Autoformalizing Natural Language to First-Order Logic: A Case Study in Logical Fallacy Detection

Abhinav Lalwani[1*] **Tasha Kim**[1*] **Lovish Chopra**[1*]
Christopher Hahn[2 †] **Zhijing Jin**[3,4,5,‡] **Mrinmaya Sachan**[4,‡]
[1]Stanford University [2]X, the moonshot factory
[3]Max Planck Institute for Intelligent Systems [4]ETH Zürich [5]University of Toronto
{lalwani,tashakim,lovish}@stanford.edu {jinzhi,msachan}@ethz.ch

Abstract

Translating natural language into formal language such as First-Order Logic (FOL) is a foundational challenge in NLP with wide-ranging applications in automated reasoning, misinformation tracking, and knowledge validation. In this paper, we introduce Natural Language to First-Order Logic (NL2FOL), a framework to autoformalize natural language to FOL step-by-step using Large Language Models (LLMs). Our approach addresses key challenges in this translation process, including the integration of implicit background knowledge. By leveraging structured representations generated by NL2FOL, we use Satisfiability Modulo Theory (SMT) solvers to reason about the logical validity of natural language statements. We present logical fallacy detection as a case study to evaluate the efficacy of NL2FOL. Being neurosymbolic, our approach also provides interpretable insights into the reasoning process and demonstrates robustness without requiring model fine-tuning or labeled training data. Our framework achieves strong performance on multiple datasets – on the LOGIC dataset, NL2FOL achieves an F1-score of 78%, while generalizing effectively to the LOGICCLIMATE dataset with an F1-score of 80%.[1]

1 Introduction

In recent years, Large Language Models (LLMs) have shown impressive advancements in understanding and generating natural language (Brown et al., 2020). Despite this progress, their ability to tackle complex reasoning tasks remains limited (Bubeck et al., 2023; Wei et al., 2022). These challenges are especially prevalent in multistep logical deductions, abstract reasoning, and knowledge integration in various domains (Dalvi et al., 2021; Chen et al., 2024). Addressing these limitations and improving the reasoning capabilities of LLMs has become a critical focus in AI research (Haluptzok et al., 2022; Gendron et al., 2024).

In contrast, formal reasoning tools such as Satisfiability Modulo Theory (SMT) solvers excel in reasoning, providing rigorous, provable guarantees by leveraging symbolic representations and logical calculus (Barrett et al., 2009; De Moura and Bjørner, 2008). However, a key limitation of formal solvers is their reliance on structured logical input, such as First Order Logic (FOL), which must accurately capture the semantics and context of natural language statements (Beltagy et al., 2016). This presents the challenge of translating unstructured natural language into a structured form required for formal reasoning while preserving essential context and meaning.

This also brings a unique opportunity: if we can reliably translate natural language into structured logical forms, we can harness the power of formal solvers to reason systematically over natural language statements. However, achieving this translation is nontrivial, as it involves accurately capturing natural language semantics (Beltagy et al., 2016). Moreover, translating to a formal logical form may cause implicit and external context to be lost, which must be reintroduced to ensure logical accuracy.

To address these challenges, we present NL2FOL, a novel framework that bridges the gap between natural language and formal reasoning systems. NL2FOL employs a structured, step-by-step pipeline to translate natural language inputs into first-order logic (FOL) representations, leveraging large language models (LLMs) at each step for enhanced precision and adaptability. A distinguishing feature of NL2FOL is its seamless integration of background knowledge into the generated logical forms, overcoming a major limitation of traditional formal logic frameworks - the inability to capture

[*]Equal contribution
[†]Work done while at Stanford University
[‡]Co-supervision
[1]Code available at: github.com/lovishchopra/NL2FOL

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 132–147

Fallacy Name	Example	Logical Form
Faulty Generalization	Sometimes flu vaccines don't work; therefore vaccines are useless.	$(\exists x \in \text{FluVaccines}(\text{DoesntWork}(x)) \wedge (\text{FluVaccines} \subseteq \text{Vaccines})) \Rightarrow (\forall y \in \text{Vaccines}\ (\text{DoesntWork}(y)))$
False Causality	Every time I wash my car, it rains. Me washing my car has a definite effect on the weather.	$\text{occuredAfter}(\text{washingCar}, \text{rain}) \Rightarrow \text{caused}(\text{washingCar}, \text{rain})$
Ad Populum	Everyone should like coffee: 95% of teachers do!	$(\text{like}(\text{coffee}, 95\%\text{Teachers})) \Rightarrow (\text{like}(\text{coffee}, \text{everyone}))$
False Dilemma	I don't want to give up my car, so I don't think I can support fighting climate change.	$\forall(a)(\text{giveUpCar}(a) \vee \text{dontSupportFightingClimateChange}(a))$

Table 1: Sample logical fallacies from Jin et al. (2022) along with examples and their logical forms. For each type of fallacy, we show one possible logical form.

implicit information embedded in natural language.

In this paper, we demonstrate the effectiveness of NL2FOL through a case study on logical fallacy detection, showcasing its ability to identify and explain faulty reasoning in natural language arguments. Detecting logical fallacies is particularly challenging as they often rely on reasoning patterns that appear plausible yet are fundamentally flawed (Jin et al., 2022). To address this, NL2FOL translates logical fallacies from natural language into FOL representations, enabling formal solvers to verify logical validity. These solvers generate counterexamples and explanations, which are interpreted back into natural language to enhance human comprehensibility. By incorporating intermediate natural language outputs, our pipeline improves interpretability, transparency, and debuggability (?).

We show that our framework achieves strong performance on the logical fallacy detection benchmarks LOGIC and LOGICCLIMATE (Jin et al., 2022), with F1 scores of 78% and 80%, respectively - outperforming existing models by 22% on the challenge set, LOGICCLIMATE. These results highlight NL2FOL as a generalizable and interpretable tool for reasoning tasks that demand the precision of formal reasoning systems. By analyzing the strengths and weaknesses of LLMs at each step of the NL2FOL pipeline, we further identify opportunities for improving logical reasoning capabilities. Even though LLMs prove to be effective in parsing and generating logical representations for structured inputs, they often struggle with ambiguities in natural language and incorporating nuanced contextual knowledge. The ability to integrate symbolic solvers with language models positions NL2FOL as a powerful neurosymbolic approach, bridging the gap between formal reasoning and natural language understanding.

