Decompose-and-Formalise: Recursively Verifiable Natural Language Inference

Anonymous ACL submission

Abstract

Recent work has shown that integrating large language models (LLMs) with theorem provers (TPs) in neuro-symbolic pipelines helps with autoformalisation, entailment verification, and proof-guided refinement of explanations for natural language inference (NLI). However, scaling such refinement to naturalistic NLI remains difficult: long, syntactically rich inputs and deep multi-step arguments amplify autoformalisation errors, where a single local mismatch can invalidate the proof. Moreover, current methods often handle failures via costly global regeneration due to the difficulty of localising the responsible span or step from prover diagnostics. Aiming to address these problems, we propose a decompose-and-formalise framework that (i) decomposes premise-hypothesis pairs into an entailment tree of atomic steps, (ii) verifies the tree bottom-up to isolate failures to specific nodes, and (iii) performs local diagnostic-guided refinement instead of regenerating the whole explanation. Moreover, to improve faithfulness of autoformalisation, we introduce θ-substitution in an event-based logical form to enforce consistent argument–role bindings. Across a range of reasoning tasks using five LLM backbones, our method achieves the highest verified explanation rates, improving over the state-of-the-art by 26.2%, 21.7%, 21.6% and 48.9%, while reducing refinement iterations and runtime while preserving strong end-NLI accuracy.

1 Introduction

Natural language explanations are the workhorse of human communication and rationality, yet they are frequently incomplete, imprecise, and logically fragile (Valentino and Freitas, 2024; Valentino et al., 2021). This motivates verifiable Natural Language Inference (NLI) systems that can make the implicit commitments of everyday explanations explicit, and refine them into solver-checkable stepwise certificates (Quan et al., 2024; Sadeddine and Suchanek, 2025). Recent works in neuro-symbolic pipelines offer a principled route to this goal by integrating Large Language Models (LLMs) with external theorem provers (TPs): LLMs autoformalise natural language statements into formal representations, and TPs then verify whether premises and explanations entail a hypothesis in the target logic (Pan et al., 2023; Olausson et al., 2023; Quan et al., 2025a; Wu et al., 2022; Xu et al., 2024, 2025c; Zhang et al., 2025; Jiang et al., 2024). TP feedback can further support iterative refinement of both formalisation and explanations (Quan et al., 2024, 2025b; Xu et al., 2025b).

Despite the pace of progress, scaling formal refinement to naturalistic and complex NLI remains an open problem. Real explanations often involve long, syntactically rich sentences and deep multi-step arguments that compress many coupled semantic commitments into a small surface form. A single local mismatch (such as a role swap, a negation-scope error, or a quantifier mistake) can derail an entire reasoning chain. Moreover, because these errors propagate across steps, end-to-end regeneration and re-proving becomes increasingly unstable and costly as chains become longer and more branched (Pan et al., 2023; Quan et al., 2025b). At the same time, autoformalisation must be both prover-compatible and semantically faithful (preserving scope, roles, quantification, and lexical commitments) because even minor semantic drift can yield proofs that are syntactically valid but misaligned with the original text (Jiang et al., 2024; Xu et al., 2025b; Quan et al., 2025b). These bottlenecks foreground three questions: *RQ1: How can we obtain autoformalisation that is both prover-compatible and semantically faithful? RQ2: What kind of verification strategy provides transparent, stepwise verification that preserves multi-step structure and reduces refinement*

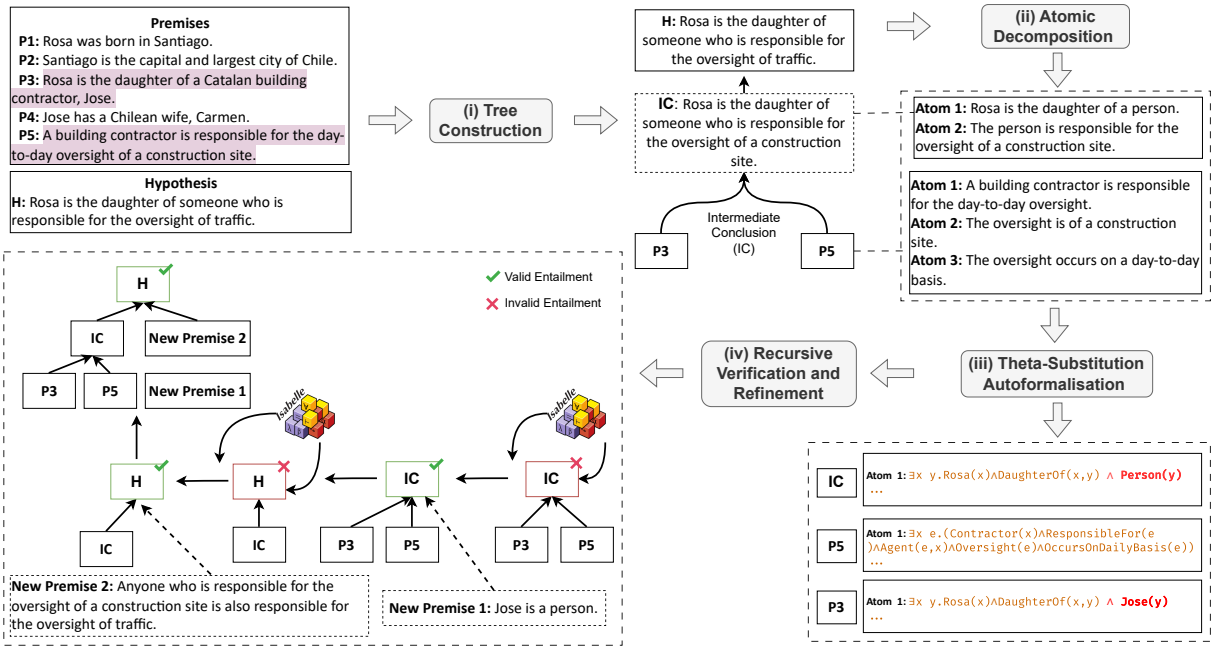


Figure 1: An illustration of the proposed framework for recursively verifiable natural language inference (NLI) via entailment trees, atomic decomposition, and θ-substitution autoformalisation. The entailment tree is initially constructed from the given premises and hypothesis, including intermediate conclusions. Each sentence is then decomposed into atomic propositions and autoformalised into logical forms. By utilising an external theorem prover, we progressively verify and refine each subtree until the final hypothesis node is reached. The reasoning chain is considered logically sound and coherent once the entire entailment tree has been fully verified and refined.

drift? RQ3: To what extent can local diagnostic-guided refinement improve efficiency and robustness without sacrificing end-task accuracy?

To address these challenges, we propose *LLM-TP Tree*, a scalable decompose-and-formalise framework built around an entailment-tree view of explanations. The core idea is to replace holistic refinement with structured decomposition and local repair. Concretely, LLM-TP Tree (i) decomposes each premise–hypothesis instance into an entailment tree of atomic steps, (ii) verifies the tree bottom-up to isolate proof failures to specific nodes, and (iii) performs local diagnostic-guided refinement on the failing nodes rather than regenerating the entire explanation. This design makes semantic preservation controllable at the smallest meaningful granularity, stabilises refinement under longer-horizon reasoning, and improves efficiency by focusing computation where proofs actually fail. To improve faithfulness in autoformalisation, we introduce a θ-substitution procedure within an event-based logical form that enforces consistent argument-role bindings during autoformalisation, making role commitments explicit and systematically repairable. Together, atomic decomposition, recursive bottom-up verification, and θ-substitution yield prover-compatible theories that better track the intended semantics of the original text while enabling targeted refinement.

We evaluate LLM-TP Tree on FOLIO, ProofWriter, PrOntoQA, and EntailmentBank using five LLM backbones (GPT-4o, GPT-5 nano, Grok-4-fast, Deepseek-V3.1 and Qwen3-Max). We compare against state-of-the-art baselines including Explanation-Refiner (Quan et al., 2024), Faithful-Refiner (Quan et al., 2025b), LINC (Olausson et al., 2023), and Logic-LM (Pan et al., 2023). Across datasets, our approach achieves the strongest refinement performance, improving the average verification rate by 26.2%, 21.7%, 21.6%, and 48.9% points on FOLIO, ProofWriter, PrOntoQA, and EntailmentBank, respectively, while reducing refinement iterations and runtime and maintaining strong NLI accuracy. In summary, our contributions are:

1. We introduce LLM-TP Tree, a recursive entailment-tree framework that verifies and refines multi-step inference chains for verifiable NLI via bottom-up checking and local repair.

2. We propose atomic decomposition, which decomposes complex sentences into atomic propositions and reduces inference-time by

16.5%, 13.7%, 21.1%, and 10.1% across datasets compared to baseline neuro-symbolic frameworks.

3. We propose a θ-substitution autoformalisation procedure that enforces consistent argument-role bindings and improves autoformalisation faithfulness by 0.09, 0.106, 0.128, and 0.21 on average over baseline models.

4. We conduct comprehensive automatic and human evaluations with ablations, showing that LLM-TP Tree substantially improves the robustness and efficiency of LLM-TP explanation refinement while preserving strong logical reasoning performance.

2 Recursively Verifiable NLI

In this work, we define an NLI instance i with a premise set $P_i = \{p_1, p_2, \ldots, p_n\}$ and a hypothesis h_i to be formally verifiable if the entailment condition can be verified by a theorem prover. Let $\Phi : \mathrm{NL} \to \mathcal{L}$ translate natural language sentences into a target logic $\mathcal{L}$, and let $\mathcal{S}$ be a solver for $\mathcal{L}$. The NLI instance i is labelled as entailment if $\mathcal{S}(\Phi(P_i) \vdash \Phi(h_i))$ is provable, and as contradiction if $\mathcal{S}(\Phi(P_i) \vdash \neg\Phi(h_i))$ is provable.

Formal verifiability certifies only the endpoint relation between P_i and h_i. We call an instance recursively verifiable if there exists an explicit inference chain $\Pi = \langle \delta_1, \delta_2, \ldots, \delta_k \rangle$ whose intermediate steps are individually solver-checkable and compose to h_i. Let $\Pi^{<j} = \langle \delta_1, \ldots, \delta_{j-1} \rangle$ denote the prefix before step j. Then Π is a recursively verifiable witness if, for all $j \in \{1, \ldots, k\}$, $\mathcal{S}\big(\Phi(P_i \cup \Pi^{<j}) \vdash \Phi(\delta_j)\big)$ is provable, and $\mathcal{S}\big(\Phi(P_i \cup \Pi) \vdash \Phi(h_i)\big)$ is provable.

3 Methodology

To enable faithful formal checking of multi-step reasoning chains and to refine them when proof obligations are not discharged, we propose LLM-TP Tree, a neuro symbolic framework for recursively verifiable NLI based on an entailment reasoning tree T_i. Figure 1 provides an overview.

3.1 Entailment Tree Construction

Given a premise set P_i and a hypothesis h_i, we first prompt an LLM to construct an entailment tree and a set of intermediate conclusions $C = \{\delta_1, \delta_2, \ldots, \delta_m\}$, optionally selecting a subset of premises $P_i' \subseteq P_i$, such that $P_i' \cup C \models h_i$ holds. The tree T_i consists of nodes labelled by propositions (premises or intermediate conclusions) and directed edges encoding entailment dependencies. For each internal node $v \in T_i$ with proposition φ_v and descendant nodes $\mathcal{D}(v)$, the corresponding local entailment obligation requires that the conjunction of descendants entails the parent, namely $\bigwedge_{u \in \mathcal{D}(v)} \varphi_u \models \varphi_v$. The prompt of the entailment tree construction is in Appendix J.

3.2 Atomic Decomposition

Sentence complexity is a major contributor to both the latency and the brittleness of autoformalisation (Quan et al., 2025b). Long and structurally complex sentences tend to increase the search space of the translation process, which not only slows down inference but also amplifies semantic inconsistencies in the resulting logical forms. To mitigate these issues, we then decompose each natural language proposition into a set of atomic propositions, so that the subsequent autoformalisation operates on shorter, semantically focused units.

After constructing the entailment reasoning tree, we apply atomic decomposition to the propositions (first subtree) that participate in proof obligations. For the hypothesis h_i, we generate a set of atoms

$$D(h_i) = \{a_1^h, a_2^h, \ldots, a_k^h\}, \qquad h_i \equiv \bigwedge_{\ell=1}^{k} a_\ell^h. \tag{1}$$

Similarly, for the premise set $P_i = \{p_1, \ldots, p_n\}$, each premise p_j is decomposed into

$$D(p_j) = \{a_1^{p_j}, a_2^{p_j}, \ldots, a_{t_j}^{p_j}\}, \qquad p_j \equiv \bigwedge_{\ell=1}^{t_j} a_\ell^{p_j}. \tag{2}$$

We define the global atom set induced by the premises as

$$D(P_i) := \bigcup_{j=1}^{n} D(p_j), \qquad P_i \equiv \bigwedge_{a \in D(P_i)} a. \tag{3}$$

Entailment-preserving requirement. We require the decomposition operator $D(\cdot)$ to be entailment-preserving: for any sentence φ and any atom $a \in D(\varphi)$, it must hold that $\varphi \models a$. Intuitively, each atom is a logically entailed consequence of the original sentence, so decomposition does not introduce new information that is not supported by the source. We obtain candidate

atoms by prompting an LLM (see Appendix I). To enforce the entailment-preserving requirement in practice, we apply a pretrained NLI classifier (`roberta-large-mnli` (Liu et al., 2019)) to verify that each atom is entailed by its source sentence within a predefined threshold (0.9). This filtering step reduces invalid decompositions and improves the stability of downstream autoformalisation.

3.3 θ-substitution Autoformalisation

Existing autoformalisation pipelines (Pan et al., 2023; Olausson et al., 2023) often rely on complex, monolithic generation procedures to map natural language sentences into fully specified logical formulas. This design is computationally inefficient and, more importantly, prone to systematic errors that frequently prevent theorem provers from certifying entailment. These observations motivate a more structured autoformalisation strategy that decomposes the translation into smaller, verifiable steps and reduces sensitivity to surface linguistic variation.

We then adopt the Neo-Davidsonian event semantics formulation used in Quan et al. (2024), where events and semantic roles (e.g., agent, patient) are represented explicitly in first-order logic to autoformalise the decomposed atomic propositions in previous step to logical forms. Building on this representation, we propose a multi-step θ-substitution procedure that constructs the final logical form by progressively instantiating an abstract template (see Example 2).

A substitution θ is a mapping from placeholders to terms:

$$\theta = \{x_1 \mapsto t_1,\ x_2 \mapsto t_2,\ \ldots,\ x_n \mapsto t_n\}. \quad (9)$$

Applying θ to a formula φ, denoted $\varphi[\theta]$, replaces each placeholder x_i with its corresponding term t_i.

For a natural language sentence, we start from an abstract logical template and apply a sequence of substitutions:

$$\varphi_0 \xrightarrow{\theta_1} \varphi_1 \xrightarrow{\theta_2} \varphi_2 \xrightarrow{\theta_3} \varphi_{\text{final}}. \quad (10)$$

Each substitution handles a specific aspect: 1) *Entity substitution* (θ_1): maps generic predicates to specific entities 2) *Event substitution* (θ_2): introduces event predicates and event structure. 3) *Role substitution* (θ_3): adds Neo-Davidsonian semantic roles and their arguments.

The final formula is obtained by composing the substitutions:

$$\varphi_{\text{final}} = \varphi_0[\theta_1 \circ \theta_2 \circ \theta_3]. \quad (11)$$

Example 2 illustrates the resulting stepwise derivation in detail.

3.4 Entailment tree verification and refinement for recursively verifiable NLI

Finally, given an entailment tree $\mathcal{T}$, we verify it recursively by turning each subtree into a set of local proof obligations that are checkable by a TP. Each node $n \in \mathcal{T}$ stores a natural language statement s_n, its atomic decomposition $D(n) = \{a_{n1}, a_{n2}, \ldots, a_{n|D(n)|}\}$, and a role indicator $\rho(n) \in \{\text{EXPLANATION}, \text{HYPOTHESIS}\}$ that is defined *with respect to the current subtree under verification*. For a subtree $\mathcal{T}_{\text{sub}} \subseteq \mathcal{T}$ with root $h = \text{root}(\mathcal{T}_{\text{sub}})$ and leaf set $E = \text{leaves}(\mathcal{T}_{\text{sub}})$, we set

$$\rho(n) = \begin{cases} \text{EXPLANATION} & \text{if } n \in E, \\ \text{HYPOTHESIS} & \text{if } n = h. \end{cases} \quad (12)$$

From atoms to axioms and lemmas. Let Φ denote autoformalisation from natural language atoms to the target logic. For every explanation node $e \in E$, each atom $a \in D(e)$ is translated into an Isabelle/HOL (Nipkow et al., 2002) axiom (example in Appendix D):

$$\text{axiom}(e, a) :\ \Phi(a). \quad (13)$$

Collecting all explanation atoms yields the axiom set

$$A(E) \ :=\ \bigcup_{e \in E} \{\Phi(a) \mid a \in D(e)\}. \quad (14)$$

For the hypothesis node h, each atom $a \in D(h)$ induces a local proof obligation:

$$\text{lemma}(h, a) :\ A(E) \ \vdash\ \Phi(a). \quad (15)$$

We accept h as verified for the current subtree if TP certifies $\text{lemma}(h, a)$ for all atoms $a \in D(h)$.

Recursive verification with localized refinement. The verification proceeds bottom-up. Intuitively, once a subtree root is verified, it can be treated as a support statement for its parent level. Algorithm 1 summarises the procedure.

When TP cannot discharge a lemma $\text{lemma}(h, a)$ for some atom $a \in D(h)$, we

Example 3.1: θ-substitution Autoformalisation

Sentence: A forest fire would cause deer to die or leave a woodland.

Logic Template Generation:

$$\varphi_0 = \forall x\ y\ z.\ P(x) \wedge Q(y) \wedge R(z) \rightarrow S \tag{4}$$

Step 1: Entity Substitution (Apply $\theta_1 = \{P/\text{ForestFire}, Q/\text{Deer}, R/\text{Woodland}\}$):

$$\varphi_1 = \varphi_0[\theta_1] = \forall x\ y\ z.\ \text{ForestFire}(x) \wedge \text{Deer}(y) \wedge \text{Woodland}(z) \rightarrow S \tag{5}$$

Step 2: Event Substitution (Apply $\theta_2 = \{S/(\text{Die}(e_1) \vee \text{Leave}(e_2))\}$):

$$\varphi_2 = \varphi_1[\theta_2] = \forall x\ y\ z.\ \text{ForestFire}(x) \wedge \text{Deer}(y) \wedge \text{Woodland}(z) \rightarrow (\text{Die}(e_1) \vee \text{Leave}(e_2)) \tag{6}$$

Step 3: Semantic Role Substitution (Apply θ_3):

$$\theta_3 = \begin{cases} \text{Die}(e_1) \mapsto (\text{Die}(e_1) \wedge \text{Agent}(e_1, y)) \\ \text{Leave}(e_2) \mapsto (\text{Leave}(e_2) \wedge \text{Agent}(e_2, y) \wedge \text{Patient}(e_2, z)) \end{cases} \tag{7}$$

Final Result:

$$\begin{aligned} \varphi_{\text{final}} = \varphi_2[\theta_3] = \forall x\ y\ z\ e_1\ e_2.\ &\text{ForestFire}(x) \wedge \text{Deer}(y) \wedge \text{Woodland}(z) \rightarrow \\ &(\text{Die}(e_1) \wedge \text{Agent}(e_1, y)) \vee (\text{Leave}(e_2) \wedge \text{Agent}(e_2, y) \wedge \text{Patient}(e_2, z)) \end{aligned} \tag{8}$$

Figure 2: θ-substitution autoformalisation example.

extract diagnostics from the prover (e.g., the failed goal and relevant constraints) and use them to localise the failure to a small set of implicated explanation nodes $\widehat{E}$. We then prompt an LLM to refine only these implicated nodes, update the subtree accordingly, and re-check the same local obligations. This loop continues until all atoms in $D(h)$ are certified for the subtree root h, after which the verified root is promoted and the algorithm proceeds to higher levels. Repeating this process bottom-up yields a recursively verifiable proof certificate for the entire entailment tree.

4 Experimental Setup

We evaluate our neuro-symbolic framework from two complementary perspectives: (i) explanation refinement apply theorem-prover checking and refine the explanation; and (ii) logical reasoning, which measures standard end-task accuracy to ensure that enhanced verification and refinement do not come at the cost of predictive performance.

Datasets We use four datasets ranging from synthetically generated deductive reasoning benchmarks to expert-written real-world corpora: ProofWriter (Tafjord et al., 2021), PrOntoQA (Saparov and He, 2023), FOLIO (Han et al., 2024), and EntailmentBank (Dalvi et al., 2021). Additional dataset details (i.e. number of samples) are provided in Appendix A.

Baselines and Models For explanation refinement, we compare against two state-of-the-art explanation refinement models, Explanation-Refiner (Quan et al., 2024) and Faithful-Refiner (Quan et al., 2025b). For logical reasoning, we consider two prompting-based baselines, including a direct standard prompting and chain-of-thought (CoT) prompting. In addition, we compare against two representative neuro-symbolic frameworks, Logic-LM (Pan et al., 2023) and LINC (Olausson et al., 2023). We evaluate five distinct LLM backbones, spanning a general-purpose chat model (GPT-4o (OpenAI, 2024)), models evaluated in non-thinking mode (GPT-5 Nano (OpenAI, 2025), Grok-4 Fast (xAI, 2025)), and models evaluated in thinking mode (DeepSeek-V3.1 (Deepseek-AI, 2025), Qwen3-Max (Qwen Team, 2025)) and additionally include GPT-3.5 in logical reasoning tasks to match the model coverage reported in prior work. LLM implementation details is described in Appendix D.

5 Empirical Results and Evaluation

LLM-TP Tree effectively to construct and refines verifiable explanation-based NLI. Table 1 reports the main results on the explanation refinement task. Across all datasets and all LLM backbones, LLM-TP Tree achieves the best *Fin.* performance, indicating that it is consistently more effective at producing a checkable witness for explanation-based NLI. Averaged over backbones, LLM-TP Tree improves the final verified rate by 26.23%, 21.73%, 21.60%, and 48.93% on each dataset compared to Explanation-Refiner. The ef-

Dataset	Approach	GPT-4o		GPT-5 nano		Grok-4 fast		Deepseek-V3.1		Qwen3-max	
		Init.	*Fin.*	*Init.*	*Fin.*	*Init.*	*Fin.*	*Init.*	*Fin.*	*Init.*	*Fin.*
FOLIO	Explanation-Refiner	53.28	63.93	46.72	59.84	45.08	52.46	50.82	62.30	55.74	68.85
	Faithful-Refiner	68.03	78.69	53.28	68.03	54.92	65.57	71.31	81.15	69.67	84.43
	LLM-TP Tree	81.15	**90.16** ↑26.23	66.39	**81.15** ↑21.31	62.30	**79.51** ↑27.05	83.61	**92.62** ↑30.32	85.25	**95.08** ↑26.23
ProofWriter	Explanation-Refiner	62.67	76.67	59.33	73.33	53.33	69.33	52.67	66.67	64.00	78.00
	Faithful-Refiner	80.00	86.67	72.00	84.67	60.00	78.67	83.33	90.67	84.00	91.33
	LLM-TP Tree	91.33	**95.33** ↑18.66	85.33	**91.33** ↑18.00	72.67	**90.00** ↑20.67	91.33	**98.00** ↑31.33	92.00	**98.00** ↑20.00
PrOntoQA	Explanation-Refiner	64.00	78.67	60.00	74.67	61.33	73.33	62.00	74.00	64.67	78.00
	Faithful-Refiner	82.67	90.00	73.33	80.00	72.00	84.00	85.33	91.33	82.00	91.33
	LLM-TP Tree	92.00	**97.33** ↑18.66	93.33	**98.00** ↑23.33	86.00	**91.33** ↑18.00	97.33	**100.00** ↑26.00	98.00	**100.00** ↑22.00
EntailmentBank	Explanation-Refiner	15.33	26.67	12.00	23.33	8.67	13.33	15.33	26.67	16.67	28.00
	Faithful-Refiner	23.33	44.00	17.33	47.33	14.00	46.00	25.33	48.00	22.00	52.00
	LLM-TP Tree	21.33	**70.00** ↑43.33	18.00	**74.00** ↑50.67	16.67	**68.67** ↑55.34	22.00	**71.33** ↑44.66	22.67	**78.67** ↑50.67

Table 1: Comparison results on the explanation refinement tasks. *Init.* shows the number of explanation that is initially logically valid. *Fin.* shows the number of explanation that is finally logically valid after refinement. **Bold** values indicate the best performance. Arrows indicate absolute performance gain of LLM-TP Tree over the baseline.

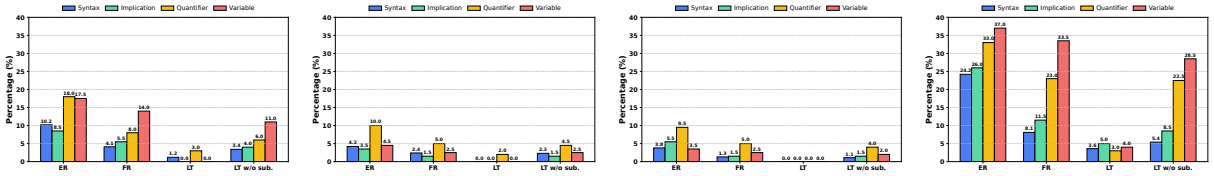
Dataset	ER	FR	LT	LT w/o AD
FOLIO	4.27	3.75	3.04	3.57
ProofWriter	3.47	3.08	2.42	2.88
PrOntoQA	3.36	2.73	2.04	2.44
EntailmentBank	4.97	4.46	4.04	4.35
Avg.	4.02	3.51	2.88	3.31

Table 2: Average refinement iterations required averaged over five LLM backbones. Comparison between ER (Explanation-Refiner), FR (Faithful-Refiner), LT (LLM-TP Tree) and LT w/o AD (LT without atomic decomposition)

fect is most pronounced on EntailmentBank: while holistic refiners remain below 52% final validity, LLM-TP Tree reaches 68.67–78.67% across backbones. Beyond final refinement quality, the *Init.* column probes whether a backbone can produce an initially provable explanatory chain before any refinement. On synthetical benchmarks such as ProofWriter and PrOntoQA, LLM-TP Tree yields substantially higher initial validity than holistic refiners; for instance, on ProofWriter with Qwen3-max, LLM-TP Tree attains 92.00% *Init.* versus 64.00% for Explanation-Refiner. Notably, EntailmentBank exhibits low initial validity across methods, suggesting that constructing globally coherent verifiable chains is difficult under noisier multi-hop explanations; in this regime, our proposed mechanism yields the largest gains after refinement.

Atomic decomposition improves the efficiency of verification-driven explanation refinement. We report the average number of refinement iterations, averaged across different LLM backbones, required to refine an explanation for each dataset, as shown in Table 2. Across all four datasets (FOLIO, ProofWriter, PrOntoQA, and EntailmentBank), LT converges in fewer iterations than baseline refiners, reducing the required iterations by an average of 1.14 relative to ER and by an average of 0.63 relative to Faithful-Refiner (FR). To isolate the advantages of atomic decomposition, we run an ablation that disables atomic sentence splitting. Relative to the full LT system, removing atomic decomposition increases the number of required refinement iterations by between 0.31 and 0.53 . This ablation indicates that atomic decomposition contributes materially to LT's efficiency gains: by splitting premises and intermediate conclusions into atomic subclaims, verification failures can be localised to a small set of fine-grained hypotheses, enabling targeted repairs of the minimal logical mismatch and thereby reducing both the number of refinement iterations and the cost of each verification/refinement pass. We also report the mean running time per refinement iteration (including theorem-proving time) under different LLM backbones, and provide the full per-backbone iteration results in Appendix F.

Multi-step θ-substitution autoformalisation improves semantic faithfulness. A faithful autoformalisation should preserve the semantic content of the original sentence as much as possible. Fol-

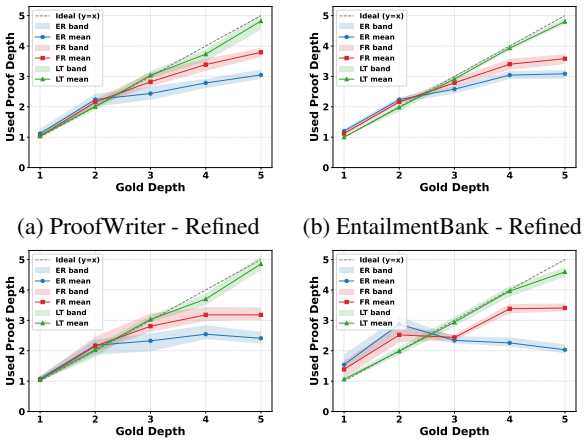


(a) FOLIO (b) ProofWriter (c) PrOntoQA (d) EntailmentBank

Figure 3: Distribution of the autoformlisation errors (syntax, implication, quantifier, variable in Isabelle/HOL Theories.

Dataset	ER	FR	LT	LT w/o sub.
FOLIO	0.766	0.822	0.856	0.828
ProofWriter	0.834	0.892	0.940	0.914
PrOntoQA	0.824	0.918	0.952	0.924
EntailmentBank	0.614	0.726	0.824	0.770
Avg.	0.760	0.840	0.893	0.859

Table 3: Average autoformalisation faithfulness (cosine similarity) averaged over five LLM backbones. Comparison between ER (Explanation-Refiner), FR (Faithful-Refiner), LT (LLM-TP Tree) and LT w/o sub. (LT without θ-substitution autoformalisation).



(a) ProofWriter - Refined (b) EntailmentBank - Refined

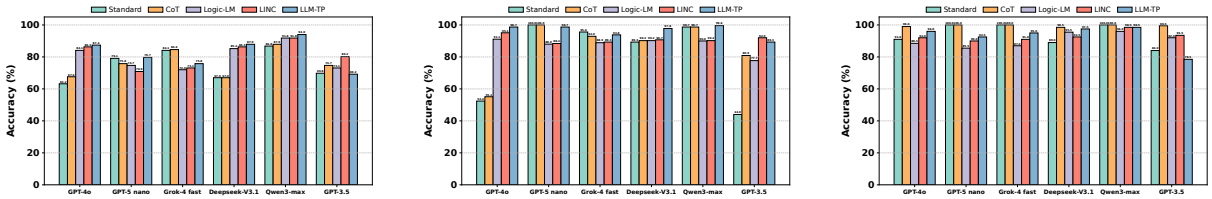
(c) ProofWriter - Unrefined (d) EntailmentBank - Unrefined

Figure 4: Proof depth alignment between gold and actual constructed proof depths across frameworks. The solid curve shows the mean used depth averaged over five LLMs, while the shaded band shows the range across backbones. Top: Trends in refined cases. Bottom: Trends in unrefined cases.

lowing the rule-based informalisation procedure by Quan et al. (2025b), we quantify faithfulness by converting each predicted logical form back into natural language and computing the cosine similarity between the informalised sentence and the original sentence. Table 3 summarises the results. Our approach achieves the highest average faithfulness across all datasets, outperforming Explanation-Refiner by an average of 0.13 and Faithful-Refiner by an average of 0.053. As expected, faithfulness is generally higher on synthetic benchmarks, whose language more closely follows rule-based templates, whereas EntailmentBank is more challenging due to its more naturalistic multi-hop explanations. Notably, our method yields the largest margin on EntailmentBank, indicating stronger robustness of θ-substitution autoformalisation under distributional and linguistic variability. We further perform an ablation by removing θ-substitution (LT w/o sub.). This variant shows a consistent drop in similarity of an average of 0.034 across the same datasets compared to the full model. We also shows the full ablation study results in Appendix E

5.1 Failure and Error Analysis

We analyse refinement drift using proof-depth alignment. Figure 4 measure proof-depth alignment by comparing the TP-extracted used proof depth of each explanation with the dataset-provided gold depth (ideal $y{=}x$). In refined cases, all frameworks show a positive trend that is close to the ideal $y{=}x$ line. By contrast, in unrefined cases, ER and FR exhibit a clear refinement drift in the form of depth compression: the used depth grows slowly, saturates, or even decreases, increasingly undershooting the gold depth. This indicates that holistic refiners tend to collapse longer multi-step explanations into shorter proofs. In contrast, LLM-TP Tree tracks the gold depth much more closely suggesting that its refinement process better preserves proof-step granularity in longer-horizon inference. Detailed distributions are provided in Appendix H. We further analyse autoformalisation failures by categorising errors in the generated Isabelle/HOL theories into four types (details in Appendix B) across 200 sampled theories. Figure 3 shows that LLM-TP Tree consistently yields the lowest error rates across datasets, with the largest margin on the more naturalistic EntailmentBank. Removing θ-substitution (LT w/o sub.) leads to a clear increase in semantic and structural errors, particularly quan-



(a) FOLIO (b) ProofWriter (c) PrOntoQA

Figure 5: Comparison on the logical reasoning accuracy tasks.

tifier and variable errors on EntailmentBank, highlighting the importance of staged substitution autoformalisation for maintaining a prover-compatible logical structure.

5.2 Logical Reasoning

Figure 5 reports accuracy on three logical reasoning benchmarks. Across all the datasets and LLMs, LLM-TP Tree is the strongest neuro-symbolic method: compared against Logic-LM and LINC, it achieves higher accuracy. The improvements are substantial on the more challenging datasets. These results indicate that explicit theorem-proving with a tree-structured search provides a reliable verification signal and reduces errors relative to existing neuro-symbolic pipelines, as long as the backbone model can produce sufficiently correct autoformalisation. However, Neuro-symbolic approaches sometimes perform worse than standard prompting. Despite its advantage over other baselines, LLM-TP Tree is not universally better than direct prompting. In particular, on ProntoQA Standard/-CoT prompting are consistently strongest across all backbones, leaving limited headroom for theorem-proving. This behaviour is consistent with two factors: (i) some datasets are already near-saturated by strong classifiers and CoT prompting, so additional symbolic steps cannot provide much benefit; (ii) neuro-symbolic methods introduce an additional failure mode (autoformalisation errors). When autoformalisation produces an incomplete or incorrect theory, the prover cannot establish either label and the system defaults to Unknown, directly harming accuracy. This is most evident with GPT-3.5, where LLM-TP Tree drops below LINC on all three datasets.

6 Related Work

Neuro-symbolic pipelines have shown strong effectiveness on multi-hop reasoning, as they combine learned language understanding with symbolic reasoning to support complex, multi-step inference. Early work such as Weber et al. (2019) integrates a symbolic reasoner (Prolog) with a rule-learning component to solve multi-hop reasoning problems. More recently, researchers have increasingly combined LLMs with symbolic solvers across a range of logical reasoning tasks. For example, Pan et al. (2023) and Olausson et al. (2023) autoformalise natural-language contexts into symbolic representations and then apply external solvers to perform deductive reasoning over the resulting formal structures. Along a related direction, Xu et al. (2025a) proposes a decompose-search-resolve framework that breaks a problem into smaller logical substructures and performs structured search to resolve each reasoning step. Prover feedback has also been used to improve reliability: Quan et al. (2025a) extracts erroneous proof steps from solver feedback and iteratively refines reasoning in a loop. In contrast to endpoint-focused verification, our work targets recursively verifiable NLI by enforcing chain-level checking and localised refinement over structured entailment trees.

7 Conclusion

We presented LLM-TP Tree, a neuro-symbolic framework for recursively verifiable NLI that follows a decompose-and-formalise paradigm. By combining entailment-tree verification with entailment-preserving atomic decomposition and θ-substitution autoformalisation, our method enables localised, refinement that targets the implicated part of an explanation rather than regenerating it holistically. Experiments on four datasets with multiple LLM backbones show improved TP-verified explanation rates, more faithful autoformalisation, and reduced refinement drift while maintaining strong end-task reasoning accuracy. In future work, we will extend this framework to broader naturalistic settings, improve the robustness of entailment-tree construction and diagnostic localisation, and further reduce verification cost through better reuse of verified subtrees.

Limitations

Our framework produces solver-checkable certificates, but the guarantee is necessarily conditional on the chosen autoformalisation function Φ, the target formalism, and the capabilities of the underlying theorem prover. In particular, a verification can be valid in the induced formal theory while still being partially misaligned with the original natural language intent if Φ introduces subtle scope, polarity, or role mismatches that are not exposed by our current faithfulness checks. This dependence is amplified in naturalistic NLI, where missing background knowledge, implicit commonsense assumptions, or discourse phenomena (e.g., coreference, modality, temporality) may not be representable in the event-based logical form, making some instances difficult to certify without additional axioms or richer semantics. The effectiveness of localised refinement also depends on the quality of the intermediate structures proposed by the LLM and on the ability to map prover diagnostics back to the responsible region. When the initial entailment tree omits critical intermediate conclusions or introduces a misleading proof plan, bottom-up checking can localise which obligations are not discharged but may not reliably recover the globally correct chain without substantial restructuring. Finally, although our approach reduces expensive global regeneration, it still incurs non-trivial compute due to repeated LLM calls, repeated autoformalisation, and repeated prover invocations. Prover timeouts or inconclusive outcomes can remain a practical bottleneck on harder instances, and our current system does not provide a formal convergence guarantee to the minimal or unique refined explanation.

References

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. *EMNLP*.

Deepseek-AI. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2025. Introducing gpt-5. `https://openai.com/index/introducing-gpt-5/`.

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

Xin Quan, Marco Valentino, Danilo Carvalho, Dhairya Dalal, and Andre Freitas. 2025a. PEIRCE: Unifying material and formal reasoning via LLM-driven neuro-symbolic refinement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 11–21, Vienna, Austria. Association for Computational Linguistics.

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

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. 2025b. Faithful and robust LLM-driven theorem proving for NLI explanations. In *Proceedings of the 63rd Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 17734–17755, Vienna, Austria. Association for Computational Linguistics.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zacchary Sadeddine and Fabian M. Suchanek. 2025. Verifying the steps of deductive reasoning chains. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 456–475, Vienna, Austria. Association for Computational Linguistics.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

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

Neha Srikanth and Rachel Rudinger. 2025. NLI under the microscope: What atomic hypothesis decomposition reveals. 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 2574–2589, Albuquerque, New Mexico. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Marco Valentino and André Freitas. 2024. On the nature of explanation: An epistemological-linguistic perspective for explanation-based natural language inference. *Philosophy & Technology*, 37(3):88.

Marco Valentino, Ian Pratt-Hartman, and André Freitas. 2021. Do natural language explanations represent valid logical arguments? verifying entailment in explainable nli gold standards. *IWCS 2021*, page 76.

Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. 2019. NLProlog: Reasoning with weak unification for question answering in natural language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6151–6161, Florence, Italy. Association for Computational Linguistics.

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

xAI. 2025. Grok 4. `https://x.ai/news/grok-4`.

Jundong Xu, Hao Fei, Meng Luo, Qian Liu, Liangming Pan, William Yang Wang, Preslav Nakov, Mong-Li Lee, and Wynne Hsu. 2025a. Aristotle: Mastering logical reasoning with a logic-complete decompose-search-resolve framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3075, Vienna, Austria. Association for Computational Linguistics.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13326–13365, Bangkok, Thailand. Association for Computational Linguistics.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. 2025b. Training llms with logicreward for faithful and rigorous reasoning. *Preprint*, arXiv:2512.18196.

Lei Xu, Pierre Beckmann, Marco Valentino, and André Freitas. 2025c. Adaptive llm-symbolic reasoning via dynamic logical solver composition. *Preprint*, arXiv:2510.06774.

Lan Zhang, Marco Valentino, and Andre Freitas. 2025. Autoformalization in the wild: Assessing LLMs on real-world mathematical definitions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1720–1738, Suzhou, China. Association for Computational Linguistics.

A Dataset Details

A.1 Explanation Refinement

ProofWriter. ProofWriter (Tafjord et al., 2021) is a synthetically generated deductive reasoning benchmark where each instance contains a natural language theory (facts and rules), a query, and a gold truth label, with controllable multi-hop proof depth. For explanation refinement, we sample 150 instances and stratify the sampling to cover a range of gold proof depths $d \in \{1, 2, 3, 4, 5\}$ on the ProofWriter D5 split under the open-world assumption. To avoid ambiguity in verification-oriented refinement, we restrict to boolean-labelled instances and exclude indeterminate cases (e.g., `Unknown` under open-world settings). Each instance is cast into an NLI-style pair (P, h), where P is the provided set of facts/rules and h is the hypothesis statement.

PrOntoQA. PrOntoQA (Saparov and He, 2023) is a synthetic dataset designed to evaluate deductive reasoning in controlled settings. Following the Logic-LM (Pan et al., 2023) setup, we use the most

challenging five-hop subset (fictional-entity reasoning). For explanation refinement, we randomly sample 150 instances from this subset. Since this subset is binary in nature, we keep boolean-labelled instances and cast each instance into the NLI-style format (P, h).

FOLIO. FOLIO (Han et al., 2024) is a human-curated first-order logic reasoning benchmark, pairing natural-language statements with formal logical structure to support theorem-prover-based verification. For explanation refinement, we use the evaluation split commonly adopted by prior neuro-symbolic work after preprocessing, and remove indeterminate labels to keep a boolean setting. Concretely, we start from 182 evaluation instances after filtering out 22 erroneous examples identified by Olausson et al. (2023) and exclude those with uncertain labels, yielding 122 instances. We then treat the remaining instances as NLI pairs (P, h) for verification and refinement.

EntailmentBank. EntailmentBank (Dalvi et al., 2021) is a multi-hop entailment benchmark that provides naturalistic premises and hypotheses together with structured explanation annotations (intermediate conclusions and entailment structure). For explanation refinement, we sample 150 instances and select examples spanning different gold proof depths to ensure coverage of both shallow and longer-horizon multi-step chains. Each instance is cast into (P, h), where P is the set of supporting statements (facts) and h is the hypothesis.

A.2 Logical Reasoning

ProofWriter. For logical reasoning, we evaluate on the ProofWriter D5 split under the open-world assumption to retain the three-way label space (`True/False/Unknown`). We sample 225 instances stratified by gold reasoning depth $d \in \{1, 2, 3, 4, 5\}$, with 45 instances per depth. Within each depth bucket, we balance labels by sampling 15 instances per class.

FOLIO. For logical reasoning, the original validation split contains 204 instances, while we select the 182 evaluation instances after filtering out 22 erroneous examples identified by Olausson et al. (2023) for evaluation. We evaluate on this 182-instance set under the dataset's original label space.

PrOntoQA. For logical reasoning, we follow prior work (Pan et al., 2023) and evaluate on the five-hop PrOntoQA subset. We randomly sample 200 instances from this subset (binary labels), which represents the most challenging multi-hop configuration in the benchmark.

B Autoformalisation Errors

Following the error taxonomy in Quan et al. (2025b), we categorise autoformalisation errors into four broad types: *syntax*, *implication*, *quantifier*, and *variable* errors. We define an autoformalisation error as any mismatch between the intended logical meaning of the input sentence(s) (as captured by a reference logic template or gold formalisation) and the produced formal statement. Notably, these categories differ in how easily they can be detected: syntax errors are often surfaced immediately by the theorem prover, whereas the remaining three categories typically produce well-typed, well-formed formulas that may still be semantically unfaithful and therefore require semantic-level checking.

Syntax errors. Syntax errors are those that prevent the proof assistant from parsing or type-checking the generated statement, and are therefore usually the easiest to detect. Typical causes include missing brackets, malformed binder syntax, incorrect use of infix operators, and type unification conflicts (e.g., using an event-typed variable where an entity is expected). Such errors are often directly indicated by the theorem prover's feedback (parse errors, unknown constants, failed unification, or unsatisfied type-class constraints), making them amenable to automatic filtering.

Implication errors. Implication errors occur when the generated formula has a valid syntax and type, but its *logical structure* encodes the wrong entailment relation compared to the intended template. This includes incorrect direction of implication, incorrect connective choice (e.g., $\wedge$ vs. $\rightarrow$), or wrong nesting/precedence that changes the meaning. These errors are difficult for theorem provers to flag because the statement may still be perfectly consistent and provable under some background axioms, while representing a different claim.

Quantifier errors. Quantifier errors arise when the autoformalisation introduces incorrect quantifiers ($\forall$ vs. $\exists$), incorrect quantifier scope, or incorrect ordering of quantifiers. These errors are especially harmful because they often produce statements that are strictly stronger or weaker than in-

Algorithm 1 VerifySubtree($\mathcal{T}_{\text{sub}}$, TP)

$h \leftarrow \text{root}(\mathcal{T}_{\text{sub}})$ {current hypothesis node}
$E \leftarrow \text{leaves}(\mathcal{T}_{\text{sub}})$ {current explanation nodes}
Construct $A(E) = \bigcup_{e \in E} \{\Phi(a) \mid a \in D(e)\}$ {axioms}
for each atom $a \in D(h)$ **do**
  Define the goal $\tau \leftarrow \Phi(a)$
  if $TP.\text{prove}(A(E) \vdash \tau) = \text{FAIL}$ **then**
    $\text{diag} \leftarrow TP.\text{diagnostics}()$
    Identify implicated explanation nodes $\widehat{E} \subseteq E$ from diag
    Refine $\widehat{E}$ using an LLM conditioned on $(\widehat{E}, \text{diag})$
    Update $\mathcal{T}_{\text{sub}}$; recompute $D(\cdot)$ and re-autoformalise $\Phi(\cdot)$
    return VerifySubtree($\mathcal{T}_{\text{sub}}$, TP) {retry locally}
  end if
end for
Mark h as verified
return h {promote as support for parent level}

Dataset	LT		w/o Atom		w/o θ	
	Init.	*Fin.*	*Init.*	*Fin.*	*Init.*	*Fin.*
Backbone: GPT-4o						
FOLIO	81.15	90.16	71.31 ↓9.84	86.07 ↓4.09	75.41 ↓5.74	88.52 ↓1.64
ProofWriter	91.33	95.33	87.33 ↓4.00	92.00 ↓3.33	89.33 ↓2.00	94.67 ↓0.66
PrOntoQA	92.00	97.33	86.67 ↓5.33	92.67 ↓4.66	90.00 ↓2.00	94.00 ↓3.33
EntailmentBank	21.33	70.00	17.33 ↓4.00	53.33 ↓16.67	19.33 ↓2.00	61.33 ↓8.67
Backbone: Qwen3-max						
FOLIO	85.25	95.08	72.13 ↓13.12	86.89 ↓8.19	79.51 ↓5.74	90.98 ↓4.10
ProofWriter	92.00	98.00	88.00 ↓4.00	93.33 ↓4.67	88.00 ↓4.00	96.00 ↓2.00
PrOntoQA	98.00	100.00	86.67 ↓11.33	94.00 ↓6.00	92.00 ↓6.00	96.00 ↓4.00
EntailmentBank	22.67	78.67	18.00 ↓4.67	61.33 ↓17.34	20.00 ↓2.67	71.33 ↓7.34

Table 4: Ablations of LLM-TP Tree on GPT-4o and Qwen3-max (Init/Fin verified rate, %). Red arrows indicate the absolute decrease compared to LT under the same backbone.

tended while remaining syntactically valid. Moreover, theorem provers typically do not report quantifier mismatches as errors; the formula is well-formed, yet its truth conditions can differ dramatically.

Variable errors. Variable errors occur when logical variables are incorrectly assigned to predicate arguments, resulting in role confusion or broken discourse links. This category covers a wide range of semantic binding mistakes, including swapping semantic roles (e.g., agent vs. patient), incorrectly linking pronouns/coreference, using the wrong event/entity variable in a clause, or mistakenly reusing a variable that should be distinct. Variable errors may be subtle because they can remain type-correct (e.g., swapping two entity variables still type-checks) yet invert the intended meaning.

C Algorithm

D Isabelle/HOL and LLMs Implementation Detail

We use the `isabelle-client` Python package (Shminke, 2022) to run Isabelle/HOL as a real-time server and to retrieve prover responses. We access all LLM backbones via API calls with the following model versions: GPT-4o (`gpt-4o`), GPT-5 nano (`gpt-5-nano`), Grok-4 fast (`grok-4-fast`), Deepseek-V3.1 (`DeepSeek-v3.1-terminus`), and Qwen3-Max (`qwen3-max-preview`). For all non-thinking models, we set the temperature to 0. We also do not limit any thinking effort for Deepseek-V3.1 and Qwen3-Max.

We build the Isabelle/HOL theory following the approach in (Quan et al., 2025b). An example of the constructed Isabelle/HOL theory is shown in Figure 6. We then apply atomic decomposition to each axiom and to the hypothesis. The decomposed hypothesis sentence and the decomposed axioms are then forming a new theory, which we use to prove theorems from each decomposed hypothesis atoms.

E Ablation Study on Explanation Refinement Task

Table 4 reports ablations of LLM-TP Tree (LT) on GPT-4o and Qwen3-max, using *Init./Fin.* logically valid explanation rates. Removing either of the two proposed components, atomic decomposition or multi-step θ-substitution autoformalisation, consistently degrades across all datasets. Across datasets, *Init.* decreases by 4.00 to 13.12 points and *Fin.* decreases by 3.33 to 17.34 points, with the most pronounced degradation on EntailmentBank. Removing θ-substitution also leads to a consistent drop, although smaller than w/o Atom indicating that multi-step substitution contributes materially to producing prover-compatible formalisations.

F Full results: Model efficiency Across LLM Backbones

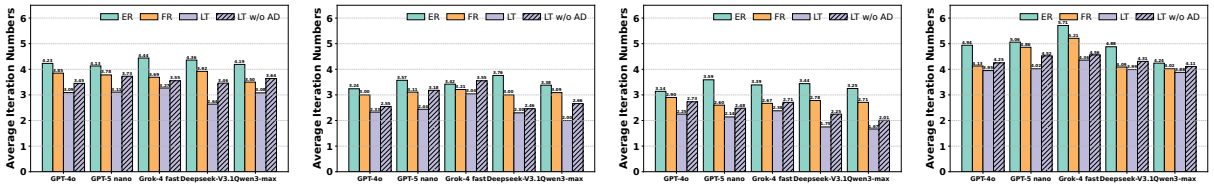
Figure 7a, 7b and 7c shows average number of iterations required to refine an explanation across all LLMs. Figure 7e, 7f, 7g and 8 shows average inference time per iteration in both thinking and non-thinking models.

G Full Results: Autoformalisation Faithfulness Across LLM Backbones

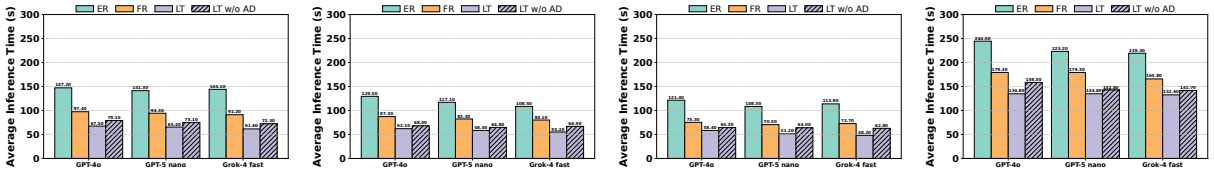
Figure 9 compares the average faithfulness of the autoformalisation process across different ap-

```
theory data_0
imports Main

begin

typedecl entity
typedecl event

consts
  Melting :: "event ⇒ bool"
  Change :: "event ⇒ bool"
  Source :: "event ⇒ entity ⇒ bool"
  Destination :: "event ⇒ entity ⇒ bool"
  Solid :: "entity ⇒ bool"
  Liquid :: "entity ⇒ bool"
  IncreaseHeatEnergy :: "entity ⇒ bool"
  By :: "event ⇒ entity ⇒ bool"
  Chocolate :: "entity ⇒ bool"
  Melts :: "event ⇒ bool"
  Agent :: "event ⇒ entity ⇒ bool"
  In :: "event ⇒ entity ⇒ bool"
  Sunlight :: "entity ⇒ bool"

(* Explanation 1: melting means changing from a solid to a liquid by increasing heat energy *)
axiomatization where
  explanation_1: "∀ e x y z. Melting e ↔ (Change e ∧ Source e x ∧ Destination e y ∧ Solid x ∧
    Liquid y ∧ IncreaseHeatEnergy z ∧ By e z)"

(* Explanation 2: chocolate melts in the sunlight *)
axiomatization where
  explanation_2: "∃ x e. Chocolate x ∧ Melts e ∧ Agent e x ∧ In e Sunlight"

theorem hypothesis:
  assumes asm: "Chocolate x ∧ Solid y ∧ Liquid z"
  (* Hypothesis: chocolate changes from a solid to a liquid in the sunlight *)
  shows False
  sledgehammer
  oops

end
```

Figure 6: An example of the constructed Isabelle/HOL Theory.

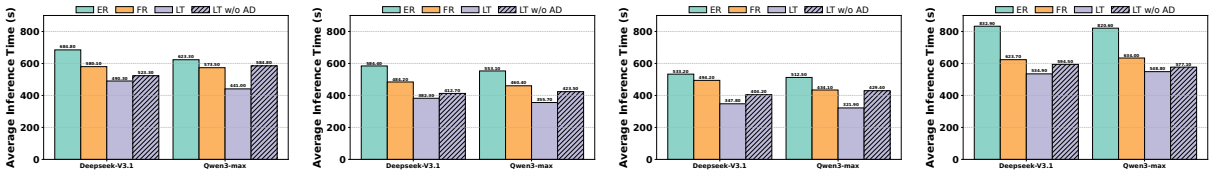


(a) FOLIO (b) ProofWriter (c) PrOntoQA (d) EntailmentBank



(e) FOLIO (f) ProofWriter (g) PrOntoQA (h) EntailmentBank

Figure 7: Top: Average number of iterations required to refine an explanation. Bottom: Average inference time per iteration. Comparison between ER (Explanation-Refiner), FR (Faithful-Refiner), LT (LLM-TP Tree) and LT w/o AD (LT without atomic decomposition).



(a) FOLIO (b) ProofWriter (c) PrOntoQA (d) EntailmentBank

Figure 8: Average inference time per iteration in thinking mode LLMs. Comparison between ER (Explanation-Refiner), FR (Faithful-Refiner), LT (LLM-TP Tree) and LT w/o AD (LT without atomic decomposition).

proaches, averaged over all LLM backbones.

H Solver Proof Depths

Table 5 and Table 6 shows Per-LLM depth alignment details for ProofWriter and EntailmentBank in refined and unrefined cases. Each cell reports the mean used proof depth averaged over instances within the corresponding gold depth bucket, for a fixed framework and LLM backbone.

I Atomic Decomposition Prompt

We build the atomic decomposition prompt based on Srikanth and Rudinger (2025) as:

SYSTEM: You are an expert in symbolic/logical reasoning, linguistic and natural language inference. You are given a sentence and its logical information. Generate a list of atomic facts that are strictly logically entailed from the given sentence.
Instructions:

1. Keep each fact independent and self-contained.
2. Each fact should make sense when read on its own.
3. Only write facts that are directly described or supported by the sentence.
4. The atomic facts must be logically entailed from the given sentence.

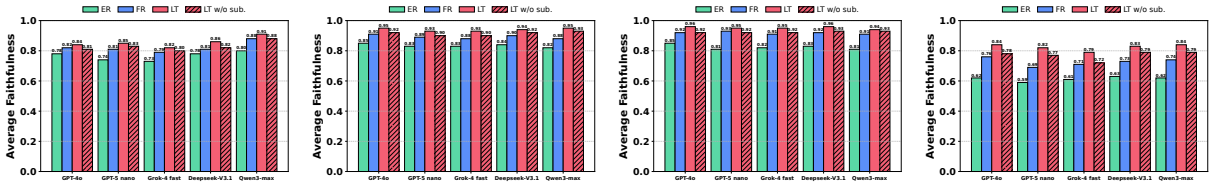
USER: Here are some examples:
Example:
Provided Sentence:
Professional actors are in a summer performance.
Answer:
Atom 1: The people are professional.
Atom 2: The people are actors.
Atom 3: The performance is during summer.
Task:
Provided Sentence: `{sentence}`

Answer:

J Entailment Tree Construction Prompt

SYSTEM: You are an expert in natural language inference and textual entailment. Given the following premise sentences and a final conclusion, generate some step-by-step intermediate conclusion sentences to finally infer the final conclusion.
Instructions:

1. The intermediate conclusion sentences must be strictly logical entailed from the premise sentences.
2. Since the reasoning is step-by-step, the intermediate conclusion sentences can be the new premise sentences for next step.
3. One intermediate conclusion sentence can be generated from multiple premise sentences.
4. If the final conclusion can be directly inferred from the premise sentences, you need to state the intermediate



(a) FOLIO (b) ProofWriter (c) PrOntoQA (d) EntailmentBank

Figure 9: Comparison of the average faithfulness of the autoformalisation process across different approaches.

(a) ProofWriter: mean used proof depth by gold depth

Framework	LLM	Gold depth d				
		1	2	3	4	5
Explanation-Refiner	GPT-4o	1.00	2.32	2.51	2.92	3.23
	GPT-5 nano	1.14	2.00	2.67	2.61	2.94
	Grok-4 fast	1.08	2.04	2.53	2.62	2.92
	Deepseek-V3.1	1.28	2.41	2.23	2.87	3.04
	Qwen3-max	1.10	2.43	2.24	2.90	3.11
Faithful-Refiner	GPT-4o	1.00	2.26	2.76	3.43	3.94
	GPT-5 nano	1.02	2.04	3.13	3.34	3.63
	Grok-4 fast	1.14	2.09	2.67	3.18	3.87
	Deepseek-V3.1	1.00	2.17	2.78	3.41	3.73
	Qwen3-max	1.00	2.22	2.77	3.58	3.79
LLM-TP Tree	GPT-4o	1.00	1.99	2.94	3.76	4.72
	GPT-5 nano	1.08	2.12	2.98	3.68	4.53
	Grok-4 fast	1.17	1.89	3.18	3.49	5.03
	Deepseek-V3.1	1.00	2.03	3.08	3.84	4.94
	Qwen3-max	1.00	2.00	3.00	3.89	4.88

(b) EntailmentBank: mean used proof depth by gold depth

Framework	LLM	Gold Depths d				
		1	2	3	4	5
Explanation-Refiner	GPT-4o	1.23	2.14	2.65	3.15	3.21
	GPT-5 nano	1.21	2.23	2.43	2.91	2.94
	Grok-4 fast	1.13	2.21	2.52	2.93	2.92
	Deepseek-V3.1	1.16	2.28	2.67	3.04	3.16
	Qwen3-max	1.28	2.32	2.63	3.18	3.20
Faithful-Refiner	GPT-4o	1.12	2.10	2.87	3.33	3.67
	GPT-5 nano	1.11	2.12	2.64	3.22	3.40
	Grok-4 fast	1.17	2.13	2.72	3.58	3.42
	Deepseek-V3.1	1.03	2.25	2.78	3.40	3.69
	Qwen3-max	1.17	2.20	2.93	3.47	3.73
LLM-TP Tree	GPT-4o	1.00	2.10	2.85	3.90	4.81
	GPT-5 nano	1.00	1.84	2.84	3.86	4.71
	Grok-4 fast	1.00	2.04	2.92	3.88	4.73
	Deepseek-V3.1	1.00	1.97	3.03	4.03	4.85
	Qwen3-max	1.00	1.96	2.94	3.98	4.92

Table 5: Per-LLM depth alignment details for ProofWriter and EntailmentBank in refined cases.

conclusion sentences are empty.

5. There might be redundant premise sentences.

USER: Here are some examples:
Example:
Initial Premises:
1. Monkeypox is an infectious disease caused by the monkeypox virus.
2. Monkeypox virus can occur in certain animals, including humans.
3. Humans are mammals.
4. Mammals are animals.
5. Symptoms of Monkeypox include fever, headache, muscle pains, feeling tired, and so on.
6. People feel tired when they get a flu.

Final Conclusion:
There is an animal.

Answer:
Monkeypox is an infectious disease caused by the monkeypox virus.
Monkeypox virus can occur in certain animals, including humans.

Conclusion: Humans can get monkeypox.

Humans are mammals.
Mammals are animals.
Conclusion:Humans are animals.

...
Humans are animals.
Therefore, there is an animal (humans) that can get monkeypox and feel tired. Conclusion:Therefore, there is an animal.

Initial Premises:

Final Conclusion:

Answer:

(a) ProofWriter: mean used proof depth by gold depth

Framework	LLM	Gold depth d				
		1	2	3	4	5
Explanation-Refiner	GPT-4o	1.00	2.34	2.49	2.57	2.33
	GPT-5 nano	1.12	1.99	2.57	2.46	2.24
	Grok-4 fast	1.10	1.87	2.48	2.38	2.32
	Deepseek-V3.1	1.18	2.37	1.98	2.84	2.54
	Qwen3-max	1.00	2.33	2.11	2.47	2.62
Faithful-Refiner	GPT-4o	1.02	2.16	2.73	3.44	3.12
	GPT-5 nano	1.02	1.95	3.23	3.24	3.02
	Grok-4 fast	1.16	2.46	2.62	3.09	3.06
	Deepseek-V3.1	1.00	1.98	2.78	2.98	3.27
	Qwen3-max	1.00	2.22	2.67	3.15	3.43
LLM-TP Tree	GPT-4o	1.02	2.13	3.01	3.72	4.72
	GPT-5 nano	1.09	2.14	2.95	3.54	4.65
	Grok-4 fast	1.15	1.79	3.16	3.50	5.13
	Deepseek-V3.1	1.02	2.13	3.04	3.86	4.87
	Qwen3-max	1.00	1.98	3.00	3.88	4.91

(b) EntailmentBank: mean used proof depth by gold depth

Framework	LLM	Gold Depths d				
		1	2	3	4	5
Explanation-Refiner	GPT-4o	1.53	2.84	2.48	2.44	2.21
	GPT-5 nano	1.39	2.99	2.33	2.31	1.95
	Grok-4 fast	1.43	2.83	2.24	2.21	1.92
	Deepseek-V3.1	1.48	2.58	2.33	2.14	1.96
	Qwen3-max	1.88	3.12	2.32	2.19	2.12
Faithful-Refiner	GPT-4o	1.34	2.42	2.48	3.23	3.55
	GPT-5 nano	1.33	2.32	2.41	3.32	3.30
	Grok-4 fast	1.49	2.28	2.43	3.54	3.42
	Deepseek-V3.1	1.10	2.85	2.34	3.38	3.45
	Qwen3-max	1.67	2.74	2.53	3.43	3.42
LLM-TP Tree	GPT-4o	1.03	2.13	2.83	3.93	4.62
	GPT-5 nano	1.13	1.86	2.80	3.78	4.51
	Grok-4 fast	1.12	2.04	2.95	3.91	4.43
	Deepseek-V3.1	1.03	1.89	3.12	4.12	4.67
	Qwen3-max	1.00	2.01	2.99	4.08	4.74

Table 6: Per-LLM depth alignment details for ProofWriter and EntailmentBank in unrefined cases.

K θ-substitution Autoformalisation Prompt

SYSTEM: You are an expert in symbolic/logical reasoning and autoformalisation. You are given a sentence.Extract the logical relation from a given natural language sentence and construct a logical template representing its structure, following the format demonstrated in the examples.

Instructions:

1. Read and understand the provided sentence.
2. Identify key entities, relationships, and logical structure (e.g., quantifiers, conjunctions, disjunctions, implications).
3. Abstract the sentence into a logical template using placeholders (e.g., $\forall$ x y z. P(x) $\wedge$ Q(y) $\wedge$ R(z) $\rightarrow$ S) that capture the logical form, not the specific content.

Follow these steps:
1. Parse the sentence and reason step-by-step about its logical components: What are its predicates, quantifiers, variables, and connectives?
2. Formulate the logical template, using generic predicate letters (P, Q, R, S, etc.) and variable placeholders (x, y, z, etc.), mirroring the schema shown in the examples.
...
Example 1:
Sentence: If someone wins the lottery, they will buy a new house or a car.

Example 1 Answer:
Logical Template: $\forall$x y z. P(x) $\wedge$ Q(y) $\rightarrow$ (R(z) $\vee$S(z))
...
Provided Sentence: `{sentence}`

Answer: