Verification and Refinement of Natural Language Explanations through LLM-Symbolic Theorem Proving

Anonymous ACL submission

Abstract

Natural language explanations represent a proxy for evaluating explainable and multi-step Natural Language Inference (NLI) models. However, assessing the validity of explanations for NLI is challenging as it typically involves the crowd-sourcing of apposite datasets, a process that is time-consuming and prone to logical errors. To address existing limitations, this paper investigates the verification and refinement of natural language explanations through the integration of Large Language Models (LLMs) and Theorem Provers (TPs). Specifically, we present a neuro-symbolic framework, named Explanation-Refiner, that integrates TPs with LLMs to generate and formalise explanatory sentences and suggest potential inference strategies for NLI. In turn, the TP is employed to provide formal guarantees on the logical validity of the explanations and to generate feedback for subsequent improvements. We demonstrate how Explanation-Refiner can be jointly used to evaluate explanatory reasoning, autoformalisation, and error correction mechanisms of state-of-the-art LLMs as well as to automatically enhance the quality of explanations of variable complexity in different domains.[1]

1 Introduction

A recent line of research in Natural Language Inference (NLI) focuses on developing models capable of generating natural language explanations in support of their predictions (Thayaparan et al., 2021; Chen et al., 2021; Valentino et al., 2022; Bostrom et al., 2022; Weir et al., 2023). Since natural language explanations can be used as a proxy to evaluate the underlying reasoning process of NLI models (Kumar and Talukdar, 2020; Zhao and Vydiswaran, 2021; Chen et al., 2021), researchers have proposed different methods for assessing their intrinsic quality (Wiegreffe and Marasovic, 2021; Camburu et al., 2020; Valentino et al., 2021; Atanasova et al., 2023; Quan et al., 2024; Dalal et al., 2024), including the adoption of language generation metrics for a direct comparison between models' generated explanations and human-annotated explanations.

However, this process is subject to different types of limitations. First, the use of language generation metrics requires the crowd-sourcing of explanation corpora to augment existing NLI datasets (Wiegreffe and Marasovic, 2021), a process that is time-consuming and susceptible to errors (Liu et al., 2022; Zhao et al., 2023; Valentino et al., 2021). Second, language generation metrics have been shown to fail capturing fine-grained properties that are fundamental for NLI such as logical reasoning, faithfulness, and robustness (Atanasova et al., 2023; Camburu et al., 2020; Chan et al., 2022; Quan et al., 2024). Third, human explanations in NLI datasets tend to be incomplete and contain logical errors that could heavily bias the evaluation (Elazar et al., 2021; Valentino et al., 2021).

In this paper, we investigate the integration of state-of-the-art LLM-based explanation generation models for NLI with external logical solvers to jointly evaluate explanatory reasoning (Pan et al., 2023a; Olausson et al., 2023; Jiang et al., 2024b) and enhance the quality of crowd-sourced explanations. In particular, we present a neuro-symbolic framework, named Explanation-Refiner, that integrates a Theorem Prover (TP) with Large Language Models (LLMs) to investigate the following research questions: *RQ1: "Can the integration of LLMs and TPs provide a mechanism for automatic verification and refinement of natural language explanations?"; RQ2: "Can the integration of LLMs and TPs improve the logical validity of human-annotated explanations?"; RQ3: "To what extent are state-of-the-art LLMs capable of explanatory reasoning, autoformalisation, and error correction for NLI in different domains?"*. To answer these questions, Explanation-Refiner employs LLMs to generate and formalise explanatory sentences and

[1] Code and data are available at: Anonymous github link

to suggest potential inference strategies for building non-redundant, complete, and logically valid explanations for NLI. In turn, the TP is adopted to verify the validity of the explanations through the construction of deductive proofs and the generation of fine-grained feedback for LLMs.

We instantiate Explanation-Refiner with state-of-the-art LLMs (i.e., GPT-4 (OpenAI, 2023), GPT-3.5 (Brown et al., 2020), LLama (Touvron et al., 2023), and Mistral (Jiang et al., 2024a)) and the Isabelle/HOL theorem prover (Nipkow et al., 2002) utilising Neo-Davidsonian event semantics (Parsons, 1990) coupled with First-Order Logic (FOL) to effectively and systematically translate natural language sentences into logical forms.

Our empirical analysis carried out on three NLI datasets of variable complexity (i.e., e-SNLI (Camburu et al., 2018), QASC (Khot et al., 2019), and WorldTree (Jansen et al., 2018)) reveals that external feedback from TPs is effective in improving the quality of natural language explanations, leading to an increase in logical validity using GPT-4 from 36% to 84%, 12% to 55%, and 2% to 37% (on e-SNLI, QASC, and WorldTree respectively). At the same time, the results demonstrate that integrating external TPs with LLMs can reduce errors in autoformalisation, with an average reduction of syntax errors of 68.67%, 62.31%, and 55.17%. Finally, we found notable differences in performance across LLMs and NLI datasets, with closed-sourced LLMs (i.e., GPT-4 and GPT-3.5) significantly outperforming open-source models (i.e., Mistral and LLama) on both explanatory reasoning and autoformalisation, along with a shared tendency of LLMs to struggle with increasing explanation complexity.

To summarise, the main contributions of this paper are:

1. We introduce *Explanation-Refiner*, a novel neuro-symbolic framework that integrates LLMs with an external theorem prover. This framework automatically verifies and refines explanatory sentences in NLI tasks using an objective external feedback.

2. We utilise Neo-Davidsonian event semantics coupled with FOL to effectively translate natural language sentences into logical forms to minimise semantic information loss. Additionally, we introduce a novel method that leverages a theorem prover and a proof assistant for verifying NLI explanations and a syntactic refiner to minimise syntax errors in responses generated by LLMs.

3. We conduct a comprehensive series of experiments with *Explanation-Refiner* across five LLMs and three datasets, including 1 to 16 explanatory sentences, covering tasks from textual entailment to complex multiple-choice question answering in various domains.

4. We perform extensive analyses to explore the explanation refinement process, delving into the LLMs' inference capabilities and revealing the strengths and limitations of different models in producing verifiable, explainable logical reasoning for NLI.

2 Explanation Verification and Refinement

Explanation-based NLI is widely adopted to evaluate the reasoning process of multi-step inference models via the construction of natural language explanations. In this work, we refer to the following formalisation for Explanation-based NLI: given a premise sentence p_i, a hypothesis sentence h_i, and an explanation E_i consisting of a set of facts $\{f_1, f_2, ..., f_n\}$, the explanation E_i is logically valid if and only if the entailment $p_i \cup E_i \models h_i$ holds. This entailment is considered verifiable if $\{p_i, E_i, h_i\}$ can be translated into a set of logical formulae Φ that compose a theory Θ. The validity of the theory Θ is subsequently determined by a theorem prover, verifying whether $\Theta \vDash \psi$, where ψ represents a logical consequence derived from the logical form of h_i.

In this paper, we aim to automatically verify the logical validity of an explanation E_i. To this end, if $\Theta \vDash \psi$ is rejected by the theorem prover, a further refinement stage should be initiated to refine the facts $\{f_1, f_2, ..., f_n\}$ based on external feedback, resulting in an updated explanation E_i'. Thus, an explanation is accepted if all the facts are logically consistent, complementary and non-redundant to support the derivation.

3 Explanation-Refiner

To verify the logical validity and refine any logical errors in explanatory sentences for NLI tasks, we present a neuro-symbolic framework, Explanation-Refiner, to iteratively check and refine the explanation E_i based on external feedback. Figure 1 shows an overview of our proposed framework.

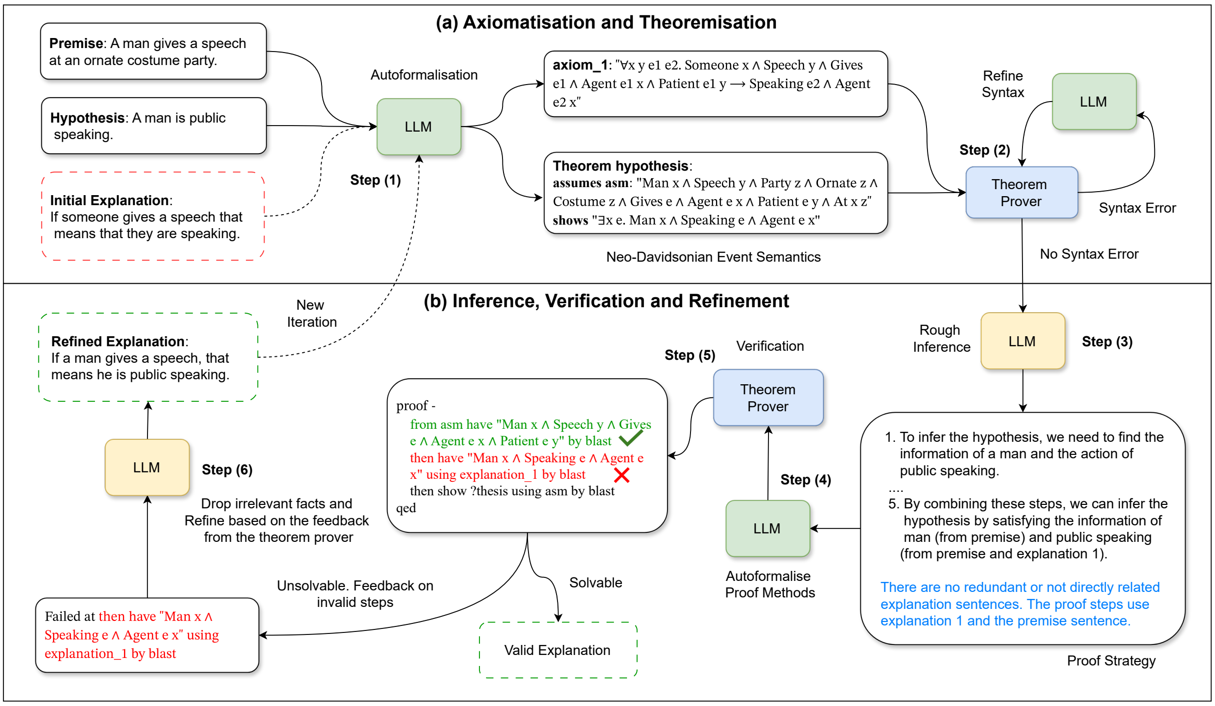


Figure 1: The overall pipeline of Explanation-Refiner: An NLI problem is converted into axioms and theorems for a theorem prover, along with some proof steps derived from a preliminary inference. In case the proof fails (logically invalid), the erroneous steps along with the constructed proof strategy are used as feedback to refine the explanation in a new iteration.

Given an NLI task, to evaluate the logical validity of the entailment, the LLM is prompted to perform an autoformalisation process that transforms natural language sentences into formal language represented in the form of an Isabelle/HOL theory. Each fact $f \in E_i$ is converted into an axiom a_i, where each a_i is an element of the set $A = \{a_1, a_2, ..., a_n\}$. The premise p_i and corresponding hypothesis h_i, is converted into a theorem for proving $p_i \wedge B \rightarrow h_i$, where $B \subseteq A$. A syntax refinement mechanism is subsequently applied to the previously transferred symbolic forms. The theorem prover is implemented as a checker to identify any syntax errors and provide these error details as feedback to an LLM, enabling the LLM to iteratively correct the syntax errors over a fixed number of iterations, denoted by t.

We can then perform automated reasoning via the theorem prover. To this end, in step 3 we use the LLM to generate a rough inference that states a preliminary proof strategy in natural language and elicit the facts $f \in E_i$ which are sufficient and necessary for entailing the hypothesis h_i. Based on this preliminary proof strategy, the LLM is prompted to construct and formalise the proof steps for proving the theorem. In step 5, the theorem prover will verify the constructed theory by attempting to prove the theorem. If it is solvable, we consider it a logically valid explanation. If the prover failed at one of the proof steps, we adopt the failed steps along with the applied axioms $B \subseteq A$ as an external feedback for the LLM. This feedback is used to refine the logical errors and consequently refine the facts $f \in E_i$.

3.1 Autoformalisation

In order to formally verify the logical validity of the explanations, we adopted Neo-Davidsonian event-based semantics and FOL.

Neo-Davidsonian Event Semantics Preventing the loss of semantic information during the representation of natural language sentences in logical forms, such as FOL, poses significant challenges when using LLMs, particularly with long and complex sentences that are crucial for logical reasoning (Olausson et al., 2023). Neo-Davidsonian event semantics (Parsons, 1990) utilises event variables to represent the verb predicates and their corresponding object arguments as semantic roles. This approach establishes a predicate-argument structure that preserves the information content and faithfulness of complex sentences, closer to the surface

```
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"
proof -
  from asm have "Woman x ∧ Violin y ∧ Playing e ∧ Agent e x ∧ Patient e y" by blast
  then have "Woman x ∧ Instrument y ∧ Playing e ∧ Agent e x ∧ Patient e y" using explanation_1 by
    blast
  then show ?thesis using asm by blast
qed
```

Figure 2: An example of representing the premise and hypothesis sentences in Isabelle/HOL theorem includes a proof constructed by the LLM for verifying the hypothesis.

form of the sentence (Quan et al., 2024). For example, the sentence 'A wolf eating a sheep is an example of a predator hunting prey' can be formalised as follows:

$$\begin{aligned}&\forall x y e_1 (\text{wolf}(x) \wedge \text{sheep}(y) \wedge \text{eating}(e_1)\\&\quad \wedge \text{agent}(e_1, x) \wedge \text{patient}(e_1, y) \rightarrow\\&\quad (\exists e_2\ \text{predator}(x) \wedge \text{prey}(y) \wedge\\&\quad \text{hunting}(e_2) \wedge \text{agent}(e_2, x) \wedge\\&\quad \text{patient}(e_2, y) \wedge \text{example}(e_1, e_2)))\end{aligned} \tag{1}$$

In 1, the verbs are represented as the events 'eating' and 'hunting,' where the agent and patient arguments correspond to the entities performing and receiving the actions within these events, respectively. The logical form example(e_1, e_2) explicitly captures the semantic meaning of this sentence: the event of a wolf eating a sheep as an exemplar of a predator hunting prey. Similarly, whenever there are no action verbs involved in a sentence, we utilise FOL to represent the static or descriptive aspects. For instance:

$$\forall x (\text{gravity}(x) \rightarrow \text{force}(x)) \tag{2}$$

$$\forall x y (\text{greater}(x, y) \rightarrow \text{larger}(x, y)) \tag{3}$$

The above logical forms correspond to the sentences 'gravity is a kind of force' and 'greater means larger'.

Isabelle/HOL Theory Construction A theory script for the Isabelle/HOL theorem prover contains theorems that need to be proven from some axioms. Therefore, we adopt the sentences in an explanation to construct the set of axioms. For instance:

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

In addition, as illustrated in Figure 2, both the premise and the hypothesis constitute parts of the theorem to be proven. In particular, the 'assumes asm' clause includes unquantified, specific propositions or conjunctions of propositions which are recognised as known truths (i.e., premises). On the other hand, the 'show' clause denotes the conclusion (i.e., hypothesis) for which we seek to build a proof through logical deductions based on the assumed propositions and axioms.

Syntax Error Refiner Recent studies (Gou et al., 2024; Olausson et al., 2023) have revealed persistent syntax errors when prompting LLMs for code and symbolic form generation tasks. We categorised the syntax errors into two distinct subdomains based on feedback from Isabelle: type unification errors and other syntax errors. Type unification errors primarily arise from mismatches between declared and actual argument types in logical clauses. Other syntax errors typically involve missing brackets, undefined entity names, or invalid logical symbols. Our process involves using Isabelle to identify syntax errors in the transferred theory, extracting these error messages, and then prompting the LLM with these messages along with few-shot examples. This guides the model on how to correct each type of syntax error over a series of iterations, allowing for continuous verification and refinement. Details of the autoformalisation prompts are described in Appendix A.4.1.

3.2 Proof Construction

A proof provides a detailed, step-by-step strategy that elucidates the logical connections and unification among axioms to support the reasoning process aimed at achieving the solver's goal. Initially, we prompt the LLM to create a preliminary proof in natural language to assess how it infers the hypothesis and to identify which explanatory sentences are relevant, redundant, or unrelated. Based on this initial inference, we then guide the LLM to develop a formal proof (figure 2) that utilises Isabelle/HOL to verify the explanatory sentences (axioms) that are required to derive the hypothesis. The general proof steps generated by an LLM are in the format 'show X using Y by Z', where the theorem prover is asked to prove X given the assumptions Y, using the automated proof tactic Z. The proof tactic often applied is 'blast', which is one of Isabelle's powerful FOL theorem proving tactics, enabling efficient and automated proof discovery across a range of logical forms (Paulson, 1999). Additional details of the proof construction process and the prompts used to guide the LLMs are described in Appendix A.4.2.

3.3 Verification and Refinement

Finally, the constructed theory, which includes axioms, theorems, and proof steps, is submitted to the theorem prover for verification. If the theory is validated, it outputs a logically sound explanation. If the proof fails or timeouts, we extract the first error from the solver's error message, identify the corresponding proof step, and locate the related explanatory sentences (axioms) from the theory. We begin by removing redundant and irrelevant facts that are not present in the preceding Isabelle/HOL proof steps or are declared as such in the text inference strategy. Then, we prompt the LLM to refine the explanatory sentences by providing it with the error message, the failed proof step, the associated proof strategy, and the relevant explanatory sentences for further iteration. This process is iterative and progressive; with each iteration, the framework addresses one or more logical errors, continually refining the explanatory sentences to ultimately yield a logically valid and verifiable explanation. Additional details on the prompts used for refinement are described in Appendix A.4.3.

4 Empirical Evaluation

4.1 Datasets

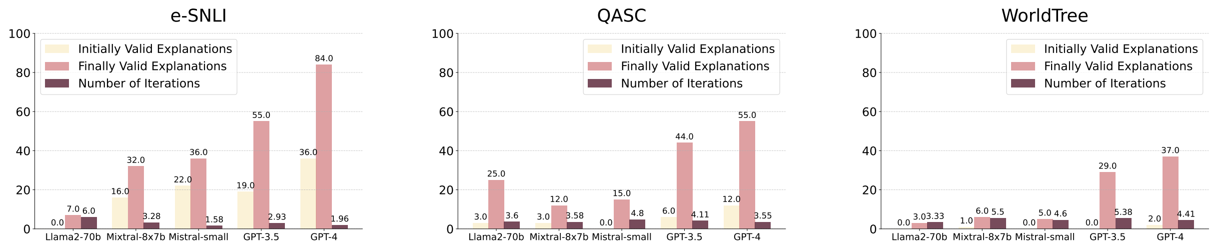
We adopted three different NLI datasets for evaluation: e-SNLI, QASC, and WorldTree, using a total of 300 samples selected via the sampling strategy defined in (Valentino et al., 2021), which maximises representativeness and mutual exclusivity across syntactic and semantic features expressed in the datasets. For multiple-choice question answering, the task includes a question q accompanied by a set of candidate answers $C = \{c_1, c_2, ..., c_n\}$, with c_i identified as the correct answer. To cast this problem into NLI, we simply convert q and the correct answer c_i into a hypothesis h_i. On the other hand, the question's context, if present, is used to build the premise p_i.

4.2 Models

To integrate Isabelle/HOL as a real-time verification tool with LLMs, we employ a Python client (Shminke, 2022) as TCP (Transmission Control Protocol) client to configure Isabelle/HOL as a server. This enables the communication of the constructed theory files and the extraction of the response messages from Isabelle. We conducted experiments using five LLMs within the proposed framework. The models include two open-sourced models: Llama2-70b (Touvron et al., 2023) and Mixtral-8x7b (Jiang et al., 2024a), as well as Mistral-small (mistral-small-latest) (Mistral AI, 2024), GPT-3.5 (gpt-3.5-turbo) (Brown et al., 2020), and GPT-4 (gpt-4-0613) (OpenAI, 2023).

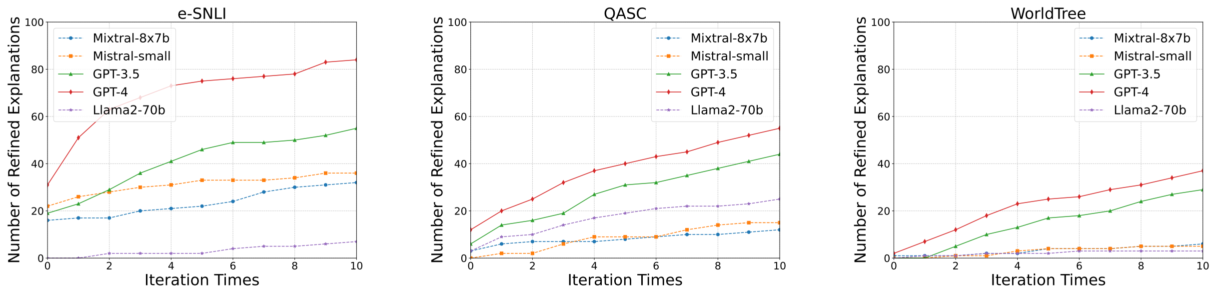
4.3 Results

Detailed feedback from an external theorem prover effectively guides LLMs in verifying and refining explanations for NLI. To assess the effectiveness of employing an external theorem prover to verify and refine explanations in NLI tasks, we conducted a comparative analysis across various LLMs (Figure 3). The initially valid explanations represent the percentage of explanations that can be verified as logically valid without any further iteration. Although the initial verification results varied among different models, all LLMs demonstrated a consistent improvement in refining the logical validity of the explanations. This process highlights the positive impact of the external feedback but also shows significant differences between models. We found that lower rates of initial valid explanations often resulted from syntactic errors, which impeded the theorem prover's ability to generate proofs. Despite this initial variability, all models demonstrate a consistent improvement in the refinement process across the datasets. Notably, GPT-4 outperformed other models, improving the validity of explanations by 48%, 43%, and 35% across the three datasets, respectively, within a maximum number of ten iterations (Figure 3). Figure 4 shows the number of explanations refined



(a) (b) (c)

Figure 3: The initial and final number of logically valid explanations, along with the average iteration times required to refine an explanation for each LLM



(a) (b) (c)

Figure 4: Number of successfully refined explanations at each iteration step.

