Probabilistic Soundness Guarantees in LLM Reasoning Chains

Weiqiu You **Anton Xue** **Shreya Havaldar** **Delip Rao** **Helen Jin**

Chris Callison-Burch **Eric Wong**

University of Pennsylvania

Abstract

In reasoning chains generated by large language models (LLMs), initial errors often propagate and undermine the reliability of the final conclusion. Current LLM-based error detection methods often fail to detect propagated errors because earlier errors can corrupt judgments of downstream reasoning. To better detect such errors, we introduce Autoregressive Reasoning Entailment Stability (ARES), a probabilistic framework that evaluates each reasoning step based solely on previously-verified premises. We find that ARES can reliably detect propagated reasoning errors that other baselines fail to find with probabilistic guarantees. [1]

1 LLMs Often Make Reasoning Errors

Large Language Models (LLMs) often produce reasoning chains with errors that propagate, undermining the final outputs [Huang et al., 2025, Lyu et al., 2024]. An error can be *ungrounded* statements, *invalid* derivations, or *propagated* errors. For example, deriving $x = 5$ from $5x = 9x - 20$ is logically valid, but can be a propagated error if the premise $5x = 9x - 20$ differs from the context. These errors compromise the reliability LLMs in high-stakes domains [Agarwal et al., 2024, Chen and Mueller, 2023].

Current error detection methods, such as LLM judges [Tyagi et al., 2024, He et al., 2025] and Process Reward Models (PRMs) [Lightman et al., 2023], typically attempt to identify all errors at once. However, these methods often fail to detect propagated errors when they are distracted by invalid premises [He et al., 2025, Turpin et al., 2023, Dhuliawala et al., 2023].

To address this issue, we introduce Autoregressive Reasoning Entailment Stability (ARES), a probabilistic framework inspired by human reasoning, which assesses each reasoning step inductively [Johnson-Laird, 2010, Mukherjee et al., 2025]. It determines a step's soundness by calculating its entailment probability based *only* on the set of preceding, sound claims, as shown in Figure 1. Empirically, we find that our method accurately certifies both sound and unsound reasoning steps. It particularly excels in long chains prone to error propagation, where existing approaches fail.

2 Soundness in Reasoning Chains

We aim to identify and certify errors within LLM-generated chain-of-thought (CoT) reasoning. This section formalizes reasoning chains by defining their constituent claims (Section 2.1), introducing probabilistic entailment between claims (Section 2.2), and establishing a notion of soundness (Section 2.3).

[1] Correspondence to `weiqiuy@seas.upenn.edu`. Code is at `https://github.com/fallcat/ares`.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

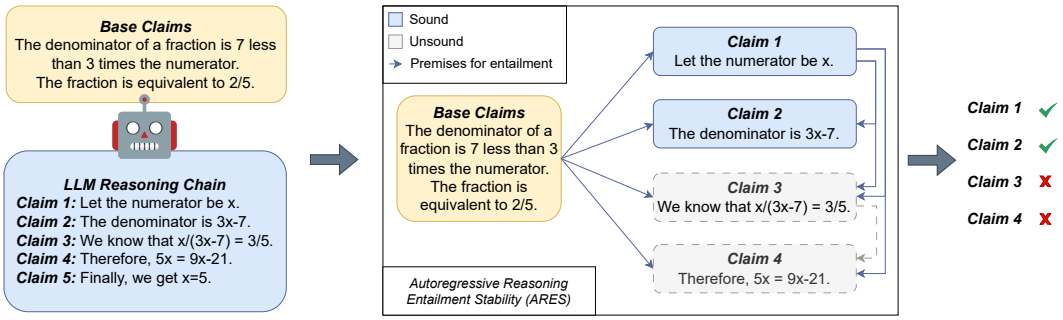

Figure 1: **Autoregressive Reasoning Entailment Stability (ARES)** When we verify a reasoning chain, we can break it down into base claims and derived claims. ARES checks each derived claim step-by-step, using previously-verified claims as premise at a probability.

2.1 Claims and Sequences of Claims

A reasoning chain is a sequence of claims, where a claim is the assertion of a proposition. For instance, "The denominator is $3x - 7$" is a claim, as is "We know that $\frac{x}{3x-7} = \frac{2}{5}$". The granularity of claims is domain-dependent, ranging from atomic statements to entire proofs.

Formally, we let $\mathcal{C}$ denote the set of all possible claims and $\mathcal{C}^\star$ represent the set of all sequences of claims. An example of such a sequence is:

$$\big(\text{"Let the numerator be } x\text{"}, \text{"The denominator is } 3x - 7\text{"}, \text{"We know that } \tfrac{x}{3x-7} = \tfrac{2}{5}\text{"}\big) \in \mathcal{C}^\star$$

2.2 Probabilistic Entailment of Claims

To capture logical entailment in natural language, we use a probabilistic entailment model, motivated by the inherent ambiguity in language [Zadeh, 2008, Yu et al., 2024]. Formally, a entailment model $\mathcal{E} : \mathcal{C}^\star \times \mathcal{C} \to [0, 1]$ accepts a premise sequence $P \in \mathcal{C}^\star$ and a hypothesis $H \in \mathcal{C}$, where $\mathcal{E}(P, H)$ is the probability that P entails H. For instance, given the premise $P =$ $\big($"Sarah put on her running shoes.", "She stretched."$\big)$ and hypothesis $H =$ "Sarah is going for a run.", a entailment model might output $\mathcal{E}(P, H) = 0.85$. This probabilistic approach generalizes classical Boolean logic.

2.3 Reasoning Chains and Soundness

We model an LLM's CoT output as a *reasoning chain*: $(C_1, \ldots, C_n, C_{n+1}, \ldots, C_{n+m})$, where $C_1, \ldots, C_n$ are given *base claims* (context) and $C_{n+1}, \ldots, C_{n+m}$ are autoregressively generated *derived claims*. We assume base claims are factually accurate and focus on whether each derived claim is soundly inferred from preceding statements.

A strict standard for this is hard soundness, which assumes a binary-valued entailment model.

Definition 2.1 (Hard Soundness)**.** A reasoning chain $(C_1, \ldots, C_{n+m})$ is *hard-sound* with respect to a deterministic entailment model $\mathcal{E}$ if, for all derived claims $i = 1, \ldots, m$:

$$\mathcal{E}((C_1, \ldots, C_{n+i-1}), C_{n+i}) = 1. \tag{1}$$

Hard soundness is a useful theoretical notion, but it is too brittle for real-world LLM outputs where reasoning is often imperfect. A single minor error invalidates an entire chain, necessitating a more flexible, probabilistic approach to measuring soundness.

3 Soundness Checks via Autoregressive Reasoning Entailment Stability

We now introduce a practical method for certifying LLM-generated reasoning chains. We aim to compute a sequence of *entailment stability scores*, $\tau_1, \ldots, \tau_m \in [0, 1]$, where each τ_k quantifies how reliably the k-th derived claim, C_{n+k}, is entailed by its predecessors $(C_1, \ldots, C_{n+k-1})$. A low τ_k indicates a likely error in claim C_{n+k}.

3.1 Entailment with Probabilistic Premises

The key challenge is assessing entailment when preceding claims are themselves uncertain. Our approach is to average the entailment likelihood over all possible subsets of valid premises. This is motivated by human cognition, where dubious statements are often discounted [Johnson-Laird, 2010], and by LLM performance improvements when irrelevant context is filtered [Mukherjee et al., 2025].

To measure the stability of a hypothesis H given a premise P with k uncertain claims, we consider all 2^k subsets of P. Each subset, determined by a binary vector $\alpha \in \{0,1\}^k$, has a probability $\Pr[\alpha]$. The stability is the expected entailment over these subsets:

$$\tau(\mathcal{E}, P, H) = \sum_{\alpha \in \{0,1\}^k} \mathcal{E}(P(\alpha), H) \cdot \Pr[\alpha]. \tag{2}$$

3.2 Autoregressive Reasoning Entailment Stability with Efficient Sampling

We extend this concept to an entire reasoning chain to compute the sequence of stability scores $\tau_1, \ldots, \tau_m$. Our approach, **Autoregressive Reasoning Entailment Stability (ARES)**, autoregressively assesses each claim C_{n+k} while accounting for the soundness of all preceding claims.

To formalize this, we define the probability $\Pr[\alpha]$ for a specific combination of included claims (where $\alpha_i = 1$) recursively.

Base Case ($k = 1$): For the first derived claim, the premises are the base claims $C_1, \ldots, C_n$. We assume each base claim C_i has a given prior probability of soundness p_i. The probability of a specific combination $\alpha_{1:n}$ is:

$$\Pr[\alpha_{1:n}] = \prod_{i=1}^{n} p_i^{\alpha_i} (1 - p_i)^{1-\alpha_i} \tag{3}$$

Inductive Case ($k > 1$): For subsequent derived claims, the probability of a premise combination is updated via the chain rule, conditioned on the entailment of the new claim given the previous combination:

$$\Pr[\alpha_{1:n+k}] = \Pr[\alpha_{1:n+k-1}] \cdot \mathcal{E}(C(\alpha_{1:n+k-1}), C_{n+k}) \tag{4}$$

where $C(\alpha_{1:n+k-1})$ is the subset of claims indexed by α.

With this, we define the *entailment stability score* τ_k for the k-th derived claim as the marginalization over all 2^{n+k-1} predecessor combinations:

$$\tau_k = \sum_{\alpha \in \{0,1\}^{n+k-1}} \mathcal{E}(C(\alpha), C_{n+k}) \cdot \Pr[\alpha] \tag{5}$$

Directly computing τ_k is intractable. Instead, we estimate it efficiently using Monte Carlo sampling: $\hat{\tau}_k = \frac{1}{N} \sum_{i=1}^{N} \mathcal{E}(C(\alpha^{(i)}), C_{n+k})$, where each $\alpha^{(i)}$ is a sample drawn according to the recursive probability definition. This estimation converges rapidly and comes with statistical guarantees.

Theorem 3.1. *Let $N \geq \frac{\log(2m/\delta)}{2\varepsilon^2}$ for any $\varepsilon > 0$ and $\delta > 0$. Given an entailment model $\mathcal{E}$ and a reasoning chain with m derived claims, use N i.i.d. samples to estimate each τ_k. Then, with probability at least $1 - \delta$, we have $|\hat{\tau}_k - \tau_k| \leq \varepsilon$ for all k.*

Proof. See Appendix A. □

Error Detection. A derived claim C_{n+k} is marked as erroneous if its estimated stability score $\hat{\tau}_k$ falls below a prescribed threshold. This procedure effectively identifies errors in reasoning chains.

4 Evaluating ARES

Experiment Setup. We compared ARES against baselines including LLM-Judge, Entail-Prev, and methods from ROSCOE and ReCEval, using both proprietary (GPT-4o-mini) and open-source (Qwen family) models. We tested on established benchmarks (PRMBench, DeltaBench) and two new synthetic datasets (ClaimTrees, CaptainCookRecipes) designed to isolate error propagation.

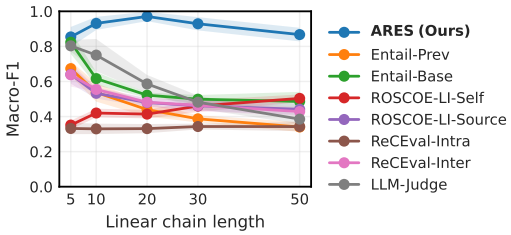

Figure 2: **(ClaimTrees) GPT-4o-mini.** ARES can robustly identify error propagations in long reasoning chains, whereas other methods fail.

Method	Step Avg	Final Step
ARES	**0.730**	**0.660**
Entail-Prev	**0.790**	0.240
Entail-Base	0.540	0.300
ROSCOE-LI-Self	0.540	0.210
ROSCOE-LI-Source	0.630	0.310
ReCEval-Intra	0.480	0.060
ReCEval-Inter	0.480	0.190
LLM-Judge	0.570	0.250

Figure 3: **(ClaimTrees) GPT-4o-mini.** ARES can robustly identify error propagations in long reasoning chains, whereas other methods fail.

Claim	**ARES (Ours)**	Entail -Prev	Entail -Base	ROSCOE -LI-Self	ROSCOE -LI-Source	ReCEval -Intra	ReCEval -Inter	LLM -Judge	*Ground Truth*
Context Rules: H3 -> AZ; SG -> C6; C6 -> GM; VD -> H3; G8 -> VD; D8 -> U8; U8 -> DG; DG -> G8. Fact: I have D8. ...									
Claim 5: I use rule (VD -> H3) to derive H3	**0.79**✓	**1.00**✓	0.00×	**1.00**✓	0.00×	**1.00**✓	0.00×	**1.00**✓	✓
Claim 6: I use rule (H3 -> AZ) to derive AZ	**0.82**✓	**1.00**✓	**1.00**✓	**1.00**✓	**1.00**✓	**1.00**✓	**1.00**✓	**1.00**✓	✓
Claim 7: I use rule (AZ -> SG) to derive SG	**0.00**×	**0.00**×	**0.00**×	1.00✓	**0.00**×	1.00✓	**0.00**×	**0.00**×	×
Claim 8: I use rule (SG -> C6) to derive C6	**0.00**×	1.00✓	**0.00**×	1.00✓	**0.00**×	1.00✓	**0.00**×	1.00✓	×

Table 1: **(ClaimTrees example)** After two correct claims 5-6, an initial error (**Claim 7**) using the non-existing rule AZ $\rightarrow$ SG causes a propagated error (**Claim 8**). Only ARES correctly judges all.

Performance was measured using Macro-F1 with 5-fold cross-validation (see Appendix C for full details and the main results in Table A3).

Excelling in Identifying Propagated Errors. The performance gap is most pronounced on our synthetic datasets designed to isolate error propagation. As shown in Figure 2, ARES maintains a high Macro-F1 score (over 89%) on chains up to 50 steps long, while baseline performance collapses. This superior performance in identifying propagated errors, illustrated with a concrete example in Table 1, confirms its unique robustness and is consistent across all datasets (Table A3).

Downstream Task Performance. In a best-of-n selection task on PRMBench, we select the better of two candidate chains (original vs. modified). Results in Figure 3 show that when using the final step's score—a stricter metric—ARES significantly outperforms all baselines. The performance of simpler approaches like Entail-Prev collapses on this metric, demonstrating ARES's robustness and reliability for downstream tasks.

5 Related Work

Reasoning Chain Verifiers. Primary approaches to verifying reasoning chains include LLM Judges [Tyagi et al., 2024, He et al., 2025] and Process Reward Models (PRMs) [Lightman et al., 2023]. While recent verifiers incorporate logic, they have limitations: ROSCOE [Golovneva et al., 2023] and ReCEval [Prasad et al., 2023] use pairwise contradiction, which is less effective with complex premises, and PARC [Mukherjee et al., 2025] provides only a binary soundness classification. Our work differs by introducing a probabilistic framework for a more nuanced assessment of each claim.

Evaluation and Guarantees. While many benchmarks exist for evaluating CoT error detectors [Tyagi et al., 2024, Jacovi et al., 2024, Song et al., 2025, He et al., 2025], they often lack a consistent definition of error. We establish a clear standard by adopting a unified definition of soundness that includes propagated errors [Lee and Hockenmaier, 2025, Mukherjee et al., 2025] and create synthetic datasets for robust evaluation. Our work also joins a trend of applying statistical guarantees to AI systems [Fayyad et al., 2024, Jin et al., 2025], but is novel in providing these guarantees over the entire multi-step reasoning process itself, in contrast to frameworks that calibrate individual components [Feng et al., 2025].

References

Chirag Agarwal, Sree Harsha Tanneru, and Himabindu Lakkaraju. Faithfulness vs. plausibility: On the (un) reliability of explanations from large language models. *arXiv preprint arXiv:2402.04614*, 2024.

Jiuhai Chen and Jonas Mueller. Quantifying uncertainty in answers from any language model and enhancing their trustworthiness. *arXiv preprint arXiv:2308.16175*, 2023.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*, 2023.

Jamil Fayyad, Shadi Alijani, and Homayoun Najjaran. Empirical validation of conformal prediction for trustworthy skin lesions classification, 2024. URL https://arxiv.org/abs/2312.07460.

Yu Feng, Ben Zhou, Weidong Lin, and Dan Roth. BIRD: A trustworthy bayesian inference framework for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=fAAaT826Vv.

Olga Golovneva, Moya Peng Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=xYlJRpzZtsY.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and Bo Zheng. Can large language models detect errors in long chain-of-thought reasoning?, 2025. URL https://arxiv.org/abs/2502.19361.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4615–4634, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.254. URL https://aclanthology.org/2024.acl-long.254/.

Helen Jin, Anton Xue, Weiqiu You, Surbhi Goel, and Eric Wong. Probabilistic stability guarantees for feature attributions. *arXiv preprint arXiv:2504.13787*, 2025.

Phil Johnson-Laird. Deductive reasoning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 1(1): 8–17, 2010.

Jinu Lee and Julia Hockenmaier. Evaluating step-by-step reasoning traces: A survey, 2025. URL https://arxiv.org/abs/2502.12289.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. Towards faithful model explanation in NLP: A survey. *Computational Linguistics*, 50(2):657–723, June 2024. doi: 10.1162/coli_a_00511. URL https://aclanthology.org/2024.cl-2.6/.

Sagnik Mukherjee, Abhinav Chinta, Takyoung Kim, Tarun Anoop Sharma, and Dilek Hakkani-Tür. Premise-augmented reasoning chains improve error identification in math reasoning with llms, 2025. URL https://arxiv.org/abs/2502.02362.

Rohith Peddi, Shivvrat Arya, Bharath Challa, Likhitha Pallapothula, Akshay Vyas, Bhavya Gouripeddi, Qifan Zhang, Jikai Wang, Vasundhara Komaragiri, Eric Ragan, Nicholas Ruozzi, Yu Xiang, and Vibhav Gogate. Captaincook4d: A dataset for understanding errors in procedural activities. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 135626–135679. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/f4a04396c2ed1342a5d8d05e94cb6101-Paper-Datasets_and_Benchmarks_Track.pdf.

Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. ReCEval: Evaluating reasoning chains via correctness and informativeness. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10066–10086, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.622. URL https://aclanthology.org/2023.emnlp-main.622/.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. Prmbench: A fine-grained and challenging benchmark for process-level reward models, 2025. URL https://arxiv.org/abs/2501.03124.

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. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

Nemika Tyagi, Mihir Parmar, Mohith Kulkarni, Aswin Rrv, Nisarg Patel, Mutsumi Nakamura, Arindam Mitra, and Chitta Baral. Step-by-step reasoning to solve grid puzzles: Where do LLMs falter? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19898–19915, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1111. URL https://aclanthology.org/2024.emnlp-main.1111/.

Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. Natural language reasoning, a survey. *ACM Computing Surveys*, 56(12):1–39, 2024.

Lotfi A Zadeh. Fuzzy logic. *Scholarpedia*, 3(3):1766, 2008.

A Proofs

Theorem 3.1. *Let $N \geq \frac{\log(2m/\delta)}{2\varepsilon^2}$ for any $\varepsilon > 0$ and $\delta > 0$. Given an entailment model $\mathcal{E}$ and a reasoning chain with m derived claims, use N i.i.d. samples to estimate each τ_k. Then, with probability at least $1-\delta$, we have $|\hat{\tau}_k - \tau_k| \leq \varepsilon$ for all k.*

Proof. Let $\mathcal{A}_i$ denote the event that $|\hat{\tau}_i - \tau_i| < \varepsilon$ for each $i \in \{n+1, \ldots, n+m\}$. We want to prove that

$$\Pr\left(\bigcap_{i=n+1}^{n+m} \mathcal{A}_i\right) = 1 - \Pr\left(\bigcup_{i=n+1}^{n+m} \bar{\mathcal{A}}_i\right) \geq 1 - \delta. \tag{6}$$

According to Boole's inequality and Hoeffding's inequality,

$$\begin{aligned}
\Pr\left(\bigcup_{i=n+1}^{n+m} \bar{\mathcal{A}}_i\right) &\leq \sum_{i=n+1}^{n+m} \Pr\left(\bar{\mathcal{A}}_i\right) && \text{(Boole's)} \\
&= \sum_{i=n+1}^{n+m} \Pr\left(|\hat{\tau}_i - \tau_i| \geq \varepsilon\right) && (7) \\
&\leq \sum_{i=n+1}^{n+m} 2\exp(-2N\varepsilon^2) && \text{(Hoeffding's)} \\
&= 2m\exp(-2N\varepsilon^2) && (8) \\
&\leq \delta \quad \text{when } N \geq \frac{\log(2m/\delta)}{2\varepsilon^2}, && (9)
\end{aligned}$$

Method	Robust	Causal	Sufficient
ARES (ours)	✓	✓	✓
Entail-Prev	✗	✓	✓
Entail-Base	✓	✓	✗
ROSCOE-LI-Self	✗	✓	✗
ROSCOE-LI-Source	✗	✓	✗
ReCEval-Intra	✓	✓	✗
ReCEval-Inter	✗	✓	✗
LLM-Judge	✗	✗	✓

Table A2: (Desiderata for methods) **Robust:** Previous errors do not adversely affect current step. **Causal:** Downstream steps do not affect current step. **Sufficient:** All relevant claims included as premise for detection.

Algorithm 1 Estimating ARES

Require: Reasoning chain $(C_1, \ldots, C_{n+m})$, tolerance (ε, δ), base priors $p_1, \ldots, p_n$, and entailment model $\mathcal{E}$.

$N \leftarrow \frac{\log(2m/\delta)}{2\varepsilon^2}$
for $i = 1, \ldots, N$ **do**
  $\alpha_1^{(i)} \sim \text{Bernoulli}(p_1), \ldots, \alpha_n^{(i)} \sim \text{Bernoulli}(p_n)$
  for $k = 1, \ldots, m$ **do**
    $p_{n+k}^{(i)} \leftarrow \mathcal{E}(C(\alpha_{1:n+k-1}^{(i)}), C_{n+k})$
    $\alpha_{n+k}^{(i)} \sim \text{Bernoulli}(p_{n+k}^{(i)})$
  end for
end for
for $k = 1, \ldots, m$ **do**
  $\hat{\tau}_k = \frac{1}{N} \sum_{i=1}^{N} p_{n+k}^{(i)}$
end for

with the estimation error of each stability rate bounded by $\delta_i = \frac{\delta}{m}$. □

B Method

There are three important desiderata for error detection methods:

1. **Robust:** Previous errors do not adversely affect current step.
2. **Causal:** Downstream steps do not affect current step.
3. **Sufficient:** All relevant claims included as premise for detection.

Appendix B shows that only ARES satisfies all desiderata while none of the baseline methods does.

Algorithm details is shown in Algorithm 1.

C Experiments

C.1 Entailment Model

We instantiate the entailment model by prompting LLMs to judge the entailment of a hypothesis given a premise, where there can be multiple claims in the premise. The LLM's output is either YES/NO in the binary case, or a 7-point Likert scale converted to a real value between 0 and 1.

C.2 Hyperparameters for ARES

In our experiments, we used $\delta = 0.1$ and $\varepsilon = 0.1$ for ARES, which determines the number of samples to take. We use $p = 0.95$ for the inclusion rate for base claims to allow buffer for information overload.

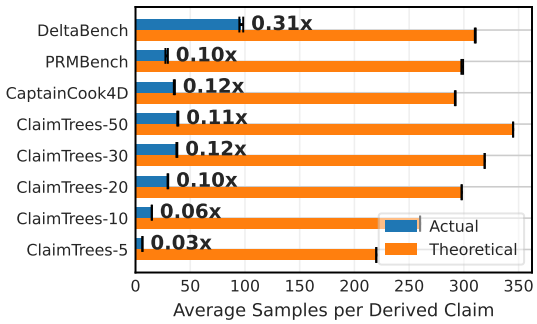

Figure A4: **(Per-Claim Samples)** ARES in practice only uses 0.03x to 0.31x of theoretical number of samples on average for each derived claim.

Long Chain Example.

Base Claims:
Rule: AZ -> DG (meaning that if I have AZ, I can derive DG)
Rule: SG -> H3 (meaning that if I have SG, I can derive H3)
I have AZ
Rule: DG -> SG (meaning that if I have DG, I can derive SG)
Reasoning Steps:
I have AZ, I use rule (AZ -> DG) to derive DG, now I have DG
I have DG, I use rule (DG -> SG) to derive SG, now I have SG
I have SG, I use rule (SG -> H3) to derive H3, now I have H3
I have H3, I use rule (H3 -> VD) to derive VD, now I have VD

Figure A5: Long Chain Example for ClaimTrees

C.3 Experiment Details

We use a subset of examples for each experiment. Experiment results are computed using 5-fold cross-validation. For each split, the thresholds are picked for the best Macro-F1 on the validation split, and the final numbers are on the test split, averaged over the 5 folds.

C.4 Controllable Datasets

ClaimTrees. One is ClaimTrees, a synthetic dataset in which the reasoning chain reasons starts from a state A, and reason all the way to another state, say T. All the reasoning rules are provided in the premise, except one, so that from that point on we know that all the claims are unsound: An example of a chain of reasoning is shown in Figure A5. In this example, rule B -> C does not actually exist, and thus the reasoning steps starting from the second derived step are unsound claims. We can construct reasoning chains with arbitrary length and errors occurring at different places.

CaptainCookRecipes. CaptainCookRecipes is derived from the recipe graphs in CaptainCook4D [Peddi et al., 2024], where certain actions must follow other actions. We then construct base claims using edges in the graph as rules, similar to how we construct the ones in ClaimTrees. In addition, we add ingredients to the base claims and randomly drop an ingredient. Then, all the claims that require the ingredient and claims that follow them become unsound. We extract the ingredients from the claims using GPT-4o-mini.

An example of results for CaptainCookRecipes is shown in Table A5. With propagated errors present, only ARES is able to capture all errors.

C.5 Computing Resources

We used an NVIDIA A100 GPU with 80GB of memory for the Qwen3-4B model. For GPT-4o-mini, we used approximately 600 USD in total for prototyping and experiments.

C.6 Efficiency

Table A4 shows the computational efficiency of ARES, demonstrating that performance remains stable even with 15x fewer samples.

ARES's computational efficiency stems from a two-tiered optimization. First, ARES uses a sampling-based strategy for soundness checking, which is inherently more efficient than an exhaustive approach. Second, we add another layer of efficiency by eliminating redundant LLM calls for the same premise-hypothesis pairs. This dual approach dramatically reduces computational overhead, as shown by the gap between theoretical and actual samples in Figure A4. The result is a highly efficient process: on shorter chains (ClaimTrees-5), we require only 0.03x the theoretical samples. Even on DeltaBench, which needs more sampling due to model uncertainty, the method remains effective at 0.31x the theoretical maximum.

C.7 Probabilistic Entailment Model Output

To obtain probabilistic entailment model output, we instruct LLM to output one of the following: Very Likely, Likely, Somewhat Likely, Neutral, Somewhat Unlikely, Unlikely, Very Unlikely and convert them to 1, 0.8, 0.6, 0.5, 0.4, 0.2, 0.0, respectively.

C.8 Best-of-N Results

For best-of-N result with standard deviations, see Table A6.

C.9 ARES Also Improves PRMs

Process Reward Models (PRMs) can sometimes rival LLMs, and can also provide a non-binary soundness score. We run additional experiments using a SOTA PRMs, Qwen2.5-Math-PRM-7B, as the base entailment model. The results show that ARES can help significantly improve upon PRM on reasoning chains with propagated errors.

The results in Table A7 show that, while the specialized PRM is a strong baseline on its in-domain dataset (PRMBench), applying ARES significantly improves performance on the abstract ClaimTrees dataset which has many propagated errors. On out-of-domain (non-math) CaptainCook4D, ARES achieves on par performance with PRM. This demonstrates ARES's value as a flexible, general-purpose framework that adds robustness, especially on tasks with propagated errors.

C.10 Discussion of Errors

Our inspection of the data and error detection outputs reveals some insights. Entail-Base fails on PRMBench because judging entailment in long math derivations is challenging. Both LLM-Judge and Entail-Base fail in DeltaBench, with Entail-Base struggling to judge entailment in very long reasoning chains. In naturally occurring datasets, error propagation is limited and not always annotated, so Entail-Prev performs close to ARES. However, synthetic data shows Entail-Prev fails with propagated errors. LLM-Judge sometimes fails to follow instructions, outputting incorrect numbers of scores relative to claims being judged. Pairwise methods in ROSCOE and ReCEval cannot detect complex errors that need multiple claims as premise. ARES can only improve upon entailment models that can already do correct entailment.

C.11 Complete Results on All Four Datasets

For complete results on all four datasets, see Table A3.

Dataset / Method	GPT-4o-mini			Qwen3-4B		
	Recall	Precision	F1	Recall	Precision	F1
PRMBench						
ARES	**0.680 ± 0.024**	**0.627 ± 0.021**	**0.640 ± 0.023**	**0.688 ± 0.020**	0.623 ± 0.011	0.636 ± 0.011
Entail-Prev	0.639 ± 0.032	0.602 ± 0.016	0.596 ± 0.024	**0.698 ± 0.016**	0.626 ± 0.015	0.641 ± 0.017
Entail-Base	0.524 ± 0.022	0.511 ± 0.011	0.484 ± 0.016	0.631 ± 0.016	0.558 ± 0.007	0.530 ± 0.011
ROSCOE-LI-Self	**0.672 ± 0.012**	0.575 ± 0.007	0.489 ± 0.022	0.458 ± 0.011	0.478 ± 0.006	0.446 ± 0.006
ROSCOE-LI-Source	**0.676 ± 0.014**	0.584 ± 0.008	0.570 ± 0.011	0.497 ± 0.003	0.496 ± 0.004	0.495 ± 0.004
ReCEval-Intra	0.563 ± 0.012	0.581 ± 0.014	0.568 ± 0.013	0.550 ± 0.007	0.573 ± 0.013	0.554 ± 0.007
ReCEval-Inter	0.664 ± 0.012	0.573 ± 0.007	0.465 ± 0.022	0.449 ± 0.004	0.476 ± 0.003	0.433 ± 0.004
LLM-Judge	0.647 ± 0.011	**0.645 ± 0.019**	**0.643 ± 0.013**	**0.695 ± 0.017**	**0.662 ± 0.016**	**0.675 ± 0.016**
DeltaBench						
ARES	**0.702 ± 0.024**	**0.728 ± 0.022**	**0.708 ± 0.026**	0.513 ± 0.013	0.512 ± 0.013	0.498 ± 0.010
Entail-Prev	**0.698 ± 0.032**	**0.709 ± 0.029**	**0.699 ± 0.031**	0.523 ± 0.011	0.522 ± 0.010	0.506 ± 0.009
Entail-Base	0.614 ± 0.010	0.596 ± 0.004	0.594 ± 0.005	**0.580 ± 0.008**	0.586 ± 0.008	**0.579 ± 0.009**
ROSCOE-LI-Self	0.579 ± 0.006	0.664 ± 0.027	0.571 ± 0.013	0.555 ± 0.007	**0.638 ± 0.039**	0.522 ± 0.003
ROSCOE-LI-Source	0.471 ± 0.006	0.456 ± 0.009	0.453 ± 0.005	0.484 ± 0.013	0.472 ± 0.021	0.457 ± 0.017
ReCEval-Intra	0.500 ± 0.000	0.357 ± 0.012	0.416 ± 0.009	0.530 ± 0.006	0.529 ± 0.005	0.528 ± 0.005
ReCEval-Inter	0.503 ± 0.007	0.508 ± 0.012	0.483 ± 0.010	0.507 ± 0.006	0.508 ± 0.006	0.505 ± 0.007
LLM-Judge	0.498 ± 0.002	0.371 ± 0.026	0.381 ± 0.027	0.548 ± 0.010	0.563 ± 0.016	0.494 ± 0.009
ClaimTrees						
ARES	**0.914 ± 0.012**	**0.921 ± 0.013**	**0.903 ± 0.020**	**0.731 ± 0.006**	0.755 ± 0.009	**0.723 ± 0.006**
Entail-Prev	0.587 ± 0.012	0.704 ± 0.025	0.491 ± 0.020	0.580 ± 0.013	0.760 ± 0.006	0.480 ± 0.022
Entail-Base	0.645 ± 0.018	0.647 ± 0.019	0.619 ± 0.021	0.586 ± 0.019	0.630 ± 0.018	0.521 ± 0.026
ROSCOE-LI-Self	0.528 ± 0.005	0.569 ± 0.016	0.430 ± 0.011	0.568 ± 0.009	0.732 ± 0.005	0.473 ± 0.017
ROSCOE-LI-Source	0.540 ± 0.012	0.543 ± 0.013	0.511 ± 0.016	0.491 ± 0.004	0.484 ± 0.006	0.448 ± 0.008
ReCEval-Intra	0.500 ± 0.000	0.254 ± 0.006	0.336 ± 0.005	0.500 ± 0.000	0.252 ± 0.003	0.335 ± 0.003
ReCEval-Inter	0.546 ± 0.013	0.548 ± 0.013	0.513 ± 0.016	0.495 ± 0.003	0.489 ± 0.005	0.451 ± 0.007
LLM-Judge	0.687 ± 0.018	0.780 ± 0.016	0.628 ± 0.027	0.602 ± 0.026	**0.769 ± 0.013**	0.502 ± 0.034
CaptainCookRecipes						
ARES	**0.636 ± 0.010**	0.657 ± 0.011	**0.633 ± 0.010**	0.532 ± 0.012	0.532 ± 0.012	0.517 ± 0.009
Entail-Prev	0.468 ± 0.004	0.462 ± 0.004	0.428 ± 0.010	0.511 ± 0.005	0.529 ± 0.014	0.384 ± 0.008
Entail-Base	0.591 ± 0.007	0.598 ± 0.008	0.589 ± 0.007	0.500 ± 0.000	0.290 ± 0.005	0.367 ± 0.005
ROSCOE-LI-Self	0.555 ± 0.005	**0.703 ± 0.018**	0.483 ± 0.011	**0.619 ± 0.007**	**0.711 ± 0.012**	**0.601 ± 0.010**
ROSCOE-LI-Source	0.500 ± 0.000	0.283 ± 0.009	0.361 ± 0.007	0.500 ± 0.000	0.290 ± 0.006	0.367 ± 0.004
ReCEval-Intra	0.515 ± 0.008	0.540 ± 0.022	0.396 ± 0.010	0.500 ± 0.000	0.290 ± 0.006	0.367 ± 0.004
ReCEval-Inter	0.500 ± 0.000	0.283 ± 0.009	0.361 ± 0.007	0.500 ± 0.000	0.290 ± 0.005	0.367 ± 0.004
LLM-Judge	0.560 ± 0.023	0.569 ± 0.024	0.530 ± 0.028	0.500 ± 0.000	0.289 ± 0.005	0.366 ± 0.004

Table A3: **(Benchmark Results)** ARES is top-performing in majority of settings (5/8), with no other single method being a consistent challenger. For each dataset+model group, **Bold** is the best and underline is the second best.

C.12 Ablations

Tables A9, A10, and A8 present detailed ablation studies on ClaimTrees, confirming the robustness of ARES against irrelevant claims and benign errors, and analyzing the impact of the base claim inclusion probability p.

C.13 Additional Experimental Results

This appendix provides further experimental details. Table A4 illustrates the computational efficiency of ARES, demonstrating stable performance even when using up to 15x fewer samples. Tables A9, A10, and A8 present detailed ablation studies on the ClaimTrees dataset. These results confirm ARES's robustness against irrelevant claims and benign errors, and analyze the impact of the base claim inclusion probability, p.

Method	**PRMBench**	**DeltaBench**	**ClaimTrees-10**	**CaptainCookRecipes**
ARES-ε0.1	**0.640**	**0.708**	**0.931**	<u>0.633</u>
ARES-ε0.2	<u>0.599</u>	<u>0.697</u>	<u>0.926</u>	0.631
ARES-ε0.3	0.582	0.694	0.919	0.621
ARES-ε0.4	0.595	0.687	0.922	**0.640**

Table A4: **(GPT-4o-mini) Performance Convergence with Samples** ARES is able to achieve high accuracy even when using a smaller number of samples. When ε =0.1, 0.2, 0.3, 0.4, a sequence of length $m = 10$ needs 265, 67, 30, 17 samples per step respectively. We can see that there is no significant performance change when we increase the ϵ to 0.4 and thus decrease the number of samples 15x.

Claim	**ARES (Ours)**	Entail -Prev	Entail -Base	ReCEval -Inter	ReCEval -Intra	ROSCOE -LI-Source	ROSCOE -LI-Self	LLM -Judge	*Ground Truth*
sent1: Only after the necessary preceding steps (put-put tomatoes on a serving plate), And if we have all the ingredients, we can then Pour-Pour the egg mixture into the pan.	–	–	–	–	–	–	–	–	–
sent2: Only after the necessary preceding steps (Take-Take a tomato), And if we have all the ingredients, we can then Cut-Cut tomato into two pieces.	–	–	–	–	–	–	–	–	–
sent3: Only after the necessary preceding steps (Stop-Stop stirring when it's nearly cooked to allow it to set into an omelette), And if we have all the ingredients, we can then Transfer-Transfer omelette to the plate and serve with the tomatoes.	–	–	–	–	–	–	–	–	–
sent4: Only after the necessary preceding steps (Chop-Chop 2 tbsp cilantro), And if we have all the ingredients, we can then add-add the chopped cilantro to the bowl.	–	–	–	–	–	–	–	–	–
sent5: Only after the necessary preceding steps (START), And if we have all the ingredients, we can then add-1/2 tsp ground black pepper to the bowl.	–	–	–	–	–	–	–	–	–
sent6: We have ground black pepper.	–	–	–	–	–	–	–	–	–
sent7: We have oil.	–	–	–	–	–	–	–	–	–
sent8: Only after the necessary preceding steps (Scoop-Scoop the tomatoes from the pan), And if we have all the ingredients, we can then put-put tomatoes on a serving plate.	–	–	–	–	–	–	–	–	–
sent9: Only after the necessary preceding steps (Pour-Pour the egg mixture into the pan), And if we have all the ingredients, we can then stir-stir gently with a wooden spoon so the egg that sets on the base of the pan moves to enable the uncooked egg to flow into the space.	–	–	–	–	–	–	–	–	–
sent10: Only after the necessary preceding steps (Transfer-Transfer omelette to the plate and serve with the tomatoes), And if we have all the ingredients, we can then END.	–	–	–	–	–	–	–	–	–
sent11: Only after the necessary preceding steps (add-add the chopped cilantro to the bowl, and crack-crack one egg in a bowl, and add-1/2 tsp ground black pepper to the bowl), And if we have all the ingredients, we can then Beat-Beat the contents of the bowl.	–	–	–	–	–	–	–	–	–
sent12: Only after the necessary preceding steps (Heat-Heat 1 tbsp oil in a non-stick frying pan), And if we have all the ingredients, we can then cook-cook the tomatoes cut-side down until they start to soften and colour.	–	–	–	–	–	–	–	–	–
sent13: Only after the necessary preceding steps (START), And if we have all the ingredients, we can then crack-crack one egg in a bowl.	–	–	–	–	–	–	–	–	–
sent14: Only after the necessary preceding steps (cook-cook the tomatoes cut-side down until they start to soften and colour), And if we have all the ingredients, we can then Scoop-Scoop the tomatoes from the pan.	–	–	–	–	–	–	–	–	–
sent15: Only after the necessary preceding steps (START), And if we have all the ingredients, we can then Take-Take a tomato.	–	–	–	–	–	–	–	–	–
sent16: Only after the necessary preceding steps (Beat-Beat the contents of the bowl, and Cut-Cut tomato into two pieces), And if we have all the ingredients, we can then Heat-Heat 1 tbsp oil in a non-stick frying pan.	–	–	–	–	–	–	–	–	–
sent17: We have egg.	–	–	–	–	–	–	–	–	–
sent18: Only after the necessary preceding steps (START), And if we have all the ingredients, we can then Chop-Chop 2 tbsp cilantro.	–	–	–	–	–	–	–	–	–
sent19: Only after the necessary preceding steps (stir-stir gently with a wooden spoon so the egg that sets on the base of the pan moves to enable the uncooked egg to flow into the space), And if we have all the ingredients, we can then Stop-Stop stirring when it's nearly cooked to allow it to set into an omelette.	–	–	–	–	–	–	–	–	–
sent20: We have tomato.	–	–	–	–	–	–	–	–	–
sent21: We now START.	–	–	–	–	–	–	–	–	–
int1: Because we have completed all previous steps (START), and have all necessary ingredients (cilantro), we can now do the step Chop-Chop 2 tbsp cilantro. And now we have completed this step Chop-Chop 2 tbsp cilantro.	**0.35**×	**0.00**×	**0.00**×	**0.00**×	1.00✓	**0.00**×	1.00✓	1.00✓	×
int2: Because we have completed all previous steps (START), and have all necessary ingredients (egg), we can now do the step crack-crack one egg in a bowl. And now we have completed this step crack-crack one egg in a bowl.	**0.85**✓	**1.00**✓	**1.00**✓	0.00×	**1.00**✓	0.00×	0.00×	**1.00**✓	✓
int3: Because we have completed all previous steps (START), and have all necessary ingredients (tomato), we can now do the step Take-Take a tomato. And now we have completed this step Take-Take a tomato.	**0.98**✓	**1.00**✓	**1.00**✓	0.00×	**1.00**✓	0.00×	0.00×	**1.00**✓	✓
int4: Because we have completed all previous steps (START), and have all necessary ingredients (ground black pepper), we can now do the step add-1/2 tsp ground black pepper to the bowl. And now we have completed this step add-1/2 tsp ground black pepper to the bowl.	**0.80**✓	**1.00**✓	**1.00**✓	0.00×	**1.00**✓	0.00×	**1.00**✓	**1.00**✓	✓
int5: Because we have completed all previous steps (Chop-Chop 2 tbsp cilantro), and have all necessary ingredients (cilantro), we can now do the step add-add the chopped cilantro to the bowl. And now we have completed this step add-add the chopped cilantro to the bowl.	**0.00**×	**0.00**×	**0.00**×	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	×
int6: Because we have completed all previous steps (Take-Take a tomato), and have all necessary ingredients (tomato), we can now do the step Cut-Cut tomato into two pieces. And now we have completed this step Cut-Cut tomato into two pieces.	**0.96**✓	**1.00**✓	**1.00**✓	0.00×	**1.00**✓	0.00×	0.00×	**1.00**✓	✓
int7: Because we have completed all previous steps (add-add the chopped cilantro to the bowl, and crack-crack one egg in a bowl, and add-1/2 tsp ground black pepper to the bowl), we can now do the step Beat-Beat the contents of the bowl. And now we have completed this step Beat-Beat the contents of the bowl.	**0.01**×	**0.00**×	1.00✓	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	×
int8: Because we have completed all previous steps (Beat-Beat the contents of the bowl, and Cut-Cut tomato into two pieces), and have all necessary ingredients (oil), we can now do the step Heat-Heat 1 tbsp oil in a non-stick frying pan. And now we have completed this step Heat-Heat 1 tbsp oil in a non-stick frying pan.	**0.00**×	**0.00**×	**0.00**×	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	×
int9: Because we have completed all previous steps (Heat-Heat 1 tbsp oil in a non-stick frying pan), and have all necessary ingredients (tomatoes), we can now do the step cook-cook the tomatoes cut-side down until they start to soften and colour. And now we have completed this step cook-cook the tomatoes cut-side down until they start to soften and colour.	**0.01**×	1.00✓	1.00✓	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	×
int10: Because we have completed all previous steps (cook-cook the tomatoes cut-side down until they start to soften and colour), we can now do the step Scoop-Scoop the tomatoes from the pan. And now we have completed this step Scoop-Scoop the tomatoes from the pan.	**0.21**×	1.00✓	1.00✓	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	×
int11: Because we have completed all previous steps (Scoop-Scoop the tomatoes from the pan), we can now do the step put-put tomatoes on a serving plate. And now we have completed this step put-put tomatoes on a serving plate.	**0.18**×	1.00✓	1.00✓	**0.00**×	**0.00**×	**0.00**×	**0.00**×	1.00✓	×
int12: Because we have completed all previous steps (put-put tomatoes on a serving plate), we can now do the step Pour-Pour the egg mixture into the pan. And now we have completed this step Pour-Pour the egg mixture into the pan.	**0.18**×	1.00✓	**0.00**×	**0.00**×	**0.00**×	**0.00**×	**0.00**×	1.00✓	×
int13: Because we have completed all previous steps (Pour-Pour the egg mixture into the pan), we can now do the step stir-stir gently with a wooden spoon so the egg that sets on the base of the pan moves to enable the uncooked egg to flow into the space. And now we have completed this step stir-stir gently with a wooden spoon so the egg that sets on the base of the pan moves to enable the uncooked egg to flow into the space.	**0.19**×	1.00✓	**0.00**×	**0.00**×	**0.00**×	**0.00**×	**0.00**×	1.00✓	×
int14: Because we have completed all previous steps (stir-stir gently with a wooden spoon so the egg that sets on the base of the pan moves to enable the uncooked egg to flow into the space), we can now do the step Stop-Stop stirring when it's nearly cooked to allow it to set into an omelette. And now we have completed this step Stop-Stop stirring when it's nearly cooked to allow it to set into an omelette.	**0.19**×	1.00✓	**0.00**×	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	×
int15: Because we have completed all previous steps (Stop-Stop stirring when it's nearly cooked to allow it to set into an omelette), we can now do the step Transfer-Transfer omelette to the plate and serve with the tomatoes. And now we have completed this step Transfer-Transfer omelette to the plate and serve with the tomatoes.	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	×
int16: Because we have completed all previous steps (Transfer-Transfer omelette to the plate and serve with the tomatoes), we can now do the step END. And now we have completed this step END.	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	**0.00**×	**0.00**×	1.00✓	×

Table A5: **(CaptainCookRecipes Example)** Only ARES is able to correctly judge all steps for soundness. Checks ✓ indicate that a method classifies the step as sound after thresholding, and crosses × indicate that the method judges that step to be erroneous. **Bold**: Correctly judged soundness.

Method	Using Step Average (acc±std)	Using Final Step (acc±std)
ARES	**0.730±0.045**	**0.660±0.049**
Entail-Prev	**0.790±0.043**	0.240±0.042
Entail-Base	0.540±0.049	0.300±0.046
ROSCOE-LI-Self	0.540±0.051	0.210±0.041
ROSCOE-LI-Source	0.630±0.049	<u>0.310±0.043</u>
ReCEval-Intra	0.480±0.050	0.060±0.024
ReCEval-Inter	0.480±0.048	0.190±0.038
LLM-Judge	0.570±0.050	0.250±0.044

Table A6: **(PRMBench Best-of-N)** ARES is a strong and robust predictor of downstream task performance. **Bold** is the best and <u>underline</u> is the second best.

Dataset / Method	Qwen2.5-Math-PRM-7B		
	Recall	Precision	F1
PRMBench			
ARES	**0.751 ± 0.017**	**0.733 ± 0.020**	**0.736 ± 0.014**
Entail-Prev	**0.751 ± 0.016**	**0.733 ± 0.020**	**0.736 ± 0.013**
Entail-Base	0.643 ± 0.022	0.632 ± 0.024	0.624 ± 0.018
ROSCOE-LI-Self	0.651 ± 0.013	0.598 ± 0.013	0.592 ± 0.006
ROSCOE-LI-Source	0.670 ± 0.020	0.621 ± 0.019	0.623 ± 0.013
ReCEval-Inter	0.644 ± 0.014	0.597 ± 0.013	0.596 ± 0.009
PRM	**0.763 ± 0.020**	**0.743 ± 0.017**	**0.749 ± 0.016**
ClaimTrees-10			
ARES	**0.739 ± 0.013**	**0.743 ± 0.012**	**0.733 ± 0.010**
Entail-Prev	<u>0.722 ± 0.016</u>	<u>0.725 ± 0.017</u>	<u>0.715 ± 0.011</u>
Entail-Base	0.611 ± 0.013	0.616 ± 0.013	0.597 ± 0.017
ROSCOE-LI-Self	0.655 ± 0.005	0.662 ± 0.005	0.644 ± 0.008
ROSCOE-LI-Source	0.604 ± 0.020	0.612 ± 0.020	0.591 ± 0.024
ReCEval-Inter	0.629 ± 0.020	0.628 ± 0.019	0.624 ± 0.020
PRM	0.607 ± 0.012	0.622 ± 0.013	0.594 ± 0.017
CaptainCook4D			
ARES	**0.551 ± 0.012**	**0.556 ± 0.014**	**0.543 ± 0.012**
Entail-Prev	**0.553 ± 0.011**	**0.560 ± 0.014**	**0.546 ± 0.010**
Entail-Base	0.531 ± 0.016	0.533 ± 0.017	0.519 ± 0.014
ROSCOE-LI-Self	<u>0.546 ± 0.008</u>	**0.563 ± 0.016**	0.529 ± 0.008
ROSCOE-LI-Source	0.469 ± 0.015	0.464 ± 0.018	0.457 ± 0.017
ReCEval-Inter	0.469 ± 0.015	0.465 ± 0.018	0.461 ± 0.017
PRM	**0.560 ± 0.013**	**0.569 ± 0.017**	**0.552 ± 0.013**

Table A7: **(Benchmark Results on Qwen2.5-Math-PRM-7B)** ARES performs the best across various datasets and backbone entailment models. For each dataset+model group, **Bold** is the best and <u>underline</u> is the second best.

Dataset / Method	Recall	Precision	F1
ClaimTrees-5			
ARES-1	0.881	0.900	0.873
ARES-0.95	0.861	0.889	0.854
ARES-bin-1	0.898	0.913	0.891
ARES-bin-0.95	**0.909**	**0.919**	**0.902**
Entail-Prev	0.704	0.813	0.673
Entail-Base	0.830	0.832	0.824
ROSCOE-LI-Self	0.499	0.500	0.351
ROSCOE-LI-Source	0.647	0.650	0.640
ReCEval-Intra	0.500	0.250	0.332
ReCEval-Inter	0.645	0.648	0.638
LLM-Judge	0.811	0.864	0.803
ClaimTrees-10			
ARES-1	0.937	0.943	0.936
ARES-0.95	0.931	0.936	0.931
ARES-bin-1	**0.960**	**0.965**	**0.962**
ARES-bin-0.95	0.947	0.951	0.948
Entail-Prev	0.608	0.783	0.538
Entail-Base	0.626	0.636	0.616
ROSCOE-LI-Self	0.524	0.589	0.420
ROSCOE-LI-Source	0.544	0.548	0.533
ReCEval-Intra	0.500	0.247	0.330
ReCEval-Inter	0.566	0.573	0.555
LLM-Judge	0.767	0.839	0.750
ClaimTrees-20			
ARES-1	**0.979**	**0.979**	**0.978**
ARES-0.95	0.971	0.971	0.971
ARES-bin-1	0.964	0.966	0.963
ARES-bin-0.95	0.968	0.970	0.968
Entail-Prev	0.551	0.760	0.440
Entail-Base	0.533	0.537	0.522
ROSCOE-LI-Self	0.521	0.580	0.414
ROSCOE-LI-Source	0.508	0.509	0.480
ReCEval-Intra	0.500	0.248	0.331
ReCEval-Inter	0.513	0.516	0.482
LLM-Judge	0.640	0.788	0.586
ClaimTrees-30			
ARES-1	**0.973**	0.972	**0.971**
ARES-0.95	0.931	0.934	0.929
ARES-bin-1	0.967	**0.973**	0.969
ARES-bin-0.95	0.957	0.960	0.956
Entail-Prev	0.530	0.731	0.387
Entail-Base	0.531	0.539	0.499
ROSCOE-LI-Self	0.543	0.595	0.460
ROSCOE-LI-Source	0.498	0.498	0.461
ReCEval-Intra	0.500	0.262	0.343
ReCEval-Inter	0.506	0.509	0.464
LLM-Judge	0.581	0.757	0.482
ClaimTrees-50			
ARES-1	**0.895**	0.899	**0.890**
ARES-0.95	0.871	0.871	0.867
ARES-bin-1	0.887	**0.904**	0.886
ARES-bin-0.95	0.892	0.892	0.888
Entail-Prev	0.512	0.601	0.340
Entail-Base	0.507	0.508	0.486
ROSCOE-LI-Self	0.555	0.581	0.504
ROSCOE-LI-Source	0.505	0.509	0.442
ReCEval-Intra	0.500	0.262	0.343
ReCEval-Inter	0.498	0.496	0.428
LLM-Judge	0.529	0.714	0.385

Table A8: **GPT-4o-mini (ClaimTrees)** ARES consistently identifies errors in long reasoning chains while other methods gradually fail.

Dataset / Method	Recall	Precision	F1
ClaimTrees-s3d3			
ARES-1	**0.921± 0.102**	**0.980± 0.018**	**0.941± 0.074**
ARES-0.95	0.904± 0.110	0.975± 0.027	0.927± 0.081
Entail-Prev	0.821± 0.046	0.951± 0.032	0.863± 0.039
Entail-Base	0.859± 0.122	0.866± 0.142	0.837± 0.134
ROSCOE-LI-Self	0.500± 0.000	0.115± 0.060	0.181± 0.078
ROSCOE-LI-Source	0.623± 0.101	0.593± 0.087	0.497± 0.161
ReCEval-Intra	0.500± 0.000	0.115± 0.060	0.181± 0.078
ReCEval-Inter	0.585± 0.081	0.562± 0.061	0.449± 0.115
LLM-Judge	0.833± 0.051	0.957± 0.022	0.875± 0.035
ClaimTrees-s3d5			
ARES-0.95	**0.867± 0.171**	**0.971± 0.037**	**0.887± 0.146**
Entail-Prev	0.718± 0.090	0.936± 0.045	0.761± 0.097
Entail-Base	0.659± 0.061	0.618± 0.076	0.610± 0.091
ROSCOE-LI-Self	0.497± 0.044	0.500± 0.242	0.460± 0.074
ROSCOE-LI-Source	0.513± 0.117	0.514± 0.077	0.340± 0.081
ReCEval-Intra	0.500± 0.000	0.100± 0.054	0.161± 0.074
ReCEval-Inter	0.550± 0.070	0.539± 0.050	0.356± 0.083
LLM-Judge	0.774± 0.178	0.942± 0.057	0.796± 0.169
ClaimTrees-s5d3			
ARES-1	**0.875± 0.217**	0.889± 0.232	**0.880± 0.223**
ARES-0.95	0.867± 0.217	**0.889± 0.232**	0.875± 0.222
Entail-Prev	0.767± 0.181	0.873± 0.223	0.799± 0.191
Entail-Base	0.824± 0.205	0.700± 0.149	0.729± 0.167
ROSCOE-LI-Self	0.500± 0.000	0.055± 0.033	0.097± 0.052
ROSCOE-LI-Source	0.650± 0.054	0.560± 0.031	0.380± 0.073
ReCEval-Intra	0.500± 0.000	0.055± 0.033	0.097± 0.052
ReCEval-Inter	0.594± 0.095	0.539± 0.043	0.357± 0.063
LLM-Judge	0.742± 0.192	0.868± 0.222	0.770± 0.201
ClaimTrees-s5d5			
ARES-1	**0.900± 0.163**	**0.990± 0.017**	**0.920± 0.139**
ARES-0.95	**0.900± 0.163**	**0.990± 0.017**	**0.920± 0.139**
Entail-Prev	0.723± 0.096	0.969± 0.018	0.783± 0.095
Entail-Base	0.692± 0.141	0.597± 0.067	0.610± 0.083
ROSCOE-LI-Self	0.481± 0.020	0.446± 0.018	0.462± 0.010
ROSCOE-LI-Source	0.578± 0.063	0.533± 0.027	0.321± 0.055
ReCEval-Intra	0.500± 0.000	0.053± 0.019	0.094± 0.031
ReCEval-Inter	0.584± 0.097	0.534± 0.059	0.310± 0.084
LLM-Judge	0.847± 0.140	0.951± 0.082	0.881± 0.111

Table A9: **GPT-4o-mini (ClaimTrees)** ARES differs from other methods in deeper trees instead of wider trees. s3d5 means trees with 3 sources and depth of 5.

Dataset / Method	Recall	Precision	F1
ClaimTrees-v5i1			
ARES-1	0.985± 0.014	0.950± 0.046	0.965± 0.032
ARES-0.95	0.990± 0.022	0.998± 0.005	0.994± 0.015
Entail-Prev	0.992± 0.011	0.974± 0.038	0.982± 0.026
Entail-Base	0.900± 0.027	0.788± 0.030	0.813± 0.038
ROSCOE-LI-Self	0.975± 0.009	0.918± 0.025	0.942± 0.019
ROSCOE-LI-Source	0.690± 0.062	0.626± 0.038	0.545± 0.058
ReCEval-Intra	0.500± 0.000	0.100± 0.000	0.167± 0.000
ReCEval-Inter	0.755± 0.047	0.671± 0.021	0.590± 0.066
LLM-Judge	**1.000± 0.000**	**1.000± 0.000**	**1.000± 0.000**
ClaimTrees-v5i2			
ARES-1	**1.000± 0.000**	**1.000± 0.000**	**1.000± 0.000**
ARES-0.95	0.995± 0.011	0.998± 0.005	0.996± 0.008
Entail-Prev	0.990± 0.010	0.981± 0.019	0.985± 0.015
Entail-Base	0.863± 0.009	0.823± 0.007	0.815± 0.013
ROSCOE-LI-Self	0.965± 0.030	0.951± 0.036	0.956± 0.033
ROSCOE-LI-Source	0.635± 0.054	0.642± 0.058	0.555± 0.057
ReCEval-Intra	0.500± 0.000	0.167± 0.000	0.250± 0.000
ReCEval-Inter	0.695± 0.029	0.721± 0.020	0.594± 0.038
LLM-Judge	0.978± 0.016	0.960± 0.028	0.967± 0.024
ClaimTrees-v5i5			
ARES-1	0.988± 0.028	0.991± 0.020	0.989± 0.026
ARES-0.95	**0.998± 0.004**	**0.998± 0.005**	**0.998± 0.005**
Entail-Prev	0.988± 0.013	0.990± 0.010	0.989± 0.011
Entail-Base	0.930± 0.019	0.950± 0.012	0.936± 0.018
ROSCOE-LI-Self	0.938± 0.012	0.955± 0.008	0.943± 0.012
ROSCOE-LI-Source	0.661± 0.006	0.736± 0.033	0.649± 0.011
ReCEval-Intra	0.500± 0.000	0.278± 0.000	0.357± 0.000
ReCEval-Inter	0.665± 0.024	0.826± 0.008	0.642± 0.034
LLM-Judge	0.982± 0.017	0.983± 0.017	0.982± 0.017

Table A10: **GPT-4o-mini (ClaimTrees)** ARES does not differ much from other methods in inserted errors that do not affect downstream reasoning. v5i2 means 5 valid claims and 2 inserted claims.