INTERLEAVING AND PROCESS-SUPERVISING LOGICAL REASONING WITH VERIFIABLE REASONING STRUCTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models have demonstrated strong reasoning capabilities through Reinforcement Learning with Verifiable Rewards (RLVR). However, RLVR typically uses outcome correctness as a binary reward and lacks a strict check on the logical validity of intermediate steps, which matters more in logical reasoning. In this paper, we propose **VERST** (VErifiable Reasoning Structure Training), a training framework that obtains a process supervision signal from a symbolic solver's strict verification. The framework introduces a lightweight verifiable structure of atomic facts and explicit inferences, which can be easily translated into SMT-LIB format and verified by a symbolic solver. Based on this, our two-stage approach applies: In *Reasoning Structure Learning* stage, the model is trained to interleave this structure with natural language by supervised fine-tuning. In *Process-supervised RLVR* stage, reinforcement learning is performed with a process reward scored by a symbolic solver on the generated structure. On five logical reasoning benchmarks, VERST improves Qwen3-4B-Instruct-2507 from 75.2% to 86.5%, outperforming Qwen3-235B-A22B (84.2%) and becoming competitive with DeepSeek-V4-Pro-0813 (86.8%) and GPT-5.6-sol (87.8%). Further analysis shows that it improves process quality and brings the impact to other reasoning domains.

1 INTRODUCTION

Large language models are commonly asked to think step by step, producing a chain-of-thought (CoT) before the final answer (Wei et al., 2022; Kojima et al., 2022). Recent work further trains these reasoning traces with outcome-based reinforcement learning, such as group-relative policy optimization (GRPO), and has obtained substantial gains on reasoning tasks (Shao et al., 2024; DeepSeek-AI et al., 2025). The difficulty of logical reasoning sits in the derivation rather than in the answer: the model must carry out a multi-step argument, yet the reported result is only a truth value or a short discrete label, which is more likely to be correct by incidence. Outcome-only supervision cannot tell whether the reasoning steps are logically valid, leading to ineffectiveness in improving logical reasoning tasks (Uesato et al., 2022; Turpin et al., 2023; Lanham et al., 2023). Process supervision may address this gap by scoring intermediate steps (Uesato et al., 2022; Lightman et al., 2024). However, process reward models (PRMs) usually rely on expensive human labels, or synthetic step labels, which are usually noisy (Lightman et al., 2024; Wang et al., 2024; Kamoi et al., 2026).

Formal tools can provide this kind of strict step-level verification. Symbolic solvers such as Z3 (de Moura & Bjørner, 2008) are SMT (Satisfiability Modulo Theories) solvers that automatically decide whether a logical formula is satisfiable over background theories. Theorem provers such as Isabelle (Nipkow et al., 2002) offer stronger logical expressiveness but typically require human-guided proofs, making them harder to automate. However, using them in model training encounters a practical obstacle: turning free-form natural language reasoning into formal logical reasoning, i.e., autoformalization of natural language.

In this paper, we propose **VERST** (VErifiable Reasoning Structure Training), a framework that addresses this obstacle by introducing the VErifiable Reasoning Structure (VERS), a step-wise and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

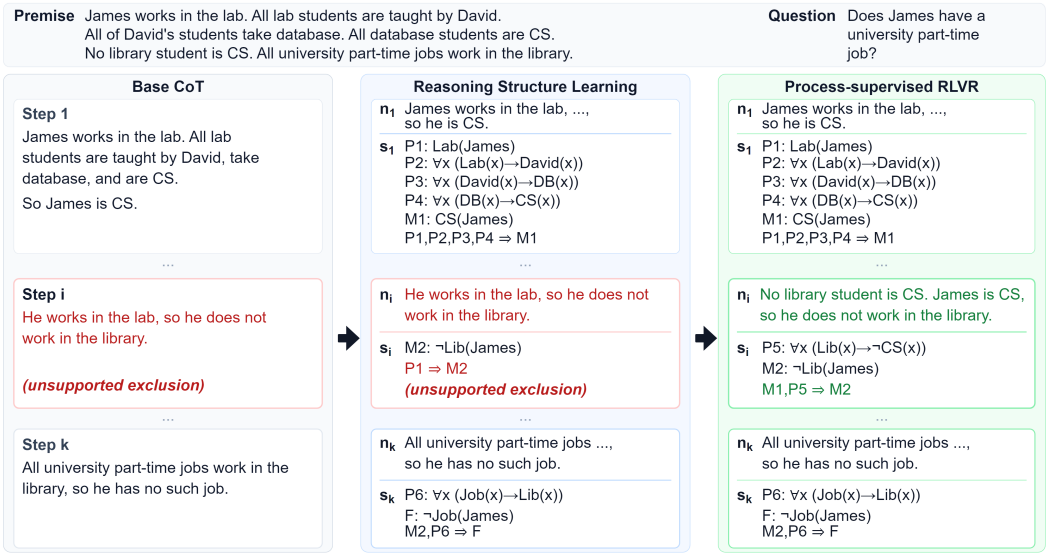

Figure 1: A FOLIO item at three stages. n_i is the natural-language step and s_i is its structured counterpart (facts and inferences). Red and green mark whether a step is logically valid, and the italic gloss illustrates the error type. The reasoning in Base CoT is hard to track. In comparison, Reasoning Structure Learning teaches the model to emit a structured counterpart for each natural-language step; Process-supervised RLVR then optimizes the model to produce valid inferences using a reward from the verifiable structure.

lightweight structure which organizes each reasoning step into atomic facts and explicit inferences. These facts and inferences can be easily translated into SMT-LIB (Barrett et al., 2017), a standard input format for SMT solvers, which is then checked by a SMT solver, providing an accurate process supervision signal for each step. VERST has two stages: (1) **Reasoning Structure Learning**: training the model to interleave its natural-language reasoning traces with VERS; (2) **Process-supervised RLVR**: training the model to improve the reasoning with GRPO (Shao et al., 2024) using an outcome reward and a process reward obtained by checking the logical validity of VERS. Figure 1 illustrates the effect of VERST on one FOLIO example.

For experiments, we train Qwen3-4B model with VERST, and evaluate it on logical reasoning benchmarks. The resulting 4B model outperforms Qwen3-235B-A22B (Yang et al., 2025), and is competitive with DeepSeek-V4-Pro-0813 (DeepSeek-AI, 2026) and GPT-5.6-sol (OpenAI, 2026) in average accuracy. It also outperforms models trained with LogicReward (Xu et al., 2026) and with FoVer (Kamoi et al., 2026). Compared with outcome-only GRPO on the same backbone, VERST improves the logical rigor of intermediate reasoning, not only the final answer.

In summary, our main contributions are as follows:

- We design a verifiable structure of atomic facts and explicit inferences that is easier to translate into the SMT-LIB format for solver verification than natural language text.
- With this structure, a symbolic solver provides a fine-grained process supervision signal, which could be combined with the outcome reward for better reinforcement learning.
- A 4B model trained with VERST matches the accuracy of a model from the same family that is roughly 50 times its size, and approaches that of much stronger closed models.

2 Related Work

2.1 Improving Logical Reasoning with Systematic Procedures

Some works improve logical reasoning by making the model follow a systematic procedure with explicit stages, rules, or search steps, rather than generating a free-form chain. SymbCoT (Xu

et al., 2024) prompts the model to translate the problem, plan with inference rules, solve, and verify its own chain, organizing reasoning into a fixed pipeline. Chain of Logic (Servantez et al., 2024) decomposes a compositional rule, solves each element, and recomposes the result, forming a decompose-solve-recompose tree. LAMBADA (Kazemi et al., 2023) reasons backward from the claim, structuring inference as a backward chain from the goal to the facts. Aristotle (Xu et al., 2025) decomposes the claim, searches for supporting steps, and resolves them, including proof by contradiction, forming a dual-path search over complementary clauses. These methods can make the reasoning more systematic, but they are mostly based on prompting or architectural design and are not training methods.

2.2 Improving Logical Reasoning with Formal Verification Tools

Formal verification tools, such as theorem provers and solvers, can perform exact logical reasoning once a problem is formalized, and have therefore been widely used in logical reasoning research.