2 Related Work

Logical fallacy detection. Existing work on classifying logical fallacies includes argument sufficiency classification (Stab and Gurevych, 2017), ad hominem fallacies from Reddit posts (Habernal et al., 2018b) and dialogues (Habernal et al., 2018a), rule parsers (Nakpih and Santini, 2020), structure-aware Transformers (Jin et al., 2022), multitask instruction based prompting (Alhindi et al., 2022), and instance-based reasoning (Sourati et al., 2022). To our knowledge, our work is the first on few-shot classification of logical fallacies in a step-by-step, explainable manner. By ensuring that the reasoning process is transparent, we allow users to understand and verify the system decision.

Natural language to formal logic. While early work on mapping text to formal logic relied heavily on grammar-based approaches (Purdy, 1991; Angeli and Manning, 2014; MacCartney and Manning, 2014), recent advances in deep learning and foundation models have enabled new data-driven techniques for translating natural language to linear temporal logic (Cosler et al., 2023; Fuggitti and Chakraborti, 2023; Liu et al., 2022) and first-order logic (Singh et al., 2020; Yang et al., 2024; Hahn et al., 2022). Neural models for parsing natural language to first-order logic (Singh et al., 2020; Yang et al., 2024) and neuro-symbolic approach combining language models with first-order logic provers (Olausson et al., 2023) have since been explored. However, these approaches still face challenges in accurately capturing implicit information or transforming complex ambiguous sentences into logical form, mainly attributed to linguistic ambiguity.

Aly et al. (2023) integrated LLMs with logical inference for fact verification, and while our method

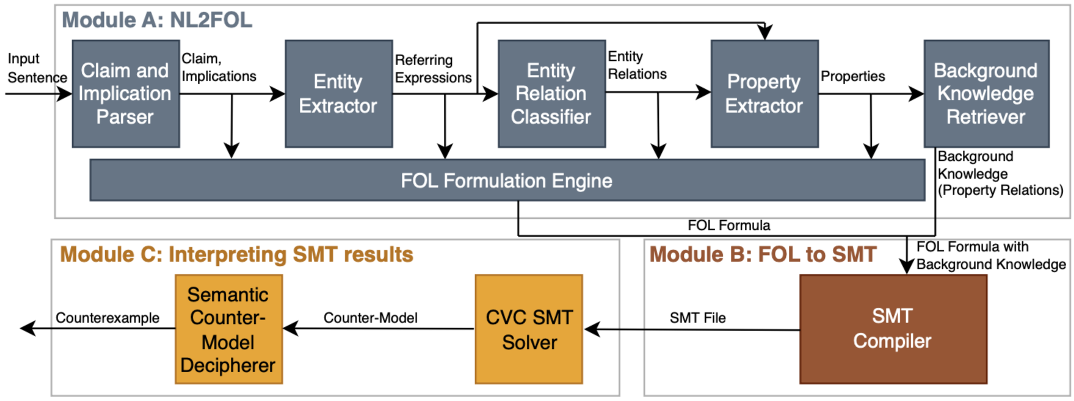

Figure 1: Overview of the proposed framework used for logical fallacy detection. *Module A* converts natural language input to a first-order logic formula merged with contextual relationships, *Module B* compiles the negation of a given logical formula to an SMT file with well-defined sorts for variables and predicates, and *Module C* runs CVC on the SMT file and if the negation is satisfiable, interprets the counter-model in natural language.

shares the fundamental idea of employing LLMs to construct proofs and analyze relationships between textual spans, our task adds a layer of contextual reasoning by requiring the incorporation of background knowledge and maintaining interdependency between proof steps, which is not present in approaches where each proof step is treated as an independent, isolated process.

Theory solvers. Recent work by Hahn et al. (2022) demonstrated the potential of integrating symbolic solvers with large language models (LLMs), such as tool-augmented LLMs, to combine neural and symbolic reasoning. While such approaches are promising, they often struggle to translate natural language into symbolic representations and effectively capture background knowledge. Other recent approaches (Olausson et al., 2023; Pan et al., 2023) have used theory solvers to logically reason with natural language, which we build on with several key advancements. First, we introduce a framework that handles naturalistic, real-world data and tasks with ambiguous premises and conclusions. Then, we present a method to incorporate background knowledge into logical formulas. Finally, we show that our approach introduces interpretability by allowing human verification and modification throughout the intermediate reasoning steps.

3 Methodology

Although powerful, LLMs struggle to detect logical fallacies in language, as it requires proper logical analysis (Jin et al., 2022). On the other hand, SMT solvers can reason over logical formulas with theoretical guarantees but require the input to be in a structured, logical form. This approach combines the strengths of both to classify logical fallacies.

Task formulation. The task input is an argument in natural language comprising one or more sentences, which is converted into formal logical form using a chain of LLMs. Following this, an SMT solver processes the logical form and returns whether it is valid. If invalid, the SMT solver provides a counterexample explaining why it is a logical fallacy, which is then interpreted with an LLM.

First-order logic. In FOL, propositions are represented using predicates that express properties or relations over objects in a domain. These predicates can be combined with constants, representing specific objects and variables that represent unspecified elements in the domain. An $\mathrm{Interpretation}$ assigns meaning to these symbols within a given context, while a Sort categorizes objects into different types, facilitating precise reasoning about their properties. Logical connectives of FOL, such as implication ($\Rightarrow$), universal quantifiers ($\forall$), existential quantifiers ($\exists$), and operators for conjunction/and ($\land$), disjunction/or ($\lor$), and negation/not ($\neg$), allow for the construction of intricate statements.

Module A: Natural language to first-order logic. Our approach for converting given natural language sentences into a logical form comprises multiple steps involving few-shot prompting of LLMs: (i) decomposing a sentence into multiple smaller parts that can be represented in first-order logic, (ii) identifying relationships between different subcomponents to merge them and obtain a resultant logical formula, and (iii) identifying real-world relationships between these sub-components (background knowledge) and augmenting them to obtain a FOL formula by incorporating background knowledge in the statement. We demonstrate with a Logical Fallacy (LF) and a Valid (V) example.