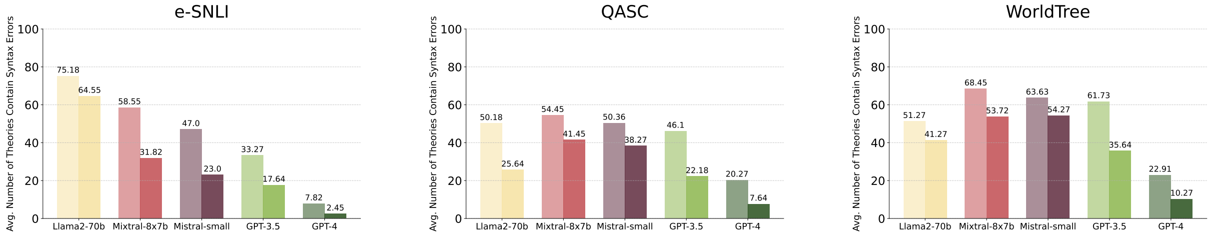
at each iteration across the e-SNLI, QASC, and WorldTree datasets. On average, we found that an increasing number of iterations leads to increasing refinement, with models requiring an average of five iterations across the datasets.

Explanation length/complexity impacts formalisation and verification. The e-SNLI dataset, which includes only a single explanatory sentence per example, shows the best overall performance. In contrast, the multiple-choice question answering datasets, QASC and WorldTree, exhibit comparatively lower performance. QASC typically contains 2 explanatory sentences, while WorldTree ranges from 1 to 16 sentences. As the number of explanatory sentences increases, so does the complexity of the logical reasoning required. The WorldTree dataset, in particular, poses the greatest challenge due to its demand for multi-hop inference strategies. Models show lower refining performance in WorldTree when compared to e-SNLI and QASC, with only 3%, 5%, and 5% of Llama-70b, Mixtral-8x7b, and Mistral-small explanations being refined in WorldTree. Meanwhile, 29% and 35% of explanations are refined by GPT-3.5 and GPT-4 in WorldTree, respectively. This process involves synthesising multiple explanatory sentences to fulfill sub-goals, which must then be integrated to meet the overall hypothesis goal.

Iterative and categorical refinement can monotonically reduce syntax errors in responses generated by LLMs. To evaluate the syntax error refinement stage, we quantified the presence of syntax errors in the Isabelle theories both before and after the iterative refinement process. After a maximum of three iterations, all models showed significant reductions, with maximum reductions of 68.67%, 62.31%, and 55.17% from 7.82 to 2.45, 20.27 to 7.64, and 22.91 to 10.27 across the three respective datasets (see Figure 5). While models like Llama2-70b and Mixtral-8x7b still exhibit some syntax errors in the refined theories' code, this is primarily due to their inability to perform complex autoformalisation, especially for multiple and more complex explanatory sentences such as those in the WorldTree dataset. This result is consistent with the percentage of explanations that were successfully refined across the models, which suggests that the autoformalisation process plays a critical role in the models' logical reasoning capability.

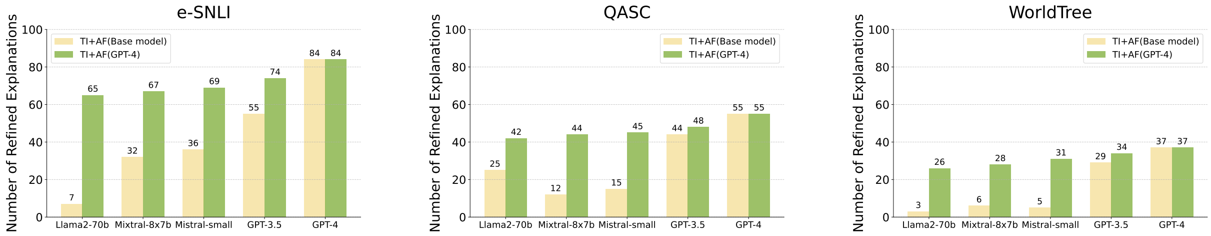
4.4 Ablation Study

We conducted an ablation study to further evaluate and disentangle the impact of autoformalisation on performance. To this end, we adopted GPT-4 exclusively for the autoformalisation component, while retaining the original models for explanation refine-



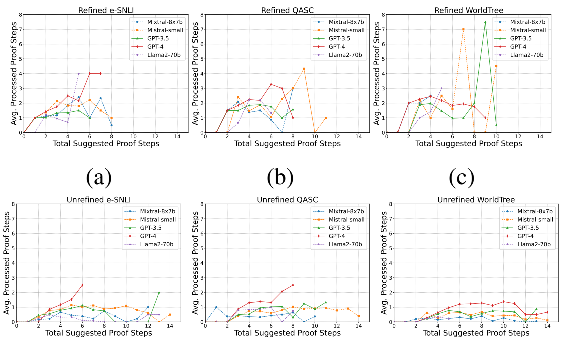
(a) (b) (c)

Figure 5: The average number of theories containing syntactic errors before and after the syntax refinement process



(a) (b) (c)

Figure 6: AF represents the autoformalisation components, and TI represents the textual inference components. TI+AF (Base Model) indicates the use of the base model for both the autoformalisation and textual inference components. TI+AF (GPT-4) indicates the use of GPT-4 for the autoformalisation components, while the base model is used for textual inference.



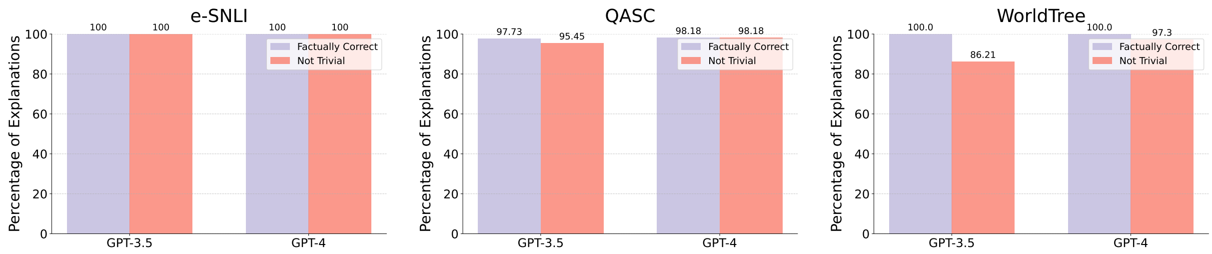
(a) (b) (c)

(d) (e) (f)

Figure 7: Average of proof steps processed by the proof assistant against the total proof steps suggested by the LLMs in refined and unrefined explanations.

ment and proof strategy generation. As shown in Figure 6, integrating GPT-4 for autoformalisation led to a significant increase in the number of explanations successfully refined across all models. For instance, Llama2-70b with GPT-4 as the formalisation component refined explanations from 7% to 65% in the e-SNLI dataset. For the multiple-choice question answering dataset, GPT-3.5 showed a relatively smaller increase from 44% to 48% and from 29% to 34%. Despite these improvements, a performance gap persists between GPT-4 and the other models, which is attributed to GPT-4's superior symbolic reasoning capabilities required for explanation refinement from the identified logical errors.

Explanations are progressively made more complete and consistent through iterative refinement. In order to deliver step-wise logical consistency, explanations need to be made complete and self-contained, leading to the introduction of additional explanatory sentences, leading to an increase in the total number of suggested proof steps. Therefore, we further evaluated how the proof steps vary when the total number of suggested proof steps increases contrasting both refined and unrefined cases. Figure 7 illustrates this trend. In general, all models show a positive trend, as the total suggested proof steps increase, the average number of proof steps processed by the proof assistant also increases. Models like Mistral-small and GPT-3.5 tend to suggest more proof steps to accomplish the logical goal, which can result in some redundant steps, such as the significant pulse shown in Figure 7c. For unrefined explanations, as shown in Figure 7d, 7e and 7f, the progression is steadier but retains a positive trend, where the models generally suggest more proof steps in response to the additional explanatory sentences introduced to correct a logical error identified from the erroneous step. We analysed the correlation between average successful explanatory sentences and total planned sentences in proofs, detailed in Appendix A.3. Ex-



(a) (b) (c)

Figure 8: Human evaluation of refined explanations in terms of factuality and triviality.

amples of refined and unrefined explanations are in Appendix A.5.

4.5 Factual Errors and Trivial Explanations

In addition to evaluating the logical validity of explanations, we also conducted a human evaluation of the refined explanations considering factual correctness and explanation triviality for the two best-performing models (GPT-3.5 and GPT-4). This evaluation focused on two questions: *"Are the refined explanatory sentences factually correct?"* and *"Is the explanation trivial, merely repeating or paraphrasing the content of the premise and hypothesis to achieve logical validity?"*. As illustrated in Figure 8, our findings indicate that all refined explanations in the e-SNLI and WorldTree datasets are consistent with commonsense knowledge. In the QASC dataset, 2.27% and 1.82% of the explanation refined by GPT-3.5 and GPT-4 contain sentences misaligned with true world knowledge. We found that the majority of these errors result from over-generalisation, such as the sentence *All tetrapods are defined to have four limbs*, which inaccurately includes snakes.

Finally, we found a relatively low number of explanations that repeat or paraphrase the content of premise and hypothesis. This phenomenon is absent in e-SNLI and becomes more evident when the explanatory sentences increase in complexity (i.e., WorldTree), leading models sometimes to generate explanations that do not include any additional information for the entailment to hold.

5 Related Work

5.1 LLMs Self-Refinement from External Feedback

Self-refinement of LLMs has demonstrated promising effectiveness in generating faithful and trustworthy responses (Pan et al., 2023b). The use of external feedback to guide LLMs has been extensively studied (Olausson et al., 2024a; Yu et al., 2023; Akyurek et al., 2023). Previous work such as Peng et al. (2023) have employed facts retrieved from external knowledge bases as sources of feedback, while Paul et al. (2024) developed a critic model to provide feedback for reasoning refinement. Additionally, Nathani et al. (2023) have explored the use of feedback models for automated feedback generation. Various works have also investigated tasks related to code generation (Chen et al., 2023; Olausson et al., 2024b) and the creation of either synthetic or expert-written logical natural language expressions (Olausson et al., 2023). Quan et al. (2024) use a differentiable logic reasoner for verifying and refining explanations via abductive reasoning, improving logical consistency in ethical NLI tasks. This paper focuses on the automated refinement of natural language sentences created by human annotators, which can identify the exact erroneous steps to effectively refine logical errors in the explanatory sentences.

6 Conclusion

In this work, we present a novel neuro-symbolic framework, Explanation-Refiner, which utilises LLMs and theorem provers for automatic verification and refinement of natural language explanations through iterative cycles. Extensive experiments on textual entailment and multiple-choice question tasks showed improved logical validity of human-annotated explanations. We investigated the model's performance from simple to complex explanatory/sentence structures and introduced a method to prevent the loss of semantic information in autoformalisation tasks with error correction. In future work, we aspire to enhance the framework's robustness towards complex and unstructured explanations with fewer iterations required to improve the model's efficiency.

7 Limitations

While this work have demonstrated significant improvements in terms of enhancing the logical consistency of explanations, the connection between improved logical consistency and AI safety still needs further investigation. While the concept of using formal solvers in conjunction with LLMs delivers a promise avenue to improve the consistency of reasoning within LLMs, these methodologies needs to be further developed and critically assessed as a mechanism which can provide guarantees of correctness, consistency and completeness within critical application domains.

References

Afra Feyza Akyurek, Ekin Akyurek, Ashwin Kalyan, Peter Clark, Derry Tanti Wijaya, and Niket Tandon. 2023. RL4F: Generating natural language feedback with reinforcement learning for repairing model outputs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7716–7733, Toronto, Canada. Association for Computational Linguistics.

Pepa Atanasova, Oana-Maria Camburu, Christina Lioma, Thomas Lukasiewicz, Jakob Grue Simonsen, and Isabelle Augenstein. 2023. Faithfulness tests for natural language explanations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 283–294, Toronto, Canada. Association for Computational Linguistics.

Kaj Bostrom, Zayne Sprague, Swarat Chaudhuri, and Greg Durrett. 2022. Natural language deduction through search over statement compositions. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4871–4883, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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.

Oana-Maria Camburu, Brendan Shillingford, Pasquale Minervini, Thomas Lukasiewicz, and Phil Blunsom. 2020. Make up your mind! adversarial generation of inconsistent natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4157–4165, Online. Association for Computational Linguistics.

Chun Sik Chan, Huanqi Kong, and Liang Guanqing. 2022. A comparative study of faithfulness metrics for model interpretability methods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5029–5038, Dublin, Ireland. Association for Computational Linguistics.

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.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *Preprint*, arXiv:2304.05128.

Dhairya Dalal, Marco Valentino, André Freitas, and Paul Buitelaar. 2024. Inference to the best explanation in large language models. *Preprint*, arXiv:2402.10767.

Yanai Elazar, Hongming Zhang, Yoav Goldberg, and Dan Roth. 2021. Back to square one: Artifact detection, training and commonsense disentanglement in the Winograd schema. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10486–10500, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. Critic: Large language models can self-correct with tool-interactive critiquing. *Preprint*, arXiv:2305.11738.

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 Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las

Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024a. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Dongwei Jiang, Marcio Fonseca, and Shay B. Cohen. 2024b. 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*.

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

Sawan Kumar and Partha Talukdar. 2020. NILE : Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8730–8742, Online. Association for Computational Linguistics.

Haochen Liu, Joseph Thekinen, Sinem Mollaoglu, Da Tang, Ji Yang, Youlong Cheng, Hui Liu, and Jiliang Tang. 2022. Toward annotator group bias in crowdsourcing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1797–1806, Dublin, Ireland. Association for Computational Linguistics.

Mistral AI. 2024. https://docs.mistral.ai/.

Deepak Nathani, David Wang, Liangming Pan, and William Wang. 2023. MAF: Multi-aspect feedback for improving reasoning in large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6591–6616, Singapore. Association for Computational Linguistics.

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.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2024a. Is self-repair a silver bullet for code generation? In *International Conference on Learning Representations (ICLR)*.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2024b. Is self-repair a silver bullet for code generation? *Preprint*, arXiv:2306.09896.

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

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023a. 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.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023b. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *Preprint*, arXiv:2308.03188.

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

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. REFINER: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1100–1126, St. Julian's, Malta. Association for Computational Linguistics.

Lawrence Charles Paulson. 1999. A generic tableau prover and its integration with isabelle. *J. Univers. Comput. Sci.*, 5:73–87.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *Preprint*, arXiv:2302.12813.

Xin Quan, Marco Valentino, Louise Dennis, and André Freitas. 2024. 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.

Boris Shminke. 2022. Python client for isabelle server. *Preprint*, arXiv:2212.11173.

Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2021. Explainable inference over grounding-abstract chains for science questions. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1–12, Online. 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. *Preprint*, arXiv:2307.09288.

Marco Valentino, Ian Pratt-Hartmann, and André Freitas. 2021. Do natural language explanations represent valid logical arguments? verifying entailment in explainable NLI gold standards. In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pages 76–86, Groningen, The Netherlands (online). Association for Computational Linguistics.

Marco Valentino, Mokanarangan Thayaparan, Deborah Ferreira, and André Freitas. 2022. Hybrid autoregressive inference for scalable multi-hop explanation regeneration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11403–11411.

Nathaniel Weir, Peter Clark, and Benjamin Van Durme. 2023. Nellie: A neuro-symbolic inference engine for grounded, compositional, and explainable reasoning. *Preprint*, arXiv:2209.07662.

Sarah Wiegreffe and Ana Marasovic. 2021. Teach me to explain: A review of datasets for explainable natural language processing. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Wenhao Yu, Zhihan Zhang, Zhenwen Liang, Meng Jiang, and Ashish Sabharwal. 2023. Improving language models via plug-and-play retrieval feedback. *Preprint*, arXiv:2305.14002.

Wenting Zhao, Justin Chiu, Claire Cardie, and Alexander Rush. 2023. Abductive commonsense reasoning exploiting mutually exclusive explanations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14883–14896, Toronto, Canada. Association for Computational Linguistics.

Xinyan Zhao and V.G.Vinod Vydiswaran. 2021. Lirex: Augmenting language inference with relevant explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14532–14539.

A Appendix

A.1 Algorithm

Algorithm 1 shows the overall framework of Explanation-Refiner.

A.2 Scalability

Figure 9 shows the average Isabelle/HOL solving time against the number of planned explanatory sentences in a proof and the length of suggested proof steps, including theories that have syntax errors, respectively. In some cases, the theorem prover may get stuck on a proof step, and we have set a termination time if the solving time exceeds 65 seconds.

A.3 Average Processed vs. Planned Explanatory Sentences per Proof

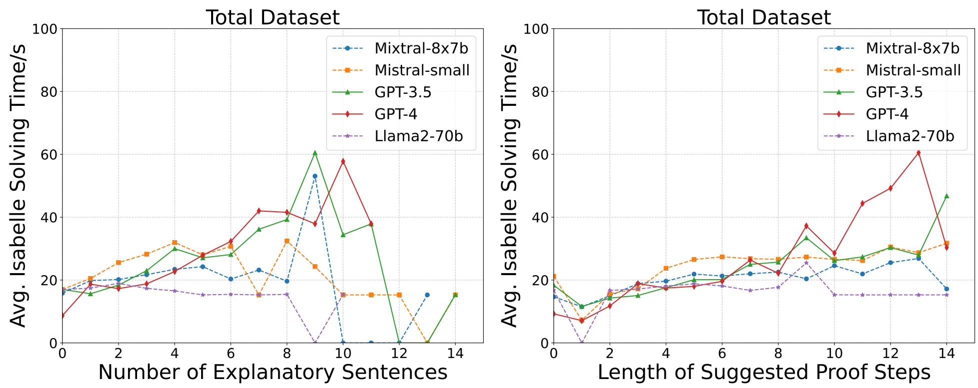
Figure 10 and Figure 11 shows experiments on average number of successfully processed explanatory sentences in one proof against total planned explanatory sentences in a suggest proof. Figure 12 also shows the comparison of average processed proof steps against total suggested proof steps in all dataset.

A.4 Prompts

Temperature settings were adjusted to 0 for GPT-3.5 and GPT-4, and to 0.01 for Llama2-70b, Mixtral-8x7b, and Mistral-small, aiming to achieve both determinism in the output and effective code generation for theorem prover.

A.4.1 Autoformalisation

Figure 13 displays the prompts used to identify action verbs (events) within the premise, explanation, and hypothesis sentences, representing events in Davidsonian-event semantics. Figure 14 displays the prompts used to transfer natural language to logical forms based on the identified events verbs. Figure 15 shows how to convert logical forms into Isabelle/HOL code (axioms and type declaration). Figure 16 shows how to convert the premise and hypothesis sentences into the Isabelle/HOL theorem code, based on the previously constructed axioms code. Figure 17 shows how to refine the syntax errors based on the types of errors, the provided code, the error messages, and the locations of the errors within the code.



(a) (b)

Figure 9: (a) Average Isabelle/HOL solving time against number of explanatory sentences planned in a proof. (b) Average Isabelle/HOL solving time against number of suggested proof steps in a proof.

A.4.2 Proof Construction

Figure 18 shows the prompts for making a preliminary inference strategy, which also identifies redundant and related explanatory sentences that will be used for proof generation. Figure 19 shows the prompts for building the proof steps used for Isabelle/HOL Proof assistant based on the provided inference strategy.

A.4.3 Explanation Refinement

Figure 20 shows how to refine the explanatory sentences based on the provided information.

A.5 Examples of Explanation Refinement

A.5.1 e-SNLI Refined Examples

Table 1 shows an example from the e-SNLI dataset of how the explanation changes after each iteration. Figures 21, 22, and 23 illustrate the Isabelle/HOL theory code changes during the refinement process. Green code indicates the proof steps that have successfully progressed, while red code shows where the proof failed at that step.

Table 2 along with Figures 24 and 25, and Table 3 with Figures 26, 27, and 28 are two more examples.

A.5.2 QASC Refined Examples

Table 4 shows an example from the QASC dataset of how the explanation changes after each iteration. Figures 29, 30 illustrate the Isabelle/HOL theory code changes during the refinement process. Green code indicates the proof steps that have successfully progressed, while red code shows where the proof failed at that step.

Table 5 along with Figures 31 and 32, and Table 6 with Figures 33, 34 are two more examples.

A.5.3 WorldTree Refined Examples

Table 7 shows an example from the WorldTree dataset of how the explanation changes after each iteration. Figures 35, 36, 37, 38, 39, 40, 41 and 42 illustrate the Isabelle/HOL theory code changes during the refinement process. Green code indicates the proof steps that have successfully progressed, while red code shows where the proof failed at that step.

Table 8 and Figures 43, 45, 46 and 47, as well as Table 9 with Figures 48, 49, 50, and 51, provide two more examples.

