LinE: Logical Query Reasoning over Hierarchical Knowledge Graphs

Zijian Huang  
The University of Auckland  
zhua764@aucklanduni.ac.nz

Meng-Fen Chiang  
The University of Auckland  
meng.chiang@auckland.ac.nz

Wang-Chien Lee  
The Pennsylvania State University  
wlee@cse.psu.edu

ABSTRACT
Logical reasoning over Knowledge Graphs (KGs) for first-order logic (FOL) queries performs the query inference over KGs with logical operators, including conjunction (∧), disjunction (∨), existential quantification (∃), and negation (¬), to approximate true answers in embedding spaces. However, most existing work imposes strong distributional assumptions (e.g., Beta distribution) to represent entities and queries into presumed distributional shape, which limits their expressive power. Moreover, query embeddings are challenging due to the relational complexities in multi-relational KGs (e.g., symmetry, anti-symmetry, and transitivity). To bridge the gap, we propose a logical query reasoning framework, LinE Embedding (LinE), for FOL queries. To relax the distributional assumptions, we introduce the logic space transformation layer, which is a generic neural function that converts embeddings from probabilistic distribution space to LinE embeddings space. To tackle multi-relational and logical complexities, we formulate neural relation-specific projections and individual logical operators to truthfully ground LinE query embeddings on logical regularities and KG facts. Lastly, to verify the LinE embedding quality, we generate a FOL query dataset from WordNet, which richly encompasses hierarchical relations. Extensive experiments show superior reasoning sensitivity of LinE on three benchmarks against strong baselines, particularly for multi-hop relational queries and negation-related queries.

CCS CONCEPTS
• Computing methodologies → Knowledge representation and reasoning
• Theory of computation → Logic.

KEYWORDS
Knowledge representation learning; Logical query reasoning

1 INTRODUCTION
Motivations. Thanks to the availability of large-scale knowledge graphs (KGs), such as Freebase [4], WordNet [20], NELL [5], and YAGO [33], recent advances in knowledge graph representation learning have sparked significant research interests in logical query reasoning over multi-relational KGs. Understanding the relational properties in a collection of structured knowledge facts plays a pivotal role in the rapidly growing field of answering complex logical queries. Numerous efforts have been devoted to modeling logical operators or introducing new operators for first-order logical (FOL) queries on incomplete KGs [12, 15, 23, 24]. For instance, GQE [15] models both relational projections and set intersection operators by training neural networks as transformation functions. Query2Box [23] and BetaE [24] propose box embeddings and beta embeddings, two novel knowledge representations, to better formalize entities and logical queries in their respective representation spaces, while proposing a sophisticated attention mechanism to accurately capture set intersection behavior. Despite the recent progress, learning robust knowledge representations to better capture both relational and logical behavior, however, remains a challenging problem, with open issues in the following aspects: (i) expressive power on knowledge representations, (ii) preservation of closure properties under relational/logical operations, and (iii) support for both hierarchical and non-hierarchical logical query reasoning.

Firstly, most recent logical query reasoning models rely on some crucial assumptions on knowledge representations to enhance the expressive power of logical operators. GQE [15] formalizes entities and queries into a vector space, assuming that logical and relational behavior can be captured by a single value at each dimension. Query2Box [23] proposes box embeddings with centre and distance to box border to improve representation quality. BetaE [24]
explores Beta distribution to improve representation quality, assuming that logical and relational behaviors can be captured by a Beta distribution at each dimension in a distributional representation space. However, these strong assumptions lead to limitations in the expressiveness and violation tolerance of logical and relational behavior due to their relatively fixed shapes of assumed distribution. For example, the relational hierarchy in the triple ("North America", contains, "USA") is not truthfully reflected in the Beta space as shown in Figure 1(a), where there is no clear indication of one probability distribution ("North America" in red) encompasses the other ("USA" in blue). To obtain generic yet versatile representations to accurately capture logical and relational behavior, it is essential to equip the representations with better fault tolerance in logical and relational operations.

Secondly, preserving the closure property under relational/logical operations in knowledge representations is critical for enabling compositional computation of complex logical queries in the reasoning process. Most prior works only support a subset of logical operations. For example, region-based representations [23] or distributional representations [24] do not preserve closure property under negation operation. Therefore, we need a knowledge representation that can comprehensively preserve the closure property under the logical and relation operations.

Lastly, most prior works tackle logical queries without considering relation hierarchy in KGs. Hierarchical relations, with anti-symmetry and partial-order transitivity, are intrinsic in KGs and thus are natural targets of logical queries. Different relational properties require different operations in the representation space to answer both logical queries with and without hierarchical relations. Nonetheless, relational annotations are usually not available to explicitly indicate a relation is hierarchical or not. We thus cannot trivially resort to supervised learning approaches to support multi-relational logical query reasoning.

**Research Contribution.** To overcome these challenges, we propose a novel KG reasoning framework based on **LinE Embedding** (LinE), for answering FOL queries over KGs. Specifically, to relax the distributional assumptions we propose to transform logical embedding from the Beta distribution space into a novel logic space, referred to as the LinE space, where we design competitive logical functions for logical operators while maintaining the closure property in the LinE space. To tackle multi-relational and logical complexities, we design an unsupervised learning approach to regulate query and KG entity embeddings in the LinE space, inspired by [6], using both curvature estimate and Krackhardt score to estimate the hierarchical relations. In addition, we rigorously generate a benchmark based on a hierarchical KG (WN18RR) for comprehensive study on reasoning ability of LinE framework. We conduct experiments over (i) the generalization reasoning setting over 14 types of logical queries, and (ii) three benchmark datasets, including the Freebase, NELL, and WordNet which encompasses logical queries rich in hierarchical relations. In addition, we explore multiple formulations of logical operators and adopt the most competitive formulations. The main contributions are as follows.

- We propose a logical query reasoning framework (LinE) to preserve multi-relational complexities and logical regularities in the proposed LinE space. In particular, we propose logical space transformation and logical query inference to ground LinE embeddings to mixed relational and logical regularities.
- We design neural relation-specific projections to sensitively capture hierarchical and non-hierarchical relational properties in the KG guided by transitivity (curvature estimate) and anti-symmetry estimates (Krackhardt score).
- We generate a dataset of first-order logical queries, which heavily involve hierarchical relations from the benchmark KG WN18RR.
- In extensive experiments on three benchmark KGs, LinE shows superior reasoning sensitivity to answer logical queries with and without hierarchical relations against dominant baselines.

2 **PRELIMINARIES**

We briefly review notions for first-order logic query and relational properties. Next, we introduce the classic probabilistic representation space, Beta distribution. Lastly, we formalize the problem of logical query reasoning in multi-relational KGs.