> **1. LF Example: A Logical Fallacy Input**
> I met a tall man who loved to eat cheese, now I believe all tall people like cheese.
>
> **2. V Example: A Valid Input**
> A boy is jumping on a skateboard in the middle of a red bridge. Thus the boy does a skateboarding trick.

Our pipeline begins with a semantic decomposition module which decomposes natural language arguments into respective claims and implications. Generally, a sentence can be split into some claims and implications based on those claims (see Prompt 2).

> **1. LF Example: Claim and Implication Parser**
> Claim: A tall man loved to eat cheese.
> Implication: All tall people like cheese.
>
> **2. V Example: Claim and Implication Parser**
> Claim: A boy is jumping on a skateboard in the middle of a red bridge.
> Implication: The boy does a skateboarding trick.

The claims and implications are split into further sub-components and used to build up the logical form of the sentence. The next step is to identify entities in the sentence. In our work, we treat noun phrases or surrogates for noun phrases as entities (see Prompt 3). Then, we find the relationship between the different entities using Zero-Shot classification via Natural Language Inference (NLI). These relationships (e.g., subset, equality, not related) are generally helpful in deciding appropriate quantifiers in the logical form. For example, if the entities are *man* and *people*, then it can be inferred that *man* is a subset of *people* and that the man would be bound by an existential quantifier in the sentence *x* (see Prompt 4).

> **1. LF Example: Entity Extractor**
> Referring expressions:
> - man: x
> - cheese: c
> - people: y
> - $x \subseteq y$
>
> **2. V Example: Entity Extractor**
> Referring expressions:
> - boy: b
> - skateboard: s
> - bridge
> - skateboardingTrick: y

The other set of sub-components are properties, which describe a trait of a referring expression or relationship between multiple referring expressions. These properties are predicates in first-order logic. We use a single module to extract the properties and the relation between properties and entities. (see Prompt 5). We also find the relationships between various properties (see Prompt 6). For instance, in the LF Example, it can be inferred that *Like* and *Love* are contextually similar. Similarly, in our valid example, *jumping over skateboard* implies *doing a skateboard trick*. These relationships provide an additional context that is not directly present in the statement.

To identify these contextual relationships, we run NLI between each pair of properties, i.e., by setting one property as the hypothesis and the other as the premise as the input to the NLI model. If we find that any one property entails the other, we add the relationship $\text{property1} \Rightarrow \text{property2}$ to our context. Before running the NLI model between a pair of properties, we replace the variables in each property with the referring expressions that they represent. This adds additional context that helps the NLI model identify relations. For instance, in the V Example, the NLI model is unable to find the relation between *JumpsOn(x, s)* and *Does(x, y)*, but it can identify the relationship between *JumpsOn(boy, skateboard)* and *Does(boy, skateboardingTrick)*.

> **1. LF Example: Property Extractor + Background Knowledge Retriever**
> Properties: Tall, Love, Like
> Property entity relations: $\text{Tall}(x), \text{Love}(x, c)$
> Background knowledge:
> 1. $\forall x(\text{Like}(x, c) \Rightarrow \text{Love}(x, c))$
> 2. $\forall x(\text{Love}(x, c) \Rightarrow \text{Like}(x, c))$
> 3. $x \subseteq y$
>
> **2. V Example: Property Extractor + Background Knowledge Retriever**
> Properties: JumpsOn, inMiddleOf, Red, Does
> Property entity relations: $\text{JumpsOn}(b, s)$, $\text{Red}(\text{bridge}), \text{inMiddleOf}(b, \text{bridge}), \text{Does}(b, y)$
> Background knowledge:
> 1. $\forall x(\text{JumpsOn}(b, s) \Rightarrow \text{Does}(b, y))$

Finally, we combine all of this information using the relationships between properties and entities to obtain the FOL form of the sentence with the help of an LLM (see Prompt 7). For a logical fallacy, the negation of the formula is expected to be satisfiable. On the contrary, for a valid statement, the negation of the formula should be unsatisfiable.

> **1. LF Example: NL2FOL Output**
> First-order logic: $((\forall x(\text{Like}(x, c) \Rightarrow \text{Love}(x, c))\wedge (\forall x(\text{Love}(x, c) \Rightarrow \text{Like}(x, c)))\wedge (\exists x(\text{Tall}(x) \wedge \text{Love}(x, c)))) \Rightarrow (\forall y(\text{Tall}(y) \Rightarrow \text{Like}(y, c)))$
>
> **2. V Example: NL2FOL Output**
> First-order logic: $(\forall x(\text{JumpsOn}(x, s) \Rightarrow \text{Does}(x, y)) \wedge \text{Red}(\text{bridge})\wedge \text{inMiddleOf}(b, \text{bridge}) \wedge \text{JumpsOn}(b, s)) \Rightarrow \text{Does}(b, y)$

Module B: First-order logic to SMT. The next step involves automatically creating an SMT file for the negation of the first-order logical formula generated. While one can easily write an SMT file for a logical formula manually, generating one automatically for an arbitrary formula has not been done before. Thus, we develop a compiler that parses a given logical formula and converts it into an SMT file that can be given to CVC as input, as described in Algorithm 1 (See Appendix).

Module C: Interpreting SMT results. To verify the validity of the logical formulas, we utilize an SMT solver, CVC4 (Barrett et al., 2011). The solver determines whether the formula is valid or invalid, hence a logical fallacy. In the case of invalidity, the model provides a counterexample to the original logical formula, which shows that the given claim or implication is a logical fallacy.

Example (Module B Output):
I met a tall man who loved to eat cheese, now I believe all tall people like cheese.

↓

First-order logic: $((\forall x(\text{Like}(x, c) \Rightarrow \text{Love}(x, c)))\wedge (\forall x(\text{Love}(x, c) \Rightarrow \text{Like}(x, c)))\wedge (\exists x(\text{Tall}(x) \wedge \text{Love}(x, c)))) \Rightarrow (\forall y(\text{Tall}(y) \Rightarrow \text{Like}(y, c)))$

↓

SMT classification: Logical fallacy
Explanation: Counterexample

↓

- John is tall (Tall(John) is True). John likes cheese (Likes(John, Cheese) is True).
- Jane is tall (Tall(Jane) is True). No constraint Jane likes cheese.

Therefore, there exists a tall person (John) who likes cheese, but it does not follow that all tall people like cheese, since Jane serves as a counterexample.

Figure 2: Example of logical fallacy detection using NL2FOL. The resulting classification is explained using a counterexample generated by the SMT solver.

The result of the SMT solver is hard to interpret, as it uses technical terminology generally only well understood by those who are familiar with CVC4 and SMT. To obtain an explanation in natural language, we prompt an LLM with the claim, implication, referring expressions, properties, FOL formula, and the counterexample generated by CVC4. The model then interprets the counterexample with natural language, as depicted in Figure 2.

4 Experiments

We evaluate our approach on both logical fallacies (positive class) and valid statements (negative class). For logical fallacies, we use the LOGIC and LOGICCLIMATE (Jin et al., 2022) datasets, originally designed for training models to identify and classify different fallacies. These datasets contain examples of logical fallacies, each labeled with multiple categories from 13 different categories, including faulty generalization, circular claim, and ad hominem. The LOGIC dataset contains 2,449 examples of common logical fallacies collected mostly from quiz websites. The LOGICCLIMATE dataset comprises 1,079 examples of logical fallacies drawn from climate change news articles on the Climate Feedback platform. It is intended to test the model's ability to generalize out-of-domain.

To test our approach with valid statements, we use the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015), which supports the development of natural language inference systems. This dataset features over 570,000 human-annotated sentence pairs, where each pair consists of a premise and a hypothesis labeled as entailment, contradiction, or neutral. We focus on the entailment class in this study, extracting over 170,000 sentence pairs where the premise entails the hypothesis. We construct valid sentences by combining the premise and hypothesis into a single sentence.

The task is set up as a simple binary classification task, where the input consists of sentences drawn from the LOGIC or LOGICCLIMATE datasets labeled as logical fallacies or from the SNLI dataset labeled as valid sentences. Here, we treat logical fallacies as the positive class. To ensure a balanced evaluation, we select an equal number of fallacies and valid statements, allowing for a fair comparison across both classes. Finally, our model is evaluated on standard binary classification metrics such as precision, recall, f1 score, and accuracy.

Models. We compare our method to pretrained language models, including Llama2-7B (Touvron et al., 2023), GPT4o-mini (OpenAI, 2024), GPT4o (OpenAI et al., 2024a) and OpenAI o1-preview (OpenAI et al., 2024b) with few-shot in-context examples (see Prompt 1). We also run NL2FOL with each of the above models used for the LLM prompting stages. Llama2-7B was chosen for our experiments as it had the best performance during testing over an initial subset of the data, outperforming Llama3.1-8B (Grattafiori et al., 2024), Llama3.2-11B (AI, 2024a), and Ministral-8B (AI, 2024b). We evaluate BART (140M parameters) (Lewis et al., 2020) finetuned on MNLI (Williams

Model	Method	Logic Acc.	Logic P.	Logic R.	Logic F1	LogicClimate Acc.	LogicClimate P.	LogicClimate R.	LogicClimate F1
Llama-7B	End-to-end	0.41	0.45	0.82	0.58	0.31	0.38	0.62	0.47
	NL2FOL (Ours)	**0.63**	**0.58**	**0.92**	**0.71**	**0.66**	**0.60**	**0.94**	**0.73**
GPT-4o-mini	End-to-end	**0.91**	**0.94**	0.88	**0.91**	0.64	**0.67**	0.55	0.60
	NL2FOL (Ours)	0.70	0.64	**0.91**	0.75	**0.73**	0.66	**0.93**	**0.77**
GPT-4o	End-to-end	**0.96**	**0.96**	**0.96**	**0.96**	0.70	**0.95**	0.42	0.58
	NL2FOL (Ours)	0.78	0.76	0.82	0.78	**0.80**	0.80	**0.80**	**0.80**
OpenAI o1-preview	End-to-end	0.93	0.89	0.98	0.93	0.73	0.84	0.56	0.67
	NL2FOL (Ours)	-	-	-	-	-	-	-	-

Table 2: Comparison of few-shot model performance metrics (abbreviations: Acc. = accuracy, P. = precision, R. = recall, F1 = F1 score) on the Logic+SNLI and LogicClimate+SNLI datasets using End-to-end vs. NL2FOL (Ours). Results on NL2FOL with o1-preview are omitted as o1-preview failed to complete the pipeline in most cases, likely due to its poor instruction following capabilities.

et al., 2018) to analyze the relationships between properties and referring expressions. We ran the experiments on a V100 GPU, with one run costing around 2 GPU hours.

Prompt tuning. For prompt tuning, 20 samples from the Logic dataset were selected and manually annotated with intermediate and final results. They were then split into 10 train and 10 validation examples. For each prompt, we start with a simple description of the task. 4-6 examples were randomly selected from the train set as in-context examples, with the relevant intermediate outputs depending on the stage. Results were tested on the validation examples, and the prompt was updated to address common mistakes. To ensure fairness, a fixed number of 5 improvement iterations was used for each prompt, and the one showing best performance over the validation examples was chosen.

5 Results and Discussion

As shown in Table 2, our method achieves an F1 score of 78% when used with GPT-4o on the Logic dataset. When run end-to-end, the Llama-7B model reached an F1 score of only 58%, but when used with the NL2FOL pipeline, reached a score of 71%. Although end-to-end classification has shown better performance in other models, comparisons can be skewed because they may have been exposed to the Logic dataset and its labels during training because this dataset was compiled from publicly accessible web sources. On average, NL2FOL demonstrated high recall, whereas end-to-end classification demonstrated high precision.

Our challenge set LogicClimate+SNLI contains real-world logical fallacies from climate change news. Since this dataset was used to test generalization, the in-context examples we provide to all models are from the Logic dataset. NL2FOL yields results that are highly similar to the results from Logic, whereas end-to-end classification saw a drop in performance. This demonstrates that our system is also robust and adapts well to real-world texts, including texts with significant domain-specific context. This makes it effective in detecting and mitigating misinformation. Specifically, on this dataset, we find that NL2FOL outperforms direct translation with all LLMs that we tested.

5.1 Quantitative Analysis

Error analysis and interpretability. The proposed method is interpretable due to the use of natural language inputs and outputs at each step of the pipeline. This structure allows for precise identification of the specific module responsible for a failure by examining intermediate results. To evaluate this aspect, we performed an in-depth error analysis by annotating the module responsible for failure in 100 incorrect predictions made by the model. The results are summarized in Table 4.

Our analysis reveals that the majority of errors occur in the ‘Background Knowledge Retriever’, involving missed or incorrectly added contextual information in the logical form. Other errors typically pertain to incorrect identification of claims, implications, or properties. In contrast, inaccuracies in the generation of logical forms are relatively infrequent, suggesting that the model performs well in constructing accurate logical representations when provided with reliable information about the constituent entities and properties within a sentence. This finding underscores the importance of improving the background knowledge retriever module to improve overall model performance.

	Type	Sentence	Logical Form	Prediction
1	LF	X has been around for years now. Y is new. Therefore, Y is better than X.	(IsNew(a) ∧ ∼ IsNew(b)) ⇒ (IsBetterThan(a,b))	LF: Correct prediction
2	LF	Everyone is doing the Low-Carb Diet.	(∃ b (∃ a (IsDoing(b,a)))) ⇒ (∃ c (∃ a (IsDoing(c,a)))).	V: Incorrect prediction - Wrong translation given when no claim given
3	V	Two dogs are fighting in a field. Consequently, the two dogs are outside.	(∃ b (∃ a (IsFighting(a, b) ∧ IsInField(b) ∧ IsInField(b)))) ⇒ (∃ a (IsOutside(a)))	LF: Incorrect prediction - Missing semantic ground truth claim: ∀ a (IsInField(a) ⇒ IsOutside(a))
4	V	A baseball player gets ready to catch a fly ball near the outfield fence. Therefore, a person is playing baseball outdoors.	(∃ a (IsGettingReady(a) ∧ (IsABaseballPlayer(a) ∧ IsCatchingFlyBall(a) ∧ IsNearOutfieldFence(a))) ∧ (∀ e (IsABaseballPlayer(e) ⇒ IsPlayingBaseball(e))) ∧ (∀ f (IsPlayingBaseball(f) ⇒ IsABaseballPlayer(f))) ∧ (∀ g (IsNearOutfieldFence(g) ⇒ IsOutdoors(g)))) ⇒ (∃ c (∃ a (IsPlayingBaseball(a) ∧ IsOutdoors(c))))	V: Correct Prediction - The method identifies additional context by establishing relationships such as *IsBaseballPlayer* implying *IsPlayingBaseball*, and *IsNearOutfieldFence* implying *IsOutdoors*.
5	V	A woman sits alone on a park bench in the sun. Hence, a woman is in a park.	(IsSittingOn(a, b) ∧ isParkBench(b) ∧ IsInSun(a)) ⇒ (IsInPark(a)).	LF: Incorrect prediction - Missing semantic ground truth claim: $\forall a \forall b$ (IsSittingOn(a, b) ∧ isParkBench(b) ⇒ IsInPark(a))
6	V	A woman is standing at a podium. Thus, a person is at a podium.	($\exists a \exists b$ (IsStandingAt(b, a)) ∧ $\forall f \forall e \forall d$ (IsStandingAt(d,e) ⇒ IsAt(f,e)) ⇒ $\exists c \exists a$ (IsAt(c, a))	V: Correct prediction - The method identifies additional context by establishing the relationship *IsStandingAt* implying *IsAt*.

Table 3: Some example outputs of our model (abbreviations: LF = Logical Fallacy, V = Valid statement)

Sub-Module with Error	Error Proportion
Claim and Implication Parser	0.19
Incorrect Label	0.01
Property Extractor	0.13
Background Knowledge Retriever	0.54
FOL Formulation Engine	0.13

Table 4: Categorization of model errors by type on NL2FOL (GPT-4o), based on a review by domain experts in the logic of 100 randomly sampled examples

Impact of adding background knowledge to NL2FOL. Based on the error analysis, missing or incorrect background knowledge was a significant contributor to incorrect predictions of our method. To quantitatively assess the impact of grounding on model performance, we evaluated several approaches for NLI in the Background Relation Extractor. These included: (a) a pipeline without any background knowledge as a baseline, (b) a model without context where the LLM (GPT4o) only processes the input properties, (c) an LLM that incorporates both the input sentence and properties and (d) a smaller model specifically fine-tuned for NLI (BART-MNLI). Results are presented in Table 5.

We see that precision and recall both improve significantly with better grounding techniques. The LLM model with sentence context achieves the highest overall performance. This is likely due to the sentence context providing information about clauses that are omitted due to the choice of representation in FOL. This indicates that integrating robust grounding mechanisms is critical to enhancing the accuracy and reliability of the method.

	LOGIC+SNLI				LOGICCLIMATE+SNLI			
Method	**Acc.**	**P.**	**R.**	**F1**	**Acc.**	**P.**	**R.**	**F1**
(a) No Grounding	0.54	0.52	**0.88**	0.66	0.57	0.54	**0.94**	0.69
(b) LLM	0.76	**0.78**	0.74	0.75	0.79	0.80	0.78	0.79
(c) LLM w/ context	**0.78**	0.76	0.82	**0.78**	**0.80**	0.80	0.80	**0.80**
(d) BART-MNLI	0.71	0.71	0.70	0.70	0.77	**0.81**	0.71	0.77

Table 5: Comparison of different grounding methods on NL2FOL (GPT4o-mini) across the LOGIC+SNLI and LogicClimate+SNLI datasets

Impact of using an SMT solver. To assess the impact of using an SMT solver in our pipeline, we compared its performance against an LLM as a baseline for classifying the logical forms as valid or fallacies. The results, summarized in Table 6, demonstrate a significant improvement in performance metrics with the integration of the SMT solver. Results reveal the SMT-based approach significantly outperforms the LLM-based approach in all metrics across both the LOGIC and LOGIC-

CLIMATE datasets. This underscores the advantage of formal reasoning systems like SMT solvers for tasks requiring precise logical inference and structured reasoning compared to LLMs, which may lack systematic consistency in such contexts.

5.2 Qualitative Analysis

5.2.1 Success Modes of NL2FOL

S1: Captures implicit information not mentioned in premises. Previous works that directly translate natural language to logical forms suffer from an inability to capture implicit information not mentioned in the premises (Olausson et al., 2023). Our "Background Knowledge Retriever" step allows us to capture this information in the final logical form. An illustration of this can be found in Example 4 of Table 3.

S2: Captures explicit information that is missed in the representation. Our pipeline is also able to capture information that is explicitly mentioned in the premises but missed due to the choice of representation in logical form. In Example 6, in Table 3, the fact that the woman is both standing and is at the podium is lost due to the choice representation *IsStandingAt*. However, the fact that the woman is at the podium is recovered in the final logical form due to the identified background knowledge *IsStandingAt* implies *IsAt*.

S3: Comparison to direct translation. To evaluate the efficacy of the multi-step LLM pipeline, we compared it against a direct translation approach, where natural language inputs were converted into logical forms with a single LLM call using a few-shot prompt. However, this task proved to be excessively complex for LLMs. Llama failed to generate any output, citing an inability to comprehend the prompt. Larger LLMs exhibited significant limitations, with over 95% of their outputs containing syntax errors. These findings highlight the inadequacy of direct translation for complex logical reasoning tasks and underscore the necessity of a structured, multi-step approach to ensure the accuracy and syntactic correctness of the logical form.

5.2.2 Failure Modes of NL2FOL

F1: Misses some background knowledge. As can be observed in Table 4, incorrect identification of background knowledge is the most common cause for incorrect classifications. This is because any gaps in background knowledge can cause a valid statement to be identified as a logical fallacy, and an incorrectly added clause can cause a fallacy to be identified as valid. One such case is present in example 3 of the Table 3. In this case, the model is not able to identify the extra context statement because the NLI model does not identify a required ground-truth relation. If this context were to be added to the claim of the logical formula, then the statement would have been predicted to be valid.

	LOGIC				LOGICCLIMATE			
Classifier	**Acc.**	**P.**	**R.**	**F1**	**Acc.**	**P.**	**R.**	**F1**
SMT	**0.78**	**0.76**	**0.82**	**0.78**	**0.80**	**0.80**	**0.80**	**0.80**
GPT-4o	0.69	0.71	0.62	0.66	0.73	0.72	0.74	0.73

Table 6: Comparison of classification methods used with NL2FOL (GPT4o) on LOGIC and LOGICCLIMATE

F2: Limitations of NLI. Our current approach is limited to discerning relationships between two properties at a time rather than handling multiple relationships concurrently. For reference, consider Example 5 in Table 3. Here, the semantic claim involves the conjunction of two properties entailing the third, while the 'Background Knowledge Retriever' only checks whether one property entails the other. Finding such complex extra context requires more advanced techniques or additional human intervention. Including them could further improve the precision of the model overall.

F3: Imprecision of LLMs. Among the logical fallacies that our model incorrectly predicted to be a valid statement, most of these predictions failed due to the imprecision of the LLM, leading to false translations and incorrect results. Example 2 demonstrates a case where the input does not have any claim but instead jumps straight to an implication. However, the model is not able to identify that the example has no claim. As a result, we obtain an incorrect translation with our technique.

6 Conclusion

We present an effective and automatic solution to detect fallacies and tackle misinformation. We developed a strategy to distinguish logical fallacies from valid statements, involving a chaining approach to convert a sentence to first-order logic using LLMs, followed by using SMT solvers to identify whether the first-order logical statement is valid or not. If not, we interpret the counter-model generated by the SMT solver in natural language. Our proposed technique shows promising results in identifying logical fallacies and valid statements, as well as good generalizability across domains.

Ethics Statement

While the intended outcome of this research is to help fight misinformation and promote rational discourse, there are several ethical challenges that we must consider. First, dependence on AI to identify logical fallacies could influence how individuals engage in debates and discussions. There is a risk that people may over-rely on AI judgments, potentially stifling complex statements or dissenting opinions that are essential for a healthy democratic process. Moreover, the use of AI in moderating discussions, especially in identifying logical fallacies, raises ethical questions about the automation of content moderation. While it can enhance the quality of public discourse by filtering out fallacious statements, it also risks automating censorship and impacting the dynamics of online communities. In the wrong hands, logical fallacy detection tools could be exploited to silence speech or suppress viewpoints under the pretext of promoting rational discourse. This potentially allows governments or organizations to stifle opposition or critique.

To address these issues, we advocate for the development of ethical guidelines for AI use that emphasize transparency, accountability, and active user engagement. These measures are crucial in encouraging public literacy in AI and logical fallacies, ultimately empowering individuals to critically assess both AI output and arguments they may encounter.

Limitations

Scope of logical reasoning tasks. Correct identification of background knowledge is crucial for our method. While we have shown its potential in detecting logical fallacies for short and structured premises, it is important to note that this approach may miss complex relational constructs (for example, $(a \wedge b) \Rightarrow (c \vee d)$)), in which richer logical patterns may often be required in real-world reasoning tasks such as those present in multi-paragraph contexts or Question-Answering (QA) datasets.

Generalizability to other tasks and domains. We have demonstrated promising results of our approach to logical fallacy detection, but whether the findings generalize to other logical tasks and domains remains unexplored. The performance of our approach in other languages is untested and may introduce unforeseen challenges.

Going beyond first-order logic. It is unknown whether our approach would be sufficiently expressive for reasoning tasks requiring higher-order or non-classical logic, as we limit our exploration to first-order logic. Conceptually, extending our method to the aforementioned domains is feasible but would require modification to the SMT integration and LLM-driven logic translation processes. Thus, further testing may include translating to logic beyond FOL, such as temporal and higher-order logic.

Computational cost. Using LLMs and SMT solvers can incur high computational costs, such as high-performance GPUs for LLM inference, CPUs optimized for SMT solvers, and high API usage, particularly for models like GPT-o1 and Llama-7B.

Acknowledgements

This research was originally conducted as a course project for CS 257 (Introduction to Automated Reasoning) and CS 224N (Natural Language Processing with Deep Learning) at Stanford University in the 2023-24 academic year. We thank the course staff for the opportunity as well as their valuable guidance and feedback.

References

Meta AI. 2024a. Llama 3.2-11b model card. Accessed: 2025-02-15.

Mistral AI. 2024b. Ministral-8b-instruct-2410 model card. Accessed: 2025-02-15.

Tariq Alhindi, Tuhin Chakrabarty, Elena Musi, and Smaranda Muresan. 2022. Multitask instruction-based prompting for fallacy recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8172–8187, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rami Aly, Marek Strong, and Andreas Vlachos. 2023. QA-NatVer: Question answering for natural logic-based fact verification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8376–8391, Singapore. Association for Computational Linguistics.

Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 534–545.

Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. 2011. Cvc4. In *Computer*

Aided Verification, pages 171–177, Berlin, Heidelberg. Springer Berlin Heidelberg.

Clark Barrett, Aaron Stump, and Cesare Tinelli. 2009. Satisfiability modulo theories. *Communications of the ACM*, 52(9):69–77.

Iz Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2016. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*, 42(4):763–808.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models.

Matthias Cosler, Christopher Hahn, Daniel Mendoza, Frederik Schmitt, and Caroline Trippel. 2023. nl2spec: Interactively translating unstructured natural language to temporal logics with large language models. In *Computer Aided Verification. CAV 2023. Lecture Notes in Computer Science*, volume 13965, Cham. 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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: an efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, page 337–340, Berlin, Heidelberg. Springer-Verlag.

Francesco Fuggitti and Tathagata Chakraborti. 2023. Nl2ltl - a python package for converting natural language (nl) instructions to linear temporal logic (ltl) formulas. In *AAAI Conference on Artificial Intelligence*.

Gaël Gendron, Qiming Bao, Michael Witbrock, and Gillian Dobbie. 2024. Large language models are not strong abstract reasoners.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Kiesel. 2018a. "dummy, grandpa, do you know anything?": Identifying and characterizing ad hominem fallacies in the wild. In *Proceedings of the 12th International AAAI Conference on Web and Social Media (ICWSM)*, pages 206–215.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018b. Before name-calling: Dynamics and triggers of ad hominem fallacies in web argumentation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 386–396, New Orleans, Louisiana. Association for Computational Linguistics.

Christopher Hahn, Frederik Schmitt, Julia J. Tillman, Niklas Metzger, Julian Siber, and Bernd Finkbeiner.

2022. Formal specifications from natural language. *CoRR*, abs/2206.01962.

Patrick M. Haluptzok, Matthew Bowers, and Adam Tauman Kalai. 2022. Language modexrls can teach themselves to program better. *ArXiv*, abs/2207.14502.

Zhijing Jin, Abhinav Lalwani, Tejas Vaidhya, Xiaoyu Shen, Yiwen Ding, Zhiheng Lyu, Mrinmaya Sachan, Rada Mihalcea, and Bernhard Schoelkopf. 2022. Logical fallacy detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7180–7198, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jason Xinyu Liu, Ziyi Yang, Benjamin Schornstein, Sam Liang, Ifrah Idrees, Stefanie Tellex, and Ankit Shah. 2022. Lang2LTL: Translating natural language commands to temporal specification with large language models. In *CoRL Workshop on Language and Robot Learning*.

B. MacCartney and C. D. Manning. 2014. Natural logic and natural language inference. In H. Bunt, J. Bos, and S. Pulman, editors, *Computing Meaning: Volume 4*, pages 129–147. Springer Netherlands, Dordrecht.

Callistus Ireneous Nakpih and Simone Santini. 2020. Automated discovery of logical fallacies in legal argumentation. *International Journal of Artificial Intelligence & Applications*.

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.

OpenAI. 2024. Gpt-4o mini: Advancing cost-efficient intelligence. Accessed: 2025-02-15.

OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna

Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024a. Gpt-4o system card.

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024b. Openai o1 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.

William C Purdy. 1991. A logic for natural language. *Notre Dame Journal of Formal Logic*, 32(3):409–425.

Hrituraj Singh, Milan Aggarwal, and Balaji Krishnamurthy. 2020. Exploring neural models for parsing natural language into first-order logic. *ArXiv*, abs/2002.06544.

Zhivar Sourati, Vishnu Priya Prasanna Venkatesh, Darshan Deshpande, Himanshu Rawlani, Filip Ilievski, Hông-Ân Sandlin, and Alain Mermoud. 2022. Robust and explainable identification of logical fallacies in natural language arguments. *Knowledge Based Systems*, 266:110418.

Christian Stab and Iryna Gurevych. 2017. Recognizing insufficiently supported arguments in argumentative essays. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 980–990, Valencia, Spain. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. 2022. Chain of thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2024. Harnessing the power of large language models for natural language to first-order logic translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6942–6959, Bangkok, Thailand. Association for Computational Linguistics.

Appendix

A Algorithms

Algorithm 1: Compiling Logical Formula to SMT

Input: Logical formula $\mathcal{L}$ in natural language or First-Order Logic (FOL)
Output: SMT file $\mathcal{S}$ formatted for formal solvers

Step 1: Tokenize Formula
$\mathcal{T} \leftarrow$ `Tokenize(`$\mathcal{L}$`)` `// Split` $\mathcal{L}$ `into tokens based on operators, parentheses, and commas`

Step 2: Process Tokens
$\mathcal{P} \leftarrow \emptyset$ `// Initialize processed tokens set`
foreach *token* $t \in \mathcal{T}$ **do**
  if *t is a predicate* **then**
    Identify arguments of t
    Recursively `ProcessTokens()` for arguments
  else if *t is an operator or variable* **then**
    Add t to $\mathcal{P}$

Step 3: Convert Formula to Prefix Notation
$\mathcal{F}_{\text{prefix}} \leftarrow$ `InfixToPrefix(`$\mathcal{P}$`)` `// Transform logical formula from infix to prefix notation`
Recursively apply `InfixToPrefix()` for predicate arguments

Step 4: Determine Sorts
$\mathcal{S}_{\text{sorts}} \leftarrow$ `UnifySort(`$\mathcal{F}_{prefix}$`)` `// Assign sorts for variables and predicates`

Step 5: Format Formula for SMT
$\mathcal{F}_{\text{SMT}} \leftarrow$ Parenthesize $\mathcal{F}_{\text{prefix}}$ according to SMT-LIB syntax

Step 6: Generate SMT File
$\mathcal{S} \leftarrow$ `GenerateSMT(`$\mathcal{S}_{sorts}, \mathcal{F}_{SMT}$`)`
Include

- `(declare-sort)` statements for sorts.
- `(declare-fun)` statements for variables and predicates.
- Negation of $\mathcal{F}_{\text{SMT}}$.
- `(check-sat)` and `(get-model)` commands.

return $\mathcal{S}$ `// Return the SMT file for use in formal solvers`

B Prompt Examples

Note: Additional in-context examples were removed for brevity and denoted '[...]' in the following prompts.

B.1 End-to-end LLM Prompts

Prompt 1. Classifying with in-context examples (Few-shot)

> A sentence is logically valid if and only if it is not possible for it to be false.
>
> Here are some examples of classifying sentences as logical fallacies or valid sentences:

Algorithm 2: UnifySort for Predicate $A(x, y)$

Input: Predicate $A(x, y)$ with arguments and potential instances
Output: Unified sort for predicate A or an error if sorts are incompatible

Step 1: Declare the Current Sort Initialize the current sort of A: (NULL, NULL, Bool)

Step 2: Process Each Instance of Predicate A **foreach** *instance of predicate* A **do**

  Step 2.1: Determine Instance Sorts foreach *argument* x_i *in the instance* **do**
    if x_i *is a formula* **then**
      Set sort(x_i) = Bool
    else if x_i *is a variable* **then**
      Set sort(x_i) = sort(variable) `// May be NULL`

  Step 2.2: Unify Current Sort with Instance Sort foreach *statement sort in current and instance sorts* **do**
    if *sorts are not NULL and different* **then**
      Raise an error: `Incompatible sorts`
    else if *current sort is NULL and instance sort is not NULL* **then**
      Update current sort: current_sort ← instance_sort
    else if *instance sort is NULL and current sort is not NULL* **then**
      Update variable sort to match current sort

Example 1:

Input: "I met a tall man who loved to eat cheese, now I believe all tall people like cheese"
Answer: Logical Fallacy

[...]

Now, classify the following sentence. Answer with either "Logical Fallacy" or "Valid" at the start of your answer.

Input:

B.2 Intermediate NL2FOL Prompts

Prompt 2. Extracting claim and implication

Here are some examples of extracting claims and implications from an input paragraph. There can be multiple claims but only one implication.

Input: "I met a tall man who loved to eat cheese, now I believe all tall people like cheese."
Output:
Claim: "A tall man loves cheese."
Implication: "All tall people like cheese."

[...]

Do not use any subordinating conjunctions in the implication. Replace pronouns with the appropriate nouns so that there are no pronouns. Now extract the claim and implication for the following input.

Input:

Prompt 3. Getting referring expressions

You are given a sentence. Referring expressions are noun phrases, pronouns, and proper names that refer to some individual objects that have some properties associated with them. Here are some examples of finding referring expressions in a sentence:

Input: "A tall man loved cheese"
Referring expressions: A tall man

[...]

Now, find the referring expressions for the following input:

Prompt 4. Getting entity relations

Please determine the relationship between the two entities provided below. Choose the number corresponding to the statement that best describes their relationship:

1. "[Entity A]" is equal to "[Entity B]".
2. "[Entity A]" is a subset of "[Entity B]".
3. "[Entity B]" is a subset of "[Entity A]".
4. "[Entity A]" is not related to "[Entity B]".

Instructions:
- Equality check: If the two entities are equal (case-insensitive after stripping whitespace), select statement 1.
- Subset determination: If they are not equal, assess whether one entity is a subset of the other based on general knowledge and logical reasoning.
  - If "[Entity A]" is a subset of "[Entity B]", select statement 2.
  - If "[Entity B]" is a subset of "[Entity A]", select statement 3.
- Unrelated entities: If none of the above statements accurately describes the relationship.

Here are some examples:

Example 1:

Entity A: "dogs"
Entity B: "animals"
Analysis: All dogs are animals, so "dogs" is a subset of "animals".
Answer: 2

[...]

Entities:
- Entity A:
- Entity B:

Your Task:
- Analyze the relationship between "Entity A" and "Entity B" based on the instructions.
- Provide only the number (1, 2, 3, or 4) that corre-

sponds to the statement you have selected.

Prompt 5. Getting properties (claim)

Given a sentence, and the referring expressions of that sentence. Properties are anything that describes a relationship between two referring expressions, or they may describe a trait of a referring expression. These properties are essentially predicates in first-order logic.

Here are some examples of finding properties in a sentence:

Example 1:

Input sentence: A tall man loves cheese
Referring expressions: tall man: a, cheese: b
Properties: IsTall(x), LovesCheese(x)

[...]

Now extract the properties for the following input:

Prompt 6. Getting property relations

You are given two logical clauses. Your task is to identify whether or not the first clause entails the second clause, taking into account external knowledge or 'common sense'. Also, take into account the context from the input sentence.

Here are some examples:

Example 1:

Input sentence: A boy is jumping on skateboard in the middle of a red bridge. Thus, the boy does a skateboarding trick.
Clause 1: JumpsOn(boy,skateboard)
Clause 2: Does(boy, skateboarding_trick)
Answer: ENTAILMENT

[...]

Now given the following clauses. identify whether the first clause entails the second clause.

Prompt 7. Retrieving FOL expression

Given a sentence, the referring expressions of that sentence, and properties which are associated with the referring expressions. Use the given properties to convert the sentence into a first-order logical form. Use -> to represent implies, & to represent and, | to represent or and   to represent negations.

Example 1:

Input Sentence: A tall man loves cheese
Referring Expressions: A tall man: x
Properties: IsTall(x), LovesCheese(x)
Logical Form: IsTall(x) & LovesCheese(x)

[...]

C Code and Artifacts

The complete set of prompt examples and code is available in our public repository at https://github.com/lovishchopra/NL2FOL.

We encourage readers to visit the repository for details and latest updates.