ProofTeller: Exposing recency bias in LLM reasoning and its side effects on communication

Mayank Jobanputra[1] **Alisa Kovtunova**[3] **Brisca Balthes**[1,2]
Fedor Grigoryevich Pogulskiy[3] **Yifan Wang**[1] **Stefan Borgwardt**[3] **Vera Demberg**[1]

[1]Saarland University, Saarbrücken, Germany [2]Zuse School ELIZA, Germany
[3]Institute of Theoretical Computer Science, TU Dresden, Germany

{mayank, briscab, yifwang, vera}@lst.uni-saarland.de
{alisa.kovtunova, fedor_grigoryevich.pogulskiy, stefan.borgwardt}@tu-dresden.de

Abstract

Large language models (LLMs) are increasingly applied in domains that demand reliable and interpretable reasoning. While formal methods can generate provably correct proofs, these proofs are often inaccessible to non-expert users. This raises a natural question: can LLMs, when given a verified proof, faithfully interpret its reasoning and communicate it clearly? We introduce `ProofTeller`, a benchmark that evaluates this ability across three tasks: (1) identifying key proof steps, (2) summarizing the reasoning, and (3) explaining the result in concise natural language. The benchmark covers three domains: *Biology*, *Drones*, and *Recipes*, representing scientific, safety-critical, and everyday reasoning scenarios. We find a consistent near-conclusion bias: LLMs tend to focus on steps closest to the final proof conclusion rather than on the most informative ones. A targeted human study confirms that explanations based on such steps are rated less appropriate for end users. These findings indicate that even when reasoning is provided, current LLMs face challenges in communicating key information in a useful manner, highlighting the need for LLMs that can communicate important details reliably.

1 Introduction

Large language models (LLMs) are increasingly being considered for use in domains where decisions have real consequences, from medical applications (Bang et al., 2025) and autonomous vehicles (Cui et al., 2024) to network infrastructure (Manocchio et al., 2024). In such settings, reliability and consistency are essential. Although in-context learning (Brown et al., 2020) enables LLMs to adapt flexibly to new tasks, and methods such as retrieval-augmented generation (RAG) (Lewis et al., 2020) have extended their scope, their outputs remain unpredictable (Abbasi Yadkori et al., 2024). Small changes in phrasing or ordering can lead to inconsistent answers (Sclar et al., 2024; Elazar et al., 2021), and explanations often appear plausible yet fail to reflect the reasoning behind the prediction (Turpin et al., 2023).

Recent "thinking" LLMs, including DeepSeek-R1 (Guo et al., 2025), Qwen3 (Yang et al., 2025a), and Gemini-2.5 (Comanici et al., 2025), have demonstrated more structured reasoning traces, but they still do not implement explicit algorithms and often produce ill-founded or inconsistent justifications (Shojaee et al., 2025). Such behaviour limits their applicability in domains where reasoning steps must be verifiable and interpretable.

A natural way to enforce reliability is to draw on *formal methods*, which provide symbolic proofs with guaranteed correctness. If an LLM is supplied with such a proof as input, the reasoning path is already complete and verifiable; the LLM's task is to interpret the proof, identify its essential steps, and communicate them clearly. This setting isolates the linguistic and communicative aspects of reasoning from the purely deductive ones, allowing us to study whether LLMs can serve as effective interfaces between formal reasoning systems and human users (Kassner et al., 2021). However, as Turpin et al. (2023) observes, faithfulness in current LLMs is easily influenced by superficial cues in the input, and the proofs themselves do not always encode every relevant detail of a scenario (Mondorf and Plank, 2024). When LLMs attempt to fill these gaps with background knowledge (Sun et al., 2025; Xie et al., 2024), they may produce inconsistent or misleading explanations.

These observations lead to a concrete question:

> *Can LLMs reliably interpret a formal proof, identify the key reasoning steps, and restate them faithfully and intelligibly for a non-expert?*

In principle, this capability should be achievable. Proofs contain explicit reasoning steps, leaving lit-

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1439–1462

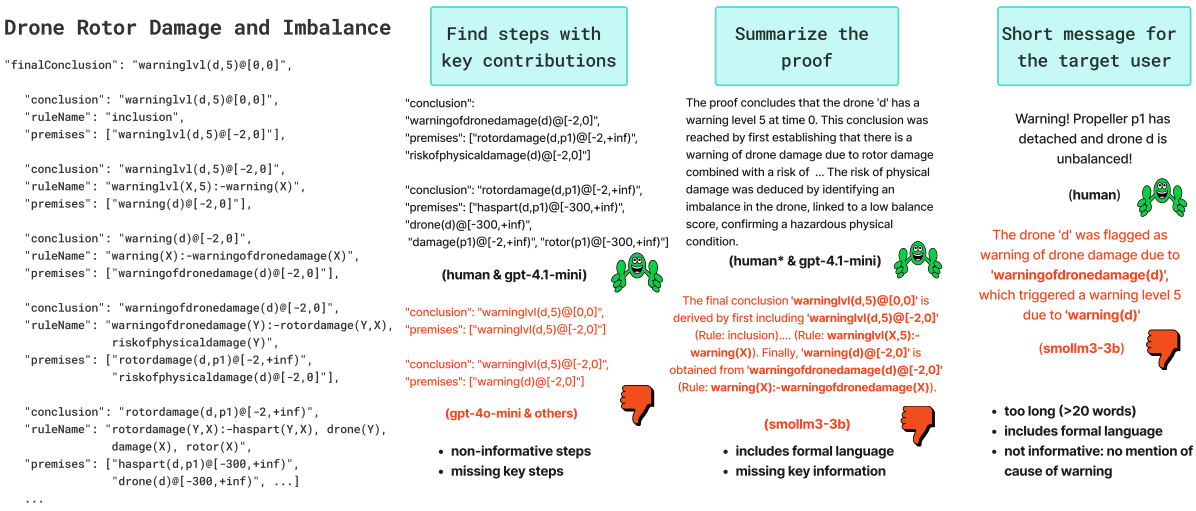

Figure 1: An example datapoint showcasing limitations of LLMs on our benchmark tasks.

tle ambiguity about logical structure, while LLMs are designed to handle natural language fluently. Combining the two should play to each component's strengths: the proof provides correctness, and the LLM focuses on communication. An LLM does not need to invent new reasoning but only to verbalize what is already entailed, using domain knowledge when necessary to make the explanation meaningful. Whether current LLMs can meet this expectation is an open empirical question.

To investigate this question, we introduce `ProofTeller`, a benchmark that evaluates how well LLMs interpret and communicate verified proofs. We use symbolic proofs generated by an automated reasoner and ask LLMs to:

1. **Highlight** the key steps leading to the proof's conclusion,
2. **Summarize** the overall reasoning faithfully
3. **Explain** the result in concise natural language to a specific user group.

We instantiate these tasks in three domains: *Biology*, *Drones*, and *Recipes*, which together span scientific, safety-critical, and everyday reasoning scenarios. This diversity enables us to analyse both the accuracy of the interpretations and the suitability of the resulting explanations.

Our evaluation of nine LLMs (seven open-weight and two proprietary) shows a consistent pattern: LLMs tend to focus on steps located near the final conclusion of the proof, whereas human annotators select informative steps distributed across the reasoning chain. This "near-conclusion bias" leads to explanations that are formally correct but less appropriate for end users. A targeted human study confirms this effect, showing that messages based on such steps are rated as less helpful, particularly in the *Drone* domain. These results suggest that even when the reasoning is provided, current LLMs face challenges in communicating it faithfully and pragmatically.

`ProofTeller` thus offers a controlled setting for examining these limitations and for guiding the development of LLMs that can communicate formal reasoning in a reliable and user-friendly manner. We also release our benchmark dataset along with the inferences from all models[1].

2 Related Work

Symbolic text understanding Many recent works evaluate the symbolic reasoning abilities of LLMs. For example, Shalyt et al. (2025) tests LLMs' abilities for mathematical problem-solving using symbolic perturbations, Kulkarni et al. (2025) makes use of SQL for robust tabular reasoning, and Vo et al. (2025) evaluates LLMs for an abstract causal discovery task from natural language text. Our benchmark focuses on evaluating LLMs' ability to *explain* formal proofs.

Data-To-Text generation Our work also shares similarity with data-to-text (D2T) generation since we also generate natural language text from structured data (i.e., JSON proofs). In the D2T literature, the ToTTo (Parikh et al., 2020) benchmark assesses

[1] https://github.com/mayankjobanputra/ProofTeller

the faithful generation of text from tables. Another such benchmark is DART (Nan et al., 2021), which focuses on generating text from complex RDF triples. Our work introduces a different structured input (i.e., logic proofs), which requires models to understand logical operators and inferences to create a summary and a user-targeted message.

Explaining logical proofs Colombo et al. (2025) generate explanations for Datalog proofs in finance using a template pipeline: a fixed set of templates determines which proof steps appear, and an LLM fills the templates. In contrast, we do not preselect steps. `ProofTeller` evaluates whether an LLM can identify key steps from the full proof, summarize the reasoning, and generate a concise message for the target user, across multiple domains and description logic formalisms.

3 Benchmark Creation

We start by generating different types of logic proofs. Our proof data come from three distinct domains — *Biology*, *Recipe*, and *Drone* — the latter is a concise label for "Critical Situations in Drones". We selected these domains for two primary reasons. First, the domains are accessible for annotation, being sufficiently close to common knowledge for people to engage with, yet complex enough to be intellectually meaningful. Second, each domain has a recognized real-world ontology, ensuring the logical proofs and their annotations are relevant to their respective fields. For this benchmark, we explore real-world scenarios and user groups that are different for each domain. For each target domain, we randomly sample 50 proofs for annotation. In this section, we describe our benchmark creation process step by step.