2.1 **Logical Query Reasoning**

A knowledge graph \(\mathcal{G} = (\mathcal{V}, \mathcal{R})\) is a multi-relational graph, where \(v \in \mathcal{V}\) represents an entity, and each relation type \(r \in \mathcal{R}\) is a binary function \(r: \mathcal{V} \times \mathcal{V} \rightarrow \{True, False\}\) that indicates the existence of a type-\(r\) directed edge between a pair of entities. A multi-relational KG \(\mathcal{G}\) can be represented by a set of knowledge triples \(\mathcal{K} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}\), which often exhibits multiple relational properties, such as symmetry, anti-symmetry and transitivity (Figure 2).

**Hierarchical Relation.** Relations can be divided into two categories, non-hierarchical and hierarchical relations, according to their relational properties. Non-hierarchical relations do not simultaneously exhibit anti-symmetric and transitive relational properties. For example, the relation `adjoin` in the triple ("England", `adjoin`, "Scotland") in Figure 2 is a non-hierarchical relation as the triple remains factually true after the exchange of positions between the subject and object entities. Hierarchical relations simultaneously exhibit anti-symmetric and transitive relational properties. The relation `contains` in the triple ("United Kingdom", `contains`, "England") in Figure 2 is an example of hierarchical relation as it is simultaneously anti-symmetric and transitive.

**Hierarchical Knowledge Graph.** KGs can be either hierarchical or non-hierarchical based on the relational properties in the graphs. Hierarchical KGs contain at least one hierarchical relation, while non-hierarchical KGs contain no hierarchical relation.
First-Order Logic Queries. First-order logic (FOL) queries are queries formulated by logical operators, including conjunction (∧), disjunction (∨), existential quantification (∃) and negation (¬). An FOL query \( q \) can be expressed in disjunctive normal form (DNF), which is a disjunction of conjunctions as follows:

\[
q[V_i] = V_i : \exists V_1, \ldots, V_w : c_1 \lor c_2 \lor \cdots \lor c_m,
\]

where \( V_i \) is the target variable, \( \{V_i\}_{1 \leq i \leq w} \) are existentially quantified bound variables, and \( \{c_i\}_{1 \leq i \leq n} \) are conjunction queries. Target variable \( V_i \) indicates the final answer after the reasoning process is completed. Each existentially quantified bound variable \( V_i \) indicates the intermediate results during the reasoning process. A conjunction query \( c_i \) comprises of one or more atomic relation queries \( \land_{j=1}^m c_{ij} \).

Atomic Relation Query. An atomic relation query \( c_{ij} \) is a relation projection between a pair of entity sets. Each entity set can be a non-variable entity, an intermediate variable, or a target variable. Formally, the atomic relation projection is defined as one of the following forms:

\[
e_{ij} = r(a_0, V) \text{ or } r(-a_0, V) \text{ or } r(V', V) \text{ or } r(-V', V), \text{ } V \neq V'.
\]

where \( a_0 \in V_{ij} \) is a non-variable anchor entity, \( V \in \{V_i\} \cup \{V_i\}_{1 \leq i \leq n} \) is in the complete set of variables, \( V' \in \{V_i\}_{1 \leq i \leq w} \) is in the set of intermediate variables, and \( r \in R \) is a relation type. An example of atomic relation query \( q' \) "list all participating countries of 2008 Beijing Summer Olympics" is illustrated in Figure 3(a).

Compositional Relation Query. A compositional relation query comprises of multiple atomic relation queries in an FOL query \( q \), which is typically structured as a computation graph. Figure 3(b) illustrates the reasoning steps for the compositional relation query \( q' \) "list all places of countries that participated in Beijing 2008 Summer Olympics and have not won any medals in Helsinki 1952 Summer Olympics". Blue circles indicate the anchor entities, "2008 Olympics" and "1952 Olympics Medal". Green circle indicates the target variable \( V_i \) for the final answer to query \( q \). Grey circles indicate the intermediate variables \( (V_1, V_2, V_3) \), \( q \) comprises of three atomic relation queries, participant, awardedTo and contains. To derive the answer, our reasoner firstly derives intermediate results for \( V_1 \) and \( V_2 \) from anchor entities. Next, our reasoner derives the complement of \( V_2 \), denoted as \( V'_2 \), via a negation operator. Then, the reasoner derives the intersection of \( V_1 \) and \( V'_2 \), resulting in \( V_3 \) ("Fiji") by an intersection operator. Finally, the reasoner derives the final answer for \( V_1 \) ("Suva") by a relational projection from \( V_3 \) via contains relation.

Hierarchical Logical Query. A logical query may involve multiple relations. We refer to the FOL queries that consist of at least one hierarchical relation \( (R_F) \) as hierarchical logical queries; whereas the FOL queries comprise of purely non-hierarchical relations \( (R_{\neg F}) \) are referred to as non-hierarchical logical queries.

Hierarchy Estimates. A hierarchy involves relations with anti-symmetry and transitivity properties [14, 17, 21], such as contains, hypernym, has_part. KGs are typically structured with mixed relations without explicit indications of hierarchical property. Therefore, we need to estimate the anti-symmetry and transitivity to distinguish hierarchical \( (R_F) \) from non-hierarchical relations \( (R_{\neg F}) \). Motivated by this, we explore two metrics, Krackhardt hierarchy score \( (KH_{G}) \) [17] and curvature \( (\epsilon_g) \) [14], to estimate the degrees of anti-symmetry and transitivity for each relation \( r \in R \), respectively. A relation \( r \) is considered highly hierarchical if its induced relation graph \( G_r \) (i.e., the graph structured only with relation \( r \)) has higher anti-symmetry scores \( (KH_{G_r}) \) and higher transitivity scores \( (\epsilon_{G_r}) \), and vice versa [14]. Our reasoner is guided by anti-symmetry scores and transitivity scores to learn respective relational regularities in KGs. (more details in Section 3.3). We detail the anti-symmetry and transitivity estimates in Appendix.

2.2 Probabilistic Representation Space

In a probabilistic representation space, both the entities and queries \( S \) are viewed as probabilistic embeddings. For instance, BetaE [24] formulates a set of entities \( S \subseteq \mathcal{V} \) as a Beta embedding, which is essentially a Beta distribution \( B(\cdot) \) shaped by two parameters \( \alpha \) and \( \beta \). The probability density function (PDF) controlled by \( (\alpha, \beta) \) is defined as \( p(x) = \frac{x^\alpha(1-x)^\beta}{B(\alpha, \beta)} \). An \( h \)-dimensional Beta embedding of an entity set \( S \in \mathbb{R}^{2\times h} \) consists of \( h \) independent Beta distributions on the interval \([0,1] \), denoted as \( B^S = \{(\alpha_i^1, \beta_i^1), \ldots, (\alpha_n^h, \beta_n^h)\} \) with \( x \in [0,1] \). Note that a single entity is equivalently a set of a single element, and thus each entity itself in \( h \)-dimensional Beta distribution space is also an \( h \)-dimensional Beta embedding. A query \( q \) after executing logical operators (\( \land, \lor, \exists \text{ and } \neg \)) and relational projections (\( r \in R \)) in the \( h \)-dimensional Beta representation space remains a \( h \)-dimensional Beta embedding, thanks for the closure property of BetaE. Nevertheless, BetaE is bounded by a strong distributional assumption, which limits representations into a presumed distributional shape. BetaE clearly limits its expressive power to (i) sensitively deal with mixed relational regularities (e.g., relational hierarchy), and (ii) represent entities and queries beyond Beta shapes. We therefore tackle the challenges of limited expressiveness by relaxing the distributional assumption and designing a more relation-sensitive representation space.

2.3 Problem Statement

Given a multi-relational KG \( G = (V, R) \) and an FOL query \( q \) with relational and logical compositions, our goal is to learn hierarchical
probabilistic representation space (BetaE)

We present our approach to design a novel neural logical reasoning framework that supports logical operators (logical OR, logical AND, existential and universal quantification) and relation-specific projections for expressible specification of the FOL query. We then return a set of entities (Ṽ ⊆ V) that satisfy q over the facts captured by G.

3 LINE EMBEDDINGS

We present our **Line** Embeddings model (LinE) that performs compositional relational projections and logical operations for logical queries. We first give an overview of the reasoning pipeline. We detail the key components and our training objectives.

3.1 Reasoning Pipeline Overview

The proposed reasoning model, LinE, supports a complete set of first-order logic operations and the relation-specific projections in the LinE space. Figure 4 illustrates the reasoning pipeline. Given a KG (G), a set of query answering (QA) pairs as a training set (q₁, q₂, ..., qₙ) and a set of queries {q₁[V₁], q₂[V₂], ..., qₙ[Vₙ]}, we train a logical query reasoner to answer FOL queries by performing relational projections and logical operations in order to closely approach the true answers in the LinE space. For each query qᵢ as input, its computation graph is generated to indicate the reasoning steps. We follow the reasoning steps to execute logical operations in the query and obtain KG entity embeddings in the BetaE space.

3.2 Logic Space Transformation

We introduce logic space transformation (LST), which transforms representations from a probabilistic representation space (source) to the LinE space (target) with enhanced expressive power. Specifically, we propose a neural transformation from the source to target logic space to i) preserve a diverse aspect of properties in Beta distribution; ii) support multi-relational projections and logical operations in the LinE space.

Figure 4: The model architecture of LinE. The green color indicates the training process, while the yellow color indicates the real-time testing process for unseen queries. (a) For training, LinE takes as input the KG and pairs of logical queries and answers. (b) Given the computation graph for each query, the entity embedding initialization (EEI) component generates the initial embeddings for KG entities in a probabilistic representation space (blue area). (c) We present the logic space transformation (LST) to better capture multi-relational and logical regularities. The statistical view of the initial entity embeddings is thus generated to learn the entity embeddings in the LinE representation space (LinE space). (d) The logical query inference (LQI) component learns a set of relation-specific neural functions and refines the embeddings for KG entities and queries based on multi-relational and logical regularities in the LinE space. (e) To optimize, two training objectives, query inference loss and reasoning loss, are proposed to jointly preserve multi-relational and logical regularities in the LinE space.
LinE Representation Space. In the LinE space, both the entities and queries $S$ are represented as LinE embeddings, a low-dimensional representation, whereby each dimension in its general form is expressed as a sequence of $k$ values. LinE embeddings are no longer restricted by any distributional shape like $\mathcal{B}(\cdot)$. Formally, an $h$-dimensional LinE embedding for the set $S$ is defined as $L^S = [(p^S_{j,1}, p^S_{j,2}, \cdots, p^S_{j,k})]_{j=1}^h \in \mathbb{R}^{k \times h}$ across $k$ positions in the range $[0, 1]$ per dimension $1 \leq j \leq h$. For instance, suppose we want to learn 200-dimensional LinE embeddings with a sequence of three values per dimension. A logic space transformation function predicts values at three positions (e.g., $(0.25, 0.50, 0.75)$) per dimension in the LinE space, and forms the 200-dimensional LinE embedding $\{(p_{j,0.25}, p_{j,0.50}, p_{j,0.75})\}_{j=1}^{200}$. Figure 5 illustrates the comparison between a Beta embedding and the LinE embedding. Note that LinE embedding preserves the closure property because the following properties hold. First, each entity itself in the LinE space is an $h$-dimensional LinE embedding because an entity is equivalently a set with a single element. Second, a query $q$, after executing logical operators ($\land$, $\lor$, $\exists$ and $\neg$) and relational projections ($r \in \mathcal{R}$) as appropriate in the LinE space, returns a set of answer entities, which is also a LinE embedding in $h$-dimensional LinE space. In this work, we learn neural transform functions in two steps: (i) augmenting the entity features based on the $n$-dimensional Beta embedding in the source logic space; and (ii) applying an entity-wise learnable function to derive LinE embeddings.

Statistical View Generation. To obtain effective features, we explore multiple features, which statistically describe the shapes of Beta distributions. The neural transformation function optimizes the network parameters so that two entities with the same shape features are likely to share the same LinE embeddings. Specifically, we consider the following shape features.