They can be used at test time: Logic-LM (Pan et al., 2023) and LINC (Olausson et al., 2023) translate natural-language premises into first-order formulas and call a theorem prover to carry out the deduction. However, this does not improve the model's own logical reasoning capability.

They can also be used during training to synthesize supervision data. Logic-Thinker (Wen et al., 2025) turns solver executions into concise chains for supervised fine-tuning(SFT). FoVer (Kamoi et al., 2026) uses Z3 and Isabelle to label formal reasoning steps and then trains a PRM to score informal traces. VeriCoT (Feng et al., 2026) and LogicReward (Xu et al., 2026) formalize unconstrained natural language steps, check them with a formal verifier, and use the results to construct SFT and DPO data for offline training. Compared to these offline methods, our framework enables process supervision for each on-policy rollout.

3 Method

In this section, we introduce the VERST framework. We first detail the design of the lightweight structure, which facilitates formalization and solver checking, and then introduce our two-stage training framework.

3.1 the Verifiable Reasoning Structure (VERS)

3.1.1 Definition

The structured response r is defined to be a sequence of T steps and a final answer a:

$$r = ((n_1, s_1), \ldots, (n_T, s_T), a)$$

where n_i is the natural-language reasoning of step i, and s_i is the VERS representation of that step; a is the final answer.

Even within a single reasoning step, i.e. separated by punctuations, natural-language reasoning may mix multiple facts with inferences among them, which makes direct formalization rather difficult. In comparison, VERS requires the model to explicitly state the facts and the inference. The VERS s_i consists of two separate kinds of lines:

- a *fact* line ($id : \varphi$): binds an identifier id to one atomic fact φ;
- an *inference* line ($id_1, \ldots, id_k \Rightarrow id_t$): claims the entailment of the target fact id_t from the set of source facts $S = \{id_1, \ldots, id_k\}$.

With these lines, the facts and inferences of the reasoning are presented clearly. Further details of the structure design are given in Appendix A.

3.1.2 Verification

The VERS representation is automatically translated into the SMT-LIB format for verification. For each fact line, a translator is employed to map the atomic fact φ to a single SMT-LIB assertion.

Empirically, a 4B model trained on a small set of translations from facts to SMT-LIB assertions distilled from a stronger teacher, performs reasonably well. The translation prompt is given in Appendix F.4.

For each inference line, a set $S \cup \{\neg id_t\}$ is built for verification. The symbolic solver, Z3 (de Moura & Bjørner, 2008), then checks whether this assertion set is satisfiable, that is, whether there exists a situation in which every assertion in the set is satisfied. An unsatisfiable result means the source facts entail the target and the inference is logically valid whereas a satisfiable result means the opposite.

3.1.3 VERS REWARD

To encourage more valid inferences, the reward for a given step is intuitively related to the ratio of valid inferences in the current step. However, there are two situations that need special attention.

Active Facts and Inferences A reasoning step may contain logically valid inferences that do not contribute to the final answer, and rewarding them does not necessarily lead to a better logic reasoning. Thus, we track from the fact of answer a and mark the inferences and facts that support it as *active*. Inferences and facts that support *active* facts are marked as *active* recursively, while the rest are marked as *dead*. The positive reward is thus restricted to *active* inferences that help to reach the answer.

Consistency Since s_i is a formal representation of the natural language reasoning n_i, it is also possible that they are not presenting exactly the same reasoning. An external model is employed to judge this consistency following the prompt in Appendix F.3. An inconsistency step will not be further verified, because s_i may not be the true reasoning.

These two situations are critical for preventing reward hacking during RL. Formally, let E_i be the inferences declared in step i, and let $E_i^{\text{ok}} \subseteq E_i$ be the inferences that are active and logically valid. The process reward of step i is defined as:

$$\mathrm{R}_{\mathrm{VERS}}(s_i, n_i) = \begin{cases} 0 & \text{if } s_i \text{ is empty, or inconsistent with } n_i, \\ \dfrac{|E_i^{\text{ok}}|}{|E_i|} & \text{otherwise.} \end{cases} \tag{1}$$

3.2 VERS TRAINING (VERST)

Asking the model to use VERS in reasoning is non-trivial. We design the following two stages for training, which first builds on structure learning and uses the solver's process supervision signal to improve logical reasoning (Figure 2).

3.2.1 REASONING STRUCTURE LEARNING

Even with a detailed prompt, a base instruction model does not generate VERS structures reliably. On-policy rollouts would therefore almost never produce the traces with verifiable structure that later process rewards can score. Thus, we first train the model to produce the above structured reasoning traces.

The idea is to re-write the base model's own reasoning into VERS structures instead of distillation the structured reasoning traces from other models, so the task in this stage is much simpler: only learning VERS, without changing its own reasoning.

We sample step-by-step chain-of-thoughts from the base model, which already contain natural-language reasoning and a final answer, and use a stronger LLM as an annotator. The annotator is asked to stick to the natural-language reasoning without inventing steps. Following the prompt in Appendix F.2, it records only inferences that already occur in the natural-language reasoning, and writes an empty structure when a step has no fact-to-fact inference.

We merge the blocks back onto the original chain-of-thought. To separate these parts, each VERS is written inside `<structure>` tag, and the answer is written inside `<answer>` tag after the last step. Appendix A.2 shows one annotated trace.

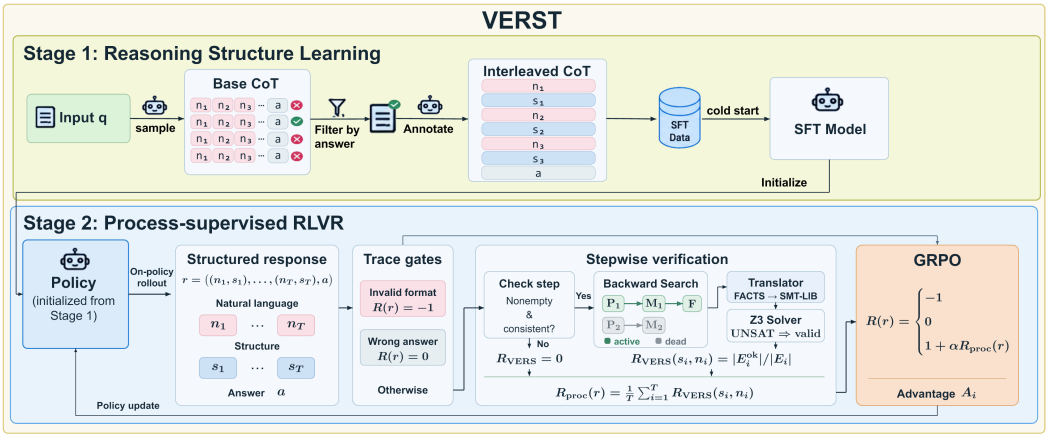

Figure 2: The two stages of VERST. Reasoning Structure Learning interleaves each natural-language step with a verifiable structure of atomic facts and explicit inferences, then fine-tunes the policy on these traces. Process-supervised RLVR scores on-policy rollouts by checking that structure and updates the policy with GRPO.

Performing supervised fine-tuning on this data teaches the model to generate the reasoning with VERS and yields the starting policy for reinforcement learning.

3.2.2 Process-supervised RLVR

In this stage, we focus on enhancing the model's reasoning with the guidance of the verifiable reward. To keep the training process on track, it is critical that the model follows the previously defined format. We also consider the outcome reward so that a valid inference leading to an incorrect answer will not be encouraged.

For a structured response r, its reward considers the following conditions.

Format Validity. Each s_i in r must follow the structure format defined in Section 3.1.1, otherwise the reasoning information cannot be extracted, and the verification cannot run. If any s_i fails to follow the format, r receives a negative reward of -1.

Outcome Reward. The final answer must be correct; otherwise the reward is 0.

