Entailment-Preserving First-order Logic Representations in Natural Language Entailment

Jinu Lee, Qi Liu, Runzhi Ma, Vincent Han, Ziqi Wang, Heng Ji, Julia Hockenmaier
University of Illinois Urbana-Champaign
{jinulee2, ziqiw9, hengji, juliahmr}@illinois.edu

Abstract

First-order logic (FOL) is often used to represent logical entailment, but determining natural language (NL) entailment using FOL remains a challenge. To address this, we propose the Entailment-Preserving FOL representations (EPF) task and introduce reference-free evaluation metrics for EPF (Entailment-Preserving Rate (EPR) family). In EPF, one should generate FOL representations from multi-premise NL entailment data (*e.g.* EntailmentBank) so that the automatic prover's result preserves the entailment labels. Furthermore, we propose a training method specialized for the task, *iterative learning-to-rank*, which trains an NL-to-FOL translator by using the natural language entailment labels as verifiable rewards. Our method achieves a 1.8–2.7% improvement in EPR and a 17.4–20.6% increase in EPR@16 compared to diverse baselines in three datasets. Further analyses reveal that iterative learning-to-rank effectively suppresses the arbitrariness of FOL representation by reducing the diversity of predicate signatures, and maintains strong performance across diverse inference types and out-of-domain data.

1 Introduction

First-order logic (FOL) expressions are frequently used as a semantic representation of natural language (NL) (Bos, 2014; Han et al., 2024; Yang et al., 2023). FOL representations are well-suited for expressing the semantics of *logical entailment*, where the hypothesis *necessarily follows* from the lexical meaning of the premises and the logical rules (*e.g.* syllogisms). Logical entailment can be easily determined with FOL representations by using the automatic theorem prover.

On the other hand, recognizing textual entailment (RTE) tasks (Dagan et al., 2005; Camburu et al., 2018; Dalvi et al., 2021) adopt a broader definition of entailment, hereby referred to as **natural language entailment**. Natural language entailment can be defined as: "P entails h if a human reading P would infer that h *is most likely true*" (Dagan et al., 2005), which is a strictly looser condition compared to logical entailment.

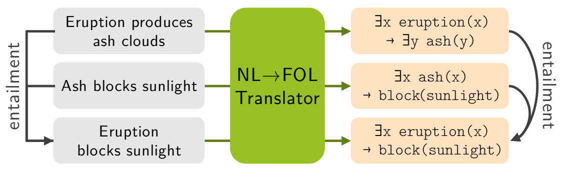

entailment
Eruption produces ash clouds
Ash blocks sunlight
Eruption blocks sunlight
NL→FOL Translator
∃x eruption(x) → ∃y ash(y)
∃x ash(x) → block(sunlight)
∃x eruption(x) → block(sunlight)
entailment

Figure 1: Overview of Entailment-Preserving FOL representations (EPF). When premises entail a hypothesis (gray), the model (green) should produce FOL representations (orange) that preserve the entailment, which can be checked by an automatic theorem prover.

Can we determine natural language entailment using FOL representations? While it is intriguing to harness the soundness and efficiency of FOL for understanding natural language semantics, no prior work on NL→FOL translation has successfully tackled the question. Classic NL→FOL parsing approaches that translate syntactic and semantic parses to FOL were not able to preserve entailment in single-premise RTE tasks (Bos and Markert, 2005; Bos, 2014). Alternatively, recent methods use large language models (LLMs) to generate FOL representations from NL, and apply an automatic theorem prover to prove or disprove the given hypothesis (Olausson et al., 2023; Pan et al., 2023). While these methods were proven effective for logical entailment tasks (Tafjord et al., 2021; Han et al., 2024), the generalizability of these methods to natural language entailment is not yet evaluated.

As a systematic approach to this long-standing problem, we formalize the **Entailment-Preserving FOL representations (EPF)** task. In the EPF task, one must generate FOL representations for each premise and hypothesis in a multi-premise RTE dataset that preserves the entailment label. Since the ground truth FOL representations for premises and hypotheses are not determined, one cannot apply supervised fine-tuning for solving EPF. Along with the task, we present a suite of *reference-free* metrics for the EPF task, namely the **entailment-preserving rate (EPR)** family.

We empirically show that existing approaches for obtaining FOL representations, including classic meaning representations-based methods and end-to-end generative models, cannot effectively solve the EPF task. To advance the state-of-the-art, we develop a novel **iterative learning-to-rank** training method which uses natural language entailment labels as *verifiable rewards*. This method rewards FOL representations that preserve the entailment and penalizes ones that cannot, pushing the model to produce more entailment-preserving FOL representations. Experiments show that a T5 (Raffel et al., 2020) model trained using the proposed method significantly outperforms diverse baselines on the EPF task on three multi-premise entailment datasets (EntailmentBank (Dalvi et al., 2021), eQASC (Jhamtani and Clark, 2020), and e-SNLI (Camburu et al., 2018)). Furthermore, our analyses show that the proposed training method can reduce the unwanted arbitrariness in FOL predicates and generalize to diverse inference types and out-of-domain data.

Our key contributions can be summarized as follows.

- We formalize the Entailment-Preserving FOL representations (**EPF**) task, where one must produce FOL representations that preserve the entailment in multi-premise RTE datasets. We empirically show that the task is challenging for diverse NL→FOL translator baselines.
- We develop a suite of reference-free metrics, the Entailment-Preserving Rate (EPR) family, for evaluating performance in EPF.
- We propose **iterative learning-to-rank** training for EPF that significantly outperforms diverse baselines. We perform multiple analyses to show that the method effectively reduces arbitrariness and is robust to various in-domain and out-of-domain data distributions.

2 Related Work

2.1 FOL and natural language semantics

Inspired by formal semantics (Montague et al., 1970; Parsons, 1990), first-order logic (FOL) has often been used as a semantic representation for natural language. FOL offers a sound, efficient, and interpretable framework for computing natural language entailment (Pan et al., 2023; Quan et al., 2024). However, its limited expressive power makes it difficult to capture more complex meaning, such as higher-order logic and uncertainty (Bos, 2014; Olausson et al., 2023).

Even when a sentence's meaning can be precisely encoded in FOL, challenges arise in modeling the *interaction* between multiple sentences. Two common issues are **arbitrariness**, where different predicate names and logical forms can be used to express the same meaning (Olausson et al., 2023), and **brittleness**, where some semantic information, including synonymy and commonsense, is lost during translation from natural language to FOL (Bos, 2014). For instance, the sentence "Eruption produces ash clouds" from Figure 1 can be interpreted as `produce(eruption,ash_cloud)` or `∃x.eruption(x) → ∃y.ash_cloud(y)` (arbitrariness), where the former fails to interact with "Ash blocks sunlight" to entail the hypothesis because there are no common predicates (brittleness).

2.2 NL→FOL translation

Since the start of the RTE challenge (Dagan et al., 2005), multiple works have applied FOL representations to solve natural language entailment. These methods first obtain the syntactic/semantic parse tree and apply a rule-based transformation to get the FOL representation (Bos and Markert, 2005; Bos, 2014). However, it was repeatedly shown that these FOL representations are not empirically effective in solving natural language entailment. For instance, Bos (2014) reported that FOL representations translated from the discourse representation structure (DRS) yield only 1.9% recall in detecting the entailment in the single-premise RTE benchmark (Dagan et al., 2005).

Independent from these works, multi-premise logical entailment benchmarks (Tafjord et al., 2021; Tian et al., 2021; Han et al., 2024) were developed to evaluate the reasoning ability of generative models. These benchmarks adopt the classic 3-way entailment label classification format (*entailment, contradiction, neutral*) of single-premise RTE tasks, in which both the NL sentences and their gold FOL representations point to the same entailment label. Recent works have applied LLMs to obtain FOL representations for these tasks, fueled by the code generation ability of LLMs (Pan et al., 2023; Olausson et al., 2023; Yang et al., 2023; Ryu et al., 2024). While they achieve significant performance in synthetic, controlled logical reasoning benchmarks, whether they can generalize to natural language entailment has remained unanswered.

2.3 Executable semantic representations

Apart from FOL, a stream of research focuses on the *executability* of semantic representations. From this perspective, semantic representations are *program codes* that can be executed to solve downstream tasks, such as query intent analysis (Yu et al., 2018; Dligach et al., 2022) and question answering (Xu et al., 2014). The performance of the semantic parser is directly assessed by the accuracy of execution results for the downstream tasks, rather than the similarity between the prediction and the reference parse.

To improve the execution accuracy that is often non-differentiable, reinforcement learning (RL) and its variants have been applied to train neural semantic parsers (Cheng et al., 2019; Cheng and Lapata, 2018). Using only the input sentence and the desired execution result, these methods learn to maximize the probability of the representations that lead to the correct execution result. However, these approaches are not directly applicable to EPF, as EPF requires taking account of *interactions between premises and hypotheses* during execution, while previous works often assume that sentences are isolated.

3 Methods

3.1 Entailment Preserving Rate (EPR)

Evaluating the quality of FOL representations is challenging because (1) a sentence can have multiple semantically valid FOL representations, and (2) natural language entailment datasets do not usually provide reference FOL representations. In this study, we define **Entailment-Preserving Rate (EPR)** and its variants, a suite of reference-free metrics that only require entailment labels.

Given an RTE dataset consisting of premises-hypothesis pairs, the EPR of a NL→FOL translator S is measured by the ratio of the pairs where S can preserve the entailment. Specifically, the translator S translates each premise and hypothesis to FOL representations, obtaining the premise representations $S(P) = S(p_1), ..., S(p_N)$ and the hypothesis representation $S(h)$. Then, an automatic prover

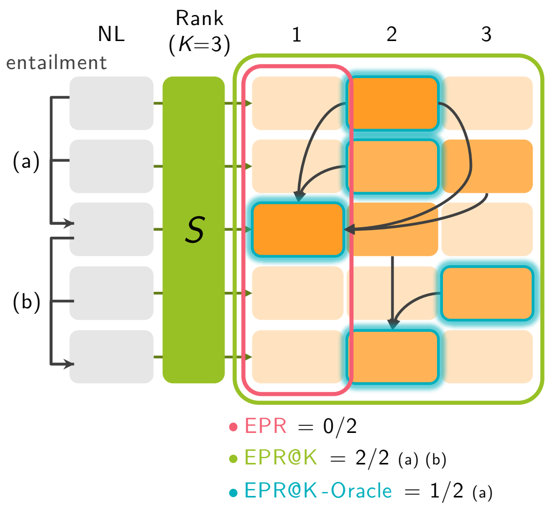


Figure 2: Comparison between EPR, EPR@K, and EPR@K-Oracle. The translator S generates $K = 3$ FOL representations (orange) for each NL sentence (gray), where curved arrows represent entailment-preserving combinations. EPR only uses the top 1 predictions for each sentence (red box), where EPR=0/2 because there are no entailment-preserving combinations. EPR@K uses all K predictions (green box) which contain such combinations for both (a) and (b), having a value of 2/2. Finally, EPR@K-Oracle (blue neon) selects one parse from each sentence that maximizes the global EPR value. In this example, there is a selection that preserves entailment in either (a) or (b) but not both, resulting in EPR@K-Oracle=1/2.

is used to determine if $S(h)$ can be proved from $S(P)$[1]. Finally, a verification step is imposed to filter out spurious entailments caused by contradiction (Appendix A).

Note that this definition is completely *reference-free* because it does not require a comparison between predicted and gold FOL representations. Furthermore, EPR allows FOL representation to have arbitrary predicate names and logical structures as long as they combine with others to complete a proof, being robust to *arbitrariness*.

We also propose a natural, loose extension of EPR to exploit multiple outputs that can be obtained by beam search and sampling. First, **EPR@K** allows up to K different parses for each premise and hypothesis. If *any* combination of FOL representations selected from each premise and hypothesis preserves the entailment, it is considered a success.

Finally, **EPR@K-Oracle** allows only one FOL

[1]Contradiction in RTE can be treated as proving $\neg S(h)$ instead of $S(h)$, without loss of generality.

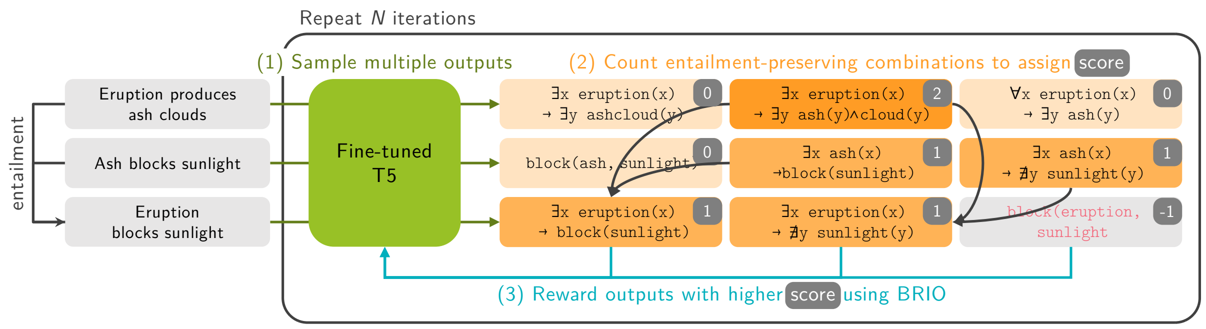

Repeat N iterations
(1) Sample multiple outputs
(2) Count entailment-preserving combinations to assign score
entailment
Eruption produces ash clouds
Ash blocks sunlight
Eruption blocks sunlight
Fine-tuned T5
∃x eruption(x) → ∃y ashcloud(y)
0
∃x eruption(x) → ∃y ash(y)∧cloud(y)
2
∀x eruption(x) → ∃y ash(y)
0
block(ash, sunlight)
0
∃x ash(x) →block(sunlight)
1
∃x ash(x) → ∄y sunlight(y)
1
∃x eruption(x) → block(sunlight)
1
∃x eruption(x) → ∄y sunlight(y)
1
block(eruption, sunlight
-1
(3) Reward outputs with higher score using BRIO

Figure 3: Iterative Learning-to-rank approach to train an entailment-preserving NL→FOL translator. (1) For each premise and hypothesis, multiple FOL representations are sampled using beam search. (2) An external solver counts all entailment-preserving combinations and assigns scores. (3) Finally, the learning-to-rank objective BRIO is applied to reward the outputs participating in the most entailment-preserving combinations, indirectly increasing the overall EPR. This training loop (1-3) is repeated for multiple iterations to maximize performance.

representation per sentence (premise and hypothesis), similar to EPR. However, instead of selecting the output with the highest model-assigned probability as EPR, outputs are selected from each sentence to maximize the global EPR. This can be viewed as adding an *oracle reranker* that always selects the globally optimal output from K candidates. As this constrained optimization problem is a variant of MAX-SAT that is NP-hard, we use Answer Set Programming (Lifschitz, 2019) to get an approximate solution (details in Appendix A.2).

This inequality holds by definition: EPR $=$ EPR@1 $\leq$ EPR@K-Oracle $\leq$ EPR@K. Notably, EPR@K-Oracle serves as an *least upper bound* for EPR. Assume an *ideal* NL→FOL translator that can capture every aspect of meaning that can be represented using first-order logic (*i.e.*, excluding higher-order or uncertainty). When a translator achieves an EPR@K-Oracle score of $n\%$, this ideal translator can achieve at least an EPR score of $n\%$. This shows the

3.2 Iterative Learning-to-rank

In this section, we describe the **iterative learning-to-rank**, which leverages entailment labels as verifiable rewards for training an NL→FOL translator.

3.2.1 Scoring function

The goal of the EPF task is to train a model that translates a sentence (premise or hypothesis) to its FOL representation. Leveraging the definition of EPR, we define a reference-free sentence-level scoring function such that optimizing it will naturally enhance the global EPR score.

Given a model output $S(p)_j$, we define the score as the *number of entailment-preserving combinations* of parses that include $S(p)_j$. If the parse contains a syntax error, the score -1 is assigned. If a sentence is included in multiple premise-hypothesis pairs, we sum the values obtained from all pairs.

For instance, consider the example in Figure 3. There are two entailment-preserving combinations annotated with curved arrows. As the second output from *Eruption produces ash clouds.* (darkest orange) is included in both combinations, the score of this output is 2. For other outputs included once (orange), the score 1 is assigned, and outputs that are not included in any combination (lightest orange) get a zero score. Finally, ones that have a syntax error (gray) are assigned a score of -1.

3.2.2 Learning-to-rank

Once we obtain a numeric score for each sampled output, theoretically, any feedback-based learning objective (Shen et al., 2016; Schulman et al., 2017; Lee et al., 2023) can be applied to maximize the score. In this work, we specifically use BRIO (Liu et al., 2022), a learning-to-rank training objective originally introduced for abstractive summarization models.

Intuitively, as shown in Figure 3, BRIO tries to increase the average token probability of outputs with higher scores compared to ones with lower scores for each row (same input). Formally, the BRIO training objective $\mathcal{L}_{BRIO}$ is defined as:

$$\mathcal{L}_{BRIO} = \sum_i \sum_j max(\hat{p}(y_j|x) - \hat{p}(y_i|x) + \Delta(j-i), 0)$$

where $1 \leq i < j \leq K$ denote the indices of outputs y sorted in descending order of the scoring function (y_1 having the highest score), $\hat{p}(y|x)$ is

Dataset	Train	Valid	Test	Prem.
EntailmentBank	2,486	276	408	2.11
eQASC	8,134	926	920	2.00
e-SNLI	100,000	9,842	9,824	2.00

Table 1: Dataset statistics. **Train**, **Valid**, and **Test** columns denote the number of premises-hypothesis pairs in each data split. **Prem.** is the average count of premises included in each premises-hypothesis pair.

the token log-probability normalized by sequence length, and Δ is the *margin* hyperparameter.

Finally, the plain cross-entropy loss $\mathcal{L}_{CE}$ is added to prevent the model from losing original generation capability, resulting in the final loss function $\mathcal{L} = \mathcal{L}_{CE} + \lambda\mathcal{L}_{BRIO}$ where λ is a *mixing rate* hyperparameter. The details of training hyperparameters are included in Appendix B.

3.2.3 Iterative training

Iterative training, which repeats the process of sampling, evaluation, and training, is widely recognized for enhancing performance across various scenarios by enabling the model to deviate further from the original fine-tuned model (Pang et al., 2024; Xiong et al., 2024).

Initially, a base model S_0 is obtained by fine-tuning a sequence-to-sequence model on NL→FOL parallel corpus using only the cross-entropy objective. Then, S_0 generates outputs using the training set, which is then evaluated using the scoring function presented in 3.2.1. After that, a new model S_1 is trained on the outputs and scores obtained from S_0 using the BRIO loss. We repeat this iteration five times, resulting in six different models $S_{t=0..5}$.

4 Experimental settings

4.1 Datasets

Three representative multi-premise RTE datasets, namely EntailmentBank (Dalvi et al., 2021), eQASC (Jhamtani and Clark, 2020), and e-SNLI (Camburu et al., 2018), are used for the experiments. The statistics of each data set are briefly introduced in Table 1.

EntailmentBank provides *entailment trees* with simple scientific facts as nodes. We decompose the trees into subtrees of depth 1, where the leaf nodes are premises that collectively entail the hypothesis in the root node.

eQASC provides 2-hop explanations for a given hypothesis derived from QASC (Khot et al., 2020), a multiple-choice question dataset from the science domain.

e-SNLI extends the single-premise SNLI dataset (Bowman et al., 2015) by adding *explanations* to the original premise-hypothesis pairs. This can be viewed as the premise and explanation together entailing the hypothesis. Due to limited computation resources, we sample 100k premises-hypothesis pairs from the train set and use the original validation/test set without modification.

4.2 NL→FOL translator

We use a sequence-to-sequence model, T5-base (Raffel et al., 2020), as our NL→FOL translator backbone. To obtain an initial model S_0, we take 34k pairs of natural language sentences and their LLM-generated FOL representation from MALLS (Yang et al., 2023), convert them into the NLTK format (Bird et al., 2009) with a rule-based translator, and train the T5-base model with standard cross-entropy loss. We refer to this model (S_0) as `T5-Iter0`, and models obtained after the N-th iteration as `T5-iter`N. As the MALLS dataset does not explicitly control the arbitrariness in its FOL representations, the model is not likely to learn to assign consistent predicates to similar concepts. Therefore, `T5-Iter0` achieves a low EPR score despite being able to generate syntactically correct FOL representations (Table 2).

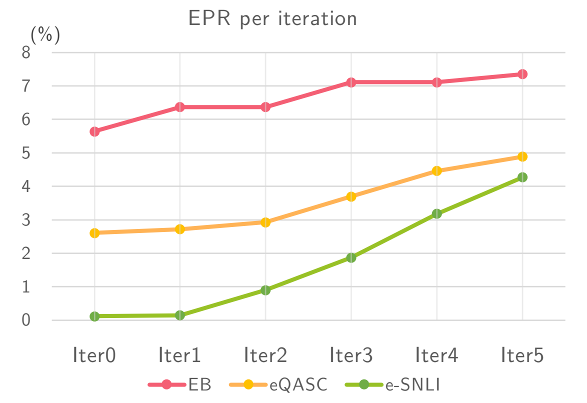
4.3 FOL Theorem prover

For the automatic theorem prover that is used to check entailment, we use Vampire (Kovács and Voronkov, 2013), one of the fastest provers currently available. As the generative models are trained on NLTK syntax, the model outputs are translated into Vampire-compatible format (TPTP (Sutcliffe, 2024)) using a rule-based translator.

4.4 Baselines

As a baseline, we adopt a wide variety of methods that translate natural language to first-order logic.

First, a suite of classic meaning representation-based methods was included as a baseline. `CCG2Lambda` (Martínez-Gómez et al., 2016) obtains Combinatory Categorical Grammar (CCG) parse trees using C&C Parser (Clark and Curran, 2007) and converts them to FOL representations via Lambda calculus. Bos (2016) and Lai et al. (2020) both convert Abstract Meaning Representations (AMR) to FOL. The AMR graph was obtained using AMRBART (Bai et al., 2022) and



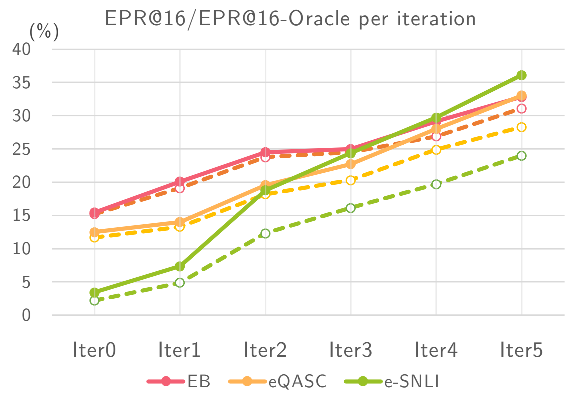


Figure 4: EPR (left), EPR@16 (right-solid), EPR@16-Oracle (right-dotted) per iteration. The continuous growth in all EPR metrics implies that the model extrapolated to unseen premises-conclusion pairs where BRIO loss is 0, demonstrating the strength of the proposed method.

translated to FOL representations using the respective implementations of rule-based translators. These methods are only evaluated by EPR and not by EPR@K(-Oracle) as they are designed to produce a single gold FOL parse for each sentence in a deterministic manner. If an error occurred during CCG/AMR parsing and their translation to FOL, we remove them from the evaluation pool (not counted in both the denominator and the numerator when calculating EPR).

Few-shot and fine-tuned LLMs were also evaluated as a baseline. First, we sample $K = 16$ FOL representations using GPT-4o and GPT-4o-mini (OpenAI, 2024) with 5 in-context examples temperature 1.0 (prompts shown in Appendix C). Also, we evaluate LogicLLaMA (Yang et al., 2023), a LLaMA-7B (Touvron et al., 2023) checkpoint directly fine-tuned on the MALLS dataset. $K = 16$ FOL representations per sentence were sampled using temperature 0.1, following the original paper.

5 Results

The results for EPF on three benchmarks are presented in Table 2.

The results show that classic meaning representations-based methods (`CCG2Lambda`, `AMR2FOL`) achieve extremely low EPR in multi-premise RTE datasets, consistently falling short under 0.1% EPR in EntailmentBank and eQASC. This extends the previous negative results in single-premise RTE datasets (Bos, 2014; Bos and Markert, 2006) to multi-premise RTE. On the other hand, end-to-end generative models (`GPT-4o`, `LogicLLaMA`, `T5-Iter0`) also demonstrate low

Metric	Method	EB	eQASC	e-SNLI
EPR	`CCG2Lambda`	0.0	0.0	0.0
	`AMR2FOL(Bos)`	0.0	0.0	2.5
	`AMR2FOL(Lai)`	0.0	0.0	1.6
	`GPT-4o-mini`	3.2	2.4	0.9
	`GPT-4o`	2.9	1.1	1.5
	`LogicLLaMA`	5.2	2.5	0.7
	`T5-Iter0`	5.6	2.6	0.1
	`T5-Iter5`	**7.4**	**4.9**	**4.3**
EPR@16	`GPT-4o-mini`	10.5	7.6	8.3
	`GPT-4o`	13.2	11.4	8.3
	`LogicLLaMA`	5.2	2.5	0.7
	`T5-Iter0`	15.4	12.5	3.4
	`T5-Iter5`	**32.8**	**33.1**	**36.1**
EPR@16 Oracle	`GPT-4o-mini`	10.5	7.4	5.6
	`GPT-4o`	13.0	10.8	5.6
	`LogicLLaMA`	5.2	2.5	0.7
	`T5-Iter0`	15.2	11.7	0.1
	`T5-Iter5`	**31.1**	**28.3**	**24.0**

Table 2: EPR, EPR@16, and EPR@16-Oracle measured on three different datasets (EntailmentBank (EB), eQASC, e-SNLI), single-run.

EPR score regardless of the model size or whether it is fine-tuned for NL→FOL translation or not. Although LLM-based generative methods have shown strong performance in nearly synthetic logical entailment tasks (Pan et al., 2023; Olausson et al., 2023), the results show that they do not generalize well to natural language entailment.

As shown in Figure 4, iterative learning-to-rank training can significantly increase the EPR score, resulting in +1.8-4.2p gain in EPR and +22.3-27.8p in EPR@16 after five iterations (`T5-Iter0`→`T5-Iter5`). The BRIO objective provides training signals only for inputs with differing output scores, such as when one output preserves entailment and another does not. The increase of

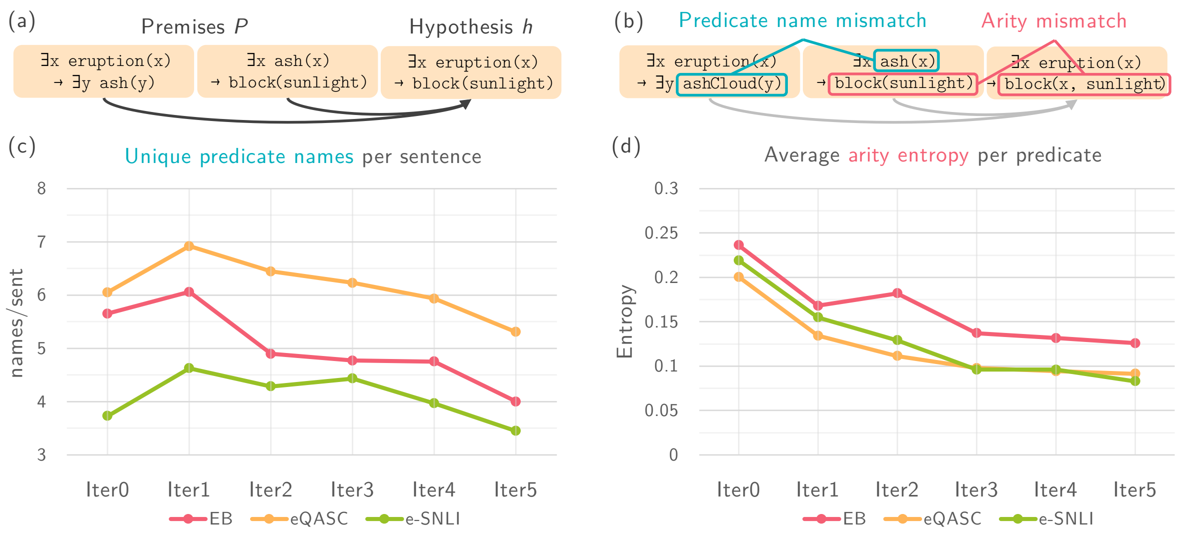


Figure 5: While the FOL premises in **(a)** can entail the hypothesis (from Figure 1), **(b)** cannot due to arbitrariness in predicate name and arity. During the iterative training, **(c)** the arbitrariness in predicate names decreases after the first iteration, and **(d)** the arity entropy is significantly reduced. These two results demonstrate that the proposed iterative learning-to-rank method can effectively reduce the arbitrariness of FOL parses.

the EPR@K score implies that outputs that previously had zero scores now preserve entailment and have positive scores, indicating that the model *extrapolated* to unseen cases. This demonstrates the effectiveness of the score function defined in Section 3.2.1 and the BRIO learning objective for EPF.

Notably, EPR@16-Oracle scores are significantly higher than EPR and close to the EPR@16 score with only a 1.7p difference in EntailmentBank[2], even though the EPR@16-Oracle score only uses a single FOL parse for each sentence. This implies that the gap between the current state-of-the-art EPR score and the *least upper bound* is large, leaving room for future improvement.

Examples of FOL representations sampled from `T5-Iter0` and `T5-Iter5` (before and after iterative learning-to-rank training) can be found in Appendix D.

6 Analysis

6.1 Arbitrariness

One aspect of arbitrariness is assigning inconsistent predicate signatures (name and arity (number of arguments)) for synonymous concepts. For instance, FOL representations in Figure 5(b) cannot entail the hypothesis because predicate names (`ash`↔`ashCloud`) and arities (`block` having 1 and 2 arguments) do not match, even though both representations are semantically plausible. We show that the proposed iterative learning-to-rank method effectively reduces arbitrariness in both predicate names and their arity, which explains the overall EPR gain.

6.1.1 Unique predicate names per sentence

Unique predicate names per sentence can be measured by counting all predicate names in the corpus and dividing by the number of NL sentences. If the number of unique predicates decreases, it implies that synonymous concepts are mapped to fewer predicates. For instance, unique predicate names per sentence are 1 in Figure 5(a), but 1.33 in Figure 5(b) due to `ashCloud` and `ash` being separated.

The results (Figure 5(b)) show that after the first iteration (`Iter1`), the number of unique predicate names constantly decreases in all datasets, indicating reduced arbitrariness.

6.1.2 Arity entropy

End-to-end generative models often fail to generate predicates with consistent *arity*. As the same predicates with different arities cannot lead to a successful proof, it is important to suppress such divergence.

To measure such variance, we adopt a new metric, *arity entropy*. For each predicate, the entropy of

[2]The EPR@16-Oracle scores are a conservative approximation of the *true* EPR@16-Oracle score due to NP-completeness of the MAX-SAT problem, which favors EntailmentBank that has the smallest test set.

Type	D/I	Train Prop.	# Test
Substitution (Sub)	D	42%	75
Inference from rule (IR)	D	33%	53
Further specification (FS)	I	15%	48
Property inheritance (PI)	I	4%	42
Infer class (IC)	I	4%	25
Sequential inference (SI)	D	3%	21

Table 3: Inference types observed from the Entailment-Bank dataset, from Dalvi et al. (2021). **D/I** column indicates if the inference type is deductive (D) or inductive (I). **Train Prop.** column denotes the proportion of the inference type found in the training set reported by the original paper, and **# Test** column is the number of test set examples annotated by the authors.

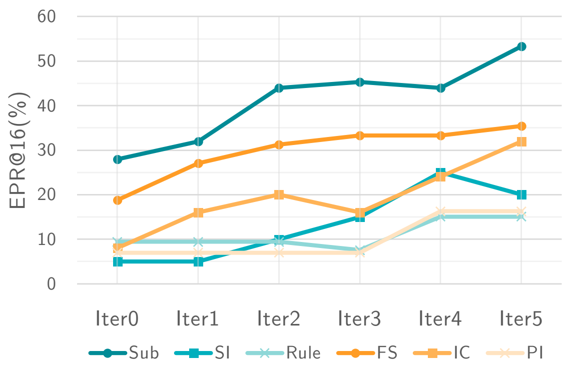


Figure 6: EPR@16 per iteration measured for each inference type in EntailmentBank. Deductive inference (*blue*) and inductive inference (*orange*) both achieve performance gain during the iterative training.

the probability distribution of a predicate's arities (*ArityEnt*), *i.e.*

$$\mathit{ArityEnt} = -\sum_{a=1}^{\max(a)} p(a)\log_2 p(a)$$

, was measured. Lower *ArityEnt* indicates that the model prefers a single arity for a given predicate throughout the entire dataset. For instance, `T5-Iter0` generates `CausesCycles()` predicate with 2 and 3 arguments 10 and 4 times respectively within the EntailmentBank dataset, resulting in *ArityEnt*=0.86. However, `T5-Iter5` only generates `CauseCycles()` with 2 arguments, having *ArityEnt*=0.

Figure 5(c) shows that during iterative training, the average *ArityEnt* of predicates decreases on all three datasets. The result also shows that our method can effectively reduce arbitrariness in order to preserve entailment.

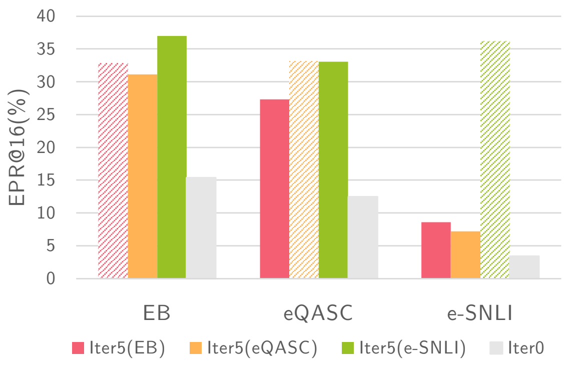


Figure 7: Out-of-domain EPR@16. A model trained on one dataset (*colored*) achieves better performance than the model trained solely on MALLS (*gray*) in other datasets. Hatched items refer to in-domain performance, where the trained and evaluated data are the same.

6.2 Inference types

To evaluate the robustness of the proposed method against diverse lexical and syntactic patterns, we analyze the EPR in various *inference types*. Dalvi et al. (2021) identified three *deductive*, three *inductive* inference types as introduced in the Entailment-Bank dataset (Table 3). Deductive inferences are often purely lexical and syntactic, as in *An animal is a kind of organism. A dog is a kind of animal.* ⊢ *A dog is a kind of organism* (Sub), while inductive inferences exhibit non-trivial logical structures, *e.g. Hunting is a kind of method for obtaining food. Animals require food for survival.* ⊢ *Some animals must hunt to survive* (IC). We manually labeled 264 examples from the EntailmentBank test set and measured the EPR@16 score for each reasoning template.

The results (Figure 6) demonstrate that our method consistently improves EPR@16 across all inference types, achieving gains ranging from 5.6p to 25.3p. This shows that our method is robust to the diversity of reasoning patterns (inductive or deductive), and low-resource settings where the proportion of a specific pattern in the training set is as low as 3%.

6.3 Out-of-domain generalization

Out-of-domain generalization is crucial for semantic parsers to cover diverse use cases not seen in the training dataset. We evaluate the out-of-domain generalization by evaluating a model trained on one dataset (*e.g.*, EntailmentBank) on another dataset (*e.g.*, eQASC).

The results are shown in Figure 7. Models trained for five iterations in any dataset consistently outperform `Iter0` (gray) in all out-of-domain settings. This implies that (1) different multi-premise RTE datasets share a similar logical structure and (2) the model was able to learn the common structure between multi-premise datasets using the BRIO objective.

The strong out-of-domain generalizability of `Iter5(e-SNLI)` and relatively weaker performance on e-SNLI of `Iter5(EB/eQASC)` shows that e-SNLI provides strong training signals for NL→FOL translator. As e-SNLI is the largest among the three datasets and includes diverse expressions originating from image descriptions, this highlights the importance of data quantity and diversity in data-driven approaches for the EPF task.

7 Conclusion

FOL representations provide an intuitive way to express logical entailment in natural language. However, using FOL for determining natural language entailment is a highly complex task due to arbitrariness and brittleness, where classic meaning representation-based NL→FOL parsers and end-to-end generative models both suffer.

In this study, we formalize the Entailment-Preserving FOL representations (EPF) task and reference-free metrics for EPF. Furthermore, we provide an effective method for training an end-to-end generative NL→FOL translator, *iterative learning-to-rank*, which significantly outperforms baselines specifically by reducing arbitrariness. These positive results shed light on a new data-driven approach for understanding natural language entailment with formal logic, addressing a long-standing challenge in computational linguistics.

8 Limitations

Scope of NL→FOL translation Whether it is plausible to use first-order logic to express natural language semantics has been a controversial topic in NLP. For instance, FOL might not be suitable for representing common linguistic phenomena, including uncertainty and implicature. Hence, the purpose of this paper is not to claim that FOL can preserve *all* natural language entailment, which is an interesting research direction but beyond the scope of this paper. Instead, the focus is on maximizing the EPR in a *subset* of natural language entailment that can be expressed by FOL, by minimizing undesired arbitrariness via RL training as shown in Section 6.1.

Gap between EPR and EPR@K-Oracle While our proposed method achieves the state-of-the-art Entailment Preservation Rate (EPR) in Entailment-Preserving FOL representation (EPF) task across multiple datasets and against a broad range of baselines, the gap between the EPR of the proposed approach and the *least upper bound* of EPF (EPR@K-Oracle) is still large. This highlights the significant potential for further advancements in data-driven approaches for NL→FOL semantic parsing, including the usage of more powerful models and larger training data with broader coverage.

Limited linguistic explainability The datasets employed in our study (EntailmentBank, eQASC, e-SNLI) lack linguistically controlled minimal pairs, which are instances designed to highlight subtle but crucial syntactic and semantic differences between sentences. This may cause the NL→FOL translator to overlook critical linguistic features for accurately expressing natural language entailment. While works that rely on explicit meaning representations (Bos, 2014; Lai et al., 2020) take account of syntactic and semantic information, our experimental results reveal that these methods struggle to capture natural entailment (Table 2). Balancing the empirical strengths of data-driven approaches with the explainability of linguistically grounded methods remains an open challenge for future work on the EPF task.

References

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Johan Bos. 2014. Is there a place for logic in recognizing textual entailment. *Linguistic Issues in Language Technology*, 9.

Johan Bos. 2016. Squib: Expressive power of Abstract Meaning Representations. *Computational Linguistics*, 42(3):527–535.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of Human Language Technology Conference* *and Conference on Empirical Methods in Natural Language Processing*, pages 628–635, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Johan Bos and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of the second PASCAL RTE challenge*, volume 26.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Jianpeng Cheng and Mirella Lapata. 2018. Weakly-supervised neural semantic parsing with a generative ranker. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 356–367, Brussels, Belgium. Association for Computational Linguistics.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2019. Learning an executable neural semantic parser. *Computational Linguistics*, 45(1):59–94.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370.

Dmitriy Dligach, Steven Bethard, Timothy Miller, and Guergana Savova. 2022. Exploring text representations for generative temporal relation extraction. In *Proceedings of the 4th Clinical Natural Language Processing Workshop*, pages 109–113, Seattle, WA. Association for Computational Linguistics.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2019. Multi-shot asp solving with clingo. *Theory and Practice of Logic Programming*, 19(1):27–82.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 2024. Folio: Natural language reasoning with first-order logic.

Harsh Jhamtani and Peter Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. *arXiv preprint arXiv:2010.03274*.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Laura Kovács and Andrei Voronkov. 2013. First-order theorem proving and vampire. In *Computer Aided Verification*, pages 1–35, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kenneth Lai, Lucia Donatelli, and James Pustejovsky. 2020. A continuation semantics for Abstract Meaning Representation. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 1–12, Barcelona Spain (online). Association for Computational Linguistics.

Youngwon Lee, Jinu Lee, and Seung-won Hwang. 2023. Learning to rank generation with pairwise partial rewards. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6078–6092, Singapore. Association for Computational Linguistics.

Vladimir Lifschitz. 2019. *Answer set programming*, volume 3. Springer Heidelberg.

Yixin Liu, Pengfei Liu, Dragomir Radev, and Graham Neubig. 2022. Brio: Bringing order to abstractive summarization. *arXiv preprint arXiv:2203.16804*.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings of ACL 2016 System Demonstrations*, pages 85–90, Berlin, Germany. Association for Computational Linguistics.

Richard Montague et al. 1970. Universal grammar. *1974*, pages 222–46.

Theo X Olausson, Alex Gu, Benjamin Lipkin, Cedegao E Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger Levy. 2023. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. *arXiv preprint arXiv:2310.15164*.

OpenAI. 2024. Gpt-4o system card.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization.

Terence Parsons. 1990. Events in the semantics of english: A study in subatomic semantics.

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. 2024. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2933–2958, Miami, Florida, USA. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. 2024. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

G. Sutcliffe. 2024. Stepping Stones in the TPTP World. In *Proceedings of the 12th International Joint Conference on Automated Reasoning*, number 14739 in Lecture Notes in Artificial Intelligence, pages 30–50.

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*, pages 3738–3747, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, Chi Jin, Tong Zhang, and Tianqi Liu. 2024. Building math agents with multi-turn iterative preference learning.

Kun Xu, Sheng Zhang, Yansong Feng, and Dongyan Zhao. 2014. Answering natural language questions via phrasal semantic parsing. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 333–344. Springer.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2023. Harnessing the power of large language models for natural language to first-order logic translation. *arXiv preprint arXiv:2305.15541*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

A Details on Entailment-Preserving Rate (EPR)

A.1 Spurious entailment detection

In first-order logic, a contradictory statement such as $P(a) \wedge \neg P(a)$ can derive any FOL formula. Therefore, the notion of entailment-preserving, $FOL(p_1), ..., FOL(p_N) \vdash FOL(h)$, is susceptible to *spurious entailments* where the hypothesis is derived only because there was a contradiction in the premises.

EPR evaluation imposes two verification steps for the external prover's results to filter out these spurious entailments and align the semantics of entailment-preserving to human instincts. First, the hypothesis must not introduce predicates and constants that were not present in the premises. Second, the automatically generated proof of the hypothesis must include all premises. These checks not only prevent trivial contradictions falsely entailing the hypothesis but also more complex cases, such as $P(c), \forall x(\neg Q(x) \rightarrow (R(x) \wedge \neg R(x))) \vdash Q(c)$ where $P(c)$ is not used to prove the conclusion because of an embedded contradiction in $\forall x(\neg Q(x) \rightarrow (R(x) \wedge \neg R(x)))$.

A.2 EPR@K-Oracle

EPR@K-Oracle score is calculated by selecting one from the K parses of each sentence that maximizes the overall EPR score. If all sentences participate only in a single premises-hypothesis pair, this score will be identical to EPR@K. However, as some sentences participate in multiple premises-hypothesis pairs and an FOL representation that preserves entailment in one pair might not in another pair (Figure 2, third sentence from the top), the EPR@K-Oracle score is generally lower than EPR@K.

This problem of finding the EPR@K-Oracle score is an instance of the constraint satisfaction problem that can be reduced to the maximum satisfiability problem, which is NP-complete. The problem can be formulated in three conditions:

- For all boolean predicates $\texttt{fol}(i, j)$ that represent $S(p_i)_j$, only one j from each i can be selected. This condition can be expressed as $\forall i\ \exists! j\ \texttt{select}(i, j)$.

- If a $(j_1, ..., j_N, j)$ tuple satisfies the b-th premises-hypothesis pair in the dataset, *i.e.* $S(p_1)_{j_1}, ..., S(p_N)_{j_N} \vdash S(h)_j$, $\texttt{select}(1, j_1) \wedge ... \wedge \texttt{select}(N, j_N) \wedge \texttt{select}(h, j)$ implies $\texttt{success}(b)$.

- The goal is to maximize the number of different `success(b)` that are true.

To solve this constraint optimization problem, we use Answer Set Programming (Lifschitz, 2019) to formulate the problem and run an ASP solver, Clingo (Gebser et al., 2019). We impose a 600-second time limit[3] for each constraint satisfaction problem instance, and use the maximum number of $\texttt{success}(b)$ returned from the solver to calculate EPR@K-Oracle.

B Training hyperparameters

Table 4 shows the hyperparameters used for bootstrapping learning-to-rank training proposed in this work. BRIO hyperparameters were obtained from the grid search $\Delta = 0, 0.01, 0.1$ and $\lambda = 1, 10, 100$ using the performance after a single iteration (*i.e.* `T5-Iter1`) in EntailmentBank. After training for 20 epochs with BRIO loss, the checkpoint with the minimum validation loss was selected.

BRIO Loss	
Δ	0.01
λ	10
Training	
Batch size	16
Optimizer	Adam
Learning rate (LR)	1e-5
LR Scheduler	Cosine annealing
Gradient clipping	5.0
Epoch	20

Table 4: Training hyperparameters.

C Prompts for GPT-4o(-mini)

Figure 8 describes the prompt used for `GPT-4o` and `GPT-4o-mini` baseline experiments. For both models, Five few-shot examples were used to perform in-context learning.

D Examples

Table 5 shows the examples sampled from three datasets (EntailmentBank, eQASC, and e-SNLI), and shows how the FOL representations changed from `T5-Iter0` to `T5-Iter5`.

First, it is noticeable that the entailment-preserving FOL representations have more *atomic*

[3]Clingo was executed on a single core of AMD EPYC-Milan. Search strategies and heuristics were set as default.

System prompt

You will see a natural language sentence. Translate it into first-order logic (FOL).
You MUST use a common set of predicates to represent the meaning in FOL format.
Below are instructions for the format of FOL logical formulas:
1. **Variables**: Use lowercase (x, y, etc.) for generic objects.
2. **Constants**: Use lowercase names (john, sun) for specific entities.
3. **Predicates**: Represent properties/relations as `Predicate(arg1, arg2)`, e.g., `Rises(sun)`, `Loves(john, mary)`.
4. **Connectives**:
- **Negation (-)**: Not, e.g., `-Rains(x)`
- **Conjunction (&)**: And, e.g., `Walks(john) & Talks(john)`
- **Disjunction (|)**: Or, e.g., `Walks(john) | Talks(john)`
- **Implication (->)**: If...then, e.g., `Rains(x) -> Wet(x)`
- **Biconditional (<->)**: If and only if, e.g., `Rains(x) <-> Wet(x)`
5. **Quantifiers**:
- **Universal (`all`)**: For all, e.g., `all x. (Human(x) » Mortal(x))`
- **Existential (`exists`)**: There exists, e.g., ‘exists x. (Human(x) & Smart(x))

User prompt (5-shot)

Few-shot Example 1:
Sentence: All planets orbit a star.
FOL Translation: `all x. (Planet(x) -> exists y. (Star(y) & Orbits(x, y)))`
Few-shot Example 2:
Sentence: Mars is a planet.
FOL Translation: `Planet(mars)`
Few-shot Example 3:
Sentence: If a person is a scientist and has access to a laboratory, they can conduct experiments.
FOL Translation: `all x. ((Scientist(x) & HasAccessToLab(x)) -> CanConductExperiments(x))`
Few-shot Example 4:
Sentence: Butterflies are insects.
FOL Translation: `all y. (Butterfly(y) -> Insect(y))`
Few-shot Example 5:
Sentence: All insects that have wings can fly.
FOL Translation: `all x. (Insect(x) & HasWings(x) -> CanFly(x))`

Figure 8: System and user prompts used for GPT-4o and GPT-4o-mini.

predicate names (*e.g.* `GlowingBand`), compared to complex predicate names generated by `T5-Iter0` (`Appears...NightSky`). This corresponds to the analyses in Section 6.1, where the bootstrapping reduces the lexical diversity of the predicate names.

Furthermore, example **e-SNLI_6529** shows that a new predicate that does not lexically match the original NL premise, Has(x, y), was introduced by the model. This shows the potential of our method to overcome the *brittleness* of FOL parses, where FOL parses omit underlying commonsense or pragmatic assumptions required for RTE tasks (Bos, 2014), by data-driven approaches in the EPF task.

E License

The datasets (EntailmentBank: Apache 2.0, eQASC: CC-BY-4.0, e-SNLI: MIT, MALLS: CC-BY-NC-4.0), models (T5: Apache 2.0), and theorem prover (Vampire: BSD 3-Clause) can be freely used for non-commercial academic purposes.

EntailmentBank_26		
NL	p_1	The Milky Way galaxy appears as a glowing band across the night sky.
	p_2	A student observes a glowing band across the night sky.
	h	The student observes the Milky Way galaxy at night.
Iter0	p_1	$\forall x.(\text{MilkyWayGalaxy}(x) \rightarrow \text{AppearsAsGlowingBandAcrossNightSky}(x))$
	p_2	$\forall xy.((\text{Student}(x) \land \text{Student}(y)) \rightarrow \text{ObservesGlowingBand}(x,y))$
	h	$\forall xy.((\text{Student}(x) \land \text{MilkyWayGalaxy}(y)) \rightarrow \text{Observes}(x,y))$
Iter5	p_1	$\forall x.(\text{MilkyWayGalaxy}(x) \rightarrow (\text{GlowingBand}(x) \land \text{NightSky}(x)))$
	p_2	$\forall xyz.((\text{Student}(x) \land \text{GlowingBand}(y) \land \text{NightSky}(z)) \rightarrow \text{Observes}(x,y))$
	h	$\forall xy.((\text{Student}(x) \land \text{MilkyWayGalaxy}(y)) \rightarrow \text{Observes}(x,y))$
eQASC_536		
NL	p_1	Mammalian teeth are important for digestion.
	p_2	Teeth help aid the digestive system in breaking down food.
	h	Mammalian teeth are important for breaking down food.
Iter0	p_1	$\forall x.(\text{MammalTeeth}(x) \rightarrow \text{ImportantForDigestion}(x))$
	p_2	$\forall x.(\text{Teeth}(x) \rightarrow \text{AidsDigestiveSystem}(x))$
	h	$\forall x.(\text{MammalTeeth}(x) \rightarrow \text{BreaksDownFood}(x))$
Iter5	p_1	$\forall x.(\text{Mammal}(x) \rightarrow (\text{Teeth}(x) \land \text{Digestive}(x)))$
	p_2	$\forall x.(\text{Teeth}(x) \rightarrow (\text{AidsDigestiveSystem}(x) \land \text{BreaksDownFood}(x)))$
	h	$\forall x.(\text{Mammal}(x) \rightarrow (\text{Teeth}(x) \land \text{BreaksDownFood}(x)))$
e-SNLI_6529		
NL	p_1	A woman with red-hair and a leather jacket holds a harp in front of a snowy forest.
	p_2	A harp is an instrument.
	h	The woman is holding an instrument.
Iter0	p_1	$\forall xy.((\text{Female}(x) \land \text{RedHair}(x) \land \text{LeatherJacket}(x) \land \text{SnowyForest}(y)) \rightarrow \text{HoldsHarpInFrontOf}(x,y))$
	p_2	$\forall x.(\text{Harp}(x) \rightarrow \text{Instrument}(x))$
	h	$\forall xy.((\text{Female}(x) \land \text{Instrument}(y)) \rightarrow \text{Holds}(x,y))$
Iter5	p_1	$\exists xy.(\text{Female}(x) \land \text{RedHair}(x) \land \text{LeatherJacket}(x) \land \text{Harp}(y) \land \text{SnowyForest}(z) \land \text{Has}(x,y) \land \text{Holds}(x,y))$
	p_2	$\forall x.(\text{Harp}(x) \rightarrow \text{Instrument}(x))$
	h	$\exists xy.(\text{Female}(x) \land \text{Instrument}(y) \land \text{Holds}(x,y))$

Table 5: Premises-hypothesis pairs sampled from EntailmentBank, eQASC, and e-SNLI datasets, where the entailment is not preserved in `T5-Iter0` but preserved in `T5-Iter5` in EPR@16-Oracle setting.