- $\alpha$ and $\beta$: $\alpha$ and $\beta$ are explicit parameters of a Beta distribution. Given an $h$-dimensional Beta embedding $B^S_i$ for entity set $S_i$, we obtain the $h$ pairs of $(\alpha, \beta)$ parameters as input features for the neural transformation function as follows.
  \[
  B^S_i = [(\alpha^S_j, \beta^S_j)]_{j=1}^h \in \mathbb{R}^{2 \times h}
  \]

- Mean and Variance: Mean ($\mu(X)$) measures the central tendency of a Beta distribution. Variance ($\text{var}(X)$) measures the statistical dispersion of a distribution. The calculation of mean and variance can be found in Appendix. Given an $h$-dimensional Beta embeddings $B^S_i$ for set $S_i$, we obtain $h$ pairs of mean and variance for $h$ Beta distributions as follows.
  \[
  p^{B^S_i} = [(\mu(X_j)^S_i, \text{var}(X_j)^S_i)]_{j=1}^h \in \mathbb{R}^{2 \times h}.
  \]

Neural Transformation. To learn an entity-wise neural transformation function, we adopt a Multi-Layer Perceptron (MLP) with the shape features as input. Formally, given $n$ entity sets $S = \{S_1, \cdots, S_n\}$, the shape features extracted from $n$ Beta embeddings $F = \{F_1, \cdots, F_n\}$ is fed to an entity-wise MLP, which generates $n$ LinE embeddings $L = \{L_1, \cdots, L_n\}$. A neural network that converts a set of Beta embeddings to a set of LinE embeddings is trained as follows.

\[
L = MLP(F)
\]

\[
L^S_i = \{p^{B^S_i} \mid [p^{B^S_i}], \forall S_i \in S \}
\]

\[
L^S_i = \{(\mu(X_j)^S_i, \text{var}(X_j)^S_i)]_{j=1}^h \in \mathbb{R}^{2 \times h}, \forall S_i \in S \}
\]

where $F_i \in F$ is the final shape feature concatenated from any of $B^S_i$ and $p^{B^S_i}$ for entity set $S_i$.

3.3 Logical Query Inference

Logical query inference (LQI) grounds in knowledge triples and logical regularities in the LinE space to support compositional logical query inference. Given the computation graph for a query $q$, LQI derives the final LinE embedding for $q$ by executing logical operators ($\land$, $\lor$, $\exists$ and $\neg$) and relation-specific projections ($r \in \mathcal{R}$) in the computation graph. We describe each logical operator and relation-specific projection in the LinE space, including relational projection $X_r$, intersection $L_{\text{Inter}}$, negation $L_{\text{Neg}}$, and union $L_{\text{Union}}$.

Relation-Specific Projection. Relations in multi-relational KGs have diverse properties: symmetric (e.g., $\text{adjoints}$), anti-symmetric (e.g., $\text{is_a_member_of}$), and transitive relations (e.g., $\text{contains}$). Preserving each type of relational behavior therefore requires a range of neural functions to capture relational properties in latent representation space.

To capture diverse relational properties in KGs, each atomic relational projections of any forms in Eq. (2) in a compositional logical query is realized by a relation-specific projection $X_r$ for each $r \in \mathcal{R}$. For illustration, given an atomic relation query $r(u_a, V)$, a relation-specific projection learns a projection function $X_r$ that takes $u_a$ as
input and projects $v_a$ closer to the representations of a set of tail entities, corresponding to the existentially quantified bound variables $V$, in the LinE space. A projection function $X_r$ is formulated as a neural network for the relation type $r \in \mathcal{R}$ by a Multi-Layer Perceptron $\text{MLP}_r$ with ReLU as activation function as follows.

$$L_i^S(S_i, r) = \text{MLP}_r(L_S^S)$$

where $L_S^S$ and $L_i^S(S_i, r)$ denote the LinE embedding of an entity set $S_i = \{v_i\}$ and the estimated LinE embedding for the entity set $S_j = \{v_j\}$ after projection via relation $r$ from $v_i$, with respect to the knowledge triples $r(S_i, S_j) \in \mathcal{K}$ in the LinE space.

**Relational Regulations.** To sensitively capture mixed structural regularities in a KG, we use hierarchy estimates in Section 2.1 to disjointly distinguish hierarchical (R$_H$) and non-hierarchical relations (R$_N$). Accordingly, we propose the following relational regulations $D_{\mathcal{R}_H}$ and $D_{\mathcal{R}_N}$ to preserve hierarchical relations (R$_H$) and non-hierarchical relations (R$_N$), respectively, where $R_H \cap \mathcal{R}_N = \emptyset$ and $R_H \cup \mathcal{R}_N = \mathcal{R}$.

To preserve the hierarchical properties in the LinE space, the hierarchical violation against knowledge triples $\mathcal{K}$ is approximated by minimizing the following order violation.

$$D_{\mathcal{R}_H} = \sum_{r(S_i, S_j) \in \mathcal{K}_{r}} \max(0, L_i^S - L_j^S(S_i, r)), \forall r \in \mathcal{R}_H$$

where $\mathcal{K}_{r} = r(S_i, S_j) \subseteq \mathcal{K}$ is the set of triple bounded by hierarchical relations $r \in \mathcal{R}_H$. In essence, Eq. (7) encourages LinE embeddings for entity set $S_i \subseteq \mathcal{V}$ associated with entity set $S_j \subseteq \mathcal{V}$ via hierarchical relation $r$ to have smaller $L_d^S$ than $L_j^S$ for every dimension $d \in [1, h]$ in the LinE space, where $L_d^S \in \mathbb{R}$. A triple $r(S_i, S_j)$ has zero order violation if $L_d^S \leq L_d^S$ in the LinE space.

To preserve the non-hierarchical properties in LinE space, the LinE embeddings associated with non-hierarchical properties are alternatively regulated by the Mean-Square Error (MSE).

$$D_{\mathcal{R}_N} = \sum_{r(S_i, S_j) \in \mathcal{K}_{r}} (L_i^S - L_j^S(S_i, r))^2, \forall r \in \mathcal{R}_N$$

where $\mathcal{K}_{r} = r(S_i, S_j) \subseteq \mathcal{K}$ is the set of triples bounded by non-hierarchical relations $r \in \mathcal{R}_N$. Eq. (8) essentially preserves the L2-distance between LinE embeddings and the projected LinE embeddings of the entity set $S_j$, i.e., forcing $L_j^S$ to be as close to $L_i^S(S_i, r)$ obtained by the neural function $\text{MLP}_r(L_S^S)$ as possible.

**Intersection Operator.** The intersection of multiple quantities is essentially the minimum of all. Following this intuition, we propose to use min function to simulate the intersection operation. As shown in Figure 6(a), given $n$ h-dimensional LinE embeddings $\{L_{S_1}^S, L_{S_2}^S, ..., L_{S_n}^S\}$, we calculate the intersection $L_{\text{Inter}}$, by simply applying the minimum function, to amplify the agreements amongst $n$ LinE embeddings across $d \in [1, h]$. The intersection operator is formally defined as follows.

$$L_{\text{Inter}} = \min[p_{S_1}^j, p_{S_2}^j, ..., p_{S_n}^j], ..., \min[p_{k}^j, p_{k}^j, ..., p_{k}^j]h$$

where $p_{S_i}^j$ represents the $j$-th position in the LinE embedding $L_{S_i}$, for $i$-th entity set and $k$ is the number of sampled positions.