Final Reward. The final reward for the whole reasoning trace is the combination of the process reward, format validity and the outcome reward.

$$R(r) = \begin{cases} -1 & \text{if format validity fails,} \\ 0 & \text{if answer of } r \text{ is incorrect,} \\ 1 + \alpha R_{\text{proc}}(r) & \text{otherwise.} \end{cases} \tag{2}$$

where $R_{\text{proc}}(r) = \frac{1}{T}\sum_{i=1}^{T} R_{\text{VERS}}(s_i, n_i)$; and $\alpha > 0$ is a fixed weight. The policy initialized from Section 3.2.1 is trained with the above reward with GRPO, which emphasizes the process reward on VERS to reinforce logically valid inferences.

4 Experiments

4.1 Experimental setup

Datasets. Experiments use five logical reasoning datasets: QMFOL (Zheng et al., 2026), FOLIO (Han et al., 2024), MultiLogiEval (Patel et al., 2024), ProofWriter (Tafjord et al., 2021), and ProverQA (Qi et al., 2025). Training and test sets are drawn from these sources. All training, including Reasoning Structure Learning and Process-supervised RLVR, uses about 1k examples from

Model	QMFOL	FOLIO	MultiLogiEval	ProofWriter	ProverQA	Avg.
Backbone model						
Qwen3-4B-Instruct-2507	57.0	77.6	76.2	89.2	75.8	75.2
Our method						
VERST	78.5*	83.3	91.2	95.3	84.2	86.5
Other methods						
LogicReward (Xu et al., 2026)	68.3	81.4	82.7	91.7	79.0	80.6
FoVer (Kamoi et al., 2026)	74.9	85.3	82.5	95.1	83.2	84.2
Outcome RL	72.1	82.8	84.8	93.6	84.2	83.5
Same-family models						
Qwen3-8B	68.8	81.7	80.1	94.8	84.4	82.0
Qwen3-14B	74.3	84.2	75.3	94.0	81.8	81.9
Qwen3-235B-A22B	79.7	82.8	81.3	93.3	83.7	84.2
Online models						
DeepSeek-V4-Flash-0731	86.7	82.4	78.0	95.3	81.7	84.8
DeepSeek-V4-Pro-0813	90.3	84.3	80.7	96.0	82.7	86.8
GPT-5.6-sol	94.0	77.0	85.7	99.0	83.3	87.8

Table 1: Exact-match accuracy on five logical reasoning benchmarks.

each of the five training splits. Evaluation uses 300 examples from each test split, or the entire split when it has fewer than 300 examples. Sources without an official train/test split are randomly partitioned. The sampled training and test sets do not overlap. Further details of the datasets are given in Appendix B.

Baselines. Baselines include existing models and prior training methods. The existing models are Qwen3-8B, Qwen3-14B, Qwen3-235B-A22B (Yang et al., 2025), DeepSeek-V4-Flash-0731, DeepSeek-V4-Pro-0813 (DeepSeek-AI, 2026), and GPT-5.6-sol (OpenAI, 2026). They are released models used for inference under a standard chain-of-thought prompt. The prior methods are LogicReward (Xu et al., 2026), FoVer (Kamoi et al., 2026), and Outcome RL. All are trained on the same backbone and the same training set as VERST. Implementation details of these methods are given in Appendix C.

Implementation and evaluation. We use Qwen3-4B-Instruct-2507 (Yang et al., 2025) as the backbone model. For each question in the training set, we sample four responses, and then use DeepSeek-V4-Flash-0731 (DeepSeek-AI, 2026) as the annotator to obtain interleaved data with structure. The Reasoning Structure Learning stage then applies full-parameter SFT with LLaMA-Factory (Zheng et al., 2024), while the Process-supervised RLVR stage employs GRPO via the veRL framework (Sheng et al., 2025). Both the translator and the consistency judge, used by the reward computation pipeline, are built on the Qwen3-4B-Instruct-2507. The translator, which we refer to as Translator-4B, is trained with LoRA (Hu et al., 2022) on translations distilled from DeepSeek-V4-Flash-0731. The consistency judge is an off-the-shelf model guided only by a judgment prompt. We set the weight α in Equation 2 to 0.5. At test time, the final answer is extracted and scored by exact match. For reasoning models, reasoning mode is enabled at the default reasoning effort. Further implementation and evaluation settings are given in Appendix D.

4.2 Main results

Table 1 reports exact-match accuracy on the five test sets. The results show the following main points.

The 4B policy matches much larger models. VERST improves the average accuracy of Qwen3-4B-Instruct-2507 by 11.3 points. It outperforms Qwen3-235B-A22B by 2.3 points and DeepSeek-V4-Flash-0731 by 1.7 points. The gain is most impressive on MultiLogiEval, where the 4B policy is even better than DeepSeek-V4-Pro-0813 and GPT-5.6-sol. On the other hand, although the improvement on QMFOL (marked *) is also quite strong (+21.5), it still falls behind on-line models. We notice that among the five benchmarks, QMFOL has the most premises and the deepest nesting,

so a correct structure is harder to emit, and the process reward scored on that structure provides less reliable supervision. Compared to stronger models such as DeepSeek-V4-Pro-0813 and GPT-5.6-sol, the 4B backbone remains weaker on these harder items. The gap may be limited by the capacity of the base model.

VERST outperforms existing reward methods under the same setting. LogicReward, FoVer, and Outcome RL start from the same backbone and the same training data. VERST improves average accuracy by 5.9 points over LogicReward, by 2.3 points over FoVer, and by 3.0 points over Outcome RL. These gains come from a more accurate process reward for on-policy GRPO, which is yielded by the structure of atomic facts and explicit inferences because it is easier to formalize and verify.

5 Analysis

5.1 VERS structure makes Autoformalization easier

The proportion of steps that can be formalized determines how much verification signal the solver can provide. VERS facilitates formalization by organizing each reasoning step into atomic facts and explicit inferences. We therefore compare this proportion on direct formalization of natural language and on formalization based on VERS.

To measure this, we randomly sample 100 pairs of natural language step and its corresponding VERS structure, and formalize the two separately using DeepSeek-V4-Pro-0813 as the translator. The formalizable rate is the proportion of steps written as valid SMT-LIB format, which means the solver can verify them. As shown in Figure 3(A), with DeepSeek-V4-Pro-0813, formalization based on VERS reaches 97.0%, against 82.0% for direct formalization of natural language. The translator no longer has to extract the facts and inferences from the text, and can therefore formalize the step more often. We further replace the translator with Translator-4B, which is trained with LoRA on translations distilled from DeepSeek-V4-Flash-0731 and is actually applied in VERST. It still reaches 93.0% on the VERS structures, while direct formalization of the natural language falls to 17.0%. This fully shows the advantage of the VERS design for formalization.

5.2 Reasoning Structure Learning lays the foundation for RLVR

Reasoning Structure Learning greatly improves format validity while maintaining answer accuracy. As shown in Figure 3(B), the gain in format validity and the preserved accuracy together lay the foundation for the subsequent process-supervised RLVR. This is because the later stage scores each rollout with the solver, on a verifiable structure of atomic facts and explicit inferences. Under the design in Section 3.2.2, extremely low format validity assigns most samples a penalty of -1 and exits the reward calculation early, so no useful supervision signal is produced. Reasoning Structure Learning uses SFT to change the model's behavior so that it emits correct structure. The symbolic solver is therefore called more often and returns more verification signals. Maintaining accuracy means that model performance is almost unchanged. The later stage can then train from this starting point with the additional verification signals.

5.3 Process-supervised RLVR improves reasoning quality

Since outcome-based RLVR rewards only the final answer, it cannot tell whether the reasoning steps are logically valid. VERST introduces a strict process supervision signal to address this problem. Therefore, we evaluate the process quality of its responses and compare it with that of the backbone and Outcome RL.

Process quality is evaluated along four dimensions. Validity asks whether each written inference follows from the premises and earlier steps. Faithfulness asks whether every claim about the problem is stated in the premises or derived from them. Completeness asks whether any link the conclusion depends on is left unstated. Conciseness asks whether the text avoids repetition and passages that do not help the derivation.

To measure this, we randomly sample 20 items from each of the five test sets on which Base, Outcome RL, and VERST all match the gold answer, 100 items in total, and use the same items for all three models. For VERST, we remove the structure from the model output and keep only the

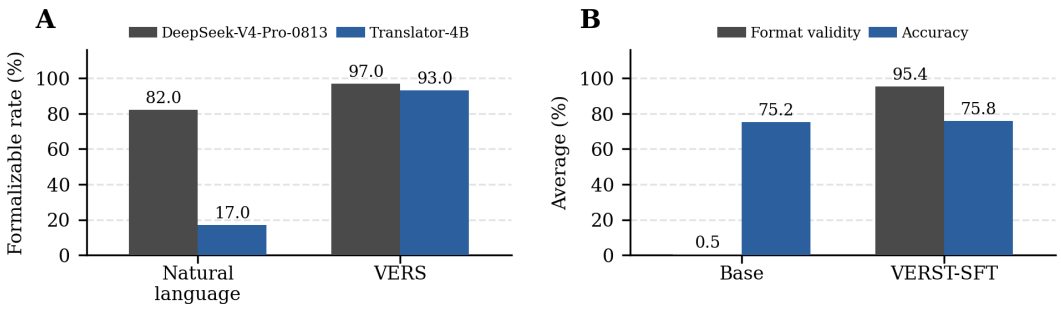

Figure 3: Panel A reports the formalizable rate on 100 paired natural-language steps and their VERS structures, comparing DeepSeek-V4-Pro-0813 and Translator-4B. Panel B reports format validity and answer accuracy of the backbone and the model after Reasoning Structure Learning.

natural-language reasoning; for Outcome RL and Base, we take the entire output, so that all three are evaluated fairly based on the natural-language reasoning. An external judge, DeepSeek-V4-Pro-0813, sees only the remaining natural-language steps after every structure block is removed, and it is not told which model produced the trace. It scores the four dimensions on a fixed 1–5 rubric, writing a short rationale and then an integer. Appendix E lists the rubric and the wrapper prompt used for this judgment.

Model	Process quality				Avg.
	Validity	Faithfulness	Completeness	Conciseness	
Base	4.26	3.74	4.68	2.76	3.86
Outcome RL	4.04	3.90	4.51	2.57	3.76
VERST	4.45	4.02	4.73	3.13	4.08

Table 2: Process scores on 100 correct responses. Avg. is the average of the four dimensions.

As shown in Table 2, VERST has higher process quality than Base (3.86) and Outcome RL (3.76), with an average score of 4.08. The main gains are in validity and conciseness. These gains are attributed to the strict process supervision signal and the design of the lightweight structure. The solver performs a strict logical validity check on the step, which directly improves validity. A backward search from F finds steps irrelevant to the answer and sets their scores to zero. This encourages the model to drop redundant inferences, which raises conciseness. The other dimensions, such as faithfulness and completeness, also improve. Overall, VERST helps the model produce traces with higher process quality, and thus more reliable reasoning on logical tasks.

5.4 Consistency keeps the supervision signal faithful

The VERS structure is a proxy for the natural-language reasoning, so a signal obtained by verifying that structure is trustworthy only when the structure is consistent with the natural language. In VERST, we therefore run a consistency check before calling the solver, so that unfaithful reward signals are excluded.

To show its necessity, we compare VERST with a run that drops it, denoted w/o consistency, keeping all other settings the same. Following the process-quality evaluation in Section 5.3, at each epoch we obtain the model's outputs on the same 100 questions, remove every structure block, score the remaining natural-language reasoning on the same four-dimension rubric, and record the training reward. The process-quality score is the average of those four dimensions.

As shown in Figure 4(A), under VERST the training reward and the independent process quality rise together. Under w/o consistency, the reward rises faster, while the process quality stays nearly flat. The higher reward therefore does not mean better natural-language reasoning: without the consistency check, the solver can credit a structure that does not record the written reasoning, so the

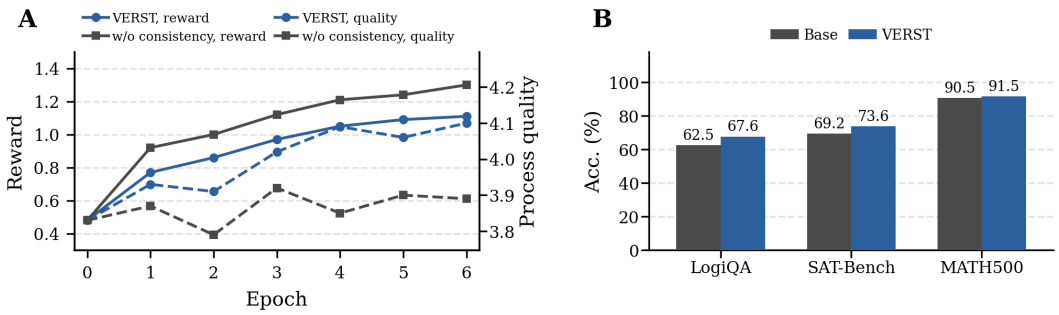

Figure 4: Panel A reports the training reward (solid) and the independent process quality (dashed) over six epochs, comparing VERST with a run that removes the consistency check. Panel B reports accuracy on three out-of-domain datasets.

verification signal is no longer a faithful proxy for that reasoning. The consistency check prevents this mismatch and keeps the process supervision signal trustworthy.

5.5 Generalizability of VERST

To assess whether VERST generalizes to unseen tasks, we evaluate on three out-of-distribution benchmarks: LogiQA (Liu et al., 2020), SAT-Bench (Wei et al., 2025), and MATH500 (Hendrycks et al., 2021; Lightman et al., 2024). As shown in Figure 4(B), the accuracy improves by 5.1 points on LogiQA and by 4.4 points on SAT-Bench. Both are logical reasoning benchmarks outside the training distribution, so these gains show that VERST generalizes to unseen datasets in the same problem domain.

On MATH500, the accuracy improves by 1.0 point, which is relatively low. A likely reason is that the supervision signal in VERST comes from the symbolic solver's verification of entailment among facts, whereas a mathematics answer is determined by calculation, which this signal does not cover. However, we still observe an interesting change in the model's behavior on mathematics problems: it continues to emit an explicit reasoning structure, which makes the derivation interpretable. Appendix A.3 shows this change on one MATH500 problem. The same structure could later host a checker for calculation in place of Z3, and thereby turn this interpretable trace into process supervision for mathematical reasoning.

6 Conclusion

This paper introduced **VERST**, a training framework that interleaves natural-language reasoning with a lightweight verifiable structure VERS and uses a solver to verify this structure during training. Reasoning Structure Learning teaches the policy to interleave each natural-language step with a VERS structure, which consists of atomic facts and explicit inferences and is shown to be easier to formalize. Process-supervised RLVR then assigns a process reward on active, consistent inferences and combines it with an outcome reward in on-policy GRPO. On five logical reasoning benchmarks, a 4B model trained with VERST outperforms Qwen3-235B-A22B, is competitive with DeepSeek-V4-Pro-0813 and GPT-5.6-sol in average accuracy, and surpasses LogicReward, FoVer, and Outcome RL trained under the same setting. Further analysis shows that VERST improves process quality and affects the model's behavior in other reasoning domains.

Future work could proceed in two directions. First, some reasoning still cannot be represented precisely in the present VERS design. A more complex structure would cover more reasoning cases, but it must remain one that the policy can actually learn, and how to trade coverage against learnability is an open design question. Second, the benchmarks we use are mostly first-order problems with explicit premises and a short conclusion, so little extra commonsense or world knowledge is required. Teaching the model to encode such implicit information in the same structure would make the intermediate process stricter and should improve generalization to a broader set of reasoning tasks.

AI USE STATEMENT

As part of the research pipeline, we used Qwen3-4B-Instruct-2507 to generate chain-of-thought traces and DeepSeek-V4-Flash-0731 to annotate their reasoning structures and generate fact-to-SMT-LIB translation pairs for training the translator. During reward computation, we used a translator fine-tuned from Qwen3-4B-Instruct-2507 for fact translation and an off-the-shelf Qwen3-4B-Instruct-2507 for consistency checking. We also used DeepSeek-V4-Pro-0813 for the formalization comparison and for rubric-based evaluation of natural-language reasoning quality. These uses are described in Sections 3 and 4, with implementation and evaluation details, prompts, and evaluation criteria provided in Appendices D, F, and E.

In addition to these experimental uses, we used generative AI tools to aid and polish the writing of this manuscript and to assist with literature retrieval and discovery, including finding related work. The authors take responsibility for the final content of this work, including its text, claims, experimental results, and AI-assisted artifacts.

ETHICS STATEMENT

This work studies logical reasoning using publicly available benchmarks and does not involve human participants or the collection of new personal data. Although symbolic verification provides exact checks on the submitted formal representations, its conclusions depend on the fidelity of those representations to the original reasoning. Model-generated annotations, translations, and judgments may contain errors or biases. The reported improvements therefore do not establish the reliability of the system for safety-critical applications.

REPRODUCIBILITY STATEMENT

We will provide anonymous code, data-construction scripts, and evaluation scripts as supplementary material. Section 3 specifies the reasoning structure, verification procedure, reward, and training stages. Section 4 describes dataset construction and evaluation. Appendix A provides additional structure specifications and examples; Appendix B reports dataset statistics; Appendix C records baseline decoding and reproduction details; and Appendix D documents the implementation and evaluation details. Appendix F provides the interleaved reasoning, annotation, consistency-checking, and translation prompts, while Appendix E provides the rubric and prompt for process-quality evaluation.

REFERENCES

Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The SMT-LIB standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa, 2017. URL https://www.smt-lib.org.

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, pp. 337–340. Springer, 2008. doi: 10.1007/978-3-540-78800-3_24. URL https://link.springer.com/chapter/10.1007/978-3-540-78800-3_24.

DeepSeek-AI. DeepSeek-V4-Pro GA release, August 2026. URL https://api-docs.deepseek.com/news/news260813/. Official release announcement. Accessed 2026-09-25.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and Huzefa Rangwala. VeriCoT: Neuro-symbolic chain-of-thought validation via logical consistency checks. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zHuV3Vatov.

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, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22017–22031. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.1229. URL https://aclanthology.org/2024.emnlp-main.1229/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, Wenpeng Yin, and Rui Zhang. Efficient PRM training data synthesis via formal verification. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8246–8265. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.403. URL https://aclanthology.org/2026.findings-acl.403/.

Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. LAMBADA: Backward chaining for automated reasoning in natural language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6547–6568. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.361. URL https://aclanthology.org/2023.acl-long.361/.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022. doi: 10.52202/068431-1613. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023. URL https://arxiv.org/abs/2307.13702.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 3622–3628, 2020. doi: 10.24963/ijcai.2020/501. URL https://www.ijcai.org/proceedings/2020/501.

Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002. doi: 10.1007/3-540-45949-9. URL https://link.springer.com/book/10.1007/3-540-45949-9.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 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*, pp. 5153–5176. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.313. URL https://aclanthology.org/2023.emnlp-main.313/.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, July 2026. URL https://openai.com/index/gpt-5-6/. Model family announcement; includes GPT-5.6 Sol. Accessed 2026-09-25.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.248. URL https://aclanthology.org/2023.findings-emnlp.248/.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. Multi-LogiEval: Towards evaluating multi-step logical reasoning ability of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20856–20879. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.1160. URL https://aclanthology.org/2024.emnlp-main.1160/.

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. Large language models meet symbolic provers for logical reasoning evaluation. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=C25SgeXWjE.

Sergio Servantez, Joe Barrow, Kristian Hammond, and Rajiv Jain. Chain of logic: Rule-based reasoning with large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2721–2733. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.159. URL https://aclanthology.org/2024.findings-acl.159/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297. Association for Computing Machinery, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.317. URL https://aclanthology.org/2021.findings-acl.317/.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 74952–74965, 2023. doi: 10.52202/075280-3275. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ed3fea9033a80fea1376299fa7863f4a-Abstract-Conference.html.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022. URL https://arxiv.org/abs/2211.14275.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Anjiang Wei, Yuheng Wu, Yingjia Wan, Tarun Suresh, Huanmi Tan, Zhanke Zhou, Sanmi Koyejo, Ke Wang, and Alex Aiken. SATBench: Benchmarking LLMs' logical reasoning via automated puzzle generation from SAT formulas. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33832–33849. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1716. URL https://aclanthology.org/2025.emnlp-main.1716/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Chengyao Wen, Qiang Cheng, Shaofei Wang, Zhizhen Liu, Deng Zhao, and Lei Liang. LogicThinker: Teaching large language models to think more logically. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 12955–12969. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.696. URL https://aclanthology.org/2025.findings-emnlp.696/.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 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)*, pp. 13326–13365. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.720. URL https://aclanthology.org/2024.acl-long.720/.

Jundong Xu, Hao Fei, Meng Luo, Qian Liu, Liangming Pan, William Yang Wang, Preslav Nakov, Mong-Li Lee, and Wynne Hsu. 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)*, pp. 3052–3075. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.153. URL https://aclanthology.org/2025.acl-long.153/.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. LogicReward: Incentivizing LLM reasoning via step-wise logical supervision. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IRhYVOKFe0.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Xinyi Zheng, Ling Shi, Tianlong Yu, Yongxin Zhao, Lorenz Goette, and Kailong Wang. QMFOL: Benchmarking large language model reasoning via quantifiable monadic first-order logic test case generation. *arXiv preprint arXiv:2606.20227*, 2026. URL https://arxiv.org/abs/2606.20227.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 400–410. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-demos.38. URL https://aclanthology.org/2024.acl-demos.38/.

A SUPPLEMENTARY DETAILS OF VERS

A.1 FURTHER DETAILS OF THE STRUCTURE DESIGN

Section 3.1.1 defines the fact and inference lines of VERS. Here we specify the identifier types and the well-formedness constraints used by the structure check before process reward is computed.

Identifier types. Identifiers are global across steps and belong to three types:

- P_j: a given premise, or background information, with no incoming inference;
- M_j: an intermediate conclusion, which must be the target of at least one inference;
- F: the unique claim about the final answer, defined only in the last step T.

Well-formedness constraints.

1. A non-empty s_i must contain at least one inference.
2. Each line is either a fact ($id\colon \varphi$) or an inference ($id_1, \ldots, id_k \Rightarrow id_t$). The two must not be combined on one line, and both sides of $\Rightarrow$ are identifiers only.
3. Every identifier is introduced at most once and must be defined before it is cited.
4. A P_j fact cannot be the target of any inference
5. A M_j fact must be the target of at least one inference before it appears in the source set of any inference.
6. F is defined in the last block s_T and nowhere else, and at least one inference has F as its target.

A.2 ANNOTATED TRACE

Below is one ProofWriter item after annotation. The natural-language steps are the base model's chain-of-thought, and each `<structure>` block is a VERS structure produced by the annotator. Steps with no fact-to-fact derivation keep an empty block. The last step defines F, and the answer matches the gold label.

Premise:
Charlie is quiet. Dave is young. Erin is smart. Fiona is smart. If someone is red then they are young. Smart people are red.

Question:
Fiona is young.

Step 1:
Identify the given premises and logical relationships.

Premises:
– Charlie is quiet. (This seems irrelevant to the question about Fiona.)
– Dave is young.
– Erin is smart.
– Fiona is smart.

Additional logical statements:
– If someone is red, then they are young. (Red –> Young)
– Smart people are red. (Smart –> Red)
<structure>
</structure>

Step 2:
Use the logical statements to derive properties of Fiona.

Fiona is smart.
From ”Smart people are red”, Fiona is red.
From ”If someone is red, then they are young”, Fiona is young.
<structure>
P1: smart(Fiona)
P2: forall x smart(x) −> red(x)
P3: forall x red(x) −> young(x)
M1: red(Fiona)
M2: young(Fiona)
P1,P2 => M1
M1,P3 => M2
</structure>

Step 3:
Therefore, based on the chain of logic above, Fiona is young.
<structure>
</structure>

Step 4:
The question is: ”Fiona is young.”
This conclusion follows directly from the premises and the logical implications.
There is no contradiction or missing information that would make this uncertain. The chain of reasoning is valid and complete.
<structure>
F: Fiona is young.
M2 => F
</structure>

<answer>True</answer>

A.3 BEHAVIOR CHANGE ON MATH500

Below are two responses sampled from the backbone and from the model trained with VERST on the same MATH500 problem. Both reach the correct answer 180. The backbone uses free-form chain-of-thought, while the VERST-trained model interleaves VERS blocks that make the tangent–radius right angles and the quadrilateral angle sum explicit.

Problem.

In circle J, HO and HN are tangent to the circle at O and N. Find the number of degrees in the sum of m∠J and m∠H.

Backbone.

Step 1: Understand the given geometric setup.
We are given a circle with center J. Points O and N lie on the circle, and segments HO and HN are tangent to the circle at O and N, respectively. So, HO and HN are tangent segments from point H to the circle.

Step 2: Use properties of tangents from a common external point.
From a point outside a circle, the two tangent segments to the circle are congruent. So, HO = HN . Also, the radius to the point of tangency is perpendicular to the tangent line. That is:
– JO ⊥ HO, so ∠JOH = 90°
– JN ⊥ HN, so ∠JNH = 90°

Step 3: Consider triangle JON.
We are asked to find the sum m∠J + m∠H, where:
– ∠J is the angle at the center J, which in triangle JON is ∠OJN

− ∠H is the angle at point H, which is ∠OHN
But note: the diagram shows H connected to O and N, and J connected to O and N, forming quadrilateral HONJ. We are to find m∠J + m∠H, which likely refers to ∠OJN (at center) and ∠OHN (at point H).
So we want ∠OJN + ∠OHN.

Step 4: Use the quadrilateral angle sum.
Points H, O, J, N form a quadrilateral HONJ. The sum of the interior angles of any quadrilateral is 360°. So:
∠H + ∠O + ∠J + ∠N = 360°
But we have:
− ∠O is ∠HOJ, which is 90° (tangent perpendicular to radius)
− ∠N is ∠HNJ, which is 90°
So:
∠H + 90° + ∠J + 90° = 360°
∠H + ∠J + 180° = 360°
∠H + ∠J = 180°

Step 5: Conclusion
Thus, the sum of m∠J and m∠H is 180°.
<answer>180</answer>

VERST.

Step 1:
Identify the given elements in the diagram. We have a circle with center J. Points O and N are points of tangency on the circle. Lines HO and HN are tangent to the circle at O and N, respectively. So, HO and HN are tangent segments from point H to the circle.
<structure>
P1: HO is tangent to circle J at O.
P2: HN is tangent to circle J at N.
M1: HO ⊥ JO and HN ⊥ JN.
P1, P2 => M1
</structure>

Step 2:
Use the property that tangent segments from a common external point to a circle are equal in length. So, HO = HN. This implies that triangle HON is isosceles with HO = HN.
<structure>
P3: HO and HN are tangent segments from H to circle J.
M2: HO = HN.
P3 => M2
</structure>

Step 3:
Since HO and HN are tangent to the circle at O and N, the radii JO and JN are perpendicular to the tangents at the points of tangency. Therefore, ∠JOH = 90° and ∠JNH = 90°.
<structure>
M3: ∠JOH = 90°.
M1 => M3
M4: ∠JNH = 90°.
M1 => M4
</structure>

Step 4:
Consider quadrilateral HONJ. It has angles at O and N each equal to 90°. So, the sum of the angles at O and N is 90° + 90° = 180°. The sum of the interior angles of any quadrilateral is 360°. Therefore, the sum of the remaining two angles, ∠H and ∠J, must be 360° − 180° = 180°.

<structure>
P4: Quadrilateral HONJ has interior angles ∠JOH, ∠JNH, ∠H, and ∠J.
M5: ∠JOH + ∠JNH = 180°.
M3, M4 => M5
M6: ∠H + ∠J = 180°.
P4, M5 => M6
</structure>

Step 5:
The problem asks for the sum of m∠J and m∠H. From the above, this sum is 180°.
<structure>
F: m∠J + m∠H = 180°.
M6 => F
</structure>

<answer>180</answer>

B DATASET DETAILS

In-distribution. Training and main evaluation use five logical reasoning benchmarks. Each example gives explicit premises, a question, and a short conclusion. QMFOL (Zheng et al., 2026), FOLIO (Han et al., 2024), ProofWriter (Tafjord et al., 2021), and ProverQA (Qi et al., 2025) are labeled True, False, or Unknown, and MultiLogiEval (Patel et al., 2024) is labeled yes or no. All training, including Reasoning Structure Learning and Process-supervised RLVR, uses about 1k examples from each training split. Evaluation uses 300 examples from each test split, or the entire split when it is smaller. Sources without an official train/test split are randomly partitioned, and the sampled training and test sets do not overlap.

Out-of-distribution. For generalization, we further evaluate on three held-out benchmarks that are never used in training: LogiQA (Liu et al., 2020), SAT-Bench (Wei et al., 2025), and MATH500 (Hendrycks et al., 2021; Lightman et al., 2024). LogiQA and SAT-Bench are logical reasoning datasets outside the training sources; MATH500 is a mathematics benchmark used to probe behavior transfer beyond formal entailment checking. Table 3 reports the number of examples.

Benchmark	Train	Test
In-distribution		
QMFOL	1,000	300
FOLIO	1,003	204
MultiLogiEval	1,000	200
ProofWriter	1,000	300
ProverQA	1,000	300
Total	5,003	1,304
Out-of-distribution		
LogiQA	—	300
SAT-Bench	—	400
MATH500	—	500

Table 3: Number of examples in the in-distribution and out-of-distribution sets.

C BASELINE DETAILS

This appendix records how the models and methods in Table 1 are evaluated or reproduced. Unless noted otherwise, all systems use the same in-distribution test splits as Section 4, and final answers are scored by exact match.

C.1 Released models

Table 4 lists the decoding settings for the released checkpoints in Table 1. All runs use a standard chain-of-thought prompt, a maximum generation length of 8192 tokens, and four samples per question; we report the mean exact-match accuracy over the four samples. The predicted label is extracted from the answer span and compared with the gold label after light normalization of common aliases (e.g., yes/no and True/False).

Model	Thinking mode	Reasoning effort	Temperature
Qwen3-4B-Instruct-2507 (Yang et al., 2025)	—	—	0.6
Qwen3-8B (Yang et al., 2025)	No	—	0.6
Qwen3-14B (Yang et al., 2025)	No	—	0.6
Qwen3-235B-A22B (Yang et al., 2025)	Yes	—	0.6
DeepSeek-V4-Flash-0731 (DeepSeek-AI, 2026)	Yes	default	0.6
DeepSeek-V4-Pro-0813 (DeepSeek-AI, 2026)	Yes	default	0.6
GPT-5.6-sol (OpenAI, 2026)	Yes	default	0.6

Table 4: Decoding settings for released models.

C.2 Prior training methods

We also compare reward-based training methods under a matched setting: the same backbone (Qwen3-4B-Instruct-2507) and the same in-distribution training set as VERST. At test time they follow the same decoding and scoring protocol as the released models (T=0.6, four samples, mean exact-match accuracy), with a chain-of-thought prompt and without requiring VERS tags.

LogicReward (Xu et al., 2026) is reproduced on our data following the official pipeline. We sample four responses per training question from the backbone, use DeepSeek-V4-Flash-0731 to formalize the steps, and score each response with Isabelle under the original reward procedure. From these labeled responses we build training data on our training set: the highest-scoring response of each question is used for SFT, and the highest- and lowest-scoring responses form a preference pair for DPO. Both stages use LoRA (rank 64, α=128) with learning rate 3×10^{-5}, cutoff 8192, and effective batch size 64. SFT runs for 1 epoch and DPO for 3 epochs with preference β=0.1.

FoVer (Kamoi et al., 2026) is trained with GRPO on the base Instruct model under a plain CoT prompt. Each response is split into steps and scored by the released process reward model Qwen-2.5-7B-FoVer-PRM-2026; the process reward is the minimum step score, and it is assigned only when the final answer is correct. We use veRL with effective batch size 128, group size 8, 6 epochs, learning rate 5×10^{-6}, KL coefficient 0.001, and process weight α=0.5.

Outcome RL uses the same GRPO setup as VERST (Appendix D), except that the process term is removed. The reward is -1 when no answer can be extracted, 0 when the answer is wrong, and 1 when it matches the gold label. It starts from the base Instruct model under a CoT prompt.

D Implementation and evaluation details

D.1 Hyperparameters of VERST

Table 5 lists the hyperparameters of the two stages.

D.2 Translator and consistency judge

Translator-4B is trained with LoRA (rank 64) on fact-to-SMT-LIB pairs distilled from DeepSeek-V4-Flash-0731; the training prompt is that of Appendix F.4. The consistency judge is the off-the-shelf backbone with the prompt in Appendix F.3.

Setting	Value
Backbone	Qwen3-4B-Instruct-2507
SFT framework	LLaMA-Factory
SFT type	full-parameter
SFT data size	≈11.5k traces annotated by DeepSeek-V4-Flash-0731
SFT epochs	1
SFT learning rate	1×10^{-5}
SFT cutoff	8192 tokens
GRPO framework	veRL
GRPO type	full-parameter
GRPO data size	≈5.0k prompts
GRPO epochs	6
GRPO group size	8
GRPO learning rate	5×10^{-6}
KL coefficient	0.001
Process weight α	0.5
Max prompt length	2048
Max response length	8192

Table 5: Training hyperparameters.

D.3 Evaluation

At test time, VERST uses the interleaved VERS prompt from training (Appendix F.1). Decoding uses temperature 0.6, a maximum generation length of 8192 tokens, and four samples per question; we report the mean exact-match accuracy over the four samples.

E Process-quality evaluation

E.1 Judge prompt

Below is the wrapper prompt. {`rubric`} is filled with that dimension's block from Appendix E.2.

You grade one dimension of a natural–language logical derivation.
The final answer is already known to be correct. Do not grade whether the answer label is right or wrong.
Score only this dimension. Do not give an overall score.
Use only the rubric below.

{rubric}

Premises:
{premise}

Question:
{question}

Answer–label instruction:
{answer_instruction}

Derivation:
{reasoning}

Write a brief rationale that points to concrete steps. On the last line write exactly:
Score: <integer from 1 to 5>

E.2 RUBRIC

The judge scores one dimension at a time. The wrapper prompt is given in Appendix E.1; `{rubric}` is filled with one of the four blocks below.

Validity.

```
Dimension: Validity
Judge only sentences that are written. A written inference is valid when its claim follows from
the premises and earlier written steps, and invalid when it does not follow or contradicts them.

Boundary: Do not treat a missing step as a validity error. If every written sentence follows,
validity stays high even if a necessary link was never stated; that gap is completeness. A written
jump such as 'A, therefore C' is a validity error when C does not follow from A. Let N be the
number of written sentences that do not follow and that the conclusion depends on. The
conclusion depends on a sentence when deleting it leaves the remaining written steps unable to
support the conclusion. A contradiction on a claim the conclusion uses is one of these sentences,
not a separate case. A sentence the conclusion does not use does not increase N.

Score rubric:
1. N is three or more.
2. N is exactly two. No other invalid sentence is one the conclusion uses.
3. N is exactly one. Example: the text says 'it is white, therefore it has antennae' where the
premise only gives 'white −> purple or antennae', and a later step uses antennae.
4. N is zero, and at least one written sentence does not follow or contradicts another step. The
conclusion does not depend on that sentence.
5. N is zero, every written inference follows from the premises or earlier written steps, and the
written steps contain no contradiction.
```

Faithfulness.

```
Dimension: Faithfulness
Judge claims about this problem: its people, objects, and relations. A claim is unsupported when
it states a fact about this problem that the premises do not give and no earlier step has derived.

Boundary: Count unsupported facts about this problem whether or not a later step uses them.
One fact is one claim that adds an entity, a relation, or a constraint the premises do not state and
no earlier step has derived. Familiar logical and mathematical facts that do not depend on this
problem are not counted. '2 is even', the meaning of 'all' and 'or', and rewriting a stated
conditional as its contrapositive are not unsupported facts. A wording that adds no entity, relation
, or constraint is not an unsupported fact.

Score rubric:
1. Three or more unsupported facts about this problem appear.
2. Exactly two unsupported facts about this problem appear.
3. Exactly one unsupported fact about this problem appears.
4. No unsupported fact appears. At least one wording goes beyond the premises and adds no
entity, relation, or constraint.
5. Every claim about this problem is stated in the premises or derived from them, and no wording
 goes beyond the premises. Familiar logical or mathematical facts that do not depend on this
problem are allowed.
```

Completeness.

```
Dimension: Completeness
Judge only what is absent. A necessary link is missing when the reader must supply an inference
that is never stated in order to connect the premises to the conclusion.
```

Boundary: Do not treat a false or non−following written sentence as a completeness error. That is validity. Use the same test as validity: the conclusion depends on a missing link when writing that link down and then deleting it would leave the remaining steps unable to support the conclusion. Count gaps, not imaginary intermediate sentences. One place where consecutive written claims do not connect is one gap, even if several sentences could be inserted there. Separate places are separate gaps. An omission that fails this test does not count.

Score rubric:
1. Three or more gaps that the conclusion depends on are never stated.
2. Exactly two gaps that the conclusion depends on are never stated.
3. Exactly one gap that the conclusion depends on is never stated, and closing it takes more than a one−step reading of a single written sentence.
4. Exactly one gap, and the conclusion depends on it, and closing it is a one−step reading of a premise or earlier step already written. The reader adds no new fact about this problem.
5. No gap that the conclusion depends on is missing.

Conciseness.

Dimension: Conciseness
Whether the text stays on the derivation, without repeated restatement or passages that do not help.

Boundary: A brief restatement that sets up the next inference is not counted. An extra restatement is one sentence that repeats a point already used and does not set up the next inference. A passage is a run of more than two sentences that does not bear on the derivation. A detour is a run of one or two such sentences. Sentences inside a passage or a detour are not also counted as extra restatements. If more than half the sentences do not bear on the derivation, score 1 and do not apply the lower counts.

Score rubric:
1. More than half the sentences do not bear on the derivation.
2. At most half the sentences are unrelated, and one of the following holds: extra restatements occur in three or more sentences, or there is more than one detour or passage.
3. There are exactly two extra restatement sentences, or exactly one passage of more than two sentences. Level 1 and level 2 do not apply.
4. There is exactly one extra restatement sentence, or exactly one detour of at most two sentences , and there is no passage of more than two sentences.
5. There is no extra restatement, no detour, and no passage. A brief restatement that sets up the next inference is acceptable.

F PROMPTS

Appendix F.1 is the interleaved VERS prompt used in Reasoning Structure Learning and at test time. Appendix F.2 asks an annotator to write `<structure>` on an existing chain-of-thought. Appendix F.3 judges whether s_i is consistent with n_i. Appendix F.4 translates active facts into SMT-LIB.

F.1 INTERLEAVED REASONING

System.

You are a careful reasoner. Solve step by step.

For every step use this layout:

Step N:
<natural−language reasoning>

<structure>
...
</structure>
Step N+1:
...
<answer>...</answer>

Inside <structure>, only:
<id>: <content> (Fact; ids are P... / M... / F)
<id1>,<id2>,... => <id> (Inference; ids only; use −> in formulas)

Requirements:
− P... = given / assumption / background (no incoming =>); M... = median information derived by =>; F = statement about final answer (only in the last step: F: ... and ... => F).
− Structure must match the NL's derivations (not every sentence). Non−empty structure needs >=1 Inference; only define Facts that appear on some =>.
− Fact content: atomic claims −−− prefer symbols when formalizable; meta / common sense / case assumptions in plain English; no plans or snake_case fake−symbols.
− Empty <structure></structure> when the step has no Fact−>Fact derivation (setup, restatement, planning, option talk).
− Define each id once before use; later steps only cite. After the last step write <answer>...</answer> and nothing else.

User.

Premise:
{premise}

Question:
{question}

{answer_instruction}

F.2 STRUCTURE ANNOTATION

System.

Annotate an existing step−by−step solution. Given a problem, numbered NL steps, and a final answer, emit one structure block per step. Do not rewrite the NL. Do not invent steps. Output only the blocks below.

Output
‘‘‘
Step N:
<structure>
...
</structure>
‘‘‘
Same indices as the input. Empty body is allowed (and common):
‘‘‘
Step N:
<structure>
</structure>
‘‘‘

Lines
− Fact: ‘<id>: <content>‘
− Inference: ‘<id1>,<id2>,... => <id>‘ (rhs = id only; in formulas use −> not =>)

Ids
– ‘P...‘ −−− no incoming Inference (given / assumption / background / common sense)
– ‘M...‘ −−− concluded by some Inference
– ‘F‘ −−− exactly one terminal answer claim; always in the ∗∗last∗∗ step as ‘F: ...‘ and ‘... => F‘

Facts
One atomic piece of information. Only Facts on either side of ‘=>‘.

Form:
– Prefer symbolic / formal when formalizable: ‘furry(Dave)‘, ‘forall x green(x) −> big(x)‘, cases , etc.
– Meta / non−formula claims −> short plain English.
Good: ‘The premises do not establish any conclusion disjunct for every herbivore.‘
Bad: ‘premises_do_not_establish_...‘ (snake_case fake−symbols are not formal)

Allowed:
– meta−facts: what the question asks; what the premise does / does not mention; whether a claim is determined / undetermined by the given info
– implicit common sense: unspoken bridges the NL relies on (category exclusions, word meanings, ”A and B cannot both hold”, etc.) −−− only if a derivation uses them
– assumptions and cases: ‘Assume ...‘; ‘Case A: ...‘; ‘Case B: ...‘; ‘Suppose ...‘

Forbidden:
– plans: ‘We will now check...‘; ‘Next we analyze...‘; ‘First identify...‘
– roadmaps: ‘There are three steps...‘; ‘Consider the following cases in turn...‘ (without stating a case assumption as content)
– decorative restatement: repeating an earlier Fact in new wording without adding information or feeding an Inference

Inference
‘srcs => target‘ records that the target Fact follows from the source Facts.
– Prefer strict logical consequence when Facts are formalizable; otherwise a clear justificatory link the NL actually uses.
– Both sides are Fact ids only −−− never put content after ‘=>‘, never combine Fact definition with Inference on one line.
– Only Facts derive Facts; every source and the target must already be defined (in this step or earlier).
– Write an Inference only when the NL performs a derivation; chain Inferences across steps until the last step reaches ‘F‘.
– Do not care about the correctness of the derivation; only care about the consistency between Structure and NL.

Critical Rules
1. ∗∗Structure must contain at least one Inference∗∗: A non−empty ‘<structure>‘ must contain at least one Inference. If the step has no Fact−>Fact derivation (and is not the final ‘... => F‘), leave the body empty −−− pure restatement, premise listing, planning, or option talk with no derivation −> empty. Only define Facts that appear on some Inference (as source or target); when an Inference happens, define any still−undefined Facts it needs, then write the Inference. Do not dump unused premises just because the NL mentioned them.

2. ∗∗Consistency between Structure and NL∗∗: Align Structure with the NL’s ∗∗logical derivations∗∗, not with every NL sentence.
– ∗∗Inferences∗∗ −−− every derivation the NL actually relies on should appear as some ‘srcs => target‘; do not invent Inferences the NL does not use.
– ∗∗Facts∗∗ −−− only Facts that appear on those Inferences need exist. NL text that never feeds an Inference needs no Fact.
– ∗∗Grain∗∗ −−− match how finely the NL unfolds derivations (jump with jumps; keep intermediate Inferences the NL actually steps through).

3. **Define Facts before using them**: Write ‘id: content‘ before any Inference cites that id. Ids are global across the whole annotation: define each id once; later steps only cite; do not redefine. Earlier Facts need not be repeated in the current block.

User.

Premise:
{premise}

Question:
{question}

Steps (index + NL text):
{steps}

Answer: {answer}

Emit Step N: <structure>...</structure> for every index. Empty unless the step has an Inference. Only Facts on some =>. Align with NL derivations (not every NL sentence). Symbolic when formalizable; meta in plain English (no snake_case fakes). F only in the last step.

F.3 CONSISTENCY CHECK

System.

Given a single step of reasoning with a non−empty <structure> block and a NL, judge the consistency of <structure> block and the NL.

You see:
− prior Fact ids this step’s Structure cites on ‘‘=>‘‘ (may be none)
− this step’s ‘<structure>‘ block, include several lines of:
  − Fact: ‘id: content‘
  − Inference: ‘id1, id2, ... => id‘
− this step’s natural−language (NL) text

Judge consistency on three aspects:
− **Inferences <−> NL derivations** (Bidirectional)
  − each ‘srcs => target‘ in Structure must match a derivation relation stated or clearly used in the NL.
  − each derivation the NL relies on in this step must appear as some ‘=>‘.
  − Derivation includes causal/grounding links, not only strict entailment.
− **Fact ids −> NL content** (Unidirectional, only Facts that appear on Inference)
  − Do **not** require every claim in the NL to be a Fact −−− unused NL detail may be omitted.
  − Facts may be premises, implicit commonsense, meta−level claims, etc., when an Inference uses them.
  − Each Fact on some Inference must be semantically equivalent to the corresponding information used in the NL.
− **Grain** (Bidirectional)
  − The granularity of the Structure must match the granularity of the NL.
  − Including the derivation chain length, the number and atomicity of Facts used per Inference, etc.
If all three aspects are consistent, return ‘1‘; otherwise return ‘0‘.

Output (strict)
Reply with exactly one token: ‘1‘ or ‘0‘.
No other text, no reasons, no markdown.

User.

Score consistency for this single step (output only 1 or 0).

Prior Fact ids used in this step:
{prior_facts}

Step {index} <structure> block:
{structure}

Step {index} NL:
{nl}

F.4 FACT TRANSLATION

Translate each natural−language / formal statement below into a Z3 SMT−LIB assertion.
Output structure:

<translate>
;; −− Declarations −−
(declare−sort Entity 0)
(declare−const ...)
(declare−fun ...)

;; −− Assertions −−
;; P1: description
(assert ...)
;; M2: meta commentary
; UNFORM
</translate>

Rules:
(1) Every listed statement gets either an (assert ...) or ‘; UNFORM‘. Do NOT skip any id.
(2) Use forall/exists when the statement says ”all”/”every”/”some”/forall/exists.
(3) If a statement cannot be translated into a precise, unambiguous object−level assertion, write ‘; UNFORM‘ after its comment.
(4) Keep predicate/entity names short, valid SMT identifiers, and consistent across statements.
(5) Translate F when it appears in the statement list (object−level claim / conclusion Fact). Do not encode inference edges −−− only the listed statements.
(6) SMT implication is ‘=>‘ (not Unicode −>); use it only between Bool terms.

UNFORM (meta−discourse −−− do NOT invent free−floating booleans):
− Entailment / contradiction / neutral labels, or ”premise vs hypothesis” relationship commentary .
− Whether the premise supports, contradicts, or mentions something; answer−choice labels (” correct answer is ...”).
− Abstract reasoning about the argument itself rather than facts about entities.
− Example UNFORM: ”The premise does not entail the hypothesis”, ”Relationship is not neutral ”.
− Still translate object−level M−nodes: concrete derived facts (e.g. ”Dumpus(Max)”, ”Dog(x) /\ Carries(x, Object)”).

SMT syntax (must parse in Z3):
− Declare every symbol used in assertions: ‘(declare−const x Entity)‘, ‘(declare−fun Pred (Entity) Bool)‘.

– Keep each predicate's arity fixed everywhere; do not call a unary predicate with two arguments .
– Do not use node ids (P1, M2, ...), English words (Premise, Claim, Hypothesis), or hyphenated pseudo−facts
(e.g. 'premise−entails−hypothesis') as undeclared constants.
– Prefer '(declare−const ruralSetting Entity)' over treating a place name as an unapplied function .

−−− Statements −−−

{nodes_text}