Incomplete Knowledge Graph Query via Fuzzy Reasoning with Efficiently Calibrated Inference

ABSTRACT

Querying incomplete knowledge graphs, including both factoid queries for entity retrieval and aggregate queries for statistical summarization, is critical for downstream applications. However, existing fuzzy reasoning models, while adept at handling uncertainty through continuous membership degrees, face two fundamental challenges: (1) error propagation, where approximation inaccuracies accumulate over multi-hop reasoning paths and are catastrophically amplified in aggregate operations; and (2) the efficiency-robustness trade-off, where robust, stepwise verification is computationally prohibitive, while efficient, direct inference suffers from uncontrolled error.

To address these challenges, we propose FRECI, a novel framework that integrates a high-recall fuzzy reasoning model with an efficiently calibrated inference pipeline. First, our fuzzy reasoning model constructs a high-quality answer candidate set by innovatively redesigning logical operators (projection, conjunction) and integrating hierarchical type information, thereby mitigating semantic inconsistency and early-stage error propagation. Building upon the fuzzy set output by reasoning model, our second contribution is an efficiently calibrated inference process designed for both of two query types. For factoid queries, we design a score for projections with early-scoring mechanisms to efficiently refine candidates. For aggregate queries, we develop a sampling strategy guided by projection scores and derive optimized, sparsity-aware confidence intervals to ensure statistical reliability. Extensive experiments on benchmark datasets demonstrate that FRECI significantly outperforms several baselines in both factoid and aggregate queries, validating its effectiveness in balancing robustness, and computational efficiency for querying incomplete KGs.

PVLDB Reference Format:
Jingyi Qiu, Aibo Song, Tianbo Zhang, Luoyu Mei, Xinyu Zhang, and Yizhu Wang. Incomplete Knowledge Graph Query via Fuzzy Reasoning with Efficiently Calibrated Inference. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

Incomplete Knowledge Graphs (KGs) pose significant challenges for effective querying due to their inherent incompleteness and noise [14, 24, 25]. KGs organize real-world knowledge as triples (h, r, t), representing entities and their relationships; these structures are foundational to critical applications such as semantic search semantic search [19], recommendation [22, 26, 49] and question answering [3, 23]. As KGs expand to millions of entities [21], the demand for efficient querying mechanisms that manage uncertainty becomes increasingly crucial. KG queries are typically classified into two types [38]: *factoid queries*, which retrieve specific entities based on given conditions, and *aggregate queries*, which compute statistical summaries over query results, as illustrated in Fig. 1. Both query types are prevalent in real-world applications [38] and are significantly affected by the inherent incompleteness of practical KGs.

Current KG querying paradigms face limitations due to this incompleteness. Subgraph matching techniques often fail because they demand strict isomorphism [16]. Knowledge graph embedding models operate within continuous space but can propagate errors in multi-hop queries, compromising reliability [11]. For complex queries, logical reasoning provides a structured approach by decomposing them into sequences of operators (e.g., projection, conjunction) executed in the embedding space, allowing for traceable multi-hop inference [29, 33]. Nonetheless, these approaches commonly yield binary outputs and lack a nuanced representation of uncertainty for approximate querying [5, 27].

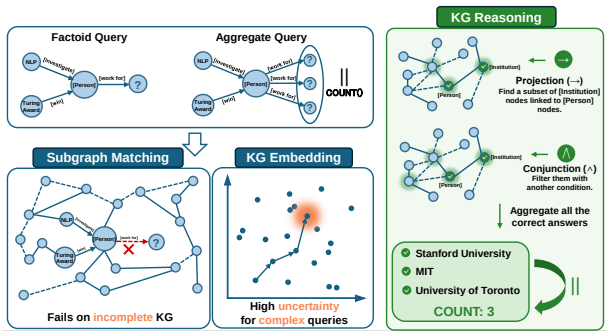


Figure 1: Examples of KG factoid and aggregate queries.

To address these challenges effectively, fuzzy set theory presents a promising framework. By enhancing logical reasoning with continuous membership degrees, it allows for direct incorporation of uncertainty into the querying process [1, 47]. This integration not only preserves the interpretability of operator-based reasoning but also enables the effective management of imprecision, an approach referred to as fuzzy logical reasoning [8, 33]. Leveraging fuzzy reasoning can thus lead to more effective querying of incomplete KGs, accommodating the complexities and uncertainties characteristic of real-world data.

Despite the strengths of fuzzy reasoning, two significant challenges persist when applying it to complex factoid and aggregate queries over incomplete KGs: (1) *Error propagation:* The vector-based nature of fuzzy reasoning leads to information loss with each reasoning step. Complex factoid queries often involve intricate semantic constraints that these reasoning models struggle to adequately capture, resulting in cumulative error propagation. Additionally, statistical operations (e.g., COUNT, AVG) used in aggregate queries do not simply sum values; they can amplify individual inaccuracies, producing unreliable aggregated results [44, 48]. Consequently, the error tolerance of current fuzzy reasoning models falls short of the precision required for effectively addressing these complex and aggregate query scenarios; (2) Efficiency-robustness trade-off: Complex factoid and aggregate queries often require evaluating numerous projection operations, especially over multi-hop paths. Executing these projections in a strictly sequential, single-hop manner—though robust—incurs prohibitive computational cost (e.g., runtime) as the number of hops increases. Conversely, attempting to assess a multi-hop path in one shot through direct embedding composition is efficient but suffers from the unconstrained error propagation described above, yielding low-quality results. This creates a fundamental trade-off between efficiency and robustness in the inference process for complex queries.

To bridge these gaps, we propose a novel approach called the Fuzzy Reasoning model with Efficient Calibrated Inference (FRECI) for queries over incomplete KGs. The core idea of FRECI is to perform efficient calibrated inference with designed projection score using a fuzzy reasoning representation that combines operator optimization and type information integration. Specifically, our fuzzy reasoning model enhances fundamental logical operators (projection and conjunction) through a redesign that aims to mitigate error propagation effectively. To further address the challenge of semantic inconsistencies, hierarchical type information is incorporated to uphold semantic coherence within the fuzzy set. Building upon this fuzzy reasoning foundation, we then introduce an efficient calibrated inference process for both factoid and aggregate queries. For factoid queries, we design an early-scoring mechanism for both single-hop and multi-hop projection to refine the retrieved fuzzy sets. For aggregate queries, we tackle the inherent low-recall challenge in incomplete KGs by deriving an optimized confidence interval that incorporates the Wilson score, thereby ensuring statistical reliability for the resultant aggregated outputs. Crucially, FRECI maintains the efficiency of embedding-based approaches while leveraging the robustness of fuzzy set theory and the precision of calibrated inference.

The main achievements of this study are as follows:

- We propose a novel fuzzy reasoning model for constructing a fuzzy set with high recall, which integrates innovatively optimized logical operators. Subsequently, we incorporate hierarchical type information for initialization and training, effectively tackling the error propagation problem associated with complex factoid and aggregate queries.
- We develop an efficient calibrated query inference mechanism that ensures both the efficiency and robustness of factoid query results, providing a statistically sound confidence interval for aggregate queries. This innovation successfully addresses the critical issue of achieving the efficiency-robustness trade-off in multi-hop inference over fuzzy sets.
- Extensive experiments on benchmark datasets demonstrate the superior performance of FR-ECI in both factoid and aggregate queries, validating its robustness and showcasing the practical benefits of integrating fuzzy reasoning with calibrated inference for managing incomplete knowledge.

The structure of this paper is organized as follows: Section 2 introduces the definitions and notations employed throughout the study. Sections 3 and 4 provide a detailed presentation of the fuzzy reasoning model and query inference model, respectively. In Section 5, we evaluate FRECI from various perspectives. Finally, Section 6 and Section 7 discuss related work and present the conclusions drawn from our research, respectively.

2 PRELIMINARY

Definition 2.1 (Knowledge Graph). A knowledge graph (KG) can be denoted as $G = (E, R, T)$, where E and R represent the sets of entities and relations, respectively. The symbol $T = \{(h, r, t)|h, t \in E, r \in R\}$ denotes the set of triples, such that $T \subseteq E \times R \times E$. The triple (h, r, t) signifies the association between the head entity h and the tail entity t through the relation r.

Definition 2.2 (Factoid Query). A factoid query q is defined in the first-order logic form as follows:

$$q = v_? \; \exists v_1, \cdots, v_k : \phi(v_?, v_1, \cdots, v_k, E, R) \tag{1}$$

where $v_?$ is the entity we intend to find, and $v_1, \cdots, v_k$ are the existential entity variables. The function ϕ represents the logical conjunction of a series of truth values (where each value is either 0 or 1). Each expression in ϕ is any of the following forms: $\mathbb{I}[(v_i, r, v_j) \in R]$, $\mathbb{I}[(h, r, v_i) \in R]$ or $\mathbb{I}[(v_i, r, v_?) \in R]$, where $v_i, v_j \in$

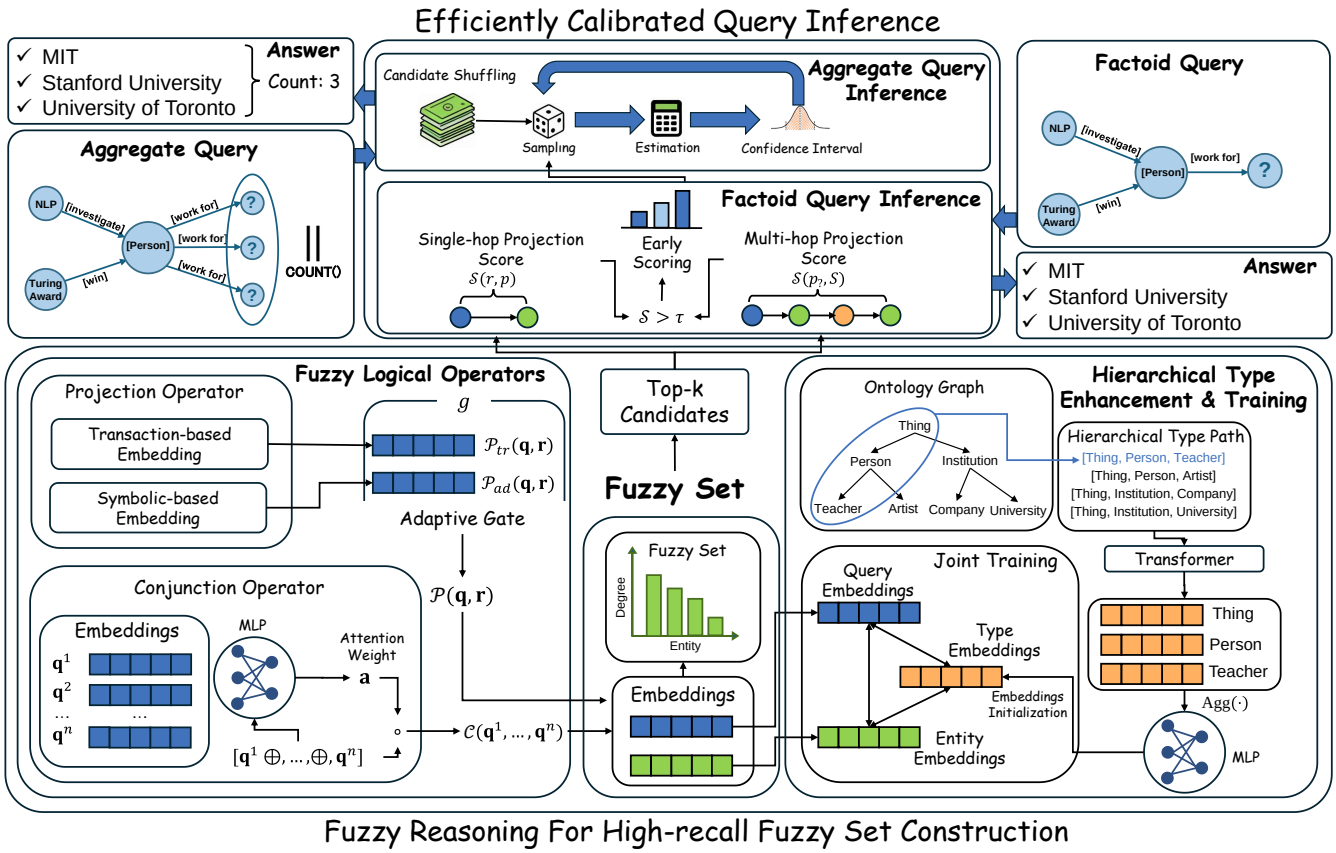


Figure 2: Overview of FRECI.

$\{v_1, \cdots, v_k\} \cup \{v_?\}$, $h \in E$ and $\forall r \in R$. The function $\mathbb{I}$ is a logical indicator function that returns 1 if its argument is true and 0 otherwise.

Example. In Fig. 1, the factoid query is formulated as $q = v_? \exists u : \phi(v_?, u, E, R) = \mathbb{I}[(\text{NLP}, \text{investigate}, u) \in R] \wedge \mathbb{I}[(\text{Turing Award}, \text{win}, u) \in R] \wedge \mathbb{I}[(u, \text{work for}, v_?) \in R]$.

Definition 2.3 (Aggregate Query). An aggregate query is defined as $\mathcal{A} = (q, f)$, where q is a factoid query and f is an aggregate function, e.g., COUNT. This type of query is essential for deriving statistical insights from the data represented in the KG.

Definition 2.4 (Embeddings). Given an entity h, its embeddings are denoted as $\mathbf{h} \in \mathbb{R}^d$ (the embedding of entity t is $\mathbf{t}$), where d is the dimension of the embeddings. The embeddings for a relation r and a query q are represented as $\mathbf{r}$ and $\mathbf{q}$, respectively, both having d-dimensional embeddings. For simplification, we represent the matrix $\mathbf{E} \in \mathbb{R}^{|E| \times d}$, which contains all the entity embeddings in E. The matrix $\mathbf{R}$ stores the embeddings of all relations in R.

Overview. The overall framework of FRECI is illustrated in Figure 2. To achieve robust and precise querying over incomplete knowledge graphs, FRECI introduces a two-stage process that aligns with the core objectives of fuzzy systems for handling uncertainty and partial truth: (1) *Dual Granularity-Enhanced Fuzzy Reasoning:* This stage integrates optimized operators and hierarchical type information to robustly handle structural uncertainty, producing a fuzzy set with candidate entities and their associated membership degrees. This process inherently models query answering as a form of fuzzy reasoning, where answers are assigned membership degrees reflecting their compatibility with incomplete evidence; (2) *Efficiently Calibrated Query Inference:* To ensure the reliability of the reasoned outputs, this stage refines the results through tailored calibration mechanisms. It employs projection-based scoring for factoid queries and calibrated projection score-guided sampling with optimized confidence intervals for aggregate queries. This refinement guarantees that the final query results are accompanied by well-calibrated confidence measures, meeting the evaluation standards for robustness in uncertain environments.

3 FUZZY REASONING FOR HIGH-RECALL FUZZY SET CONSTRUCTION

To directly address the error propagation challenge, this section presents our enhanced fuzzy reasoning model. Its primary objective is to construct a high-recall fuzzy set F that comprehensively captures potential answers, with their membership degrees reflecting their likelihood of being correct. This high-recall, uncertainty-aware representation provides the essential foundation for the subsequent inference stage. As depicted in the lower part of Figure 2, our model integrates two synergistic innovations: (1) *Enhanced*

Fuzzy Logical Operators These operators robustly handle uncertainty through a process known as neural-symbolic fusion (Section 3.1); (2) *Hierarchical Type Incorporation* This ensures semantic consistency with query constraints, aiding the interpretation of complex relationships (Section 3.2). These components are jointly optimized through a unified training objective (Section 3.3), which aligns entity, query, and type embeddings. This systematic approach facilitates the construction of high-quality fuzzy sets F that are resilient to the error propagation inherent in reasoning over incomplete knowledge graphs.

3.1 Fuzzy Logical Operators

The fuzzy logical operators form the computational backbone for systematically constructing the high-recall fuzzy set F. They operate by transforming a logical query q into a composite query embedding $\mathbf{q}$, which is then compared with all entity embeddings to generate the final fuzzy set F. In this set, each entity's membership degree reflects its semantic proximity to$\mathbf{q}$, enabling a more nuanced understanding of potential answers. To robustly address error propagation, we innovatively redesign the projection and conjunction operators while employing standard formulations for disjunction and negation.

3.1.1 Projection Operator. The projection operator is fundamental for traversing relations while constructing the high-recall fuzzy set F. It addresses a critical challenge in incomplete KGs: accurately estimating which entities are reachable via a relation r from a source set, despite missing edges that could lead to cascading errors in multi-hop and aggregate queries. Formally, given $E_q \subseteq E$ and $r \in \mathcal{R}$, it outputs: $E_{\mathcal{P}} = \{t \mid \exists h \in E_q, (h, r, t) \in T\}$. In embedding-based reasoning, this corresponds to a one-step link prediction task, requiring high precision to reliably support downstream reasoning. To achieve robust performance over incomplete KGs, we adopt the well-validated TransE model [7] as the backbone. This choice is influenced by the observation that simpler translation-based models like TransE often display stronger robustness to missing facts compared to more complex models, likely owing to their stable semantic principle ($\mathbf{h} + \mathbf{r} \approx \mathbf{t}$).

In our framework, the projection operation is realized by a function $\mathcal{P}(\mathbf{q}, \mathbf{r})$ that operates in the vector space. Given a query embedding $\mathbf{q}$ (which may represent a single entity or a set of entities), the function aims to produce a new embedding $\mathbf{q}' = \mathcal{P}(\mathbf{q}, \mathbf{r})$. This output is designed to represent the potential tail entities in $E_{\mathcal{P}}$, effectively transforming the input embedding towards regions of entities likely connected via relation r.

However, TransE can suffer from error propagation in multi-hop reasoning: minor inaccuracies in early hops can accumulate, distorting the final answer. To combat this accumulation, we integrate symbolic reasoning. Even an incomplete KG contains deterministic connectivity information—existing edges provide reliable priors that can correct accumulating errors from purely embedding-based translations. For instance, in the two-hop query Turing Award $\xrightarrow{\text{win}}$ [Person] $\xrightarrow{\text{work for}}$?, if the KG already records Turing Award $\xrightarrow{\text{win}}$ Hinton $\xrightarrow{\text{work for}}$ MIT, these known edges offer a deterministic path that counters the drift caused by successive TransE operations. Thus, our projection operator is not simply a translation mechanism but an enhanced backbone informed by symbolic priors. Formally, the projection embedding $\mathcal{P}(\mathbf{q}, \mathbf{r})$ is computed using an adaptive gating mechanism:

$$\mathcal{P}(\mathbf{q}, \mathbf{r}) = g \circ \mathcal{P}_{tr}(\mathbf{q}', r) + (1 - g) \circ \mathcal{P}_{ad}(\mathbf{q}', r)$$

where $\circ$ denotes element-wise multiplication. The adaptive gate g dynamically balances the two components based on the specific query context and the availability of symbolic evidence. The gate g is calculated by:

$$g = \text{MLP}(\mathcal{P}_{tr}(\mathbf{q}, \mathbf{r}) \oplus \mathcal{P}_{ad}(\mathbf{q}, \mathbf{r}))$$

where MLP represents a multi-layer perceptron with Sigmoid (the latest layer) and ReLU (in earlier layers) as activation functions, and $\oplus$ denotes the concatenation operation. The embedding $\mathcal{P}_{tr}(\mathbf{q}, \mathbf{r})$ denotes the translation component as $\mathcal{P}_{tr}(\mathbf{q}, \mathbf{r}) = \mathbf{q} + \mathbf{r}$. Meanwhile, $\mathcal{P}_{ad}(\mathbf{q}, \mathbf{r})$ represents the symbolic component, which encodes deterministic local connectivity from the adjacency matrix $\mathbf{M}_r$ corresponding to relation r. Given the head entity h and tail entity t with their embeddings stored in the i-th row and the j-th row of $\mathbf{E}$, the (i, j)-th entry m_{ij} of $\mathbf{M}_r$ is 1 if $(h, r, t) \in R$; otherwise $m_{ij} = 0$. Therefore, we compute:

$$\mathcal{P}_{ad}(\mathbf{q}, \mathbf{r}) = \text{Agg}\left(\boldsymbol{s}' \cdot \mathbf{E}\right)$$

$$\boldsymbol{s}' = \frac{\boldsymbol{s} \cdot \mathbf{M}_r}{||\boldsymbol{s} \cdot \mathbf{M}_r||_1}$$

where $\boldsymbol{s}$ is a multi-hot embedding with dimension $|E|$ [41]. The Agg is the mean aggregation function and $||\cdot||_1$ is the L1 norm of the input.

The final output of the projection operator is the fuzzy set $F_{\mathcal{P}}$, computed as:

$$F_{\mathcal{P}} = \text{Sigmoid}(\mathbf{E} \cdot \mathcal{P}^{\top}(\mathbf{q}, \mathbf{r})) \tag{2}$$

where $\mathcal{P}^{\top}(\mathbf{q}, \mathbf{r})$ is the transpose of $\mathcal{P}(\mathbf{q}, \mathbf{r})$. This fuzzy set $F_{\mathcal{P}}$ provides a distribution over all entities, where each value represents a continuous membership degree reflecting the entity's likelihood of being reachable via relation r. By outputting such a graded membership distribution rather than a hard set, the projection operator directly contributes to constructing the comprehensive, high-recall fuzzy set F that is resilient to the challenges posed by missing edges and error propagation in incomplete KGs.

3.1.2 Conjunction Operator. The conjunction operator addresses a critical challenge in constructing the high-recall fuzzy set F: reliably intersecting multiple fuzzy sets while minimizing the amplification of noise and uncertainty from individual sub-queries. This problem epitomizes error propagation, particularly in complex and aggregate queries where the containment of such errors is essential. Formally, given n input factoid queries $\{q^1, \ldots, q^n\}$ with their corresponding answer entity sets $\{E_q^1, \ldots, E_q^n\}$, the conjunction operator selects the entities that satisfy all input queries, i.e., the set intersection $E_C = \bigcap_{i=1}^{n} E_i$. This corresponds to answering the composite logical query $q = q_1 \wedge \cdots \wedge q_n$. To realize this operation in the embedding space, we introduce a function C as the conjunction operator that maps the set of query embeddings $\{\mathbf{q}^1, \ldots, \mathbf{q}^n\}$ to a single composite embedding $C(\mathbf{q}^1, \ldots, \mathbf{q}^n)$, which should be geometrically closest to the embeddings of entities in E_C.

Directly computing the intersection of multiple fuzzy sets $F_{\mathcal{P}}$ obtained from projection operations can unduly amplify noise introduced by $\mathcal{P}$. Low-confidence scores from any sub-query can

propagate through the set operation, depressing the overall membership degrees. To mitigate this form of error propagation, we implement C using a learned attention mechanism over the sub-query embeddings. Unlike a strict fuzzy intersection (e.g., taking the element-wise minimum), which penalizes all low-confidence scores equally, our attention mechanism learns to create a context-aware composite embedding. It achieves this by evaluating all sub-queries $\mathbf{q}^1, \ldots, \mathbf{q}^n$ jointly. Formally, the attention weights $\mathbf{a}$ and the composite embedding are computed as:

$$\mathbf{a} = \text{MLP}([\mathbf{q}^1 \oplus, \ldots, \oplus \mathbf{q}^n])$$

$$C(\mathbf{q}^1, \ldots, \mathbf{q}^n) = [\mathbf{q}^1 \oplus, \ldots, \oplus \mathbf{q}^n] \circ \mathbf{a}$$

Intuitively, the MLP can assign lower weights to sub-query embeddings that are outliers or semantically inconsistent within the current conjunction context, thereby reducing their potential noise in the final fuzzy set F_C. This provides a soft, adaptive alternative to a hard minimum operation, enhancing robustness for downstream reasoning.

Then the final fuzzy set is produced via $F_C = \text{Sigmoid}(\mathbf{E} \cdot C^\top(\mathbf{q}^1, \ldots, \mathbf{q}^n))$. This attention-based conjunction mechanism ensures that the constructed fuzzy set F_C maintains high recall by avoiding overly punitive intersections of uncertain membership degrees. Consequently, it contributes to the overall quality of the high-recall fuzzy set F while effectively controlling error propagation in multi-hop scenarios.

3.1.3 Other Operators. For logical operators beyond projection and conjunction, we utilize their standard fuzzy logic formulations [8]. This approach is grounded in the recognition that our novel operators specifically address the active error propagation inherent in inferring connections (projection) and intersecting uncertain answer sets (conjunction) within incomplete KGs. In contrast, operators like disjunction simply perform deterministic set aggregation. The output uncertainty of these operators is entirely determined by their input fuzzy sets. Consequently, no learnable parameterization is necessary, as such an adjustment would not effectively mitigate the core challenges associated with missing knowledge. Thus, our framework concentrates on the logical operators of projection and conjunction, which are particularly susceptible to error propagation, while maintaining the standard formulations for the other operators. This approach preserves logical consistency without compromising the integrity of the reasoning process.

3.2 Hierarchical Type-Enhanced Reasoning

Beyond the fuzzy reasoning operators introduced above, we integrate hierarchical type information to enhance the quality of the constructed fuzzy set F, ensuring its consistency with semantic constraints and mitigating semantic error propagation. While the fuzzy reasoning operators in Section 3.1 effectively handle relational uncertainty, they do not explicitly account for ontological constraints, such as hierarchical type relationships. This limitation is particularly critical for queries like "researchers who have won the Turing Award," which require not only matches in the KG but also that answer entities belong to the type "Researcher" or its sub-types. Traditional embedding initializations, which rely on random assignment or solely on triple facts, fail to consider ontology information. This gap can lead to type-inconsistent results in the fuzzy set F, introducing additional errors that compound in complex factoid and aggregate queries. To address this issue, we propose a hierarchical type-enhanced initialization method that injects hierarchical semantic knowledge directly into the embedding space. Our method, illustrated in Fig. 2, consists of three stages:

- Hierarchical Type Path Extraction: We perform a depth-first search on the KG's ontology graph [53] to extract all possible paths from root types to leaf types, forming a path set Ω. Each path $\omega = [o_1, o_2, \ldots, o_{|\omega|}] \in \Omega$ represents a sequence of types from general to specific (e.g., Thing $\rightarrow$ Institution $\rightarrow$ University in Fig. 2), explicitly preserving the hierarchical structure.
- Contextualized Path Encoding: Each type path ω is treated as a sequential sentence of semantic labels. We use a BERT model to encode this sequence, obtaining contextualized embeddings for each type $o \in \omega$ within its path-specific context: $[\boldsymbol{o}_1|\omega; \ldots; \boldsymbol{o}_{|\omega|}|\omega] = \text{BERT}((o_1, \ldots, o_{|\omega|}))$. Here, $\boldsymbol{o}_i|\omega$ represents the embedding of o_i in the hierarchical path ω.
- Type Representation Aggregation: A type o may appear in multiple paths. Its initialization embedding $\boldsymbol{o}$ is derived by applying a multi-layer perceptron over the average of its contextualized representations across all relevant paths to adjust the dimensionality of type embeddings:

$$\boldsymbol{o} = \text{MLP}\left(\frac{\sum_{\omega \in \Omega} \mathbb{I}[o \in \omega] \cdot \boldsymbol{o}|\omega}{\sum_{\omega \in \Omega} \mathbb{I}[o \in \omega]}\right)$$

This initialization ensures that types with similar hierarchical contexts (e.g., Teacher and Artist, both under Person) are positioned closer together in the embedding space. By providing initial embeddings incorporated with hierarchical type information, the model is guided to produce fuzzy sets F that are not only relationally plausible but also type-consistent. This semantic grounding enhances the overall quality of the constructed fuzzy set F, reduces training inefficiency, and effectively mitigates type-based error propagation in complex factoid and aggregate queries. These initialized type embeddings are directly utilized in the joint loss function (Section 3.3) to align entity, query, and type semantics within the unified fuzzy reasoning framework.

3.3 Training Objective

To jointly optimize the fuzzy reasoning operators and ontology-aware embeddings for constructing high-quality fuzzy sets F, we introduce a unified training objective. This objective ensures that the learned embeddings produce fuzzy sets F whose membership degrees accurately reflect both factual correctness and semantic type consistency—critical for mitigating error propagation in complex factoid and aggregate queries. The objective integrates three alignment losses that operate within a shared embedding space for queries, entities, and types:

$$\mathcal{L} = \lambda \mathcal{L}_n(\boldsymbol{q}, \boldsymbol{t}) + \gamma \mathcal{L}_n(\boldsymbol{q}, \boldsymbol{o}) + (1 - \lambda - \gamma)\mathcal{L}_n(\boldsymbol{t}, \boldsymbol{o}) \tag{3}$$

where $\lambda, \gamma \in [0, 1]$ are balancing hyperparameters. Here, $\mathcal{L}_n(\boldsymbol{x}, \boldsymbol{y})$ denotes the contrastive loss with negative sampling between embeddings $\boldsymbol{x}$ and $\boldsymbol{y}$, defined as:

$$\mathcal{L}_n(\boldsymbol{x}, \boldsymbol{y}) = -\log\left(\text{Sigmoid}\left(\|\boldsymbol{x}^+ - \boldsymbol{y}^+\|_1\right)\right) - \frac{1}{M}\sum_{i=1}^{M}\log\left(\text{Sigmoid}\left(\|\boldsymbol{x}^+ - \boldsymbol{y}_i^-\|_1\right)\right)$$

where $\boldsymbol{x}^+$ and $\boldsymbol{y}^+$ represent the embeddings of a positive pair (e.g., a query and its correct answer), $\boldsymbol{y}_i^-$ denotes the i-th negative sample embedding of type $\boldsymbol{y}$, and M is the number of negative samples.

Entity-Query Alignment ($\mathcal{L}_n(\boldsymbol{q}, \boldsymbol{t})$) ensures that query embeddings are closer to correct answer entities than to negative ones, directly improving the factual accuracy reflected in the fuzzy set F's membership degrees.

Type-Query Alignment ($\mathcal{L}_n(\boldsymbol{q}, \boldsymbol{o})$) aligns queries with correct answer types, enforcing that the fuzzy set F respects hierarchical type constraints—vital for queries with explicit or implicit type requirements.

Entity-Type Alignment ($\mathcal{L}_n(\boldsymbol{t}, \boldsymbol{o})$) grounds entity embeddings in type semantics, providing a coherent foundation for type-aware fuzzy reasoning and ensuring semantic consistency in the constructed fuzzy set F.

This joint formulation enables the model to learn representations where queries are simultaneously aligned with both correct answers and their hierarchical type constraints. Consequently, the resulting fuzzy reasoning model produces high-quality fuzzy sets F that are both factually accurate and semantically consistent, effectively supporting the subsequent verification stage in achieving high-precision query answering while containing error propagation across complex query structures.

4 EFFICIENTLY CALIBRATED QUERY INFERENCE OVER FUZZY SET

The high-recall fuzzy set F from Section 3 provides a comprehensive candidate pool but lacks the inference process to generate final answers. To bridge this gap with an efficiency-robustness trade-off, this section introduces an efficient calibrated inference pipeline. Our design explicitly navigates the trade-off by combining a high-recall prior (F) with a compute-efficient scoring and filtering mechanism. As illustrated in Figure 2, it implements a "fuzzy-to-inference" pipeline over two kinds of KG queries: for factoid queries, it prioritizes and calibrates candidates from F using a single/multi-hop projection score with early scoring mechanism (Section 4.1); for aggregate queries, it performs robust statistical estimation over calibrated factoid results with optimized confidence intervals (Section 4.2). This pipeline transforms the high-recall fuzzy set F into precise, confidence-calibrated query answers while explicitly managing computational cost.

4.1 Inference Process on Factoid Queries

We employ a score-filter pipeline to to concentrate computation on the most promising candidates, thereby achieving robustness without incurring the cost of exhaustive evaluation. First, it generates a focused candidate set C by selecting the top-k entities with the highest membership degrees from F. Since complex logical queries can be decomposed into sequences of fundamental projection operations, we then score each candidate's plausibility by computing a tailored projection score for both single-hop and multi-hop cases. Candidates whose final score is zero are filtered out. This design ensures computational effort is focused, effectively balancing recall with efficiency to support high-confidence inference.

Single-hop Projection Score. To score candidates for single-hop projections, we adapt the path scoring strategy from PSS [37]. This method addresses a core challenge in incomplete KGs: missing direct edges. When the correct triple (h, r, t) is absent, we instead consider indirect connecting paths $p = [(h, r_i, t'), \ldots, (t'', r_j, t)]$ between the head entity h and a candidate t. The projection score for a candidate is derived by computing the semantic similarity between the query relation r and the alternative path p (subject to a length constraint $|p| \le L$ as in [37]) using the geometric mean of cosine similarities:

$$\mathcal{S}(r, p) = \sqrt[|p|]{\prod_{r_j \in p}^{|p|} \text{Cos}(r, r_j)} \tag{4}$$

A candidate is considered calibrated for this path if $\mathcal{S}(r, p) > \tau$, where τ is a predefined confidence threshold. This scoring mechanism provides a robust basis for filtering unlikely candidates in single-hop queries, and its principle is extended to multi-hop projections as described next, ensuring a unified calibration approach across different query complexities.

Multi-hop Projection Score. While single-hop projection scores individual indirect paths, verifying multi-hop projection candidate paths demands a method that can handle missing intermediate facts and control computational efficiency. To address this, we introduce a novel calibration method that efficiently searches and scores segmented paths. For an m-hop query $p_? = [h, r_1, \ldots, r_m, ?]$, and a candidate t, we aim to find a connecting path p whose optimal segmentation into m sub-paths maximizes the sequential match to $[r_1, \ldots, r_m]$. The score for a segmentation $s = [\theta_1, \ldots, \theta_m]$ is defined as:

$$\mathcal{S}(p_?, s) = \sqrt[m]{\prod_{i=1}^{m} \text{ReLU}\left(\mathcal{S}(r_i, \theta_i) - \tau\right)} \tag{5}$$

where $\mathcal{S}(r_i, \theta_i)$ is the single-hop projection score defined in Eq. (4).

We explain the early scoring mechanism by an example shown in Figure 3: Given a 2-hop projection $p_? = (h, r_1, r_2, ?)$, candidate t, and a candidate path $p = [h \xrightarrow{r_3} h_1 \xrightarrow{r_4} h_2 \xrightarrow{r_5} t]$, we explore valid 2-segmentations, i.e., $s_1 = [[h, h_1], [h_2, t]]$ and $s_2 = [[h, h_1, h_2], [t]]$. For the segmentation s_1, it can be early scored because the score for the first segment θ_1 is computed as $\mathcal{S}(r_1, \theta_1) = 0.4 < \tau = 0.5$. Eq. (5) immediately yields $\mathcal{S}(p_?, s_1) = 0$ via the ReLU function, without needing to compute the score for the second segment θ_2. For the second segmentation s_2, its final score is 0.71. Hence, the formulation in Eq. (5) is the key to efficiency: the ReLU function causes the product to be zero if any segment score $\mathcal{S}(r_i, \theta_i) < \tau$, enabling immediate termination of scoring for that segmentation (as shown for s_1 in Figure 3). This early-scoring mechanism ensures that our inference remains efficient within the overall score-filter pipeline, as substantial portions of the search space can be abandoned with minimal computation.

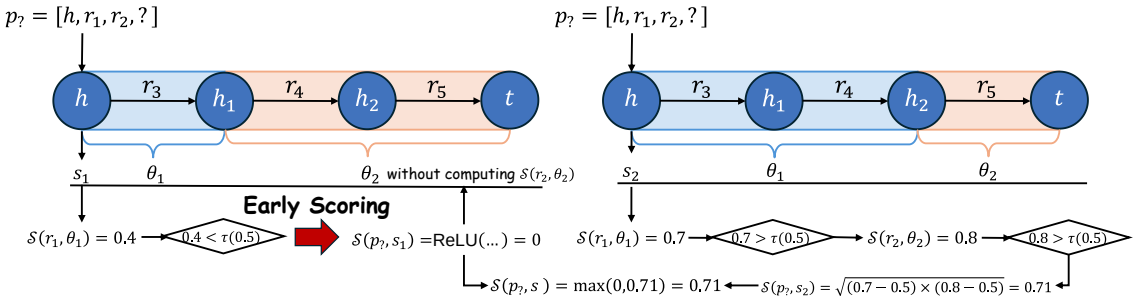


Figure 3: An illustrative example of multi-hop projection score with early scoring.

Factoid Query Inference Algorithm. Combining all aforementioned modules, Algorithm 1 presents the complete inference pipeline for factoid queries, concretely implementing the score-filter process over the fuzzy set F from Section 3. The algorithm is designed around two core principles of our framework: (1) leveraging the membership degrees in F to focus computation via Top_k(F), and (2) applying efficient, projection-based calibration with early scoring (via Eq. (5)). To efficiently find the optimal segmentation of a candidate path p that maximizes the score $\mathcal{S}(p_?, s)$ (Eq. (5)), we employ a dynamic programming (DP) algorithm enhanced with early-scoring, mechanism.

Initially, the top-k entities from the fuzzy set F are selected as the candidate set (line 1). Subsequently, the query q is decomposed into several single-hop or multi-hop projection paths $p_?$ using [2] (line 2). The following operations are then performed for each $p_?$ and each candidate t (lines 3-13): First, all candidate paths P_t connecting the known head entity h to t are extracted by [45] (line 6). For each path $p \in P_t$, the core calibration step is to compute the maximum score Δscore over all possible segmentations s of p. This is achieved by the function DP-max$(p, \mathcal{S})$ (line 7), which is based on the dynamic programming strategy [1]: if a specific segment θ is assessed to yield a score of 0 for the relation r_i within a segmentation, then the dynamic programming strategy immediately scores all segmentation that would assign segment θ to the i-th position as 0. This optimization, grounded in the properties of $\mathcal{S}$, dramatically reduces the DP search space. If the optimal score Δscore for a candidate t (considering all its paths P_t) is 0, it indicates that t cannot satisfy the current query path $p_?$ under any segmentation. The candidate is therefore immediately scored to 0 and skipping further processing for this $p_?$ (line 9). Otherwise, Δscore contributes to the candidate's accumulated score (line 10). Finally, the top-k entities based on their final scores are returned as the calibrated answers A (line 13).

Optimization of k. The size k of the candidate set C directly impacts the efficiency and recall of our entire calibration pipeline. To strike an optimal balance—maintaining high recall while ensuring calibration remains computationally tractable—we determine k adaptively based on the knowledge graph scale. Specifically, we set $k = \lceil \rho \cdot |E| \rceil$, where $|E|$ is the total number of entities in the KG and ρ is a controllable fraction. This proportional scaling ensures that the candidate set grows reasonably with the KG size, preventing the calibration cost from becoming prohibitive for large graphs while still preserving coverage. The parameter ρ is tuned to achieve a target recall (e.g., >99% in our experiments) across diverse query workloads. By adapting k in this manner, we guarantee that the score-filter process operates on a candidate set that is both sufficiently comprehensive for high recall and appropriately bounded for efficiency, a crucial design choice for scalable query inference over incomplete KGs.

Algorithm 1 FactoidQueryInference

Require: Query q, fuzzy set F
Ensure: Answers A
1: $C \leftarrow$ Top_k(F)
2: Decompose query q into several projection paths $p_?$
3: **for** each $p_?$ **do**
4: **for** each candidate $t \in C$ **do**
5: t.score $\leftarrow 0$; $h \leftarrow$ HeadEntity($p_?$)
6: **for** each path $p \in P_t$ **do**
7: Δscore $\leftarrow$ DP-max($p, \mathcal{S}$) {Early scoring via Eq. (5)}
8: **end for**
9: **if** Δscore = 0 **then** t.score $\leftarrow 0$; **continue**
10: t.score += Δscore
11: **end for**
12: **end for**
13: $A \leftarrow$ Top_k(C)
14: **return** A

4.2 Inference Process on Aggregate Queries

4.2.1 Algorithm Workflow. To extend our efficiently calibrated query inference framework to aggregate queries (e.g., COUNT, SUM, AVG), we must address two amplified challenges in incomplete KGs: (1) Error Propagation, where inaccuracies from individual factoid calibrations compound in the aggregate result, and (2) Computational Inefficiency, as naively verifying all candidates is prohibitive—a manifestation of the efficiency-robustness trade-off.

Building upon the calibrated factoid query inference (Section 4.1), we adapt an online aggregation paradigm with three key innovations designed for efficiency and robustness: (1) using calibration scores directly for sampling, replacing error-prone random walks; (2) partitioning the candidate set for efficient processing; and (3) employing robust confidence intervals (e.g., Wilson score) suitable for sparse results. This approach ensures that aggregate estimation is both computationally tractable and statistically reliable, directly leveraging the calibrated outputs from the previous stage.

Algorithm 2 realizes our efficient aggregate inference workflow. It begins with the high-recall fuzzy set F (line 1), focusing on the top-k candidates. The key efficiency innovation is partitioning them into N shuffled subsets $\{C_1, \ldots, C_N\}$ (line 2). This structure enables sequential processing of smaller subsets and—crucially—allows the algorithm to terminate once the confidence criterion is met, often

Algorithm 2 AggregateQueryInference

Require: Aggregate query $\mathcal{A}$, reasoned fuzzy set F, significance level α, user-specific error bound b, shuffle number N
Ensure: Estimated aggregate value $\hat{y}$
1: $C \leftarrow$ Top-$k(F)$
2: $\{C_1, \ldots, C_N\} \leftarrow$ Shuffle(C, N) {Partition candidates into shuffled subsets}
3: $A \leftarrow \emptyset$, $\boldsymbol{\pi} \leftarrow \emptyset$
4: **for** each partition C_i **do**
5: $\quad \boldsymbol{\pi} \leftarrow \boldsymbol{\pi} \cup$ SamplingProbabilities$(\mathcal{A}, C_i, N)$ {Compute probabilities per partition}
6: $\quad$ **while** the half width of CI less than or equal to $\frac{y \cdot b}{1+b}$ or $i \leq N$ **do**
7: $\quad\quad A \leftarrow A \cup$ AddSample$(q, C_i, \boldsymbol{\pi})$
8: $\quad\quad \hat{y} \leftarrow$ Estimate(q, A)
9: $\quad\quad$ CI $\leftarrow$ ConfidenceInterval$(A, \hat{y}, \alpha, b)$
10: $\quad$ **end while**
11: $\quad$ **if** the half width of CI less than or equal to $\frac{y \cdot b}{1+b}$ **break**; **end if**
12: **end for**
13: **return** $\hat{y}$

without processing all subsets, thereby guaranteeing efficiency. For each partition C_i, sampling probabilities $\boldsymbol{\pi}$ are derived from projection scores via SamplingProbabilities (line 5), replacing error-prone random walks. The core loop performs iterative sampling and estimation: it adds samples to the sample set A according to sampling probability $\boldsymbol{\pi}$ (line 7), recomputes the aggregate estimate $\hat{y}$ using the Horvitz-Thompson estimator [38] (line 9), and updates the confidence interval CI (line 10). The loop terminates when the half-width of CI meets the relative error bound $\frac{\hat{y} \cdot b}{1+b}$ derived in [38] (line 12). This partition-aware, score-driven sampling directly addresses the challenges of error propagation and computational cost, providing an efficient pathway from the fuzzy set F to a reliable aggregate estimate $\hat{y}$.

4.2.2 Sampling using projection score. The projection scores t.score produced during factoid query inference (Algorithm 1) are an ideal basis for defining sampling probabilities in aggregate queries. These scores inherently integrate two complementary sources of evidence: they incorporate prior knowledge from existing edges in the KG via symbolic reasoning, while also leveraging reasoning embeddings to infer plausible connections for missing edges. Furthermore, they are computed efficiently through our projection-based calibration with early termination. This dual grounding makes the scores robust indicators of a candidate's likelihood of being correct. Therefore, we directly use the projection scores—avoiding the flaw of error-prone, on-the-fly random walks—and normalize them to define valid, efficiency-preserving sampling probabilities for our aggregate query inference.

Algorithm 3 SamplingProbabilities

Require: Query $\mathcal{A}$, partitioned candidate set C_i, shuffle number N
Ensure: Probability vector $\Delta\boldsymbol{\pi} = \{\pi_t \mid t \in C_i\}$
1: $\Delta\boldsymbol{\pi} \leftarrow \emptyset$, $W_{\text{total}} \leftarrow 0$
2: Extract the factoid query q from aggregate query $\mathcal{A}$
3: $\Delta A \leftarrow$ FactoidQueryInference(q, C_i) {Returns answers with scores}
4: $W_{total} \leftarrow$ Sum$(t.\text{score})$ {Sum scores of candidates}
5: **for** each candidate $t \in \Delta A$ **do**
6: $\quad \pi_t \leftarrow t.\text{score}/(W_{\text{total}} \cdot N)$
7: $\quad \Delta\boldsymbol{\pi} \leftarrow \Delta\boldsymbol{\pi} \cup \{\pi_t\}$
8: **end for**
9: **return** $\Delta\boldsymbol{\pi}$

The probability computation is formalized in Algorithm 3. It takes the aggregate query $\mathcal{A}$, a partition C_i from the shuffled set, and the partition count N. First, it extracts the core factoid query q_f embedded within $\mathcal{A}$ (line 3). It then executes the factoid inference (Algorithm 1) on C_i to obtain a set of answers ΔA, each with its accumulated projection score t.score (line 4). The total score W_{total} is the sum of all scores in ΔA (line 5). For each answer $t \in \Delta A$, its sampling probability is calculated as $\pi_t = t.\text{score}/(W_{\text{total}} \cdot N)$ (lines 6-9). The factor N ensures proper normalization across all partitions. For candidates are scored with 0, they are calibrated as an erroneous candidate and implicitly receive a probability of zero.

4.2.3 Optimization of Confidence Interval. The ConfidenceInterval function must produce reliable confidence intervals for aggregate estimates $\hat{y}$ despite the sparse, zero-inflated distributions typical of query results in incomplete KGs. Conventional variance estimators fail here, yielding intervals that are either overconfidently narrow or excessively wide. To address this, we develop a hybrid confidence estimation strategy that dynamically adapts to the specific aggregation type and data sparsity. Our core innovation is a zero-inflated variance correction for AVG/SUM queries that explicitly decomposes uncertainty, coupled with the use of Wilson score intervals for COUNT queries to handle extreme proportions robustly.

Given the confidence level $1 - \alpha$, the confidence interval is calculated via the central limit theorem [48] and the unbiasedness of the Horvitz-Thompson estimator [18]:

$$\text{CI} = \hat{y} \pm z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{|A|}} \tag{6}$$

where σ is the standard deviation of the estimator and $z_{\alpha/2}$ is the normal critical value with right-tail probability $\alpha/2$. The key challenge is accurately estimating σ under zero inflation.

For AVG/SUM queries, we correct the variance estimation by explicitly modeling the two sources of uncertainty in zero-inflated data: (1) whether a sampled answer is correct (non-zero), and (2) the value distribution of correct answers. This modeling is naturally supported by our projection score, i.e., t.score for a candidate is zero if it is calibrated as incorrect (Algorithm 1, lines 8-9), and positive otherwise. Let δ denote the estimated proportion of correct answers in the current sample set A, which corresponds to the proportion of samples with a non-zero t.score. This leads to the corrected variance estimator:

$$\hat{\sigma}^2_{\text{AVG}} = \delta \cdot \text{Var}\left(y_t^+\right) + \delta(1-\delta) \cdot \left(\text{Mean}(y_t^+)\right)^2, \tag{7}$$

where δ is the estimated proportion of correct (non-zero) answers, and $\text{Mean}(y_t^+)$ and $\text{Var}\left(y_t^+\right)$ are the mean and variance of the correctly calibrated answers. The first term captures variance within correct answers. The second term is critical for sparse results: it quantifies the substantial variance introduced by the inherent uncertainty in whether a randomly sampled candidate is even correct, which dominates when δ is small. This decomposition allows the variance estimate to adapt to the actual sparsity level δ, which is inferred directly from the calibration outcomes. For SUM queries, the variance scales with the square of the candidate set size: $\sigma^2_{\text{SUM}} = |C|^2 \cdot \sigma^2_{\text{AVG}}$.

For COUNT queries, the zero-inflated model reduces to the standard binomial variance $\hat{\sigma}^2_{\text{COUNT}} = \delta(1-\delta)$ (by setting $\text{Mean}(y_t^+) = 1$ and $\text{Var}(y_t^+) = 0$ in Eq. (7)). Here, δ simply represents the fraction of sampled candidates that were calibrated as correct (non-zero t.score). However, when δ is near 0—common in incomplete

KGs—this can produce invalid confidence boundaries (e.g., negative lower bounds). Therefore, for COUNT queries we employ the Wilson score interval, which is specifically designed for extreme proportions and remains bounded in $[0, 1]$:

$$\mathrm{CI} = \frac{\delta + \frac{z^2_{\alpha/2}}{2|A|}}{1 + \frac{z^2_{\alpha/2}}{|A|}} \pm \frac{z_{\alpha/2}}{1 + \frac{z^2_{\alpha/2}}{|A|}} \sqrt{\frac{\delta(1-\delta)}{|A|} + \frac{z^2_{\alpha/2}}{4|A|^2}} \tag{8}$$

The Wilson interval incorporates the estimation uncertainty of δ itself, providing superior coverage for sparse results.

Thus, our ConfidenceInterval function implements a hybrid strategy: for AVG and SUM queries, it computes the interval via Eq. (6) using the corrected variance from Eq. (7); for COUNT queries, it directly applies the Wilson interval from Eq. (8). This approach ensures robust confidence estimation across all aggregation types by directly addressing the statistical challenges posed by zero-inflated distributions in incomplete knowledge graphs, while firmly rooting the statistical parameters in our earlier calibration results.

5 EXPERIMENT

5.1 Experimental Setup

Environment. All experiments were conducted on a Linux server running Ubuntu 22.04, equipped with an Intel i7-8700 CPU, 64 GB of RAM, and two NVIDIA GeForce RTX 3080 Ti GPUs (each with 12 GB of VRAM). Our implementation is based on Python 3.8.4 and PyTorch 2.0.0 with CUDA 12.5 support.

Datasets. We conduct our experiments on two knowledge graphs: DBpedia [4] and YAGO4 [31]. DBpedia contains 28,824 entities with 327 distinct relations and 981 entity types, while YAGO4 comprises 32,465 entities with by 75 relations and 8,382 types. For factoid queries, we adopt the 14 query structures originally defined in [29] as illustrated in Figure 4, with 100 queries per structure. To evaluate the model's generalization ability, the model is trained on a subset of structures (projection and conjunction) and tested on all structures, including those with unseen but complex logical compositions. Aggregate queries are derived directly from the factoid test queries in DBpedia by applying COUNT, SUM, and AVG functions to their answer sets. The final benchmark comprises 100 queries per aggregation function.

Baselines. For a comprehensive comparison of factoid queries, we select five baselines in KG reasoning, including neural network-based reasoning GQE [16], geometric transformation-based reasoning Query2Box [28], probability-based reasoning BetaE [29], fuzzy logic-based reasoning FuzzyQE [8], and multi-hop logical reasoning LQAC [33]. For aggregate query baselines, we select subgraph matching-based methods including VF2 [10], Gup [2], and embedding-based methods EAQ [20] and AQS [38].

Metrics. To simulate knowledge graph incompleteness, we randomly mask 5% of triples during training and query inference. First we rank the entities by their scores obtained in Algorithm 1, i.e., t.score. We then evaluate factoid query performance on the complete KG by using two complementary metrics: Hits at N (Hit@N) [28] and Mean Reciprocal Rank (MRR) [33].

For aggregate query evaluation, we employ Average Jaccard Similarity (AJS), Mean Absolute Percentage Error (MAPE), and response time to assess both effectiveness and efficiency [38]. Since

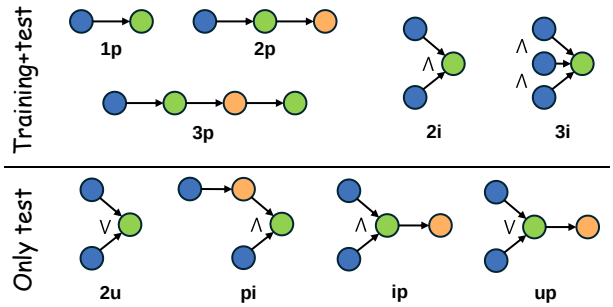


Figure 4: The query structure of all queries used for training and test, where the p, i and u stands for the projection, intersection and union, respectively.

aggregate estimates are derived from a sampled subset of candidate answers, the precision of these estimates directly depends on how many true answers are captured in the sample. To quantify this crucial sample quality, AJS measures the overlap between the ground-truth answer set A^+ (generated from the complete KG) and the returned set A, reflecting the completeness of correct answers captured—crucial for ensuring accurate aggregation. It is defined as the average of the Jaccard Similarity between A and A^+, i.e., $AJS = \mathrm{Mean}(|A \cap A^+|/|A \cup A^+|)$.

Parameter Settings. Model embeddings are configured with dimension $d = 768$, and we sample $M = 10$ negative instances per training step. The model is optimized using Adam with a learning rate of 10^{-4} and batch size of 1024. For aggregate query experiments, we set the similarity threshold $\tau = 0.95$, the user-specified error bound $b = 0.1$, and the confidence level $1 - \alpha = 95\%$.

5.2 Experimental Results

5.2.1 Results on Factoid Queries. For factoid query evaluation, we test reasoning models over nine query structures, with results summarized in Table 1. The model is trained on five query types (p and i categories) and tested on all nine types to assess generalization capability.

As shown in Table 1, FRECI achieves superior performance in most scenarios, with LQAC consistently ranking second. Specifically, FRECI attains an average absolute improvement of 8% in Hit@3 and 7.3% in MRR over the best baseline on DBpedia. On the more sparse and more type-rich YAGO4, it achieves 4.8% and 4.1% improvements in Hit@3 and MRR, respectively. For multi-hop projection queries (2p, 3p), we observe an average improvement in MRR of 10.5%, demonstrating the effectiveness of our hybrid neural-symbolic projection operator in containing error propagation. On conjunction queries (2i, 3i), we gain a maximum absolute improvement of 14.6% in Hits@3 on DBpedia, which validates the robustness of our attention-based conjunction operator in intersecting uncertain fuzzy sets. Furthermore, for complex queries involving unseen logical structures such as disjunction (2u), FRECI consistently outperforms baselines with an average improvement of 9.2% in Hits@3, highlighting the generalization capability afforded by the integration of hierarchical type constraints during reasoning.

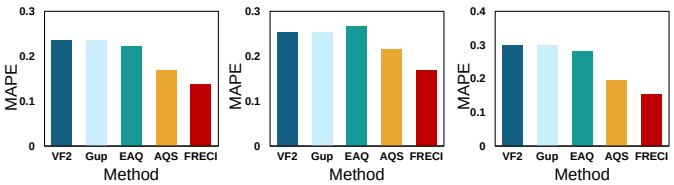
Table 1: The performance on factoid queries. The bold and underline represent the best and the second best result, respectively.

	Query	GQE		Query2Box		BetaE		FuzzyQE		LQAC		FRECI	
		Hit3	MRR	Hit3	MRR	Hit3	MRR	Hit3	MRR	Hit3	MRR	Hit3	MRR
DBpedia	1p	25.5	19.6	21.3	16.3	20.9	20.2	18.5	15.1	34.6	28.8	**37.9**	**32.6**
	2p	18.9	16.1	16.5	13.7	23.2	20.1	16.3	14.1	28.0	24.5	**28.6**	**26.5**
	3p	19.8	18.2	17.7	15.4	21.7	19.3	18.5	16.3	29.0	24.4	**29.6**	**26.9**
	2i	26.5	31.4	27.5	22.6	27.5	25.4	26.5	22.3	44.6	38.4	**56.7**	**46.8**
	3i	43.6	38.2	31.9	28.5	32.9	29.7	33.0	29.1	54.8	46.3	**69.4**	**60.5**
	pi	17.2	16.8	19.7	18.1	27.5	24.2	32.6	27.6	21.4	20.1	**60.2**	**50.6**
	ip	33.3	28.5	25.7	22.8	28.4	25.2	25.4	23.1	40.6	33.6	**43.0**	**36.1**
	2u	12.6	10.1	9.1	7.4	13.0	12.3	8.0	6.8	17.8	14.0	**18.5**	**16.4**
	up	21.1	18.0	15.4	12.9	**25.1**	**24.2**	16.0	13.7	23.0	20.6	22.3	20.3
	avg	24.2	21.9	20.7	17.5	24.5	22.3	21.6	18.7	32.6	27.9	**40.6**	**35.2**
YAGO4	1p	29.7	25.8	28.4	24.5	31.4	28.2	30.7	26.4	39.6	34.9	**41.8**	**36.6**
	2p	17.3	15.8	20.0	17.2	22.3	19.7	17.3	15.6	27.3	23.8	**28.1**	**25.7**
	3p	4.6	4.6	7.0	6.8	12.0	9.5	4.1	4.0	18.4	15.1	**19.2**	**16.5**
	2i	29.0	26.4	29.5	25.7	33.5	30.1	30.0	26.8	56.6	50.9	**62.5**	**55.9**
	3i	31.4	28.4	34.4	28.8	37.5	33.4	32.3	28.3	70.2	61.6	**71.1**	**64.9**
	pi	23.5	21.7	27.0	24.5	31.3	28.3	22.9	21.2	34.2	31.6	**60.4**	**45.1**
	ip	18.6	18.1	21.3	20.2	24.1	22.0	18.1	18.0	39.1	35.3	**42.1**	**38.3**
	2u	13.6	9.6	11.6	8.6	12.5	10.6	12.5	10.4	14.9	13.9	**20.0**	**19.2**
	up	17.4	18.2	18.5	17.7	23.8	20.7	18.6	18.1	23.7	21.3	**24.1**	**22.5**
	avg	20.6	18.7	22.0	19.3	25.5	22.5	20.7	18.8	36.0	32.0	**40.8**	**36.1**

Table 2: Average AJS of aggregate queries. The bold and underline represent the best and the second best result, respectively.

Query	DBpedia					YAGO4				
	VF2	Gup	EAQ	AQS	FRECI	VF2	Gup	EAQ	AQS	FRECI
1p	39.5	39.5	37.5	42.4	**49.2**	25.8	25.8	23.5	28.8	**32.5**
2p	39.2	39.2	38.8	43.1	**49.5**	67.6	67.6	68.1	70.8	**76.9**
3p	40.8	40.8	40.1	44.7	**50.4**	58.3	58.3	54.1	63.0	**67.3**
2i	9.2	9.2	17.2	24.7	**28.2**	11.5	11.5	12.9	14.3	**21.2**
3i	3.7	3.7	9.8	16.5	**20.8**	9.1	9.1	8.5	15.9	**19.0**
pi	17.7	17.7	15.9	32.3	**41.2**	15.7	15.7	17.3	24.0	**29.5**
ip	12.8	12.8	14.2	16.4	**22.7**	12.8	12.8	10.5	18.7	**22.8**
2u	12.7	12.7	9.1	14.8	**15.4**	12.9	12.9	11.4	14.5	**15.0**
up	23.6	23.6	20.9	28.9	**31.4**	24.3	24.3	20.6	27.5	**30.6**
avg	22.1	22.1	22.6	29.3	**34.3**	26.4	26.4	25.2	30.8	**34.9**

5.2.2 Results on Aggregate Queries. For aggregate queries, we employ Average Jaccard Similarity (AJS) and Mean Absolute Percentage Error (MAPE) to evaluate answer set quality and aggregation accuracy, respectively. Then we utilize the response time to measure the efficiency of aggregate queries. Table 2 presents AJS results on DBpedia and YAGO4, while Figure 5 categorizes MAPE by aggregation function. Table 3 reports response times to assess efficiency.



(a) COUNT (b) SUM (c) AVG

Figure 5: The MAPE of Aggregate Queries

As shown in Table 2, FRECI significantly outperforms all embedding-based baselines in answer-set quality, achieving an average relative improvement of 15% in AJS over the best baseline (AQS). This improvement is directly attributable to our reasoning model and inference pipeline, which integrates the hierarchical type information and prioritizes candidates using the projection score, respectively. The superior answer-set quality translates directly to higher aggregate query effectiveness. As shown in Figure 5, FRECI achieves the lowest MAPE across all aggregate functions, reducing the MAPE by approximately 20% compared to AQS. This strong correlation confirms that query inference quality is paramount for accurate aggregation, with our optimized confidence intervals providing additional robustness for sparse queries. In terms of efficiency, Table 3 reports that FRECI reduces the average response time by 36% compared to AQS, a gain stemming from the early-scoring mechanism in our multi-hop calibration (Algorithm 1) which avoids exhaustive calculation.

Table 3: The response time of aggregate query (ms)

Model	DBpedia			YAGO4		
	COUNT	AVG	SUM	COUNT	AVG	SUM
VF2	420.8	569.5	583.5	585.7	814.1	828.3
Gup	313.0	383.7	381.8	423.8	514.1	530.7
EAQ	585.2	360.4	654.6	523.8	568.5	583.2
AQS	127.3	136.0	203.6	171.0	178.2	238.8
FRECI	74.6	84.2	139.5	110.6	107.9	161.2

Table 4: Ablation study on reasoning model.

Model	DBpedia				YAGO4			
	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
FRECI	35.2	23.7	40.6	58.3	36.1	27.3	40.8	53.8
-op	32.6	21.5	38.7	56.1	32.5	24.3	36.5	50.4
-ht	34.3	22.4	39.6	58.0	35.2	26.4	38.8	52.7
-jl	30.8	19.5	35.1	53.6	30.3	21.4	34.1	48.2

Table 5: Ablation study on query inference.

KG	Model	MAPE			Time		
		COUNT	AVG	SUM	COUNT	AVG	SUM
DBpedia	FRECI	13.7	16.9	15.2	74.6	84.2	139.5
	-es	14.2	16.5	15.5	168.1	205.3	285.1
	-ok	27.8	25.6	30.1	120.5	141.4	203.1
	-ci	14.8	17.8	16.9	95.3	104.6	157.3
YAGO4	FRECI	18.1	15.3	17.4	110.6	107.9	161.2
	-es	17.8	15.7	17.9	190.3	251.0	289.3
	-ok	31.9	24.1	19.7	165.6	153.1	225.0
	-ci	18.5	16.4	18.1	123.9	125.8	178.4

5.3 Ablation Study

This section presents ablation studies of our reasoning model and query inference framework. We evaluate the reasoning model components using factoid queries, while employing the more complex aggregate queries to analyze the query inference pipeline.

Ablation of Reasoning Components. Table 4 demonstrates the impact of individual components in our reasoning model, where "-op", "-ht", "-jl" represent our fuzzy logical reasoning model without the optimized logical operators (using the projection and conjunction in FuzzyQE [8]), hierarchical type enhancement (initializing the type embedding randomly) and the joint loss function (only using the loss of entity-query alignment), respectively. Without optimized logical operators, MRR and Hit@3 decrease by 2.6%/1.9% on DBpedia and 3.6%/4.3% on YAGO4, confirming the value of our symbolic-enhanced projection and attention-based conjunction. Removing hierarchical type enhanced-initialization for type embeddings causes noticeable performance degradation, as random initialization fails to capture semantic relationships between hierarchical types. Most significantly, eliminating the type constraint loss reduces MRR by 16% relative, with substantial drops across all metrics. These results collectively validate that both operator optimization and hierarchical type-aware learning are essential for robust reasoning on query.

Ablation of Query Inference. Table 5 further examines the query inference components, including removing the multi-hop projection score with early-score mechanism ("-es"), optimization of k ("-ok"), optimization of confidence interval ("-ci"). Eliminating multi-hop projection score with early-score mechanism maintains comparable MAPE but doubles execution time, confirming that early-score mechanism successfully eliminates invalid segmentations without compromising accuracy. Removing optimization of k increases MAPE by approximately 10% and adds 50ms latency, demonstrating that sample space reduction effectively controls both error and computation time. The precision degradation stems from incorporating excessive irrelevant samples, while the efficiency loss comes from computing projection score for these additional candidates. Disabling optimization of confidence interval causes minor regression in both precision (1% MAPE) and efficiency (10-20ms), as the Wilson interval resolves boundary violations and the zero-inflated model provides better variance estimation in small-correct answers scenarios. Our inference pipeline achieves an optimal balance between accuracy and efficiency through coordinated early scoring, candidate number optimization and statistical estimation.

5.4 Sensitivity Analysis

This section examines parameter sensitivity across factoid and aggregate queries, evaluating embedding dimension d and negative sample size M for factoid query performance, and sampling ratio ρ, similarity threshold τ, error tolerance e, and confidence level $1-\alpha$ for aggregate query.

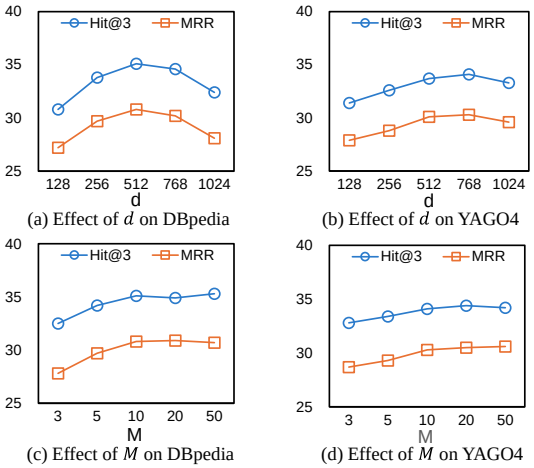
Reasoning Parameters. Figures 6(a)-(d) present the impact of embedding dimension and negative sampling number. Dimensions $d = \{128, 256, 512, 768, 1024\}$ show consistent MRR and Hit@3 improvements until peak performance at $d = 512$ and $d = 768$, followed by degradation at $d = 1024$—suggesting diminished returns from overfitting beyond optimal capacity. For negative samples $M = \{3, 5, 10, 20, 50\}$, performance gains plateau beyond $M = 10$, indicating saturation in discriminative information.

Query Inference Parameters. Figure 7(a) demonstrates the coverage-sensitivity trade-off for sampling ratio ρ: although lower values reduce candidate coverage, its coverage is always larger than 85%, excluding the queries with disjunction operators. [] Semantic threshold $\tau \in \{0.8, 0.85, 0.9, 0.95, 1.0\}$ (Fig. 7(b)) reveals optimal AJS at $\tau = 0.95$, where the projection score from reasoning complements exact subgraph matching for accommodating knowledge graph incompleteness. The performance drop at $\tau = 1.0$ confirms the necessity of flexible matching. Error tolerance $b \in \{0.05, 0.10, 0.15, 0.20\}$ (Figs. 7(c)-(d)) shows expected precision-efficiency trade-offs: higher tolerance accelerates convergence at precision costs. Confidence levels $1-\alpha = \{80\%, 85\%, 90\%, 95\%\}$ (Figs. 7(e)-(f)) validate statistical principles—tighter intervals require more samples, increasing latency while reducing estimation error.

6 RELATED WORK

6.1 Knowledge Graph Query

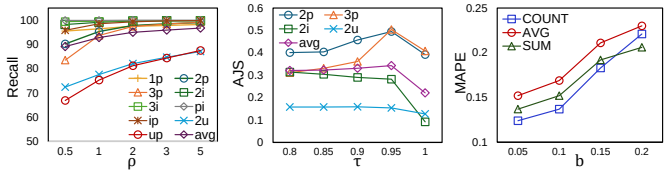
Factoid Queries. Depending on the various styles of knowledge graph (KG) querying, prevalent methods can be categorized into factoid queries and aggregate queries. For factoid queries, subgraph matching algorithms aim to identify isomorphic subgraphs within the complete KG as answers by employing a filtering-ordering-ranking pipeline. Early approaches such as LDF [35] and VF2 [10] utilized local node features for filtering, while more recent methods like VEQ [15] and CaLiG [43] have integrated neighborhood safety



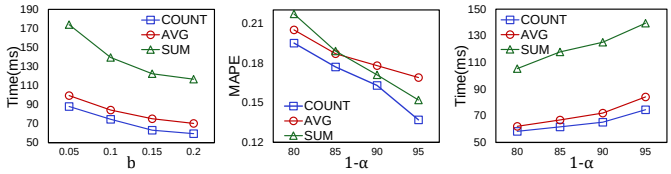
(a) Effect of d on DBpedia (b) Effect of d on YAGO4

(c) Effect of M on DBpedia (d) Effect of M on YAGO4

Figure 6: Parameter effect on reasoning model.



(a) Effect of ρ (b) Effect of τ (c) Effect of b on MAPE



(d) Effect of b on Time (e) Effect of $1-\alpha$ on MAPE (f) Effect of $1-\alpha$ on Time

Figure 7: Parameter effect on inference model.

considerations and bipartite graph matching techniques to enhance candidate set refinement. The optimization of matching order encompasses data-agnostic methods (e.g., RI [6]), data-dependent strategies (e.g., QuickSI [30]), and dynamic ordering approaches (e.g., DPiso [12]). Despite these advancements in optimization, subgraph matching remains fundamentally constrained by issues related to KG incompleteness and NP-complete complexity, which result in missing answers and scalability challenges. Unlike subgraph matching, which relies heavily on graph structures, knowledge graph (KG) embedding provides an alternative technique for KG querying by mapping entities and queries into low-dimensional spaces. Given that KG querying necessitates high precision, embeddings derived from translation-based models [7, 34, 39] and neural network models [16, 28, 29] are particularly suitable due to their superior accuracy in link prediction.

Aggregate Queries. To enhance the efficiency of aggregate queries, which inherently necessitate the processing of multiple factoid queries, recent research has adapted the online aggregation framework from relational databases to the domain of knowledge graphs. This framework employs a sampling-estimation pipeline that delivers results by statistically approximating over sampled subsets, thereby reducing computational load [13]. However, the unique data structures characteristic of knowledge graphs require substantial modifications to traditional online aggregation techniques. For example, EAQ [20] integrates embedding compression with R-tree indexing to expedite sampling processes, while AQS [38] leverages TransE embeddings and random walks to yield approximate results accompanied by confidence guarantees. Although AQS extends its support for complex queries through decomposition methods, it incurs significant computational overhead and remains fundamentally constrained by its reliance on the complete knowledge graph assumption.

6.2 Knowledge Graph Reasoning

KG reasoning aims to infer missing facts or answer complex queries over incomplete KGs, which has been extended to multi-modal tasks or temporal KGs. Existing methods for handling complex logical and aggregate queries can be broadly categorized into the following paradigms based on their core reasoning mechanisms.

Logical Reasoning. This line of work, which is most relevant to our own, directly models complex queries by composing logical operators (e.g., projection, conjunction) within embedding spaces. GQE [16] pioneered this direction by embedding queries as points and defining geometric operators. Query2Box [28], HypE [9] and ConE [51] extended this using geometric shapes to naturally support logical operators. BetaE [29] introduced probabilistic reasoning via Beta distributions to model uncertainty and support full first-order logic, including negation. FuzzQE [8] further incorporated fuzzy logic to ensure operator differentiability while adhering to logical axioms, as well as the research [5, 27, 54]. However, when applied to incomplete KGs, these models face critical limitations: their neural operators are heuristic and lack explicit mechanisms to constrain error propagation across multiple hops, leading to unreliable results. Furthermore, they are primarily designed for entity retrieval and offer no native support or reliability guarantees for aggregate queries (e.g., COUNT, AVG), which require statistical estimation over answer sets.

Path-based Reasoning. Another paradigm addresses multi-hop queries by searching for or learning from relational paths. Early methods like PRA [17] used random walks in discrete spaces, while later approaches like NeuralLP integrated logical rule learning. LQAC [33] achieves multi-hop logical reasoning by representing concepts and queries as fuzzy sets and implementing set operations such as t-norm and t-conorm in fuzzy logic, thereby providing answers at both the instance level and the concept level when responding to queries. Recently, reinforcement learning has been widely adopted by training agents to find reasoning paths, offering interpretable traces [40, 50, 52]. Multi-agent [32, 46] and hierarchical reinforcement learning [36, 42] variants have been proposed to improve search efficiency. Nevertheless, these methods fundamentally treat reasoning as a search problem over a combinatorial action space, which creates a severe tension between path exploration efficiency and answer recall. More importantly, their objective is to locate a single target entity, making them intrinsically ill-suited for aggregate queries that require summarizing over an entire answer set.

In summary, while existing paradigms have advanced KG reasoning, a significant gap remains: a unified framework that can execute

logical reasoning with controlled error propagation, achieve computational efficiency without sacrificing robustness, and natively support statistically reliable aggregate query answering over incomplete KGs. Bridging this gap is the central objective of our work.

7 CONCLUSION

In this study, we propose FRECI, a novel KG query model that incorporates a reasoning-inference framework. This model is designed to efficiently return precise results for both factoid and aggregate queries. Initially, it optimizes logical operators to enhance the effectiveness of projection and conjunction operators over queries while integrating hierarchical type information into reasoning models. Subsequently, FRECI offers an efficiently calibrated inference process to measure the correctness of candidates with an early-scoring mechanism for factoid queries and sampling in aggregate querying algorithm, then optimizes the confidence interval for aggregate queries. Finally, experiments conducted on two real-world KGs demonstrate that FRECI surpasses several baseline models in both reasoning and inference phases, achieving the highest accuracy in query results for both factoid and aggregate queries. Furthermore, its query efficiency has been confirmed to be satisfactory.

REFERENCES

[1] Mahmoodirad A., Garg H., and Niroomand S. 2026. *Recent Advances on Fuzzy Sets - Theory and Applications*. Vol. 440. Springer.

[2] J. Arai, Y. Fujiwara, and M. Onizuka. 2023. GuP: Fast Subgraph Matching by Guard-based Pruning. In *In Proc. SIGMOD Conf.*, Vol. 1. 167:1–167:26.

[3] F. Atif, O. E. Khatib, and D. Difallah. 2023. BeamQA: Multi-hop Knowledge Graph Question Answering with Sequence-to-Sequence Prediction and Beam Search. In *In Proc. SIGIR Conf.* 781–790.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *In Proc. ISWC Conf.* 722–735.

[5] Y. Bai, X. Lv, J. Li, and L. Hou. 2022. Answering complex logical queries on knowledge graphs via query tree optimization. *arXiv preprint*, Article arXiv:2212.09567 (2022).

[6] V. Bonnici, R. Giugno, A. Pulvirenti, D. E. Shasha, and A. Ferro. 2013. A subgraph isomorphism algorithm and its application to biochemical data. *BMC Bioinform.* 14, S-7 (2013), S13.

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

[8] X Chen, Z Hu, and Y. Sun. 2022. Fuzzy logic based logical query answering on knowledge graphs. In *In Proc. AAAI Conf.* 3939–3948.

[9] N. Choudhary, N. Rao, S. Katariya, K. Subbian, and C. K. Reddy. 2021. Self-Supervised Hyperboloid Representations from Logical Queries over Knowledge Graphs. In *In Proc. WWW Conf.* 1373–1384.

[10] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. 2004. A (Sub) Graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 10 (2004), 1367–1372.

[11] Z. Du, C. Zhou, J. Yao, T. Tu, L. Cheng, H. Yang, J. Zhou, and J. Tang. 2023. CogKR: Cognitive Graph for Multi-Hop Knowledge Reasoning. *IEEE Trans. Knowl. Data Eng.* 35, 2 (2023), 1283–1295.

[12] M. Han, H. Kim, G. Gu, K. Park, and W. Han. 2019. Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together. In *In Proc. SIGMOD Conf.* 1429–1446.

[13] J. M. Hellerstein, P. J. Haas, and H. J. Wang. 1997. Online Aggregation. In *In Proc. SIGMOD Conf.* 171–182.

[14] O. E. Khatib. 2025. Reasoning over Incomplete Knowledge Graphs. In *In Proc. CIKM Conf.* 6785–6788.

[15] H. Kim, Y. Choi, K. Park, X. Lin, S. Hong, and W. Han. 2021. Versatile Equivalences: Speeding up Subgraph Query Processing and Subgraph Matching. In *In Proc. SIGMOD Conf.* 925–937.

[16] Hamilton. W. L., P. Bajaj, M. Zitnik, D. Jurafsky, and J. Leskovec. 2018. Embedding Logical Queries on Knowledge Graphs. In *In Proc. NIPS Conf.* 2030–2041.

[17] N. Lao, T. Mitchell, and W. W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *In Proc. EMNLP Conf.* 529–539.

[18] F. Li, B. Wu, K. Yi, and Z. Zhao. 2016. Wander join: Online aggregation via random walks. In *In Proc. ICDE Conf.* 615–629.

[19] L. Li, H. Wang, J. Li, X. Xu, Y. Wang, and T. Ren. 2024. Integrating Structure and Text for Enhancing Hyper-relational Knowledge Graph Representation via Structure Soft Prompt Tuning. In *In Proc. CIKM Conf.* 1226–1234.

[20] Y. Li, T. Ge, and C. X. Chen. 2020. Online Indices for Predictive Top-k Entity and Aggregate Queries on Knowledge Graphs. In *In Proc. ICDE Conf.* 1057–1068.

[21] Y. Li, L. Zhang, H. Yan, T. Zhao, Z. Ma, M. Huang, and J. Liu. 2025. SAGE: Scale-Aware Gradual Evolution for Continual Knowledge Graph Embedding. In *In Proc. SIGKDD Conf.* 1600–1611.

[22] F. Lin, X. Zhu, Z. Zhao, D. Huang, Y. Yu, X. Li, Z. Zheng, T. Xu, and E. Chen. 2025. Knowledge Graph Pruning for Recommendation. *ACM Trans. Inf. Syst.* 44, 1 (2025), 20.

[23] L. Liu, Y. Chen, M. Das, H. Yang, and H. Tong. 2023. Knowledge Graph Question Answering with Ambiguous Query. In *In Proc. WWW Conf.* 2477–2486.

[24] L. Liu, B. Du, J. Xu, Y. Xia, and H. Tong. 2022. Joint Knowledge Graph Completion and Question Answering. In *In Proc. SIGKDD Conf.* 1098–1108.

[25] P. Lo and E. Lim. 2024. Non-monotonic Generation of Knowledge Paths for Context Understanding. *ACM Trans. Manag. Inform. Syst.* 15, 1 (2024), 1.

[26] J. Luo, Y. Pan, and G. Huang. 2025. Collaborative Interest Mining Network for Knowledge Graph-based Recommendation. In *In Proc. CIKM Conf.* 2000–2010.

[27] F. Luus, P. Sen, P. Kapanipathi, R. Riegel, N. Makondo, T. Lebese, and A. Gray. 2021. Logic Embeddings for Complex Query Answering. *arXiv preprint*, Article arXiv:2103.00418 (2021).

[28] H. Ren, W. Hu, and J. Leskovec. 2020. Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. In *In Proc. ICLR Conf.* 1–17.

[29] H. Ren and J. Leskovec. 2020. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. In *In Proc. NIPS Conf.* 19716–19726.

[30] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu. 2017. The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing. *Proc. VLDB Endow.* 11, 4 (2017), 420–431.

[31] F. M. Suchanek and G. G. Kasneci. 2007. Yago: a core of semantic knowledge. In *In Proc. WWW Conf.* 697–706.

[32] Fu C.and Chen T., Qu M., Jin W., and X. Ren. 2019. Collaborative Policy Learning for Open Knowledge Graph Reasoning. In *In Proc. EMNLP Conf.* 2672–2681.

[33] Z. Tang, S. Pei, X. Peng, F. Zhuang, X. Zhang, and R Hoehndorf. 2023. Neural Multi-hop Logical Query Answering with Concept-Level Answers. In *In Proc. ISWC Conf.* 522–540.

[34] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *In Proc. ICML Conf.*, Vol. 48. 2071–2080.

[35] J. R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *J. ACM* 23, 1 (1976), 31–42.

[36] G. Wan, S. Pan, C. Gong, C. Zhou, and G. Haffari. 2020. Reasoning Like Human: Hierarchical Reinforcement Learning for Knowledge Graph Reasoning. In *In Proc. IJCAI Conf.* 1926–1932.

[37] Y. Wang, A. Khan, T. Wu, J. Jin, and H. Yan. 2020. Semantic Guided and Response Times Bounded Top-k Similarity Search over Knowledge Graphs. In *In Proc. ICDE Conf.* 445–456.

[38] Y. Wang, A. Khan, X. Xu, J. Jin, Q. Hong, and Y. Fu. 2022. Aggregate Queries on Knowledge Graphs: Fast Approximation with Semantic-aware Sampling. In *In Proc. ICDE Conf.* 2914–2927.

[39] Z. Wang, J. Zhang, J. Feng, and Z. Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *In Proc. AAAI Conf.* 1112–1119.

[40] W. Xiong, T. Hoang, and W. Y. Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *In Proc. EMNLP Conf.* 564–573.

[41] Z. Xu, Z. Wen, P. Ye, H. Chen, and H. Chen. 2022. Neural-Symbolic Entangled Framework for Complex Query Answering. In *In Proc. NIPS Conf.* 522–540.

[42] Xia Y., Luo J., Lan M., Zhou G., Li Z., and S. Liu. 2023. Reason more like human: Incorporating meta information into hierarchical reinforcement learning for knowledge graph reasoning. *Appl. Intell.* 53, 11 (2023), 13293–13308.

[43] R. Yang, Z. Zhang, W. Zheng, and J. X. Yu. 2023. Fast Continuous Subgraph Matching over Streaming Graphs via Backtracking Reduction. In *In Proc. SIGMOD Conf.*, Vol. 1. 15:1–15:6.

[44] S. Ye, X. Xu, Y. Wang, and Y. Fu. 2023. Efficient Complex Aggregate Queries with Accuracy Guarantee Based on Execution Cost Model over Knowledge Graphs. *Mathematics* 11, 18, Article 3908 (2023).

[45] H. Yıldırım, V. Chaoji, and M.J. Zaki. 2012. GRAIL: a scalable index for reachability queries in very large graphs. *VLDB J.* 21, Article arXiv:2205.10128 (2012), 509–534 pages.

[46] Li Z., Jin X., Guan S., Wang Y., and Cheng X. 2018. Path Reasoning over Knowledge Graph: A Multi-agent and Reinforcement Learning Based Method. In *In Proc. ICDM Workshops Conf.* 929–936.

[47] L. A. Zadeh. 1965. Fuzzy Logic. *Computer* 21, 4 (1965), 83–93.

[48] S. Zeighami, R. Seshadri, and C. Shahabi. 2024. A Neural Database for Answering Aggregate Queries on Incomplete Relational Data. *IEEE Trans. Knowl. Data Eng.* 36, 7 (2024), 2790–2802.

[49] H. Zhang, X. Shen, B. Yi, J. Liu, and Y. Xie. 2025. A Plug-in Critiquing Approach for Knowledge Graph Recommendation Systems via Representative Sampling. In *In Proc. WWW Conf.* 322–333.

[50] Y. Zhang, H. Wang, W. Shen, and G. Peng. 2025. DuAK: Reinforcement Learning-Based Knowledge Graph Reasoning for Steel Surface Defect Detection. *IEEE Trans Autom. Sci. Eng.* 22 (2025), 557–569.

[51] Z. Zhang, J. Wang, J. Chen, S. Ji, and F. Wu. 2021. ConE: Cone Embeddings for Multi-Hop Reasoning over Knowledge Graphs. In *In Proc. NIPS Conf.*, Vol. 34. 19172–19183.

[52] S. Zheng, W. Chen, P. Zhao, A. Liu, J. Fang, and L. Zhao. 2021. When Hardness Makes a Difference: Multi-Hop Knowledge Graph Reasoning over Few-Shot Relations. In *In Proc. CIKM Conf.* 2688–2697.

[53] Ganggao Zhu and Carlos A Iglesias. 2017. Computing semantic similarity of concepts in knowledge graphs. *IEEE Trans. Knowl. Data Eng.* 29, 1 (2017), 72–89.

[54] Z. Zhu, W. X. Zhao, S. Bian, Y. Zhou, J. R. Wen, and J. Yu. 2021. Neural-symbolic models for logical queries on knowledge graphs. *arXiv preprint*, Article arXiv:2205.10128 (2021).