3.1 Generating Logical Proofs

Our benchmark relies on verified proofs derived from well-established logical formalisms that enable automated reasoning. For the *Biology* and *Recipe* domains, we use the description logics (DLs) $\mathcal{EL}$ and $\mathcal{ALC}$ (Baader et al., 2017), which are decidable fragments of first-order logic widely used for ontological reasoning. They support constructors such as conjunction, existential restriction, and (for $\mathcal{ALC}$) negation and disjunction, allowing the representation of hierarchical and relational knowledge. For the *Drone* domain, we use DatalogMTL (Brandt et al., 2018), an extension of Datalog with metric temporal operators for modeling temporal dependencies and event sequences. These formalisms provide a balance between expressive power and computational tractability, making them suitable for large-scale, verifiable reasoning.

```
"conclusion" : "drone(d)@[−300,+∞)",
"ruleName" : "reverse ⊞",
"premises" : ["⊞[0,+∞)drone(d)@[−300,−300]"],
```

Figure 2: Additional inference step for the proof fragment in Figure 1.

Proofs are automatically generated using existing reasoners: ELK (Kazakov et al., 2014; Kazakov and Klinov, 2014) for $\mathcal{EL}$, LETHE (Koopmann, 2020; Alrabbaa et al., 2020) for $\mathcal{ALC}$, and MeTeoR (Wang et al., 2022) for DatalogMTL, with EVEE (Alrabbaa et al., 2022a) used to extract size-minimal proofs. The formal definitions of the logics and full example proofs for each domain are provided in Appendix A and B, respectively.

3.2 Proofs, Target User Group and Scenarios

Table 1 provides quantitative information about the domains and the extracted proofs. Although *Drones* are formally specified in DatalogMTL rather than DL, we adopt DL terminology throughout this work for consistency. The proof size is the number of conclusions, see Figure 1 (left).

Biology. Our first domain is based on the Cell Line Ontology,[2] a community-driven ontology developed to standardize and integrate cell line information and support computer-assisted reasoning.
Scenario: a 10th-grade student learning about the characteristics of various cells.

Recipe. We created a food and recipe ontology a formal semantic model to represent and reason about culinary recipes, dietary restrictions, and allergen content. This ontology builds upon established recipe and food ontologies (Qi et al., 2018; Dooley et al., 2018).
Scenario: a 6th-grade student learning about food allergens and and dietary classifications (vegan, vegetarian, non-vegetarian).

Drone. The drone ontology models complex situations occurring during a drone flight.
Scenario: a drone pilot monitoring an autonomous drone, needing help identifying (potential) critical situations.

[2] `https://obofoundry.org/ontology/clo.html`

Domain	Ontology		Proofs	
	#Predicates (Arity)	#Axioms	#Proofs (DL)	Proof Size
Biology	43,327 (unary), 249 (binary)	72,830	2,429 ($\mathcal{EL}$)	4–100
Recipe	35,369 (unary), 195 (binary)	52,248	74 ($\mathcal{EL}$), 12 ($\mathcal{ALC}$)	4–100
Drone	201 (of maximal arity 4)	332	50 (DatalogMTL)	9–59

Table 1: Metrics of Ontologies and Logical Proofs. $\mathcal{EL}$ is a lightweight Description Logic and $\mathcal{ALC}$ is a more complex Description Logic. More details on these description logics are available in Appendix A.

The ontology uses metric temporal operators as well as numerical predicates. The associated proofs incorporate different urgency levels to prioritize critical situations. Figure 1 (left) together with Figure 2 provide a proof fragment for the drone domain. The proof begins with the final conclusion $\texttt{warninglvl}(\texttt{d},5)@[0,0]$ and traces its supporting reasons backwards until reaching asserted statements. We use the convention that capital letters (e.g., X, Y) represent variables, and lowercase letters (e.g., d, p1) represent constants. The notation $\texttt{warning}(\texttt{d})@[\texttt{-2},0]$ indicates that object d is in a warning state during the time interval [-2, 0], i.e., in the previous two seconds. If prefixed with a temporal operator, e.g. $\boxplus[0,+\infty)\texttt{drone}(\texttt{d})@[\texttt{-300},\texttt{-300}]$, it means the complex expression holds for a time interval. In this example, the fact that d is a drone holds from time -300 onwards. Rules here must be read from right to left, e.g. $\texttt{rotordamage}(\texttt{Y},\texttt{X}) \texttt{ :- } \texttt{haspart}(\texttt{Y},\texttt{X}),\texttt{drone}(\texttt{Y}),\texttt{damage}(\texttt{X}),\texttt{rotor}(\texttt{X})$ means that upon detection of damage to any $\texttt{rotor}(\texttt{X})$ of $\texttt{drone}(\texttt{Y})$, the system shall trigger a $\texttt{rotordamage}(\texttt{Y},\texttt{X})$ event.

3.3 Task Annotation

To annotate these proofs for all three tasks (highlight, summarize, explain), we recruited two graduate-level students. We divide the task-specific annotation into two phases: Pilot and Main. In the pilot phase, we gave each human annotator 10 proofs per domain and asked them to annotate the generated proofs for the following tasks: (i) identify the key contributing steps to the conclusion from the proof (maximum up to 3 steps), (ii) summarize proofs, and (iii) generate a short user-targeted message.

At the end of the pilot phase, we asked our expert annotator to review the pilot annotations from both our annotators and share detailed feedback on their annotations. We then asked our student annotators to incorporate the feedback. This feedback particularly helped align the most contributing steps between the student annotators. Finally, we divided the remaining 120 proofs equally among them for individual annotations. Throughout the annotation process, the student annotators were prohibited from using any AI tools for annotation purposes. By the end of this annotation process, we collected over 68,000 human-annotated tokens[3] across domains and tasks. We provide detailed annotation statistics in Table 2.

We provide the final version of our annotation guidelines in the Appendix I. We plan to release our full benchmark dataset and code with the camera-ready version of this work.

4 Experiments

We evaluate LLM performance on three explanation tasks given a verified proof as input. Because the tasks are connected, we run each condition both without and with the outputs of earlier tasks included in the context.

4.1 Prompting strategy

We vary two factors: the number of examples in context (zero shot or one shot) and the interaction structure (atomic single turn or chained multi turn). In chained settings we use the sequence highlight key steps, then summarise the proof, then generate the user message. The key steps task is always first and uses no example, so its outputs are identical across strategies. We restrict chained settings to one shot to control context length. All prompt templates are given verbatim in Appendix C.

4.2 LLM coverage

We evaluate nine LLMs: GPT 4.1 mini, GPT 4o mini (Achiam et al., 2023), SmolLM3 3B (Bakouch et al., 2025), Llama 3.1 8B, Llama 3.3

[3]We use GPT-4.1-mini tokenizer to count the number of tokens throughout our work.

Domain	Steps (in tokens)	Steps (in words)	Summary (in tokens)	Summary (in words)	Target User Msg (in tokens)	Target User Msg (in words)
Biology	13331	5530	5691	4674	900	698
Drone	14320	1315	4080	3326	878	688
Recipe	24488	10352	3858	3254	903	649
Total	52139	17197	13629	11254	2681	2035

Table 2: Token and word statistics for human-annotated proofs, summaries, and target user messages.

70B (Grattafiori et al., 2024), Mistral 3.2 24B, Magistral 24B (Rastogi et al., 2025), Qwen3 8B, and Qwen3 32B (Yang et al., 2025b). Further details, including LLM properties and inference settings, are provided in Appendix D.

4.3 Inputs and context length

Proof lengths vary by domain (Table 3). This motivates the one-shot choice in chained settings, which balances comparability across LLMs with input size constraints.

Domain	Min tokens	Max tokens	Avg tokens
Biology	296	2612	1180
Drone	466	3267	1307
Recipe	1272	4058	2021

Table 3: Token statistics of proof input lengths

4.4 Implementation Details

We use vLLM (Kwon et al., 2023) to perform inference on all open-source models in their native precision (i.e., fp16/bf16). For each model, we use the recommended inference parameters (refer to the Appendix E) provided in their model card. We use 4x H100 GPUs to perform all our experiments with open-source models. For the closed-source models, we use OpenAI hosted API services. For both open and closed-source models, we prompt LLMs to generate a JSON response and use a fixed seed value for reproducible outputs.

4.5 Metrics

Steps similarity To quantitatively evaluate the similarity in choosing the most contributing steps, we measure the average depth of the selected proof steps within the reference proof. We represent each proof as a Directed Acyclic Graph (DAG), $G = (V, E)$, where vertices $v \in V$ are the inferences and a directed edge $(u, v) \in E$ indicates that the conclusion of v is a premise for u.

The depth of a step v, denoted $d(v)$, is the length of the longest path from the proof's final conclusion (v_{root}) to v. To standardize this measure across different proofs, we compute a normalized depth, $\hat{d}(v)$, as:

$$\hat{d}(v) = \frac{d(v)}{\max_{u \in V} d(u)}$$

This ensures that $\hat{d}(v) \in [0, 1]$, where 0 corresponds to the root (final conclusion) and 1 to the deepest leaf nodes (asserted conditions). For a given explanation consisting of a set of selected steps $S = \{s_1, \ldots, s_k\}$, we calculate the mean normalized depth $\bar{d}(S)$:

$$\bar{d}(S) = \frac{1}{k} \sum_{i=1}^{k} \hat{d}(s_i)$$

By analyzing the distributions of $\bar{d}(S)$ for each LLM and human annotators, we can identify biases in their selection preferences and measure their alignment with human reasoning patterns. The justification for this metric and additional exact match results are provided in Appendix F.

Summary & target user message We evaluate similarity between the LLM-generated text and the human annotations using the following standard metrics, such as BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), chrF++ (Popović, 2017), and BERTScore (Zhang et al., 2020).

5 Results

Key steps similarity Figure 3 presents the distributions of the mean normalized step depth for most contributing steps identified by humans and LLMs across all domains. This result remains the same across all four baseline strategies, since it is the first step in the chain, and we also do not provide any examples for this task.

Human-identified steps exhibit the highest mean step depth (≈ 0.4) and the widest distribution, cov-

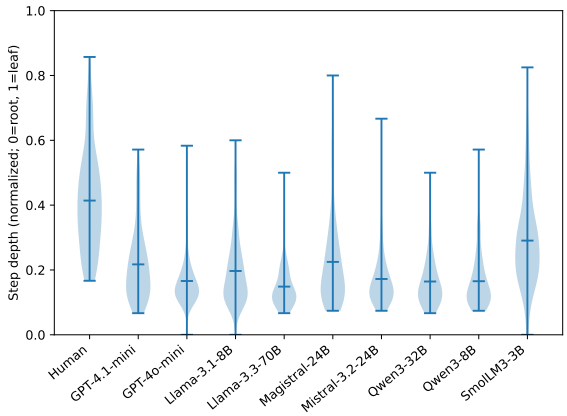

Figure 3: Most contributing steps distribution (averaged across domains)

ering nearly the entire normalized range. This indicates that humans do not adhere to a single fixed strategy. Instead, they flexibly select steps from all levels of the proof graph, from high-level conclusions to foundational premises. In contrast, most LLMs display a strong and consistent bias toward low-depth steps, indicating a preference for inferences that are structurally close to the final conclusion. We also observe that no model successfully replicates the human distribution.

For this task, SmolLM3-3B comes closest to the human average mean depth, outperforming even larger models such as Magistral-24B and GPT-4.1-mini, which rank second and third on this task, respectively. One major limitation we observe in SmolLM3-3B, GPT-4o-mini, and Llama-3.1-8B is that they sometimes pick the final conclusion step itself as the most contributing step. We then dive deeper to evaluate if these patterns are consistent across domains.

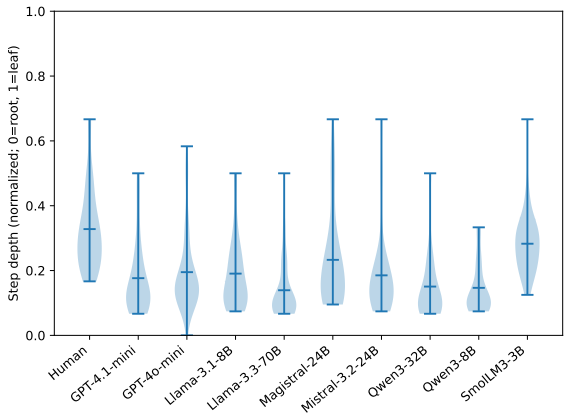

Figure 4: Most contributing steps distribution (Biology)

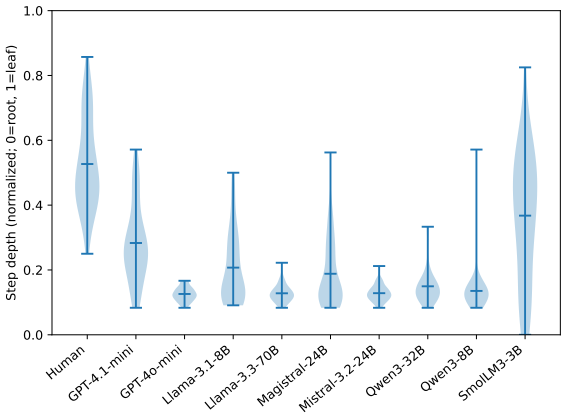

Figure 5: Most contributing steps distribution (Drone)

We observe clear differences in distribution patterns across domains. In the Biology domain (Figure 4), human-identified steps have a lower mean depth compared to other domains, indicating a preference for steps closer to the final conclusion, while still maintaining a wide distribution. Among the LLMs, SmolLM3-3B again exhibits a mean depth closest to the human average in this specific domain.

In the Drone domain, as shown in Figure 5, human-selected steps have the highest mean depth across all domains (≥ 0.5) and a broader distribution. On the contrary, all LLMs exhibit a strong preference for low-depth steps, creating the most significant gap between human and model distributions observed in our study. While SmolLM3-3B remains the best-performing model in terms of matching the mean, it also at times picks the final conclusion step as the most contributing one.

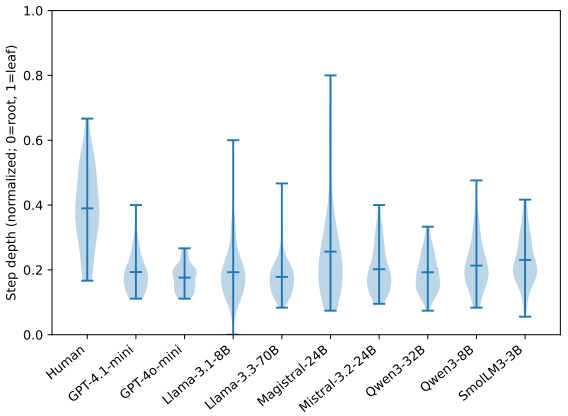

Figure 6: Most contributing steps distribution (Recipe)

Figure 6 shows the distributions for the Recipe domain. The human distribution is similar to the

cross-domain average, with a mean depth of ≈ 0.4 and noticeable variance. Most LLMs again gravitate towards steps structurally close to the conclusion. In this domain, Magistral-24B outperforms SmolLM3-3B, though all the LLMs fall short of replicating the breadth and average depth of human-selected steps.

In conclusion, the normalized step depth metric reveals a fundamental difference between human flexibility and LLM bias. While humans select key contributing steps from all levels of a reasoning chain, LLMs consistently favor low-depth steps near the final conclusion. This opens up another future research opportunity in developing LLMs that can replicate human understanding, leading to a more human-aligned identification of pivotal reasoning steps.

Summary & target user message similarity We evaluate the LLMs' performance against human-written references using BLEU, ROUGE-L, chrF++, and BERTScore. The performance under the one-shot atomic strategy is nearly identical, with the chained context providing a marginal overall improvement. The results for the rest three prompting strategies are available in Appendix G.

The results for the one-shot chained strategy are presented in Figure 7. These results suggest that, for the *summary* task, Llama-3.1-8B and Mistral-3.2-24B consistently achieve the highest scores across all four metrics. However, for the more constrained *target user message* task, GPT-4o-mini emerges as one of the top-performing LLMs along with Mistral-3.2-24B. In contrast, LLMs like GPT-4.1-mini and SmolLM3-3B generally rank lower on these generation tasks compared to their performance on the step selection task.

6 Human Evaluation

While automated metrics like BLEU, ROUGE, and BERTScore offer scalable evaluation, these metrics often fail to capture nuances of *faithfulness*, *readability*, and *appropriateness*, which are vital for user-facing summaries and messages. The results in Figure 7 show that top-performing models, such as Llama-3.1-8B and Mistral-3.2-24B, achieve very similar scores. Automated metrics alone make it difficult to determine if these small numerical differences translate into meaningful improvements in quality or to verify if the generated content is factually correct and truly useful.

To address these limitations and provide a more robust assessment, we conduct a human evaluation study. We select a diverse set of models for comparison against a Human baseline: the two top performers according to automated metrics (Mistral-3.2-24B and Llama-3.1-8B), a larger model from the same family (Llama-3.3-70B), and a recent proprietary model (GPT-4.1-mini). This selection allows us to validate whether the top automated scores correspond to actual qualitative improvements, and to explore the performance differences between various model types and sizes. To carry out the evaluation, we recruit five human annotators to rate the outputs for both tasks, using the following task-specific criteria.[4]

Summary criteria The evaluators rate the summary based on the following aspects. *Faithfulness* assesses the degree to which the reference proof factually supports every statement in the summary. *Readability* judges the summary based on its clarity, grammar, and ease of comprehension. *Conciseness* assesses if the summary only contains essential information or not. *Coverage* determines if the summary captures all key reasoning steps and the main conclusion.

Target message criteria The evaluators rate the target user message based on the following aspects. *Faithfulness* assesses the degree to which the reference proof factually supports the target message. *Appropriateness* judges the overall suitability of the message for its target audience, combining aspects of faithfulness, clarity, and conciseness. *Coverage* determines if the summary target message contains the main conclusion and at least one main reason.

6.1 Results

The human evaluation statistics are presented in Table 4. These indicate qualitative differences between the LLMs and Human annotators. Based on this evaluation, GPT-4.1-mini and Mistral-3.2-24B are the best-performing LLMs overall and they score higher compared to the Llama family LLMs. For the summary task, GPT-4.1-mini and Mistral-3.2-24B obtained the highest scores for the Coverage criteria, but they fall short on the Conciseness criteria. This indicate the verbose nature of these LLMs. We confirm this by plotting average summary and target message lengths for all the LLMs. The plots are available in Appendix H. In the Target Message task, GPT-4.1-mini achieves the highest

[4]The human evaluation guide is available in Appendix I.

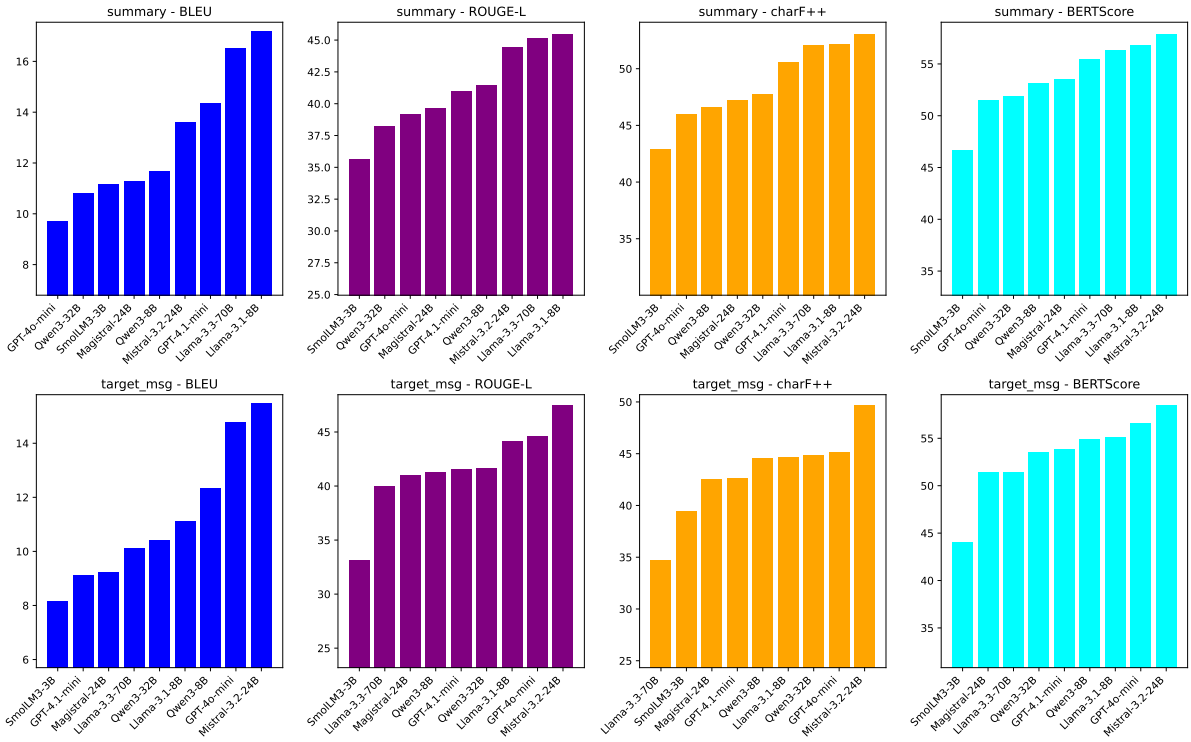

Figure 7: Automated metrics of the best performing strategy (One-shot chained)

	Summary criteria				Target Message criteria		
Model Name	Conciseness	Coverage	Faithfulness	Readability	Appropriateness	Coverage	Faithfulness
Llama-3.1-8B	4.07	3.65	3.76	4.05	3.54	3.69	3.67
Mistral-3.2-24B	4.01	4.63	3.86	3.83	3.85	4.19	4.11
GPT-4.1-mini	3.79	4.60	4.00	3.57	3.89	4.33	4.24
Llama-3.3-70B	4.14	3.84	4.00	3.84	3.62	3.46	4.44
Human	**4.55**	**4.80**	**4.80**	**4.35**	**4.59**	**4.47**	**4.96**

Table 4: Averaged human evaluation scores across all domains and annotators

scores across all three criteria: Appropriateness, Coverage, and Faithfulness.

These findings diverge from the automated evaluation results shown in Figure 7. While automated metrics ranked Llama-3.1-8B and Mistral-3.2-24B as having similar top-tier performance, human annotators rated Llama-3.1-8B lower than both Mistral-3.2-24B and GPT-4.1-mini. This suggests a limitation of automated metrics in capturing nuanced aspects of text quality and demonstrates the utility of human judgment for such assessments.

We finally assess the reliability of the human evaluation by calculating inter-annotator agreement using Fleiss' Kappa (κ). The results indicate that the five annotators reached a substantial level of agreement, which supports the reliability of our findings. For the Summary criteria, annotators achieved almost perfect agreement for Faithfulness ($\kappa = 0.813$), and substantial agreement for Readability ($\kappa = 0.677$) and Conciseness ($\kappa = 0.721$). Agreement for Coverage ($\kappa = 0.442$) was moderate, suggesting a higher degree of subjectivity for this criterion. For the Target Message criteria, all three aspects showed substantial agreement: Faithfulness ($\kappa = 0.774$), Appropriateness ($\kappa = 0.678$), and Coverage ($\kappa = 0.656$). These agreement scores demonstrate that annotators applied the evaluation criteria consistently and that the criteria themselves were well defined.

6.2 Error Analysis

In order to understand the ratings a bit better, we asked our human evaluators to provide us with the reasoning for the scores for 5 low-scoring samples per domain per LLM. We present our findings here.

Summary We observe that almost all the LLMs use some technical terms from the proof verbatim, especially Mistral-3.2-24B and GPT-4.1-mini. This

leads to lower readability scores in general. For Llama-3.1-8B, we observed a consistent pattern of incorrectly using the term “equivalent" when the proof demonstrates a subclass relationship. This recurring issue indicates a potential weakness in understanding logical operators.

Target user message We observed that for the GPT-4.1-mini uses left-branching sentence structure, which increases memory load and scores lower on the appropriateness criteria. An example target message showcasing this issue is – *Because saucy shepherd pie has carrots as ingredients, it needs special allergen labels about carrots.* Llama-3.1-8B suffers again from its inability to understand logical operators. Interestingly, Llama-3.3-70B produces very short target messages (i.e., *Cell derivation from Mus musculus*) and achieves lower ratings for coverage criteria. This observation is also supported by Figure 19 in the Appendix.

6.3 Downstream implications of the near conclusion bias

Our step selection results (Section 5 show that LLMs prefer low depth steps near the final conclusion. We test the practical effect of this bias on user facing communication. We run a targeted expert study on 20 Drone instances in which an LLM selected near conclusion steps. The expert rates the target messages on Appropriateness from 1 to 5 (Higher is better). Table 5 reports the scores for one-shot atomic and one-shot chained settings.

LLM	One-shot chained	One-shot atomic
GPT-4.1-mini	3.40	3.80
Llama-3.3-70B	2.50	2.60
Llama-3.1-8B	3.10	3.10
Mistral-3.2-24B	4.10	4.20

Table 5: Appropriateness in the Drone domain when selected steps are near the final conclusion.

In both *one-shot atomic* and *one-shot chained* settings, LLMs that relied on final conclusion steps received consistently lower Appropriateness scores, with Mistral-3.2-24B as the exception. The drop arises because the messages often included unhelpful technical details such as “Warning level: X” and omitted one or more causal reasons that explain the critical situation. By contrast, our student annotators did not include any warning level details in their target messages and never selected steps containing `warninglvl` as key steps in any of the 50 annotated proofs.

7 Conclusion

We introduce `ProofTeller`, a benchmark to evaluate LLM reliability in explaining formal proofs via key step identification, summarization, and user messaging. Our experiments with nine LLMs reveal reliability gaps. We find that LLMs consistently favor low depth steps near the conclusion, whereas humans select steps from all over the proof. This suggests LLMs lack a holistic understanding of the reasoning chain. Further, our human evaluation of summarization tasks highlights qualitative deficiencies in faithfulness and conciseness not captured by automated metrics. A targeted expert study also shows that this near conclusion bias lowers Appropriateness in user facing messages by encouraging unhelpful technical details and omitting causal reasons.

Overall, LLMs can interpret a formal proof and restate it fluently, but they do not yet do so reliably or intelligibly for a non-expert, largely because they exhibit a recency bias when interpreting external reasoning traces.

Limitations

Experimental limitations While we took a systematic approach, our exploration of the vast prompt engineering space was limited. We did our initial testing with three distinct seed variations, which may not fully capture the possible variability in output. Furthermore, we restricted prompt variations for each task to five, potentially overlooking other effective phrasings or structures that could yield superior performance. Finally, we confined the initial evaluation of the system prompt’s influence to three language models before finalizing the one used for this work, meaning findings may not be consistent across all possibly suited system prompts.

Scope limitation Our work is limited to three domains within DL and DatalogMTL formalisms. Moreover, to maintain the original notations, the proof syntax employs logic-specific Unicode symbols (e.g. $\sqsubseteq$, $\boxplus$) and specialized terminology (e.g. `eliminate` and `IntersectionComposition`) requiring LLMs to first recognize their semantic meaning before interpreting the logical implications.

Evaluation subjectivity Evaluation of explanation quality can be inherently subjective, especially with respect to aspects such as appropriateness and

faithfulness. Different annotators may interpret the relevance and accuracy of an explanation in diverse ways, leading to potential variability in the assigned scores. To mitigate this, we provided clear guidelines and examples, but acknowledge that some level of subjectivity is unavoidable in human evaluations of natural language explanations.

Ethics Statement

All annotators involved in the evaluation process are either co-authors of this paper or were fairly compensated for their time, receiving above the minimum wage of 14.5 EUR per hour. This ensures ethical standards in data annotation and helps maintain the quality and reliability of the evaluation results.

Acknowledgments

The authors extend gratitude to Saumil Shah, Leixin Zhang, Zaynab Reza, Pratistha Kansakar and Zhenyu Feng for their help in the human evaluation study. We would like to thank Yash Sarrof, Dongqi Liu, Soyoung Oh, and Blerta Veseli for discussions and feedback on the draft. This work is funded by Deutsche Forschungsgemeinschaft (DFG) grant 389792660 as part of TRR 248 – CPEC, see https://perspicuous-computing.science. Brisca Balthes was further supported by the Konrad Zuse School of Excellence in Learning and Intelligent Systems (ELIZA) through the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research. Yifan Wang was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – GRK 2853/1 "Neuroexplicit Models of Language, Vision, and Action" - project number 471607914. The authors gratefully acknowledge the computing time made available to them on the high-performance computer at the NHR Center of TU Dresden. This center is jointly supported by the Federal Ministry of Research, Technology and Space of Germany and the state governments participating in the NHR (www.nhr-verein.de/unsere-partner).

References

Yasin Abbasi Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvari. 2024. To believe or not to believe your llm: Iterative prompting for estimating epistemic uncertainty. *Advances in Neural Information Processing Systems*, 37:58077–58117.

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

Christian Alrabbaa, Franz Baader, Stefan Borgwardt, Patrick Koopmann, and Alisa Kovtunova. 2020. Finding small proofs for description logic entailments: Theory and practice. In *LPAR23. LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 73 of *EPiC Series in Computing*, pages 32–67. EasyChair.

Christian Alrabbaa, Stefan Borgwardt, Tom Friese, Patrick Koopmann, Julián Méndez, and Alexej Popovic. 2022a. On the eve of true explainability for OWL ontologies: Description logic proofs with evee and evonne. In *Proceedings of the 35th International Workshop on Description Logics (DL 2022) co-located with Federated Logic Conference (FLoC 2022), Haifa, Israel, August 7th to 10th, 2022*, volume 3263 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Christian Alrabbaa, Stefan Borgwardt, Patrick Koopmann, and Alisa Kovtunova. 2022b. Finding good proofs for answers to conjunctive queries mediated by lightweight ontologies. In *Proceedings of the 35th International Workshop on Description Logics (DL 2022) co-located with Federated Logic Conference (FLoC 2022), Haifa, Israel, August 7th to 10th, 2022*, volume 3263 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. 2017. *An Introduction to Description Logic*. Cambridge University Press.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, and 4 others. 2025. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3.

Yejin Bang, Ziwei Ji, Alan Schelten, Anthony Hartshorn, Tara Fowler, Cheng Zhang, Nicola Cancedda, and Pascale Fung. 2025. Hallulens: Llm hallucination benchmark. *arXiv preprint arXiv:2504.17550*.

Stefan Borgwardt, Vera Demberg, Mayank Jobanputra, Alisa Kovtunova, and Duy Nhu. 2024. Explaining critical situations over sensor data streams using proofs and natural language. In *Proceedings of the*

37th International Workshop on Description Logics (DL 2024), Bergen, Norway, June 18-21, 2024, volume 3739 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Sebastian Brandt, Elem Güzel Kalayci, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. 2018. Querying log data with metric temporal logic. *J. Artif. Intell. Res.*, 62:829–877.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Andrea Colombo, Teodoro Baldazzi, Luigi Bellomarini, Emanuel Sallinger, and Stefano Ceri. 2025. Template-based explainable inference over high-stakes financial knowledge graphs. In *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025*, pages 503–515. OpenProceedings.org.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Can Cui, Yunsheng Ma, Zichong Yang, Yupeng Zhou, Peiran Liu, Juanwu Lu, Lingxi Li, Yaobin Chen, Jitesh H Panchal, Amr Abdelraouf, and 1 others. 2024. Large language models for autonomous driving (llm4ad): Concept, benchmark, experiments, and challenges. *arXiv preprint arXiv:2410.15281*.

Damion M. Dooley, Emma J. Griffiths, Gurinder S. Gosal, Pier L. Buttigieg, Robert Hoehndorf, Matthew C. Lange, Lynn M. Schriml, Fiona S. L. Brinkman, and William W. L. Hsiao. 2018. FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration. *npj Science of Food*, 2(23).

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Andrew Grattafiori and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Nora Kassner, Oyvind Tafjord, Hinrich Schütze, and Peter Clark. 2021. BeliefBank: Adding memory to a pre-trained language model for a systematic notion of belief. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8849–8861, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yevgeny Kazakov and Pavel Klinov. 2014. Goal-directed tracing of inferences in EL ontologies. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, volume 8797 of *Lecture Notes in Computer Science*, pages 196–211. Springer.

Yevgeny Kazakov, Markus Krötzsch, and Frantisek Simancik. 2014. The incredible ELK – from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. *J. Autom. Reasoning*, 53(1):1–61.

Patrick Koopmann. 2020. LETHE: Forgetting and uniform interpolation for expressive description logics. *KI - Künstliche Intelligenz*.

Atharv Kulkarni, Kushagra Dixit, Vivek Srikumar, Dan Roth, and Vivek Gupta. 2025. Llm-symbolic integration for robust temporal tabular reasoning. *arXiv preprint arXiv:2506.05746*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Liam Daly Manocchio, Siamak Layeghy, Wai Weng Lo, Gayan K Kulatilleke, Mohanad Sarhan, and Marius Portmann. 2024. Flowtransformer: A transformer framework for flow-based network intrusion detection systems. *Expert Systems with Applications*, 241:122564.

Philipp Mondorf and Barbara Plank. 2024. Comparing inferential strategies of humans and large language models in deductive reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9370–9402, Bangkok, Thailand. Association for Computational Linguistics.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, and 5 others. 2021. DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Miao Qi, Yarden Neeman, Forest Eckhardt, and Kevin Blissett. 2018. WhatToMake: a semantic web application for recipe recommendation. *Rensselaer Polytechnic Institute*.

A. Rastogi and 1 others. 2025. Magistral. *arXiv preprint arXiv:2506.10910*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*.

Michael Shalyt, Rotem Elimelech, and Ido Kaminer. 2025. Asymob: Algebraic symbolic mathematical operations benchmark. *arXiv preprint arXiv:2505.23851*.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*.

Kaiser Sun, Fan Bai, and Mark Dredze. 2025. What is seen cannot be unseen: The disruptive effect of knowledge conflict on large language models. *arXiv preprint arXiv:2506.06485*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965.

Vy Vo, Lizhen Qu, Tao Feng, Yuncheng Hua, Xiaoxi Kang, Songhai Fan, Tim Dwyer, Lay-Ki Soon, and Gholamreza Haffari. 2025. ACCESS : A benchmark for abstract causal event discovery and reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1049–1074, Albuquerque, New Mexico. Association for Computational Linguistics.

Dingmin Wang, Pan Hu, Przemyslaw Andrzej Walega, and Bernardo Cuenca Grau. 2022. MeTeoR: Practical reasoning in datalog with metric temporal operators. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 5906–5913. AAAI Press.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang and 1 others. 2025b. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Preliminaries

A.1 DL Proofs

The proofs in our benchmark are based on the two description logics (DLs) $\mathcal{EL}$ and $\mathcal{ALC}$ (Baader et al., 2017) as well as DatalogMTL (Brandt et al., 2018), an extension of Datalog with metric temporal operators for querying temporal data.

The syntax of DLs is based on disjoint, countably infinite sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of *concept names* A, B, ... and *role names* r, s, ..., respectively. In $\mathcal{EL}$, *concepts* are built from concept names by applying the constructors $\top$ (top), $C \sqcap D$ (conjunction), and $\exists r.C$ (existential restriction for a role name r). A *general concept inclusion (GCI)* η is of the form $C \sqsubseteq D$, where C and D are $\mathcal{EL}$ concepts, and a finite set of GCIs is called a *TBox* or *ontology*. The DL $\mathcal{ALC}$ extends $\mathcal{EL}$ by the concept constructors $\bot$ (bottom), $C \sqcup D$ (disjunction), $\forall r.C$ (value restriction), and $\neg C$ (negation). For the semantics, in particular when a GCI η is *entailed* by a TBox $\mathcal{T}$ (written $\mathcal{T} \models \eta$), we refer the reader to (Baader et al., 2017).

In contrast to $\mathcal{EL}$ and $\mathcal{ALC}$, temporal reasoning in DatalogMTL also takes facts about constants into account. A (function-free first-order) *atom* has the form $P(\tau)$ with P a predicate of some arity n and τ an n-ary tuple consisting of constants and variables. A *literal* (or *metric atom*) A takes one of the following forms, where ϱ is a non-empty positive rational interval:

$$A := \top \mid \bot \mid P(\tau) \mid \diamondminus_\varrho A \mid \boxminus_\varrho A \mid \diamondplus_\varrho A \mid \boxplus_\varrho A \mid A \,\mathcal{S}_\varrho A \mid A \,\mathcal{U}_\varrho A$$

A *rule* with *body literals* $A_1, \ldots, A_n$, $n \geq 1$, and *head literal* B is of the form:

$$B \;:-\; A_1, \ldots, A_n,$$

with B not containing the operators $\diamondminus$, $\diamondplus$, $\mathcal{S}$ or $\mathcal{U}$. If an atom, literal or rule contains no variable, we call it *ground*. A *fact* F is defined as an expression of the form $A@\varrho$ where A is a ground atom and ϱ a rational interval. Moreover, we call a finite set D of facts a *dataset* and a finite set Π of rules a *program*. In this context, entailments are of the form $\Pi, \mathcal{D} \models F$. However, for simplicity, we now denote entailments in DLs and DatalogMTL uniformly by $\mathcal{T} \models \eta$.

Our goal is to explain a logical consequence $\mathcal{T} \models \eta$, where $\mathcal{T}$ is either a TBox or a program together with a dataset, and η is a GCI or a fact, respectively. Following (Alrabbaa et al., 2020, 2022b), *proofs* of $\mathcal{T} \models \eta$ are finite, acyclic, directed hypergraphs, where vertices v are labeled with GCIs or facts $\ell(v)$ and hyperedges are of the form (S, d), with S a tuple of vertices and d a vertex such that $\{\ell(v) \mid v \in S\} \models \ell(d)$; the leafs of a proof must be labeled by elements of $\mathcal{T}$ and the unique sink vertex by η. In addition, an edge labeling function (see the `ruleName` key in Figures 8, 9, 10) indicates which logical rule derived a conclusion $\ell(d)$ from the premises. The *size* of a proof is the number of its vertices.

A logical proof starts from a conclusion and identifies the premises that entail it. These premises are then justified by further reasoning, building a chain of inference that ultimately rests on assertions—statements that are accepted as true without derivation.

For this benchmark, $\mathcal{EL}$ proofs were generated using the reasoner ELK (Kazakov et al., 2014; Kazakov and Klinov, 2014), while for $\mathcal{ALC}$, we employed the forgetting tool LETHE (Koopmann, 2020; Alrabbaa et al., 2020). For DatalogMTL, we extended the Metric Temporal Reasoner (MeTeoR) (Wang et al., 2022) to trace the applied reasoning steps (Borgwardt et al., 2024). We also used EVEE (EVincing Expressive Entailments) (Alrabbaa et al., 2022a), a Java library that can extract size-minimal proofs from the output of a reasoner.

B Proof Examples

Biology Example In Figure 8, one can see an example of a proof that RMUG-S is an immortal human cell line cell (IhCL for short). An *immortal cell line* (ICL) is expected to be capable of an unlimited number of divisions, and is thus able to support indefinite propagation *in vitro*. RMUG-S is a human (lat. *Homo sapiens*) ovarian adenocarcinoma cell line originated from a 62 year old Japanese female.

Recipe example In Figure 9, one can see an example of a vegan bread recipe.

Critical Situations in Drone example The proof in Figure 10 show a temporal proof for the drone probably having internal damage at time point 0 by detecting an internal temperature above the threshold temperature.

```
finalConclusion: RMUG-S ⊑ IhCL,
inferences: [ {
        conclusion: RMUG-S ⊑ IhCL,
        ruleName: Class Hierarchy,
        premises: [ RMUG-S ⊑ (∃derived from.∃part
             of.Homo sapiens ⊓ ICL),
            (∃derived from.∃part of.Homo sapiens ⊓
                 ICL) ⊑ IhCL ]
    },{
        conclusion: RMUG-S ⊑ (∃derived from.∃part
             of.Homo sapiens ⊓ ICL),
        ruleName: Intersection Composition,
        premises: [ RMUG-S ⊑ ICL,
            RMUG-S ⊑ ∃derived from.∃part of.Homo
                 sapiens ]
    },{
        conclusion: RMUG-S ⊑ ICL,
        ruleName: Asserted Conclusion
    },{
        conclusion: RMUG-S ⊑ ∃derived from.∃part
             of.Homo sapiens,
        ruleName: Asserted Conclusion
    },{
        conclusion: (∃derived from.∃part of.Homo
             sapiens ⊓ ICL) ⊑ IhCL,
        ruleName: Equivalent Classes Decomposition,
        premises: IhCL ≡ (∃derived from.∃part
             of.Homo sapiens ⊓ ICL)
    },{
        conclusion: IhCL ≡ (∃derived from.∃part
             of.Homo sapiens ⊓ ICL),
        ruleName: Asserted Conclusion } ]
```

Figure 8: Biology proof. The proof begin with a final conclusion and trace its supporting reasons backwards until reaching asserted statements. A guide to the symbols used in this proof. The symbol $\sqsubseteq$ means "is a type of". The symbol $\equiv$ means "is the same as". The symbol $\sqcap$ means "and". The symbol $\exists$ means "some". For example, RMUG-S $\sqsubseteq \exists$dF.$\exists$pOf.Homo sapiens can be read as RMUG-S is a type of an entity that is derived from some part of some Homo sapience.

```
finalConclusion: bread ⊑ vegan recipe,
inferences: [ {
        conclusion: bread ⊑ vegan recipe,
        ruleName: eliminate 'flour',
        premises: [ (∀ingr.(water ⊔ flour) ⊓
             bread) ⊑ vegan recipe,
            bread ⊑ ∀ingr.(water ⊔ flour) ]
    },{
        conclusion: (∀ingr.(water ⊔ flour) ⊓
             bread) ⊑ vegan recipe,
        ruleName: eliminate 'vegan',
        premises: [ water ⊑ vegan,
            flour ⊑ vegan,
            (∀ingr.vegan ⊓ bread) ⊑ vegan recipe ]
    },{
        conclusion: water ⊑ vegan,
        ruleName: asserted
    },{
        conclusion: flour ⊑ vegan,
        ruleName: asserted
    },{
        conclusion: (∀ingr.vegan ⊓ bread) ⊑ vegan
             recipe,
        ruleName: eliminate 'food recipe',
        premises: [ bread ⊑ (food recipe ⊓
             ∃ingr.flour ⊓ ∃ingr.water),
            vegan recipe ≡ (∀ingr.vegan ⊓ food
                 recipe) ]
    },{
        conclusion: bread ⊑ (food recipe ⊓
             ∃ingr.flour ⊓ ∃ingr.water),
        ruleName: asserted
    },{
        conclusion: vegan recipe ≡ (∀ingr.vegan ⊓
             food recipe),
        ruleName: asserted
    },{
        conclusion: bread ⊑ ∀ingr.(water ⊔ flour),
        ruleName: asserted } ]
```

Figure 9: Recipe proof. The proof begin with a final conclusion and trace its supporting reasons backwards until reaching asserted conclusions. A guide to the symbols used in this proof. The symbol $\sqsubseteq$ means "is a type of". The symbol $\equiv$ means "is the same as". The symbol $\sqcap$ means "and", $\sqcup$ means "or". The symbol $\exists$ means "some". The symbol $\forall$ means "every". For example, bread $\sqsubseteq \forall$ingr.(water $\sqcup$ flour) can be read as bread is a type of entity, every ingredient of which is either flour or water .

```
finalConclusion : warninglvl(d,4)@[0,0],
inferences : [ {
        conclusion : warninglvl(d,4)@[0,0],
        ruleName : inclusion,
        premises : warninglvl(d,4)@[-1,9]
    }, {
        conclusion : warninglvl(d,4)@[-1,9],
        ruleName : warninglvl(X,4) :- risk(X),
        premises : risk(d)@[-1,9]
    }, {
        conclusion : risk(d)@[-1,9],
        ruleName : risk(X) :-
             riskofinternaldamage(X),
        premises : riskofinternaldamage(d)@[-1,9]
    }, {
        conclusion : riskofinternaldamage(d)@[-1,9],
        ruleName : reverse ⊞,
        premises :
             ⊞[0,10]riskofinternaldamage(d)@[-1,-1]
    }, {
        conclusion :
             ⊞[0,10]riskofinternaldamage(d)@[-1,-1],
        ruleName :
             ⊞[0,10]riskofinternaldamage(Y) :-
             internaltemperature(Y,S),drone(Y),S>40,
        premises : [
             internaltemperature(d,48)@[-1,-1],
            drone(d)@[-300,+∞) ]
    }, {
        conclusion :
             internaltemperature(d,48)@[-1,-1],
        ruleName : Asserted
    }, {
        conclusion : drone(d)@[-300,+∞),
        ruleName : reverse ⊞,
        premises : ⊞[0,+∞)drone(d)@[-300,-300]
    }, {
        conclusion : ⊞[0,+∞)drone(d)@[-300,-300],
        ruleName : ⊞[0,+∞)drone(X) :- drone(X),
        premises : drone(d)@[-300,-300]
    }, {
        conclusion : drone(d)@[-300,-300],
        ruleName : Asserted } ]
```

Figure 10: Example proof for a critical scenario for drones. The proof begin with a final conclusion and trace its supporting reasons backwards until reaching asserted statements. A guide to the symbols used in this proof. We use the convention that late-alphabet capital letters (e.g., `X`, `Y`) represent variables, and early lowercase letters (e.g., `d`) represent constants. The notation $\texttt{risk(d)@}[-1, 9]$ indicates that object `d` is at `risk` during the time interval $[-1, 9]$. If prefixed with a temporal operator, e.g. $\boxplus[0, 10]\texttt{riskofinternaldamage(d)@}[-1, -1]$, it means the complex expression holds for a time interval. For instance, for any start time within the interval $[-1, -1]$, the fact `riskofinternaldamage(d)` will last for the next `10` consecutive time points. Rules here must be read from right to left, e.g. $\boxplus[0, 10]\texttt{riskofinternaldamage(Y)} :- \texttt{internaltemperature(Y,S)}, \texttt{drone(Y)}, \texttt{S} > 40$ means an internal temperature `S` exceeding `40` degrees in a `drone(Y)` triggers a `riskofinternaldamage(Y)` that persists for the following `10` consecutive time points.

C Prompts

We provide all the prompts verbatim in Figures 11–14.

D Models used in experiments

Table 6 summarises the LLMs used in our experiments. We include whether a model is open weight and whether it exposes a native thinking mode.

LLM	Open	Thinking tokens
GPT 4.1 mini	No	No
GPT 4o mini	No	No
SmolLM3 3B	Yes	No
Llama 3.1 8B	Yes	No
Mistral 3.2 24B	Yes	No
Llama 3.3 70B	Yes	No
Qwen3 8B	Yes	Yes
Magistral 24B	Yes	Yes
Qwen3 32B	Yes	Yes

Table 6: LLMs used in our experiments.

E Inference details

For the open weights LLMs, we use the exact same inference parameters mentioned in the LLM model card on their respective HuggingFace model page. For more details, please refer to our codebase here:

F Graph convention and additional metrics

Graph convention. We represent each proof as a finite, acyclic, directed hypergraph. The final conclusion is the root at depth 0 and axioms appear at larger depths. The opposite orientation is also common in proof theory. Both views are equivalent up to edge reversal and induce the same family of trees. Any depth based statistic can be translated between the two views by a monotone transform. We choose the root at the conclusion because it simplifies normalisation across domains and visual inspection of what lies closer to the decision point.

Why average depth beyond exact match. Exact match measures whether an LLM selects the same steps as a human. In our setting there are many valid subsets of informative steps and near misses contain useful signal. Average normalised depth captures the distributional tendency of where an LLM looks in the proof. This metric exposed the near conclusion bias that exact match alone did not reveal.

System Prompt

You are an expert on Description Logic and DatalogMTL proofs. You are also good at explaining complex things in an easy way.

Task 1: Identify Most Contributing Steps

`{{proof_method_description}}`

`### TASK - 1 ###`
List the steps that contribute the most in deriving the final conclusion (at most 3 steps). Put the exact steps verbatim from the proof below, including conclusion, rule name, and premises.

`{{proof}}`

Output **strictly** a JSON object matching this schema:
`{{ StepsOutput.model_json_schema() | tojson }}`

LLM response

`{{JSON_response}}`

Task 2: Summarize the Proof

`### TASK - 2 ###`
An example summary for an example proof was written by a human expert as shown below.
Example Proof: *`{{example_proof}}`*
Example Summary: *`{{example_summary}}`*

Summarize the proof shown in Task 1 (i.e., Finding the most contributing steps) similar to a human expert. The summary should contain the conclusion of the proof and how it was reached.

Output **strictly** a JSON object matching this schema:
`{{ SummaryOutput.model_json_schema() | tojson }}`

LLM response

`{{JSON_response}}`

Task 3: Generate Targeted Message

`### TASK - 3 ###`
An example message for the targeted user was written by a human expert as shown below. The human expert utilized the 'Example Proof' and the 'Example Summary' to write this message.
Example Target Message: *`{{example_target_msg}}`*

Generate a message for the targeted user similar to a human expert for the proof shown in Task 1. The message should be a maximum of 15-20 words long and it should explain to `{{user}}` what exact condition led to the final conclusion.

Output **strictly** a JSON object matching this schema:
`{{ TargetMsgOutput.model_json_schema() | tojson }}`

LLM response

`{{JSON_response}}`

Figure 11: Prompt template for One-shot chained; same color for all tasks indicates multi-turn response chain, adding the previous context for all the tasks.

Exact match results. For completeness, Table 7 reports exact match accuracy by domain. The low and variable scores confirm that exact match is a coarse indicator for this task, while the depth analysis in the main paper captures the systematic selection bias.

System Prompt

You are an expert on Description Logic and DatalogMTL proofs. You are also good at explaining complex things in an easy way.

Task 1: Identify Most Contributing Steps

```
{{proof_method_description}}
```

TASK - 1
List the steps that contribute the most in deriving the final conclusion (at most 3 steps). Put the exact steps verbatim from the proof below, including conclusion, rule name, and premises.

```
{{proof}}
```

Output **strictly** a JSON object matching this schema:
`{{ StepsOutput.model_json_schema() | tojson }}`

LLM response

`{{JSON_response}}`

Task 2: Summarize the Proof

TASK - 2
Summarize the proof shown in Task 1 (i.e., Finding the most contributing steps) similar to a human expert. The summary should contain the conclusion of the proof and how it was reached.

Output **strictly** a JSON object matching this schema:
`{{ SummaryOutput.model_json_schema() | tojson }}`

LLM response

`{{JSON_response}}`

Task 3: Generate Targeted Message

TASK - 3
Generate a message for the targeted user similar to a human expert for the proof shown in Task 1. The message should be a maximum of 15-20 words long and it should explain to `{{user}}` what exact condition led to the final conclusion.

Output **strictly** a JSON object matching this schema:
`{{ TargetMsgOutput.model_json_schema() | tojson }}`

LLM response

`{{JSON_response}}`

Figure 12: Prompt template for Zero-shot chained; same color for all tasks indicates multi-turn response chain, adding the previous context for all the tasks.

G Additional Results on Summary & Target User Message similarity automated metrics

The results in Figure 7 and 15 show that providing the context trail leads to a slight but consistent improvement across all models and metrics for both the summary and target user message tasks. For instance, the ROUGE-L score for Llama-3.1-8B on the summary task improves from approximately 44.5 to 45.0 when the context trail is included. Similarly, the BERTScore for GPT-4o-mini on the target message task increases from around 57.0 to 57.5. While minor, this trend suggests that access to the full reasoning path, even when generating a summary of it, provides valuable context that helps the models produce outputs that are more aligned with the human-written references. This indicates that for generation tasks grounded in a logical proof, providing the complete proof structure is beneficial.

System Prompt

You are an expert on Description Logic and DatalogMTL proofs. You are also good at explaining complex things in an easy way.

Task 1: Identify Most Contributing Steps

`{{proof_method_description}}`

`### TASK - FIND THE MOST CONTRIBUTING STEPS ###`
List the steps that contribute the most in deriving the final conclusion (at most 3 steps). Put the exact steps verbatim from the proof below, including conclusion, rule name, and premises.

`{{proof}}`

Output **strictly** a JSON object matching this schema:
`{{ StepsOutput.model_json_schema() | tojson }}`

LLM response

`{{JSON_response}}`

Task 2: Summarize the Proof

`### TASK - SUMMARIZE PROOF ###`
An example summary for an example proof was written by a human expert as shown below.
Example Proof: *`{{example_proof}}`*
Example Summary: *`{{example_summary}}`*

Summarize the proof shown in Task 1 (i.e., Finding the most contributing steps) similar to a human expert. The summary should contain the conclusion of the proof and how it was reached.

Output **strictly** a JSON object matching this schema:
`{{ SummaryOutput.model_json_schema() | tojson }}`

LLM response

`{{JSON_response}}`

Task 3: Generate Targeted Message

`### TASK - TARGET-USER MESSAGE ###`
An example message for the targeted user was written by a human expert as shown below. The human expert utilized the 'Example Proof' and the 'Example Summary' to write this message.
Example Target Message: *`{{example_target_msg}}`*

Generate a message for the targeted user similar to a human expert for the proof shown in Task 1. The message should be a maximum of 15-20 words long and it should explain to `{{user}}` what exact condition led to the final conclusion.

Output **strictly** a JSON object matching this schema:
`{{ TargetMsgOutput.model_json_schema() | tojson }}`

LLM response

`{{JSON_response}}`

Figure 13: Prompt template for One-shot atomic; different colors for all tasks indicate single-turn responses for each independent task.

H Length analysis for Summary and Target User Message tasks

Figures 18 and 19 show this analysis.

I Annotation Guidelines

We provide the annotation guidelines for the proof annotation task in Tables 8 and 9.

System Prompt

You are an expert on Description Logic and DatalogMTL proofs. You are also good at explaining complex things in an easy way.

Task 1: Identify Most Contributing Steps

TASK - FIND THE MOST CONTRIBUTING STEPS
List the steps that contribute the most in deriving the final conclusion (at most 3 steps). Put the exact steps verbatim from the proof below, including conclusion, rule name, and premises.

```
{{proof}}
```

Output **strictly** a JSON object matching this schema:
```
{{ StepsOutput.model_json_schema() | tojson }}
```

LLM response

```
{{JSON_response}}
```

Task 2: Summarize the Proof

TASK - SUMMARIZE PROOF
Summarize the proof below similar to a human expert. The summary should contain the conclusion of the proof and how it was reached.

```
{{proof}}
```

Output **strictly** a JSON object matching this schema:
```
{{ SummaryOutput.model_json_schema() | tojson }}
```

LLM response

```
{{JSON_response}}
```

Task 3: Generate Targeted Message

TASK - TARGET-USER MESSAGE
Generate a message for the targeted user similar to a human expert for the proof given below. The message should be a maximum of 15-20 words long and it should explain to `{{user}}` what exact condition led to the final conclusion.

```
{{proof}}
```

Output **strictly** a JSON object matching this schema:
```
{{ TargetMsgOutput.model_json_schema() | tojson }}
```

LLM response

```
{{JSON_response}}
```

Figure 14: Prompt template for Zero-shot atomic; different colors for all tasks indicate single-turn responses for each independent task.

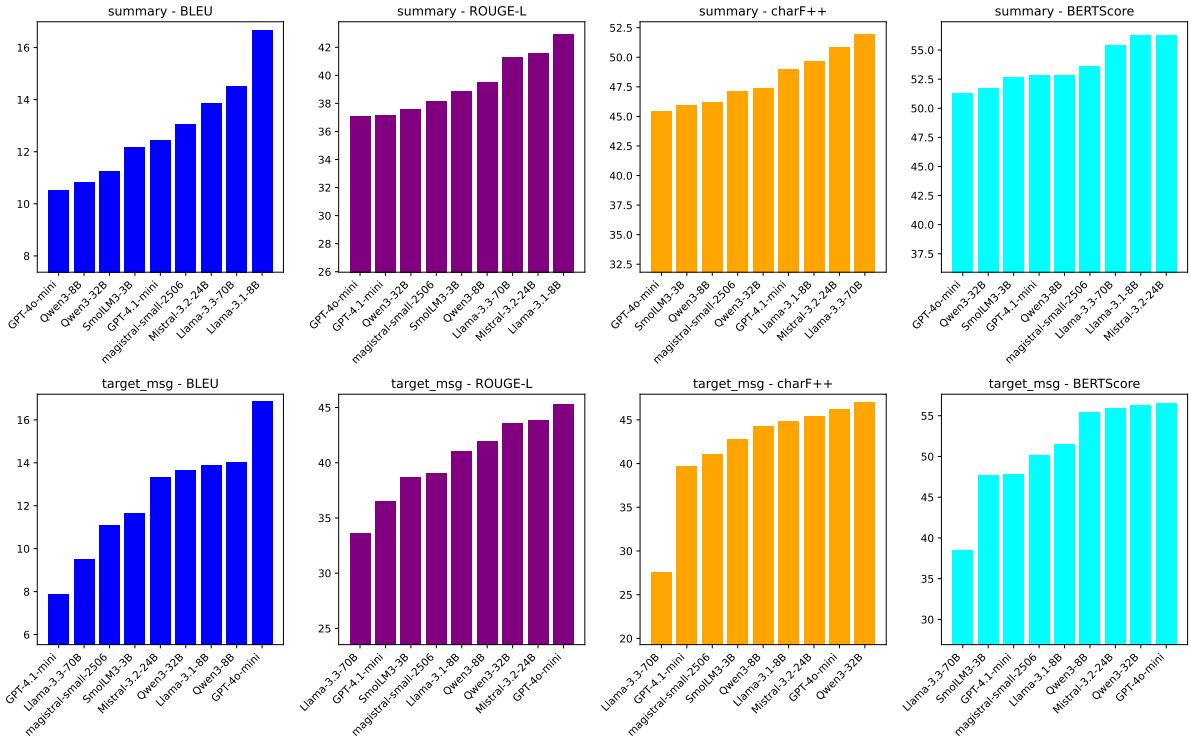

Figure 15: Automated metrics of One-shot atomic strategy

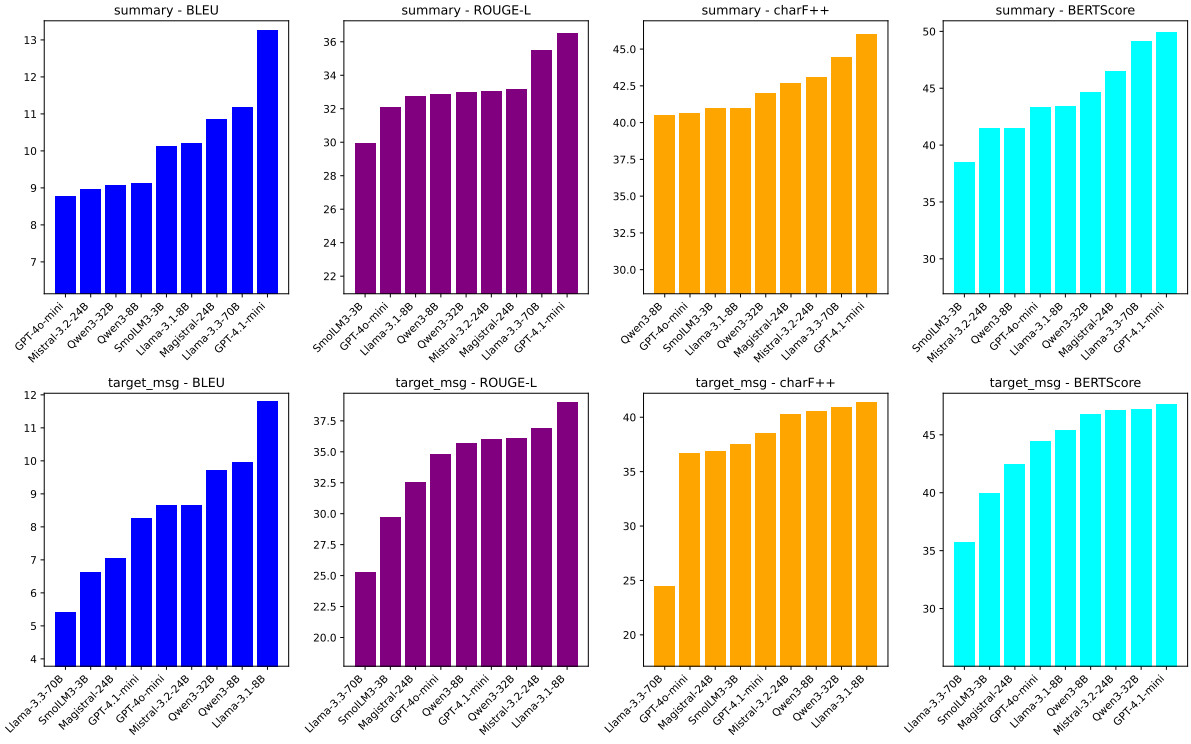

Figure 16: Automated metrics of Zero-shot chained strategy

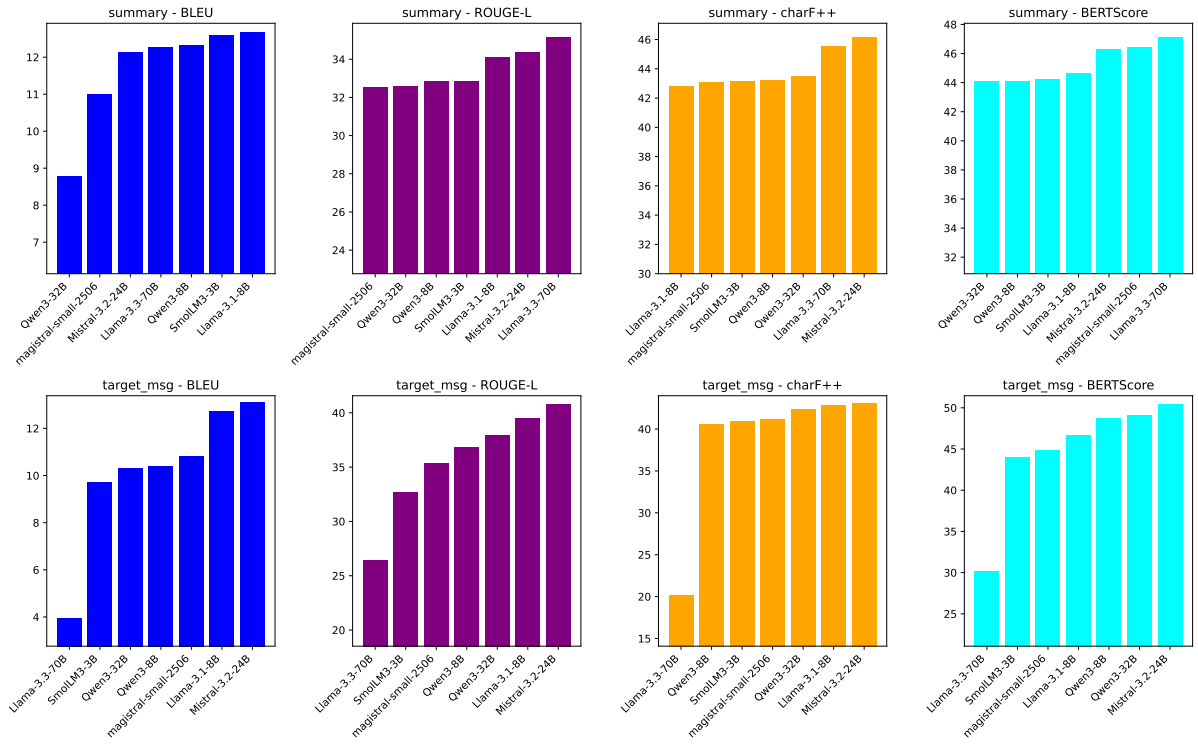

Figure 17: Automated metrics of Zero-shot atomic strategy

Model	Drone	Biology	Recipe
GPT-4.1-mini	0.38	0.29	0.25
GPT-4o-mini	0.11	0.43	0.25
Llama-3.1-8B	0.22	0.33	0.25
Llama-3.3-70B	0.12	0.20	0.26
Magistral-24B	0.21	0.33	0.25
Mistral-3.2-24B	0.12	0.34	0.26
Qwen3-32B	0.17	0.19	0.23
Qwen3-8B	0.12	0.23	0.19
SmolLM3-3B	0.25	0.42	0.18

Table 7: Exact match accuracy for key step selection by domain.

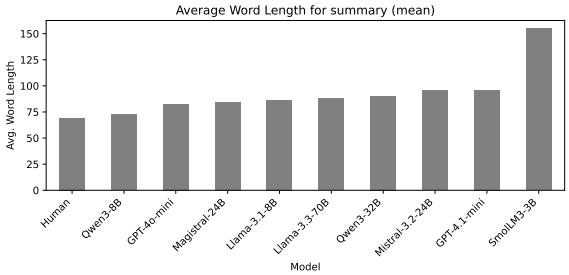

Figure 18: Average length of summary for One-shot chained strategy

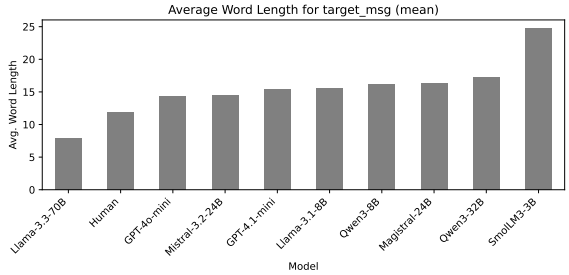

Figure 19: Average length of target user message for One-shot chained strategy

Table 8: Rubrics and Workflow for Evaluating the *Summary*

Input Component	Interpretation	Why It Matters
Description Logic Proof (JSON)	Correct reasoning chain; assume it is the gold standard.	Used to verify whether the Summary faithfully reflects the proof.
Candidate Summary	Model's attempt to compress reasoning into ≈4–5 sentences.	Evaluated for clarity, precision, and faithfulness.

Structure of the Proof: JSON entries link premises via rules in `inferences`. Each step uses asserted or inferred premises, applies a `ruleName`, and yields a conclusion, culminating in the `finalConclusion`.

Field	Max Length	Purpose	Typical Content
Summary	≈ 4–5 sentences	Capture reasoning and conclusion faithfully.	Key conclusions, intermediate reasoning, essential facts.

Rating Rubrics (5-point scale):

- **Faithfulness** – Must directly align with the proof.
- **Readability** – Clear structure, appropriate tone.
- **Conciseness** – No unnecessary content.
- **Coverage** – Includes all essential reasoning steps.

Summary Scoring Guidelines

Score	Faithfulness	Readability	Conciseness	Coverage
5	Fully supported by proof	Flawless writing	Essential info only	All key steps included
4	Minor paraphrases ($\geq 95\%$)	Very clear	Slight redundancy	Misses one trivial step
3	Several weakly supported statements	Understandable	Some extra phrasing	Omits $\geq$ 2 secondary steps
2	Misstates key facts	Hard to follow	Verbose	Omits a critical step
1	Major contradictions	Incoherent	Irrelevant or long	Misses main conclusion

Summary Annotation Workflow:

1. Read the proof; determine the main conclusion and key steps.
2. Evaluate the Summary under all rubric criteria.
3. Ensure accuracy and completeness (no contradictions).
4. Click "Save all" before moving on.

Table 9: Rubrics and Workflow for Evaluating the *Target Message*

Input Component	Interpretation	Why It Matters
Description Logic Proof (JSON)	Ground-truth reasoning and final conclusion.	Ensures the target message does not misrepresent or exceed the proof.
Target Message	$\geq$ 20-word alert to the user.	Must be precise, actionable, and fully supported.

Structure of the Proof: JSON inference steps derive conclusions from premises; the Target Message should reflect only the core actionable outcome.

Field	Max Length	Purpose	Typical Content
Target Message	$\geq$ 20-words	Deliver a concise user-facing alert.	Trigger condition, effect, or simple instruction.

Rating Rubrics (5-point scale):

- **Faithfulness** – Must follow directly from the proof.
- **Appropriateness** – Clear, safe, user-oriented.
- **Coverage** – Includes the essential actionable element.

Target Message Scoring Guidelines

Score	Faithfulness	Appropriateness	Coverage
5	Fully justified by proof	Perfect tone; no overreach	All essential actionable info
4	Minor paraphrase	Very good tone; small phrasing issue	One trivial omission / addition
3	Weak support here or there	Understandable; awkward	Missing $\geq$ 1 important detail
2	Misstates crucial fact	Hard to interpret safely	Misses core actionable element
1	Contradiction / hallucination	Inappropriate or confusing	Not based on proof

Target Message Annotation Workflow:

1. Identify the final actionable conclusion in the proof.
2. Check the Target Message for faithfulness and safety.
3. Ensure brevity and clarity within the 20-word limit.
4. Click “Save all” before moving on.