Faithful and Robust LLM-Driven Theorem Proving for NLI Explanations

Xin Quan[1], **Marco Valentino**[2,3], **Louise A. Dennis**[1], **André Freitas**[1,2,4]
[1]Department of Computer Science, University of Manchester, UK
[2]Idiap Research Institute, Switzerland
[3]School of Computer Science, University of Sheffield, UK
[4]National Biomarker Centre, CRUK-MI, University of Manchester, UK
[1]{name.surname}@manchester.ac.uk
[2]{name.surname}@idiap.ch

Abstract

Natural language explanations play a fundamental role in Natural Language Inference (NLI) by revealing how premises logically entail hypotheses. Recent work has shown that the interaction of large language models (LLMs) with theorem provers (TPs) can help verify and improve the validity of NLI explanations. However, TPs require translating natural language into machine-verifiable formal representations, a process that introduces the risk of semantic information loss and unfaithful interpretation, an issue compounded by LLMs' challenges in capturing critical logical structures with sufficient precision. Moreover, LLMs are still limited in their capacity for rigorous and robust proof construction within formal verification frameworks. To mitigate issues related to faithfulness and robustness, this paper investigates strategies to (1) alleviate semantic loss during autoformalisation, (2) efficiently identify and correct syntactic errors in logical representations, (3) explicitly use logical expressions to guide LLMs in generating structured proof sketches, and (4) increase LLMs' capacity of interpreting TP's feedback for iterative refinement. Our empirical results on e-SNLI, QASC and WorldTree using different LLMs demonstrate that the proposed strategies yield significant improvements in autoformalisation (+18.46%, +34.2%, +39.77%) and explanation refinement (+29.5%, +51.5%, +41.25%) over the state-of-the-art model. Moreover, we show that specific interventions on the hybrid LLM-TP architecture can substantially improve efficiency, drastically reducing the number of iterations required for successful verification.[1]

1 Introduction

Recent studies in Natural Language Inference (NLI) have developed models to leverage natural language explanations as a mechanism for reasoning in support of a hypothesis (Wiegreffe and Marasović, 2021; Chen et al., 2021; Thayaparan et al., 2020; Valentino et al., 2022). Providing sound and logically valid natural language explanations lies at the core of NLI, as such transparent justifications enhance both interpretability and reliability for downstream tasks (Camburu et al., 2018; Valentino et al., 2022; He et al., 2024). Recent methods, in particular, have leveraged the inferential and linguistic capabilities of large language models (LLMs) by integrating them with external theorem provers (TPs) to automatically verify the logical validity of explanations for NLI (Pan et al., 2023; Olausson et al., 2023; Quan et al., 2024b; Dalal et al., 2024).

However, these integrated neuro-symbolic approaches still face notable challenges. First, automated theorem provers (ATP) require a machine-verifiable formal language, yet LLMs often fail to produce precise autoformalisations, underscoring their limited capacity to faithfully convert complex natural language inputs into rigorous formal representations (Wu et al., 2022; Jiang et al., 2024; Quan et al., 2024b). Second, syntactic errors are frequently introduced during the autoformalisation process, leading to reduced theorem-proving success rates when dealing with more complex material inferences (Pan et al., 2023; Olausson et al., 2023; Zhang et al., 2024). Third, when provided with external feedback on complex explanations, LLMs often struggle to combine axioms (explanations) into cohesive proofs and effectively self-correct, limiting their effectiveness in more complex NLI settings (Quan et al., 2024a,b).

In this paper, we build upon the state-of-the-art LLM-based theorem proving framework for NLI, Explanation-Refiner (Quan et al., 2024b). In particular, we explore methodologies to improve the faithfulness of autoformalisation and deliver a more robust way to effectively and efficiently provide logically valid explanations. We further examine how varying degrees of dataset complex-

[1]Code and data are available at: https://github.com/neuro-symbolic ai/faithful_and_robust_nli_refinement

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 17734–17755

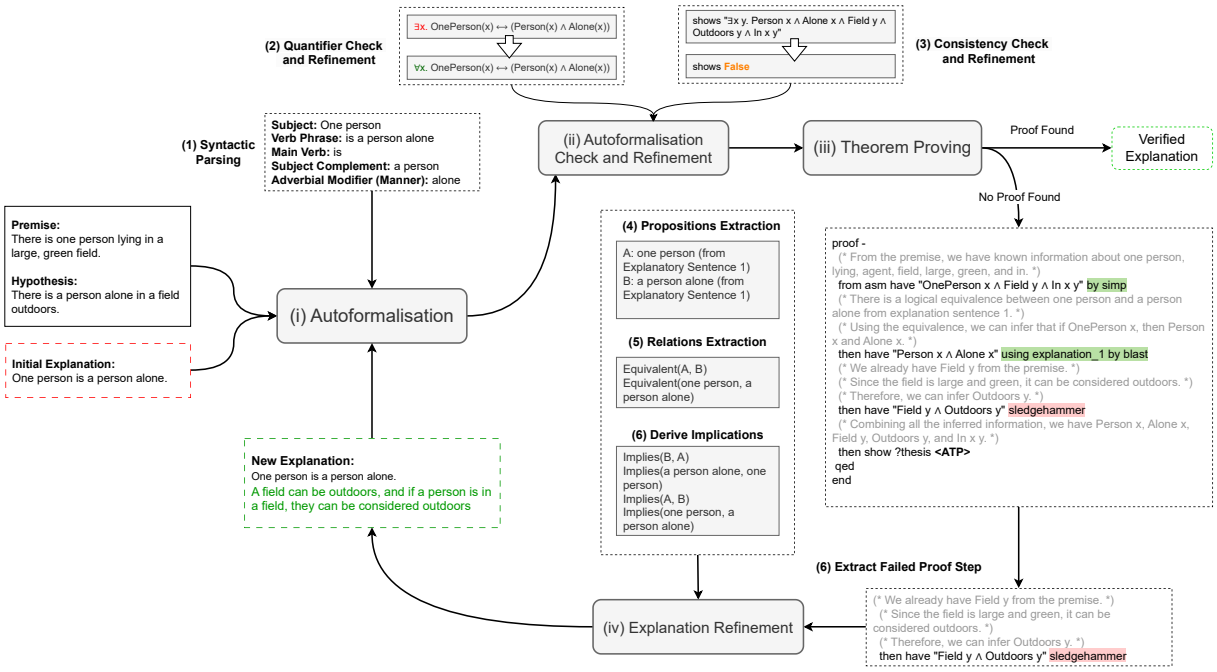


Figure 1: An illustration of our proposed interventions for improving LLM-driven theorem proving for NLI. The interventions employ different techniques including syntactic parsing, quantifier refinement, logical consistency refinement, and logical expression extraction to guide LLMs in generating more faithful and robust proof sketches for NLI and effectively refine natural language explanations. This approach provides more structured and explicit feedback by pinpointing the exact logical errors identified in the explanations.

ity in multi-hop reasoning affect the reliability of proof step generation in LLM-Driven theorem proving. In general, we implement a neuro-symbolic framework to address the following research questions: *RQ1: "To what extent can we deliver faithful autoformalisation that preserves semantic information?" RQ2: "What types of syntactic errors commonly appear in formal representations, and how effectively can state-of-the-art LLMs refine these errors?" RQ3: "Can state-of-the-art LLMs generate structured proof steps that can effectively provide feedback to refine explanations with complex sentences and logical relations?"*

To answer these questions, we investigate how to systematically leverage syntactic parsing during autoformalisation to guide LLMs generate logical representation of explanations. In addition, we define the general autoformalisation error types and use LLMs to refine these errors explicitly from the output message of a TP. Furthermore, we propose a method to extract the logical propositions, relations and implications to guide LLMs to generate proof sketches for automated theorem proving and explanation refinement.

Our empirical evaluation on e-SNLI (Camburu et al., 2018), QASC (Khot et al., 2019), and WorldTree (Jansen et al., 2018) shows that the proposed framework improves the faithfulness of autoformalisation by 18.46%, 34.2%, 39.77%, respectively, compared to Explanation-Refiner. Additionally, the number of refined explanations produced by our framework exceeds that of Explanation-Refiner across all LLMs: raising refinement rates from 41% to 95%, 17% to 90%, and 7% to 73% across all three datasets. To summarise, the main contributions of this paper are:

1. We introduce Faithful-Refiner, a novel neuro-symbolic framework that provides more robust and faithful verification and refinement of explanations in NLI, surpassing existing LLM-driven theorem-proving approaches.

2. We conduct a quantitative evaluation of explanation refinement and autoformalisation across different LLMs, achieving an average improvement of 29.5%, 51.5%, and 41.25% more refined explanations, as well as 5.06%, 6.86%, and 32.16% on syntactic errors reduction compared to the state-of-the-art.

3. We adopt a range of automatic metrics to measure the quality of explanations and autoformalisation, showing that the proposed framework significantly improve the faithfulness of

the autoformalisation process.

4. We also perform a manual evaluation to assess the perceived quality of the formalised logical forms and conduct an extensive ablation study, elucidating the role of each proposed component and identifying key factors influencing automated theorem proving for NLI.

2 Automated Theorem Proving for Explanation-Based NLI

In this paper, we define an *explanation* E_i as a set of facts $\{f_1, f_2, \ldots, f_n\}$ that establish a logically valid entailment between *premises* p_i and a *hypothesis* h_i, such that $p_i \cup E_i \models h_i$ holds.

In this work, we leverage an external theorem prover TP to systematically verify these entailments in an automated manner. Specifically, given the set of input sentences $S = p_i \cup \{h_i\} \cup E_i$, we aim to build a set of logical forms $\phi = \{\Phi(s) \mid s \in S\}$, where Φ is the *autoformalisation process* that converts natural language sentences into symbolic representations. From these logical forms, we construct a theory $\Theta = (A, \tau)$, where $A = \{a_1, a_2, \ldots, a_n\}$ is the set of axioms derived from formalising E_i, and τ is the theorem to be proven, composed of p_i and h_i. If an automated theorem prover (ATP) can derive a valid proof for Θ, we conclude that E_i is *sound and logically valid*. Otherwise, we refine E_i by using the failed proof steps as feedback, iteratively generating a refined explanation E_i' that ultimately leads to a valid justification.

