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, **Lin**e **E**mbedding (**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 factoids. 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

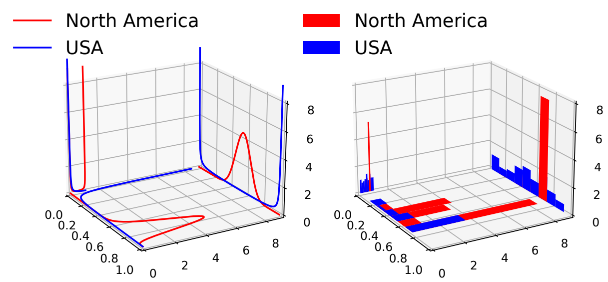
Knowledge representation learning; Logical query reasoning

ACM Reference Format:

Zijian Huang, Meng-Fen Chiang, and Wang-Chien Lee. 2022. LinE: Logical Query Reasoning over Hierarchical Knowledge Graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539338

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539338



(a) Beta Representation Space (b) LinE Representation Space

Figure 1: Comparison of Representation Space.

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 **Lin**e **E**mbedding (**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.

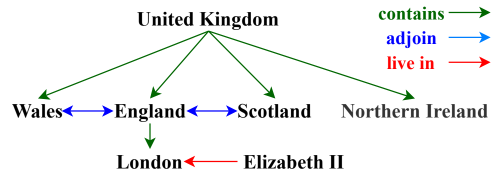


Figure 2: Examples of symmetric ('*adjoin*"), anti-symmetric ("*live in*") and transitive ("*contains*") relations on Freebase.

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

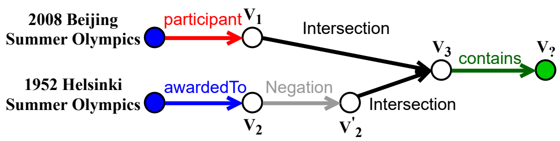
A knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R})$ is a multi-relational graph, where $v \in \mathcal{V}$ represents an entity, and each relation type $r \in \mathcal{R}$ is a binary function $r : \mathcal{V} \times \mathcal{V} \to \{True, False\}$ that indicates the existence of a type-r directed 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.

(a) Atomic Relation Query

$q = V_?.\ \exists V_1, V_2 :\ (participant(2008\ Olympics, V_1)$
$\wedge\ \neg awardedTo(1952\ Olympics\ Medal, V_2))\ \wedge\ contains(V_3, V_?)$



(b) Compositional Relation Query

Figure 3: Computation graphs.

First-Order Logic Queries. First-order logic (FOL) queries are queries formulated by logical operators, including conjunction ($\wedge$), disjunction ($\vee$), existential quantification ($\exists$) and negation ($\neg$). An FOL query q can be expressed in disjunctive normal form (DNF), which is a disjunction of conjunctions as follows.

$$\begin{aligned} q[V_?] &= V_?.\ \exists\ V_1, \ldots, V_w : c_1 \vee c_2 \vee \ldots \vee c_n, \\ c_i &:= e_{i1} \wedge e_{i2} \wedge \ldots \wedge e_{im} \end{aligned} \quad (1)$$

where $V_?$ is the target variable, $\{V_i | 1 \le i \le w\}$ are existentially quantified bound variables, and $\{c_i | 1 \le i \le n\}$ are conjunction queries. Target variable $V_?$ 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 $\wedge_{j=1}^{m} e_{ij}$.

Atomic Relation Query. An atomic relation query e_{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:

$$e_{ij} = r(v_a, V)\ or\ \neg r(v_a, V)\ or\ r(V', V)\ or\ \neg r(V', V), V \neq V' \quad (2)$$

where $v_a \in \mathcal{V}_a$ is a non-variable anchor entity, $V \in \{V_?\} \cup \{V_i | 1 \le i \le w\}$ is in the complete set of variables, $V' \in \{V_i | 1 \le i \le w\}$ is in the set of intermediate variables, and $r \in \mathcal{R}$ 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 structurized 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 $V_?$ 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 $V_?$ ("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 ($\mathcal{R}_{\mathcal{T}}$) as *hierarchical logical queries*; whereas the FOL queries comprise of purely non-hierarchical relations ($\mathcal{R}_{\overline{\mathcal{T}}}$) 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 ($\mathcal{R}_{\mathcal{T}}$) from non-hierarchical relations ($\mathcal{R}_{\overline{\mathcal{T}}}$). Motivated by this, we explore two metrics, *Krackhardt hierarchy score* ($Khs_{\mathcal{G}_r}$) [17] and *curvature* ($\xi_{\mathcal{G}_r}$) [14], to estimate the degrees of anti-symmetry and transitivity for each relation $r \in \mathcal{R}$, respectively. A relation r is considered highly hierarchical if its induced relation graph $\mathcal{G}_r$ (i.e., the graph structured only with relation r) has higher anti-symmetry scores ($Khs_{\mathcal{G}_r}$) and higher transitivity scores ($\xi_{\mathcal{G}_r}$), 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 $\mathbf{B}(\cdot)$ shaped by two parameters α and β. The probability density function (PDF) controlled by (α,β) is defined as $p(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathbf{B}(\alpha,\beta)}$. An h-dimensional Beta embedding of an entity set S ($B^S \in \mathbb{R}^{2\times h}$) consists of h independent Beta distributions on the interval [0,1], denoted as $B^S = [(\alpha_1^S, \beta_1^S), \cdots, (_h^S, \beta_h^S)]$ 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 ($\wedge$, $\vee$, $\exists$ and $\neg$) and relational projections ($r \in \mathcal{R}$) 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 $\mathcal{G} = (\mathcal{V}, \mathcal{R})$ and an FOL query q with relational and logical compositions, our goal is to enable *hierarchical*

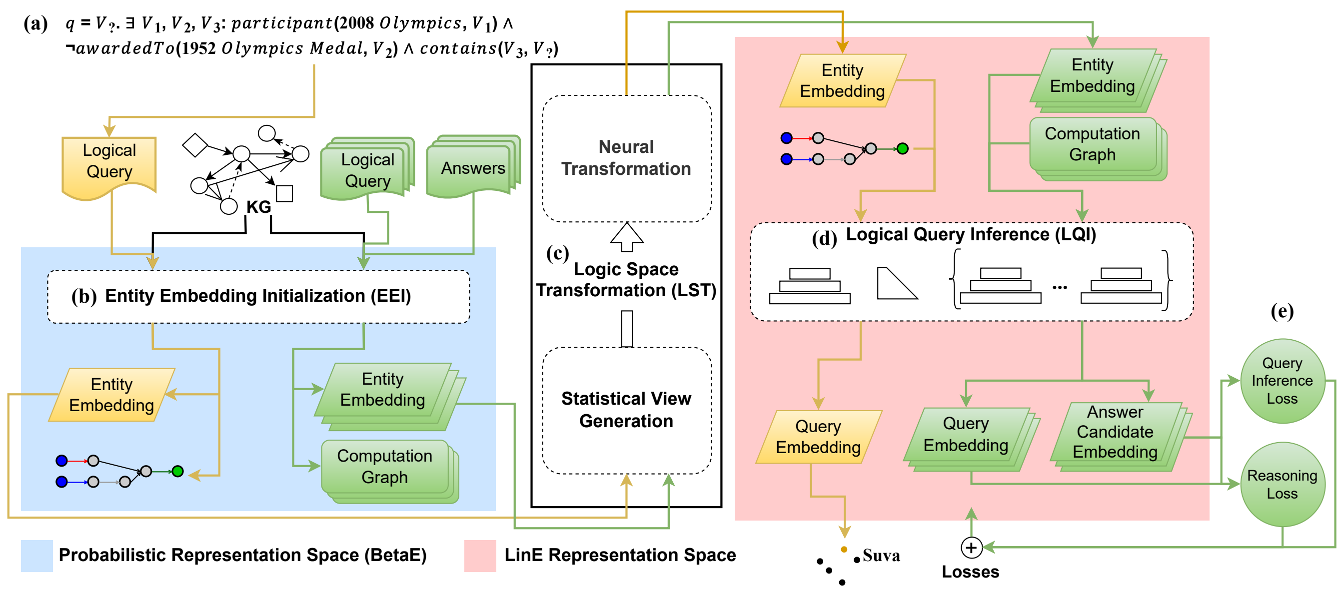


Figure 4: The model architecture of LinE. The green color indicates the training process, while the yellow color indicates the real-time testing process for unseen queries. (a) For training, LinE takes as input the KG and pairs of logical queries and answers. (b) Given the computation graph for each 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 (pink area). (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.

logical query reasoning of q over $\mathcal{G}$. Our approach is to design a novel neural logical reasoning framework that supports logical operators ($\wedge$, $\vee$, $\exists$ and $\neg$) and relation-specific projections ($\mathcal{R}$) for expressive specification of the FOL query (q); and return a set of entities ($V_? \subseteq \mathcal{V}$) that satisfy q over the facts captured by $\mathcal{G}$.

3 LINE EMBEDDINGS

We present our **Lin**e **E**mbeddings 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 ($\mathcal{G}$), a set of query answering (QA) pairs as a training set $\mathcal{T}_Q = \{q_1, q_2, \cdots, q_n\}$ and $\mathcal{T}_\mathcal{A} = \{q_1[V_?], q_2[V_?], \cdots, q_n[V_?]\}$, 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 q_i as input, its computation graph is generated to indicate the reasoning steps. We follow the reasoning steps to execute logical operators in the query and obtain KG entity embeddings in the BetaE space as the initial entity embeddings. To enhance the expressive power, we propose a logic space transformation (LST) to project the initial entity embeddings from the BetaE space to the LinE space. The logical query inference (LQI) is proposed to refine the transformed embeddings for KG entities and query q_i by imposing multi-relational and logical regularities in the LinE space. Logical operations ($\wedge$, $\vee$, $\exists$ and $\neg$) and relation-specific projections ($r \in \mathcal{R}$) are mathematically formulated. To preserve the multi-relational and logical regularities, we formulate (i) *reasoning loss* to estimate the distances between the query embedding and answer candidates; and (ii) *query inference loss* to estimate relational violations and deviations from logical regularities, respectively. Lastly, the final answers are obtained by performing nearest-neighbor search (NNS) for the final query embedding and returning the closest entities ($V_? \subseteq \mathcal{V}$) to query q_i in the LinE 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.

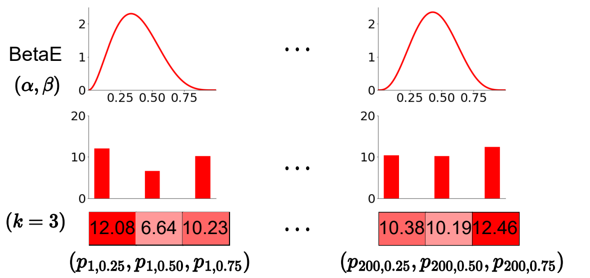
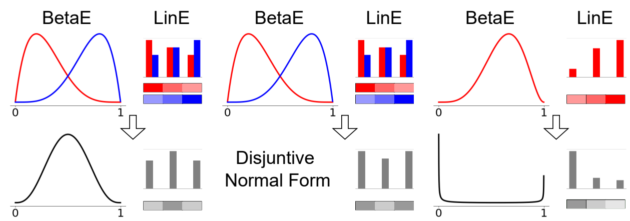


Figure 5: Logic Space Transformation: the top and bottom row depict the 200-dimensional Beta embedding and 200-dimensional LinE Embedding at positions $(0.25, 0.50, 0.75)$**.**



(a) Intersection (b) Union (c) Negation

Figure 6: Logical operations for BetaE and LinE embeddings: (a) intersection, (b) union, and (c) negation operations. Note that BetaE cannot directly handle union operations.

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 $\mathbf{B}(\cdot)$. Formally, an h-dimensional LinE embedding for the set S is defined as $L^S = [(p^S_{j,1}, p^S_{j,2}, \cdots, p^S_{j,k})]^h_{j=1} \in \mathbb{R}^{k\times h}$ across k positions in the range $[0, 1]$ per dimension $1 \le j \le h$. 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 $[(p_{j,0.25}, p_{j,0.50}, p_{j,0.75})]^{200}_{j=1}$. 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 ($\wedge$, $\vee$, $\exists$ 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 initial LinE embeddings. Specifically, we consider the following shape features.

- **α and β**: α and β 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 (α,β) parameters as input features for the neural transformation function as follows.
$$B^{S_i} = [(\alpha^{S_i}_j, \beta^{S_i}_j)]^h_{j=1} \in \mathbb{R}^{2\times h} \tag{3}$$
- **Mean and Variance**: Mean ($\mu(\mathbf{X})$) measures the central tendency of a Beta distribution. Variance ($var(\mathbf{X})$) 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.
$$P^{S_i} = [(\mu(\mathbf{X}_j)^{S_i}, var(\mathbf{X}_j)^{S_i})]^h_{j=1} \in \mathbb{R}^{2\times h}. \tag{4}$$

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 $\mathbf{S}$=$\{S_1, \cdots, S_n\}$, the shape features extracted from n Beta embeddings $\mathbf{F}$=$\{F_1, \cdots, F_n\}$ is fed to an entity-wise MLP, which generates n LinE embeddings $\mathbf{L}$=$\{L_1, \cdots, L_n\}$. A neural network that converts a set of Beta embeddings to a set of LinE embeddings is trained as follows.

$$\mathbf{L} = \mathbf{MLP}(\mathbf{F}) \tag{5}$$
$$F^{S_i} = [B^{S_i}] \mid [P^{S_i}], \forall S_i \in \mathbf{S}$$
$$L^{S_i} = [(p^{S_i}_{j,1}, \cdots, p^{S_i}_{j,k})]^h_{j=1} \in \mathbb{R}^{k\times h}, \forall S_i \in \mathbf{S}$$

where $F_i \in F$ is the final shape feature concatenated from any of B^{S_i} and P^{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 ($\wedge$, $\vee$, $\exists$ 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_{Inter}, negation L_{Neg}, and union L_{Union}.

Relation-Specific Projection. Relations in multi-relational KGs have diverse properties: symmetric (e.g., *adjoins*), anti-symmetric (e.g., *is_a_member_of*), and transitive relations (e.g., *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 $r(v_a, V)$, a relation-specific projection learns a projection function X_r that takes v_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 $\mathbf{MLP}_r$ with ReLU as activation function as follows.

$$\hat{L}^{S_j}(S_i, r) = \mathbf{MLP}_r(L^{S_i}) \quad (6)$$

where L^{S_i} and $\hat{L}^{S_j}(S_i, r)$ 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 ($\mathcal{R}_{\mathcal{T}}$) and non-hierarchical relations ($\mathcal{R}_{\overline{\mathcal{T}}}$). Accordingly, we propose the following relational regulations $\mathcal{D}_{\mathcal{T}}$ and $\mathcal{D}_{\overline{\mathcal{T}}}$ to preserve hierarchical relations ($\mathcal{R}_{\mathcal{T}}$) and non-hierarchical relations ($\mathcal{R}_{\overline{\mathcal{T}}}$), respectively, where $\mathcal{R}_{\mathcal{T}} \bigcap \mathcal{R}_{\overline{\mathcal{T}}} = \varnothing$ and $\mathcal{R}_{\mathcal{T}} \bigcup \mathcal{R}_{\overline{\mathcal{T}}} = \mathcal{R}$.

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.

$$\mathcal{D}_{\mathcal{T}} = \sum_{r(S_i,S_j) \in \mathcal{K}_{\mathcal{T}}} \max\{0, L^{S_i} - \hat{L}^{S_j}(S_i, r)\}, \forall r \in \mathcal{R}_{\mathcal{T}} \quad (7)$$

where $\mathcal{K}_{\mathcal{T}} = r(S_i, S_j) \subseteq \mathcal{K}$ is the set of triple bounded by hierarchical relations $r \in \mathcal{R}_{\mathcal{T}}$. 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 $L_d^{S_i}$ than $L_d^{S_j}$ 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 $L_d^{S_i} \leq L_d^{S_j}$ 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).

$$\mathcal{D}_{\overline{\mathcal{T}}} = \sum_{r(S_i,S_j) \in \mathcal{K}_{\overline{\mathcal{T}}}} (L^{S_j} - \hat{L}^{S_j}(S_i, r))^2, \forall r \in \mathcal{R}_{\overline{\mathcal{T}}} \quad (8)$$

where $\mathcal{K}_{\overline{\mathcal{T}}} = r(S_i, S_j) \subseteq \mathcal{K}$ is the set of triples bounded by non-hierarchical relations $r \in \mathcal{R}_{\overline{\mathcal{T}}}$. 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^{S_j} to be as close to $\hat{L}^{S_j}(S_i, r)$ obtained by the neural function $\mathbf{MLP}_r(L_{S_i})$ 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 $\{L^{S_1}, L^{S_2}, ..., L^{S_n}\}$, we calculate the intersection L_{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.

$$L_{Inter} = [\min\{p_1^{S_1}, p_1^{S_2}, ..., p_1^{S_n}\}, ..., \min\{p_k^{S_1}, p_k^{S_2}, ..., p_k^{S_n}\}]^h \quad (9)$$

where $p_j^{S_i}$ 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.

$$L_{Union} = [\max\{p_1^{S_1}, p_1^{S_2}, ..., p_1^{S_n}\}, ..., \max\{p_k^{S_1}, p_k^{S_2}, ..., p_k^{S_n}\}]^h \quad (10)$$

where $p_j^{S_i}$ 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 (α,β) for a set S, we follow a prior work [24] to compute $(\frac{1}{\alpha}, \frac{1}{\beta})$ as the negation of (α, β) as illustrated in Figure 6(c). Formally, we define the negation operation for LinE embeddings for the set S as follows.

$$L_{Neg} = [\frac{1}{p_1^S}, \frac{1}{p_2^S}, ..., \frac{1}{p_k^S}]^h \quad (11)$$

where p_j^S 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 ($\wedge$) is formulated as an MSE estimator as follows.

$$\mathcal{D}_{FOL} = \sum_{\hat{L}_{Inter} := \cap_{i=1}^{n} L^{S_i}} (\hat{L}_{Inter} - L_{Inter})^2 \quad (12)$$

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

$$\mathcal{L} = \mathcal{L}_{\mathcal{A}} + \lambda \mathcal{L}_{Q} \quad (13)$$

where $\mathcal{L}_{\mathcal{A}}$ is the reasoning loss, $\mathcal{L}_Q$ is the query inference loss, and λ 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 $S_y = \{v_y | v_y \in \mathcal{V}\}$ be the true answers to the query q. For each true query and answer pair (q, v_y), we randomly select K false answers $v_{y'}$, $S_{y'} = \{v_{y'} | v_{y'} \in \mathcal{V}\}$. To optimize, we minimize the skip-gram loss $\mathcal{L}_{\mathcal{A}}$ on training pairs $\mathcal{T}_Q$ and $\mathcal{T}_{\mathcal{A}}$ as follows.

$$\mathcal{L}_{\mathcal{A}} = -\log(\gamma - \mathcal{D}_{QA}(L^{v_y}, L^q)) - \sum_{v_{y'} \in S_{y'}} \frac{1}{K} \log(\mathcal{D}_{QA}(L^{v_{y'}}, L^q) - \gamma) \quad (14)$$

where L^q is the query embedding, L^{v_y} is the true answer embedding in the LinE space, $\mathcal{D}_{QA}$ is the MSE estimator for a QA pair, and γ is a margin. Eq. (14) encourages the query L^q to be positioned closer to the true answers within γ L2-distance while far away from the false answers at least γ L2-distance in the LinE space.

Table 1: The Statistics of logical queries datasets.

KG	$\|\mathcal{V}\|$	$\|\mathcal{R}\|$	$\|\mathcal{K}\|$	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
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
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

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.

$$\mathcal{L}_Q = \sum_{r \in \mathcal{R}_{\overline{\mathcal{T}}}} \mathcal{D}_{\overline{\mathcal{T}}} + \sum_{r \in \mathcal{R}_{\mathcal{T}}} \mathcal{D}_{\mathcal{T}} + \sum_{\cap} \mathcal{D}_{FOL} \tag{15}$$

where $\mathcal{D}_{\overline{\mathcal{T}}}$ and $\mathcal{D}_{\mathcal{T}}$ are the relational violations measured against hierarchical and non-hierarchical relations, respectively. $\mathcal{D}_{FOL}$ 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 won medals in 1952 Helsinki Olympics V via "*awardedTo*" relational projection (Eq.6); (iii) the complement of set V (V_2), obtained by performing negation (Eq.11); (iv) the set of countries (V_3) by performing intersection between V_1 and V_2 (Eq.9); and lastly (v) the set of places in set V_3 ($V_?$) via "*contains*" relational projection (Eq.6). Entities in $V_?$ 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 6.69% and 98.76% for avg_p and avg_l 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_l 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 ($r \in \mathcal{R}$) 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).

Results. 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_l, 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 $\text{LinE}_{\alpha,\beta}$ compared to the best baseline.

Model	MRR									HITS@3								
	FB15k-237			NELL995			WN18RR			FB15k-237			NELL995			WN18RR		
	avg_p	avg_l	avg_n	avg_p	avg_l	avg_n	avg_p	avg_l	avg_n	avg_p	avg_l	avg_n	avg_p	avg_l	avg_n	avg_p	avg_l	avg_n
GQE	9.85	10.43	-	12.26	12.95	-	8.41	11.62	-	12.58	12.24	-	18.06	15.79	-	12.70	16.48	-
Q2B	13.04	13.89	-	14.59	16.17	-	9.80	16.25	-	15.15	16.07	-	21.79	**19.04**	-	15.97	27.08	-
BetaE	17.50	16.41	4.78	19.79	16.60	5.11	19.28	**32.83**	17.87	18.50	17.47	3.97	21.61	17.84	4.51	20.11	**35.64**	19.08
HypE	17.34	8.88	-	18.25	14.50	-	9.62	19.83	-	18.46	8.82	-	20.53	15.65	-	14.46	24.43	-
$\text{LinE}_{\alpha,\beta}$	**18.50**	**17.65**	**5.71**	**22.27**	**17.52**	**5.46**	**21.39**	28.21	**18.72**	**19.69**	**19.04**	**4.40**	**24.32**	18.87	**4.72**	**22.17**	30.12	**20.37**
Rel. Gain (%)	5.71	7.56	19.46	12.53	5.54	6.85	10.94	-14.07	4.76	6.43	8.99	10.83	11.61	-0.89	4.66	10.24	-15.49	6.76

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

	Model	WN18RR																
		1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni	avg_p	avg_l	avg_n
MRR	GQE	18.02	4.54	2.68	19.32	23.97	10.60	9.91	2.37	3.55	-	-	-	-	-	8.41	11.62	-
	Q2B	22.46	4.63	2.31	25.59	41.23	11.04	13.27	2.89	3.47	-	-	-	-	-	9.80	16.25	-
	BetaE	44.13	9.85	3.86	**57.19**	**76.26**	**17.97**	**32.59**	7.57	5.39	**12.77**	59.98	5.07	4.04	7.48	19.28	**32.83**	17.87
	HypE	20.93	5.40	2.54	30.10	58.06	9.30	13.44	3.52	4.54	-	-	-	-	-	9.62	19.83	-
	$\text{LinE}_{\mu,var}$	44.02	9.70	4.63	44.84	65.12	13.20	21.95	7.26	5.49	11.98	58.78	5.87	4.21	7.55	19.45	26.31	17.68
	$\text{LinE}_{\alpha,\beta}$	**45.12**	**12.35**	**6.70**	47.11	67.13	14.73	24.87	**8.49**	**6.93**	12.50	**60.81**	**7.34**	**5.20**	**7.74**	**21.39**	28.21	**18.72**
	Rel. Gain (%)	2.24	25.38	73.58	-17.63	-11.97	-18.03	-23.69	12.15	28.57	-2.11	1.38	44.77	28.71	3.48	10.94	-14.07	4.76
HITS@3	GQE	30.86	4.68	2.57	31.67	37.90	11.02	13.30	2.25	2.77	-	-	-	-	-	17.70	16.48	-
	Q2B	42.11	3.73	2.07	48.27	76.55	11.39	20.53	3.10	2.62	-	-	-	-	-	15.97	27.08	-
	BetaE	46.01	10.42	3.92	**63.99**	**81.20**	**19.89**	**35.11**	7.90	5.73	13.12	65.75	5.01	4.21	7.33	20.11	**35.64**	19.09
	HypE	36.07	5.00	2.32	38.59	73.75	10.73	16.36	3.17	3.98	-	-	-	-	-	14.46	24.43	-
	$\text{LinE}_{\mu,var}$	46.03	11.00	4.91	47.28	70.52	14.22	24.49	7.73	5.52	13.03	65.63	5.84	4.39	8.27	20.65	28.29	19.43
	$\text{LinE}_{\alpha,\beta}$	**47.08**	**12.87**	**6.56**	49.46	72.12	15.85	26.66	**9.40**	**7.24**	**13.22**	**66.56**	**8.25**	**5.79**	**8.04**	**22.17**	30.12	**20.37**
	Rel. Gain (%)	2.33	23.51	67.35	-22.71	-11.18	-20.31	-24.07	18.99	26.35	0.76	1.23	64.67	37.53	9.69	10.24	-15.49	6.71

relation-heavy queries, LinE achieves nearly 10.94% improvement in MRR (avg_p). Particularly, LinE improves the answer accuracy for 2p and 3p queries with nearly 25.38% and 73.58% in MRR, respectively. For negation-related queries, LinE slightly outperforms BetaE with 4.76% improvement in MRR. For logic-heavy queries, BetaE occasionally outperforms LinE, which constitutes 14.07% overall loss in MRR. We leave this improvement for future work.

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, (α,β) and (μ, var), given the same MLPs setting. In Table 3, we observe that $\text{LinE}_{\alpha,\beta}$ outperforms $\text{LinE}_{\mu,var}$ across 14 query types on WN18RR. This suggests that overall $\text{LinE}_{\alpha,\beta}$ gives a more reliable performance with (α,β) 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 $\text{LinE}_{\alpha,\beta}$ 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 $\text{LinE}_{\alpha,\beta}$ 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 $\exists$ and $\wedge$. Query2Box [23] formalized entities and logical queries in box representation space, supporting $\exists$, $\wedge$ and $\vee$ operators. HypE [12] formalizes entities as hyperboloid into Poincaré ball to better supports 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 $\exists$, $\wedge$, $\vee$, and $\neg$. 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 $\wedge$, $\vee$ and $\neg$. 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

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

[2] Ben Athiwaratkun and Andrew Gordon Wilson. 2018. Hierarchical Density Order Embeddings. In *ICLR*.

[3] Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. Multi-relational Poincaré Graph Embeddings. In *NeurIPS*.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*.

[5] Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *WSDM*.

[6] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. In *ACL*.

[7] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In *WWW*.

[8] Xuelu Chen, Ziniu Hu, and Yizhou Sun. 2021. Fuzzy Logic based Logical Query Answering on Knowledge Graph. *arXiv:2108.02390* (2021).

[9] Kewei Cheng, Ziqing Yang, Ming Zhang, and Yizhou Sun. 2021. UniKER: A Unified Framework for Combining Embedding and Definite Horn Rule Reasoning for Knowledge Graph Inference. In *EMNLP*.

[10] Meng-Fen Chiang, Ee-Peng Lim, Wang-Chien Lee, Xavier Jayaraj Siddarth Ashok, and Philips Kokoh Prasetyo. 2019. One-class order embedding for dependency relation prediction. In *SIGIR*.

[11] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. 2021. Probabilistic Entity Representation Model for Reasoning over Knowledge Graphs. In *NeurIPS*.

[12] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K Reddy. 2021. Self-Supervised Hyperboloid Representations from Logical Queries over Knowledge Graphs. In *WWW*.

[13] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

[14] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. 2018. Learning mixed-curvature representations in product spaces. In *ICLR*.

[15] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding Logical Queries on Knowledge Graphs. In *NeurIPS*.

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

[17] David Krackhardt. 2014. Graph theoretical dimensions of informal organizations. In *Computational organization theory*. 107–130.

[18] Lihui Liu, Boxin Du, Heng Ji, Cheng Xiang Zhai, and Hanghang Tong. 2021. Neural-Answering Logical Queries on Knowledge Graphs. In *SIGKDD*.

[19] Francois Luus, Prithviraj Sen, Pavan Kapanipathi, Ryan Riegel, Ndivhuwo Makondo, Thabang Lebese, and Alexander Gray. 2021. Logic Embeddings for Complex Query Answering. *arXiv:2103.00418*.

[20] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[21] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*.

[22] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Denny Zhou, Jure Leskovec, and Dale Schuurmans. 2021. SMORE: Knowledge Graph Completion and Multi-hop Reasoning in Massive Knowledge Graphs. *arXiv:2110.14890* (2021).

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

[24] Hongyu Ren and Jure Leskovec. 2020. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. In *NeurIPS*.

[25] Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. 2020. Neural Logic Reasoning. In *CIKM*.

[26] Haitian Sun, Andrew O Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W Cohen. 2020. Faithful embeddings for knowledge base queries. In *NeurIPS*.

[27] Komal Teru, Etienne Denis, and Will Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *ICML*.

[28] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. In *ICLR*.

[29] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. 2018. Probabilistic embedding of knowledge graphs with box lattice measures. In *ACL*.

[30] Guojia Wan, Shirui Pan, Chen Gong, Chuan Zhou, and Gholamreza Haffari. 2020. Reasoning Like Human: Hierarchical Reinforcement Learning for Knowledge Graph Reasoning. In *IJCAI*.

[31] Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. 2021. Direct Multi-hop Attention based Graph Neural Network. In *IJCAI*.

[32] Kai Wang, Yu Liu, Dan Lin, and Quan Z. Sheng. 2021. Hyperbolic Geometry is Not Necessary: Lightweight Euclidean-Based Models for Low-Dimensional Knowledge Graph Embeddings. In *EMNLP*.

[33] Gerhard Weikum. 2007. A core of semantic knowledge unifying wordnet and wikipedia. In *WWW*.

[34] Lu Zhang, Mo Yu, Tian Gao, and Yue Yu. 2020. MCMH: Learning Multi-Chain Multi-Hop Rules for Knowledge Graph Reasoning. In *EMNLP*.

[35] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. 2021. ConE: Cone Embeddings for Multi-Hop Reasoning over Knowledge Graphs. In *NeurIPS*.

Appendices

A BETA DISTRIBUTION

Mean. The expected value μ of a Beta distributed random variable $\mathbf{X}$ is formally defined with (α,β) as follows.

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

Variance. The expected variance var of a Beta distributed random variable X is formally defined with (α,β) as follows.

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

B HIERARCHY ESTIMATE

Anti-symmetry Scores. The Krackhardt hierarchy score $Khs_{\mathcal{G}_r}$ [17] that captures the local anti-symmetry property of a relation graph $\mathcal{G}_r$ is defined as follows.

$$Khs_{\mathcal{G}_r} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}(1-A_{j,i})}{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}} \tag{18}$$

where A is the adjacency matrix. $A_{i,j}=1$ if there is an edge from node v_i to node v_j and 0 otherwise. $Khs_{\mathcal{G}_r}$ is in the range [0,1], where $Khs_{\mathcal{G}_r}$=0 if r is a fully symmetric relation and $Khs_{\mathcal{G}_r}$=1 if r is a fully anti-symmetric relation.

Transitive Scores. The curvature estimate $\xi_{\mathcal{G}_r}$ [14] that captures the global transitive behaviours for a given relation graph $\mathcal{G}_r$ is formally defined as follows.

$$\xi_{\mathcal{G}_r} = \frac{1}{|\Delta_{\mathcal{G}_r}|}\sum_{(v_i,v_j,v_k)\in\Delta_{\mathcal{G}_r}} \frac{1}{2\mathcal{D}_{\mathcal{G}_r}(v_i,v_m)}\Big(\mathcal{D}_{\mathcal{G}_r}(v_i,v_m)^2 + (\mathcal{D}_{\mathcal{G}_r}(v_j,v_k)^2)/4 - (\mathcal{D}_{\mathcal{G}_r}(v_i,v_j)^2 + \mathcal{D}_{\mathcal{G}_r}(v_i,v_k)^2)/2\Big) \tag{19}$$

where $\Delta_{\mathcal{G}_r}$ refers to a sample set of triangles from $\mathcal{G}_r$. $\mathcal{D}_{\mathcal{G}_r}$ is the shortest path distance for given node pair in $\mathcal{G}_r$. Given a triangle $(v_i,v_j,v_k)\in\Delta_{\mathcal{G}_r}$, we find the midpoint v_m of the shortest path connecting v_j to v_k to estimate how it structurally fits into the global topology. $\xi_{\mathcal{G}_r}$ is zero for triangles in lines, positive for triangles in circles, and negative for triangles in trees. Intuitively negative $\xi_{\mathcal{G}_r}$ 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/ip/pi/2u/up), and (iii) *negation-related*: queries that involve negation operator (2in/3in/inp/pin/pni). 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

Relation	$Khs_{\mathcal{G}_r}$	$\xi_{\mathcal{G}_r}$	Hierarchical
memberMeronym	1.00	-2.90	✓
hypernym	1.00	-2.46	✓
hasPart	1.00	-0.82	✓
instance hypernym	1.00	-0.78	✓
memberOfDomainRegion	1.00	-0.78	✓
memberOfDomainUsage	1.00	-0.74	✓
synsetDomainTopicOf	0.99	-0.69	✓
alsoSee	0.36	-2.09	✗
derivationallyRelatedForm	0.07	-3.84	✗
similarTo	0.07	-1.00	✗
verbGroup	0.07	-0.50	✗

Algorithm 1: Logical Query and Answer Pair Generation

Input: $\mathcal{G}=(\mathcal{V},\mathcal{R})$, $\mathcal{K}$, 14 query types S;

Output: queries $\mathcal{T}_Q$, answers $\mathcal{T}_{\mathcal{A}}$;

Function `GoundQueries`$(G_s, \mathcal{G}, \mathcal{K})$**:**
- $v \leftarrow$ uniformly sample (w/o replacement) an entity $v\in\mathcal{V}$
- $\mathcal{T}_{Q,s} \leftarrow$ assign entity v for the target root node $\boldsymbol{r_s}\in V_s$
- **foreach** *node* $\boldsymbol{v_s}\in V_s\setminus\{\boldsymbol{r_s}\}$ *in pre-order traversal ordering* **do**
  - $v \leftarrow$ the entity assigned for the parent of node $\boldsymbol{v_s}$
  - $\mathcal{K}_v \leftarrow$ set of triples with the object as v via any relation
  - $r(v', v) \leftarrow$ uniformly sample a triple from $\mathcal{K}_v$
  - $\mathcal{T}_{Q,s} \leftarrow$ assign entity v' for the node $\boldsymbol{v_s}$
  - $\mathcal{T}_{Q,s} \leftarrow$ assign relation r for the edge from $\boldsymbol{v_s}$ to its parent
- **return** $\mathcal{T}_{Q,s}$

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*
- $\mathcal{T}_Q \leftarrow \mathcal{T}_Q \cup$ *GoundQueries*$(G_s, \mathcal{G}, \mathcal{K})$
- `/* step 2: generate answers */`
- $\mathcal{T}_{\mathcal{A},s} \leftarrow$ *collect target entities as the final answers for* $\mathcal{T}_{Q,s}$
- $\mathcal{T}_{\mathcal{A}} \leftarrow \mathcal{T}_{\mathcal{A}} \cup \mathcal{T}_{\mathcal{A},s}$

return $\mathcal{T}_Q$, $\mathcal{T}_{\mathcal{A}}$

D WN18RR-QA BENCHMARK

To study the reasoning sensitivity for hierarchical logical queries, we consider to generate QA pairs[1] 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 ($Khs_{\mathcal{G}_r}$) and negative transitive scores ($\xi_{\mathcal{G}_r}$).

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, 2in, 3in, 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 pre-order 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 $v\in\mathcal{V}$ in the KG

[1] https://github.com/nelsonhuangzijian/WN18RR-QA

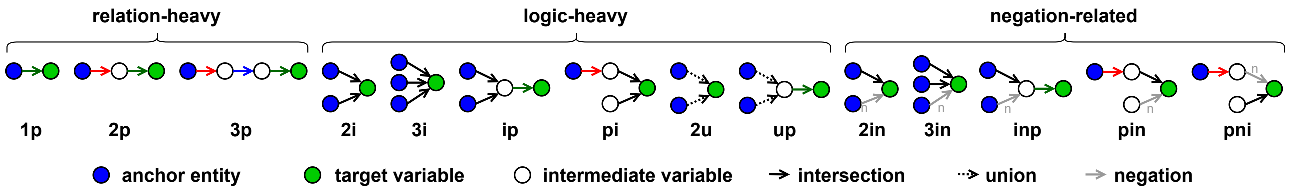


Figure 7: Illustration of the query structures for 14 query types with relation-specific projections (p), intersection (i), union (u) and negation (n) operations. Different relations are depicted in different colors (red/blue/green arrows).

Table 5: Examples of constructed queries and their answer sets.

Logical Query	**Answer**
"**2p**" $q = V_?.\ \exists V_1 :\ hasPart(England, V_1) \wedge hasPart(V_1, V_?)$	Tower of London
"**ip**" $q = V_?.\ \exists V_1 :\ (derivationallyRelatedForm(V_1, Britannic) \wedge hasPart(V_1, England)) \wedge memberOfDomainRegion(V_1, V_?)$	Panda Car
"**inp**" $q = V_?.\ \exists V_1 :\ (hasPart(V_1, NorthernIreland) \wedge \neg hasPart(V_1, England)) \wedge memberOfDomainRegion(V_1, V_?)$	Gaelic (Irish)

Table 6: Number of training, validation, and testing queries generated for different query structures.

KG	Query	1p	2p	3p	2i	3i	ip	pi	2u	up	2in	3in	inp	pin	pni
FB15k-237	Training	149,689	149,689	149,689	149,689	149,689	-	-	-	-	14,968	14,968	14,968	14,968	14,968
	Validation	20,101	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
	Testing	22,812	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
NELL995	Training	107,982	107,982	107,982	107,982	107,982	-	-	-	-	10,798	10,798	10,798	10,798	10,798
	Validation	16,927	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000
	Testing	17,034	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000	4,000
WN18RR	Training	103,509	103,509	103,509	103,509	103,509	-	-	-	-	10,350	10,350	10,350	10,350	10,350
	Validation	5,202	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
	Testing	5,356	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000

Table 7: Ablation study on WN18RR for formulations of union operator in MRR (%).

Model	1p	2p	3p	2i	3i	ip	pi	2u		up		2in	3in	inp	pin	pni	avg_p	avg_l	avg_n
								DNF	**U**	**DNF**	**U**								
$\text{LinE}_{\alpha,\beta}$	45.12	12.35	6.70	47.11	67.13	14.73	24.87	**8.49**	7.00	**6.93**	5.99	12.50	60.81	7.34	5.20	7.74	21.39	28.21	18.72

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, $\mathcal{T}_Q = \{q_1, q_2, \cdots, q_n\}$ and $\mathcal{T}_\mathcal{A} = \{q_1[V_?], q_2[V_?], \cdots, q_n[V_?]\}$, 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 each query type of (2in/3in/inp/pin/pni), we generate at one tenth the "1p" query (i.e., 10,350) as training set. For validation and testing queries, we generate approximately one fifth 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 $\text{LinE}_{\alpha,\beta}$ 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.