparison with our approach in terms of necessary and sufficient explanations: (1) *First-level:* The first level of the query execution graph refers to the atoms that contain an anchor entity (e.g., a_1 in Figure 1). In this baseline, we randomly select one atom from the first level and execute it differently from the others—symbolically in the necessary scenario and neurally in the sufficient scenario. (2) *Last-level:* This baseline is defined similarly to the first-level one, except that the randomly selected atom is chosen from the last level of the query graph (i.e., atoms containing the target variable). (3) *Random:* A randomly chosen atom in the query is executed differently. It is worth mentioning that, for some query types such as $2u$, $2i$, and $3i$, the query graph has only one level. Therefore, the results for the first-level, last-level, and random selection approaches are identical. (4) *Score-based:* The query is first run using the neurosymbolic approach, and the link prediction scores assigned to each atom along a path to the target answer are collected. For most query types, the atom with the lowest link prediction score is selected for different execution, based on the assumption that lower scores indicate greater importance of missing link prediction, while higher scores typically reflect observed links. However, when there is a union operation between atoms, such as in the $2u$ and $2u1p$ queries, we select the higher score between the atoms involved in the union, reflecting the use of t-conorms, which are complementary to the t-norms used for other operations.

(5) *CQD-SHAP:* Our proposed method computes Shapley values for all atoms with respect to a target answer and selects the atom with the highest Shapley value for different execution.

Hardware. All experiments are conducted on a machine equipped with an NVIDIA H100-20C GPU (20GB VRAM), 128GB RAM, and a 16-core CPU.

5.2 Quantitative Results

Table 2 shows the quantitative evaluation results for different query types and datasets under each explanation evaluation scenario as produced by the explanation methods defined in Section 5.1. Since there is only one level of atoms in $2u$, $2i$, and $3i$, the results of the First-level and Last-level approaches are the same as those of the Random approach for these query types. In general, CQD-SHAP outperforms the baselines in almost all cases, except for a few instances such as the sufficient evaluation for $2u1p$ queries, where it achieves results very close to the top-performing method. Considering both datasets, the explanations produced by CQD-SHAP reduce MRR by between 0.642 and 0.999 in the necessary scenario and increase MRR by between 0.205 and 0.521 in the sufficient scenario, depending on the query type.

CQD-SHAP demonstrates a clearly stronger effect compared with the best baseline in several cases. For $3i$ queries, the MRR decreases by an additional 0.274 (from -0.671 to -0.945) in the necessary evaluation and increases by 0.204 (from 0.295 to 0.499) in the sufficient evaluation on FB15k-237, while on NELL995 it changes by -0.148 and +0.153, respectively. A similar pattern is observed for $1p2i$ queries, with MRR differences of -0.201 and +0.081 on FB15k-237, and -0.128 and +0.066 on NELL995 for the necessary and sufficient evaluations, respectively.

For CQD-SHAP, we observe that moving from $2i$ to $3i$ queries increases explanation effectiveness (greater reductions in necessary evaluations and greater improvements in sufficient evaluations), while moving from $2p$ to $3p$ queries decreases explanation effectiveness. This pattern arises from the structural differences between these query types. In projection chain queries, there are multiple possible paths to an answer, so if we execute one atom differently, the other atoms can still find alternative paths to the correct answer. Therefore, adding more steps in the chain provides more recovery opportunities, allowing the modified execution to still assign a high rank to the target answer, which reduces the measured effectiveness. In contrast, in intersection queries, answers are aggregated through score products across all branches. Therefore, the most important branch (atom) acts as a critical filter, and executing it differently has a strong impact on the output. We can expect larger changes when there are more conditions to be satisfied, so the effectiveness of the explanation increases. Overall, this demonstrates that the CQD-SHAP approach correctly captures these structural characteristics of query types, while this pattern is not completely observed for other baselines.

For $2u$ queries, CQD-SHAP achieves nearly perfect necessary explanation effectiveness (-0.999) but moderate sufficient effectiveness (+0.348 for FB15k-237, +0.521 for NELL995). Interestingly, the

Table 2: Evaluation of the explanations' necessity and sufficiency on the FB15k-237 and NELL995 datasets for different query types (2p, 2u, ...) and baselines (First-level, Last-level, ...) in terms of changes in mean reciprocal rank (ΔMRR) and Hits@1 (ΔHits@1). n denotes the number of queries underlying a particular evaluation. Lower values for necessity and higher values for sufficiency are better.

Necessary evaluation (FB15k-237 dataset)																
	2p (n = 1549)		**2u** (n = 501)		**2i** (n = 1714)		**3i** (n = 2049)		**3p** (n = 1413)		**2u1p** (n = 730)		**2i1p** (n = 1774)		**1p2i** (n = 1718)	
	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1
First-level	-0.240	-0.277	-0.653	-0.712	-0.502	-0.544	-0.344	-0.388	-0.175	-0.204	-0.545	-0.600	-0.172	-0.204	-0.308	-0.352
Last-level	-0.500	-0.554	-0.653	-0.712	-0.502	-0.544	-0.344	-0.388	-0.479	-0.526	-0.644	-0.701	-0.695	-0.725	-0.420	-0.469
Random	-0.377	-0.424	-0.653	-0.712	-0.502	-0.544	-0.344	-0.388	-0.300	-0.346	-0.578	-0.631	-0.346	-0.376	-0.330	-0.378
Score-based	-0.551	-0.610	**-0.999**	**-1.000**	-0.782	-0.827	-0.671	-0.727	-0.553	-0.609	-0.794	-0.836	-0.655	-0.692	-0.587	-0.641
CQD-SHAP (ours)	**-0.685**	**-0.752**	**-0.999**	**-1.000**	**-0.931**	**-0.971**	**-0.945**	**-0.994**	**-0.665**	**-0.730**	**-0.950**	**-0.961**	**-0.857**	**-0.896**	**-0.788**	**-0.842**
Necessary evaluation (NELL995 dataset)																
	2p (n = 1501)		**2u** (n = 416)		**2i** (n = 1834)		**3i** (n = 2167)		**3p** (n = 1307)		**2u1p** (n = 757)		**2i1p** (n = 1845)		**1p2i** (n = 1834)	
	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1
First-level	-0.246	-0.265	-0.617	-0.662	-0.443	-0.501	-0.316	-0.363	-0.243	-0.269	-0.573	-0.607	-0.152	-0.174	-0.244	-0.285
Last-level	-0.569	-0.610	-0.617	-0.662	-0.443	-0.501	-0.316	-0.363	-0.362	-0.396	-0.654	-0.697	-0.668	-0.696	-0.367	-0.420
Random	-0.399	-0.430	-0.617	-0.662	-0.443	-0.501	-0.316	-0.363	-0.298	-0.338	-0.602	-0.639	-0.324	-0.349	-0.304	-0.352
Score-based	-0.739	-0.785	**-0.999**	**-1.000**	-0.773	-0.839	-0.746	-0.811	-0.577	-0.637	-0.879	-0.900	-0.728	-0.763	-0.624	-0.689
CQD-SHAP (ours)	**-0.793**	**-0.845**	**-0.999**	**-1.000**	**-0.861**	**-0.928**	**-0.894**	**-0.973**	**-0.642**	**-0.709**	**-0.982**	**-0.986**	**-0.871**	**-0.914**	**-0.752**	**-0.823**
Sufficient evaluation (FB15k-237 dataset)																
	2p (n = 4937)		**2u** (n = 5000)		**2i** (n = 4870)		**3i** (n = 4343)		**3p** (n = 4929)		**2u1p** (n = 4946)		**2i1p** (n = 4982)		**1p2i** (n = 4835)	
	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1
First-level	+0.195	+0.157	+0.177	+0.126	+0.207	+0.189	+0.295	+0.328	+0.121	+0.100	+0.066	+0.050	+0.077	+0.066	+0.100	+0.090
Last-level	+0.223	+0.148	+0.177	+0.126	+0.207	+0.189	+0.295	+0.328	+0.173	+0.110	**+0.214**	**+0.138**	+0.144	+0.083	+0.146	+0.126
Random	+0.207	+0.152	+0.177	+0.126	+0.207	+0.189	+0.295	+0.328	+0.138	+0.103	+0.119	+0.082	+0.099	+0.070	+0.124	+0.112
Score-based	+0.246	+0.172	**+0.348**	+0.249	+0.347	+0.313	+0.161	+0.184	+0.188	+0.131	+0.205	+0.132	+0.155	+0.112	+0.230	+0.201
CQD-SHAP (ours)	**+0.318**	**+0.242**	**+0.348**	**+0.250**	**+0.429**	**+0.373**	**+0.499**	**+0.511**	**+0.274**	**+0.210**	+0.205	+0.137	**+0.227**	**+0.168**	**+0.311**	**+0.264**
Sufficient evaluation (NELL995 dataset)																
	2p (n = 3996)		**2u** (n = 4000)		**2i** (n = 3750)		**3i** (n = 3269)		**3p** (n = 3991)		**2u1p** (n = 3992)		**2i1p** (n = 3967)		**1p2i** (n = 3767)	
	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1	ΔMRR	ΔHits@1
First-level	+0.223	+0.190	+0.267	+0.211	+0.190	+0.187	+0.173	+0.210	+0.174	+0.151	+0.081	+0.064	+0.073	+0.071	+0.112	+0.098
Last-level	+0.247	+0.164	+0.267	+0.211	+0.190	+0.187	+0.173	+0.210	+0.184	+0.116	**+0.254**	**+0.167**	+0.164	+0.099	+0.146	+0.119
Random	+0.233	+0.175	+0.267	+0.211	+0.190	+0.187	+0.173	+0.210	+0.174	+0.137	+0.134	+0.093	+0.104	+0.082	+0.127	+0.116
Score-based	+0.319	+0.236	**+0.521**	**+0.412**	+0.325	+0.302	+0.336	+0.371	+0.277	+0.217	+0.245	+0.165	+0.188	+0.141	+0.218	+0.184
CQD-SHAP (ours)	**+0.388**	**+0.306**	**+0.521**	**+0.412**	**+0.401**	**+0.361**	**+0.489**	**+0.516**	**+0.356**	**+0.286**	+0.242	+0.165	**+0.248**	**+0.189**	**+0.284**	**+0.235**

score-based method shows very similar performance, suggesting a fundamental characteristic of union queries. Examining the dataset reveals that 2u queries typically ask for semantically distinct entity types (e.g., "an athlete OR a city"), meaning the target naturally belongs to only one branch. Consequently, the link corresponding to that branch consistently receives the highest score and is correctly identified as the most important atom by both explanation methods. In the necessary case, switching this critical link from neural to symbolic execution causes the target to drop dramatically, as the top-ranked answers now come mostly from the other branch, explaining the near-perfect reduction. However, in the sufficient case, executing only the most important link neurally cannot guarantee the target reaches rank 1, because multiple other entities from the same branch may be predicted with higher scores by the neural model. Notably, CQD-SHAP achieves similar or slightly better results than the score-based method, demonstrating that it effectively captures the importance structure of union queries.

5.3 Case Study

By investigating the insights provided by CQD-SHAP explanations for certain query types, we aim to validate their effectiveness. An example of a 2i query involving the intersection of two projections is shown in Figure 2. Suppose we seek to explain `Paul Weller` as a hard answer to this query. When the query is executed by CQD, this answer has a rank of 61 (after excluding all other answers according to Equation 5). The Shapley values for the two atoms in this query, with respect to this particular answer, are +123.5 and -128.5. This indicates that leveraging neural inference for the first atom has a positive impact on improving the rank, while the effect is the opposite for the second atom.

This can be validated by examining the existing links in the graph. The link (`Rock music`, `/music/genre/artists`, `Paul Weller`) already exists in the graph, so inference is unnecessary. Although the CQD link predictor can still assign a high likelihood to this answer, the existence of many possible answers for this link causes the rank to worsen as compared to symbolic reasoning. In contrast, the link

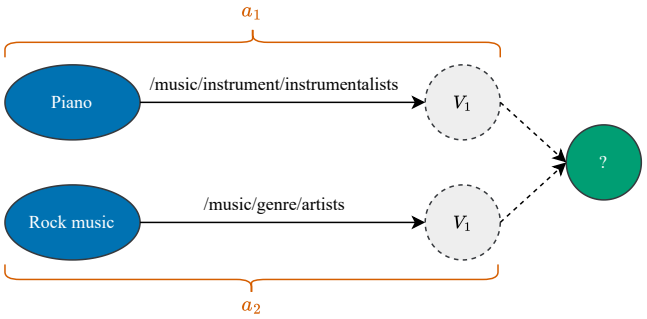

Figure 2: An example of a $2i$ query in the FB15k-237 test data. The query asks for piano instrumentalists who are also artists in the Rock music genre. This query has 22 hard answers (e.g., `Serj Tankian`, `Jon Bon Jovi`) and 113 easy answers.

(`Piano`, `/music/instrument/instrumentalists`, `Paul Weller`) does not exist in the graph, so leveraging CQD enables the model to predict this answer at a better rank.

If the query is executed in a way that aligns with the insights from the Shapley values—running the first atom neurally and the second atom symbolically—the rank of the target answer improves to 33. Furthermore, according to Equation (9), summing these Shapley values yields -5, which exactly matches the difference in ranking obtained by subtracting the neurosymbolic result (rank 61) from the symbolic result (rank 56). In conclusion, this demonstrates that, for this particular answer, executing the query entirely with the neurosymbolic approach results in a worse ranking than symbolic reasoning, and the Shapley values clearly indicate which parts of the query are responsible and by how much.

It is also worth mentioning that this approach can be used to explain false answers. In the same example, CQD predicts the entity `Billy Joel` at rank 5, even though this is not a correct answer (neither easy nor hard). For this case, the Shapley values are +219 and +39, indicating that the first atom contributed most to this incorrect prediction. Such explanations can help users identify and debug cases where the link predictor is underperforming or returning undesirable results.

6 Discussion

Based on the results reported in Table 2, it can be concluded that the most important query atom found by CQD-SHAP is usually necessary to be executed neurally in order to achieve a better ranking for the desired target, but it is not sufficient on its own. Therefore, executing only the most important atom neurally while retrieving the remaining parts of the query symbolically does not guarantee improved performance. Similarly, executing all atoms with negative Shapley values symbolically and all those with positive values neurally will not necessarily lead to a better ranking in every case. This is because Shapley values reflect the *average* marginal contribution of each player across all possible coalitions.

Based on the definition of the sufficient evaluation, each hard query is expected to be considered in this evaluation, as such queries contain at least one hard answer. However, a closer inspection of the number of evaluated queries (n) in the sufficiency results in Table 2 shows that this number is smaller than the total number of hard queries for each query type, except for $2u$. Further examination of the datasets revealed that some answers labeled as hard in the CQD work are, in fact, not genuinely hard and can be retrieved using the symbolic approach. This observation raises concerns about the query generation procedure in that work, as similar issues are also noted in [13]. For instance, in a $2u1p$ query, the only missing link may lie in one branch of the union, yet the target answer can still be derived from the other branch. Future research could benefit from developing more carefully designed datasets for complex query answering, ensuring that evaluation benchmarks better capture the intended query difficulty. However, according to our definitions of the explanation evaluation scenarios, such cases are already excluded from consideration.

The runtime for computing Shapley values depends on the size of the KG and the number of atoms in the query. In our experiments, the average computation time for a single query and answer was 147 ms on FB15k-237 and 300 ms on NELL995.

Future work could focus on redefining the goal of explanations in complex query answering. Even when keeping the focus on query atom explanations, the Shapley game could be designed differently, for example, by exploring alternative cooperative settings instead of switching between execution methods. Another direction is to change the explanation level. Instead of working at the query atom level, future studies could aim to identify the most influential triples in the KG that contribute to a target answer, extending link prediction explanation methods to the CQA domain. Future research could also focus on developing global explanations of the system, for example, by aggregating the local explanations produced by our work to obtain a broader understanding of the model's behavior.

7 Conclusion

We defined an explanation goal in the context of complex query answering and presented CQD-SHAP, a Shapley-based method that quantifies the contribution of each query atom to the ranking of a target answer by assessing the benefit of neural execution over symbolic retrieval. We also redefined the concepts of necessary and sufficient explanations for this domain and demonstrated the effectiveness of CQD-SHAP through quantitative evaluations on two well-studied knowledge graphs. The results show that CQD-SHAP achieves the best effectiveness in both evaluation scenarios and across both datasets for almost all query types. The most important atoms identified by this method are generally necessary but not as strongly sufficient to improve rankings. An example query and its corresponding Shapley values illustrated how the approach reveals which parts of a query rely more on the neural model and can also assist in debugging incorrect answers. In the discussion section, we further explained that executing atoms solely based on the sign of their Shapley values does not guarantee improvement, as these values represent average contributions across all possible execution combinations. Finally, we highlighted a potential issue in the original query generation process and suggested several directions for future work, including redefining explanation goals, designing alternative Shapley games, and developing global explanations beyond single-answer interpretations.

References

[1] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2021. Complex Query Answering with Neural Link Predictors. In *ICLR*. OpenReview.net.

[2] Erik Arakelyan, Pasquale Minervini, Daniel Daza, Michael Cochez, and Isabelle Augenstein. 2023. Adapting Neural Link Predictors for Data-Efficient Complex Query Answering. In *NeurIPS*.

[3] Meghyn Bienvenu, Diego Figueira, and Pierre Lafourcade. 2024. Shapley Value Computation in Ontology-Mediated Query Answering. In *KR*.

[4] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*. ACM, 1247–1250.

[5] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.

[6] Heng Chang, Jiangnan Ye, Alejo Lopez-Avila, Jinhua Du, and Jia Li. 2024. Path-based Explanation for Knowledge Graph Completion. In *KDD*. ACM, 231–242.

[7] Adriane Chapman and H. V. Jagadish. 2009. Why not?. In *SIGMOD Conference*. ACM, 523–534.

[8] Tanya Chowdhury, Yair Zick, and James Allan. 2025. RankSHAP: Shapley Value Based Feature Attributions for Learning to Rank. In *ICLR*. OpenReview.net.

[9] Daniel Daza and Michael Cochez. 2020. Message Passing for Query Answering over Knowledge Graphs. *CoRR* abs/2002.02406 (2020).

[10] Caglar Demir, Michel Wiebesiek, Renzhong Lu, Axel-Cyrille Ngonga Ngomo, and Stefan Heindorf. 2023. LitCQD: Multi-hop Reasoning in Incomplete Knowledge Graphs with Numeric Literals. In *ECML/PKDD*. Springer, 617–633.

[11] Daniel Deutch, Nave Frost, Benny Kimelfeld, and Mikaël Monet. 2022. Computing the Shapley Value of Facts in Query Answering. In *SIGMOD Conference*. ACM, 1570–1583.

[12] Sneha Gathani, Peter Lim, and Leilani Battle. 2020. Debugging Database Queries: A Survey of Tools, Techniques, and Users. In *CHI*. ACM, 1–16.

[13] Cosimo Gregucci, Bo Xiong, Daniel Hernández, Lorenzo Loconte, Pasquale Minervini, Steffen Staab, and Antonio Vergari. 2024. Is Complex Query Answering Really Complex? *CoRR* abs/2410.12537 (2024).

[14] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Querying Complex Networks in Vector Space. *CoRR* abs/1806.01445 (2018).

[15] Maria Heuss, Maarten de Rijke, and Avishek Anand. 2025. RankingSHAP - Faithful Listwise Feature Attribution Explanations for Ranking Models. In *SIGIR*. ACM, 381–391.

[16] Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. 2020. The Shapley Value of Tuples in Query Answering. In *ICDT (LIPIcs, Vol. 155)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 20:1–20:19.

[17] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *NIPS*. 4765–4774.

[18] Debasis Mohapatra, Satnam Singh, Rupsalin Pradhan, and Soumya Ranjan Behera. 2023. Integrating structural features for link prediction in directed social network using supervised machine learning with SHapley Additive exPlanations. In *OCIT*. IEEE, 468–472.

[19] Simon Ott, Melisachew Wudage Chekol, Christian Meilicke, and Heiner Stuckenschmidt. 2025. Training-Free Score Calibration for Complex Query Decomposition. In *ESWC*, Vol. 15718. Springer, 188–207.

[20] Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. 2019. Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications. In *NAACL-HLT*. ACL, 3336–3347.

[21] Venetia Pliatsika, João Fonseca, Kateryna Akhynko, Ivan Shevchenko, and Julia Stoyanovich. 2025. ShaRP: Explaining Rankings and Preferences with Shapley Values. *Proc. VLDB Endow.* 18, 11 (2025), 4131–4143.

[22] Hongyu Ren, Mikhail Galkin, Michael Cochez, Zhaocheng Zhu, and Jure Leskovec. 2023. Neural Graph Reasoning: Complex Logical Query Answering Meets Graph Databases. *CoRR* abs/2303.14617 (2023).

[23] Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. In *ICLR*. OpenReview.net.

[24] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. 2021. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. *ACM Trans. Knowl. Discov. Data* 15, 2 (2021), 14:1–14:49.

[25] Andrea Rossi, Donatella Firmani, Paolo Merialdo, and Tommaso Teofili. 2022. Explaining Link Prediction Systems based on Knowledge Graph Embeddings. In *SIGMOD Conference*. ACM, 2062–2075.

[26] Lloyd S. Shapley. 1953. A Value for n-Person Games. In *Contributions to the Theory of Games II*. Princeton University Press, 307–317.

[27] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *CVSC*. ACL, 57–66.

[28] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[29] Meihong Wang, Linling Qiu, and Xiaoli Wang. 2021. A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry* 13, 3 (2021), 485.

[30] Zihao Wang, Yangqiu Song, Ginny Y. Wong, and Simon See. 2023. Logical Message Passing Networks with One-hop Inference on Atomic Formulas. In *ICLR*. OpenReview.net.

[31] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *WWW*. ACM, 515–526.

[32] Eyal Winter. 2002. The Shapley value. *Handbook of Game Theory with Economic Applications* 3 (2002), 2025–2054.

[33] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *EMNLP*. ACL, 564–573.

[34] Bohui Zhang, Filip Ilievski, and Pedro A. Szekely. 2022. Enriching Wikidata with Linked Open Data. In *Wikidata@ISWC (CEUR Workshop Proceedings, Vol. 3262)*. CEUR-WS.org.

[35] Shichang Zhang, Jiani Zhang, Xiang Song, Soji Adeshina, Da Zheng, Christos Faloutsos, and Yizhou Sun. 2023. PaGE-Link: Path-based Graph Neural Network Explanation for Heterogeneous Link Prediction. In *WWW*. ACM, 3784–3793.

A Runtime

Table 3: Average runtime (in milliseconds) per query type for computing Shapley values of a given query–answer pair on the FB15k-237 and NELL995 datasets.

	2p	2u	2i	3i	3p	2u1p	2i1p	1p2i
FB15k-237	41	29	30	91	306	129	149	147
NELL995	100	85	81	232	474	287	327	335

The runtime for computing Shapley values for all atoms of a query with respect to an answer is shown in Table 3. Although even the most time-consuming computation takes less than half a second, the runtime on NELL995 is consistently higher than on FB15k-237, as NELL995 is a larger knowledge graph.

B Query Statistics

Table 4: Average number of answers per query. Hard answers are subdivided in necessary (Nec.) and sufficient (Suff.).

	FB15k-237				NELL995			
	Hard			Easy	Hard			Easy
	Total	Nec.	Suff.	Total	Total	Nec.	Suff.	Total
2p	19.71	2.66	17.04	111.66	9.09	2.34	6.74	47.49
2u	3.46	0.17	3.28	32.12	2.08	0.12	1.95	12.28
2i	9.78	0.69	9.09	59.25	7.47	0.89	6.57	22.86
3i	7.55	0.58	6.96	41.33	5.76	1.00	4.75	10.15
3p	31.94	2.58	29.35	183.33	15.58	1.59	13.98	49.76
2u1p	11.29	0.29	10.99	116.40	8.65	0.36	8.28	53.73
2i1p	69.54	5.12	64.41	468.10	52.39	9.34	43.04	257.09
1p2i	40.51	4.59	35.91	218.19	33.43	6.82	26.61	111.54

Table 4 summarizes the statistics of different query structures across the FB15k-237 and NELL995 datasets. For each query type, we report the average number of easy and hard answers based on the validation graph, where easy answers can be retrieved from it and hard answers cannot (i.e., only retrievable from the test graph). In addition, we compute how many of the hard answers satisfy the necessary and sufficient conditions. The necessary scenario corresponds to the number of hard answers that CQD can correctly predict at rank 1, whereas the sufficient scenario refers to those hard answers that CQD fails to predict at rank 1.