A.5.4 Unrefined Example

Table 10 and Table 11 shows an example from the WorldTree dataset that does not refine within 10 iterations and is not caused by a syntax error. The figures that follow show the detailed Isabelle/HOL theory code of the related iterations.

A.6 Datasets and Theorem Prover

The datasets used in our experiments, including samples from e-SNLI (Camburu et al., 2018), QASC (Khot et al., 2019), and WorldTree (Jansen et al., 2018), are all sourced from open academic works. We employed Isabelle as the theorem prover, which is distributed under the revised BSD license. Additionally, the TCP client used for the Isabelle server (Shminke, 2022) is licensed under Apache-2.0.

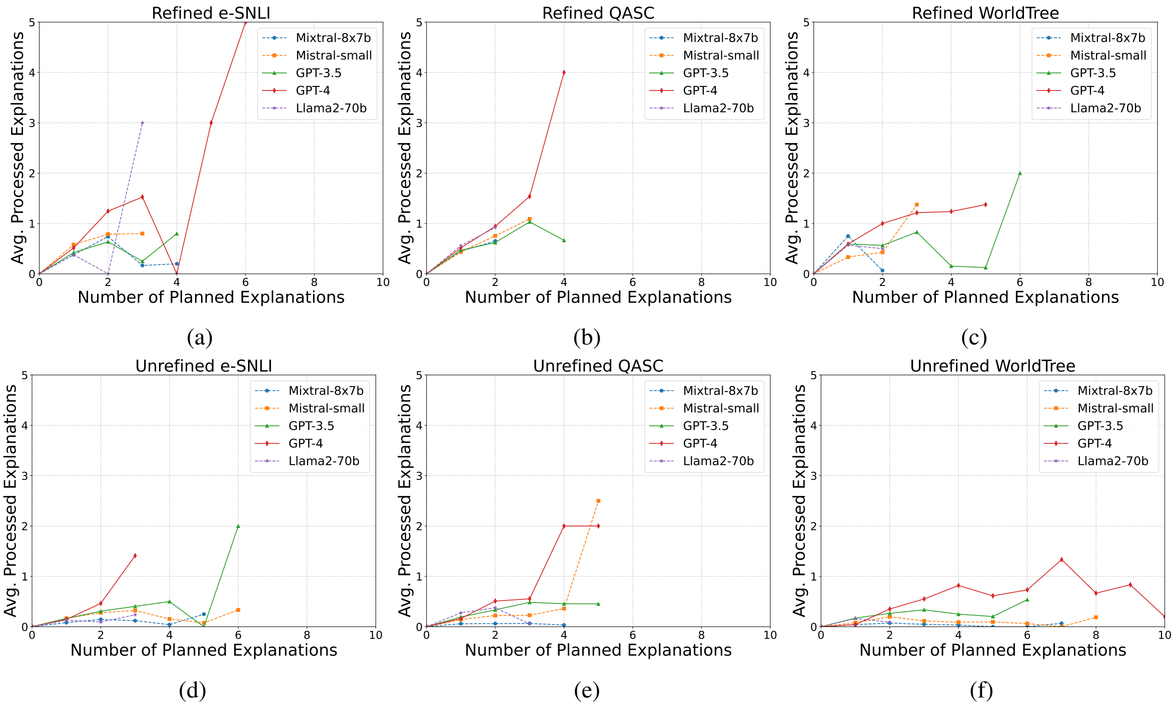


Figure 10: Average Progressed Explanations against Number of Planned Explanations in Refined and Unrefined e-SNLI, QASC and WorldTree Dataset

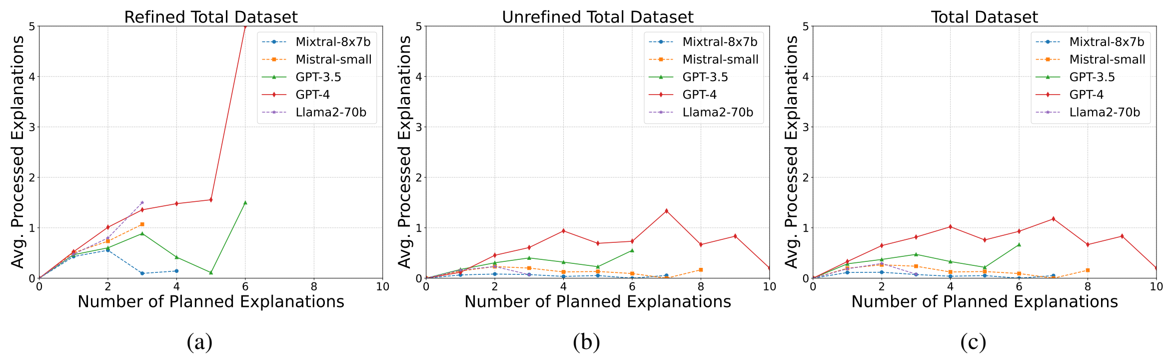


Figure 11: Average Progressed Explanations against Number of Planned Explanations for Refined, Unrefined, and Combined Across All Datasets

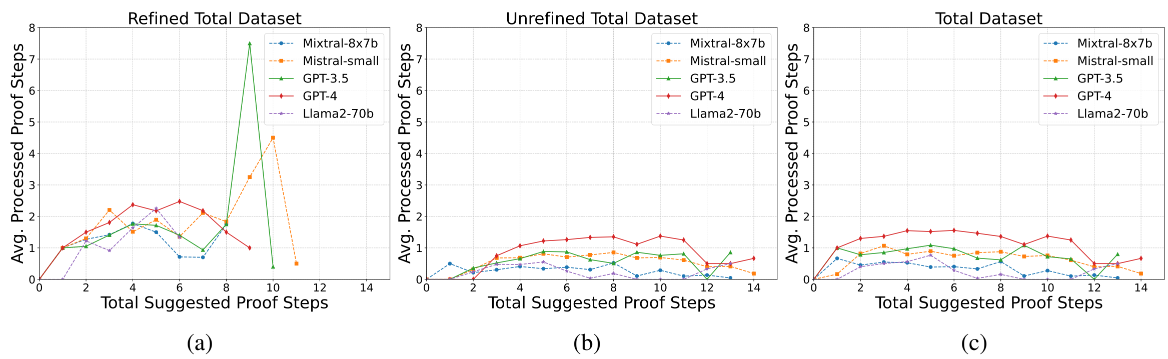


Figure 12: Average Processed Proof Steps against Total Suggested Proof Steps for Refined, Unrefined, and Combined Across All Datasets

Algorithm 1: Explanation-Refiner

Input : Premise p, Explanation E, Hypothesis h, Isabelle//HOL server $isabelle$, Autoformalisation model m_a, Isabelle syntax refinement model m_{sr}, Rough inference model m_{ri}, Proof step build model m_{pr}, Facts filter model m_f, Explanation refinement model m_e

Output : Updated Explanation E

```
valid ← false
isabelle_theory ← [ ]
iterations ← 0
max_iterations ← 11
has_syntax_error ← false
while not valid and iterations < max_iterations do
    session_id ← session_build(HOL, isabelle)
    isabelle.start(session_id)
    isabelle_theory ← transfer_to_symbolic(p, E, h, m_a)
    messages, error_content, error_code ← isabelle.check(isabelle_theory)
    if syntax_errors in messages then
        has_syntax_error ← true
        it ← 0
        while has_syntax_error and it < 3 do
            isabelle_theory = refine_syntax(messages, error_content, error_code, isabelle_theory,
              m_sr)
            messages, error_content, error_code ← isabelle.check(isabelle_theory)
            if syntax_errors in messages then
                has_syntax_error ← true
                it ← it + 1
            else
                break
            end if
        end while
    end if
    rough_inference ← make_rough_inference(p, E, h, m_ri)
    proof_steps ← build_proof(rough_inference, m_pr)
    isabelle_theory ← isabelle_theory + proof_steps
    messages, error_content, error_code ← isabelle.check(isabelle_theory)
    if messages is not empty then
        message ← messages[0]
        E ← filter(E, rough_inference, proof_steps, m_f)
        E ← refine_explanation(message, error_content, error_code, rough_inference, proof_steps,
          p, E, H, m_e)
    else
        valid ← true
        break
    end if
    iterations ← iterations + 1
    isabelle.shutdown()
end while
return E
```

```
SYSTEM: You are an expert in linguistics. You will be provided
with some sentences, find any action verbs of these sentences.
You need to ignore auxiliary verbs and modal verbs.
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.
Has action: Yes
Actions: 1. playing

Explanation Sentence:
1. A violin is an instrument.
Has action: No
Actions: none

Premise Sentence:
1. A smiling woman is playing the violin in front of a
turquoise background.
Has action: Yes
Actions: 1. playing
###
...
###
<<<<<<<<<<<<<<<<<<<<<<<<
Strictly follow the instructions that I have claimed.

Provided sentences:
{{input_sentence}}

Answer:
```

Figure 13: Prompts for detecting event-related words in the given sentences

```
SYSTEM: You are an expert in semantics, formal language and
neo-davidsonian event semantics. You will be provided with
some sentences and the action verbs involved in those
sentences. You need to transfer the sentences into symbolic
language. If the sentence has no action, transfer it into
formal language using first-order language. If the sentence
has one action, transfer it using first-order language and
davidsonian event semantics within one event. If the sentence
has two more actions, transfer it using first-order language
and davidsonian event semantics within at most two events.
Some instructions:
1. Capture All Information: Ensure the logical form reflects
   every detail from the sentence.
2. Use '→' for Certain Verbs: Represent actions like 'cause',
'lead', 'help' that represent an implication, causal relation
with '→' for clarity.
3. Event Variable 'e': Use 'e' for events, actions, with
action predicates having 'e' as their sole argument.
...
USER: Here are some examples:
###
Sentence: Grass is a kind of plant.
Has action: No
Actions:
Logical form: ∀x. Grass(x) → Plant(x)
###
Sentence: Squirrels typically eat nuts for energy.
Has action: Yes
Actions: 1. eat
Logical form: ∀x y z. Squirrels(x) ∧ Nuts(y) → (∃e. Eat(e) ∧
Agent(e, x) ∧ Patient(e, y) ∧ ForEnergy(y, x))
###
...
<<<<<<<<<<<<<<<<<<<<<<<<
Strictly followed the instructions that I have claimed.

Provided sentences:
{{input_sentence}}

Answer:
```

Figure 14: Prompts for converting natural language sentences into logical form representations

```
SYSTEM: You are an expert in Isabelle theorem prover, first-
order logic and Davidsonian event semantics. You will be
provided with some sentences and corresponding logical forms
(first-order logic and davidsonian event semantics) of those
sentences. You need to transfer such logical forms into
Isabelle axioms code and define the consts and of the symbolic
forms.
Some instructions:
1. Isabelle axioms code use ∧, ∨, ∀, ∃, ¬, ↔, ⟶ as logic
symbols. Please write the axiom code with these logic symbols.
2. Isabelle consts code use ⇒ as logic symbols. Please define
...
The code structure for axioms is:
```
begin

typedecl entity
typedecl event
consts
  [define the consts here]

(* Explanation 1: [provided sentence 1 in natural language] *)
axiomatization where
  explanation_1: [Transfer the logical form into isabelle code
here, non-bracketed of the predicate-argument form]
:::
USER: Here are some examples:
###
Provided sentences:

Explanation Sentence:
1. If the infant is crying, it can be assumed that they are
unhappy.
Logical form: ∀x e. Infant(x) ∧ Crying(e) ∧ Agent(e, x) ⟶
Unhappy(x)

Answer:
```
begin
typedecl entity
typedecl event

consts
  Unhappy :: "entity ⇒ bool"
  Infant :: "entity ⇒ bool"
  Crying :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"

(* Explanation 1: If the infant is crying, it can be assumed
that they are unhappy. *)
axiomatization where
  explanation_1: "∀x e. Infant x ∧ Crying e ∧ Agent e x ⟶
Unhappy x"

```
###
...
###
<<<<<<<<<<<<<<<<<<<<<<<
Strictly follow the instructions that I have claimed.

Provided sentences:
{{explanatory_sentences}}

Answer:

```
answer goes here
```
```

Figure 15: Prompts for converting logical form into Isabelle/HOL code format for building the axioms and type declaration

```
SYSTEM: You are an expert in Isabelle theorem prover, first-
order logic and Davidsonian event semantics. You will be
provided with a Hypothesis sentence and a Premise sentence
with their corresponding logical forms (first-order logic and
davidsonian event semantics).
...
Some instructions:
1. Isabelle code use ∧, ∨, ∀, ∃, ¬, ↔, ⟶ as logic symbols.
Please write the code with these logic symbols.
...
The code structure for theorem hypothesis is:
```
theorem hypothesis:
  (* Premise: [provided premise sentence in natural language]
*)
...
end
```
USER: Here are some examples:
###
Provided sentences:
...
Provided code:
...
Answer:
```
imports Main

begin

typedecl entity
typedecl event

consts
  AdultSponges :: "entity ⇒ bool"
  Eggs :: "entity ⇒ bool"
  Sperm :: "entity ⇒ bool"
  Gametes :: "entity ⇒ bool"
  Produce :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"

(* Explanation 1: Adult sponges produce eggs and sperm. *)
axiomatization where
  explanation_1: "∀x. AdultSponges x ⟶ (∃e y z. Eggs y ∧
Sperm z ∧ Produce e ∧ Agent e x ∧ Patient e y ∧ Patient e z)"

(* Explanation 2: Sperm and eggs are cells known as gametes.
*)
axiomatization where
  explanation_2: "∀x y. Sperm x ∧ Eggs y ⟶ Gametes x ∧
Gametes y"

theorem hypothesis:
  (* Premise: Students are studying adult sponges. *)
  assumes asm: "Students x ∧ AdultSponges y ∧ Studying e ∧
Agent e x ∧ Patient e y"
  (* Hypothesis: Adult sponges produce gametes. *)
  shows "∃x y e. AdultSponges x ∧ Gametes y ∧ Produce e ∧
Agent e x ∧ Patient e y"
proof -


qed

end

```
###
...
###
<<<<<<<<<<<<<<<<<<<<<<<
Strictly follow the instructions that I have claimed.

Provided sentences:
{{input_sentence}}

Provided code:
{{axiom_code}}

Answer:

```
answer code goes here (complete isabelle code )
```
```

Figure 16: Prompts for building the theorem code part of the Isabelle/HOL theory
```

````
SYSTEM: You are an expert in the Isabelle theorem prover and
familiar with HOL session syntax and Davidsonian event
semantics. You will be provided with Isabelle code containing
some syntax errors, along with details of the errors and their
locations in the code. You need to fix the code (logical form)
of the related error.
Some instructions:
1. Do not change code structure, you just need to fix the
syntax error.
2. Type unification failed errors indicates the defined consts
and the acutal preidcates are not consistent. There are only
two types: event and entity. The type defined in the consts
should be same as the type represented in the logical form
codes.
...
USER: Here are some examples:
###
Provided code:
...
Error Identified:

1. Error on line 15: Type unification failed: Clash of types
"entity" and "event"

Type error in application: incompatible operand type

...
Code Cause This Error:

explanation_1: "∀x c. Cute x ∧ Couple x ∧ Club c ∧ At x c ⟶
(∃y. Couple y ∧ Club c ∧ At y c)"
...

Answer:
From the error found, the error type is type unification
failed, which means the operand type defined in consts is not
consistent with the operand in the code.
As the error indicates the Operator 'At' in code is defined as
At :: "event ⇒ entity ⇒ bool" but in the code it is stated as
At x c where x is defined as entity. That's the reason cause
type unification failed.
It should have the same type which can be refined as:
```
...
Couple :: "entity ⇒ bool"
  Club :: "entity ⇒ bool"
  At :: "entity ⇒ entity ⇒ bool"
  Cute :: "entity ⇒ bool"

(* Explanation 1: A cute couple at a club means a couple is at
a club. *)
axiomatization where
  explanation_1: "∀x c. Cute x ∧ Couple x ∧ Club c ∧ At x c ⟶
(∃y. Couple y ∧ Club c ∧ At y c)"

theorem hypothesis:
  (* Premise: A cute couple at a club *)
  assumes asm: "Cute x ∧ Couple x ∧ Club c ∧ At x c"
  (* Hypothesis: The couple is at a club. *)
  shows "∃x. Couple x ∧ Club c ∧ At x c"
proof -

qed

end
```
The At :: "event ⇒ entity ⇒ bool" has been refined as At ::
"eneity ⇒ entity ⇒ bool", then the types are consistent for
both consts and following logical code.
###

<<<<<<<<<<<<<<<<<<<<<<<<
Strictly follow the instructions that I have claimed.
Provided code:
{{code}}

Error Identified:
{{error_detail}}

Code Cause This Error:
{{code_cause_error}}

Answer:

```
answer code goes here (complete refined isabelle code)
```
````

Figure 17: Prompts for how to refine the identified syntax errors in the constructed code

```
SYSTEM: You are an expert in natural language inference,
textual entailment and linguistic semantics. You will be
provided with a premise sentence, some explanatory sentences
and a hypothesis sentence. The premise sentence and
explanatory sentences should entail the hypothesis sentence.
You need to write a step-by-step natural language inference to
state how the explanatory sentences will entail the hypothesis
sentence from the premise sentences.
Instructions:
1. You must elicit the explanatory sentences which are
redundant and not directly related (if there are no redundant
or all related state it as no).
2. You must state on which step of the proof each explanatory
sentence is used.
3. You must elicit the used explanatory sentences in the
natural language inference steps.
USER: Here are some examples:
###
Provided Premise Sentence:
A group of students are studying non-contact force.

Provided Explanation Sentences:
1. Non-contact forces can affect objects that are not
touching.
2. A magnet attracts magnetic and ferromagnetic metals through
magnetism.
3. Magnetism does not require contact between objects to act.
4. A paper clip is a kind of object.
5. A magnet is a kind of object.
6. Magnetism is a kind of force.
7. A kind of something is an example of that something.

Provided Hypothesis Sentence:
A paper clip attracted to a magnet is an example of a non-
contact force acting on an object.

Natural Language Inference Steps:
1. As we need to infer the hypothesis, we need to find the
information of paper clip, magnet, non-contact force and
object. The action event of attracted and acting. The
relationship of is an example of.
2. From the premise, we can get the information of non-contact
force.
3. From explanation 4 and 5, we deduce that both a paper clip
and a magnet are objects.
4. Explanation 2 establishes that a magnet can attract certain
metals through magnetism, which is a force (due to explanation
6).
...

Explanation 1 is redundant. There is no not directly related
explanation sentence.
The proof steps use explanation 2, explanation 3, explanation
4, explanation 5, explanation 6, explanation 7.

###
...
###

<<<<<<<<<<<<<<<<<<<<<<<<
Strictly follow the instructions that I have claimed.

Provided Premise Sentence:
{{premise}}

Provided Explanation Sentences:
{{explanation}}

Provided Hypothesis Sentence:
{{hypothesis}}

Natural Language Inference Steps:
```

Figure 18: Prompts for how to make a step-by-step preliminary inference strategy

```
SYSTEM: You are an expert in Isabelle theorem prover, first-
order logic and Davidsonian event semantics. You will be
provided with an Isabelle code which consistent of some
axioms, a theorem hypothesis that needs to be proven. The
logical form of axioms indicates some explanatory sentences,
the logical form after "assume asm:" indicates a premise
sentence and the logical form after "shows" indicates a
hypothesis sentence.
...
Some instructions:
1. 'sorry' and 'fix' command is not allowed.
...
USER: Here are some examples:
###
Provided Isabelle Code:
```
...
begin
typedecl entity
typedecl event
consts
  PlantReproduction :: "entity ⇒ bool"
...
(* Explanation 1: Plant reproduction often requires pollen. *)
axiomatization where
  explanation_1: "∀x y e. PlantReproduction x ∧ Pollen y ∧
Require e ∧ Agent e x ∧ Patient e y"

theorem hypothesis:
  (* Premise: Students are studying plant reproduction
process. *)
  assumes asm: "Students x ∧ PlantReproduction y ∧ Studying e
∧ Agent e x ∧ Patient e y"
  (* Hypothesis: Plant reproduction often requires bees. *)
  shows "∃x y e. PlantReproduction x ∧ Bee y ∧ Require e ∧
Agent e x ∧ Patient e y"
proof -

qed

end
```
Provided Natural Language Inference Strategy:

1. As we need to infer the hypothesis, we need to find the
information of plant, reproduction process, requires action
and bees.
2. From explanation 1, we get the information of plant
reproduction, which requires pollen.
...
Explanation 3 and 4 is not related and Explanation 5 is
redundant.
The proof steps use explanation 1 and explanation 2.

Answer:
```
proof -
  from asm have "PlantReproduction x" by simp
  then obtain e1 where e1: "Require e1 ∧ Agent e1 x ∧ Patient
e1 y" using explanation_1 by blast
  then have "Bee y" using explanation_2 by blast
  have conclusion: "Require e1 ∧ Agent e1 x ∧ Patient e1 y"
using e1 by simp
  show ?thesis using asm conclusion `Bee y` by blast
qed
```
###
...
<<<<<<<<<<<<<<<<<<<<<<<<
Strictly follow the instructions that I have claimed.

Provided Isabelle Code:
{{isabelle_code}}

Provided Natural Language Inference Strategy:
{{rough_inference}}

Answer:
```

Figure 19: Prompts for how to build a proof for Isabelle/HOL proof assistant

```
SYSTEM: You are an expert in Isabelle theorem prover, first-
order, Davidsonian event semantics and natural language
inference. You will be provided with three types of sentences:
Premise Sentence, Explanation Sentence and Hypothesis
sentence.
...
Some instructions:
1. Only refine the related axioms/explanatory sentence in
natural language sentences.
...
USER: Here are some examples:
###
Provided Premise Sentence:
...
Natural Language Inference steps:
1. To infer the hypothesis, we need to identify the
information related to a tennis ball, water, and the action of
floating. The relationship of "will" indicates a future or
potential action.
...

Isabelle code:
...

(* Explanation 5: water is a kind of liquid. *)
axiomatization where
  explanation_5: "∀x. Water x ⟶ Liquid x"


...
proof -
  from asm have "TableTennisBall x " by simp
  then have "Object x" using explanation_1 by blast
  then obtain e1 where e1: "Contains e1 ∧ Agent e1 x ∧ Patient
e1 y" using explanation_2 by blast
...
qed
...

Proof failed at:
then have "Object x" using explanation_1 by blast

Refine strategy:
From the provided error location, it failed at the step of
"then have "Object x" using explanation_1 by blast" using
explanation 1.
...
Updated explanatory sentences:
1. a table tennis ball is a kind of object.
2. a tennis ball contains air.
3. something that contains air is usually buoyant.
4. buoyant means able to float in a liquid or gas.
5. water is a kind of liquid.
###
...
<<<<<<<<<<<<<<<<<<<<<<<<
Strictly follow the instructions that I have claimed.

Provided Premise Sentence:
{{premise}}

Provided Explanation Sentences:
{{explanation}}

Provided Hypothesis Sentence:
{{hypothesis}}

Natural Language Inferece steps:
{{rough_inference}}

Isabelle code:
{{isabelle_code}}

Proof failed at:
{{error_code}}

Refine strategy:


Updated explanatory sentences:
```

Figure 20: Prompts for how to refine the explanatory sentences

Dataset	Sentences	Explanation	Iteration	Validity
e-SNLI	**Premise**: A woman in black framed glasses peruses a photo album while sitting in a red wicker chair. **Hypothesis**: There is a lady with a book.	The lady is looking through a photo album which is a type of book.	0	Invalid
e-SNLI	**Premise**: A woman in black framed glasses peruses a photo album while sitting in a red wicker chair. **Hypothesis**: There is a lady with a book.	A woman can be referred to as a lady. A photo album is a type of book.	1	Invalid
e-SNLI	**Premise**: A woman in black framed glasses peruses a photo album while sitting in a red wicker chair. **Hypothesis**: There is a lady with a book.	A woman can be referred to as a lady. A photo album is a type of book. If a woman is perusing a photo album, then the woman is with a book.	2	Valid

Table 1: An example of how the explanation sentences in e-SNLI can be refined with Explanation-Refiner

```
theory question_28_0

imports Main

begin
typedecl entity
typedecl event

consts
  Lady :: "entity ⇒ bool"
  PhotoAlbum :: "entity ⇒ bool"
  Book :: "entity ⇒ bool"
  Looking :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Woman :: "entity ⇒ bool"
  Glasses :: "entity ⇒ bool"
  BlackFramed :: "entity ⇒ bool"
  Chair :: "entity ⇒ bool"
  Red :: "entity ⇒ bool"
  Wicker :: "entity ⇒ bool"
  Peruses :: "event ⇒ bool"
  Sitting :: "entity ⇒ entity ⇒ bool"
  With :: "entity ⇒ entity ⇒ bool"

(* Explanation 1: The lady is looking through a photo album which is a type of book. *)
axiomatization where
    explanation_1: "∃x y e. Lady x ∧ PhotoAlbum y ∧ Book y ∧ Looking e ∧ Agent e x ∧ Patient e y"

theorem hypothesis:
(* Premise: A woman in black framed glasses peruses a photo album while sitting in a red wicker chair. *)
assumes asm: "Woman x ∧ Glasses x ∧ BlackFramed x ∧ PhotoAlbum y ∧ Chair z ∧ Red z ∧ Wicker z ∧ Peruses e ∧ Agent e
x ∧ Patient e y ∧ Sitting x z"
(* Hypothesis: There is a lady with a book. *)
shows "∃x y. Lady x ∧ Book y ∧ With x y"
proof -
  from asm have "Woman x ∧ PhotoAlbum y" by blast
  from this have "Lady x ∧ Book y" using explanation_1 by blast
  then show ?thesis using `Lady x` `Book y` by blast
qed

end
```

Figure 21: The Isabelle theory code for table 1 iteration 0

```
theory question_28_1

imports Main

begin
typedecl entity
typedecl event

consts
  Woman :: "entity ⇒ bool"
  Lady :: "entity ⇒ bool"
  PhotoAlbum :: "entity ⇒ bool"
  Book :: "entity ⇒ bool"
  Glasses :: "entity ⇒ bool"
  BlackFramed :: "entity ⇒ bool"
  Peruses :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Chair :: "entity ⇒ bool"
  Red :: "entity ⇒ bool"
  Wicker :: "entity ⇒ bool"
  SittingIn :: "entity ⇒ entity ⇒ bool"
  With :: "entity ⇒ entity ⇒ bool"

(* Explanation 1: A woman can be referred to as a lady. *)
axiomatization where
    explanation_1: "∀x. Woman x ⟶ Lady x"

(* Explanation 2: A photo album is a type of book. *)
axiomatization where
    explanation_2: "∀x. PhotoAlbum x ⟶ Book x"

theorem hypothesis:
(* Premise: A woman in black framed glasses peruses a photo album while sitting in a red wicker chair. *)
assumes asm: "Woman x ∧ Glasses x ∧ BlackFramed x ∧ PhotoAlbum y ∧ Peruses e ∧ Agent e x ∧ Patient e y ∧ Chair z ∧
Red z ∧ Wicker z ∧ SittingIn x z"
(* Hypothesis: There is a lady with a book. *)
shows "∃x y. Lady x ∧ Book y ∧ With x y”
proof -
  from asm have "Woman x" by blast
  then have "Lady x" using explanation_1 by blast
  from asm have "PhotoAlbum y" by blast
  then have "Book y" using explanation_2 by blast
  show ?thesis using `Lady x` `Book y` by blast
qed

end
```

Figure 22: The Isabelle theory code for table 1 iteration 1

```
theory question_28_2

imports Main

begin
typedecl entity
typedecl event

consts
  Woman :: "entity ⇒ bool"
  Lady :: "entity ⇒ bool"
  PhotoAlbum :: "entity ⇒ bool"
  Book :: "entity ⇒ bool"
  Perusing :: "event ⇒ bool"
...

(* Explanation 1: A woman can be referred to as a lady. *)
axiomatization where
    explanation_1: "∀x. Woman x ⟶ Lady x"

(* Explanation 2: A photo album is a type of book. *)
axiomatization where
    explanation_2: "∀x. PhotoAlbum x ⟶ Book x"

(* Explanation 3: If a woman is perusing a photo album, then the woman is with a book. *)
axiomatization where
    explanation_3: "∀x y e. Woman x ∧ PhotoAlbum y ∧ Perusing e ∧ Agent e x ∧ Patient e y ⟶ With x y"

theorem hypothesis:
(* Premise: A woman in black framed glasses peruses a photo album while sitting in a red wicker chair. *)
assumes asm: "Woman x ∧ Glasses x ∧ BlackFramed x ∧ PhotoAlbum y ∧ Chair z ∧ RedWicker z ∧ Perusing e ∧ Agent e x ∧
Patient e y ∧ SittingIn x z"
(* Hypothesis: There is a lady with a book. *)
shows "∃x y. Lady x ∧ Book y ∧ With x y"
proof -
  from asm have "Woman x ∧ PhotoAlbum y ∧ Perusing e ∧ Agent e x ∧ Patient e y" by blast
  then have "With x y" using explanation_3 by blast
  from asm have "Woman x" by blast
  then have "Lady x" using explanation_1 by blast
  from asm have "PhotoAlbum y" by blast
  then have "Book y" using explanation_2 by blast
  show ?thesis using `Lady x` `Book y` `With x y` by blast
qed

end
```

Figure 23: The Isabelle theory code for table 1 iteration 2

Dataset	Sentences	Explanation	Iteration	Validity
e-SNLI	**Premise**: A couple being romantic under the sunset. **Hypothesis**: People are enjoying sundown.	A couple consists of people. Sunset is sundown.	0	Invalid
e-SNLI	**Premise**: A couple being romantic under the sunset. **Hypothesis**: People are enjoying sundown.	A couple consists of people. Sunset is sundown. Being romantic under the sunset implies enjoying the sundown.	1	Valid

Table 2: An example of how the explanation sentences in e-SNLI can be refined with Explanation-Refiner

```
theory question_38_0

imports Main

begin
typedecl entity
typedecl event

consts
  Couple :: "entity ⇒ bool"
  People :: "entity ⇒ bool"
  Sunset :: "entity ⇒ bool"
  Sundown :: "entity ⇒ bool"
  Romantic :: "event ⇒ bool"
  Enjoying :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Under :: "entity ⇒ entity ⇒ bool"

(* Explanation 1: A couple consists of people. Sunset is sundown. *)
axiomatization where
    explanation_1: "∀x y. Couple x ⟶ People x ∧ Sunset y ⟶ Sundown y"

theorem hypothesis:
(* Premise: A couple being romantic under the sunset. *)
assumes asm: "Couple x ∧ Sunset y ∧ Romantic e ∧ Agent e x ∧ Under x y"
(* Hypothesis: People are enjoying sundown. *)
shows "∃x y e. People x ∧ Sundown y ∧ Enjoying e ∧ Agent e x ∧ Patient e y"
proof -
  from asm have "Couple x ∧ Sunset y" by blast
  then have "People x ∧ Sundown y" using explanation_1 by blast
  then show ?thesis using asm by blast
qed

end
```

Figure 24: The Isabelle theory code for table 2 iteration 0

```
theory question_38_1

imports Main

begin

typedecl entity
typedecl event

consts
  Couple :: "entity ⇒ bool"
  People :: "entity ⇒ bool"
  Sunset :: "entity ⇒ bool"
  Sundown :: "entity ⇒ bool"
  Romantic :: "entity ⇒ bool"
  Under :: "entity ⇒ entity ⇒ bool"
  Enjoying :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Being :: "event ⇒ bool"

(* Explanation 1: A couple consists of people. *)
axiomatization where
    explanation_1: "∀x. Couple x ⟶ People x"

(* Explanation 2: Sunset is sundown. *)
axiomatization where
    explanation_2: "∀x. Sunset x ⟶ Sundown x"

(* Explanation 3: Being romantic under the sunset implies enjoying the sundown. *)
axiomatization where
    explanation_3: "∀x y z e. People x ∧ Romantic x ∧ Sunset y ∧ Under x y ⟶ Enjoying e ∧ Agent e x ∧ Sundown z ∧
Patient e z"

theorem hypothesis:
(* Premise: A couple being romantic under the sunset. *)
assumes asm: "Couple x ∧ Sunset y ∧ Being e ∧ Agent e x ∧ Under x y ∧ Romantic x"
(* Hypothesis: People are enjoying sundown. *)
shows "∃x y e. People x ∧ Sundown y ∧ Enjoying e ∧ Agent e x ∧ Patient e y"
proof -
  from asm have "Couple x ∧ Sunset y ∧ Being e ∧ Agent e x ∧ Under x y ∧ Romantic x" by blast
  then have "People x ∧ Sunset y ∧ Being e ∧ Agent e x ∧ Under x y ∧ Romantic x" using explanation_1 by blast
  then have "People x ∧ Sundown y ∧ Enjoying e ∧ Agent e x ∧ Patient e y" using explanation_3 by blast
  then show ?thesis using asm by blast
qed

end
```

Figure 25: The Isabelle theory code for table 2 iteration 1

Dataset	Sentences	Explanation	Iteration	Validity
e-SNLI	**Premise**: A male bartender dressed in all black with his sleeves rolled up to elbow height making a drink in a martini glass. **Hypothesis**: A person in black	A bartender, who is a person, is wearing black.	0	Invalid
e-SNLI	**Premise**: A male bartender dressed in all black with his sleeves rolled up to elbow height making a drink in a martini glass. **Hypothesis**: A person in black	A bartender is a person. If a person is wearing black, then the person is in black.	1	Invalid
e-SNLI	**Premise**: A male bartender dressed in all black with his sleeves rolled up to elbow height making a drink in a martini glass. **Hypothesis**: A person in black	A bartender is a person. If a person is dressed in black, then the person is in black.	2	Valid

Table 3: An example of how the explanation sentences in e-SNLI can be refined with Explanation-Refiner

```
theory question_66_0

imports Main

begin
typedecl entity
typedecl event

consts
  Bartender :: "entity ⇒ bool"
  Person :: "entity ⇒ bool"
  Black :: "entity ⇒ bool"
  Wearing :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Male :: "entity ⇒ bool"
  Drink :: "entity ⇒ bool"
  MartiniGlass :: "entity ⇒ bool"
  Making :: "event ⇒ bool"
  In :: "entity ⇒ entity ⇒ bool"

(* Explanation 1: A bartender, who is a person, is wearing black. *)
axiomatization where
    explanation_1: "∃x e. Bartender x ∧ Person x ∧ Black x ∧ Wearing e ∧ Agent e x ∧ Patient e x"

theorem hypothesis:
(* Premise: A male bartender dressed in all black with his sleeves rolled up to elbow height making a drink in a
martini glass. *)
assumes asm: "Bartender x ∧ Male x ∧ Black x ∧ Drink y ∧ MartiniGlass z ∧ Making e ∧ Agent e x ∧ Patient e y ∧ In z
y"
(* Hypothesis: A person in black *)
shows "∃x. Person x ∧ Black x"
proof -
  from asm have "Bartender x ∧ Black x" by blast
  from this have "Person x ∧ Black x" using explanation_1 by blast
  then show ?thesis by blast
qed

end
```

Figure 26: The Isabelle theory code for table 3 iteration 0

```
theory question_66_1

imports Main

begin
typedecl entity
typedecl event

consts
  Bartender :: "entity ⇒ bool"
  Person :: "entity ⇒ bool"
  Wearing :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Black :: "event ⇒ bool"
  InBlack :: "entity ⇒ bool"
  Male :: "entity ⇒ bool"
  Dress :: "event ⇒ bool"
  Sleeves :: "entity ⇒ bool"
  RolledUp :: "event ⇒ bool"
  ElbowHeight :: "event ⇒ bool"
  Drink :: "entity ⇒ bool"
  MartiniGlass :: "entity ⇒ bool"
  Making :: "event ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  In :: "entity ⇒ entity ⇒ bool"

(* Explanation 1: A bartender is a person. *)
axiomatization where
    explanation_1: "∀x. Bartender x ⟶ Person x"

(* Explanation 2: If a person is wearing black, then the person is in black. *)
axiomatization where
    explanation_2: "∀x e. Person x ∧ Wearing e ∧ Agent e x ∧ Black e ⟶ InBlack x"

theorem hypothesis:
(* Premise: A male bartender dressed in all black with his sleeves rolled up to elbow height making a drink in a
martini glass. *)
assumes asm: "Bartender x ∧ Male x ∧ Black e1 ∧ Dress e1 ∧ Agent e1 x ∧ Sleeves x ∧ RolledUp e2 ∧ Agent e2 x ∧
ElbowHeight e2 ∧ Drink y ∧ MartiniGlass z ∧ Making e3 ∧ Agent e3 x ∧ Patient e3 y ∧ In z y"
(* Hypothesis: A person in black *)
shows "∃x. Person x ∧ InBlack x"
proof -
  from asm have "Bartender x" by blast
  then have "Person x" using explanation_1 by blast
  from asm have "Black e1 ∧ Dress e1 ∧ Agent e1 x" by blast
  then have "InBlack x" using `Person x` explanation_2 by blast
  show ?thesis using `Person x` `InBlack x` by blast
qed

end
```

Figure 27: The Isabelle theory code for table 3 iteration 1

```
theory question_66_2

imports Main

begin
typedecl entity
typedecl event

consts
  Bartender :: "entity ⇒ bool"
  Person :: "entity ⇒ bool"
  DressedInBlack :: "entity ⇒ bool"
  InBlack :: "entity ⇒ bool"
  Male :: "entity ⇒ bool"
  SleevesRolledUpToElbowHeight :: "entity ⇒ bool"
  Drink :: "entity ⇒ bool"
  MartiniGlass :: "entity ⇒ bool"
  Making :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  In :: "entity ⇒ entity ⇒ bool"

(* Explanation 1: A bartender is a person. *)
axiomatization where
    explanation_1: "∀x. Bartender x ⟶ Person x"

(* Explanation 2: If a person is dressed in black, then the person is in black. *)
axiomatization where
    explanation_2: "∀x. Person x ∧ DressedInBlack x ⟶ InBlack x"

theorem hypothesis:
(* Premise: A male bartender dressed in all black with his sleeves rolled up to elbow height making a drink in a
martini glass. *)
assumes asm: "Male x ∧ Bartender x ∧ DressedInBlack x ∧ SleevesRolledUpToElbowHeight x ∧ Drink y ∧ MartiniGlass z ∧
Making e ∧ Agent e x ∧ Patient e y ∧ In z y"
(* Hypothesis: A person in black *)
shows "∃x. Person x ∧ InBlack x"
proof -
  from asm have "Bartender x ∧ DressedInBlack x" by blast
  then have "Person x ∧ DressedInBlack x" using explanation_1 by blast
  then have "Person x ∧ InBlack x" using explanation_2 by blast
  then show ?thesis by blast
qed

end
```

Figure 28: The Isabelle theory code for table 3 iteration 2

Dataset	Sentences	Explanation	Iteration	Validity
QASC	**Hypothesis**: Meiosis is the type of cell division that produces haploid cells.	Meiosis is the type of cell division that produces gametes. Gametes are haploid egg or sperm that fuse to form a zygote.	0	Invalid
QASC	**Hypothesis**: Meiosis is the type of cell division that produces haploid cells.	Meiosis is the type of cell division that produces gametes. Gametes are haploid cells. Haploid cells are either egg or sperm that fuse to form a zygote.	1	Valid

Table 4: An example of how the explanation sentences in QASC can be refined with Explanation-Refiner

```
theory question_67_0

imports Main

begin

typedecl entity
typedecl event

consts
  Meiosis :: "entity ⇒ bool"
  Gametes :: "entity ⇒ bool"
  Produces :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  HaploidEggOrSperm :: "entity ⇒ bool"
  Zygote :: "entity ⇒ bool"
  Fuse :: "event ⇒ bool"
  Form :: "event ⇒ bool"
  HaploidCells :: "entity ⇒ bool"

(* Explanation 1: Meiosis is the type of cell division that produces gametes. *)
axiomatization where
    explanation_1: "∀x y e. Meiosis x ∧ Gametes y ∧ Produces e ∧ Agent e x ∧ Patient e y"

(* Explanation 2: Gametes are haploid egg or sperm that fuse to form a zygote. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. Gametes x ∧ HaploidEggOrSperm y ∧ Zygote z ∧ Fuse e1 ∧ Agent e1 y ∧ Patient e1 x
∧ Form e2 ∧ Agent e2 y ∧ Patient e2 z"

theorem hypothesis:
assumes asm: "Meiosis x"
(* Hypothesis: Meiosis is the type of cell division that produces haploid cells. *)
shows "∃x y e. Meiosis x ∧ HaploidCells y ∧ Produces e ∧ Agent e x ∧ Patient e y"
proof -
  from asm have "Meiosis x" by simp
  then obtain y e where e: "Gametes y ∧ Produces e ∧ Agent e x ∧ Patient e y" using explanation_1 by blast
  then have "HaploidCells y" using explanation_2 by blast
  have conclusion: "Produces e ∧ Agent e x ∧ Patient e y" using e by simp
  show ?thesis using asm conclusion `HaploidCells y` by blast
qed

end
```

Figure 29: The Isabelle theory code for table 4 iteration 0

```
theory question_67_1

imports Main

begin

typedecl entity
typedecl event

consts
  Meiosis :: "entity ⇒ bool"
  CellDivision :: "entity ⇒ bool"
  Gametes :: "entity ⇒ bool"
  Produces :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  HaploidCells :: "entity ⇒ bool"
  Egg :: "entity ⇒ bool"
  Sperm :: "entity ⇒ bool"
  Fuse :: "event ⇒ bool"
  Form :: "event ⇒ bool"
  Zygote :: "entity ⇒ bool"

(* Explanation 1: Meiosis is the type of cell division that produces gametes. *)
axiomatization where
    explanation_1: "∀x y e. Meiosis x ∧ CellDivision x ∧ Gametes y ∧ Produces e ∧ Agent e x ∧ Patient e y"

(* Explanation 2: Gametes are haploid cells. *)
axiomatization where
    explanation_2: "∀x. Gametes x ⟶ HaploidCells x"

(* Explanation 3: Haploid cells are either egg or sperm that fuse to form a zygote. *)
axiomatization where
    explanation_3: "∀x y z e1 e2. HaploidCells x ∧ Egg y ∧ Sperm z ∧ Fuse e1 ∧ Agent e1 y ∧ Agent e1 z ⟶ Form e2 ∧
Patient e2 x ∧ Zygote x"

theorem hypothesis:
assumes asm: "Meiosis x ∧ CellDivision x"
(* Hypothesis: Meiosis is the type of cell division that produces haploid cells. *)
shows "∃x y e. Meiosis x ∧ CellDivision x ∧ HaploidCells y ∧ Produces e ∧ Agent e x ∧ Patient e y"
proof -
  from asm have "Meiosis x ∧ CellDivision x" by simp
  then obtain y e where e: "Gametes y ∧ Produces e ∧ Agent e x ∧ Patient e y" using explanation_1 by blast
  then have "HaploidCells y" using explanation_2 by blast
  have conclusion: "Produces e ∧ Agent e x ∧ Patient e y" using e by simp
  show ?thesis using asm conclusion `HaploidCells y` by blast
qed

end
```

Figure 30: The Isabelle theory code for table 4 iteration 1

Dataset	Sentences	Explanation	Iteration	Validity
QASC	**Hypothesis**: Antigens are found on cancer cells and the cells of transplanted kidneys.	Antigens are found on cancer cells and the cells of transplanted organs. Examples of animal organs are kidneys and hearts.	0	Invalid
QASC	**Hypothesis**: Antigens are found on cancer cells and the cells of transplanted kidneys.	Antigens are found on cancer cells and the cells of transplanted organs. Kidneys are examples of animal organs. Transplanted organs include kidneys.	1	Valid

Table 5: An example of how the explanation sentences in QASC can be refined with Explanation-Refiner

```
theory question_36_0

imports Main

begin

typedecl entity
typedecl event

consts
  Antigens :: "entity ⇒ bool"
  CancerCells :: "entity ⇒ bool"
  TransplantedOrgans :: "entity ⇒ bool"
  Found :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Kidneys :: "entity ⇒ bool"
  Hearts :: "entity ⇒ bool"
  AnimalOrgans :: "entity ⇒ bool"
  TransplantedKidneys :: "entity ⇒ bool"

(* Explanation 1: Antigens are found on cancer cells and the cells of transplanted organs. *)
axiomatization where
    explanation_1: "∀x y z e. Antigens x ∧ CancerCells y ∧ TransplantedOrgans z ∧ Found e ∧ Agent e x ∧ Patient e y
∧ Patient e z"

(* Explanation 2: Examples of animal organs are kidneys and hearts. *)
axiomatization where
    explanation_2: "∀x y. Kidneys x ∧ Hearts y ⟶ AnimalOrgans x ∧ AnimalOrgans y"

theorem hypothesis:
assumes asm: "Antigens x ∧ CancerCells y ∧ TransplantedKidneys z"
(* Hypothesis: Antigens are found on cancer cells and the cells of transplanted kidneys. *)
shows "∃x y z e. Antigens x ∧ CancerCells y ∧ TransplantedKidneys z ∧ Found e ∧ Agent e x ∧ Patient e y ∧ Patient e
z"
proof -
  from asm have "Antigens x ∧ CancerCells y ∧ TransplantedKidneys z" by simp
  then obtain e where e: "Found e ∧ Agent e x ∧ Patient e y ∧ Patient e z" using explanation_1 by blast
  have "Kidneys z" using explanation_2 by blast
  have conclusion: "Found e ∧ Agent e x ∧ Patient e y ∧ Patient e z" using e by simp
  show ?thesis using asm conclusion `Kidneys z` by blast
qed

end
```

Figure 31: The Isabelle theory code for table 5 iteration 0

```
theory question_36_1

imports Main

begin

typedecl entity
typedecl event

consts
  Antigens :: "entity ⇒ bool"
  CancerCells :: "entity ⇒ bool"
  TransplantedOrgans :: "entity ⇒ bool"
  Found :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Kidneys :: "entity ⇒ bool"
  AnimalOrgans :: "entity ⇒ bool"
  Include :: "event ⇒ bool"
  TransplantedKidneys :: "entity ⇒ bool"

(* Explanation 1: Antigens are found on cancer cells and the cells of transplanted organs. *)
axiomatization where
    explanation_1: "∀x y z e. Antigens x ∧ CancerCells y ∧ TransplantedOrgans z ∧ Found e ∧ Agent e x ∧ Patient e y
∧ Patient e z"

(* Explanation 2: Kidneys are examples of animal organs. *)
axiomatization where
    explanation_2: "∀x. Kidneys x ⟶ AnimalOrgans x"

(* Explanation 3: Transplanted organs include kidneys. *)
axiomatization where
    explanation_3: "∀x y e. TransplantedOrgans x ∧ Kidneys y ∧ Include e ∧ Agent e x ∧ Patient e y"

theorem hypothesis:
assumes asm: "Antigens x ∧ CancerCells y ∧ TransplantedKidneys z"
(* Hypothesis: Antigens are found on cancer cells and the cells of transplanted kidneys. *)
shows "∃x y z e. Antigens x ∧ CancerCells y ∧ TransplantedKidneys z ∧ Found e ∧ Agent e x ∧ Patient e y ∧ Patient e
z"
proof -
  from asm have "Antigens x ∧ CancerCells y ∧ TransplantedKidneys z" by simp
  then obtain e1 where e1: "Found e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Patient e1 z" using explanation_1 by blast
  then have "Kidneys z" using explanation_3 by blast
  have conclusion: "Found e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Patient e1 z" using e1 by simp
  show ?thesis using asm conclusion `Kidneys z` by blast
qed

end
```

Figure 32: The Isabelle theory code for table 5 iteration 1

Dataset	Sentences	Explanation	Iteration	Validity
QASC	**Hypothesis**: Azathioprine interferes with the assembly of proteins.	DNA contains genetic instructions for proteins, and RNA helps assemble the proteins. Azathioprine interferes with DNA and RNA synthesis	0	Invalid
QASC	**Hypothesis**: Azathioprine interferes with the assembly of proteins.	DNA contains genetic instructions for proteins, and RNA, synthesized from DNA, helps assemble the proteins. The assembly of proteins is dependent on the synthesis of DNA and RNA. Azathioprine interferes with DNA and RNA synthesis.	1	Valid

Table 6: An example of how the explanation sentences in QASC can be refined with Explanation-Refiner

```
theory question_92_0

imports Main

begin

typedecl entity
typedecl event

consts
  DNA :: "entity ⇒ bool"
  GeneticInstructions :: "entity ⇒ bool"
  Proteins :: "entity ⇒ bool"
  Contains :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  RNA :: "entity ⇒ bool"
  Helps :: "entity ⇒ bool"
  Assemble :: "event ⇒ bool"
  HelpsAssemble :: "entity ⇒ event ⇒ bool"
  Azathioprine :: "entity ⇒ bool"
  Interferes :: "event ⇒ bool"
  Synthesis :: "event ⇒ bool"
  InterferesWith :: "event ⇒ event ⇒ bool"
  Assembly :: "event ⇒ bool"

(* Explanation 1: DNA contains genetic instructions for proteins, and RNA helps assemble the proteins. *)
axiomatization where
    explanation_1: "∀x y z e1 e2 e3. DNA x ∧ GeneticInstructions y ∧ Proteins z ∧ Contains e1 ∧ Agent e1 x ∧
Patient e1 y ∧ RNA e2 ∧ Helps e2 ∧ Assemble e3 ∧ Agent e3 e2 ∧ Patient e3 z ⟶ HelpsAssemble e2 e3"

(* Explanation 2: Azathioprine interferes with DNA and RNA synthesis. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. Azathioprine x ∧ DNA y ∧ RNA z ∧ Interferes e1 ∧ Agent e1 x ∧ Synthesis e2 ∧
Patient e2 y ∧ Patient e2 z ⟶ InterferesWith e1 e2"

theorem hypothesis:
assumes asm: "Azathioprine x ∧ Proteins y"
(* Hypothesis: Azathioprine interferes with the assembly of proteins. *)
shows "∃x y e1 e2. Azathioprine x ∧ Proteins y ∧ Interferes e1 ∧ Agent e1 x ∧ Assembly e2 ∧ Patient e2 y ∧
InterferesWith e1 e2"
proof -
  from asm have "Azathioprine x ∧ Proteins y" by simp
  then obtain e1 e2 where e1: "DNA x ∧ GeneticInstructions y ∧ Proteins z ∧ Contains e1 ∧ Agent e1 x ∧ Patient e1 y
∧ RNA e2 ∧ Helps e2 ∧ Assemble e3 ∧ Agent e3 e2 ∧ Patient e3 z" using explanation_1 by blast
  then obtain e3 where e3: "Azathioprine x ∧ DNA y ∧ RNA z ∧ Interferes e3 ∧ Agent e3 x ∧ Synthesis e2 ∧ Patient e2
y ∧ Patient e2 z" using explanation_2 by blast
  have conclusion: "Interferes e3 ∧ Agent e3 x ∧ Assembly e2 ∧ Patient e2 y ∧ InterferesWith e3 e2" using e3 by
simp
  show ?thesis using asm conclusion by blast
qed

end
```

Figure 33: The Isabelle theory code for table 6 iteration 0

```
theory question_92_1
imports Main

begin

typedecl entity
typedecl event

consts
  DNA :: "entity ⇒ bool"
  Proteins :: "entity ⇒ bool"
  GeneticInstructions :: "entity ⇒ bool"
  Contains :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Synthesized :: "event ⇒ bool"
  RNA :: "entity ⇒ bool"
  Helps :: "event ⇒ bool"
  Assemble :: "event ⇒ bool"
  Assembly :: "entity ⇒ bool"
  Synthesis :: "event ⇒ entity ⇒ bool"
  Dependent :: "event ⇒ bool"
  Azathioprine :: "entity ⇒ bool"
  Interferes :: "event ⇒ bool"

(* Explanation 1: DNA contains genetic instructions for proteins, and RNA, synthesized from DNA, helps assemble the
proteins. *)
axiomatization where
    explanation_1: "∀x y z e1 e2 e3. DNA x ∧ Proteins y ∧ GeneticInstructions z ∧ Contains e1 ∧ Agent e1 x ∧
Patient e1 z ∧ Synthesized e2 ∧ Agent e2 x ∧ RNA x ∧ Helps e3 ∧ Agent e3 x ∧ Assemble e3 ∧ Patient e3 y"

(* Explanation 2: The assembly of proteins is dependent on the synthesis of DNA and RNA. *)
axiomatization where
    explanation_2: "∀x y e1 e2. Proteins x ∧ DNA y ∧ RNA y ∧ Assembly x ∧ Synthesis e1 y ∧ Agent e1 y ∧ Dependent
e2 ∧ Agent e2 x ∧ Patient e2 x"

(* Explanation 3: Azathioprine interferes with DNA and RNA synthesis. *)
axiomatization where
    explanation_3: "∀x y z e. Azathioprine x ∧ DNA y ∧ RNA z ∧ Synthesis e y ∧ Synthesis e z ∧ Interferes e ∧ Agent
e x ∧ Patient e y ∧ Patient e z"

theorem hypothesis:
assumes asm: "Azathioprine x ∧ Proteins y"
(* Hypothesis: Azathioprine interferes with the assembly of proteins. *)
shows "∃x y e. Azathioprine x ∧ Proteins y ∧ Interferes e ∧ Agent e x ∧ Patient e y"
proof -
  from asm have "Azathioprine x ∧ Proteins y" by simp
  then obtain e1 where e1: "Contains e1 ∧ Agent e1 x ∧ Patient e1 z ∧ Synthesized e2 ∧ Agent e2 x ∧ RNA x ∧ Helps
e3 ∧ Agent e3 x ∧ Assemble e3 ∧ Patient e3 y" using explanation_1 by blast
  then obtain e2 where e2: "Assembly x ∧ Synthesis e1 y ∧ Agent e1 y ∧ Dependent e2 ∧ Agent e2 x ∧ Patient e2 x"
using explanation_2 by blast
  then have "Interferes e ∧ Agent e x ∧ Patient e y" using explanation_3 by blast
  have conclusion: "Interferes e ∧ Agent e x ∧ Patient e y" using e1 e2 `Interferes e ∧ Agent e x ∧ Patient e y` by
simp
  show ?thesis using asm conclusion by blast
qed

end
```

Figure 34: The Isabelle theory code for table 6 iteration 1

Dataset	Sentences	Explanation	Iteration	Validity
WorldTree	**Hypothesis**: Plugging many appliances into one outlet is not a way to stay safe around electricity.	when an electrical conductor is plugged into an outlet , a circuit is completed. a complete electrical circuit is a source of electrical energy. electricity means electrical energy. safe is the opposite of harm; danger. plugging many appliances into one outlet is dangerous. to be in danger means to be around; to be near something dangerous. electrocution causes harm to an organism. if electricity flows through; is transferred through the body of an animal then that animal is electrocuted. an animal is a kind of organism.	0	Invalid
WorldTree	**Hypothesis**: Plugging many appliances into one outlet is not a way to stay safe around electricity.	if an event is dangerous, it is not safe. plugging many appliances into one outlet is dangerous.	1	Invalid
WorldTree	**Hypothesis**: Plugging many appliances into one outlet is not a way to stay safe around electricity.	if an event is dangerous, it is not safe. plugging many appliances into one outlet is dangerous. An event is considered safe around electricity if it is safe.	2	Invalid
WorldTree	**Hypothesis**: Plugging many appliances into one outlet is not a way to stay safe around electricity.	if an event is dangerous, it is not safe. plugging many appliances into one outlet is dangerous. An event is considered not safe around electricity if it is not safe.	3	Invalid
WorldTree	**Hypothesis**: Plugging many appliances into one outlet is not a way to stay safe around electricity.	if an event is dangerous, it is not safe. plugging many appliances into one outlet is dangerous. If an event is not safe, it is considered not safe around electricity.	4	Invalid
WorldTree	**Hypothesis**: Plugging many appliances into one outlet is not a way to stay safe around electricity.	if an event is dangerous, it is not safe. plugging many appliances into one outlet is dangerous. If an event related to electricity is not safe, it is considered not safe around electricity.	5	Invalid
WorldTree	**Hypothesis**: Plugging many appliances into one outlet is not a way to stay safe around electricity.	if an event is dangerous, it is not safe. plugging many appliances into one outlet is dangerous. Plugging many appliances into one outlet is an event related to electricity. If an event related to electricity is not safe, it is considered not safe around electricity.	6	Invalid
WorldTree	**Hypothesis**: Plugging many appliances into one outlet is not a way to stay safe around electricity.	if an event is dangerous, it is not safe. plugging many appliances into one outlet is dangerous. Plugging many appliances into one outlet is an event related to electricity. If an event related to electricity is not safe, it is considered not safe around electricity. If an event is not safe around electricity, it is not a way to stay safe around electricity.	7	Valid

Table 7: An example of how the explanation sentences in WorldTree can be refined with Explanation-Refiner

```
theory question_11_0

imports Main

begin

typedecl entity
typedecl event

consts
  ElectricalConductor :: "entity ⇒ bool"
  Outlet :: "entity ⇒ bool"
  Circuit :: "entity ⇒ bool"
  Plugged :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
...
(* Explanation 1: when an electrical conductor is plugged into an outlet , a circuit is completed. *)
axiomatization where
    explanation_1: "∀x y z e1 e2. ElectricalConductor x ∧ Outlet y ∧ Circuit z ∧ Plugged e1 ∧ Agent e1 x ∧ Patient
e1 y ⟶ Completed e2 ∧ Agent e2 z"

(* Explanation 2: a complete electrical circuit is a source of electrical energy. *)
axiomatization where
    explanation_2: "∀x y. CompleteElectricalCircuit x ⟶ SourceOfElectricalEnergy y"

(* Explanation 3: electricity means electrical energy. *)
axiomatization where
    explanation_3: "∀x y. Electricity x ⟷ ElectricalEnergy y"

(* Explanation 4: safe is the opposite of harm; danger. *)
axiomatization where
    explanation_4: "∀x y z. Safe x ⟷ ¬Harm y ∧ ¬Danger z"

(* Explanation 5: plugging many appliances into one outlet is dangerous. *)
axiomatization where
    explanation_5: "∀x y e. Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶ Dangerous e"

(* Explanation 6: to be in danger means to be around; to be near something dangerous. *)
axiomatization where
    explanation_6: "∀x y. InDanger x ⟷ AroundDangerous y"

(* Explanation 7: electrocution causes harm to an organism. *)
axiomatization where
    explanation_7: "∀x y e. Electrocution x ∧ Organism y ∧ Causes e ∧ Agent e x ∧ Patient e y"

(* Explanation 8: if electricity flows through; is transferred through the body of an animal then that animal is
electrocuted. *)
axiomatization where
    explanation_8: "(∃e1 e2 x y. Electricity x ∧ BodyOfAnimal y ∧ Flows e1 ∧ Agent e1 x ∧ Patient e1 y ∧
Transferred e2 ∧ Agent e2 x ∧ Patient e2 y) ⟶ (∃e3 z. Animal z ∧ Electrocuted e3 ∧ Agent e3 z)"

(* Explanation 9: an animal is a kind of organism. *)
axiomatization where
    explanation_9: "∀x. Animal x ⟶ Organism x"

theorem hypothesis:
assumes asm: "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y"
(* Hypothesis: Plugging many appliances into one outlet is not a way to stay safe around electricity. *)
shows "¬SafeAround e z"
proof -
  from asm have "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y" by simp
  then have "Dangerous e" using explanation_5 by blast
  then have "¬Safe e" using explanation_4 by blast
  show ?thesis using asm `¬Safe e` by blast
qed

end
```

Figure 35: The Isabelle theory code for table 7 iteration 0

```
theory question_11_1

imports Main

begin

typedecl entity
typedecl event

consts
  Dangerous :: "event ⇒ bool"
  Safe :: "event ⇒ bool"
  Appliances :: "entity ⇒ bool"
  Outlet :: "entity ⇒ bool"
  Plugging :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Electricity :: "entity ⇒ bool"
  SafeAround :: "event ⇒ entity ⇒ bool"

(* Explanation 1: if an event is dangerous, it is not safe. *)
axiomatization where
    explanation_1: "∀e. Dangerous e ⟶ ¬Safe e"

(* Explanation 2: plugging many appliances into one outlet is dangerous. *)
axiomatization where
    explanation_2: "∀x y e. Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶ Dangerous e"

theorem hypothesis:
assumes asm: "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y"
(* Hypothesis: Plugging many appliances into one outlet is not a way to stay safe around electricity. *)
shows "¬SafeAround e z"
proof -
  from asm have "Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y" by simp
  then have "Dangerous e" using explanation_2 by blast
  then have "¬Safe e" using explanation_1 by blast
  then show ?thesis using asm `¬Safe e` by blast
qed

end
```

Figure 36: The Isabelle theory code for table 7 iteration 1

```
theory question_11_2

imports Main

begin

typedecl entity
typedecl event

consts
  Dangerous :: "event ⇒ bool"
  Safe :: "event ⇒ bool"
  Appliances :: "entity ⇒ bool"
  Outlet :: "entity ⇒ bool"
  Plugging :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Event :: "event ⇒ bool"
  Electricity :: "entity ⇒ bool"
  SafeAround :: "event ⇒ entity ⇒ bool"

(* Explanation 1: if an event is dangerous, it is not safe. *)
axiomatization where
    explanation_1: "∀e. Dangerous e ⟶ ¬Safe e"

(* Explanation 2: plugging many appliances into one outlet is dangerous. *)
axiomatization where
    explanation_2: "∀x y e. Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶ Dangerous e"

(* Explanation 3: An event is considered safe around electricity if it is safe. *)
axiomatization where
    explanation_3: "∀e z. Event e ∧ Electricity z ∧ Safe e ⟶ SafeAround e z"

theorem hypothesis:
assumes asm: "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y"
(* Hypothesis: Plugging many appliances into one outlet is not a way to stay safe around electricity. *)
shows "¬SafeAround e z"
proof -
  from asm have "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y" by simp
  then have "Dangerous e" using explanation_2 by blast
  then have "¬Safe e" using explanation_1 by blast
  then show ?thesis using explanation_3 `Electricity z` by blast
qed

end
```

Figure 37: The Isabelle theory code for table 7 iteration 2

```
theory question_11_3

imports Main

begin

typedecl entity
typedecl event

consts
  Dangerous :: "event ⇒ bool"
  Safe :: "event ⇒ bool"
  Appliances :: "entity ⇒ bool"
  Outlet :: "entity ⇒ bool"
  Plugging :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Event :: "event ⇒ bool"
  Electricity :: "entity ⇒ bool"
  SafeAround :: "event ⇒ entity ⇒ bool"

(* Explanation 1: if an event is dangerous, it is not safe. *)
axiomatization where
    explanation_1: "∀e. Dangerous e ⟶ ¬Safe e"

(* Explanation 2: plugging many appliances into one outlet is dangerous. *)
axiomatization where
    explanation_2: "∀x y e. Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶ Dangerous e"

(* Explanation 3: An event is considered not safe around electricity if it is not safe. *)
axiomatization where
    explanation_3: "∀e z. Event e ∧ Electricity z ∧ ¬Safe e ⟶ ¬SafeAround e z"

theorem hypothesis:
assumes asm: "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y"
(* Hypothesis: Plugging many appliances into one outlet is not a way to stay safe around electricity. *)
shows "∃x y z e. Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y ∧ ¬SafeAround e z"
proof -
from asm have "Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y" by simp
then have "Dangerous e" using explanation_2 by blast
then have "¬Safe e" using explanation_1 by blast
then have "¬SafeAround e z" using explanation_3 asm by blast
show ?thesis using asm `¬SafeAround e z` by blast
qed

end
```

Figure 38: The Isabelle theory code for table 7 iteration 3

```
theory question_11_4

imports Main

begin

typedecl entity
typedecl event

consts
  Dangerous :: "event ⇒ bool"
  Safe :: "event ⇒ bool"
  Appliances :: "entity ⇒ bool"
  Outlet :: "entity ⇒ bool"
  Plugging :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Electricity :: "entity ⇒ bool"
  SafeAround :: "event ⇒ entity ⇒ bool"

(* Explanation 1: if an event is dangerous, it is not safe. *)
axiomatization where
    explanation_1: "∀e. Dangerous e ⟶ ¬Safe e"

(* Explanation 2: plugging many appliances into one outlet is dangerous. *)
axiomatization where
    explanation_2: "∀x y e. Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶ Dangerous e"

(* Explanation 3: If an event is not safe, it is considered not safe around electricity. *)
axiomatization where
    explanation_3: "∀e z. ¬Safe e ∧ Electricity z ⟶ ¬SafeAround e z"

theorem hypothesis:
assumes asm: "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y"
(* Hypothesis: Plugging many appliances into one outlet is not a way to stay safe around electricity. *)
shows "¬SafeAround e z"
proof -
  from asm have "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y" by simp
  then have "Dangerous e" using explanation_2 by blast
  then have "¬Safe e" using explanation_1 by blast
  then show ?thesis using explanation_3 `Electricity z` by blast
qed

end
```

Figure 39: The Isabelle theory code for table 7 iteration 4

```
theory question_11_5

imports Main

begin

typedecl entity
typedecl event

consts
  Dangerous :: "event ⇒ bool"
  Safe :: "event ⇒ bool"
  Appliances :: "entity ⇒ bool"
  Outlet :: "entity ⇒ bool"
  Plugging :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Event :: "event ⇒ bool"
  RelatedTo :: "event ⇒ entity ⇒ bool"
  Electricity :: "entity ⇒ bool"
  SafeAround :: "event ⇒ entity ⇒ bool"

(* Explanation 1: if an event is dangerous, it is not safe. *)
axiomatization where
    explanation_1: "∀e. Dangerous e ⟶ ¬Safe e"

(* Explanation 2: plugging many appliances into one outlet is dangerous. *)
axiomatization where
    explanation_2: "∀x y e. Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶ Dangerous e"

(* Explanation 3: If an event related to electricity is not safe, it is considered not safe around electricity. *)
axiomatization where
    explanation_3: "∀e z. Event e ∧ RelatedTo e z ∧ Electricity z ∧ ¬Safe e ⟶ ¬SafeAround e z"

theorem hypothesis:
assumes asm: "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y"
(* Hypothesis: Plugging many appliances into one outlet is not a way to stay safe around electricity. *)
shows "∃x y z e. Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y ∧ ¬SafeAround e z"
proof -
  from asm have "Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y" by simp
  then have "Dangerous e" using explanation_2 by blast
  then have "¬Safe e" using explanation_1 by blast
  then have "¬SafeAround e z" using explanation_3 asm by blast
  show ?thesis using asm `¬SafeAround e z` by blast
qed

end
```

Figure 40: The Isabelle theory code for table 7 iteration 5

```
theory question_11_6

imports Main

begin

typedecl entity
typedecl event

consts
  Dangerous :: "event ⇒ bool"
  Safe :: "event ⇒ bool"
  Appliances :: "entity ⇒ bool"
  Outlet :: "entity ⇒ bool"
  Plugging :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Related :: "event ⇒ entity ⇒ bool"
  Electricity :: "entity"
  SafeAround :: "event ⇒ entity ⇒ bool"
  StaySafe :: "event ⇒ bool"
  Around :: "event ⇒ entity ⇒ bool"

(* Explanation 1: if an event is dangerous, it is not safe. *)
axiomatization where
    explanation_1: "∀e. Dangerous e ⟶ ¬Safe e"

(* Explanation 2: plugging many appliances into one outlet is dangerous. *)
axiomatization where
    explanation_2: "∀x y e. Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶ Dangerous e"

(* Explanation 3: Plugging many appliances into one outlet is an event related to electricity. *)
axiomatization where
    explanation_3: "∀x y e. Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶ Related e
Electricity"

(* Explanation 4: If an event related to electricity is not safe, it is considered not safe around electricity. *)
axiomatization where
    explanation_4: "∀e. Related e Electricity ∧ ¬Safe e ⟶ ¬SafeAround e Electricity"

theorem hypothesis:
assumes asm: "Appliances x ∧ Outlet y ∧ Plugging e1 ∧ Agent e1 x ∧ Patient e1 y"
(* Hypothesis: Plugging many appliances into one outlet is not a way to stay safe around electricity. *)
shows "∃x y e1 e2. Appliances x ∧ Outlet y ∧ Plugging e1 ∧ Agent e1 x ∧ Patient e1 y ∧ ¬StaySafe e2 ∧ Around e2
Electricity"
proof -
  from asm have "Appliances x ∧ Outlet y ∧ Plugging e1 ∧ Agent e1 x ∧ Patient e1 y" by simp
  then have "Dangerous e1" using explanation_2 by blast
  then have "¬Safe e1" using explanation_1 by blast
  then have "Related e1 Electricity" using explanation_3 asm by blast
  then have "¬SafeAround e1 Electricity" using explanation_4 `¬Safe e1` by blast
  then have "¬StaySafe e2 ∧ Around e2 Electricity" by simp
  show ?thesis using asm `¬StaySafe e2 ∧ Around e2 Electricity` by blast
qed

end
```

Figure 41: The Isabelle theory code for table 7 iteration 6

```
theory question_11_7

imports Main

begin

typedecl entity
typedecl event

consts
Dangerous :: "event ⇒ bool"
Safe :: "event ⇒ bool"
Appliances :: "entity ⇒ bool"
Outlet :: "entity ⇒ bool"
Plugging :: "event ⇒ bool"
Agent :: "event ⇒ entity ⇒ bool"
Patient :: "event ⇒ entity ⇒ bool"
Electricity :: "entity ⇒ bool"
RelatedTo :: "event ⇒ entity ⇒ bool"
SafeAround :: "event ⇒ entity ⇒ bool"
WayToStaySafe :: "event ⇒ entity ⇒ bool"

(* Explanation 1: if an event is dangerous, it is not safe. *)
axiomatization where
    explanation_1: "∀e. Dangerous e ⟶ ¬Safe e"

(* Explanation 2: plugging many appliances into one outlet is dangerous. *)
axiomatization where
    explanation_2: "∀x y e. Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶ Dangerous e"

(* Explanation 3: Plugging many appliances into one outlet is an event related to electricity. *)
axiomatization where
    explanation_3: "∀x y z e. Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y ⟶
RelatedTo e z"

(* Explanation 4: If an event related to electricity is not safe, it is considered not safe around electricity. *)
axiomatization where
    explanation_4: "∀e z. RelatedTo e z ∧ ¬Safe e ⟶ ¬SafeAround e z"

(* Explanation 5: If an event is not safe around electricity, it is not a way to stay safe around electricity. *)
axiomatization where
    explanation_5: "∀e z. ¬SafeAround e z ⟶ ¬WayToStaySafe e z"

theorem hypothesis:
assumes asm: "Appliances x ∧ Outlet y ∧ Electricity z ∧ Plugging e ∧ Agent e x ∧ Patient e y"
(* Hypothesis: Plugging many appliances into one outlet is not a way to stay safe around electricity. *)
shows "¬SafeAround e z"
proof -
  from asm have "Appliances x ∧ Outlet y ∧ Plugging e ∧ Agent e x ∧ Patient e y" by simp
  then have "Dangerous e" using explanation_2 by blast
  then have "¬Safe e" using explanation_1 by blast
  have "RelatedTo e z" using asm explanation_3 by blast
  then have "¬SafeAround e z" using `¬Safe e` explanation_4 by blast
  show ?thesis using `¬SafeAround e z` by simp
qed

end
```

Figure 42: The Isabelle theory code for table 7 iteration 7

Dataset	Sentences	Explanation	Iteration	Validity
WorldTree	**Hypothesis**: Sound energy can travel by vibrating particles of air.	sound is a kind of energy. sound is a kind of air vibration. sound can travel through air. form means kind.	0	Invalid
WorldTree	**Hypothesis**: Sound energy can travel by vibrating particles of air.	Sound energy is a kind of energy. Sound is a kind of air vibration. Sound can travel through air. Form means kind.	1	Invalid
WorldTree	**Hypothesis**: Sound energy can travel by vibrating particles of air.	Sound energy is the energy that sound carries. Sound is a kind of air vibration. Sound can travel through air by vibrating particles of air.	2	Invalid
WorldTree	**Hypothesis**: Sound energy can travel by vibrating particles of air.	Sound energy is the energy that sound carries. Sound energy is a kind of air vibration. Sound can travel through air by vibrating particles of air.	3	Invalid
WorldTree	**Hypothesis**: Sound energy can travel by vibrating particles of air.	Sound energy is the energy that sound carries. Sound energy can cause air vibration. Sound can travel through air by vibrating particles of air.	4	Valid

Table 8: An example of how the explanation sentences in WorldTree can be refined with Explanation-Refiner

```
theory question_59_0

imports Main

begin

typedecl entity
typedecl event

consts
  Sound :: "entity ⇒ bool"
  Energy :: "entity ⇒ bool"
  AirVibration :: "entity ⇒ bool"
  Air :: "entity ⇒ bool"
  Travel :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Through :: "event ⇒ entity ⇒ bool"
  Form :: "entity ⇒ bool"
  Kind :: "entity ⇒ bool"
  SoundEnergy :: "entity ⇒ bool"
  ParticlesOfAir :: "entity ⇒ bool"
  Vibrating :: "event ⇒ bool"
  Patient :: "event ⇒ event ⇒ bool"

(* Explanation 1: sound is a kind of energy. *)
axiomatization where
    explanation_1: "∀x. Sound x ⟶ Energy x"

(* Explanation 2: sound is a kind of air vibration. *)
axiomatization where
    explanation_2: "∀x. Sound x ⟶ AirVibration x"

(* Explanation 3: sound can travel through air. *)
axiomatization where
    explanation_3: "∀x y e. Sound x ∧ Air y ∧ Travel e ∧ Agent e x ∧ Through e y"

(* Explanation 4: form means kind. *)
axiomatization where
    explanation_4: "∀x y. Form x ⟷ Kind y"

theorem hypothesis:
assumes asm: "SoundEnergy x ∧ ParticlesOfAir y"
(* Hypothesis: Sound energy can travel by vibrating particles of air. *)
shows "∃x y e1 e2. SoundEnergy x ∧ ParticlesOfAir y ∧ Travel e1 ∧ Agent e1 x ∧ Vibrating e2 ∧ Agent e2 y ∧ Patient
e1 e2"
proof -
  from asm have "SoundEnergy x" by simp
  then have "Energy x" using explanation_1 by blast
  then have "AirVibration x" using explanation_2 by blast
  from asm have "ParticlesOfAir y" by simp
  then obtain e1 where e1: "Travel e1 ∧ Agent e1 x ∧ Through e1 y" using explanation_3 by blast
  then obtain e2 where e2: "Vibrating e2 ∧ Agent e2 y" using explanation_3 by blast
  have conclusion: "Travel e1 ∧ Agent e1 x ∧ Vibrating e2 ∧ Agent e2 y ∧ Patient e1 e2" using e1 e2 by simp
  show ?thesis using asm conclusion by blast
qed

end
```

Figure 43: The Isabelle theory code for table 8 iteration 0

```
theory question_59_1
imports Main
begin

typedecl entity
typedecl event

consts
  SoundEnergy :: "entity ⇒ bool"
  Energy :: "entity ⇒ bool"
  Sound :: "entity ⇒ bool"
  AirVibration :: "entity ⇒ bool"
  Air :: "entity ⇒ bool"
  Travel :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Form :: "entity ⇒ entity ⇒ bool"
  Kind :: "entity ⇒ entity ⇒ bool"
  ParticlesOfAir :: "entity ⇒ bool"
  Vibrating :: "event ⇒ bool"

(* Explanation 1: Sound energy is a kind of energy. *)
axiomatization where
    explanation_1: "∀x. SoundEnergy x ⟶ Energy x"

(* Explanation 2: Sound is a kind of air vibration. *)
axiomatization where
    explanation_2: "∀x. Sound x ⟶ AirVibration x"

(* Explanation 3: Sound can travel through air. *)
axiomatization where
    explanation_3: "∀x y e. Sound x ∧ Air y ∧ Travel e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: Form means kind. *)
axiomatization where
    explanation_4: "∀x y. Form x y ⟷ Kind x y"

theorem hypothesis:
assumes asm: "SoundEnergy x ∧ ParticlesOfAir y"
(* Hypothesis: Sound energy can travel by vibrating particles of air. *)
shows "∃x y e1 e2. SoundEnergy x ∧ ParticlesOfAir y ∧ Travel e1 ∧ Agent e1 x ∧ Vibrating e2 ∧ Agent e2 y ∧ Patient
e1 y"
proof -
  from asm have "SoundEnergy x" by simp
  then have "Energy x" using explanation_1 by blast
  then have "Sound x" using explanation_2 by blast
  then obtain e1 where e1: "Travel e1 ∧ Agent e1 x ∧ Patient e1 y" using explanation_3 by blast
  then have "ParticlesOfAir y" using asm by simp
  then obtain e2 where e2: "Vibrating e2 ∧ Agent e2 y" using explanation_3 by blast
  show ?thesis using asm e1 e2 by blast
qed

end
```

Figure 44: The Isabelle theory code for table 8 iteration 1

```
theory question_59_2

imports Main

begin

typedecl entity
typedecl event

consts
  Sound :: "entity ⇒ bool"
  Energy :: "entity ⇒ bool"
  SoundEnergy :: "entity ⇒ bool"
  Carries :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  AirVibration :: "entity ⇒ bool"
  Air :: "entity ⇒ bool"
  ParticlesOfAir :: "entity ⇒ bool"
  Travel :: "event ⇒ bool"
  Vibrating :: "event ⇒ bool"

(* Explanation 1: Sound energy is the energy that sound carries. *)
axiomatization where
    explanation_1: "∀x y e. Sound x ∧ Energy y ∧ SoundEnergy x ∧ Carries e ∧ Agent e x ∧ Patient e y"

(* Explanation 2: Sound is a kind of air vibration. *)
axiomatization where
    explanation_2: "∀x. Sound x ⟶ AirVibration x"

(* Explanation 3: Sound can travel through air by vibrating particles of air. *)
axiomatization where
    explanation_3: "∀x y z e1 e2. Sound x ∧ Air y ∧ ParticlesOfAir z ∧ Travel e1 ∧ Agent e1 x ∧ Patient e1 y ∧
Vibrating e2 ∧ Agent e2 z ∧ Patient e2 x"

theorem hypothesis:
assumes asm: "SoundEnergy x ∧ ParticlesOfAir y"
(* Hypothesis: Sound energy can travel by vibrating particles of air. *)
shows "∃x y e1 e2. SoundEnergy x ∧ ParticlesOfAir y ∧ Travel e1 ∧ Agent e1 x ∧ Vibrating e2 ∧ Agent e2 y ∧ Patient
e2 x"
proof -
  from asm have "SoundEnergy x ∧ ParticlesOfAir y" by simp
  then obtain e where e: "Carries e ∧ Agent e x ∧ Patient e y" using explanation_1 by blast
  then have "AirVibration x" using explanation_2 by blast
  then obtain e1 e2 where e1e2: "Travel e1 ∧ Agent e1 x ∧ Vibrating e2 ∧ Agent e2 y ∧ Patient e2 x" using
explanation_3 by blast
  show ?thesis using asm e1e2 by blast
qed

end
```

Figure 45: The Isabelle theory code for table 8 iteration 2

```
theory question_59_3

imports Main

begin

typedecl entity
typedecl event

consts
  Sound :: "entity ⇒ bool"
  Energy :: "entity ⇒ bool"
  SoundEnergy :: "entity ⇒ bool"
  Carries :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  AirVibration :: "entity ⇒ bool"
  Air :: "entity ⇒ bool"
  ParticlesOfAir :: "entity ⇒ bool"
  Travel :: "event ⇒ bool"
  Through :: "event ⇒ entity ⇒ bool"
  Vibrating :: "event ⇒ bool"
  By :: "event ⇒ event ⇒ bool"

(* Explanation 1: Sound energy is the energy that sound carries. *)
axiomatization where
    explanation_1: "∀x y e. Sound x ∧ Energy y ∧ SoundEnergy x ∧ Carries e ∧ Agent e x ∧ Patient e y"

(* Explanation 2: Sound energy is a kind of air vibration. *)
axiomatization where
    explanation_2: "∀x. SoundEnergy x ⟶ AirVibration x"

(* Explanation 3: Sound can travel through air by vibrating particles of air. *)
axiomatization where
    explanation_3: "∀x y z e1 e2. Sound x ∧ Air y ∧ ParticlesOfAir z ∧ Travel e1 ∧ Agent e1 x ∧ Through e1 y ∧
Vibrating e2 ∧ Agent e2 z ∧ By e1 e2"

theorem hypothesis:
assumes asm: "SoundEnergy x ∧ ParticlesOfAir y"
(* Hypothesis: Sound energy can travel by vibrating particles of air. *)
shows "∃x y e1 e2. SoundEnergy x ∧ ParticlesOfAir y ∧ Travel e1 ∧ Agent e1 x ∧ Vibrating e2 ∧ Agent e2 y ∧ By e1
e2"
proof -
  from asm have "SoundEnergy x" and "ParticlesOfAir y" by simp+
  then obtain e where e: "Carries e ∧ Agent e x" using explanation_1 by blast
  then have "AirVibration x" using explanation_2 by blast
  then obtain e1 e2 where e1e2: "Travel e1 ∧ Agent e1 x ∧ Vibrating e2 ∧ Agent e2 y ∧ By e1 e2" using explanation_3
by blast
  show ?thesis using asm e e1e2 by blast
qed

end
```

Figure 46: The Isabelle theory code for table 8 iteration 3

```
theory question_59_4

imports Main

begin

typedecl entity
typedecl event

consts
  Sound :: "entity ⇒ bool"
  Energy :: "entity ⇒ bool"
  Carries :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  SoundEnergy :: "entity ⇒ bool"
  AirVibration :: "entity ⇒ bool"
  Cause :: "event ⇒ bool"
  Air :: "entity ⇒ bool"
  ParticlesOfAir :: "entity ⇒ bool"
  Travel :: "event ⇒ bool"
  Vibrating :: "event ⇒ bool"

(* Explanation 1: Sound energy is the energy that sound carries. *)
axiomatization where
    explanation_1: "∀x y e. Sound x ∧ Energy y ∧ Carries e ∧ Agent e x ∧ Patient e y"

(* Explanation 2: Sound energy can cause air vibration. *)
axiomatization where
    explanation_2: "∀x y e. SoundEnergy x ∧ AirVibration y ∧ Cause e ∧ Agent e x ∧ Patient e y"

(* Explanation 3: Sound can travel through air by vibrating particles of air. *)
axiomatization where
    explanation_3: "∀x y z e1 e2. Sound x ∧ Air y ∧ ParticlesOfAir z ∧ Travel e1 ∧ Agent e1 x ∧ Patient e1 y ∧
Vibrating e2 ∧ Agent e2 z ∧ Patient e2 x"

theorem hypothesis:
assumes asm: "SoundEnergy x ∧ ParticlesOfAir y"
(* Hypothesis: Sound energy can travel by vibrating particles of air. *)
shows "∃x y e1 e2. SoundEnergy x ∧ ParticlesOfAir y ∧ Travel e1 ∧ Agent e1 x ∧ Vibrating e2 ∧ Agent e2 y ∧ Patient
e2 x"
proof -
  from asm have "SoundEnergy x ∧ ParticlesOfAir y" by simp
  then obtain e1 where e1: "Carries e1 ∧ Agent e1 x ∧ Patient e1 y" using explanation_1 by blast
  then obtain e2 where e2: "Cause e2 ∧ Agent e2 x ∧ Patient e2 y" using explanation_2 by blast
  then obtain e3 where e3: "Travel e3 ∧ Agent e3 x ∧ Vibrating e2 ∧ Agent e2 y ∧ Patient e2 x" using explanation_3
by blast
  show ?thesis using asm e1 e2 e3 by blast
qed

end
```

Figure 47: The Isabelle theory code for table 8 iteration 4

Dataset	Sentences	Explanation	Iteration	Validity
WorldTree	**Premise**: most birds sit on their eggs until they hatch. **Hypothesis**: The reason why birds sit on their eggs is to keep the eggs warm.	an egg requires warmth to survive; hatch; develop. sitting on something provides warmth to that something. providing warmth is synonymous with keeping warm. form means kind.	0	Invalid
WorldTree	**Premise**: most birds sit on their eggs until they hatch. **Hypothesis**: The reason why birds sit on their eggs is to keep the eggs warm.	Eggs require warmth to survive, hatch, and develop. Sitting on something provides warmth to that something. Providing warmth is synonymous with keeping warm.	1	Invalid
WorldTree	**Premise**: most birds sit on their eggs until they hatch. **Hypothesis**: The reason why birds sit on their eggs is to keep the eggs warm.	Birds sitting on their eggs provides warmth to the eggs. Providing warmth is synonymous with keeping warm.	2	Invalid
WorldTree	**Premise**: most birds sit on their eggs until they hatch. **Hypothesis**: The reason why birds sit on their eggs is to keep the eggs warm.	Birds sit on their eggs to provide warmth to the eggs. Providing warmth is synonymous with keeping warm.	3	Valid

Table 9: An example of how the explanation sentences in WorldTree can be refined with Explanation-Refiner

```
theory question_92_0

imports Main

begin

typedecl entity
typedecl event

consts
  Egg :: "entity ⇒ bool"
  RequireWarmth :: "event ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Survive :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Hatch :: "event ⇒ bool"
  Develop :: "event ⇒ bool"
  Something :: "entity ⇒ bool" d
  Warmth :: "entity ⇒ bool"
  Sit :: "event ⇒ bool"
  Provide :: "event ⇒ bool"
  KeepWarm :: "event ⇒ bool"
  Birds :: "entity ⇒ bool"
  Eggs :: "entity ⇒ bool"

(* Explanation 1: An egg requires warmth to survive; hatch; develop. *)
axiomatization where
    explanation_1: "∀x e1 e2 e3 e4. Egg x ⟶ (RequireWarmth e1 ∧ Patient e1 x) ∧ (Survive e2 ∧ Agent e2 x) ∧ (Hatch
e3 ∧ Agent e3 x) ∧ (Develop e4 ∧ Agent e4 x)"

(* Explanation 2: Sitting on something provides warmth to that something. *)
axiomatization where
    explanation_2: "∀x y e1 e2. Something x ∧ Warmth y ∧ Sit e1 ∧ Agent e1 x ⟶ Provide e2 ∧ Agent e2 x ∧ Patient
e2 y"

(* Explanation 3: Providing warmth is synonymous with keeping warm. *)
axiomatization where
    explanation_3: "∀x y e. Warmth x ∧ Provide e ∧ Agent e x ⟶ KeepWarm e ∧ Agent e x"

theorem hypothesis:
(* Premise: Most birds sit on their eggs until they hatch. *)
assumes asm: "Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Hatch e2 ∧ Agent e2 y"
(* Hypothesis: The reason why birds sit on their eggs is to keep the eggs warm. *)
shows "∃x y e1 e2. Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ KeepWarm e2 ∧ Agent e2 x ∧ Patient e2 y"
proof -
  from asm have "Eggs y" by simp
  then obtain e1 e2 e3 e4 where e1e2e3e4: "(RequireWarmth e1 ∧ Patient e1 y) ∧ (Survive e2 ∧ Agent e2 y) ∧ (Hatch
e3 ∧ Agent e3 y) ∧ (Develop e4 ∧ Agent e4 y)" using explanation_1 by blast
  then obtain e2 where e2: "Provide e2 ∧ Agent e2 x ∧ Patient e2 y" using explanation_2 by blast
  then have "KeepWarm e2 ∧ Agent e2 x" using explanation_3 by blast
  show ?thesis using asm `KeepWarm e2 ∧ Agent e2 x` by blast
qed

end
```

Figure 48: The Isabelle theory code for table 9 iteration 0

```
theory question_92_1
imports Main
begin

typedecl entity
typedecl event

consts
  Eggs :: "entity ⇒ bool"
  Require :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Warmth :: "entity ⇒ bool"
  Survive :: "event ⇒ bool"
  Hatch :: "event ⇒ bool"
  Develop :: "event ⇒ bool"
  Something :: "entity ⇒ bool"
  Sit :: "event ⇒ bool"
  Provide :: "event ⇒ bool"
  KeepWarm :: "event ⇒ bool"
  Birds :: "entity ⇒ bool"

(* Explanation 1: Eggs require warmth to survive, hatch, and develop. *)
axiomatization where
    explanation_1: "∀x e1 e2 e3 e4 e5. Eggs x ∧ Require e1 ∧ Agent e1 x ∧ Warmth e5 ∧ Patient e1 e5 ⟶ Survive e2 ∧
Agent e2 x ∧ Hatch e3 ∧ Agent e3 x ∧ Develop e4 ∧ Agent e4 x"

(* Explanation 2: Sitting on something provides warmth to that something. *)
axiomatization where
    explanation_2: "∀x y e1 e2. Something x ∧ Warmth y ∧ Sit e1 ∧ Agent e1 x ⟶ Provide e2 ∧ Agent e2 x ∧ Patient
e2 y"

(* Explanation 3: Providing warmth is synonymous with keeping warm. *)
axiomatization where
    explanation_3: "∀x y e. Warmth x ∧ Provide e ∧ Agent e x ⟷ KeepWarm e ∧ Agent e x"

theorem hypothesis:
(* Premise: Most birds sit on their eggs until they hatch. *)
assumes asm: "Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Hatch e2 ∧ Agent e2 y"
(* Hypothesis: The reason why birds sit on their eggs is to keep the eggs warm. *)
shows "∃x y e1 e2. Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ KeepWarm e2 ∧ Agent e2 x ∧ Patient e2 y"
proof -
  from asm have "Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y" by simp
  then obtain e2 where e2: "Provide e2 ∧ Agent e2 x ∧ Patient e2 y" using explanation_2 by blast
  then have "KeepWarm e2 ∧ Agent e2 x ∧ Patient e2 y" using explanation_3 by blast
  show ?thesis using asm `KeepWarm e2 ∧ Agent e2 x ∧ Patient e2 y` by blast
qed

end
```

Figure 49: The Isabelle theory code for table 9 iteration 1

```
theory question_92_2

imports Main

begin

typedecl entity
typedecl event

consts
  Birds :: "entity ⇒ bool"
  Eggs :: "entity ⇒ bool"
  Sitting :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  ProvidesWarmth :: "event ⇒ bool"
  KeepWarm :: "event ⇒ bool"
  Hatch :: "event ⇒ bool"
  Sit :: "event ⇒ bool"

(* Explanation 1: Birds sitting on their eggs provides warmth to the eggs. *)
axiomatization where
    explanation_1: "∀x y e1 e2. Birds x ∧ Eggs y ∧ Sitting e1 ∧ Agent e1 x ∧ Patient e1 y ⟶ ProvidesWarmth e2 ∧
Agent e2 x ∧ Patient e2 y"

(* Explanation 2: Providing warmth is synonymous with keeping warm. *)
axiomatization where
    explanation_2: "∀e1 e2. ProvidesWarmth e1 ⟷ KeepWarm e2"

theorem hypothesis:
(* Premise: Most birds sit on their eggs until they hatch. *)
assumes asm: "Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Hatch e2 ∧ Patient e2 y"
(* Hypothesis: The reason why birds sit on their eggs is to keep the eggs warm. *)
shows "∃x y e1 e2. Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ KeepWarm e2 ∧ Agent e2 x ∧ Patient e2 y"
proof -
  from asm have "Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y" by simp
  then obtain e2 where e2: "ProvidesWarmth e2 ∧ Agent e2 x ∧ Patient e2 y" using explanation_1 by blast
  then have "KeepWarm e2" using explanation_2 by blast
  have conclusion: "Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ KeepWarm e2 ∧ Agent e2 x ∧ Patient e2
y" using asm e2 `KeepWarm e2` by simp
  show ?thesis using conclusion by blast
qed

end
```

Figure 50: The Isabelle theory code for table 9 iteration 2

```
theory question_92_3

imports Main

begin

typedecl entity
typedecl event

consts
  Birds :: "entity ⇒ bool"
  Eggs :: "entity ⇒ bool"
  Sit :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  ProvideWarmth :: "event ⇒ bool"
  KeepWarm :: "event ⇒ bool"
  Hatch :: "event ⇒ bool"

(* Explanation 1: Birds sit on their eggs to provide warmth to the eggs. *)
axiomatization where
    explanation_1: "∀x y e1 e2. Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ⟶ ProvideWarmth e2 ∧ Agent
e2 x ∧ Patient e2 y"

(* Explanation 2: Providing warmth is synonymous with keeping warm. *)
axiomatization where
    explanation_2: "∀e1 e2. ProvideWarmth e1 ⟷ KeepWarm e2"

theorem hypothesis:
(* Premise: Most birds sit on their eggs until they hatch. *)
assumes asm: "Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Hatch e2 ∧ Patient e2 y"
(* Hypothesis: The reason why birds sit on their eggs is to keep the eggs warm. *)
shows "∃x y e1 e2. Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ KeepWarm e2 ∧ Agent e2 x ∧ Patient e2 y"
proof -
  from asm have "Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y" by simp
  then obtain e2 where e2: "ProvideWarmth e2 ∧ Agent e2 x ∧ Patient e2 y" using explanation_1 by blast
  then have "KeepWarm e2" using explanation_2 by blast
  have conclusion: "Birds x ∧ Eggs y ∧ Sit e1 ∧ Agent e1 x ∧ Patient e1 y ∧ KeepWarm e2 ∧ Agent e2 x ∧ Patient e2
y" using asm e2 `KeepWarm e2` by simp
  show ?thesis using conclusion by blast
qed

end
```

Figure 51: The Isabelle theory code for table 9 iteration 3

Dataset	Sentences	Explanation	Iteration	Validity
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	reflecting light; sound is the opposite of absorbing light; sound. if an object; a substance absorbs solar energy then that object; that substance will increase in temperature. if an object; something is in the sunlight then that object; that something will absorb solar energy. shiny things; objects reflect light. keeping cool means not increasing temperature. a car is a kind of object.	0	Invalid
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	if an object; a substance absorbs solar energy then that object; that substance will increase in temperature. if an object; something is in the sunlight then that object; that something will absorb solar energy. shiny things; objects reflect light. keeping cool means not increasing temperature. a parked car is a kind of object.	1	Invalid
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	if an object; a substance absorbs solar energy then that object; that substance will increase in temperature. if an object; something is in the sunlight and it does not reflect light, then that object; that something will absorb solar energy. shiny things; objects reflect light. keeping cool means not increasing temperature. a parked car is a kind of object. A shiny aluminum screen is a kind of shiny object.	2	Invalid
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	if an object; a substance absorbs solar energy then that object; that substance will increase in temperature. if an object; something is in the sunlight and it reflects light, then that object; that something will not absorb solar energy. shiny things; objects reflect light. keeping cool means not increasing temperature. A shiny aluminum screen is a kind of shiny object.	3	Invalid
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	if an object; a substance absorbs solar energy then that object; that substance will increase in temperature. if an object; something is in the sunlight and it reflects light, then that object; that something will not absorb solar energy. A shiny aluminum screen reflects light. keeping cool means not increasing temperature. A shiny aluminum screen is a kind of shiny object.	4	Invalid
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	if an object; a substance absorbs solar energy then that object; that substance will increase in temperature. if an object; something reflects light, then that object; that something will not absorb solar energy. A shiny aluminum screen reflects light. keeping cool means not increasing temperature.	5	Invalid
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	if an object; a substance absorbs solar energy then that object; that substance will increase in temperature. if an object; something reflects light, then that object; that something will not absorb solar energy. A shiny aluminum screen reflects sunlight. keeping cool means not increasing temperature.	6	Invalid

Table 10: An example of an explanation in WorldTree that does not refine within 10 iterations using Explanation-Refiner

Dataset	Sentences	Explanation	Iteration	Validity
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	If a shiny aluminum screen absorbs solar energy then that shiny aluminum screen will increase in temperature. If a shiny aluminum screen reflects light, then that shiny aluminum screen will not absorb solar energy. A shiny aluminum screen reflects sunlight. Keeping cool means not increasing temperature.	7	Invalid
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	If a shiny aluminum screen absorbs solar energy then that shiny aluminum screen will increase in temperature. If a shiny aluminum screen reflects sunlight, then that shiny aluminum screen will not absorb solar energy. A shiny aluminum screen reflects sunlight. Keeping cool means not increasing temperature.	8	Invalid
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	If a shiny aluminum screen absorbs solar energy then that shiny aluminum screen will increase in temperature. A shiny aluminum screen that reflects sunlight does not absorb solar energy. A shiny aluminum screen reflects sunlight. Keeping cool means not increasing temperature.	9	Invalid
WorldTree	**Premise**: A shiny aluminum screen can be placed on the windshield of a parked car. **Hypothesis**: The screen helps to keep the car cool because it reflects the sunlight.	If a shiny aluminum screen absorbs solar energy then that shiny aluminum screen will increase in temperature. A shiny aluminum screen that reflects sunlight does not absorb any energy. A shiny aluminum screen reflects sunlight. Keeping cool means not increasing temperature.	10	Invalid

Table 11: An example of an explanation in WorldTree that does not refine within 10 iterations using Explanation-Refiner

```
theory question_45_0

imports Main

begin

typedecl entity
typedecl event

consts
  Light :: "entity ⇒ bool"
...
(* Explanation 1: reflecting light; sound is the opposite of absorbing light; sound. *)
axiomatization where
    explanation_1: "∀x y e1 e2. Light x ∧ Sound y ∧ Reflecting e1 ∧ Agent e1 x ⟶ ¬Absorbing e2 ∧ Agent e2 y"

(* Explanation 2: if an object; a substance absorbs solar energy then that object; that substance will increase in
temperature. *)
axiomatization where
    explanation_2: "∀x y e1 e2. Object x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶
IncreaseInTemperature e2 ∧ Agent e2 x"

(* Explanation 3: if an object; something is in the sunlight then that object; that something will absorb solar
energy. *)
axiomatization where
    explanation_3: "∀x y e1 e2. Object x ∧ Sunlight y ∧ IsIn e1 ∧ Agent e1 x ∧ Patient e1 y ⟶ Absorb e2 ∧ Agent e2
x ∧ Patient e2 y"

(* Explanation 4: shiny things; objects reflect light. *)
axiomatization where
    explanation_4: "∀x y e. ShinyThings x ∧ Light y ∧ Reflect e ∧ Agent e x ∧ Patient e y"

(* Explanation 5: keeping cool means not increasing temperature. *)
axiomatization where
    explanation_5: "∀x e1 e2. Object x ∧ KeepingCool e1 ∧ Agent e1 x ⟶ ¬IncreaseInTemperature e2 ∧ Agent e2 x"

(* Explanation 6: a car is a kind of object. *)
axiomatization where
    explanation_6: "∀x. Car x ⟶ Object x"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ CanBePlaced e ∧ Agent e x ∧ Patient e y ∧
Location e z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2 e3. Screen x ∧ Car y ∧ Sunlight z ∧ Reflects e1 ∧ Agent e1 x ∧ Patient e1 z ∧ Helps e2 ∧ Agent
e2 x ∧ Patient e2 y ∧ KeepCool e3 ∧ Agent e3 x ∧ Patient e3 y"
proof -
  from asm have "Object z" using explanation_6 by blast
  then have "Absorb e ∧ Agent e z ∧ Patient e y" using explanation_3 by blast
  then have "¬IncreaseInTemperature e ∧ Agent e z" using explanation_2 by blast
  then have "KeepingCool e ∧ Agent e z" using explanation_5 by blast
  have "Reflect e ∧ Agent e x ∧ Patient e y" using explanation_4 by blast
  show ?thesis using asm `Object z` `Absorb e ∧ Agent e z ∧ Patient e y` `¬IncreaseInTemperature e ∧ Agent e z`
`KeepingCool e ∧ Agent e z` `Reflect e ∧ Agent e x ∧ Patient e y` by blast
qed

end
```

Figure 52: The Isabelle theory code for table 10 iteration 0

```
theory question_45_1

imports Main

begin

typedecl entity
typedecl event

consts
  Object :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "event ⇒ bool"
  IncreaseInTemperature :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  IsIn :: "event ⇒ bool"
  Absorb :: "event ⇒ bool"
  ShinyThings :: "entity ⇒ bool"
  Light :: "entity ⇒ bool"
  Reflect :: "event ⇒ bool"
  KeepingCool :: "entity ⇒ bool"
  ParkedCar :: "entity ⇒ bool"
  Screen :: "entity ⇒ bool"
...
(* Explanation 1: if an object; a substance absorbs solar energy then that object; that substance will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. Object x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶
IncreaseInTemperature e2 ∧ Agent e2 x"

(* Explanation 2: if an object; something is in the sunlight then that object; that something will absorb solar
energy. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. Object x ∧ Sunlight y ∧ IsIn e1 ∧ Agent e1 x ∧ Patient e1 y ⟶ Absorb e2 ∧ Agent
e2 x ∧ Patient e2 y"

(* Explanation 3: shiny things; objects reflect light. *)
axiomatization where
    explanation_3: "∀x y e. ShinyThings x ∧ Light y ∧ Reflect e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: keeping cool means not increasing temperature. *)
axiomatization where
    explanation_4: "∀x e. KeepingCool x ⟶ ¬IncreaseInTemperature e ∧ Agent e x"

(* Explanation 5: a parked car is a kind of object. *)
axiomatization where
    explanation_5: "∀x. ParkedCar x ⟶ Object x"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ CanBePlaced e ∧ Agent e x ∧ Patient e y ∧
Location e z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2 e3. Screen x ∧ Car y ∧ Sunlight z ∧ Reflects e1 ∧ Agent e1 x ∧ Patient e1 z ∧ Helps e2 ∧ Agent
e2 x ∧ Patient e2 y ∧ KeepCool e3 ∧ Agent e3 x ∧ Patient e3 y"
proof -
  from asm have "Object z" using explanation_5 by blast
  then have "Absorb e ∧ Agent e z ∧ Patient e y" using explanation_2 by blast
  then have "¬IncreaseInTemperature e ∧ Agent e z" using explanation_1 explanation_4 by blast
  then have "Reflect e ∧ Agent e x ∧ Patient e y" using explanation_3 by blast
  then have "Helps e2 ∧ Agent e2 x ∧ Patient e2 y" by simp
  then have "KeepCool e3 ∧ Agent e3 x ∧ Patient e3 y" using explanation_4 by blast
  show ?thesis using asm `Object z` `Absorb e ∧ Agent e z ∧ Patient e y` `¬IncreaseInTemperature e ∧ Agent e z`
`Reflect e ∧ Agent e x ∧ Patient e y` `Helps e2 ∧ Agent e2 x ∧ Patient e2 y` `KeepCool e3 ∧ Agent e3 x ∧ Patient e3
y` by blast
qed

end
```

Figure 53: The Isabelle theory code for table 10 iteration 1

```
theory question_45_2

imports Main

begin

typedecl entity
typedecl event

consts
  Object :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  IncreaseInTemperature :: "event ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  Light :: "entity ⇒ bool"
  Reflect :: "event ⇒ bool"
  In :: "entity ⇒ entity ⇒ bool"
  AbsorbSolarEnergy :: "event ⇒ bool"
  ShinyThings :: "entity ⇒ bool"
  KeepingCool :: "entity ⇒ bool"
  ParkedCar :: "entity ⇒ bool"
...

(* Explanation 1: if an object; a substance absorbs solar energy then that object; that substance will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. Object x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶
IncreaseInTemperature e2 ∧ Agent e2 x"

(* Explanation 2: if an object; something is in the sunlight and it does not reflect light, then that object; that
something will absorb solar energy. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. Object x ∧ Sunlight y ∧ Light z ∧ ¬Reflect e1 ∧ Agent e1 x ∧ Patient e1 z ∧ In x
y ⟶ AbsorbSolarEnergy e2 ∧ Agent e2 x"

(* Explanation 3: shiny things; objects reflect light. *)
axiomatization where
    explanation_3: "∀x y e. ShinyThings x ∧ Light y ∧ Reflect e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: keeping cool means not increasing temperature. *)
axiomatization where
explanation_4: "∀x e. KeepingCool x ⟶ ¬IncreaseInTemperature e ∧ Agent e x"

(* Explanation 5: a parked car is a kind of object. *)
axiomatization where
    explanation_5: "∀x. ParkedCar x ⟶ Object x"

(* Explanation 6: A shiny aluminum screen is a kind of shiny object. *)
axiomatization where
    explanation_6: "∀x. ShinyAluminumScreen x ⟶ ShinyObject x"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar y ∧ Placed e ∧ Agent e x ∧ Patient e y"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2 e3. Screen x ∧ Car y ∧ Sunlight z ∧ Helps e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Keep e2 ∧ Agent e2 x
∧ Patient e2 y ∧ Reflects e3 ∧ Agent e3 x ∧ Patient e3 z "
proof -
  from asm have "ShinyObject x" using explanation_6 by simp
  then obtain e1 where e1: "Reflect e1 ∧ Agent e1 x ∧ Patient e1 y" using explanation_3 by blast
  then have "¬AbsorbSolarEnergy e2 ∧ Agent e2 x" using explanation_2 by blast
  then have "¬IncreaseInTemperature e3 ∧ Agent e3 x" using explanation_1 by blast
  then have "KeepingCool x" using explanation_4 by blast
  have conclusion: "Reflect e1 ∧ Agent e1 x ∧ Patient e1 y" using e1 by simp
  show ?thesis using asm conclusion `KeepingCool x` by blast
qed

end
```

Figure 54: The Isabelle theory code for table 10 iteration 2

```
theory question_45_3

imports Main

begin

typedecl entity
typedecl event

consts
  Object :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Increase :: "event ⇒ bool"
  Temperature :: "event ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  Light :: "entity ⇒ bool"
  Is :: "event ⇒ entity ⇒ entity ⇒ bool"
  Reflects :: "event ⇒ bool"
  Absorb :: "event ⇒ entity ⇒ entity ⇒ bool"
  ShinyThings :: "entity ⇒ bool"
  Reflect :: "event ⇒ bool"
...

(* Explanation 1: if an object; a substance absorbs solar energy then that object; that substance will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. Object x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶ Increase e2 ∧
Agent e2 x ∧ Temperature e2"

(* Explanation 2: if an object; something is in the sunlight and it reflects light, then that object; that
something will not absorb solar energy. *)
axiomatization where
    explanation_2: "∀x y z e1 e2 e3. Object x ∧ Sunlight y ∧ Light z ∧ Is e1 x y ∧ Reflects e2 ∧ Agent e2 x ∧
Patient e2 z ⟶ ¬Absorb e3 x y"

(* Explanation 3: shiny things; objects reflect light. *)
axiomatization where
    explanation_3: "∀x y e. ShinyThings x ∧ Light y ⟶ Reflect e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: keeping cool means not increasing temperature. *)
axiomatization where
    explanation_4: "∀x e. KeepingCool x ⟶ ¬Increase e ∧ Temperature e"

(* Explanation 5: A shiny aluminum screen is a kind of shiny object. *)
axiomatization where
    explanation_5: "∀x. ShinyAluminumScreen x ⟶ ShinyObject x"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ Placed e ∧ Agent e x ∧ Patient e y ∧ Location e
z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2 e3. Screen x ∧ Car y ∧ Sunlight z ∧ Helps e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Keep e2 ∧ Agent e2 x
∧ Patient e2 y ∧ Reflects e3 ∧ Agent e3 x ∧ Patient e3 z "
proof -
  from asm have "ShinyObject x" using explanation_5 by simp
  then obtain e1 where e1: "Reflect e1 ∧ Agent e1 x ∧ Patient e1 y" using explanation_3 by blast
  then have "¬Absorb e3 x y" using explanation_2 by blast
  then have "¬Increase e2 ∧ Temperature e2" using explanation_1 by blast
  then have "KeepingCool x" using explanation_4 by blast
  have conclusion: "Reflect e1 ∧ Agent e1 x ∧ Patient e1 y" using e1 by simp
  show ?thesis using asm conclusion `KeepingCool x` by blast
qed

end
```

Figure 55: The Isabelle theory code for table 10 iteration 3

```
theory question_45_4

imports Main

begin

typedecl entity
typedecl event

consts
  Object :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Increase :: "event ⇒ bool"
  Temperature :: "entity ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  Light :: "entity ⇒ bool"
  Is :: "event ⇒ entity ⇒ entity ⇒ bool"
  Reflects :: "event ⇒ bool"
  ShinyAluminumScreen :: "entity ⇒ bool"
  Cool :: "entity ⇒ bool"
  ShinyObject :: "entity ⇒ bool"
  Screen :: "entity ⇒ bool"
  Car :: "entity ⇒ bool"
...

(* Explanation 1: if an object; a substance absorbs solar energy then that object; that substance will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. Object x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶ Increase e2 ∧
Agent e2 x ∧ Temperature x"

(* Explanation 2: if an object; something is in the sunlight and it reflects light, then that object; that
something will not absorb solar energy. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. Object x ∧ Sunlight y ∧ Light z ∧ Is e1 x y ∧ Reflects e1 ∧ Agent e1 x ∧ Patient
e1 z ⟶ ¬Absorbs e2 ∧ Agent e2 x ∧ SolarEnergy y"

(* Explanation 3: A shiny aluminum screen reflects light. *)
axiomatization where
    explanation_3: "∀x y e. ShinyAluminumScreen x ∧ Light y ∧ Reflects e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: keeping cool means not increasing temperature. *)
axiomatization where
    explanation_4: "∀x e. Cool x ⟶ ¬Increase e ∧ Temperature x"

(* Explanation 5: A shiny aluminum screen is a kind of shiny object. *)
axiomatization where
    explanation_5: "∀x. ShinyAluminumScreen x ⟶ ShinyObject x"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ Placed e ∧ Agent e x ∧ Patient e y ∧ Location e
z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2. Screen x ∧ Car y ∧ Sunlight z ∧ Reflects e1 ∧ Agent e1 x ∧ Patient e1 z ∧ Helps e2 ∧ Agent e2
x ∧ Patient e2 y ∧ Cool y"
proof -
  from asm have "ShinyAluminumScreen x" by simp
  then have "Reflects e ∧ Agent e x ∧ Patient e y" using explanation_3 by blast
  then have "¬Absorbs e ∧ Agent e x ∧ SolarEnergy y" using explanation_2 by blast
  then have "¬Increase e ∧ Temperature x" using explanation_1 by blast
  then have "Cool x" using explanation_4 by blast
  show ?thesis using asm `Cool x` `Reflects e ∧ Agent e x ∧ Patient e y` by blast
qed

end
```

Figure 56: The Isabelle theory code for table 10 iteration 4

```
theory question_45_5

imports Main

begin

typedecl entity
typedecl event

consts
  Object :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  IncreaseInTemperature :: "event ⇒ bool"
  Light :: "entity ⇒ bool"
  Reflects :: "event ⇒ bool"
  ShinyAluminumScreen :: "entity ⇒ bool"
  Cool :: "entity ⇒ bool"
  Screen :: "entity ⇒ bool"
  Car :: "entity ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  Helps :: "event ⇒ bool"
  Windshield :: "entity ⇒ bool"
  ParkedCar :: "entity ⇒ bool"
  Placed :: "event ⇒ bool"
  Location :: "event ⇒ entity ⇒ bool"

(* Explanation 1: if an object; a substance absorbs solar energy then that object; that substance will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. Object x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶
IncreaseInTemperature e2 ∧ Agent e2 x"

(* Explanation 2: if an object; something reflects light, then that object; that something will not absorb solar
energy. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. Object x ∧ Light y ∧ SolarEnergy z ∧ Reflects e1 ∧ Agent e1 x ∧ Patient e1 y ⟶
¬Absorbs e2 ∧ Agent e2 x ∧ Patient e2 z"

(* Explanation 3: A shiny aluminum screen reflects light. *)
axiomatization where
    explanation_3: "∃x y e. ShinyAluminumScreen x ∧ Light y ∧ Reflects e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: keeping cool means not increasing temperature. *)
axiomatization where
    explanation_4: "∀x e. Cool x ⟷ ¬IncreaseInTemperature e ∧ Agent e x"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ Placed e ∧ Agent e x ∧ Patient e y ∧ Location e
z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2. Screen x ∧ Car y ∧ Sunlight z ∧ Helps e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Cool y ∧ Reflects e2 ∧
Agent e2 x ∧ Patient e2 z "
proof -
  from asm have "ShinyAluminumScreen x" by simp
  then obtain y e where e: "Light y ∧ Reflects e ∧ Agent e x ∧ Patient e y" using explanation_3 by blast
  then have "¬Absorbs e ∧ Agent e x ∧ Patient e y" using explanation_2 by blast
  then have "¬IncreaseInTemperature e ∧ Agent e x" using explanation_1 by blast
  then have "Cool x" using explanation_4 by blast
  have conclusion: "Reflects e ∧ Agent e x ∧ Patient e y" using e by simp
  show ?thesis using asm conclusion `Cool x` by blast
qed

end
```

Figure 57: The Isabelle theory code for table 10 iteration 5

```
theory question_45_6

imports Main

begin

typedecl entity
typedecl event

consts
  Object :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  Increase :: "event ⇒ bool"
  Temperature :: "entity ⇒ bool"
  Light :: "entity ⇒ bool"
  Reflects :: "event ⇒ bool"
  ShinyAluminumScreen :: "entity ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  Cool :: "entity ⇒ bool"
  Screen :: "entity ⇒ bool"
  Car :: "entity ⇒ bool"
  Helps :: "event ⇒ bool"
  Keep :: "event ⇒ bool"
  Windshield :: "entity ⇒ bool"
  ParkedCar :: "entity ⇒ bool"
  Placed :: "event ⇒ bool"
  Location :: "event ⇒ entity ⇒ bool"

(* Explanation 1: if an object; a substance absorbs solar energy then that object; that substance will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. Object x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶ Increase e2 ∧
Agent e2 x ∧ Temperature x"

(* Explanation 2: if an object; something reflects light, then that object; that something will not absorb solar
energy. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. Object x ∧ Light y ∧ SolarEnergy z ∧ Reflects e1 ∧ Agent e1 x ∧ Patient e1 y ⟶
¬Absorbs e2 ∧ Agent e2 x ∧ Patient e2 z"

(* Explanation 3: A shiny aluminum screen reflects sunlight. *)
axiomatization where
    explanation_3: "∃x y e. ShinyAluminumScreen x ∧ Sunlight y ∧ Reflects e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: keeping cool means not increasing temperature. *)
axiomatization where
    explanation_4: "∀x e. Cool x ⟷ ¬Increase e ∧ Temperature x"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ Placed e ∧ Agent e x ∧ Patient e y ∧ Location e
z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2 e3. Screen x ∧ Car y ∧ Sunlight z ∧ Reflects e1 ∧ Agent e1 x ∧ Patient e1 z ∧ Helps e2 ∧ Agent
e2 x ∧ Patient e2 y ∧ Keep e3 ∧ Agent e3 x ∧ Patient e3 y ∧ Cool y"
proof -
  from asm have "ShinyAluminumScreen x" by simp
  then obtain e1 where e1: "Reflects e1 ∧ Agent e1 x ∧ Patient e1 y" using explanation_3 by blast
  then have "¬Absorbs e2 ∧ Agent e2 x ∧ Patient e2 z" using explanation_2 by blast
  then have "¬Increase e2 ∧ Temperature x" using explanation_1 by blast
  then have "Cool x" using explanation_4 by blast
  have conclusion: "Reflects e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Helps e2 ∧ Agent e2 x ∧ Patient e2 y ∧ Keep e3 ∧
Agent e3 x ∧ Patient e3 y ∧ Cool y" using e1 `Cool x` by simp
  show ?thesis using asm conclusion by blast
qed

end
```

Figure 58: The Isabelle theory code for table 10 iteration 6

```
theory question_45_7

imports Main

begin

typedecl entity
typedecl event

consts
  ShinyAluminumScreen :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  IncreaseInTemperature :: "event ⇒ bool"
  Light :: "entity ⇒ bool"
  Reflects :: "event ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  Cool :: "entity ⇒ bool"
  Means :: "event ⇒ bool"
  IncreasingTemperature :: "event ⇒ bool"
  Screen :: "entity ⇒ bool"
  Car :: "entity ⇒ bool"
  Helps :: "event ⇒ bool"
  Keep :: "event ⇒ bool"
  Windshield :: "entity ⇒ bool"
  ParkedCar :: "entity ⇒ bool"
  Placed :: "event ⇒ bool"
  Location :: "event ⇒ entity ⇒ bool"

(* Explanation 1: If a shiny aluminum screen absorbs solar energy then that shiny aluminum screen will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. ShinyAluminumScreen x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶
IncreaseInTemperature e2 ∧ Agent e2 x"

(* Explanation 2: If a shiny aluminum screen reflects light, then that shiny aluminum screen will not absorb solar
energy. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. ShinyAluminumScreen x ∧ Light y ∧ SolarEnergy z ∧ Reflects e1 ∧ Agent e1 x ∧
Patient e1 y ⟶ ¬(Absorbs e2 ∧ Agent e2 x ∧ Patient e2 z)"

(* Explanation 3: A shiny aluminum screen reflects sunlight. *)
axiomatization where
    explanation_3: "∀x y e. ShinyAluminumScreen x ∧ Sunlight y ∧ Reflects e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: Keeping cool means not increasing temperature. *)
axiomatization where
    explanation_4: "∀x e1 e2. Cool x ⟶ (Means e1 ∧ Agent e1 x ∧ ¬IncreasingTemperature e2)"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ Placed e ∧ Agent e x ∧ Patient e y ∧ Location e
z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2 e3. Screen x ∧ Car y ∧ Sunlight z ∧ Helps e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Keep e2 ∧ Agent e2 x
∧ Patient e2 y ∧ Reflects e3 ∧ Agent e3 x ∧ Patient e3 z "
proof -
  from asm have "ShinyAluminumScreen x" by simp
  then obtain e1 where e1: "Reflects e1 ∧ Agent e1 x ∧ Patient e1 y" using explanation_3 by blast
  then have "¬(Absorbs e2 ∧ Agent e2 x ∧ Patient e2 z)" using explanation_2 by blast
  then have "¬IncreaseInTemperature e2 ∧ Agent e2 x" using explanation_1 by blast
  then have "Cool x" using explanation_4 by blast
  have conclusion: "Helps e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Keep e2 ∧ Agent e2 x ∧ Patient e2 y ∧ Reflects e3 ∧
Agent e3 x ∧ Patient e3 z" using e1 by simp
  show ?thesis using asm conclusion `Cool x` by blast
qed

end
```

Figure 59: The Isabelle theory code for table 11 iteration 7

```
theory question_45_8

imports Main

begin

typedecl entity
typedecl event

consts
  ShinyAluminumScreen :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  IncreaseInTemperature :: "event ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  Reflects :: "event ⇒ bool"
  Cool :: "entity ⇒ bool"
  Means :: "event ⇒ bool"
  Screen :: "entity ⇒ bool"
  Car :: "entity ⇒ bool"
  Helps :: "event ⇒ bool"
  Keep :: "event ⇒ bool"
  Windshield :: "entity ⇒ bool"
  ParkedCar :: "entity ⇒ bool"
  Placed :: "event ⇒ bool"
  Location :: "event ⇒ entity ⇒ bool"

(* Explanation 1: If a shiny aluminum screen absorbs solar energy then that shiny aluminum screen will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. ShinyAluminumScreen x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶
IncreaseInTemperature e2 ∧ Agent e2 x"

(* Explanation 2: If a shiny aluminum screen reflects sunlight, then that shiny aluminum screen will not absorb
solar energy. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. ShinyAluminumScreen x ∧ Sunlight y ∧ SolarEnergy z ∧ Reflects e1 ∧ Agent e1 x ∧
Patient e1 y ⟶ ¬(Absorbs e2 ∧ Agent e2 x ∧ Patient e2 z)"

(* Explanation 3: A shiny aluminum screen reflects sunlight. *)
axiomatization where
    explanation_3: "∀x y e. ShinyAluminumScreen x ∧ Sunlight y ∧ Reflects e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: Keeping cool means not increasing temperature. *)
axiomatization where
    explanation_4: "∀x e1 e2. Cool x ∧ Means e1 ∧ Agent e1 x ⟶ ¬IncreaseInTemperature e2"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ Placed e ∧ Agent e x ∧ Patient e y ∧ Location e
z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2 e3. Screen x ∧ Car y ∧ Sunlight z ∧ Helps e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Keep e2 ∧ Agent e2 x
∧ Patient e2 y ∧ Reflects e3 ∧ Agent e3 x ∧ Patient e3 z "
proof -
  from asm have "ShinyAluminumScreen x" by simp
  then have "∃e. Reflects e ∧ Agent e x ∧ Patient e y" using explanation_3 by blast
  then have "¬(Absorbs e ∧ Agent e x ∧ Patient e y)" using explanation_2 by blast
  then have "¬IncreaseInTemperature e" using explanation_1 by blast
  then have "Cool x" using explanation_4 by blast
  then have "Helps e ∧ Agent e x ∧ Patient e y" by simp
  then have "Keep e ∧ Agent e x ∧ Patient e y" by simp
  show ?thesis using asm `Helps e ∧ Agent e x ∧ Patient e y` `Keep e ∧ Agent e x ∧ Patient e y` `Reflects e ∧ Agent
e x ∧ Patient e y` by blast
qed

end
```

Figure 60: The Isabelle theory code for table 11 iteration 8

```
theory question_45_9

imports Main

begin

typedecl entity
typedecl event

consts
  ShinyAluminumScreen :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  IncreaseInTemperature :: "event ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  Reflects :: "event ⇒ bool"
  Cool :: "entity ⇒ bool"
  Means :: "event ⇒ bool"
  Screen :: "entity ⇒ bool"
  Car :: "entity ⇒ bool"
  Helps :: "event ⇒ bool"
  Keep :: "event ⇒ bool"
  Windshield :: "entity ⇒ bool"
  ParkedCar :: "entity ⇒ bool"
  Placed :: "event ⇒ bool"
  Location :: "event ⇒ entity ⇒ bool"

(* Explanation 1: If a shiny aluminum screen absorbs solar energy then that shiny aluminum screen will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. ShinyAluminumScreen x ∧ SolarEnergy y ∧ Absorbs e1 ∧ Agent e1 x ∧ Patient e1 y ⟶
IncreaseInTemperature e2 ∧ Agent e2 x"

(* Explanation 2: A shiny aluminum screen that reflects sunlight does not absorb solar energy. *)
axiomatization where
    explanation_2: "∀x y z e1. ShinyAluminumScreen x ∧ Sunlight y ∧ SolarEnergy z ∧ Reflects e1 ∧ Agent e1 x ∧
Patient e1 y ⟶ ¬(∃e2. Absorbs e2 ∧ Agent e2 x ∧ Patient e2 z)"

(* Explanation 3: A shiny aluminum screen reflects sunlight. *)
axiomatization where
    explanation_3: "∀x y e. ShinyAluminumScreen x ∧ Sunlight y ∧ Reflects e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: Keeping cool means not increasing temperature. *)
axiomatization where
    explanation_4: "∀x e. Cool x ∧ Means e ∧ Agent e x ⟶ ¬(∃e2. IncreaseInTemperature e2 ∧ Agent e2 x)"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ Placed e ∧ Agent e x ∧ Patient e y ∧ Location e
z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2 e3. Screen x ∧ Car y ∧ Sunlight z ∧ Helps e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Keep e2 ∧ Agent e2 x
∧ Patient e2 y ∧ Reflects e3 ∧ Agent e3 x ∧ Patient e3 z "
proof -
  from asm have "ShinyAluminumScreen x" by simp
  then have "∃e. Reflects e ∧ Agent e x ∧ Patient e y" using explanation_3 by blast
  then have "¬(∃e2. Absorbs e2 ∧ Agent e2 x ∧ Patient e2 y)" using explanation_2 by blast
  then have "¬(∃e2. IncreaseInTemperature e2 ∧ Agent e2 x)" using explanation_1 by blast
  then have "Cool x" using explanation_4 by blast
  then have "Helps e ∧ Agent e x ∧ Patient e y" by simp
  then have "Keep e ∧ Agent e x ∧ Patient e y" by simp
  show ?thesis using asm `Helps e ∧ Agent e x ∧ Patient e y` `Keep e ∧ Agent e x ∧ Patient e y` `∃e. Reflects e ∧
Agent e x ∧ Patient e y` by blast
qed

end
```

Figure 61: The Isabelle theory code for table 11 iteration 9

```
theory question_45_10
imports Main
begin

typedecl entity
typedecl event

consts
  ShinyAluminumScreen :: "entity ⇒ bool"
  SolarEnergy :: "entity ⇒ bool"
  Absorbs :: "entity ⇒ entity ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  Patient :: "event ⇒ entity ⇒ bool"
  IncreaseInTemperature :: "event ⇒ bool"
  Sunlight :: "entity ⇒ bool"
  Energy :: "entity ⇒ bool"
  Reflects :: "event ⇒ bool"
  Cool :: "entity ⇒ bool"
  Means :: "event ⇒ bool"
  Screen :: "entity ⇒ bool"
  Car :: "entity ⇒ bool"
  Helps :: "event ⇒ bool"
  Keep :: "event ⇒ bool"
  Windshield :: "entity ⇒ bool"
  ParkedCar :: "entity ⇒ bool"
  Placed :: "event ⇒ bool"
  Location :: "event ⇒ entity ⇒ bool"

(* Explanation 1: If a shiny aluminum screen absorbs solar energy then that shiny aluminum screen will increase in
temperature. *)
axiomatization where
    explanation_1: "∀x y e1 e2. ShinyAluminumScreen x ∧ SolarEnergy y ∧ Absorbs x y ∧ Agent e1 x ∧ Patient e1 y ⟶
IncreaseInTemperature e2 ∧ Agent e2 x"

(* Explanation 2: A shiny aluminum screen that reflects sunlight does not absorb any energy. *)
axiomatization where
    explanation_2: "∀x y z e1. ShinyAluminumScreen x ∧ Sunlight y ∧ Energy z ∧ Reflects e1 ∧ Agent e1 x ∧ Patient
e1 y ⟶ ¬Absorbs x z"

(* Explanation 3: A shiny aluminum screen reflects sunlight. *)
axiomatization where
    explanation_3: "∀x y e. ShinyAluminumScreen x ∧ Sunlight y ∧ Reflects e ∧ Agent e x ∧ Patient e y"

(* Explanation 4: Keeping cool means not increasing temperature. *)
axiomatization where
    explanation_4: "∀x e1 e2. Cool x ∧ Means e1 ∧ Agent e1 x ⟶ ¬IncreaseInTemperature e2"

theorem hypothesis:
(* Premise: a shiny aluminum screen can be placed on the windshield of a parked car. *)
assumes asm: "ShinyAluminumScreen x ∧ Windshield y ∧ ParkedCar z ∧ Placed e ∧ Agent e x ∧ Patient e y ∧ Location e
z"
(* Hypothesis: The screen helps to keep the car cool because it reflects the sunlight. *)
shows "∃x y z e1 e2 e3. Screen x ∧ Car y ∧ Sunlight z ∧ Helps e1 ∧ Agent e1 x ∧ Patient e1 y ∧ Keep e2 ∧ Agent e2 x
∧ Patient e2 y ∧ Reflects e3 ∧ Agent e3 x ∧ Patient e3 z "
proof -
  from asm have "ShinyAluminumScreen x" by simp
  then obtain e1 where e1: "Reflects e1 ∧ Agent e1 x ∧ Patient e1 y" using explanation_3 by blast
  then have "¬Absorbs x z" using explanation_2 by blast
  then have "¬IncreaseInTemperature e2" using explanation_4 by blast
  have conclusion: "Reflects e1 ∧ Agent e1 x ∧ Patient e1 y" using e1 by simp
  show ?thesis using asm conclusion `¬IncreaseInTemperature e2` by blast
qed

end
```

Figure 62: The Isabelle theory code for table 11 iteration 10