**Union Operator.** Prior work dealt with union operations by drastically restructuring the computation graphs into DNF [24]. On the contrary, we directly formulate the union operator with a maximum function for given LinE embeddings as shown in Figure 6(b). Formally, the union is defined as follows.

$$L_{\text{Union}} = \max[p_{1}^S, p_{2}^S, ..., p_{k}^S], ..., \max[p_{k}^S, p_{k}^S, ..., p_{k}^S]^h$$

where $p_{S_i}^j$ represents the $j$-th value in the LinE embedding $L_{S_i}$ for the entity set $S_i$, and $k$ is the number of sampled positions.

**Negation Operator.** Given the Beta embedding $(a, \beta)$ for a set $S$, we follow a prior work [24] to compute $(\frac{a}{a + \beta})$ as the negation of $(a, \beta)$ as illustrated in Figure 6(c). Formally, we define the negation operation for LinE embeddings for the set $S$ as follows.

$$L_{\text{Neg}} = [\frac{1}{p_{1}^S}, \frac{1}{p_{2}^S}, ..., \frac{1}{p_{k}^S}]h$$

where $p_{S_i}^j$ represents the $j$-th position in the LinE embedding $L^S$ and $k$ is the number of sampled positions.

**Logical Regulations.** Intuitively, the closer the resulting LinE embeddings after intersection operators to the initial transformed LinE embeddings for ground-truth answer entities, the better quality of refined LinE embeddings. To preserve the logical laws in the LinE space, the violations against the intersection operators $(\land)$ is formulated as an MSE estimator as follows.

$$D_{\text{FOL}} = \sum_{L_{\text{inter}} := \cap_{r(S_i, S_j)} L_{S_i}^S} (\hat{L}_{\text{Inter}} - L_{\text{Inter}})^2$$

where $\hat{L}_{\text{Inter}}$ is the LinE embedding after performing intersection for $n$ LinE embeddings $L_{S_i}^S$ with $i \in [1, n]$, and $L_{\text{Inter}}$ is the LinE embedding of the true answers.

### 3.4 Logical Query Reasoning

**Joint Learning Objective.** We consider both reasoning loss and query inference loss to jointly optimize LinE embeddings and parameterized neural functions as follows.

$$L = L_{\mathcal{A}} + \lambda L_{\mathcal{Q}}$$

where $L_{\mathcal{A}}$ is the reasoning loss, $L_{\mathcal{Q}}$ is the query inference loss, and $\lambda$ is a hyper-parameter controlling their respective importance. We describe each of them in the following.

**Reasoning Loss.** The reasoning loss estimates the distance between a logical query $q$ and its true answer in the LinE space. Let $S_q = \{v_q | v_q \in \mathcal{V}\}$ be the true answers to the query $q$. For each true query and answer pair $(q, v_q)$, we randomly select $K$ false answers $v'_q$, $S_{q'} = \{v'_q | v'_q \in \mathcal{V}\}$. To optimize, we minimize the skip-gram loss $L_{\mathcal{A}}$ on training pairs $T_q$ and $T_{\mathcal{A}}$ as follows.

$$L_{\mathcal{A}} = -\log(y - D_{\mathcal{QA}}(L_{v_q}^L, L^q)) = -\sum_{v'_q \in S_{q'}} 1 \log(D_{\mathcal{QA}}(L_{v'_q}^L, L^q) - y)$$

where $L^q$ is the query embedding, $L_{v_q}^q$ is the true answer embedding in the LinE space, $D_{\mathcal{QA}}$ is the MSE estimator for a QA pair, and $y$ is a margin. Eq. (14) encourages the query $L^q$ to be positioned closer to the true answers within $y$ L2-distance while far away from the false answers at least $y$ L2-distance in the LinE space.
Table 1: The Statistics of logical queries datasets.

| KG   | |V| | |R| | |K| |1p| |2p| |3p| |2i| |3i| |ip| |pi| |2u| |up| |2in| |3in| |inp| |pin| |pni|
|------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| FB15k-237 | 14,505 | 273 | 149,689 | | | | | 192,602 | 159,689 | 159,689 | 159,689 | 159,689 | 10,000 | 10,000 | 10,000 | 10,000 | 24,968 | 24,968 | 24,968 | 24,968 | 24,968 | 24,968 | 24,968 | 24,968 |
| NELL995 | 63,361 | 200 | 107,982 | | | | | 141,943 | 115,982 | 115,982 | 115,982 | 115,982 | 8,000 | 8,000 | 8,000 | 8,000 | 18,798 | 18,798 | 18,798 | 18,798 | 18,798 | 18,798 | 18,798 | 18,798 | 18,798 |
| WN18RR | 40,943 | 11 | 103,509 | | | | | 114,067 | 105,509 | 105,509 | 105,509 | 105,509 | 2,000 | 2,000 | 2,000 | 2,000 | 12,350 | 12,350 | 12,350 | 12,350 | 12,350 | 12,350 | 12,350 | 12,350 | 12,350 |

Query Inference Loss. The logical inference loss estimates the relational violations and the deviations from logical regularities, in particular, intersections executed in a computation graph for query \( q \). The query inference loss is formally defined as follows.

\[
L_Q = \sum_{r \in R_T} D_T + \sum_{r \in R_F} D_F + \sum_{\cap} D_{FOL}
\]

where \( D_T \) and \( D_F \) are the relational violations measured against hierarchical and non-hierarchical relations, respectively; \( D_{FOL} \) measures the logical violations against intersection operations. Note that due to limitations in training QA pairs, we only regulate LinE embeddings with respect to queries using intersection operators throughout the corresponding computation graphs.

Example. Given a query \( q \), LinE optimizes LinE embeddings and the set of parameterized relational/logical functions by Eq.13. The specific learnable items for the example query in Figure 3(b) are as follows: (i) the set of participant countries of 2008 Beijing Olympics (\( V_1 \)) via “participant” relational projection (Eq.6); (ii) the set of countries X won medals in 1952 Helsinki Olympics \( V_2 \) via “awardedTo” relational projection (Eq.6); (iii) the complement of set \( V_1 \) \( V_2 \) (Eq.9); and (iv) the set of countries \( V_1 \) by performing intersection between \( V_1 \) and \( V_2 \) (Eq.11); and finally (v) the set of places in set \( V_3 \) (Eq.9) via “contains” relational projection (Eq.6). Entities in \( V_1 \) are returned as the final answers to query \( q \), which is the capital city in Fiji (“Suva”).

4 EXPERIMENTS

To understand the reasoning performance of LinE, we study four research questions as follows. **RQ1**: What is the reasoning ability across complex logical queries? **RQ2**: What is the reasoning ability for logical queries with and without hierarchical relations? **RQ3**: What is the impact of logic space transformation?, and **RQ4**: What is the impact of relation projections and logical operators?

4.1 Datasets

We consider three KG benchmark datasets for FOL query reasoning. **FB15k-237** [4] and **NELL995** [5] are collections of relations between entities constructed from FB15k and the Never-Ending Language Learning (NELL) system, respectively. **WN18RR** [13] is a hierarchical collection of relations between words created from WordNet [20]. Data statistics on logical queries and KGs are summarized in Table 1 (see Appendix C and D for more details).

4.2 Experimental Settings

Baselines. We consider two dominant categories of baselines for FOL queries reasoning on KGs: (i) generic logical query reasoners (GQE [15], Q2B [23], BetaE [24]), which formulate embeddings in Euclidean space; and (ii) hierarchical logical query reasoners (HypE [12]), which formulate embeddings in Hyperbolic space.

Metrics. To evaluate the performance of examined methods, we measure the answer quality by the ranking of the true answers. We report two standard evaluation metrics: MRR and HITS@N, which is the fraction of correct answers in the top-N candidates.

4.3 Query Reasoning Complexity (RQ1)

Setup. To study the reasoning complexity of logical queries, we follow the formulation of the generalization reasoning task on 14 queries with at least one link prediction [23] to evaluate the performance in generalizing to plausible answers. We evaluate the task on three benchmarks, including FB15k-237, NELL995 and WN18RR. To evaluate the reasoning sensitivity to query diversity, we divide 14 query structures into three categories: (i) relation-heavy: queries that are purely tied to relation projections (1p/2p/3p), (ii) logic-heavy: queries that are heavily tied to logical operations (2i/3i/ip/pi/2u/up), and (iii) negation-related: queries that involve negation operator as shown in Figure 7 in Appendix C.

Result. Table 2 reports the comparative results for each query group on three benchmarks. First, we observe that BetaE and LinE consistently outperform other baselines (GQE, Q2B and HypE) across three benchmarks. For example, LinE achieves significant performance gain against HypE by nearly 6.6% and 98.76% for avg\(_p\) and avg\(_i\) in MRR on FB15k-237, respectively. Note that HypE cannot deal with negation operator and thus no comparison on avg\(_n\). Second, LinE achieves considerable performance gain in most cases on three benchmarks compared to BetaE. In particular on FB15k-237, LinE shows superior reasoning ability on relation-heavy (5.71% gain in avg\(_p\) MRR), logic-heavy (7.56% gain in avg\(_i\) MRR) and negation-related queries (19.46% gain in avg\(_n\) MRR). This suggests that LinE generally shows superior reasoning ability, particularly relation-heavy and negation-related queries despite the occasional miss on logic-heavy queries.

4.4 Hierarchical Logical Query (RQ2)

Setup. To study the reasoning ability for hierarchical logical queries, we adopt curvature estimates and Krackhardt scores to estimate the relational hierarchy for each relation (\( r \in R \)) on three benchmarks. The estimation suggests that WN18RR richly contains hierarchical relations compared to FB15k-237 and NELL995 as shown in Table 4. Thus, we additionally generate 14 types of FOL queries for WN18RR (Table 1) to investigate the reasoning sensitivity to hierarchical logical queries (Appendix D).

Result. In Table 3, we observe that both LinE and BetaE outperform other baselines (GQE, Q2B and HypE) across each query group in both MRR and HITS@3. For example, compare to BetaE on
Table 2: Performance comparison in MRR and HITS@3 (%) on benchmarks. avg$_{p}$, avg$_{i}$, and avg$_{n}$ denote the average MRR on relation-heavy, logic-heavy, and negation-related queries, respectively. Best (second best) of each column are in bold (underlined). The last row shows relative improvement (%) of LinE$_{\alpha,\beta}$ compared to the best baseline.

<table>
<thead>
<tr>
<th>Model</th>
<th>MRR</th>
<th>HITS@3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FB15k-237</td>
<td>NELL995</td>
</tr>
<tr>
<td></td>
<td>avg$_{p}$</td>
<td>avg$_{i}$</td>
</tr>
<tr>
<td>GQE</td>
<td>18.02</td>
<td>3.55</td>
</tr>
<tr>
<td>Q2B</td>
<td>22.46</td>
<td>3.47</td>
</tr>
<tr>
<td>BetaE</td>
<td>44.13</td>
<td>5.39</td>
</tr>
<tr>
<td>LinE$_{\alpha,\beta}$</td>
<td>45.12</td>
<td>6.93</td>
</tr>
<tr>
<td>Rel. Gain (%)</td>
<td>2.11</td>
<td>3.84</td>
</tr>
</tbody>
</table>

Table 3: Performance comparison on WN18RR. Best (second best) of each column are in bold (underlined). The last row shows relative improvement (%) of LinE$_{\alpha,\beta}$ compared to the best baseline.

<table>
<thead>
<tr>
<th>Model</th>
<th>MRR</th>
<th>HITS@3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FN18RR</td>
<td>NELL995</td>
</tr>
<tr>
<td></td>
<td>1p</td>
<td>2i</td>
</tr>
<tr>
<td>GQE</td>
<td>30.60</td>
<td>2.77</td>
</tr>
<tr>
<td>Q2B</td>
<td>42.10</td>
<td>2.62</td>
</tr>
<tr>
<td>BetaE</td>
<td>63.99</td>
<td>7.90</td>
</tr>
<tr>
<td>HypE</td>
<td>36.07</td>
<td>3.98</td>
</tr>
<tr>
<td>LinE$_{\alpha,\beta}$</td>
<td>47.08</td>
<td>9.24</td>
</tr>
<tr>
<td>Rel. Gain (%)</td>
<td>2.22</td>
<td>3.73</td>
</tr>
</tbody>
</table>

4.5 Logic Space Transformation (RQ3)

To evaluate the effectiveness of statistical signals, we study the neural transformation functions that project entity representations from the Beta distribution (BetaE) to the LinE space. We compare two types of statistical signals, (\(\alpha,\beta\)) and (\(\mu,\nu\)), given the same MLPs setting. In Table 3, we observe that LinE$_{\alpha,\beta}$ outperforms LinE$_{\mu,\nu}$ across 14 query types on WN18RR. This suggests that overall LinE$_{\alpha,\beta}$ gives a more reliable performance with (\(\alpha,\beta\)) as the primary statistical signals for logic space transformation with and without hierarchical relations.

4.6 Relation and Logical Operators (RQ4)

In this section, we study the effectiveness of relational projection and union operator in the LinE space.

Relation-specific Projection. We search for the optimal setting with 1,600 hidden dimensions and the two layers as our final MLPs setting. Table 2 reports the average MRR (avg$_{p}$) and HITS@3 (avg$_{p}$) for relation-heavy queries (1p/2p/3p) across benchmarks. Our relational projections in LinE$_{\alpha,\beta}$ significantly outperform BetaE across benchmarks. LinE achieves 73.58% relative gain in MRR, particularly for the most complex queries (3p) on WN18RR (Table 3).

Union. DNF [23, 24] has proven superiority in the literature. We study both i) DNF, and ii) U (Eq.10) to evaluate their effectiveness. Table 7 (Appendix E) shows that our LinE$_{\alpha,\beta}$ with DNF outperforms U on union queries (2u/up). As a result, the DNF is still the most effective formulation for union operators. Note that although U is slightly less effective than DNF for union queries, DNF takes extra computation to alter the logical query structures. Overall, our straightforward formulation is fairly competitive to DNF.

5 RELATED WORK

Logical Query Reasoning. Logical query reasoning has been recently received growing interest, in particular, the class of existential first-order logical queries (EPFO) which includes the logical and existential operator. To answer complex logical queries over
the KGs, some attempts have been made to formalize entities and queries as points [1, 15], or as regions [12, 22, 23], or as distributions [24] in high-dimensional representation space. GQE [15] embedded logical queries and entities in vector space, nonetheless GQE only supported conjunctive queries with ∃ and ∧. Query2Box [23] formalized entities and logical queries in box representation space, supporting ∃, ∧, ∨ and ¬ operators. HypE [12] formalizes entities as hyperboloid into Poincaré ball to better support FOL queries except the negation operator. BetaE [24] formalizes entities and queries as Beta distributions. The closure property of Beta distribution enables BetaE to tackle FOL queries with ∃, ∧, ∨, and ¬. Other attempts have been made to formalize entities and queries using different estimators [8, 11, 19, 26, 27, 35]. LogicE [19] combined query embeddings with the inductive bias of real-valued logic and also supports ∧, ∨ and ¬. Others attempted to learn logic operators as neural modules for reasoning [7, 18, 25].

Multi-relational Graph Embeddings. Our work is related to existing efforts on multi-relational knowledge graph embeddings, which solve knowledge graph reasoning by learning entity and relation embeddings in latent spaces. Some studies addressed limitations in conventional vector spaces by learning better representations for multi-relational knowledge graphs [3, 6, 32]. MuRP [3] embedded multi-relational graph data into Poincaré ball in hyperbolic space, and proposed to use Krackhardt score as hierarchy estimates for relations. ReFH, RoTH, and AttH are classic hyperbolic knowledge graph embedding that captures hierarchical information by adopting both curvature estimate and Krackhardt score to estimate the relational hierarchy. Order embeddings capture relational transitivity effectively [2, 10, 28, 29]. Some addressed the complexity of multi-hop knowledge graph completion [9, 16, 30, 31, 34]. For example, RLH [30] is proposed to solve multi-hop knowledge graph reasoning that addresses the multiple semantic issues where a relation in knowledge graphs may carry different meanings. MCMH [34] provided a new method for knowledge graph completion through learning multi-chain multi-hop rules.

6 CONCLUSIONS
We present a logical query reasoning framework (LinE) to preserve multi-relational complexities and logical regularities in the LinE space. LinE consists of a logic space transformation component to better support relational and logical operations by relaxing strong distributional assumptions. We design neural relation-specific projections to sensitively capture mixed relational properties in the KG guided by curvature estimate and Krackhardt score. We also generate FOL queries of 14 types from WN18RR to investigate the reasoning sensitivity for hierarchical logical queries. The results demonstrate the superior reasoning sensitivity of LinE for diverse FOL queries against dominant baselines.

REFERENCES
[16] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021. Explainable Subgraph Reasoning for Forecasting on Temporal Knowledge Graphs. In ICLR.
Appendices

A BETA DISTRIBUTION

Mean. The expected value $\mu$ of a Beta distributed random variable $X$ is formally defined with $(\alpha, \beta)$ as follows.

$$\mu(X) = E[X] = \frac{\alpha}{\alpha + \beta} \quad (16)$$

Variance. The expected variance $\sigma^2$ of a Beta distributed random variable $X$ is formally defined with $(\alpha, \beta)$ as follows.

$$\sigma^2(X) = E[(X - \mu(X))^2] = \frac{\alpha \beta}{(\alpha + \beta)^2 (\alpha + \beta + 1)} \quad (17)$$

B HIERARCHY ESTIMATE

Anti-symmetry Scores. The Krackhardt hierarchy score $K_{h\bar{a}r}$ is defined as follows.

$$K_{h\bar{a}r} = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}(1 - A_{ji}) \quad (18)$$

where $A$ is the adjacency matrix. $A_{ij} = 1$ if there is an edge from node $v_i$ to node $v_j$ and 0 otherwise. $K_{h\bar{a}r}$ is in the range $[0,1]$, where $K_{h\bar{a}r} = 0$ if $r$ is a fully symmetric relation and $K_{h\bar{a}r} = 1$ if $r$ is a fully anti-symmetric relation.

Transitive Scores. The transitive hierarchy $\xi_{h\bar{a}r}$ [14] that captures the global transitive behaviours for a given relation graph $G_r$ is formally defined as follows.

$$\xi_{h\bar{a}r} = \frac{1}{|\Delta_{G_r}|} \sum_{(v_i, v_j, v_k) \in \Delta_{G_r}} \left( \frac{1}{2D_{G_r}(v_i, v_m)^2} \left( D_{G_r}(v_j, v_m)^2 \right) \right)$$

$$+ \frac{(D_{G_r}(v_j, v_k)^2) / A - (D_{G_r}(v_i, v_j)^2 + D_{G_r}(v_i, v_k)^2) / 2}{A}$$

$$\quad (19)$$

where $\Delta_{G_r}$ refers to a sample set of triangles from $G_r$. $D_{G_r}$ is the shortest path distance for given node pair in $G_r$. For a given triangle $(v_i, v_j, v_k) \in \Delta_{G_r}$, we find the midpoint $v_m$ of the shortest path connecting $v_i$ to $v_k$ to estimate how it structurally fits into the global topology. $\xi_{h\bar{a}r}$ is zero for triangles in lines, positive for triangles in circles, and negative for triangles in trees. Intuitively negative $\xi_{h\bar{a}r}$ suggests that $r$ exhibits a strong transitivity.

C COMPLEX QUERY STRUCTURES

We follow [24] to examine FOL queries across 14 types of query structures. Figure 7 illustrates the 14 query structures considered in our experiments in computation graphs. To study the reasoning ability for diverse query types, we divide all query structures into three categories: (i) relation-heavy: queries that are purely tied to relation projections (1p/2p/3p), (ii) logic-heavy: queries that are heavily tied to logical operations (2i/3i/1i/2u/1u), and (iii) negation-related: queries that involve negation operator (2n/3n/1n/pin/pni/pnu). For example, “ip” indicates the following reasoning step: two relational projections for two anchor entities, followed by an intersection and another relational projection from the above intermediate results to arrive at the target answer (green node).

Table 4: Hierarchical Knowledge Graph: WN18RR

<table>
<thead>
<tr>
<th>Relation</th>
<th>$K_{h\bar{a}r}$, $\xi_{h\bar{a}r}$, Hierarchical</th>
</tr>
</thead>
<tbody>
<tr>
<td>memberMeronym</td>
<td>1.00 -2.90</td>
</tr>
<tr>
<td>hypernym</td>
<td>1.00 -2.46</td>
</tr>
<tr>
<td>hasPart</td>
<td>1.00 -0.82</td>
</tr>
<tr>
<td>instance hypernym</td>
<td>1.00 -0.78</td>
</tr>
<tr>
<td>memberOfDomainRegion</td>
<td>1.00 -0.78</td>
</tr>
<tr>
<td>memberOfDomainUsage</td>
<td>1.00 -0.74</td>
</tr>
<tr>
<td>synsetDomainTopicOf</td>
<td>0.99 -0.69</td>
</tr>
<tr>
<td>alsoSee</td>
<td>0.36 -2.09</td>
</tr>
<tr>
<td>derivationallyRelatedForm</td>
<td>0.07 -3.84</td>
</tr>
<tr>
<td>similarTo</td>
<td>0.07 -1.00</td>
</tr>
<tr>
<td>verbGroup</td>
<td>0.07 -0.50</td>
</tr>
</tbody>
</table>

Algorithm 1: Logical Query and Answer Pair Generation

Input: $G = (V, R)$, $K$, 14 query types $S$:

Output: queries $Q$, answers $T$;

Function GoundQueries($G$, $S$):

$\nu$ uniformly sample (w/o replacement) an entity $v \in V$

$\nu \leftarrow$ assign entity $v$ for the target root node $v_s \in V$

foreach node $v_t \in V_s \{ r_s \}$ in pre-order traversal ordering do

$\nu \leftarrow$ the entity assigned for the parent of node $v_t$

$K_a \leftarrow$ set of triples with the object as $v$ via any relation

$r'(v', v) \leftarrow$ uniformly sample a triple from $K_a$

$T_{v_a} \leftarrow$ assign entity $v'$ for the node $v_a$

$T_{v_a} \leftarrow$ assign relation $r$ for the edge from $v_a$ to its parent

return $T_{v_a}$

End Function

foreach query structure $s \in S$ do

/* step 1: generate queries */

$G_s = (V_s, E_s) \leftarrow$ induce a DAG according to the query structure $s$

$T_{Q_s} \leftarrow Q_s \cup GoundQueries(G_s, G, K)$

/* step 2: generate answers */

$T_{A_s} \leftarrow$ collect target entities as the final answers for $T_{Q_s}$

$T_{A_s} \leftarrow T_{A_s} \cup T_{A_s}$

return $T_{Q_s}$, $T_{A_s}$

D WN18RR-QA BENCHMARK

To study the reasoning sensitivity for hierarchical logical queries, we consider to generate QA pairs for WN18RR, which richly contains hierarchical relations. In Table 4, seven out of 11 relations on WN18RR are viewed as hierarchical relations due to their high anti-symmetry ($K_{h\bar{a}r}$) and negative transitive scores ($\xi_{h\bar{a}r}$).

Grounding Query Structures. To generate queries for training, we follow prior work [23, 24] to generate ten types of query structures, including 1p, 2p, 3p, 2i, 3i, 2n, 3n, inp, pin and pni. For evaluation, we consider all 14 query structures that are both seen and unseen during the training process. Given a KG and a query structure $s$ seen as a directed acyclic graph (DAG), we adopt preorder traversal to assign entities and relation types for each node and edge in the DAG to construct the query $q$ with type $s$. That is, starting from the target root node to the anchor leaf nodes for the given DAG, we uniformly sample an entity $v \in V$ in the KG.

1https://github.com/nelsonhuangzijian/WN18RR-QA
as the target node. For each child node linked to the target node in the DAG, we uniformly sample a triplet \(r(v',v)\) with object entity \(v\) via relation \(r\) in the KG. We then assign the relation \(r\) to the edge and entity \(v\) to the child node. Iteratively, we continue the next assignment of edges and nodes via pre-order traversal until each edge and node in the DAG are grounded with specific relation and entity in the KG. As a result, the leaf nodes in the DAG are viewed as the anchor nodes and the target root node is collected as the ground-truth answer to the query \(q\). We follow this procedure to generate the set of QA pairs, \(\mathcal{T}_Q = \{q_1, q_2, \ldots, q_n\}\) and \(\mathcal{T}_A = \{q_1[V_1], q_2[V_2], \ldots, q_n[V_n]\}\), for training, validation and testing on WN18RR. The overall procedure of logical query and answer pair generation is summarized in Algorithm 1. Examples of QA pairs for three query categories are illustrated in Table 5.

**Evaluation Protocol.** For fair performance comparison, we follow the split setup for the generated queries. Namely, the distribution of the number of queries for each query structure remains approximately identical to FB15k-237 and NELL995 in BetaE [24]. Specifically, we generate all “1p” queries (10,350), which is exactly the number of triplets \(|\mathcal{K}|\) in the original WN18RR (Table 1). The entire set of “1p” queries are used for training. For each query type of (2p/3p/2i/3i), we generate the same amount of queries as “1p” query (i.e., 103,509) as training set. For validation and testing queries, we generate one tenth the “1p” query (i.e., 10,350) as training set. For validation and testing queries, we generate approximately one fifth of the “1p” query in validation and testing query sets (~5K) for each query type (i.e., 1K). For example, given 5,202 and 5,356 of “1p” query in validation and testing sets from WN18RR, respectively, we generate 1,000 queries for other 13 query types for validation and testing. Table 6 reports the detail distributions of query types for FB15k-237, NELL995, and WN18RR.

**E ADDITIONAL EXPERIMENTS**

In Table 7, we observe that LinE_{α,β} with DNF outperforms U on union queries (2u/up). As a result, the DNF is still the most effective formulation for union operators. Note that although U is slightly less effective than DNF for union queries, DNF takes extra computation to alter the logical query structures. Overall, our straightforward formulation is fairly competitive to DNF.