3 Methodology

To effectively enhance the joint inference capabilities and robustness between LLMs and theorem provers for explanation-based NLI, we propose a novel framework to enhance three key components: autoformalisation, logical and syntactic error checking and refinement, and LLM-guided proof construction. As illustrated in Figure 1, the pipeline begins with the automated formalisation of natural language into logical representations.

Unlike the previous state-of-the-art approach (i.e., Explanation-Refiner), we begin with a syntactic parsing step that guides LLMs in translating natural language elements into a formal specification compatible with theorem provers. The LLM is prompted to automatically formalise the explanatory sentences into axioms and construct a theorem composed of assumption clauses (drawn from the premise) and a proof goal (derived from the hypothesis). After formalising the input sentences, we apply a quantifier and a logical consistency check along with a refinement process.

Similar to Jiang et al. (2022b) and Quan et al. (2024b), we adopt Isabelle/HOL (Nipkow et al., 2002) to formally verify the constructed theory. Specifically, we invoke the Sledgehammer tool (Paulson and Blanchette, 2012) within Isabelle/HOL to call upon multiple automated theorem provers (e.g., CVC4[2], Vampire[3]), which attempt to prove the theorem derived from the translated NLI tasks. If any prover succeeds, we conclude that the explanation is logically sound, thereby confirming that the premise entails the hypothesis.

If no proof is found, we use an LLM to extract logical propositions and relations from the natural language explanations. We then employ an intermediate propositional representation to derive further implications among these propositions, prompting the LLM to generate a step-by-step proof sketch—rather than having the LLM serve directly as a proof planner as in Explanation-Refiner.

Finally, we iteratively attempt to prove each subproof step, gathering information about failed steps, using it as feedback to prompt the LLM to generate an updated explanation to refine the logical errors identified in the previous proof sketch and start a new iteration.

3.1 Isabelle/HOL Theory Generation

Autoformalisation plays a critical role in integrating theorem provers with LLMs, especially for complex sentence structures. Similar to Quan et al. (2024b), we apply Neo-Davidsonian event-based semantics (Parsons, 1990) to formalising the natural language sentences within each aspect of an event with distinct predicates. This approach provides a robust foundation for formalising explanatory sentences while maximising content preservation (Maienborn et al., 2011).

However, simply using few-shot prompting for autoformalisation does not guarantee a faithful process, which may lead to inconsistencies between the natural and formal languages expressions. To alleviate this, we begin by performing syntactic parsing via the LLMs on all provided sentences to extract their grammatical structure, identifying the

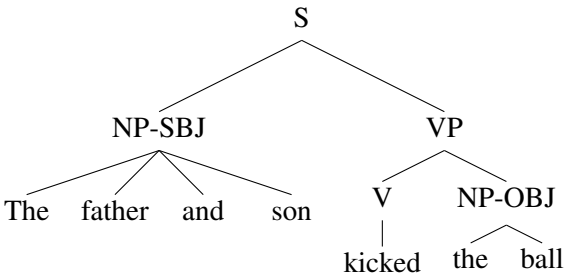
[2] https://cvc4.github.io/

[3] https://vprover.github.io/projects.html

```
(* Explanation 1: A man and woman are at the park. *)
axiomatization where
  explanation_1: "∃x y z. Man x ∧ Woman y ∧ Park z ∧ At x z ∧ At y z"
theorem hypothesis:
  (* Premise: A man and woman sit on a park bench with a set of newlyweds behind *)
  assumes asm: "Man x ∧ Woman y ∧ ParkBench z ∧ Newlyweds w ∧ Sit e ∧ Agent e x ∧ Agent e y ∧ Patient e z
       ∧ Behind w z"
  (* Hypothesis: People outside *)
  shows "∃x. People x ∧ Outside x"
proof -
  (* From the premise, we have information about a man and a woman sitting on a park bench. *)
  from asm have "Man x ∧ Woman y" by blast
  (* Explanation 1 states that a man and a woman are at the park. *)
  (* This implies that they are outside, as parks are typically outdoor locations. *)
  from explanation_1 have "∃x y z. Man x ∧ Woman y ∧ Park z ∧ At x z ∧ At y z" by blast
  (* Since a man and a woman are at the park, they are outside. *)
  then have "People x ∧ Outside x" <ATP>
  then show ?thesis <ATP>
qed
```

Figure 2: An example of a proof sketch constructed by the model to verify an explanation from the e-SNLI dataset. In this example, while ATPs find proofs for the first two steps using proof tactics, they fail to derive *People* $x \wedge$ *Outside* x due to missing premises. The feedback provided by Isabelle is then adopted in the next iteration to refine the explanation and the proof sketch.

main predicate-argument structure. These elements are subsequently mapped onto the agent, event action, and patient roles within a Neo-Davidsonian event semantics framework. For example, consider the sentence *"The father and son kicked the ball"*. We can parse it as:



indicating that "The father and son" is the subject while "the ball" is the object. Thus we could build the Neo-Davidsonian event semantics to formalise it as:

```
∃xyze. (Father(x) ∧ Son(y) ∧ Ball(z) ∧ Kicked(e) ∧
     Agent(e, x) ∧ Agent(e, y) ∧ Patient(e, z))
```

By leveraging such a process, we construct a clear representation indicating that the father and the son are the agents performing the event (kick), while the ball is the patient receiving the action, thus capturing all relevant semantic information in the transition from natural language to formal language. We then construct the Isabelle/HOL theory with axioms (explanatory sentences) and the theorem (premise and hypothesis sentences).

3.2 Autoformalisation Critiques

Recent studies have identified errors and inconsistencies in LLM-generated outputs as a challenge in autoformalisation and have proposed several methods (Pan et al., 2023; Zhang et al., 2024; Gandarela et al., 2025) to address them. In our work, we categorise the errors in this phase into three main dimensions: quantifier scoping error, syntax errors, and logical inconsistencies.

Quantifier Scoping Error The quantifiers indicate the scope of logical deductions. In synthetically generated datasets quantifiers are constrained to predefined settings. In contrast, in naturally occurring NL settings, incorrect quantifiers in axioms may still prove a theorem within a formal system, but when those logical forms are restated in natural language, their soundness may fail to hold in the real world. For example, one cannot declare "all animals are mammals." Thus, we introduce a *quantifier check and refinement* soft-critique stage to prompt the LLM to compare the quantifiers in the logical forms against real-world knowledge, thereby avoiding any over-scoped quantifiers.

Syntax Errors Internal syntax errors, primarily those caused by missing brackets or type unification conflicts of logical variables, can often be identified through the theorem prover's output. Once identified through a hard critique via the TP, these errors can be systematically refined or corrected by adjusting the syntax or revising type declarations. We then employ an LLM for refinement to support the systematic correction of these output errors (constrained within up to five iterations).

Logical Inconsistencies In a formal system, if contradictory or meaningless axioms are introduced, the system becomes inconsistent. By the

principle of explosion (*ex falso [sequitur] quodlibet*), any proposition can then be derived from such an inconsistency. To test such errors within the autoformalised axioms, we construct a modified theorem τ_{False} by replacing the conclusion of τ with "False". We then attempt to prove this modified theorem, if the TP finds a proof, it indicates a contradiction within the axioms. In this case, we use an LLM to refine the axioms and attempt to solve the contradictions.

3.3 Proof, Verification and Refinement

After autoformalisation checking and refinement, we employ the theorem prover TP to verify the logical validity of the axioms and determine whether $A \models \tau$ holds. We first use the Sledgehammer tool in Isabelle/HOL for ATPs to automatically find a proof of the theorem. If a proof is found, we extract all possible proofs from Sledgehammer's results and state that the explanation is logically valid. If Sledgehammer fails to find a proof, we construct a proof sketch to attempt a step-by-step proving using ATPs based on a set of logical interpretations.

Logical Propositions, Relations and Implications Liu et al. (2025) employ logical expressions to guide LLMs and mitigate information loss in intermediate reasoning processes. Similarly, we begin with a logical proposition extraction step. In this step, we use an LLM to extract logical propositions and relations from the explanation E_i. Consider the following extracted logical relations as an example: A: it is raining; B: the grass is wet; C: kids can play outside; D: kids are happy as well as the following logical relations: $A \rightarrow B$ (if it is raining, the grass is wet) and $B \rightarrow \neg C$ (if the grass is wet, kids cannot play outside). Next, we leverage the extracted logical relations using a SymPy-based propositional-level representation (Meurer et al., 2017) [4] to derive additional implications based on formal logical laws. For instance, from the example above, SymPy can deduce $A \rightarrow \neg C$ (if it is raining, kids cannot play outside). Algorithm 1 shows the implementation of SymPy to find derived logical implications.

Proof Sketch By combining the logical propositions, relations, and these derived implications, the LLM can construct a step-by-step guided proof sketch that establishes a logical reasoning chain to prove the goal. As shown in Figure 2, the comments partially indicates how the logical expression guides LLMs to build the step-wise proof steps, while we replace the proof tactics with <ATP>, which uses Sledgehammer (Isabelle's automated theorem proving tool) to search for proofs. In Isabelle/HOL, proof tactics are commands that systematically decompose complex proofs into simpler sub-goals, automating routine steps such as simplification. Typically, one is asked to prove a statement X given assumptions Y by using proof tactics Z, where Z includes commands like simp (for simplification), auto (for automatic reasoning), and blast (for first-order reasoning). These tactics instruct Isabelle's proof engine on how to process a proof step by applying appropriate rules, simplifications, or other reasoning methods. Once Sledgehammer finds a proof, only the explanatory sentences (axioms) used in that proof are retained as the final refined explanation. For example, if the proof is written as "assms explanation_1 explanation_2 by blast", then only explanation 1 and explanation 2 constitute the minimal set of explanatory sentences required to entail the hypothesis.

Explanation Refinement If the automated theorem prover fails or finds no proofs in a previous proof step, we extract that proof step along with the proof strategy from the comments part as feedback to prompt the LLM to refine the logical error (i.e., missing premises) of the related explanatory sentences and process into next iteration to iteratively verify and refine the explanation. After the explanation refinement, we drop any explanatory sentences that are not included in the proof, as they are deemed unnecessary to deriving the hypothesis from the premise and then proceed to the next iteration cycle. We followed the same prompts used in Explanation-Refiner (Quan et al., 2024b) for autoformalisation. Prompts used for syntactic parsing, quantifier refinement, logical consistency and proof steps generation are reported in Appendix E.

4 Empirical Evaluation

4.1 Datasets and Models

We conducted experiments with four state-of-the-art LLMs within the proposed framework: GPT-4o (OpenAI, 2023), GPT-4o-mini (OpenAI, 2023), Llama3.1-70b (Grattafiori et al., 2024), Deepseek-V3 (DeepSeek-AI, 2024). Following Quan et al. (2024b), we applied three sampled NLI datasets of e-SNLI (Camburu et al., 2018), QASC (Khot et al., 2019), and WorldTree (Jansen et al., 2018)

[4] https://www.sympy.org/en/index.html

	e-SNLI				QASC				WorldTree			
	Init.	**Final**	**#Iter**	**#Calls**	**Init.**	**Final**	**#Iter**	**#Calls**	**Init.**	**Final**	**#Iter**	**#Calls**
Explanation-Refiner												
Llama3.1-70b	23%	51%	4.08	34.56	4%	18%	4.07	37.49	2%	15%	5.23	51.61
GPT-4o-mini	13%	30%	3.65	32.55	3%	20%	5.12	44.84	0%	4%	5.00	46.12
GPT-4o	31%	**71%**	3.62	32.34	4%	26%	4.35	38.45	2%	13%	**4.18**	39.26
Deepseek-V3	25%	69%	**2.82**	27.74	4%	**38%**	**3.71**	35.97	3%	**31%**	4.52	42.64
Our Approach												
Llama3.1-70b	36%	78%	2.38	16.28	11%	68%	2.90	25.40	6%	52%	4.62	35.72
GPT-4o-mini	32%	77%	2.27	16.62	12%	71%	3.35	27.10	5%	47%	4.75	36.50
GPT-4o	39%	89%	1.54	10.24	10%	79%	3.22	22.32	9%	56%	3.86	26.16
Deepseek-V3	41%	**95%**	**1.50**	**9.52**	17%	**90%**	**2.53**	**20.18**	7%	**73%**	**3.55**	**25.30**

Table 1: Comparison of our approach with Explanation-Refiner on different LLMs across three datasets. Init. represents the number of explanations that are initially verified as logically valid. Final indicates the number of explanations that are refined within a maximum of 10 iterations, while #Iter indicates the average iteration required to refine an explanation. #Calls shows the average number of LLM calls needed to fully refine an explanation.

each comprising 100 instances. We compare our approach with Explanation-Refiner (Quan et al., 2024b), a state-of-the-art LLM-driven theorem prover for NLI that adopts a similar pipeline but without incorporating the specific strategies for guiding autoformalisation via syntactic parsing, performing consistency and quantification checks, and guide refinement via proof sketches and explicit implication derivation.

4.2 Results

The proposed architectural interventions effectively improve the verification and refinement of natural language explanations. Table 1 and Figure 3 compares our proposed framework with Explanation-Refiner on the tasks of verifying and refining natural language explanations across multiple LLMs. The results show that our approach more effectively and efficiently refines explanatory sentences for explanation-based NLI. In contrast, Explanation-Refiner achieves substantially lower refinement rates, for example, 51% versus 78% in e-SNLI for Llama3.1, 69% versus 95% for Deepseek-V3, 30% versus 77% for GPT-4o-mini, and 71% versus 89% for GPT-4o. Furthermore, Explanation-Refiner generally requires more iterations to refine each explanation, indicating that although it may identify specific logical errors, it is less efficient. For instance, Explanation-Refiner requires an average of 4.31 iterations in the QASC dataset, compared to 3.0 for our approach. Its performance is particularly limited on the WorldTree dataset, which contains complex, real-world scientific explanations requiring multi-hop reasoning. By contrast, our framework refines a significantly larger number of explanations in WorldTree, underscoring its capacity to handle more challenging inference scenarios. Furthermore, the current approach requires fewer LLM calls on average to fully refine an explanation, resulting in reduced inference time and cost compared to the Explanation-Refiner (See Table 1). The average number of LLM calls was reduced by 58.60%, 39.39%, and 31.15% on e-SNLI, QASC, and WorldTree, respectively, across all LLMs. Applying our approach with Deepseek-V3 on the e-SNLI dataset yields the most significant reduction, at 65.68%.

The refinement process effectively corrects autoformalisation errors. Figure 3d, 3e, and 3f present the number of theories in the last iteration containing syntactic and inconsistency errors over five syntax error refinement iterations, comparing our proposed framework with Explanation-Refiner. Overall, our framework yields fewer syntactic errors. By incorporating syntactic parsing into autoformalisation, it guides LLMs to capture fine-grained logical properties of natural language sentences, thereby reducing type unification errors in constructed theories. Empirically, most syntactic errors diminish considerably within the first three iterations, after which the rate of improvement stabilises. The evaluation results of the number of theories that contain logical consistency errors are shown in Figure 5.

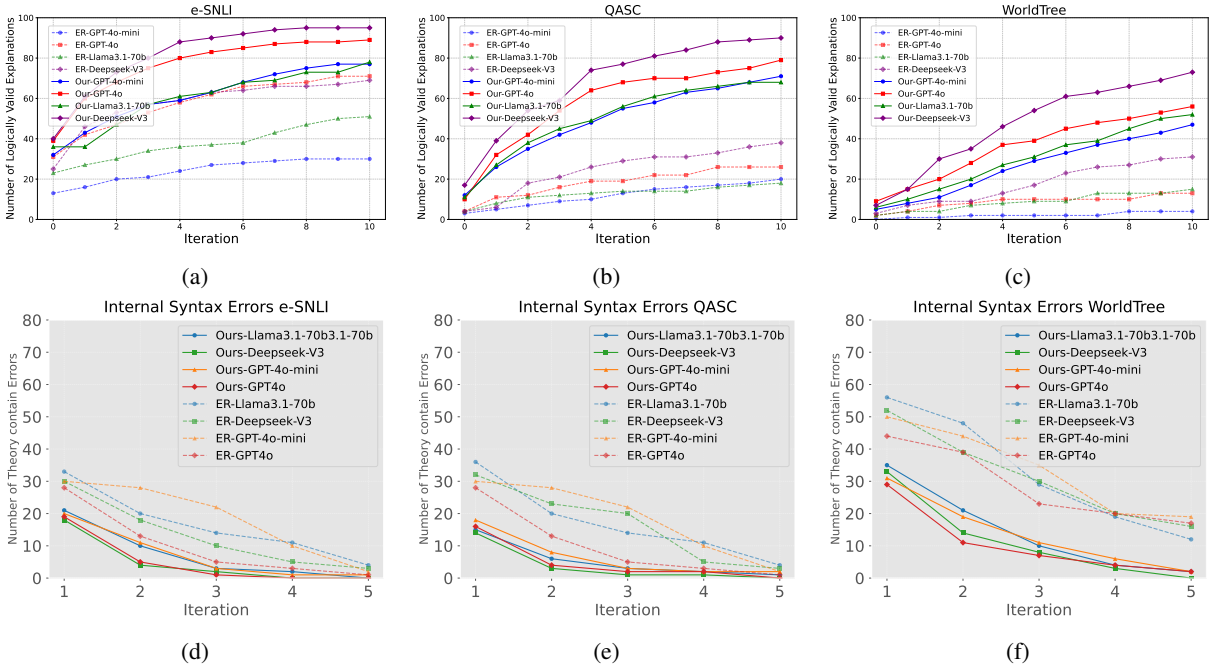


Figure 3: Top – Number of logically valid explanations at each refinement iteration. Bottom – Number of theories that contain internal syntactic errors at each syntax error refinement stage.

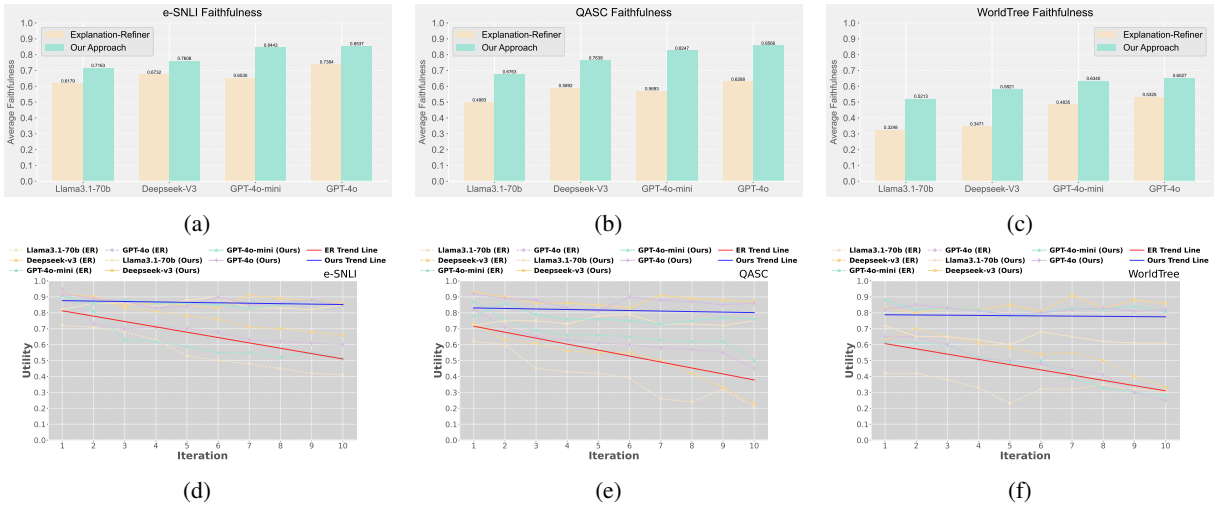


Figure 4: Top – The average faithfulness of the autoformalisation process across different LLMs. Bottom – The utility of explanation at different refinement iterations. A higher utility indicating the newly refined explanation are more likely be used in the proof of next iteration.

Syntactic parsing improves faithfulness in autoformalisation. We convert the autoformalised logical forms back into natural language sentences using a rule-based algorithm that reconstructs each sentence from its action/verb predicates and corresponding argument information. We then calculate the cosine similarity between these reconstructed (informalised) sentences between the original sentences as the faithfulness of autoformalisation, as shown in Figure 4. Our approach shows a generally higher faithfulness compared to Explanation-Refiner, with an average of 0.7938, 0.7804, and 0.5975, compared to 0.6706, 0.5714, and 0.4220 across all three datasets. Our findings indicate that certain models exhibit comparatively lower similarity scores than others. Further investigation reveals that models such as Llama3.1-70b tend to generate non-existent predicates during formalisation in Explanation-Refiner, resulting in over-generation that undermines faithfulness and introduces extraneous information into the theory. More details about the rule-based algorithm are included in the

	e-SNLI			QASC			WorldTree		
	Init.	**Final**	**#Iter**	**Init.**	**Final**	**#Iter**	**Init.**	**Final**	**#Iter**
Ablations on our approach									
GPT-4o (- **logical relations**)	34%	74%(−**15**%)	2.24	12%	58%(−21%)	3.46	6%	38%(−18%)	4.36
GPT-4o (- **detailed feedback**)	35%	83%(−6%)	2.86	13%	56%(−**23**%)	4.45	5%	17%(−**39**%)	6.46
GPT-4o (- refine quantifiers)	34%	87%(−2%)	1.63	14%	83%(+4%)	2.89	7%	49%(−7%)	3.65
GPT-4o (- **refine syntax errors**)	21%	74%(−**15**%)	2.34	5%	58%(−21%)	4.11	2%	24%(−32%)	6.48
Deepseek-V3 (- logical relations)	39%	89%(−6%)	1.68	16%	77%(−13%)	2.64	10%	58%(−15%)	4.01
Deepseek-V3 (- **detailed feedback**)	31%	86%(−9%)	3.22	22%	69%(−21%)	4.12	6%	41%(−**32**%)	6.13
Deepseek-V3 (- refine quantifiers)	34%	96%(+1%)	1.64	14%	93%(+3%)	1.89	6%	70%(−3%)	3.23
Deepseek-V3 (- **refine syntax errors**)	28%	77%(−**18**%)	2.69	12%	68%(−**22**%)	2.84	4%	46%(−27%)	5.32

	e-SNLI			QASC			WorldTree		
	V.	**I.**	**Q.**	**V.**	**I.**	**Q.**	**V.**	**I.**	**Q.**
Explanation-Refiner									
GPT-4o	**9%**	**3%**	**6%**	**18%**	**9%**	**18%**	**33%**	**8%**	**16%**
Deepseek-V3	27%	**3%**	10%	34%	9%	25%	44%	23%	31%
Ablations									
GPT-4o (- refine quantifiers)	9%	2%	10%(+4%)	13%	2%	23%(+5%)	38%	6%	19%(+6%)
GPT-4o (- refine syntax errors)	16%(+7%)	1%	3%	38%(+**20**%)	3%	7%	56%(+**23**%)	3%	16%
Deepseek-V3 (- refine quantifiers)	25%	5%	16%(+**6**%)	31%	14%	35%(+**10**%)	32%	11%	38%(+**7**%)
Deepseek-V3 (- refine syntax errors)	38%(+**11**%)	4%	11%	51%(+17%)	9%	15%	67%(+**23**%)	21%	27%

Table 2: Top – Ablation study on the impacts of removing components from the overall architecture. Bottom – Comparison of manually evaluated variable, implication, and quantifier errors in the autoformalisation process from a randomly sampled set of 100 Isabelle/HOL theories across all iterations for each LLM.

Appendix B.

Logically guided proof sketches provide effective feedback for explanation refinement. By constructing proof steps from logical propositions, relations, and derived implications, our method more precisely pinpoints logical errors, enabling the LLM to iteratively refine explanatory sentences in subsequent attempts to prove the theorem. As shown in Figure 4, the average utility defined as the proportion of newly introduced explanations that are applied in the next iteration's proof remains consistently higher for our approach compared to Explanation-Refiner, even as the number of iterations increases. In contrast, Explanation-Refiner's utility markedly decreases over successive iterations.

4.3 Ablation Study

We conducted several ablation studies to evaluate the impact of the proposed components. Table 2 shows the results on GPT-4o and Deepseek-V3, while Table 3 in Appendix A shows the full ablations.

Detailed feedback and syntax error refinement have the highest impact. The most significant drop in performance is observed from removing detailed feedback and syntax error refinement steps. Providing detailed, step-level feedback to the LLM proves significantly more effective than using only a binary signal (i.e., *provable* or *unprovable*). When replacing detailed with binary feedback, the number of refined explanations dropped substantially; for instance, GPT-4o showed a 39% decrease in refined explanations in the WorldTree dataset. Excluding the syntactic error refinement stage frequently yielded theories that failed under theorem prover scrutiny, thereby producing little to no useful feedback for subsequent refinement.

Logical expression aids LLMs in proof sketches generation, reducing hallucinations that could lead to incorrect or failed proof construction for explanation refinement. Eliminating the logical expression-guided proof step generation component led to an increase in required iterations for explanation refinement and a reduction in the total number of successfully refined explanations. These findings highlight the importance of logical expressions in constructing coherent proofs and mitigat-

ing hallucinations that otherwise result in incorrect or failed proofs.

Variable and quantifier errors significantly impact the faithfulness of autoformalisation. We further conducted a human evaluation on three types of errors: variable errors (identifiable by the theorem prover), implication errors, and quantifier errors (not identifiable by the theorem prover) as shown in Table 2. Our findings suggest that using LLMs for autoformalisation still leaves notable gaps, particularly in accurately handling variables and quantifiers. As shown in Table 2, removing the quantifier refinement did not substantially alter the number of refined explanations. However, human evaluation indicates that the number of quantifier-related errors increased when this refinement was omitted. Explanation-Refiner does not apply a syntactic parsing and quantifier refinement, resulting in more errors being introduced for variable, implication and quantifier errors as shown in Table 2. Thus, we introduced both syntax error refinement and quantifier error refinement processes. Our results show a significant reduction in the overall error rate following the corresponding soft-critique model refinements.

5 Related Work

LLM-Symbolic Integration Recent approaches have attempted to integrate LLMs with external symbolic models through autoformalisation – i.e., the translation of informal statements into formal representations. Recent work explores this task in both mathematical (Wu et al., 2022; Jiang et al., 2022b; Agrawal et al., 2022; Zhang et al., 2024; Xin et al., 2024a; Lu et al., 2024a,b; Leang et al., 2025) and logical (Olausson et al., 2023; Quan et al., 2024a; Kirtania et al., 2024; Lee et al., 2025; Raza and Milic-Frayling, 2025) domains using the support of automated theorem provers. Several studies (Pan et al., 2023; Jiang et al., 2024; Quan et al., 2024b) transform natural language sentences into logical forms based on LLMs. Qi et al. (2025) integrate symbolic provers into the generation process of logical reasoning problems and propose a multi-hop first-order logic (FOL) reasoning dataset. In contrast, our work tackles real-world occurrences of material inferences rather than purely synthetic data, thereby requiring more robust semantic representations and autoformalisation process to capture the complexity of multi-step reasoning over material inferences.

Theorem-Proving with LLMs Proof generation refers to the task of generating intermediate proof steps as tactic predictions in automated theorem proving (Li et al., 2024). Recent work harness LLMs to produce formal proof scripts (Polu and Sutskever, 2020; Jiang et al., 2022a; Zhao et al., 2023; Xin et al., 2024b; First et al., 2023; Frieder et al., 2024; Welleck and Saha, 2023a,b; Thakur et al., 2024; Poulsen et al., 2024), often by translating high-level reasoning into low-level tactics. Quan et al. (2024b), for example, prompts the LLM to first produce a rough, informal inference strategy in natural language, and then automatically formalise this strategy into Isabelle/HOL proof steps. Jiang et al. (2024) utilises a tactic generator and a proof search module to select tactics during proof construction to build a proof tree from the root goal. Liu et al. (2025) use propositional logic, introduce Logic-of-Thought prompting, a technique that appends logically enriched descriptions to the original context, thereby improving informational completeness and bolstering the LLMs logical reasoning ability. In contrast, our approach synthesises logical reasoning guidance in close iterative dialogue with automated provers to provide more robust and interpretable proofs in contrast to LLM-driven single-pass methods.

6 Conclusion

In this paper, we proposed formally-guided methods to address the challenges involved in using external theorem provers to verify and refine natural language explanations in the domain of natural language inference. By incorporating syntactic parsing, targeted syntactic error checking, logical-relation guidance, and detailed feedback at each proof step, our approach significantly outperforms prior work in both faithfulness of autoformalisation and robustness of iterative explanation refinement. Ablation studies underscore the importance of each component in reducing syntactic issues, maintaining consistency, and promoting more efficient logical verification and refinement. This framework opens avenues for more transparent, reliable, and scalable NLI systems. Moving forward, we plan to explore more advanced theorem-proving strategies with LLMs and domain-specific expansions, ultimately advancing toward increasingly interpretable and robust end-to-end NLI pipelines.

Limitations

Although our framework substantially improves both the consistency of autoformalisation and the robustness of explanation verification, certain limitations remain. First, LLMs can still introduce variable inconsistencies, erroneous implications, and incorrect quantifiers that are not fully resolved by automated checking. Second, some explanations require nuanced real-world knowledge or domain-specific axioms that exceed current formal reasoning capabilities, requiring expert oversight. Finally, the reliability of our iterative refinement pipeline hinges on high-quality LLM output and proof-step feedback; degraded model performance or noisy system responses can hinder successful verification. Future work may explore more advanced semantic checks, stronger model calibration, and selective human intervention to further enhance faithfulness and correctness.

Ethical statement

While this work focuses on the introduction of mechanisms for improving the control and logical consistency properties of LLM-based NLI, having an overall positive impact, further investigations are needed to understand the specific conditions in which these methods can perform. The application of these methods on real-world or critical settings need to be complemented by human supervision or extensive quantitative and qualitative assessment.

Acknowledgments

This work was partially funded by the SNSF project NeuMath (200021_204617), by the EPSRC grant EP/T026995/1, "EnnCore" under Security for all in an AI-enabled society, by the CRUK National Biomarker Centre, and supported by the Manchester Experimental Cancer Medicine Centre and the NIHR Manchester Biomedical Research Centre.

References

Ayush Agrawal, Siddhartha Gadgil, Navin Goyal, Ashvni Narayanan, and Anand Tadipatri. 2022. Towards a mathematics formalisation assistant using large language models. *Preprint*, arXiv:2211.07524.

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.

Qianglong Chen, Feng Ji, Xiangji Zeng, Feng-Lin Li, Ji Zhang, Haiqing Chen, and Yin Zhang. 2021. KACE: Generating knowledge aware contrastive explanations for natural language inference. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2516–2527, Online. Association for Computational Linguistics.

Dhairya Dalal, Marco Valentino, Andre Freitas, and Paul Buitelaar. 2024. Inference to the best explanation in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 217–235, Bangkok, Thailand. Association for Computational Linguistics.

DeepSeek-AI. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. 2023. Baldur: Whole-proof generation and repair with large language models. *Preprint*, arXiv:2303.04910.

Simon Frieder, Julius Berner, Philipp Petersen, and Thomas Lukasiewicz. 2024. Large language models for mathematicians. *Preprint*, arXiv:2312.04556.

João Pedro Gandarela, Danilo S. Carvalho, and André Freitas. 2025. Inductive learning of logical theories with llms: An expressivity-graded analysis. *Preprint*, arXiv:2408.16779.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and Ahmad Al-Dahle. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Xuanli He, Yuxiang Wu, Oana-Maria Camburu, Pasquale Minervini, and Pontus Stenetorp. 2024. Using natural language explanations to improve robustness of in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13477–13499, Bangkok, Thailand. Association for Computational Linguistics.

Peter Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton Morrison. 2018. WorldTree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Albert Jiang, Konrad Czechowski, Mateja Jamnik, Piotr Milos, Szymon Tworkowski, Wenda Li, and Yuhuai Tony Wu. 2022a. Thor: Wielding hammers to integrate language models and automated theorem provers. In *NeurIPS*.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2022b. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *ArXiv*, abs/2210.12283.

Dongwei Jiang, Marcio Fonseca, and Shay Cohen. 2024. LeanReasoner: Boosting complex logical reasoning with lean. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7497–7510, Mexico City, Mexico. Association for Computational Linguistics.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Alexander Jansen, and Ashish Sabharwal. 2019. QASC: A dataset for question answering via sentence composition. In *AAAI*.

Shashank Kirtania, Priyanshu Gupta, and Arjun Radhakrishna. 2024. LOGIC-LM++: Multi-step refinement for symbolic formulations. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024)*, pages 56–63, Bangkok, Thailand. Association for Computational Linguistics.

Joshua Ong Jun Leang, Giwon Hong, Wenda Li, and Shay B. Cohen. 2025. Theorem prover as a judge for synthetic data generation. *Preprint*, arXiv:2502.13137.

Jinu Lee, Qi Liu, Runzhi Ma, Vincent Han, Ziqi Wang, Heng Ji, and Julia Hockenmaier. 2025. Entailment-preserving first-order logic representations in natural language entailment. *Preprint*, arXiv:2502.16757.

Zhaoyu Li, Jialiang Sun, Logan Murphy, Qidong Su, Zenan Li, Xian Zhang, Kaiyu Yang, and Xujie Si. 2024. A survey on deep learning for theorem proving. In *First Conference on Language Modeling*.

Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Yuting Zeng, Jiaxing Wang, Xingyu Wang, Hailong Yang, and Jing Li. 2025. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models. *Preprint*, arXiv:2409.17539.

Jianqiao Lu, Yingjia Wan, Yinya Huang, Jing Xiong, Zhengying Liu, and Zhijiang Guo. 2024a. Formalalign: Automated alignment evaluation for autoformalization. *Preprint*, arXiv:2410.10135.

Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen, Hui Jin, Jipeng Zhang, Haiming Wang, Zhicheng Yang, Jing Tang, and Zhijiang Guo. 2024b. Process-driven autoformalization in lean 4. *Preprint*, arXiv:2406.01940.

C. Maienborn, K. von Heusinger, and P. Portner. 2011. *Semantics: An International Handbook of Natural Language Meaning*. Number v. 1 in Handbooks of Linguistics and Communication Science. De Gruyter Mouton.

Aaron Meurer et al. 2017. SymPy: symbolic computing in Python. *PeerJ Comput. Sci.*, 3:e103.

Tobias Nipkow, Markus Wenzel, and Lawrence C Paulson. 2002. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer.

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. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

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.

Terence Parsons. 1990. *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press.

Lawrence Charles Paulson and Jasmin Christian Blanchette. 2012. Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. In *IWIL@LPAR*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *Preprint*, arXiv:2009.03393.

Seth Poulsen, Sami Sarsa, James Prather, Juho Leinonen, Brett A. Becker, Arto Hellas, Paul Denny, and Brent N. Reeves. 2024. Solving proof block problems using large language models. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2024, page 1063–1069, New York, NY, USA. Association for Computing Machinery.

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. 2025. Large language models meet symbolic provers for logical reasoning evaluation. *Preprint*, arXiv:2502.06563.

Xin Quan, Marco Valentino, Louise Dennis, and Andre Freitas. 2024a. Enhancing ethical explanations of large language models through iterative symbolic refinement. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–22, St. Julian's, Malta. Association for Computational Linguistics.

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. 2024b. 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.

Mohammad Raza and Natasa Milic-Frayling. 2025. Instantiation-based formalization of logical reasoning tasks using language models and logical solvers. *Preprint*, arXiv:2501.16961.

Amitayush Thakur, George Tsoukalas, Yeming Wen, Jimmy Xin, and Swarat Chaudhuri. 2024. An in-context learning agent for formal theorem-proving. *Preprint*, arXiv:2310.04353.

Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2020. A survey on explainability in machine reading comprehension. *Preprint*, arXiv:2010.00389.

Marco Valentino, Mokanarangan Thayaparan, and André Freitas. 2022. Case-based abductive natural language inference. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1556–1568, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Sean Welleck and Rahul Saha. 2023a. Llmstep: Llm proofstep suggestions in lean. *arXiv preprint arXiv:2310.18457*.

Sean Welleck and Rahul Saha. 2023b. Llmstep: Llm proofstep suggestions in lean. *Preprint*, arXiv:2310.18457.

Sarah Wiegreffe and Ana Marasović. 2021. Teach me to explain: A review of datasets for explainable natural language processing. *Preprint*, arXiv:2102.12060.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 32353–32368. Curran Associates, Inc.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024a. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *Preprint*, arXiv:2405.14333.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. 2024b. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *Preprint*, arXiv:2408.08152.

Lan Zhang, Xin Quan, and Andre Freitas. 2024. Consistent autoformalization for constructing mathematical libraries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4020–4033, Miami, Florida, USA. Association for Computational Linguistics.

Xueliang Zhao, Wenda Li, and Lingpeng Kong. 2023. Decomposing the enigma: Subgoal-based demonstration learning for formal theorem proving. *Preprint*, arXiv:2305.16366.

A Ablation study

Table 3 shows the overall results on the ablation study for all LLMs.

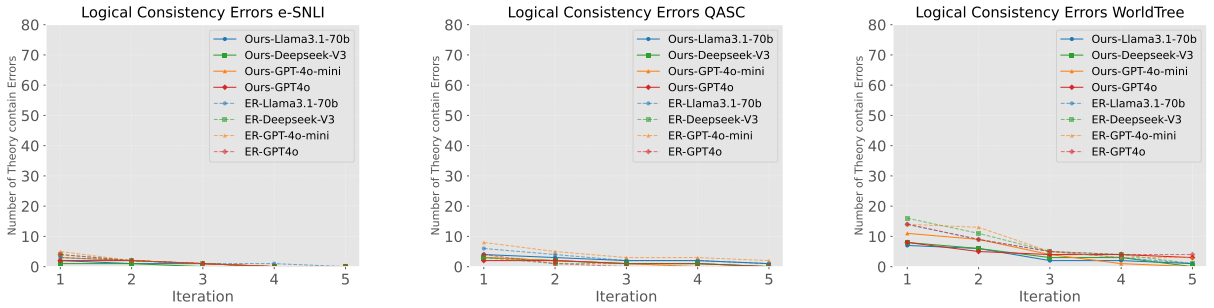
B Informalisation

We perform an autoformalisation process that transforms natural language sentences into Neo-Davidsonian event-based semantics by leveraging their underlying structure. One way to measure the faithfulness of this autoformalisation is to translate the constructed logical forms back into natural language and then compare the generated (informalised) sentences with the original ones using cosine similarity.

We employ a rule-based method to transform Neo-Davidsonian logical forms back into coherent natural language. First, we parse a logical form that may contain multiple conjuncts, typically connected by the logical "and" operator ($\wedge$). Each conjunct is treated as an atomic predicate with the general structure $\text{Predicate}(\text{arg}_1, \text{arg}_2, \dots)$. Once the form is separated into atomic predicates, we distinguish between role predicates (e.g., $\texttt{Agent}(e_1, x)$, $\texttt{Patient}(e_1, y)$) and entity-attribute predicates (e.g., $\texttt{Child}(x)$, $\texttt{Blonde}(x)$). The role predicates specify how each entity participates in the event (agent, patient, theme, location, etc.), while the attribute predicates detail intrinsic properties of those entities (for instance, "child," "blonde," "small," or "plastic").

After identifying these predicates, we group together all attributes describing the same entity variable. In particular, we parse the attributes from right to left, treating the rightmost attribute as the head noun and the preceding ones as adjectives. For example, if a single entity x is associated with $\texttt{Child}(x)$ and $\texttt{Blonde}(x)$, we combine those attributes to form a concise descriptor such as "blonde child." Likewise, if another entity y has attributes $\texttt{Plastic}(y)$ and $\texttt{Small}(y)$, we might call it "small plastic".

Next, we convert these role–entity pairings into simple event-level sentences. For each event e_i, we identify which entity is the `Agent` and which is the `Patient` (or any other role labels), then build a straightforward sentence. For instance, if x is



(a) (b) (c)

Figure 5: Number of theories that contain logical consistency error at each syntax error refinement stage.

```
Original Sentence: The boy is inside of the building.
Logical Form 1: ∃x y e. Boy(x) ∧ Building(y) ∧ Inside(e) ∧ Agent(e, x) ∧ Patient(e, y)
Informalised Sentence 1: Boy in side building.
Sentence Similarity: 0.9344

Logical Form 2: ∃x y e. Boy(x) ∧ Building(y) ∧ Inside(e) ∧ Agent(e, x)
Informalised Sentence 2: Boy in side.
Sentence Similarity: 0.8127
```

Figure 6: An example of the faithfulness between two informalised logical forms

"blonde child" and y is "small plastic item," the corresponding natural language description might by constructed from the event verb "Puts" as "blonde child puts small plastic item" The specific event verb ("puts," "picks," "hands over," etc.) would depend on how the event predicate itself is represented in the logical form.

In cases where the logical form contains implication, we divide the logical forms into sub-formulas. Complex operators and connectives (i.e. $\vee$) will be mapped carefully to their closest equivalents in English.

As shown in Figure 6, different formalised logical forms can affect the faithfulness of the autoformalisation. For instance, Logical Form 2 omits the *Patient* argument, causing the rule-based system to skip translating the predicate information for the *building* back into natural language, and thus producing an unfaithful representation.

C Datasets, LLMs and Theorem Prover

The datasets used in our experiments are sourced from open academic works and include samples from e-SNLI (Camburu et al., 2018), QASC (Khot et al., 2019), and WorldTree (Jansen et al., 2018). We employed Isabelle/HOL (Nipkow et al., 2002) as the theorem prover, which is distributed under the revised BSD license, and used Explanation-Refiner (Quan et al., 2024b) as our baseline work, which is under the MIT license. Additionally, we utilised API calls for GPT-4o (gpt-4o-2024-08-06) (OpenAI, 2023), GPT-4o-mini (gpt-4o-mini-2024-07-18) (OpenAI, 2023), Deepseek-V3 (Deepseek-V3-671b) (DeepSeek-AI, 2024), and Llama3.1-70b (LLama3.1-70b-Instruct) (Grattafiori et al., 2024). All temperature is set to 0.

D Runtime Examples

Tables 4, 5, and 6, together with Figures 7, 8, 9, 10, and 11, show the runtime examples for the e-SNLI, QASC, and WorldTree datasets, respectively.

E Prompts

Tables 7, 9, 8, and 10 show the prompts we used for syntactic parsing, logical proposition extraction, logical relation extraction, and proof construction. Complete prompts details can be found at https://github.com/neuro-symbolic-ai/faithful_and_robust_nli_refinement.

Algorithm 1: Deriving logical implications with SymPy

Input : logical_information: string with propositions and relations
Output : result: string with processed relations and implications

```
logical_props, logical_exprs ← ParseInput(logical_information)
if logical_exprs = ∅ then
    result ← format_propositions(logical_props)
    return result
else
    Initialise symbols_dict, symbol_meanings ← {}
    foreach (key, value) ∈ logical_props do
        sanitized_key ← sanitize(key)
        symbol ← create_symbol(sanitized_key)
        Update symbols_dict and symbol_meanings
    end foreach
    // Define SymPy logical operators dictionary
    logical_operators ← {
        symbols_dict, Not: SymPy negation,
        And: SymPy conjunction,
        Or: SymPy disjunction,
        Implies: SymPy implication,
        Equivalent: SymPy equivalence
    }
    propositions ← []
    initial_implications ← ∅
    foreach expr ∈ logical_exprs do
        expr ← replace_symbols(expr)
        // Evaluate using SymPy's logical operators
        prop ← evaluate_with_sympy(expr, logical_operators)
        // Apply SymPy's simplification rules
        simplified_prop ← sympy.simplify(prop)
        propositions.append(prop)
        initial_implications.add(simplified_prop)
    end foreach
    derived_implications ← ∅
    logical_atoms ← get_atoms(propositions)
    literals ← logical_atoms ∪ {¬atom | atom ∈ logical_atoms}
    // Use SymPy's satisfiability checker
    foreach (antecedent, consequent) ∈ literals × literals do
        if antecedent ≠ consequent then
            implication ← antecedent ⟹ consequent
            // Check using SymPy's logical rules
            is_new ← ¬equivalent_to_any(implication, initial_implications)
            is_valid ← check_entailment(propositions, implication)
            if is_new and is_valid then
                derived_implications.add(implication)
            end if
        end if
    end foreach
    result ← format_output(logical_props, logical_exprs, derived_implications)
    return result
end if
```

	e-SNLI			QASC			WorldTree		
	Init.	**Final**	**#Iter**	**Init.**	**Final**	**#Iter**	**Init.**	**Final**	**#Iter**
Ablations on our approach									
Llama3.1-70b (- logical relations)	34%	74%(−4%)	2.43	9%	58%(−10%)	2.94	7%	44%(−8%)	5.42
Llama3.1-70b (- **detailed feedback**)	32%	66%(−12%)	3.42	10%	34%(−**34**%)	3.64	5%	24%(−**28**%)	8.12
Llama3.1-70b (- refine quantifiers)	28%	77%(−1%)	2.18	9%	68%(−0%)	2.88	3%	50%(−2%)	4.52
Llama3.1-70b (- **refine syntax errors**)	18%	57%(−**21**%)	4.58	5%	53%(−15%)	4.47	3%	30%(−20%)	6.12
GPT-4o-mini (- logical relations)	27%	65%(−12%)	2.31	11%	57%(−14%)	4.12	6%	27%(−20%)	5.19
GPT-4o-mini (- detailed feedback)	30%	62%(−15%)	4.56	9%	46%(−25%)	3.87	3%	19%(−28%)	6.21
GPT-4o-mini (- refine quantifiers)	26%	78%(+4%)	2.10	5%	73%(+2%)	2.92	4%	46%(−1%)	5.13
GPT-4o-mini (- **refine syntax errors**)	15%	43%(−**34**%)	2.86	3%	34%(−**37**%)	3.65	3%	10%(−**37**%)	5.21
GPT-4o (- **logical relations**)	34%	74%(−**15**%)	2.24	12%	58%(−21%)	3.46	6%	38%(−18%)	4.36
GPT-4o (- **detailed feedback**)	35%	83%(−6%)	2.86	13%	56%(−**23**%)	4.45	5%	17%(−**39**%)	6.46
GPT-4o (- refine quantifiers)	34%	87%(−2%)	1.63	14%	83%(+4%)	2.89	7%	49%(−7%)	3.65
GPT-4o (- **refine syntax errors**)	21%	74%(−**15**%)	2.34	5%	58%(−21%)	4.11	2%	24%(−32%)	6.48
Deepseek-V3 (- logical relations)	39%	89%(−6%)	1.68	16%	77%(−13%)	2.64	10%	58%(−15%)	4.01
Deepseek-V3 (- **detailed feedback**)	31%	86%(−9%)	3.22	22%	69%(−21%)	4.12	6%	41%(−**32**%)	6.13
Deepseek-V3 (- refine quantifiers)	34%	96%(+1%)	1.64	14%	93%(+3%)	1.89	6%	70%(−3%)	3.23
Deepseek-V3 (- **refine syntax errors**)	28%	77%(−**18**%)	2.69	12%	68%(−**22**%)	2.84	4%	46%(−27%)	5.32

	e-SNLI			QASC			WorldTree		
	V.	**I.**	**Q.**	**V.**	**I.**	**Q.**	**V.**	**I.**	**Q.**
Explanation-Refiner									
Llama3.1-70b	24%	10%	10%	43%	12%	34%	45%	15%	27%
GPT-4o-mini	18%	8%	7%	41%	8%	32%	39%	9%	29%
GPT-4o	**9%**	**3%**	**6%**	**18%**	**9%**	**18%**	**33%**	**8%**	**16%**
Deepseek-V3	27%	**3%**	10%	34%	9%	25%	44%	23%	31%
Ablations									
Llama3.1-70b (- refine quantifiers)	23%	8%	13%(+3%)	39%	11%	43%(+9%)	43%	13%	36%(+9%)
Llama3.1-70b (- refine syntax errors)	41%(+17%)	10%	9%	53%(+10%)	8%	34%	65%(+20%)	11%	23%
GPT-4o-mini (- refine quantifiers)	14%	5%	11%(+4%)	35%	10%	47%(+15%)	41%	10%	34%(+5%)
GPT-4o-mini (- refine syntax errors)	39%(+21%)	4%	9%	63%(+21%)	7%	12%	55%(+16%)	7%	23%
GPT-4o (- refine quantifiers)	9%	2%	10%(+4%)	13%	2%	23%(+5%)	38%	6%	19%(+3%)
GPT-4o (- refine syntax errors)	16%(+7%)	1%	3%	38%(+20%)	3%	7%	56%(+23%)	3%	16%
Deepseek-V3 (- refine quantifiers)	25%	5%	16%(+6%)	31%	14%	35%(+10%)	32%	11%	38%(+7%)
Deepseek-V3 (- refine syntax errors)	38%(+11%)	4%	11%	51%(+17%)	9%	15%	67%(+23%)	21%	27%

Table 3: Top – Ablation study on the impacts of removing components on the analysis of number of explanation refined across three datasets. Bottom – Comparison of manually evaluated variable, implication, and quantifier errors in the autoformalisation process from a randomly sampled set of 100 Isabelle/HOL theories across all iterations for each LLM.

Dataset	**Problem**	**Explanation**	**Logic Info**	**Iteration**	**Validity**
e-SNLI	**Premise**: A smiling woman is playing the violin in front of a turquoise background. **Hypothesis**: A woman is playing an instrument.	A violin is an instrument.	**Logical Propositions:** A: a violin (from Explanatory Sentence 1) B: an instrument (from Explanatory Sentence 1) **Logical Relations:** Implies(A, B) Implies(a violin, an instrument) **Derived Implications:** none	0	Valid

Table 4: An example of a verified logically valid explanation in the e-SNLI dataset without refinement.

```
theory esnli_3_0
imports Main
...
consts
Violin :: "entity ⇒ bool"
Instrument :: "entity ⇒ bool"
Woman :: "entity ⇒ bool"
...
Patient :: "event ⇒ entity ⇒ bool"
InFrontOf :: "entity ⇒ entity ⇒ bool"

(* Explanation 1: A violin is an instrument. *)
axiomatization where
  explanation_1: "∀x. Violin x ⟶ Instrument x"

theorem hypothesis:
  (* Premise: A smiling woman is playing the violin in front of a turquoise background. *)
  assumes asm: "Woman x ∧ Violin y ∧ Background z ∧ Turquoise z ∧ Smiling x ∧ Playing e ∧ Agent e x ∧ Patient e y ∧
InFrontOf x z"
  (* Hypothesis: A woman is playing an instrument. *)
  shows "∃x y e. Woman x ∧ Instrument y ∧ Playing e ∧ Agent e x ∧ Patient e y"
  using assms explanation_1 by blast

end
```

Figure 7: An example of an autoformalised Isabelle/HOL theory in the e-SNLI dataset.

Dataset	Problem	Explanation	Logic Info	Iteration	Validity
QASC	**Hypothesis**: Some viruses have a coating of phospholipids.	Some viruses have an envelope of phospholipids and proteins. Proteins are sometimes coats of a virus.	**Logical Propositions:** A: some viruses have an envelope of phospholipids and proteins (from Explanatory Sentence 1) B: proteins are coats of a virus (from Explanatory Sentence 2) **Logical Relations:** none **Derived Implications:** none	0	Invalid
QASC	**Hypothesis**: Some viruses have a coating of phospholipids.	Some viruses have an envelope of phospholipids and proteins. Proteins are sometimes coats of a virus. An envelope can be considered a type of coating. Phospholipids are a component of the envelope of some viruses.	**Logical Propositions:** A: viruses have an envelope of phospholipids and proteins (from Explanatory Sentence 1) B: proteins are coats of a virus (from Explanatory Sentence 2) C: an envelope is a type of coating (from Explanatory Sentence 3) D: phospholipids are a component of the envelope (from Explanatory Sentence 4) **Logical Relations:** Implies(A, D) Implies(viruses have an envelope of phospholipids and proteins, phospholipids are a component of the envelope) **Derived Implications:** none	1	Valid

Table 5: An example of how an explanation in the QASC dataset can be refined in one iteration..

```
theory qasc_16_0
imports Main
...
consts
Virus :: "entity ⇒ bool"
...
Coat :: "entity ⇒ bool"
Coating :: "entity ⇒ bool"

(* Explanation 1: Some viruses have an envelope of phospholipids and proteins. *)
axiomatization where
  explanation_1: "∃x y z w e. Virus x ∧ Envelope y ∧ Phospholipids z ∧ Proteins w ∧ Have e ∧ Agent e x ∧ Patient e y ∧ Of y z ∧ Of y w"

(* Explanation 2: Proteins are sometimes coats of a virus. *)
axiomatization where
  explanation_2: "∃x y. Protein x ∧ Virus y ∧ Coat x ∧ Of x y"

theorem hypothesis:
  assumes asm: "Virus x ∧ Phospholipids z"
  (* Hypothesis: Some viruses have a coating of phospholipids. *)
  shows "∃x y z e. Virus x ∧ Coating y ∧ Phospholipids z ∧ Have e ∧ Agent e x ∧ Patient e y ∧ Of y z"
proof –
  (* From Explanation 1, we know that some viruses have an envelope of phospholipids and proteins. *)
  from explanation_1 obtain x y z w e where "Virus x ∧ Envelope y ∧ Phospholipids z ∧ Proteins w ∧ Have e ∧ Agent e x ∧ Patient e y ∧ Of y z ∧ Of y w" by blast
  (* We need to show that some viruses have a coating of phospholipids. *)
  (* Since an envelope can be considered a type of coating, we can infer that y is a coating. *)
  then have "Virus x ∧ Coating y ∧ Phospholipids z ∧ Have e ∧ Agent e x ∧ Patient e y ∧ Of y z" sledgehammer
  then show ?thesis <ATP>
qed

end
```

Figure 8: An example of an autoformalised Isabelle/HOL theory in the QASC dataset.

```
theory qasc_16_1
imports Main
...
consts
Virus :: "entity ⇒ bool"
...
PartOf :: "entity ⇒ entity ⇒ bool"

(* Explanation 1: Some viruses have an envelope of phospholipids and proteins. *)
axiomatization where
  explanation_1: "∃x y z w e. Virus x ∧ Envelope y ∧ Phospholipids z ∧ Proteins w ∧ Have e ∧ Agent e x ∧ Patient e y ∧ Component y z ∧ Component y w"

(* Explanation 2: Proteins are sometimes coats of a virus. *)
axiomatization where
  explanation_2: "∃x y. Protein x ∧ Virus y ∧ Coat x y"

(* Explanation 3: An envelope can be considered a type of coating. *)
axiomatization where
  explanation_3: "∀x. Envelope x ⟶ Coating x"

(* Explanation 4: Phospholipids are a component of the envelope of some viruses. *)
axiomatization where
  explanation_4: "∃x y z. Phospholipids x ∧ Envelope y ∧ Virus z ∧ Component y x ∧ PartOf y z"

theorem hypothesis:
  assumes asm: "Virus x ∧ Phospholipids z"
  (* Hypothesis: Some viruses have a coating of phospholipids. *)
  shows "∃x y z e. Virus x ∧ Coating y ∧ Phospholipids z ∧ Have e ∧ Agent e x ∧ Patient e y ∧ Component y z"
  using explanation_1 explanation_3 by blast

end
```

Figure 9: An example of an autoformalised Isabelle/HOL theory in the QASC dataset.

Dataset	Problem	Explanation	Logic Info	Iteration	Validity
WorldTree	**Hypothesis**: A forest fire would cause deer to die or leave a woodland.	wildfire is when a forest catches fire. fire causes harm to trees; to forests; to living things. a deer is a kind of animal. an animal is a kind of living thing. killing means harming something causing death. a deer lives in a forest. woodland means forest. natural disasters can cause animals to leave an environment. a wildfire is a kind of natural disaster. a forest is a kind of environment.	**Logical Propositions:** A: wildfire (from Explanatory Sentence 1) B: forest catches fire (from Explanatory Sentence 1) C: fire causes harm to trees (from Explanatory Sentence 2) D: fire causes harm to forests (from Explanatory Sentence 2) E: fire causes harm to living things (from Explanatory Sentence 2) F: deer (from Explanatory Sentence 3) ... Q: forest is a kind of environment (from Explanatory Sentence 10) **Logical Relations:** Equivalent(A, B) Implies(C, D) ... Implies(M, Q) **Derived Implications:** Implies(Not(O), Not(A)) Implies(B, A) ... Implies(L, Q)	0	Invalid
WorldTree	**Hypothesis**: A forest fire would cause deer to die or leave a woodland.	Woodland means forest. A wildfire is a kind of natural disaster. A deer is a kind of animal. An animal is a kind of living thing. A forest is a kind of environment. Natural disasters can cause animals to leave an environment. Fire causes harm to trees; to forests; to living things. A forest fire is a kind of wildfire. A deer lives in a forest.	**Logical Propositions:** A: woodland (from Explanatory Sentence 1) B: forest (from Explanatory Sentence 1) ... L: lives in a forest (from Explanatory Sentence 9) **Logical Relations:** Equivalent(A, B) Implies(C, D) ... Implies(I, Implies(B, Not(G))) **Derived Implications:** Implies(E, Not(D)) Implies(E, Not(K)) ... Implies(Not(G), Not(E))	1	Valid

Table 6: An example of how an explanation in the WorldTree dataset can be refined in one iteration.

```
theory worldtree_19_0
imports Main
...
Die :: "event ⇒ bool"

(* Explanation 1: Wildfire is when a forest catches fire. *)
axiomatization where
  explanation_1: "∀x. Wildfire x ⟷ (∃y e z. Forest y ∧ Catch e ∧ Agent e y ∧ Patient e z ∧ Fire z)"

(* Explanation 2: Fire causes harm to trees; to forests; to living things. *)
axiomatization where
  explanation_2: "∀x y e. Fire x ∧ Harm e ∧ Agent e x ⟶ (Patient e y ∧ (Tree y ∨ Forest y ∨ LivingThing y))"

(* Explanation 3: A deer is a kind of animal. *)
axiomatization where
  explanation_3: "∀x. Deer x ⟶ Animal x"

(* Explanation 4: An animal is a kind of living thing. *)
axiomatization where
  explanation_4: "∀x. Animal x ⟶ LivingThing x"

(* Explanation 5: Killing means harming something causing death. *)
axiomatization where
  explanation_5: "∀e1 e2 x y. Killing e1 ⟷ (Harming e2 ∧ Agent e2 x ∧ Cause e2 ∧ Patient e2 y ∧ Death y)"

(* Explanation 6: A deer lives in a forest. *)
axiomatization where
  explanation_6: "∃x y e. Deer x ∧ Forest y ∧ Live e ∧ Agent e x ∧ In x y"

(* Explanation 7: Woodland means forest. *)
axiomatization where
  explanation_7: "∀x. Woodland x ⟷ Forest x"

(* Explanation 8: Natural disasters can cause animals to leave an environment. *)
axiomatization where
  explanation_8: "∀x y z. NaturalDisaster x ∧ Animal y ∧ Environment z ⟶ (∃e1 e2. Cause e1 ∧ Agent e1 x ∧ Leave e2 ∧ Agent e2 y ∧ Patient e2 z)"

(* Explanation 9: A wildfire is a kind of natural disaster. *)
axiomatization where
  explanation_9: "∀x. Wildfire x ⟶ NaturalDisaster x"

(* Explanation 10: A forest is a kind of environment. *)
axiomatization where
  explanation_10: "∀x. Forest x ⟶ Environment x"

theorem hypothesis:
  assumes asm: "ForestFire x ∧ Deer y ∧ Woodland z"
  (* Hypothesis: A forest fire would cause deer to die or leave a woodland. *)
  shows "∃e1 e2. (Die e1 ∧ Agent e1 y) ∨ (Leave e2 ∧ Agent e2 y ∧ Patient e2 z)"
proof -
  (* From the premise, we have known information about ForestFire, Deer, and Woodland. *)
  from asm have "ForestFire x" and "Deer y" and "Woodland z" apply simp
  (* Explanation 7: Woodland means forest. *)
  (* Using the equivalence between Woodland and Forest, we can infer Forest z. *)
  from `Woodland z` have "Forest z" apply (simp add: assms)
  (* Explanation 9: A wildfire is a kind of natural disaster. *)
  (* Since ForestFire is a type of Wildfire, we can infer that it is a NaturalDisaster. *)
  from `ForestFire x` have "Wildfire x" by (simp add: assms)
  then have "NaturalDisaster x" sledgehammer
  (* Explanation 3 and 4: A deer is a kind of animal, and an animal is a kind of living thing. *)
  (* We can infer that Deer y is an Animal and a LivingThing. *)
  from `Deer y` have "Animal y" <ATP>
  then have "LivingThing y" <ATP>
  (* Explanation 10: A forest is a kind of environment. *)
  (* Since we have Forest z, we can infer that it is an Environment. *)
  from `Forest z` have "Environment z" <ATP>
  (* Explanation 8: Natural disasters can cause animals to leave an environment. *)
  (* Since we have a NaturalDisaster x, Animal y, and Environment z, we can infer that the animal may leave the environment. *)
  from `NaturalDisaster x` and `Animal y` and `Environment z` have "∃e2. Leave e2 ∧ Agent e2 y ∧ Patient e2 z" <ATP>
  (* We have shown that a forest fire would cause deer to leave a woodland. *)
  then show ?thesis <ATP>
qed
end
```

Figure 10: An example of an autoformalised Isabelle/HOL theory in the WorldTree dataset.

Table 7: Prompts used for syntactic parsing.

SYSTEM: You are an expert in linguistics. You will be provided with some sentences, please do a syntactic parse for each word in that sentence. Some instructions:
1. You must give me the answer for all provided sentences. 2. Do not add any notes. 3. If no premise sentence provided, include it in the answer as none. 4. Retain the answer words in their original form within the provided sentence.
USER: Here are some examples:
Hypothesis Sentence:
1. A woman is playing an instrument.
Subject: A woman
Verb Phrase: is playing an instrument
- Main Verb: playing
- Auxiliary Verb: is
Direct Object: an instrument
...
<<<<<<<<<<<<<
Provided sentences:

Answer:

```
theory worldtree_19_1
imports Main
...
Die :: "event ⇒ bool"

(* Explanation 1: Woodland means forest. *)
axiomatization where
  explanation_1: "∀x. Woodland x ⟷ Forest x"

(* Explanation 2: A wildfire is a kind of natural disaster. *)
axiomatization where
  explanation_2: "∀x. Wildfire x ⟶ NaturalDisaster x"

(* Explanation 3: A deer is a kind of animal. *)
axiomatization where
  explanation_3: "∀x. Deer x ⟶ Animal x"

(* Explanation 4: An animal is a kind of living thing. *)
axiomatization where
  explanation_4: "∀x. Animal x ⟶ LivingThing x"

(* Explanation 5: A forest is a kind of environment. *)
axiomatization where
  explanation_5: "∀x. Forest x ⟶ Environment x"

(* Explanation 6: Natural disasters can cause animals to leave an environment. *)
axiomatization where
  explanation_6: "∀x y z. NaturalDisaster x ∧ Animal y ∧ Environment z ⟶ (∃e. Leave e ∧ Agent e y ∧ Patient e z)"

(* Explanation 7: Fire causes harm to trees; to forests; to living things. *)
axiomatization where
  explanation_7: "∀x y z w. Fire x ∧ (∃e. Harm e ∧ (Tree y ∨ Forest z ∨ LivingThing w) ⟶ (Patient e y ∨ Patient e z ∨ Patient e w))"

(* Explanation 8: A forest fire is a kind of wildfire. *)
axiomatization where
  explanation_8: "∀x. ForestFire x ⟶ Wildfire x"

(* Explanation 9: A deer lives in a forest. *)
axiomatization where
  explanation_9: "∃x y e. Deer x ∧ Forest y ∧ Lives e ∧ Agent e x ∧ In x y"

theorem hypothesis:
  assumes asm: "ForestFire x ∧ Deer y ∧ Woodland z"
  (* Hypothesis: A forest fire would cause deer to die or leave a woodland. *)
  shows "∃e1 e2. (Die e1 ∧ Agent e1 y) ∨ (Leave e2 ∧ Agent e2 y ∧ Patient e2 z)"
  by (meson assms explanation_1 explanation_2 explanation_3 explanation_5 explanation_6 explanation_8)

end
```

Figure 11: An example of an autoformalised Isabelle/HOL theory in the WorldTree dataset.

Table 8: Prompts used for extracting logical propositions and relations

SYSTEM: You are an expert in symbolic reasoning. You will be provided with an explanation. You need to extract the logical propositions and the corresponding logical relations from the explanation.
USER: Here are some examples:
Provided Explanatory Sentences:
Explanatory Sentence 1: If it is raining, the grass will be wet.
Explanatory Sentence 2: Having a picnic is equivalent to having a meal on the grass.

Answer:
Logical Propositions:
A: it is raining (from Explanatory Sentence 1)
B: the grass will be wet (from Explanatory Sentence 1)
C: having a picnic (from Explanatory Sentence 2)
D: having a meal on the grass (from Explanatory Sentence 2)

Logical Relations:
Implies(A, B): $A \rightarrow B$
Equivalent(C, D): $C \leftrightarrow D$

<<<<<<<<<<<<<
Provided Explanatory Sentences:

Answer:

Logical Propositions:

Logical Relations:

Table 9: Prompts used for refining quantifiers.

SYSTEM: You are an expert in semantics, formal language and neo-davidsonian event semantics. You will be provided with some sentences. These sentences have been transferred into Isabelle/HOL symbolic language. However, the quantifiers in the logical form may not be defined correctly.There might be missing variables after the quantifiers for arguments inside the parentheses of the predicate-argument forms of an axiom or a theorem. The quantifier may not reflect to real-world knowledge. Refine the logical forms if there are any quantifiers that are not defined correctly.

Here are some examples:
Provided Iabelle code:
(* Explanation 1: Many consumers feed at more than one trophic level. *)
axiomatization where
explanation_1: "∀x e. Consumer x ⟶ (Feed e ∧ Agent e x ∧ At e y ∧ $MoreThanOneTrophicLevel$ y)"

Answer:
Explanation 1 states "Many consumers" and in real-world knowledge, some consumers are omnivores or generalists that feed across multiple trophic levels, but it use the universal quantifier '∀' in explanation_1. We should use the existential quantifier '∃' instead.
For the quantifier variables in explanation_1, the variable 'y' is missing.
...
<<<<<<<<<<<<<
Strictly follow my instructions.

Provided Isabelle code:

Answer:

Table 10: Prompts used for building proofs.

SYSTEM: You are an expert in Isabelle theorem prover, first-order logic and Davidsonian event semantics. You will be provided with premises, explanations and hypothesis sentences. You will be provided with an Isabelle code which consists of some axioms, a theorem hypothesis that needs to be proven. The logical form of axioms indicates some explanation sentences, the logical form after "assume asm:" indicates a premise sentence and the logical form after "shows" indicates a hypothesis sentence. The natural language form is stated as the comments. You will be provided with some logical propositions, logical relations and derived logical rules from the explanation sentences to help you construct the proof. You need to construct a proof about how to prove the theorem hypothesis in "proof -" and "qed" sections using the premise (logical form after "assume asm:") and explanations (axioms). The proof should be derived from the premise and explanation sentences. You don't need to state the automated theorem prover you will need to use. You just need to write a proof sketch.
Some instructions:
1. 'sorry' and 'fix' command is not allowed.
...
5. leave the automated theorem prover and proof tactic as <ATP>
...
<<<<<<<<<<<<<<
Strictly follow my instructions.

Premise Sentence:

Explanation Sentences:

Hypothesis Sentence:

Provided Isabelle Code:

Logical Information:

Known Information:

Try to prove:

Answer: