Decompose-and-Formalise: Recursively Verifiable Entailment Trees for Explanation-Based 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 entailment verification and proof-guided refinement of explanations for natural language inference (NLI). However, long multi-step explanations remain brittle: 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 six benchmarks spanning human-written and synthetically generated explanations and five LLM backbones, our method achieves the best final explanation refinement rate on every dataset, with especially large gains on the human-written benchmarks (29.0%, 60.0%, and 48.9% over Explanation-Refiner on e-SNLI, QASC, and EntailmentBank). It also improves autoformalisation faithfulness, reduces refinement iterations and runtime, and preserves competitive downstream label accuracy on the datasets where end-task evaluation is defined.

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 explanation-based 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 recent progress, scaling formal refinement to long and linguistically rich explanation settings remains difficult. 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 reproving 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*

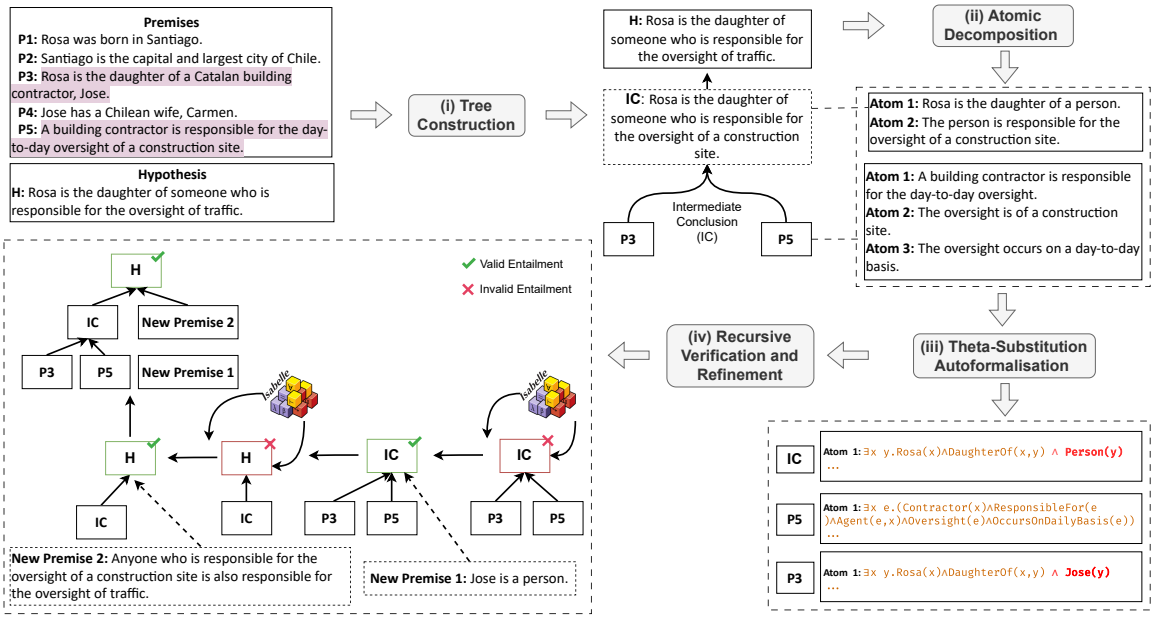


Figure 1: An illustration of the proposed framework for recursively verifiable natural language inference (NLI) via entailment trees, atomic decomposition, and θ-substitution autoformalisation. The entailment tree is initially constructed from the given premises and hypothesis with generated intermediate 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 across six benchmarks using five LLM backbones. 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). Our approach achieves the best refinement performance on every dataset, with gains over Explanation-Refiner of 29.0%, 60.0%, and 48.9% on e-SNLI, QASC, and EntailmentBank, and 26.2%, 21.7%, and 21.6% on FOLIO, ProofWriter, and PrOntoQA, respectively. It also attains the highest autoformalisation faithfulness on all six datasets, reduces refinement iterations and runtime, and maintains competitive downstream label accuracy where end-task evaluation is directly comparable. In summary, our contributions are:

1. We introduce LLM-TP Tree, a recursive entailment-tree framework that verifies and refines multi-step explanations for verifiable explanation-based 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

42.9%, 38.4%, 36.8%, 35.3%, 38.8%, and 40.4% 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 0.136, 0.234, 0.210, 0.090, 0.106, and 0.128 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 Explanation-Based NLI

In this work, we define an explanation-based NLI instance i with an explanation 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 verifies 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 explanation witnesses in explanation-based NLI. The core data structure is an entailment reasoning tree T_i. Figure 1 provides an overview.

3.1 Entailment Tree Construction

Given an explanation 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 explanatory sentences $P_i' \subseteq P_i$, 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 $\bigwedge_{u \in \mathcal{D}(v)} \varphi_u \models \varphi_v$.

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 that participate in proof obligations. For the hypothesis h_i, we generate a set of atoms

$$D(h_i) = \{a_1^h, a_2^h, \ldots, a_k^h\}, \quad \forall a \in D(h_i),\ h_i \vDash a. \tag{1}$$

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

$$D(p_j) = \{a_1^{p_j}, a_2^{p_j}, \ldots, a_{t_j}^{p_j}\}, \quad \forall a \in D(p_j),\ p_j \vDash a. \tag{2}$$

We define the global atom set induced by the premises as

$$D(P_i) := \bigcup_{j=1}^{n} D(p_j). \tag{3}$$

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 $\varphi \vDash a$. Intuitively, each atom is a logically entailed consequence of the original sentence, so decompo-

sition does not introduce new information that is not supported by the source. We obtain candidate atoms by prompting an LLM (see Appendix J). 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 into logical forms. Building on this representation, we use a multi-step θ-substitution *structured prompting-and-checking protocol* 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:

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

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 localised 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 2 summarises the procedure.

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.

When TP cannot discharge a lemma $\text{lemma}(h, a)$ for some atom $a \in D(h)$, we 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 the implicated subtree. This local update may revise which premises from P_i are used in that region, the intermediate conclusions generated there, their atomic decompositions $D(\cdot)$, and the corresponding autoformalisation outputs $\Phi(\cdot)$, while previously verified disjoint subtrees are left unchanged. We then 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 on two types of datasets for explanation-based natural language inference: human-written explanation benchmarks and synthetically generated deductive reasoning benchmarks. The human-written group includes e-SNLI (Camburu et al., 2018), QASC (Khot et al., 2019), and EntailmentBank (Dalvi et al., 2021); these datasets contain human-written explanations that are often material-inference based rather than fully deductive, and prior work has shown that gold explanations in explanation-oriented NLI benchmarks are not uniformly logically valid under formal verification (Valentino et al., 2021). The synthetically generated group includes ProofWriter (Tafjord et al., 2021), FOLIO (Han et al., 2024), and PrOntoQA (Saparov and He, 2023). Detailed protocols for the fully reported datasets, including sample sizes, are provided in Appendix A.

Baselines and Models We compare against two state-of-the-art explanation refinement models, Explanation-Refiner (Quan et al., 2024) and Faithful-Refiner (Quan et al., 2025b). 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)). LLM implementation details is described in Appendix D.

5 Empirical Results and Evaluation

LLM-TP Tree effectively verifies and refines explanations for explanation-based NLI. Table 1 reports the main results on the explanation refinement task. Across the six reported datasets and all LLM backbones, LLM-TP Tree achieves the best final refinement performance (*Fin.*), indicating that it is consistently more effective at producing a checkable witness for explanation-based NLI. Averaged over backbone LLMs, LLM-TP Tree improves the final verified rate over Explanation-Refiner

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.*
Human-written explanation											
e-SNLI	Explanation-Refiner	31.00	71.00	26.00	52.00	21.00	47.00	29.00	68.00	33.00	65.00
	Faithful-Refiner	39.00	89.00	33.00	67.00	26.00	62.00	32.00	93.00	35.00	91.00
	LLM-TP Tree	38.00	**96.00** ↑25.00	30.00	**81.00** ↑29.00	32.00	**73.00** ↑26.00	37.00	**98.00** ↑30.00	35.00	**100.00** ↑35.00
QASC	Explanation-Refiner	4.00	26.00	2.00	21.00	3.00	22.00	6.00	37.00	5.00	39.00
	Faithful-Refiner	10.00	79.00	8.00	71.00	7.00	69.00	13.00	92.00	11.00	90.00
	LLM-TP Tree	13.00	**92.00** ↑66.00	12.00	**82.00** ↑61.00	14.00	**77.00** ↑55.00	15.00	**96.00** ↑59.00	14.00	**98.00** ↑59.00
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
Synthetically generated explanation											
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

Table 1: *Init.* is the percentage of generated explanation trees that are already logically valid before any local repair. *Fin.* is the percentage that become logically valid after verification-driven localised refinement. **Bold** values indicate the best performance. Arrows indicate the absolute performance gain of LLM-TP Tree over the baseline.

Dataset	Iterations				Inference time			
	ER	FR	LT	LT w/o AD	ER	FR	LT	LT w/o AD
e-SNLI	3.91	1.95	1.54	1.93	361.22	308.04	206.18	266.02
QASC	4.40	3.17	2.47	2.80	413.52	331.14	254.88	290.38
EntailmentBank	4.97	4.46	4.05	4.35	468.10	355.64	295.88	322.92
FOLIO	4.27	3.75	3.04	3.57	348.14	287.30	225.16	266.92
ProofWriter	3.47	3.08	2.42	2.88	298.52	238.86	182.74	207.18
PrOntoQA	3.36	2.73	2.03	2.44	277.84	229.36	165.50	204.94
Avg.	4.06	3.19	2.59	2.99	361.22	291.72	221.72	259.73

Table 2: Average refinement iterations required and average inference time per iteration, both 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).

by 29.00%, 60.00%, 48.93%, 26.23%, 21.73%, and 21.60% on e-SNLI, QASC, EntailmentBank, FOLIO, ProofWriter, and PrOntoQA, respectively. These gains are especially strong on the human-written explanation benchmarks. The largest absolute gain appears on QASC, while EntailmentBank remains the most challenging multi-hop benchmark in our suite: there, whole-explanation refiners remain below 52% final validity, whereas 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 cycle. On synthetic benchmarks such as ProofWriter and PrOntoQA, LLM-TP Tree yields substantially higher initial validity than whole-explanation 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 more realistic 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. Table 2 reports two complementary efficiency views: the average number of refinement iterations and the average inference time per refinement itera-

Dataset	ER	FR	LT	LT w/o sub.
e-SNLI	0.726	0.810	0.862	0.826
QASC	0.630	0.820	0.864	0.828
EntailmentBank	0.614	0.726	0.824	0.770
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
Avg.	0.732	0.831	0.883	0.848

Table 3: Average autoformalisation faithfulness (cosine similarity) averaged over five LLM backbones.

tion. Across all datasets, LT requires fewer repair loops than the baseline refiners, lowering the iteration count by 1.47 relative to ER and by 0.60 relative to Faithful-Refiner on average. The same advantage appears in running time: LT reduces the mean per-iteration cost by 139.50 relative to ER and by 70.00 relative to FR, with the largest absolute savings on EntailmentBank and QASC. To isolate the role of atomic decomposition, we disable sentence splitting in LT w/o AD. Relative to the full LT system, removing atomic decomposition increases the number of required iterations by 0.30 to 0.53 and increases the mean per-iteration running time by 24.44 to 59.84 across datasets. This pattern supports the intended mechanism: decomposing premises and intermediate conclusions into atomic subclaims allows verification failures to be localised to a smaller implicated set, so the system can repair the minimal logical mismatch with less repeated generation and prover checking. Detailed per-backbone runtime plots are provided in Appendix G.

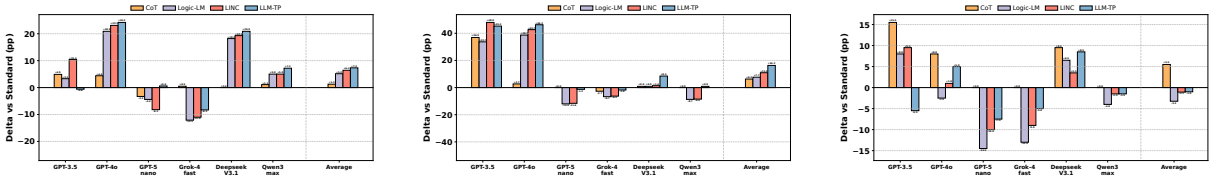
Multi-step θ-substitution autoformalisation improves semantic faithfulness. A faithful autoformalisation should preserve the semantic content of the original sentence as much as possible. Following the rule-based informalisation procedure by Quan et al. (2025b), we quantify faithfulness by converting each predicted logical form back into natural language and computing the cosine similarity between the informalised sentence and the original sentence. Table 3 reports the backbone-averaged results on all six datasets. LLM-TP Tree achieves the highest faithfulness score on every dataset, reaching an average of 0.883 across datasets and improving over Explanation-Refiner by 0.151 and over Faithful-Refiner by 0.052. The gains are especially clear on the more challenging human-written benchmarks: relative to Explanation-Refiner, the improvement is 0.136 on e-SNLI, 0.234 on QASC, and 0.210 on EntailmentBank. As expected, the absolute scores remain higher on synthetic benchmarks, whose language more closely follows templated deductive patterns, whereas human-written explanations pose greater challenges for preserving scope, roles, and composition decisions during autoformalisation. We further perform an ablation by removing θ-substitution (LT w/o sub.). This variant lowers faithfulness by 0.035 on average relative to the full model, with the largest drop on EntailmentBank (0.054), indicating that staged substitution is particularly important when the language is more naturalistic and less template-like. We also show the full ablation study results in Appendix E. We also analyse autoformalisation errors in Appendix B.

5.1 Explanation Refinement Drift

We analyse refinement drift using proof-depth alignment. Figure 4 measures 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 whole-explanation 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 I. We further analyse raw LLM-generated explanations for refinement and verification on the human-written explanation datasets in Appendix E.1.

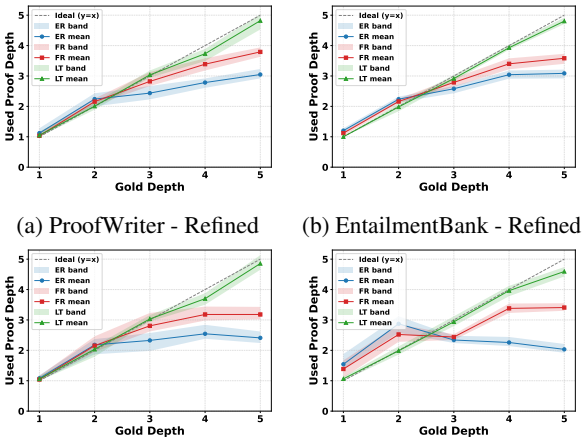
5.2 Logical Reasoning

We further evaluate downstream logical reasoning on the synthetically generated benchmarks by comparing standard prompting, chain-of-thought (CoT), Logic-LM (Pan et al., 2023), and LINC (Olausson et al., 2023). Figure 3 reports accuracy deltas relative to standard prompting on FOLIO, ProofWriter, and PrOntoQA, with absolute accuracies deferred to Appendix F. We restrict this comparison to benchmarks with a directly comparable label-prediction setting. By contrast, the human-written explanation benchmarks are used here to evaluate whether human-authored expla-



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

Figure 3: Delta in logical reasoning accuracy relative to standard prompting.



(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.

nation structures can be verified and refined into logically valid witnesses, rather than to instantiate a matched end-task label-prediction protocol.

LLM-TP Tree yields the largest average gains on FOLIO and ProofWriter, improving over standard prompting by 7.3 and 16.3 points, respectively, and outperforming the other neuro-symbolic baselines on average. This suggests that recursive tree-structured verification is most beneficial when the task requires multi-step compositional reasoning beyond what standard prompting reliably captures. The pattern changes on PrOntoQA, where CoT is strongest on average (+5.5) and the neuro-symbolic methods are near-neutral or slightly negative, with LLM-TP Tree at -1.0, LINC at -1.1, and Logic-LM at -3.2. This indicates that when direct prompting is already highly effective, the additional formalisation and proving stages may offer limited accuracy headroom. Even in this regime, LLM-TP Tree remains the strongest neuro-symbolic variant and retains the advantage of producing solver-checkable proofs.

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 explanation-based NLI that follows a recursively 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 six datasets with multiple LLM backbones show improved TP-refined explanation rates, more faithful autoformalisation, and reduced refinement drift while maintaining strong end-task reasoning accuracy.

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

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

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.

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

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

a premise-hypothesis 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 a premise-hypothesis 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 premise-hypothesis pairs (P, h) for verification and refinement.

e-SNLI. e-SNLI (Camburu et al., 2018) extends natural language inference examples with human-written explanations. Each instance contains one premise sentence, one hypothesis sentence, and one explanation sentence. For explanation refinement, we adopt 100 instances following the mutually exclusive selection protocol used in Valentino et al. (2021). In our setting, each example is cast into a verification-oriented format centred on the hypothesis h, with the premise sentence and the human-written explanation sentence retained as the natural-language support set P.

QASC. QASC (Khot et al., 2019) originates from multiple-choice science question answering and is designed to test sentence composition via a short explanatory chain. In the explanation-oriented setting used here, each instance is represented by one hypothesis together with two explanatory sentences that jointly support that hypothesis. This makes QASC a useful human-written benchmark for studying whether a verifier can certify a compact two-step natural-language explanation rather than a single isolated entailment statement. For explanation refinement, we adopt 100 instances following the mutually exclusive selection protocol used in Valentino et al. (2021). Each instance is cast into (P, h), where P contains the two explanatory sentences and h is the hypothesis.

EntailmentBank. EntailmentBank (Dalvi et al., 2021) is a multi-hop entailment benchmark that provides naturalistic premises and hypotheses together with structured explanation annotations. For explanation refinement, we use the Task 2 subset with gold proof depths from 1 to 5 and sample 150 instances spanning that range to ensure coverage of both shallow and longer-horizon chains. At inference time, we discard the annotated intermediate conclusions and retain only the provided supporting statements as the premise set P together with the hypothesis h. Gold explanation structure is used only for depth-balanced sampling and for the post-hoc proof-depth alignment analysis. Each instance is thus cast into (P, h), where P contains supporting statements only 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.

We further analyse autoformalisation failures by categorising errors in the generated Isabelle/HOL theories into four types across 200 sampled theories. Figure 5 shows that LLM-TP Tree consistently yields the lowest error rates across datasets, with the largest margin on the more naturalistic EntailmentBank cases. Removing θ-substitution (LT w/o sub.) leads to a clear increase in semantic and structural errors, particularly quantifier and variable errors on EntailmentBank, highlighting the importance of staged substitution autoformalisation for maintaining a prover-compatible logical structure.

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 intended 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.

Algorithm 1 VerifyAndRefineTree($\mathcal{T}$, TP)

$\mathcal{T}_{\text{cur}} \leftarrow \mathcal{T}$
$\mathcal{V} \leftarrow \emptyset$ {verified subtree roots}
while $\text{root}(\mathcal{T}_{\text{cur}}) \notin \mathcal{V}$ **do**
  $\mathcal{F} \leftarrow \{v \in \text{internal}(\mathcal{T}_{\text{cur}}) \mid$ every child of v is a premise leaf or already in $\mathcal{V}\}$
  for each $v \in \mathcal{F}$ **do**
    $\mathcal{T}_{\text{sub}} \leftarrow \text{extract_subtree}(\mathcal{T}_{\text{cur}}, v)$
    $\mathcal{T}^{\star}_{\text{sub}} \leftarrow \text{VerifySubtree}(\mathcal{T}_{\text{sub}}, TP)$
    $r \leftarrow \text{root}(\mathcal{T}^{\star}_{\text{sub}})$
    Update $\mathcal{T}_{\text{cur}}$ by replacing $\mathcal{T}_{\text{sub}}$ with $\mathcal{T}^{\star}_{\text{sub}}$
    Promote r as a support statement for its parent level
    $\mathcal{V} \leftarrow (\mathcal{V} \setminus \text{descendants}(r)) \cup \{r\}$
  end for
end while
return $\mathcal{T}_{\text{cur}}$ {verified explanation witness}

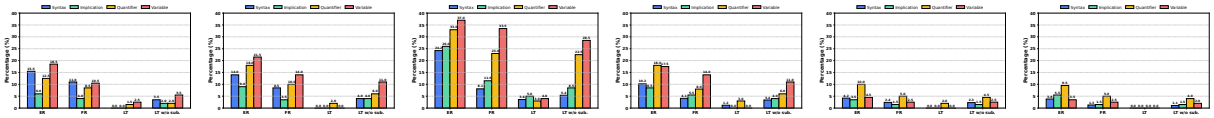
Variable errors. Variable errors occur when logical variables are incorrectly assigned to predicate arguments, resulting in role confusion or broken discourse links. This category covers a wide range of semantic binding mistakes, including swapping semantic roles (e.g., agent vs. patient), incorrectly 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

We provide both the global tree-level verification-and-refinement controller and the local recursive subtree routine used inside it.

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



(a) e-SNLI (b) QASC (c) EntailmentBank (d) FOLIO (e) ProofWriter (f) PrOntoQA

Figure 5: Distribution of autoformalisation errors in generated Isabelle/HOL theories across syntax, implication, quantifier, and variable categories.

Algorithm 2 VerifySubtree($\mathcal{T}_{\text{sub}}$, TP)

1: $h \leftarrow \text{root}(\mathcal{T}_{\text{sub}})$ {current hypothesis node}
2: $E \leftarrow \text{support_leaves}(\mathcal{T}_{\text{sub}})$ {premises or verified child roots}
3: Construct $A(E) = \bigcup_{e \in E} \{\Phi(a) \mid a \in D(e)\}$ {axioms}
4: **for** each atom $a \in D(h)$ **do**
5: Define the goal $\tau \leftarrow \Phi(a)$
6: **if** $TP.\text{prove}(A(E) \vdash \tau) = \text{FAIL}$ **then**
7: diag $\leftarrow TP.\text{diagnostics}()$
8: Identify implicated explanation nodes $\widehat{E} \subseteq E$ from diag
9: Refine $\widehat{E}$ using an LLM conditioned on $(\widehat{E}, \text{diag})$
10: Update $\mathcal{T}_{\text{sub}}$; recompute $D(\cdot)$ and re-autoformalise $\Phi(\cdot)$
11: **return** VerifySubtree($\mathcal{T}_{\text{sub}}$, TP) {retry locally}
12: **end if**
13: **end for**
14: Mark h as verified
15: **return** $\mathcal{T}_{\text{sub}}$ {verified and possibly refined subtree}

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 the full ablation results of LLM-TP Tree (LT), using *Init./Fin.* logically valid explanation rates. Removing either atomic decomposition or multi-step θ-substitution autoformalisation consistently degrades refinement quality across datasets and backbones. Across the reported settings, *Init.* decreases by 1.33 to 13.12 points and *Fin.* decreases by 0.00 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 in most settings, indicating that multi-step substitution contributes materially to producing prover-compatible formalisations.

Dataset	LT		w/o Atom		w/o θ	
	Init.	*Fin.*	*Init.*	*Fin.*	*Init.*	*Fin.*
Backbone: GPT-4o						
e-SNLI	38.00	96.00	31.00	85.00	33.00	94.00
QASC	13.00	92.00	8.00	82.00	11.00	88.00
EntailmentBank	21.33	70.00	17.33	53.33	19.33	61.33
FOLIO	81.15	90.16	71.31	86.07	75.41	88.52
ProofWriter	91.33	95.33	87.33	92.00	89.33	94.67
PrOntoQA	92.00	97.33	86.67	92.67	90.00	94.00
Backbone: GPT-5 nano						
e-SNLI	30.00	81.00	23.00	65.00	20.00	72.00
QASC	12.00	82.00	4.00	68.00	6.00	74.00
EntailmentBank	18.00	74.00	11.33	50.00	16.67	61.33
FOLIO	66.39	81.15	55.74	70.49	59.01	72.95
ProofWriter	85.33	91.33	76.00	86.00	84.00	87.33
PrOntoQA	93.33	98.00	91.33	96.67	93.33	96.00
Backbone: Grok-4 fast						
e-SNLI	32.00	73.00	23.00	67.00	30.00	69.00
QASC	14.00	77.00	6.00	71.00	9.00	70.00
EntailmentBank	16.67	68.67	6.67	53.33	11.33	54.67
FOLIO	62.30	79.51	49.18	55.73	63.93	70.49
ProofWriter	72.67	90.00	68.00	82.00	69.33	88.00
PrOntoQA	86.00	91.33	79.33	88.67	84.00	91.33
Backbone: Deepseek-V3.1						
e-SNLI	37.00	98.00	33.00	91.00	29.00	94.00
QASC	15.00	96.00	11.00	95.00	12.00	96.00
EntailmentBank	22.00	71.33	13.33	56.00	16.00	64.00
FOLIO	83.61	92.62	72.13	84.43	75.41	87.70
ProofWriter	91.33	98.00	90.00	96.00	91.33	98.00
PrOntoQA	97.33	100.00	91.33	95.33	91.33	98.00
Backbone: Qwen3-max						
e-SNLI	35.00	100.00	30.00	94.00	33.00	96.00
QASC	14.00	98.00	8.00	90.00	12.00	92.00
EntailmentBank	22.67	78.67	18.00	61.33	20.00	71.33
FOLIO	85.25	95.08	72.13	86.89	79.51	90.98
ProofWriter	92.00	98.00	88.00	93.33	88.00	96.00
PrOntoQA	98.00	100.00	86.67	94.00	92.00	96.00

Table 4: Ablations of LLM-TP Tree across all reported LLM backbones (Init/Fin verified rate, %).

E.1 Raw LLM-Generated Explanations on Human-Written Benchmarks

Table 5 further shows that raw LLM-generated explanations are themselves often not logically well-formed, even when generation is guided by prompt examples. On the human-written benchmarks, the initial logically valid rates of these raw explanations remain low, indicating that explanation generation alone does not reliably produce verifier-ready proof structures. When these raw explanations are passed to whole-explanation refiners, refinement performance drops markedly: averaged over backbone LLMs, the final verified rate of Explanation-Refiner falls from 60.6 to 35.6 on e-SNLI, from 29.0 to 21.2 on QASC, and from 23.6 to 14.0 on EntailmentBank, while Faithful-Refiner drops from

```
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.

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.*
e-SNLI	Explanation-Refiner raw LLM	11.00	43.00	14.00	25.00	10.00	31.00	15.00	38.00	19.00	41.00
	Explanation-Refiner	31.00	71.00	26.00	52.00	21.00	47.00	29.00	68.00	33.00	65.00
	Faithful-Refiner raw LLM	12.00	62.00	10.00	57.00	9.00	54.00	19.00	67.00	17.00	66.00
	Faithful-Refiner	39.00	89.00	33.00	67.00	26.00	62.00	32.00	93.00	35.00	91.00
	LLM-TP Tree raw LLM	34.00	95.00	29.00	82.00	30.00	69.00	35.00	94.00	29.00	95.00
	LLM-TP Tree	38.00	96.00	30.00	81.00	32.00	73.00	37.00	98.00	35.00	100.00
QASC	Explanation-Refiner raw LLM	2.00	13.00	4.00	18.00	5.00	14.00	5.00	32.00	2.00	29.00
	Explanation-Refiner	4.00	26.00	2.00	21.00	3.00	22.00	6.00	37.00	5.00	39.00
	Faithful-Refiner raw LLM	12.00	65.00	9.00	48.00	3.00	44.00	10.00	70.00	6.00	67.00
	Faithful-Refiner	10.00	79.00	8.00	71.00	7.00	69.00	13.00	92.00	11.00	90.00
	LLM-TP Tree raw LLM	14.00	93.00	16.00	79.00	18.00	75.00	11.00	92.00	21.00	96.00
	LLM-TP Tree	13.00	92.00	12.00	82.00	14.00	77.00	15.00	96.00	14.00	98.00
EntailmentBank	Explanation-Refiner raw LLM	4.00	11.33	5.33	8.00	6.00	11.33	12.00	19.33	12.67	20.00
	Explanation-Refiner	15.33	26.67	12.00	23.33	8.67	13.33	15.33	26.67	16.67	28.00
	Faithful-Refiner raw LLM	4.67	30.00	6.00	28.67	9.33	40.00	11.33	32.00	12.00	44.00
	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 raw LLM	18.00	74.00	16.67	68.00	12.00	66.00	19.33	68.67	24.00	71.33
	LLM-TP Tree	21.33	70.00	18.00	74.00	16.67	68.67	22.00	71.33	22.67	78.67

Table 5: Logical validity on the human-written explanation benchmarks when each framework is applied either to raw LLM-generated explanations (rows marked "raw LLM") or to the corresponding benchmark setting reported in Table 1. In both cases, *Init.* and *Fin.* denote the logically valid rate before and after refinement.

80.4 to 61.2, from 80.2 to 58.8, and from 47.5 to 34.9, respectively. By contrast, LLM-TP Tree remains much more stable under the same shift, with final verified rates changing only from 89.6 to 87.0 on e-SNLI, from 89.0 to 87.0 on QASC, and from 72.5 to 69.6 on EntailmentBank. Consistent with the refinement-drift analysis above, this robustness comes from the recursive tree structure: local verification follows explicit proof traces and rewrites only implicated subtrees, which helps preserve multi-step reasoning structure rather than collapsing it into explanation drift.

F Full results: Absolute Logical Reasoning Accuracy

G Full results: Model efficiency Across LLM Backbones

Figure 8 shows the average number of refinement iterations (top) and the average inference time per iteration for non-thinking models (bottom) across all reported datasets. Figure 9 shows the average inference time per iteration for thinking-mode models.

H Full Results: Autoformalisation Faithfulness Across LLM Backbones

Figure 10 compares the average faithfulness of the autoformalisation process across different approaches, averaged over all LLM backbones.

I Solver Proof Depths

Table 6 and Table 7 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.

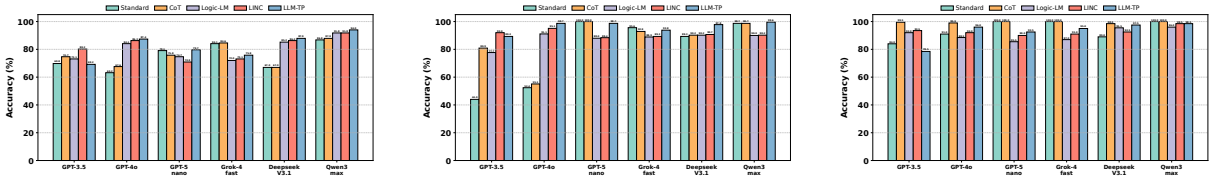
J Atomic Decomposition Prompt

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:

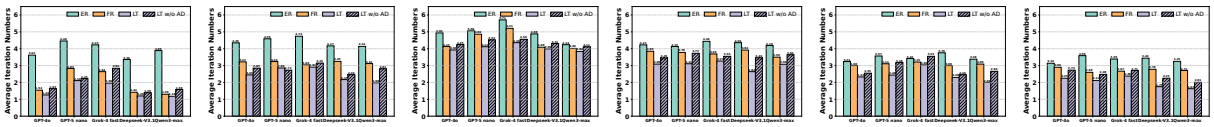
1. Keep each fact independent and self-contained.
2. Each fact should make sense when read on its own.
3. Only write facts that are directly described or supported by the sentence.
4. The atomic facts must be logically entailed from the given sentence.

USER: Here are some examples:
Example:
Provided Sentence:
Professional actors are in a summer performance.

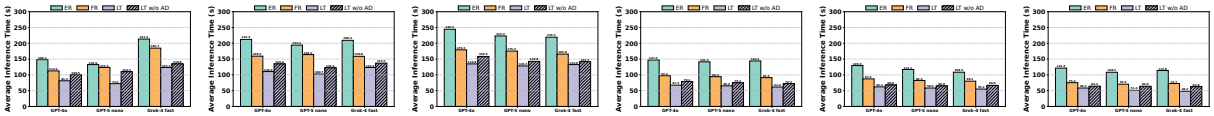


(a) FOLIO (b) ProofWriter (c) PrOntoQA

Figure 7: Absolute logical reasoning accuracy on FOLIO, ProofWriter, and PrOntoQA.

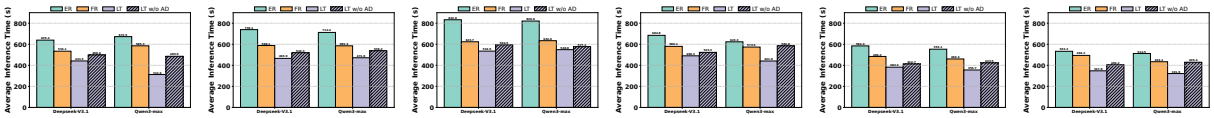


(a) e-SNLI (b) QASC (c) EntailmentBank (d) FOLIO (e) ProofWriter (f) PrOntoQA



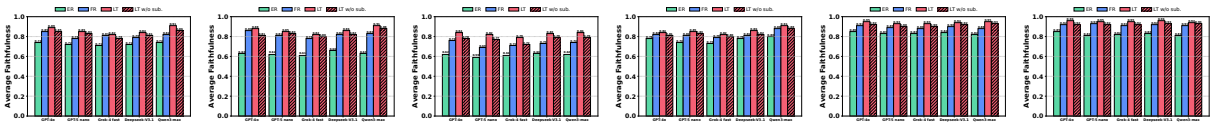
(g) e-SNLI (h) QASC (i) EntailmentBank (j) FOLIO (k) ProofWriter (l) PrOntoQA

Figure 8: 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) e-SNLI (b) QASC (c) EntailmentBank (d) FOLIO (e) ProofWriter (f) PrOntoQA

Figure 9: 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).



(a) e-SNLI (b) QASC (c) EntailmentBank (d) FOLIO (e) ProofWriter (f) PrOntoQA

Figure 10: Comparison of the average faithfulness of the autoformalisation process across different approaches.

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:

K 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 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.

(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 6: Per-LLM depth alignment details for ProofWriter and EntailmentBank in refined cases.

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:

L 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:

(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 7: Per-LLM depth alignment details for ProofWriter and EntailmentBank in unrefined cases.

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

Answer: