Extending First-Order Logic for Factual Reasoning over Knowledge Graphs

Yuanzhen Hao
University of Chinese Academy of Sciences
Beijing, China
haoyuanzhen22@mails.ucas.ac.cn

Desheng Wu*
University of Chinese Academy of Sciences
Beijing, China
dwu@ucas.ac.cn

Abstract

First-order logic (FOL) is a fundamental formalism for factual reasoning over knowledge graphs (KGs), e.g. in researches of KG-based fact verification and logical consistency or reasoning of large language models (LLM). However, existing benchmarks and approaches insufficiently capture many claims that require comparison or counting, and lack support for several FOL quantifiers and connectives. To address these challenges and expand the expressive capacity of FOL for KG-based reasoning, we introduce FOLX-KG, a novel extended FOL σ-structure over KGs that incorporates comparison predicates and counting quantifiers. Using this extended logic, we construct Fact-FOLX-KG, a fact verification dataset consisting of 43,821 KG-based claim–formula pairs designed to enable systematic study of richer logical forms and reasoning types. We further propose FOLX Prover, an executable program-guided logic reasoning pipeline adapted for KG-based factual reasoning under the extended FOL. Experimental results show that our method achieves state-of-the-art performance on Fact-FOLX-KG, while previous methods experience performance drop on claims requiring comparison and counting. These findings demonstrate the importance of extended logical expressiveness for robust factual reasoning over KGs.[1]

1 Introduction

Factual reasoning over knowledge graphs (KGs) has emerged as an important research direction in evidence-based fact checking (Kim et al., 2023b) and in evaluating the logical consistency of large language models (LLMs) (Ghosh et al., 2025). Because KGs provide structured and symbolic representations of world knowledge, they offer a natural foundation for verifiable inference, particularly when integrated with first-order logic (FOL), whose formal semantics allow explicit and interpretable reasoning. Recent studies have demonstrated the effectiveness of FOL-based approaches for factual reasoning (Han et al., 2024; Pei et al., 2025; Wang and Shu, 2023), and KG-based fact verification methods increasingly incorporate FOL to improve reasoning accuracy (Kim et al., 2023b; Hao and Wu, 2025).

Despite recent progress, existing work on factual reasoning over KGs faces two key limitations. First, current datasets (Kim et al., 2023b; Ghosh et al., 2025) and methods (Kim et al., 2023a; Hao and Wu, 2025) typically support only a restricted subset of FOL quantifiers, connectives, and predicates. Second, standard FOL formalisms lack essential expressive capabilities required for real-world reasoning, such as counting and arithmetic operations. As a result, many common claim types—especially those involving entity comparison or cardinality-based reasoning—cannot be faithfully represented or verified using the existing FOL σ-structure over KG.

To address these challenges, we introduce **FOLX-KG**, a novel extended FOL σ-structure designed for KG-based logical reasoning. FOLX-KG extends traditional FOL through predicate and quantifier extensions, enabling direct expression of entity comparison and counting capabilities while preserving the semantics of standard FOL. The counting quantifier $\exists^{\geq k}x$ of FOLX-KG means there are at least k distinct variables x, which is a more general quantifier than Existential and Universal quantifier. This extended structure substantially enhances the representational capacity of FOL-over-KG reasoning and enables more claim types.

We further propose a unified KG-FOL-NL construction framework for building KG-based fact verification datasets. The framework proceeds by first designing FOL formulae corresponding to spe-

*Corresponding author.
[1] https://github.com/hydragon/FOLX

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 26722–26738
July 2-7, 2026

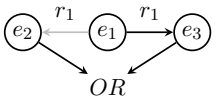
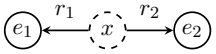
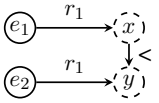
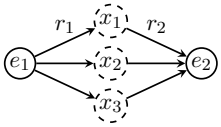
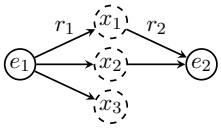
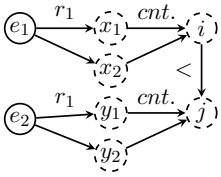
Type	Claim	Logic	Graph
Conj.	The Acura TLX (e_1) has (r_1) a Honda K engine (e_2) manufactured by (r_2) Honda (e_3).	$r_1(e_1, e_2) \wedge r_2(e_2, e_3)$	
Disj.	The A.E Dimitra Efxeinoupolis club (e_1) is located in (r_1) Pakistan (e_2) or Greece (e_3).	$r_1(e_1, e_2) \vee r_1(e_1, e_3)$	
Quant $\exists$	A city (x) in (r_1) Belgium (e_1) is served by (r_2) the Antwerp international airport (e_2).	$\exists x\,(r_1(x, e_1) \wedge r_2(x, e_2))$	
Cmp. e_i	The Daihatsu Tanto (e_1) (x) is narrower (r_1) than ($<$) the Hummer H1 (e_2) (y).	$\exists x \exists y\,(r_1(e_1, x) \wedge r_1(e_2, y) \wedge x < y)$	
Quant $\forall$	All the alma maters (x) of (r_1) Toshiki Kaifu (e_1) are located in (r_2) Tokyo (e_2).	$\forall x\,(r_1(e_1, x) \rightarrow r_2(x, e_2))$	
CntQ $\exists^{=}$	Jonatha Brooke (e_1) has exactly 2 written (r_1) works (x) released (r_2) under Capitol Records (e_2).	$\exists^{=2} x\,(r_1(e_1, x) \wedge r_2(x, e_2))$	
Cnt Cmp.	The manufacturer (r_1, x, y) count of Alekseyev I-21 (e_1) falls below ($<$) that of Volkswagen Tiguan (e_1).	$\exists i \exists j (\exists^{=i} x\, r_1(e_1, x) \wedge \exists^{=j} y\, r_1(e_2, y) \wedge i < j)$	

Table 1: Different reasoning types of FOLX-KG including two quantifier-free logic forms (Conjunction $\wedge$ and Disjunction $\vee$) and five quantifier-bounded logic forms (Existential $\exists$, Entity (e_i) Comparison, Universal $\forall$, Counting quantifiers $\exists^{=}$, and Counting Comparison (Cnt Cmp.)).

cific reasoning types, then extracts KG subgraphs that satisfy these formulae, and finally generates natural-language (NL) claims using a combination of template-based generation and LLM-based rewriting. This process gives fine-grained control over the logical structure of each instance and produces datasets with transparent and diverse reasoning patterns. Using this framework, we construct **Fact-FOLX-KG**, a large-scale dataset containing 43,821 claims paired with their FOLX-KG formulae, with all verification evidence derived from DBpedia (Lehmann et al., 2015). For different reasoning types, Table 1 illustrates some examples of the corresponding logical formulae and the KG subgraph structures used to generate the claims.

Finally, we present **FOLX Prover**, a pipeline for FOL-based reasoning over KGs. Given a claim, a few-shot prompted LLM generates an executable program implementing the logical semantics of the FOLX-KG formula. The program interacts directly with the KG to perform symbolic inference and produce a prediction. Experiments on Fact-FOLX-KG show that our method outperforms other KG-based fact verification baselines. Prior methods exhibit substantial performance degradation on the extended reasoning types introduced in FOLX-KG, whereas our pipeline achieves high accuracy. These findings show the importance of extended FOL expressiveness and executable logical reasoning over FOLX-KG for robust KG-based fact verification. More broadly, extending FOL is also crucial for studying logical reasoning in natural language and improving logical reasoning in LLMs.

2 Related Work

2.1 Fact Verification over KG

Fact verification over knowledge graphs (KGs) refers to the task of verifying whether a natural-language claim is supported by the KG evidence. FactKG (Kim et al., 2023b), the recent benchmark for this task, categorizes claims into five reasoning categories based on the logics and structure of the KG subgraphs: *one-hop*, *existence*, *conjunctive*, *multi-hop*, and *negation*. FactKG adopts a graph-evidence reasoning pipeline inspired by the GEAR (Zhou et al., 2019), which aggregates KG triples to support claim verification.

Motivated by the LLMs, recent work has increasingly explored structured reasoning over KGs using LLMs to improve verification accuracy (Kim et al., 2023a; Pham et al., 2025; Hao and Wu, 2025).

2.2 Factual Reasoning with FOL

Existing multi-step fact-checking approaches introduced partial FOL for logical reasoning (Pan et al., 2023; Wang and Shu, 2023), while KG-based verification methods have leveraged quantifier-free FOL to enhance verification in graph reasoning (Hao and Wu, 2025).

There have been many researches of integrating FOL in reasoning (Xu et al., 2025; Tian et al., 2021; Liu et al., 2020; Tafjord et al., 2021; Saparov and He, 2023). The introduction of the FOLIO dataset further proposed research on complex FOL for natural-language reasoning, highlighting the need for benchmarks that capture richer logical phenomena (Han et al., 2024). Some approaches combine LLM with FOL to guide reasoning (Olausson et al., 2023; Ye et al., 2023). There has been various methods to enhance logical reasoning of LLMs (Wei et al., 2022; Kojima et al., 2022; Chen et al., 2024; Zhou et al., 2023). FoVer uses LLMs to generate executable programs that implement FOL reasoning with Chain-of-Thought strategies (Wei et al., 2022). By translating logical forms into programs, FoVer enables proving of complex FOL formulae (Pei et al., 2025). Factual reasoning with FOL can also be used for evaluating logical consistency of LLMs (Ghosh et al., 2025; Liu et al., 2025).

2.3 FOL with extension

Standard FOL over KG typically defines entities as elements of the universe and KG relations as predicate symbols. This σ-structure is sufficient for the reasoning types represented in FactKG (Kim et al., 2023b). However, natural-language claims frequently involve richer semantic phenomena, such as comparison, counting, and arithmetic reasoning, that cannot be adequately expressed using basic FOL.

Within the mathematical definition of FOL, predicate symbols is a set that can be enriched by introducing new types of predicates, which enables to add numerical comparison predicates not inherently presented in KG relations but essential for modeling reasoning. The expressive limitations of the existential and universal quantifiers have motivated the development of generalized quantifiers (LINDSTRÖM, 1966), which give FOL the ability to express cardinality constraints. Subsequent researches have provided more formalized definitions of counting quantifiers (Otto, 1997; Libkin, 2004). Logical reasoning mirrors human-level cognitive processes closely (Xu et al., 2025; Cummins et al., 1991), and we give comparison between FOL and query languages in Appendix C.

In our work, we adopt a second-sorted universe (Libkin, 2004) to formally define an extended FOL σ-structure capable of representing counting quantifiers.

3 FOLX-KG

This section introduces the mathematical definitions of FOLX-KG with predicative and quantifier extension. **Def. 1** and **Def. 2** define the original FOL over KG; **Def. 3** introduces extended predicates; **Def. 4** and **Def. 5** define the counting-quantifier ($\exists^{\geq k}$) extension. All the examples of reasoning types expressed by FOLX-KG are illustrated in Table 1.

The propositions provide theoretical proof: **Proposition 1** derives some different counting quantifiers; **Proposition 2** derives the comparison symbols for the counting results; **Proposition 3** proves De Morgan's Laws of counting comparison formulae.

3.1 Knowledge Graph

A Knowledge Graph (KG) is a structured representation of entities and their semantic relations.

Formally, a knowledge graph is defined as:

$$\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}),$$

where $\mathcal{E}$ is the set of entities; $\mathcal{R}$ denotes the set of relations; $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of triples, each

of the form (h, r, t), meaning that h (head entity) is connected to t (tail entity) via relation r.

3.2 First-Order Logic over KG

We define **First-Order Logic over KG (FOL-KG)** following the standard definitions in mathematical logic. The σ-structure of FOL-KG normally uses entities as constant symbols and relations as predicate symbols.

Definition 1. *The FOL σ-structure over the KG can be defined as :*

$$\mathfrak{A} = \langle A, \{e_i^{\mathfrak{A}}\}, \{r_i^{\mathfrak{A}}\}\rangle$$

The σ-structure $\mathfrak{A}$ has an interpretation: each constant symbol $e_i^{\mathfrak{A}} \in \mathcal{E}$; each 2-ary relation symbol $r_i^{\mathfrak{A}} \in \mathcal{R}$ is a predicate symbol and $r_i(e_j, e_k) = True \equiv (e_j, r_i, e_k) \in \mathcal{T}$. The universe of $\mathfrak{A}$ over KG is $A = \mathcal{E}$.

Since $\mathcal{E}$ is finite, the universe of $\mathfrak{A}$ is also finite.

The vocabulary σ of FOL-KG is at most countable and relational. We define the free and bounded variables, and formula semantics of FOL-KG.

Definition 2. *We assume a countably infinite set of variables. The terms and formulae of the FOL-KG can be defined as follows:*

Terms t include: variable (denoted by $x, y, z, ...$), constant symbol (e_i).

Formulae φ include: predicate formulae ($r(x, e), r(e, x), r(x, y)$), and any formulae with quantifiers and connectives.

With FOL-KG, we can design reasoning types of Conjunction, Disjunction, Existential, and Universal as shown in Table 1.

3.3 Extending FOL-KG

Here we define the σ-structure of **FOL with extension over KG (FOLX-KG)**. It can be defined in two extension steps: 1. predicative extension; 2. quantifier extension.

First, we define a σ-structure of FOL-KG with extension of predicative symbols (FOL$'$-KG).

Definition 3. *The σ-structure of FOL$'$-KG is defined as:*

$$\mathfrak{A}' = \langle \{S_l^{\mathfrak{A}}\}, \{P_l^{\mathfrak{A}'}\}\rangle$$

$\{S_l^{\mathfrak{A}}\}$ is all the symbols of FOL-KG. $\{P_l^{\mathfrak{A}'}\}$ is a set of extended predicative symbols that for each $P_l \notin \mathcal{R}$.

For example, we define a set of $\{=, <, >, \leq, \geq\}$ as general comparison symbols to extend the entity comparison ability of FOL-KG. As shown in Tabel 1, these extended predicative symbols can be used for Entity Comparison reasoning.

Then, we extend the quantifier of FOL$'$-KG with counting quantifiers and define the FOLX-KG σ-structure $\mathfrak{A}^*$.

Definition 4. *The σ-structure of FOLX-KG is $\mathfrak{A}^*$:*

$$\langle \{S_l^{\mathfrak{A}'}\}, \{0, ..., n-1\}, +, \times, \underline{\min}, \underline{\max}\rangle$$

$\{S_l^{\mathfrak{A}'}\}$ is all the symbols of FOL$'$-KG. $\{0, ..., n-1\}$ is the counting universe. $\underline{\min}, \underline{\max}$ denotes the counting minimum and maximum in $\mathfrak{A}^*$. The symbols $+$ and $\times$ are relations used for arithmetic operations $+(t_1, t_2, t_3)$ and $\times(t_1, t_2, t_3)$, which can be normally written as $t_1 + t_2 = t_3$ and $t_1 \times t_2 = t_3$.

The universe $\{e_0, ..., e_{n-1}\}$ of FOL$'$-KG is the first-sorted universe. The extended Counting universe is the second-sorted universe (Libkin, 2004).

To express numerical counting on the number of elements satisfying a given formula, FOL can be extended with **counting quantifiers**. The counting quantifier of FOLX-KG can be written as the form:

$$\exists^{\geq k} x\, \varphi(x)$$

which means there are at least k distinct variables x that satisfies $\varphi(x)$ (Otto, 1997). We can prove all the counting symbols $\{\exists^{=}, \exists^{<}, \exists^{>}, \exists^{\leq}\}$ with this definition.

Proposition 1. *The different counting quantifiers can be derived via the following proof:*

$$\exists^{=k} x\, \varphi(x) = \exists^{\geq k} x\, \varphi(x) \wedge \neg \exists^{\geq k+1} x\, \varphi(x)$$

With the $\exists^{=k} x\, \varphi(x)$, we can further get:

$$\exists^{>k} x\, \varphi(x) = \exists^{\geq k} x\, \varphi(x) \wedge \neg \exists^{=k} x\, \varphi(x)$$

$$\exists^{<k} x\, \varphi(x) = \neg \exists^{\geq k} x\, \varphi(x)$$

$$\exists^{\leq k} x\, \varphi(x) = \neg \exists^{>k} x\, \varphi(x)$$

Definition 5. *We assume a countably infinite set of variables. The terms and formulae of the FOLX-KG can be defined as follows:*

Terms include: variable (denoted by $i, j, k, ...$), constant symbol ($0, ..., n-1$).

Formulae include: $\exists^{\geq i} x\, \varphi(x), i \geq j$, and other formulae defined in FOL-KG.

With all the counting quantifiers, we can describe "equal to", "more than", "less than", and "at most" in FOL expressions.

Additionally, the comparison expression of counting results is also defined in the FOLX-KG ($\#x\,\varphi_1(x) \geq \#y\,\varphi_2(y)$, $\#x$ means counting x).

Proposition 2. *The comparison expression is derived from FOLX-KG as:*

$$i \geq j \equiv \exists k\,(j + k = i)$$

Then, we can compare the counting results between two formulae as:

$$\exists i \exists j (\exists^{=i} x \varphi_1(x) \wedge \exists^{=j} y \varphi_2(y) \wedge (i \geq j))$$

For other comparison operations, we can prove it in the similar ways.

The comparison of counting results is written in a FOL formula with two sorted universes. Now, we prove a negation consistency between the FOL formula and the common sense. If we write the counting comparison in the form $\#x\,\varphi_1(x) \geq \#y\,\varphi_2(y)$, then the negation of it is $\neg(\#x\,\varphi_1(x) \geq \#y\,\varphi_2(y)) = \#x\,\varphi_1(x) < \#y\,\varphi_2(y)$.

But in FOL formula based on De Morgan's Laws:

$$\neg(\exists i \exists j (\exists^{=i} x \varphi_1(x) \wedge \exists^{=j} y \varphi_2(y) \wedge (i \geq j)))$$

is equal to:

$$\forall i \forall j \neg(\exists^{=i} x \varphi_1(x) \wedge \exists^{=j} y \varphi_2(y) \wedge (i \geq j))$$

Here, we need to prove that:

$$\forall i \forall j (\exists^{\neq i} x \varphi_1(x) \vee \exists^{\neq j} y \varphi_2(y) \vee (i < j))$$

is equal to:

$$\exists i \exists j (\exists^{=i} x \varphi_1(x) \wedge \exists^{=j} y \varphi_2(y) \wedge (i < j))$$

The complete proof is given in Appendix E.

Proposition 3. *In FOLX-KG, the negation of counting comparison formulae follows De Morgan's Laws, and is consistent with the negation behavior of numerical comparison operators.*

4 Dataset Construction

We propose KG-FOL-NL, a framework for constructing a KG-based fact verification dataset by extracting structured information from KG using FOL and generating natural-language claims via template-based and LLM-augmented methods. Our goal is to produce linguistically diverse claims paired with formally grounded logical representations that span a wide range of reasoning types. As shown in Table 2, our dataset Fact-FOLX-KG contains more logical operators and symbols than the previous benchmark FactKG. Some examples of the reasoning types with logical formulae, claims and graph structures are shown in Tabel 1. We categorize FOL formulae into five reasoning types: Quantifier Free, FOL Quantifier, Entity Comparison, Counting Quantifier, and Counting Comparison. The construction framework is illustrated in Figure 1. For each category, we design FOL formulae that reflect KG-consistent reasoning patterns and write corresponding natural-language templates expressing these formulae. Finally, to increase linguistic diversity, we apply LLMs for stylistic rewriting and construct presupposition utterance for claims.

Dataset	¬	∧	∨	∃	∀	CMP.	CNT.
FactKG	✓	✓	×	✓	×	×	×
Ours	✓	✓	✓	✓	✓	✓	✓

Table 2: Comparison of the coverage of quantifiers, connectives, and predicate symbols in FOL across datasets (CMP. is comparison predicates, CNT. is counting quantifiers).

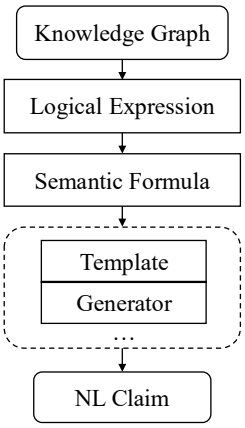


Figure 1: The framework of Fact-FOLX-KG dataset construction.

4.1 FOL and Claim Construction

4.1.1 Quantifier Free

The Quantifier Free category includes formulae without quantifiers; all predicate arguments are constants, and formulae rely solely on Boolean connectives (conjunction, disjunction, and negation). We begin by constructing conjunctive formulae and their claims using WebNLG data (Gardent

et al., 2017), where linguists provide high-quality textual statements of KG triples; a similar strategy is employed in FactKG. In contrast, disjunctive reasoning is not present in either WebNLG or FactKG, so we generate it by the template construction. Specifically, we first extract one-hop claims from WebNLG and construct disjunctions using two strategies: (1) concatenating two sentences with connective patterns such as "or" or "Or"; and (2) identifying tail entities sharing the same relation and replacing the original single tail with a coordinated structure such as "e1, e2 or e3".

Negation is introduced following the insertion-based strategy used in FactKG: we place explicit negation markers (e.g. "not", "no", ...) at syntactically appropriate positions and update the corresponding logical operators in the FOL formula.

For REFUTED samples, we adopt an entity-replacement strategy. We substitute the correct entity in the original sentence with another entity not connected in the corresponding KG triple, ensuring that the paired FOL formula evaluates to false. This procedure yields well-structured negative claims in KG semantics.

4.1.2 FOL Quantifier

For FOL expressions, the primary quantifiers are the existential and universal quantifiers. In the FactKG dataset, existential quantifiers appear in the *Existence* and *Multi-hop* reasoning types. Following a similar construction strategy, we generate existential-quantifier claims by replacing certain entity positions in WebNLG triples with relation-driven abstract references.

To incorporate universal quantification, we identify KG subgraphs in which a head entity is connected through a given relation to *all* of its tail entities. We then construct FOL formulae that explicitly encode this universal constraint. For each subgraph type, we design natural-language templates that convey the intended logical semantics. These templates primarily target constructions expressible through patterns such as "... *of* ... *all* ...". The resulting claims and formulae jointly capture the logical behavior of universally quantified reasoning grounded in KG structures.

4.1.3 Entity Comparison

The Entity Comparison category introduces a predicative extension of FOL designed to support comparison operations between numerical entities. We first identify relation types in the KG whose tail entities are numerical (e.g., dimensions, dates, lengths, ...). For each relation, we collect KG subgraphs in which multiple entities share comparable numerical values under the same relation. These values are then used in comparison predicates such as $\{<, >\}$ within the extended logical formula.

Based on these subgraphs, we construct template expressions that explicitly encode numerical comparison between entities, capturing reasoning patterns such as "*entity* e_1 *has a larger value than entity* e_2 *under relation* r". Corresponding natural-language templates are designed to align with each logical structure.

4.1.4 Counting Quantifier

The Counting Quantifier category extends traditional FOL by enriching quantifier expressiveness with explicit counting capabilities, thereby enabling more generalized quantification. To construct data for this reasoning category, we first identify KG subgraphs in which a head entity is connected to a set of tail entities through the same relation. The cardinality of this set determines the numerical value used in the counting operation. We then incorporate counting quantifiers into the logical formulae to precisely describe these subgraph properties, capturing statements of the form "*there exist exactly* n *entities satisfying predicate* P" or other variants.

4.1.5 Counting Comparison

The Counting Comparison category builds on the counting-quantifier extension by enabling explicit comparison between two counting variables, to enhance the numerical reasoning capacity of extended FOL. Unlike the Counting Quantifier type, which focuses on expressing cardinality, this category incorporates comparison operators $\{\geq, =, >, <, \leq\}$. Following the same subgraph selection strategy used for counting quantifiers, we extract subgraphs in which two triple groups associated with two head entities yield counting results. These paired counts serve as resources for constructing counting comparison FOL expressions. Natural-language claims adopt patterns such as "... *of* ... *is more than* ... *of* ...", where the comparative phrasing (e.g., "more than", "less than", "no fewer than") conveys the logical relation without mentioning explicit counts.

4.2 Linguistic Diversification

The diverse logical formula types, human-written claims and over 1000 claim templates ensure the

Category	True	False	Total
Quantifier Free	2873	2756	5629
FOL Quantifier	7338	6296	13634
Entity Comparison	2861	2873	5734
Counting Quantifier	4947	4949	9896
Counting Comparison	4465	4463	8928
Overall	**22484**	**21337**	**43821**

Table 3: Statistics of logical categories in the Fact-FOLX-KG dataset.

basic diversity of our dataset. To further enhance the linguistic diversity of the generated claims, we employ two strategies: 1. LLM rewriting; 2. presupposition. The details of templates and diversification strategies are given in Appendix B.

4.3 Statistics

The distribution of logical categories in our dataset and the number of true and false instances across five reasoning categories is shown in Table 3. The dataset is balanced in true and false labels across five categories. The detailed logical abstract syntax trees are given in Appendix A. To ensure the reliability of the logical formulae and the claims in the dataset, we conduct the **Quality Control** that shows high quality of the data, which is given in Appendix D.

5 FOLX Prover

We propose a novel pipeline, FOLX Prover, for factual reasoning over KGs that utilizes executable program implementing FOLX-KG logical reasoning. Unlike previous program-guided methods (such as FoVer that generates FOL reasoning programs with first-sorted universe and unary predicate symbols (Pei et al., 2025), or PGR that designs step-by-step KG retrieval programs for graph reasoning (Hao and Wu, 2025)), our method leverages the structural clarity of FOLX-KG formulae while enabling logical inference grounded in KG semantics. The overall pipeline consists of two key stages: (1) generating executable FOLX-KG programs and (2) executing these programs over the KG to obtain verification predictions.

5.1 FOLX Prover Program Generation

Given a claim, we employ the LLM to generate a program that can execute the FOLX-KG logical formula of this claim. This program is defined over the primitives of the KG, and reflects the compositional semantics of the FOL expression. Each logical operator (e.g., conjunction, disjunction, quantification, comparison, counting) is mapped to a procedural operation that can be executed directly. We introduce three pre-defined functions for program generation: `SEARCH`, `PREDICATE`, and `COMPARISON`. `SEARCH` is for searching variables from KG, `PREDICATE` is to execute a verification of a predicate operation, `COMPARISON` is to process entity comparison.

Using the pre-defined functions for predicate operation and graph retrieval, the program can process the FOL logical reasoning steps in different program blocks. As shown in Figure 2, the program blocks ⓐ, ⓑ, ⓒ are designed for the step-by-step logical execution, with each block interpret a partial logical formula.

5.2 Program Execution over KG

The program can execute directly over the KG to complete the reasoning process by the python interpreter. Before execution, we import necessary functions and packages for each program. The program performs symbolic queries that correspond to the predicates and logical operations specified in the FOLX-KG. For each logical block, the program retrieves candidate entities and relations, applies quantifier-based selection or filtering, and computes comparison or counting outcomes where required. The final truth value of the claim is determined by composing these results according to the logical structure. The detailed execution methods are in Appendix F.

This execution framework ensures that reasoning is both interpretable and verifiable: all inference steps and the final prediction reflect the logical semantics of the FOLX-KG expression.

6 Experiments

We evaluate FOLX Prover on the Fact-FOLX-KG dataset compared with a series of baselines to assess the effectiveness of executable FOLX-KG reasoning for factual verification. All experiments are conducted on the Fact-FOLX-KG dataset with DBpedia as the KG. The dataset is split into training, development, and test sets following a 7:1.5:1.5 ratio.

6.1 Baselines

We compare our method with several baselines for KG-based fact verification. These methods can be categorized into two groups: *without evidence* and

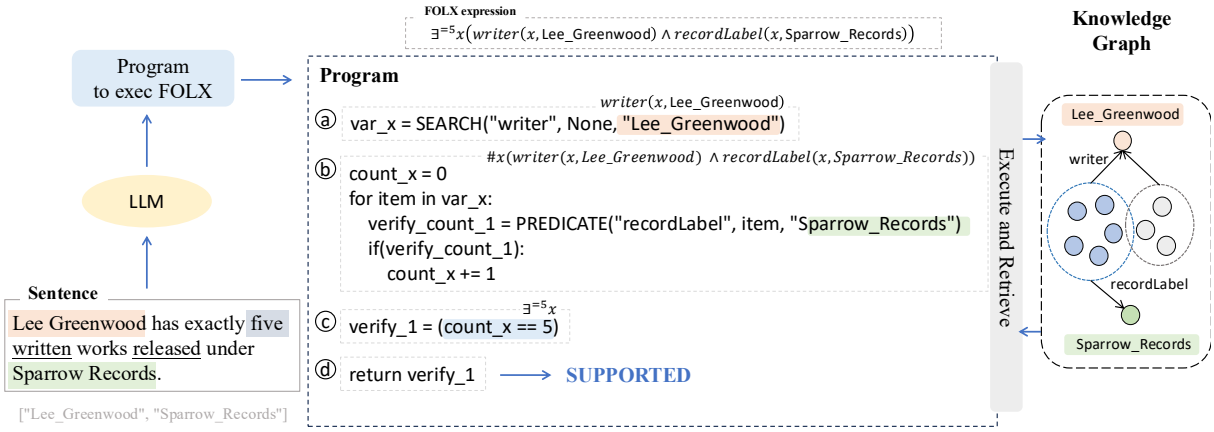


Figure 2: The pipeline of FOLX Prover to reason and execute over KG.

Method	Evi.	Quant-Free	FOL-Quant	Entity-Comp.	Count-Quant	Count-Comp.	Total
zero-shot							
GPT-4o	w/o	73.25	59.50	71.77	60.69	56.15	62.47
Deepseek-V3.2	w/o	66.50	52.47	64.72	56.05	52.46	56.70
full-training							
BERT	w/o	78.50	66.22	48.70	81.08	63.18	68.26
BERT	w/	92.25	66.37	52.80	80.43	61.42	70.12
GEAR(KG)	w/	91.17	68.45	55.12	80.81	66.32	72.50
few-shot							
KG-GPT*	w/	78.00	47.89	44.22	48.85	48.96	51.75
KG-GPT	w/	78.14	48.02	46.59	49.90	50.21	53.61
PGR(exec)*	w/	93.50	57.47	50.68	55.15	59.58	61.19
PGR*	w/	93.38	33.63	49.81	24.75	35.70	41.91
PGR	w/	93.51	36.19	51.11	30.27	37.93	44.84
FOLX-Prover*	w/	93.88	71.73	87.83	91.50	84.58	83.77
FOLX-Prover	w/	94.75	70.35	90.17	90.78	90.01	84.71

Table 4: Performance comparison across reasoning categories on Fact-FOLX-KG (using Accuracy as the metric).

with evidence. As the different training strategy, they can also be categorized into: *full training*, *zero shot* and *few shot*.

Models and methods selected as baselines on Fact-FOLX-KG: **BERT** (Devlin et al., 2019): a pre-trained language model for classification; **GPT-4o** (Achiam et al., 2023): a general LLM for language generation; **DeepSeek** (DeepSeek-AI et al., 2025): an open-sourced LLM for language reasoning and generation; **GEAR (KG)**[2]: a graph-based evidence aggregation model for fact checking; **KG-GPT**[3]: a LLM-based reasoning framework over KG; **PGR**[4]: a programmatic graph reasoning approach over KG, and **PGR(exec)** is a modified PGR with Python-based program execution.

6.2 Experiment Setup

We employ `gpt-4o-2024-08-06`[5] (without marks) and `deepseek-chat(V3.2)`[6] (marked with "*") as the generation model for executable programs. During program execution, we apply the same generation model for relation and triple retrieval over KG. For each instance in the dataset, we provide the claim and its corresponding entities, following the same design as FactKG. All the implementation details and prompts are in Appendix I and H.

6.3 Results

Table 4 summarizes the accuracy results of baselines and our proposed pipeline. Our method achieves the SOTA accuracy on the Fact-FOLX-KG test set, outperforming other full-training, zero-shot, and few-shot methods. Notably, methods that incorporate explicit logical or structural reasoning (KG-GPT and PGR) exhibit performance degra-

[2]https://github.com/jiho283/FactKG
[3]https://github.com/jiho283/KG-GPT
[4]https://github.com/hydragon/PGR
[5]https://platform.openai.com/docs/models
[6]https://www.deepseek.com

dation on claims requiring comparison or counting, these reasoning categories that are poorly supported by previous methods. Full-training KG-based baselines also struggle to consistently resolve the extended logical forms in FOLX-KG.

By contrast, our pipeline fully exploits the expressiveness of FOLX-KG, producing stable and interpretable reasoning results. The empirical study demonstrates that extending FOL to include comparison and counting is beneficial and necessary for more robust and comprehensive factual reasoning over KGs. Our pipeline also shows that programmatically executing FOLX-KG logical blocks provides a flexible mechanism for supporting varied logical reasoning requirements. More detailed result analysis and error analysis are given in Appendix G.

7 Conclusion

In this paper, we introduce FOLX-KG, a novel extended FOL σ-structure that augments KG-based factual reasoning with comparison and counting capabilities beyond standard FOL. Using this logical structure, we build Fact-FOLX-KG, a large-scale dataset pairing structured FOLX-KG formulae with diverse natural-language claims, enabling evaluation of enriched logical reasoning. We further propose FOLX Prover, an executable program–based reasoning method that converts FOLX-KG logic into interpretable procedures and executes them directly over the KG. Experiments show that our approach significantly outperforms other baselines. These results underscore the importance of extended logical expressiveness in fact verification.

Our work establishes a principled foundation for factual reasoning integrating symbolic logic and KG-based inference. Future work includes extending the FOLX-KG logical structure, improving logical reasoning robustness, and exploring retrieval-augmented methods for stronger logic–KG alignment.

Limitations

Fact-FOLX-KG is builded on DBpedia 2015 with limited knowledge before 2015. FOLX-KG can be used in more kinds of KGs and more research areas. Although Fact-FOLX-KG includes many distinct FOLX-KG syntax trees, there are still other complex syntax trees that are not considered in the fact verification dataset construction.

We build the dataset based on FOLX-KG in a case of considering extending predicate symbols of comparison. There may be more predicate symbols that can be extended in the factual reasoning dataset. The usage of counting quantifiers is for counting and comparison, which can also be extended to more kinds of arithmetic forms.

The benchmark is designed to evaluate arithmetic and logical reasoning under subgraph completeness conditions. KG incompleteness can affect universal and counting claims, which is a future research direction.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 72210107001 and in part by the Chinese Academy of Science President's International Fellowship Initiative (CAS PIFI) international Outstanding Team Project under Grant 2024PG0013. This work was supported by the National Social Science Fund of China (Major Program, Grant No. 25&ZD161).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Meiqi Chen, Yubo Ma, Kaitao Song, Yixin Cao, Yan Zhang, and Dongsheng Li. 2024. Improving large language models in event relation logical prediction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9451–9478. Association for Computational Linguistics.

Denise D Cummins, Todd Lubart, Olaf Alksnis, and Robert Rist. 1991. Conditional reasoning and causation. *Memory & cognition*, 19(3):274–282.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

Bishwamittra Ghosh, Sarah Hasan, Naheed Anjum Arafat, and Arijit Khan. 2025. Logical consistency of large language models in fact-checking. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.

Yuanzhen Hao and Desheng Wu. 2025. Fact verification on knowledge graph via programmatic graph reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5480–5495, Suzhou, China. Association for Computational Linguistics.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023a. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9410–9421, Singapore. Association for Computational Linguistics.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023b. FactKG: Fact verification via reasoning on knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16190–16206, Toronto, Canada. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Leonid Libkin. 2004. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

PER LINDSTRÖM. 1966. First order predicate logic with generalized quantifiers. *Theoria*, 32(3):186–195.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org.

Yinhong Liu, Zhijiang Guo, Tianya Liang, Ehsan Shareghi, Ivan Vulić, and Nigel Collier. 2025. Aligning with logic: Measuring, evaluating and improving logical preference consistency in large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 38518–38539. PMLR.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

Martin Otto. 1997. *Bounded variable logics and counting - a study in finite models*, volume 9 of *Lecture Notes in Logic*. Springer.

Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. 2023. Fact-checking complex claims with program-guided reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6981–7004, Toronto, Canada. Association for Computational Linguistics.

Yu Pei, Yongping Du, and Xingnan Jin. 2025. Fover: First-order logic verification for natural language reasoning. *Trans. Assoc. Comput. Linguistics*, 13:1340–1359.

Hoang Pham, Thanh-Do Nguyen, and Khac-Hoai Nam Bui. 2025. ClaimPKG: Enhancing claim verification via pseudo-subgraph generation with lightweight specialized LLM. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5271–5290, Vienna, Austria. Association for Computational Linguistics.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International*

Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through logicnli. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3738–3747. Association for Computational Linguistics.

Haoran Wang and Kai Shu. 2023. Explainable claim verification via knowledge-grounded reasoning with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6288–6304, Singapore. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Jundong Xu, Hao Fei, Meng Luo, Qian Liu, Liangming Pan, William Yang Wang, Preslav Nakov, Mong-Li Lee, and Wynne Hsu. 2025. Aristotle: Mastering logical reasoning with a logic-complete decompose-search-resolve framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3075, Vienna, Austria. Association for Computational Linguistics.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2023. Satlm: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

G. Yule. 1996. *Pragmatics*. Oxford Introduction to Language Study ELT. OUP Oxford.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. GEAR: Graph-based evidence aggregating and reasoning for fact verification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 892–901, Florence, Italy. Association for Computational Linguistics.

A Fact-FOLX-KG Distinct Abstract Syntax Trees

The distinct Abstract Syntax Tree (AST) statistics is given in Table 5. There are more than 20 different ASTs in Fact-FOLX-KG to diversify the logical types. For each AST, there are several different triple combinations and templates to diversify the claims.

In Table 6, we summarize both the total occurrence claim counting and the maximum number of occurrences within a single claim.

B Templates and Diversification of Dataset

B.1 Templates

To generate natural-language claims aligned with the semantics of each reasoning type, we design a diverse collection of templates that correspond directly to the underlying FOLX-KG logical structures. For every reasoning category, we examine the logical operators, predicate types, and KG subgraph patterns involved, and construct multiple natural-language templates that capture the same semantics while increasing linguistic variation.

As shown in Table 7, each reasoning type is associated with a set of templates that abstract over entity placeholders. Once a KG subgraph is matched to a particular FOL formula, the corresponding template can simply substitute the appropriate entities into the corresponding places. This templated strategy enables scalable claim generation while preserving faithful alignment between logical forms and natural-language expressions. We asked two NLP PhD students to write natural-language templates corresponding to each FOL formula with relational semantics, and two Master's students reviewed these templates to ensure their correctness.

In addition to the base templates used for direct semantic rendering of FOL formulae, we also design a collection of presupposition prefix templates to further diversify the linguistic forms of generated claims. These prefixes fall into two categories: *factive* presuppositions and *non-factive* presuppositions (Kim et al., 2023b; Yule, 1996). The full set

Category	FOL AST Formula Types
Quant Free	$\varphi_1 \wedge ... \wedge \varphi_k$; $\varphi_1 \wedge ... \wedge \neg\varphi_k$; $\neg(\varphi_1 \wedge ... \wedge \varphi_k)$; $\varphi_1 \vee ... \vee \varphi_k$; $\neg(\varphi_1 \vee ... \vee \varphi_k)$; $\varphi_1 \vee \neg\varphi_2$
FOL Quant	$\exists x\, \varphi(x)$; $\neg\exists x\, \varphi(x)$; $\forall x\, \varphi(x)$; $\forall x\, \neg\varphi(x)$; $\neg\forall x\, \varphi(x)$; $\forall x\, \exists y\, \varphi(x, y)$ $\neg(\forall x\, \exists y\, \varphi(x, y))$
Entity Comparison	$\exists x\, \exists y\, (\varphi_1(x) \wedge \varphi_2(y) \wedge x\{>, <\}y)$; $\neg(\exists x\, \exists y\, (\varphi_1(x) \wedge \varphi_2(y) \wedge x\{>, <\}y))$
Counting Quant	$\{\exists^{=k}, \exists^{\geq k}, \exists^{\leq k}, \exists^{>k}, \exists^{<k}\} x\, \varphi(x)$; $\neg\{\exists^{=k}, ...\} x\, \varphi(x)$; $\neg(\{\exists^{=k}, ...\} x\, (\varphi(x) \wedge \psi(x)))$; $\{\exists^{=k}, ...\} x\, (\varphi(x) \wedge \psi(x))$
Counting Comparison	$\exists i\, \exists j(\exists^{=i} x\, \varphi(x) \wedge \exists^{=j} y\, \psi(y) \wedge i\{\geq, >, =, \leq, <\}j)$; $\neg(\exists i\, \exists j(\exists^{=i} x\, \varphi(x) \wedge \exists^{=j} y\, \psi(y) \ \wedge \ i\{\geq, ...\}j))$

Table 5: Categories of Claims and Their Corresponding FOL distinct AST.

Symbol	Total	Max
$\wedge$	20,710	3
$\vee$	2,308	2
$\neg$	4,256	1
$\rightarrow$	11,803	1
$\exists$	14,164	2
$\forall$	11,803	1
$\exists^{\geq}$	18,824	2

Table 6: Statistics of logical symbol occurrences in the dataset. Total claims containing the symbol and Max occurrences per claim.

of presuppositional prefixes is provided in Table 8.

Together, these templates support flexible and semantically faithful claim generation across different reasoning types, ensuring linguistic diversity while maintaining strict logical correspondence to the FOL formulae.

B.2 Diversification

The first diversification strategy uses LLMs to rewrite template-generated claims while preserving their original logical content. This rewriting produces claims that express more natural, varied, and sometimes more conversational styles, which reduce stylistic similarity across claims. We quantify lexical diversity using Type–Token Ratio (TTR) after LLM rewriting:

Before LLM rewrite: TTR = 0.0828;

After LLM rewrite: TTR = 0.0906.

The second strategy introduces diversity through presuppositional augmentation (Yule, 1996). Following the presupposition-insertion paradigm, we prepare either factive or non-factive presuppositional utterances to the original claim, creating richer pragmatic contexts without altering the claims. In addition to the presupposition types adopted in FactKG, we include more presuppositional triggers, further increasing stylistic and pragmatic variety.

These methods make our dataset reflect a wide range of linguistic expressions while maintaining precise alignment with the FOL logical structures.

C Comparing FOL with Query Language

FOLX-KG is intended as a logical formalization of reasoning problems over KGs. Since FOL is close to human-level logical reasoning (Xu et al., 2025), it is directly used in many reasoning researches (Xu et al., 2025; Liu et al., 2020; Tafjord et al., 2021; Saparov and He, 2023). Query languages are primarily designed for data retrieval and construction over specific storage models. FOL is a foundational formalism in mathematical logic for expressing truth-conditional statements over logical structures.

FOL provides: direct symbolic logical reasoning; mathematical analysis and proofs; and theoretic semantics enabling rigorous analysis of satisfiability, entailment, equivalence, and locality. Therefore, FOL serves as a more basic and mathematically principled representation.

D Data Quality Control

Quality control procedures ensure the reliability of both components of the dataset: the FOL logical formulae and the natural-language claims. Our validation process consists of two stages.

Quality Control of FOL Formula. To assess the correctness of the FOL expressions, we implement the FOL interpreter capable of executing each formula over the KG. The interpreter evaluates predicate operations, searches for supporting evidence in the KG, and computes the truth value of the full logical expression. For every reasoning type in our dataset, we compare the interpreter's

Reasoning Type	FOLX-KG Formula	Template Examples
FOL Quant (∀)	$\forall x\,(\textit{predecessor}(e_1, x) \rightarrow \textit{placeOfBirth}(x, e_2))$	All predecessors of ***e_1*** were born in ***e_2***. All the people who preceded ***e_1*** were born in ***e_2***.
Entity Comparison	$\exists x \exists y\,(\textit{width}(e_1, x) \wedge \textit{width}(e_2, y) \wedge x < y)$	The ***e_2*** is noticeably wider than the ***e_1***. The ***e_1*** is narrower than the ***e_2***.
Counting Quantifier	$\exists^{\geq num} x\,(\textit{writer}(e_1, x) \wedge \textit{genre}(x, e_2))$	There are at least ***num*** ***e_2*** works written by ***e_1***. ***e_1***'s authored pieces include no fewer than ***num*** ***e_2*** works.
Counting Comparison	$\exists i \exists j\,(\exists^{=i} x\,\textit{manufacturer}(e_1, x) \wedge \exists^{=j} y\,\textit{manufacturer}(e_2, y) \wedge i > j)$	***e_1*** has more manufacturers than ***e_2***. The number of manufacturers of ***e_1*** exceeds that of ***e_2***. Compared with ***e_2***, ***e_1*** has more manufacturers.

Table 7: Examples of templates aligned with FOLX-KG formulae in different reasoning types.

Factive Presupposition Prefixes
I forgot that; I realized that; I didn't know that; I wasn't aware that; I explained that; I emphasized that; I remembered that; I understand that; It isn't odd that; Everybody knows that; Everybody is aware that; All people know that; All people are aware that; Everyone knows that
Non-factive Presupposition Prefixes
I imagined that; I wish that; If only; I dreamed that; We imagined that

Table 8: Factive and non-factive presupposition templates.

output against the annotated labels. The results show perfect agreement: all FOL formulae in the dataset achieve a 100% match accuracy between predicted value and ground-truth annotation.

Quality Control of Claim Consistency. To ensure the semantic correctness of natural-language claims, we conduct manual evaluation focusing on the alignment between each claim and its corresponding FOLX-KG formula. Annotators assess whether the claim faithfully expresses the logical content encoded in the formula. For template-generated claims, two graduate students majoring in natural language processing were asked to annotate 5000 randomly selected claims for the semantic consistency and the average correctness reaches 96%. For claims obtained through substitution-based augmentation, LLM rewriting, and presupposition insertion, we also asked two graduate students to annotate 5000 randomly selected claims, and the average accuracy is 94%. We computed IAA on an overlapping subset of 200 samples and the Cohen's Kappa is 0.86, which indicates strong agreement and supports the reliability of the annotation process. These results demonstrate that the dataset maintains high logical correctness: the FOL formulae are correct with respect to KG, and the associated claims accurately reflect the intended logical semantics.

E Proof of Counting Comparison Negation

The existential quantifier formula

$$\exists i \exists j(\exists^{=i} x \varphi_1(x) \wedge \exists^{=j} y \varphi_2(y) \wedge (i < j))$$

can be written as a disjunctive formula with i, j range over the universe:

Formula 1: $\bigvee_{i,j} \big((\#1 = i) \wedge (\#2 = j) \wedge (i < j)\big)$

where $\exists^{=i} x \varphi_1(x)$ and $\exists^{=j} y \varphi_2(y)$ are simplified as $\#1 = i$ and $\#2 = j$.

The universal quantifier formula

$$\forall i \forall j(\exists^{\neq i} x \varphi_1(x) \vee \exists^{\neq j} y \varphi_2(y) \vee (i < j))$$

can be written as a conjunctive formula with i, j range over the universe:

Formula 2: $\bigwedge_{i,j}\big((\#1 \neq i) \vee (\#2 \neq j) \vee (i < j)\big)$

As shown in Figure 3, with the Truth Table of Formula 1 and 2, we can draw all the constant (i^c, j^c) and its quantifier-free formula true-false value.

For all pairs (i, j), only the pair $(\#1, \#2)$ makes

$$(\#1 = i \wedge \#2 = j) = \text{True},$$

it follows that:

$$\text{Formula 1 is True} \iff \#1 < \#2 \text{ is True.}$$

Similarly, the same proof applies to Formula 2, hence:

$$\text{Formula 2 is True} \iff \#1 < \#2 \text{ is True.}$$

Therefore, we can get Formula 1 $\equiv$ Formula 2

F FOLX Prover Execution Process

To realize FOLX-KG formulae as executable reasoning procedures, we implement a program execution pipeline that directly interfaces with the KG. Each generated program is interpreted using a Python-based execution environment, into which we import the predefined `SEARCH`, `PREDICATE`, and `COMPARISON` functions. In Figure 4, there are some examples of programs with logical blocks correspond to FOLX-KG formulae.

F.1 Search and Predicate Execution

The `SEARCH` and `PREDICATE` functions implement KG-matching operations and follow the searching and matching strategies established in the PGR framework. Specifically, `SEARCH` identifies candidate entities or triples by querying the KG for relations involving the given head or tail nodes, while `PREDICATE` evaluates whether a particular triple is matched in KG. These procedures ensure consistency with prior graph-based reasoning pipelines.

To improve computational efficiency, we introduce a global variable `relation_match_map` that caches previously matched relations during program execution. This caching mechanism reduces redundant LLM-based KG matching within search loops, especially for formulae involving multiple quantifiers or repeated predicate evaluations.

Algorithm 1 Comparison Function

```
function COMPARE(op, A, B)
    is_date ← CHECKDATE(A, B)
    X ← NORMALIZE(A, is_date)
    Y ← NORMALIZE(B, is_date)
    if X = ∅ or Y = ∅ then
        return False
    end if
    return COMPAREOPERATION(op, X, Y)
end function
```

Algorithm 2 NORMALIZE

Require: A node value v

```
if v matches one of the predefined date patterns then
    convert v into a floating-point timestamp
    return timestamp
else if v is numeric then
    return FLOAT(v)
else
    return None
end if
```

F.2 Comparison Function

The `COMPARISON` function is newly defined to support the extended logical capabilities of FOLX. It handles numerical comparisons between entity values retrieved from the KG, enabling evaluation of predicates such as $<$, $>$, $\leq$, $\geq$, and $=$. This function plays a central role in executing formulae belonging to the ENTITY COMPARISON. By integrating numerical reasoning into the symbolic execution pipeline, the program can faithfully reproduce the semantics of the comparison predicates. The algorithm of `COMPARISON` function is shown in Algorithm 1.

G Result Analysis

We analyze model performance across different experimental settings to understand how various reasoning strategies handle the extended logical phenomena introduced in FOLX-KG.

In the **zero-shot** setting, both `GPT-4o` and `DeepSeek` exhibit limited effectiveness when no KG evidence is provided. Relying solely on parametric knowledge, these models struggle to accurately verify KG-grounded claims, particularly when claims involve structured predicates or logical dependencies that cannot be inferred from general model knowledge. The consistently low accu-

$\forall i \forall j(\#1 \neq i \vee \#2 \neq j \vee i < j)$

i	j	$\#1 \neq i$	$\#2 \neq j$	$i < j$	$\#1 \neq i \vee \#2 \neq j \vee i < j$
0	0	T	T	F	T
...	...	...	...	...	...
0	n-1	T	T	T	T
...	...	...	...	...	...
#1	0	F	T	F	T
...	...	...	...	...	...
#1	#2	F	F	T (F)	T (F)
...	...	...	...	...	...
#1	n-1	F	T	T	T
...	...	...	...	...	...
n-1	0	T	T	F	T
...	...	...	...	...	...
n-1	n-1	T	T	F	T

$\exists i \exists j(\#1 = i \wedge \#2 = j \wedge i < j)$

i	j	$\#1 = i$	$\#2 = j$	$i < j$	$\#1 = i \wedge \#2 = j \wedge i < j$
0	0	F	F	F	F
...	...	...	...	...	...
0	n-1	F	F	T	F
...	...	...	...	...	...
#1	0	T	F	F	F
...	...	...	...	...	...
#1	#2	T	T	T (F)	T (F)
...	...	...	...	...	...
#1	n-1	T	F	T	F
...	...	...	...	...	...
n-1	0	F	F	F	F
...	...	...	...	...	...
n-1	n-1	F	F	F	F

Figure 3: Truth table of counting comparison formula.

racy across reasoning categories underscores the necessity of evidence-aware and logic-grounded methods for KG-based factual verification.

For **full-training** baselines, **BERT (without evidence)** achieves moderate accuracy on some claim types but remains below the performance required to handle all the reasoning categories. When trained with evidence, both **BERT** and **GEAR** show improvements primarily within the QUANT FREE category, but exhibit minimal gains on claims requiring complex quantifier, comparison operations, or counting. These results suggest that NLI models—even when augmented with retrieved KG triples—struggle to capture the quantifier-bounded semantics inherent in factual reasoning.

Among **few-shot** baselines, **KG-GPT** extracts structured evidence and uses an LLM to conduct reasoning, yet its performance remains high only on QUANT FREE claims. Accuracy decrease sharply for other reasoning categories, reflecting difficulty of KG-GPT in retrieving correct evidence and consistently executing multi-step counting or comparison logic.

PGR, which generates graph reasoning programs before executing them over the KG, performs more reliably on QUANT FREE instances. However, its accuracy drops substantially on other reasoning categories. This degradation is attributable to design limitations in PGR's logic forms and program execution mechanism.

FOLX Prover achieves the highest accuracy across all reasoning categories, consistently outperforming both LLM-based and program-guided baselines. The strong performance across all categories demonstrates that FOLX Prover is well-suited for verifying claims involving richer logical structures, and further indicates that extending FOL is crucial for capturing the full reasoning phenomena required in KG-based factual verification.

G.1 Error Analysis

The error analysis is given in Table 9 with three error types: Syntactic Error; Logical Error and Execution Error. Syntactic Error is the program syntax issues that make the program unable to execute. Logical Error is the wrong logic in the program expressing the claim logic. Execution Error is the wrong execution of the program with correct logic to output wrong result. We randomly selected 100 wrong predicted results from each method and labeled these results with error types.

FOLX Prover has lower Syntatic error ratio than PGR, because the program execution of PGR is designed by a specific code interpreter that cannot process additional code logics. FOLX Prover is designed to execute the program using python interpreter, and import pre-defined functions before execution, which can reduce syntactic errors and execute more code logic. Different LLMs also affect the error ratios in the generated programs, most syntatic errors of FOLX Prover (DeepSeek-V3.2) are Counting Comparison claims, of which the LLM incorrectly identifies the Entity Comparison and Counting Comparison claims.

G.2 Results on FactKG

We conducted additional experiments of FOLX Prover on FactKG, comparing against previous methods. The accuracy results on FactKG are

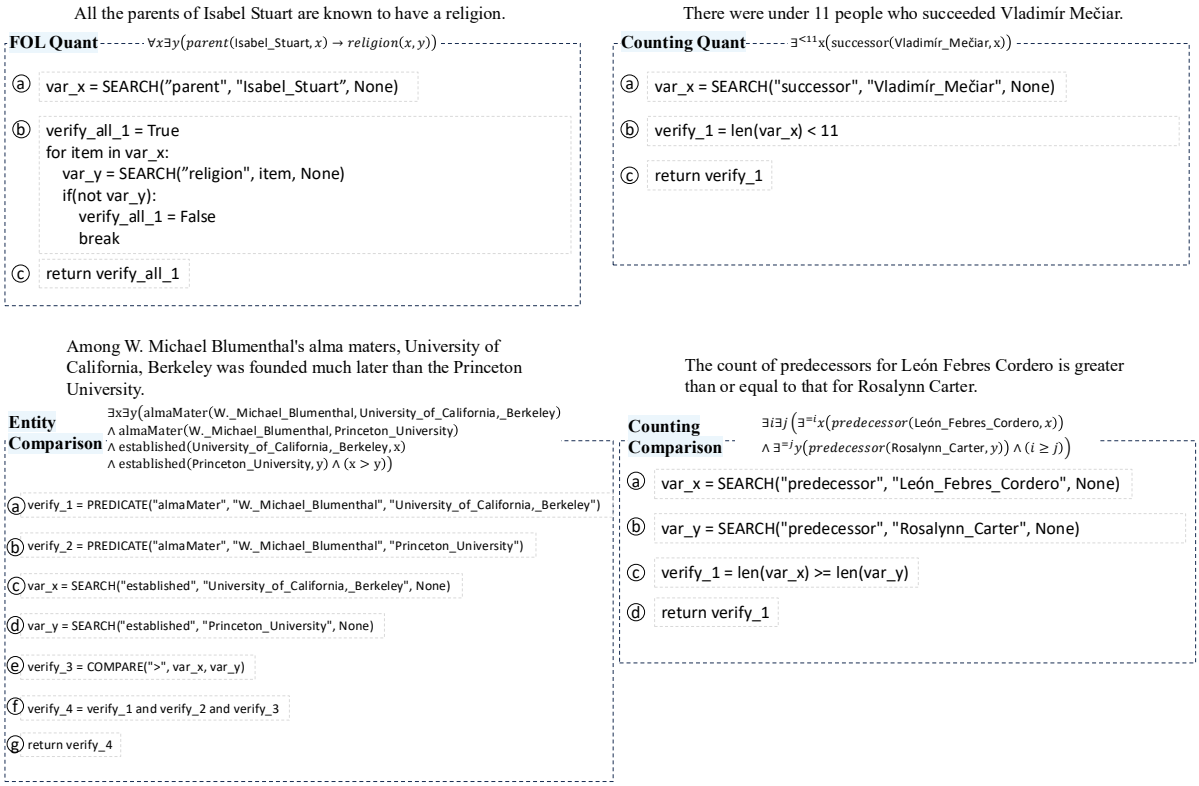


Figure 4: Examples of executable program blocks of FOLX Prover.

	FOLX Prover		
Error Type	**GPT-4o**	**DeepSeek-V3.2**	**PGR**
Syntactic	1%	8%	31%
Logical	53%	43%	48%
Execution	46%	49%	21%

Table 9: Distribution of Error Types of FOLX Prover in different LLM settings, and PGR error distribution.

shown in the Table 10.

The results show that FOLX Prover achieves performance close to previous SOTA methods, without great degradation across reasoning types. Importantly, this was achieved without task-specific redesign of the framework, demonstrating that FOLX Prover transfers effectively across benchmarks. All few-shot methods are implemented 12-shot and use DeepSeek as the base model. For FOLX Prover, we keep the original prompt design and only added two FactKG-specific examples.

G.3 Efficiency

We have computed the following statistics on the test set, including different function calls and LLM usage per executing program:

Primitive Call Counting.

- Average SEARCH calls: 1.54
- Average PREDICATE calls: 0.87
- Average COMPARE calls: 0.15

LLM Inference Cost.

- Average LLM calls per instance: 1.89
- Average input tokens: 3180
- Average output tokens: 169

Effect of Caching (relation match map): LLM inference calls are reduced by approximately 30%.

H Prompts used in Experiment

The prompt of FOLX Prover is a few-shot prompt with program function explanations, as shown in Figure 5.

The zero-shot prompt for LLMs to verify claims is "*Verify the given claim and output the verification result (True or False). Choose one of True, False and output in JSON format: [true] or [false]*".

Method	Existence	One-hop	Multi-claim	Multi-hop	Negation	Overall
GEAR (KG)	81.61%	83.23%	77.68%	68.84%	79.41%	77.65%
KG-GPT	79.21%	87.43%	88.08%	62.61%	61.06%	77.43%
PGR	89.17%	90.41%	89.32%	73.14%	78.24%	84.18%
FOLX Prover	82.97%	85.03%	85.41%	71.97%	76.57%	80.93%

Table 10: Accuracy across different claim types in FactKG.

```
Please generated a program for the given <<<Sentence>>> in Your Task.
You can use the 3 functions 'SEARCH', 'PREDICATE' and 'COMPARE' in the program.

1. 'SEARCH':
  - Input: A triple `(relation, ..., ...)` with one `None` value (missing entity).
  - Functionality: Find out the missing entities based on the triple.
  - Output: A list of possible entities for the missing part.

2. 'PREDICATE':
  - Input: A complete triple `(relation, entity, entity)`, entity is given in the
sentence or output of 'SEARCH'.
  - Functionality: Verify if the triple exists in the graph.
  - Output: Returns `True` if the triple exists; otherwise, `False`.

3. 'COMPARE':
  - Input: A complete triple `(relation, entity, entity)`, relation is a comparison
symbol, entity is the output of 'SEARCH'.
  - Functionality: Verify if the comparison between two entities is correct.
  - Output: Returns `True` if it is correct; otherwise, `False`.

Examples:

<<<Sentence>>>:
English is spoken in Great Britain and is the language used in 1634: The Ram
Rebellion.
<<<Entities>>>:
["Great_Britain", "1634:_The_Ram_Rebellion", "English_language"]
<<<Program>>>:
def program():
   verify_1 = PREDICATE("language", "1634:_The_Ram_Rebellion",
"English_language")
   verify_2 = PREDICATE("spokenIn", "English_language", "Great_Britain")
   verify_3 = verify_1 and verify_2
   return verify_3
...

Your Task:
```

Figure 5: The prompt of FOLX Prover.

I Experimental Implementation

I.1 LLM Configuration

For both zero-shot and few-shot experiments, we employ a consistent set of hyperparameters for the LLMs: temperature is fixed at 0.2 and top-p at 0.1. This configuration is the same with the setups used in KG-GPT and PGR, and is chosen to reduce sampling variability and promote stable outputs during generation. KG-GPT and PGR are all 12-shot prompted for generation. FOLX Prover is 10-shot prompted for program generation.

I.2 Full-Training Models

For models trained or fine-tuned on the dataset, we follow standard training protocols established in previous work.

The GEAR(KG) model is trained following the same settings as reported in the original FactKG paper, including optimizer configuration, evidence aggregation strategy, and KG retrieval pipeline.

For the BERT (`bert-base-uncased`) model with KG evidence, we extract subgraphs using the similar KG retrieval strategy as GEAR(KG), selecting triples directly connected to the entities. The model is trained for three epochs, and we select the checkpoint that achieves the highest performance on the development set. This checkpoint is then evaluated on the test set for final results.