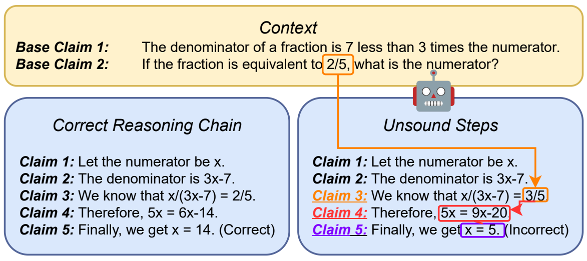
Probabilistic Soundness Guarantees in LLM Reasoning Chains

Weiqiu You[†] **Anton Xue**[⋆] **Shreya Havaldar**[†] **Delip Rao**[†] **Helen Jin**[†]
Chris Callison-Burch[†] **Eric Wong**[†]

[†]University of Pennsylvania
[⋆]University of Texas at Austin

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. This inductive method yields a nuanced score for each step and provides certified statistical guarantees of its soundness, rather than a brittle binary label. ARES achieves state-of-the-art performance across four benchmarks (72.1% Macro-F1, +8.2 points) and demonstrates superior robustness on very long synthetic reasoning chains, where it excels at detecting propagated errors (90.3% F1, +27.6 points). [1]



Base Claim 1: The denominator of a fraction is 7 less than 3 times the numerator.
Base Claim 2: If the fraction is equivalent to 2/5, what is the numerator?

Claim 1: Let the numerator be x.
Claim 2: The denominator is 3x-7.
Claim 3: We know that x/(3x-7) = 2/5.
Claim 4: Therefore, 5x = 6x-14.
Claim 5: Finally, we get x = 14. (Correct)

Claim 1: Let the numerator be x.
Claim 2: The denominator is 3x-7.
Claim 3: We know that x/(3x-7) = 3/5
Claim 4: Therefore, 5x = 9x-20
Claim 5: Finally, we get x = 5. (Incorrect)

Figure 1: **Faulty LLM reasoning due to propagated errors from ungrounded and invalid steps.** An unsound step is a step that is either **ungrounded** (incorrect with respect to the context), **invalid** (logically incorrectly derived), or contains **propagated** errors. In this example, **Step 3 is ungrounded** because it contains information different from the base claim 2. **Step 4 is invalid** because it contains an incorrect mathematical computation. **Step 5 is a propagated error**, even though it is logically correct with respect to Step 4. This figure is adapted from an example from Lee and Hockenmaier (2025).

1 Introduction

Large Language Models (LLMs) are taking on increasingly sophisticated reasoning tasks in critical fields like medicine and scientific discovery. Yet, a fundamental challenge persists: the chain-of-thought processes that lead to their outputs are frequently flawed with errors (Huang et al., 2025; Lyu et al., 2024). This critically compromises the reliability of LLM-generated content, diminishing user confidence and impeding the broader adoption of LLMs in high-stakes applications (Agarwal et al., 2024; Chen and Mueller, 2024).

As illustrated in Figure 1, one type of error is an *ungrounded step*—a step that is incorrect with respect to the given context. For example, the model might incorrectly copy a 2/5 in the context to be 3/5. Another common error is an *invalid derivation*—for example, deriving $5x = 9x - 20$ from $x/(3x - 7) = 3/5$—which is a logical misstep or miscalculation (Lee and Hockenmaier, 2025). A third type of error involves *error propagation*: even if the logic is valid, an incorrect starting assumption can lead to a wrong conclusion. For instance, using the incorrect claim $5x = 9x - 20$ to derive $x = 5$ is logically valid but the derived claim is incorrect due to the initial error (Tyagi et al., 2024).

Current error detection methods typically aim to identify all errors at once. For example, LLM judges are prompted to evaluate the entire chain and assess each step for correctness (Tyagi et al., 2024; He et al., 2025). Similarly, Process Reward Models (PRMs) are language models trained with step-level classification heads on this same objective (Lightman et al., 2024).

However, existing error detection methods often fall short. Specifically, they are often distracted by the presence of propagated errors (He et al., 2025; Turpin et al., 2023; Dhuliawala et al., 2024). In the example from Figure 1, if steps 3, 4, and 5

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

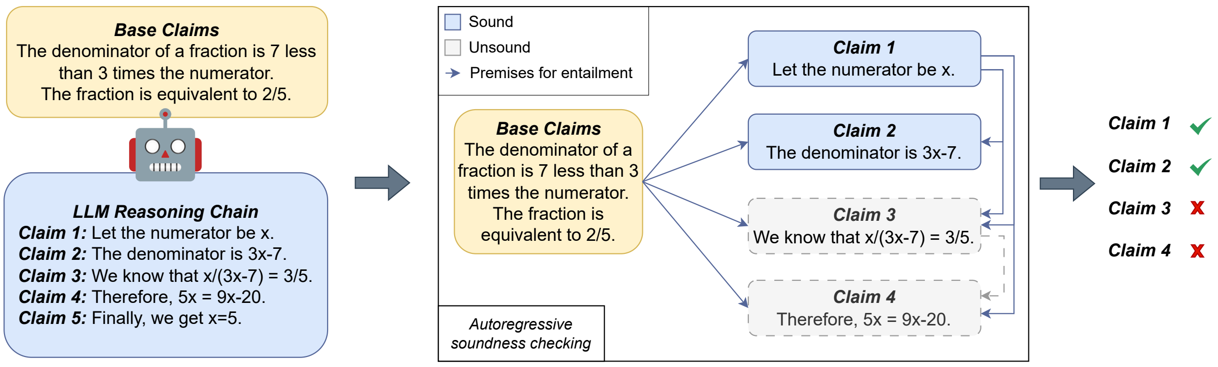

Base Claims
The denominator of a fraction is 7 less than 3 times the numerator.
The fraction is equivalent to 2/5.
LLM Reasoning Chain
Claim 1: Let the numerator be x.
Claim 2: The denominator is 3x-7.
Claim 3: We know that x/(3x-7) = 3/5.
Claim 4: Therefore, 5x = 9x-20.
Claim 5: Finally, we get x=5.
Sound
Unsound
Premises for entailment
Base Claims
The denominator of a fraction is 7 less than 3 times the numerator. The fraction is equivalent to 2/5.
Claim 1
Let the numerator be x.
Claim 2
The denominator is 3x-7.
Claim 3
We know that x/(3x-7) = 3/5.
Claim 4
Therefore, 5x = 9x-20.
Autoregressive soundness checking
Claim 1 ✓
Claim 2 ✓
Claim 3 ✗
Claim 4 ✗

Figure 2: (**Autoregressive Soundness Checking**) When we verify an LLM generated reasoning chain, we can break the context and reasoning chain down to base claims and derived claims. An autoregressive soundness checker can then check each derived claim step-by-step, using only claims already identified to be sound as the premise.

are evaluated together, an LLM may incorrectly mark step 5 as sound by incorrectly relying on step 4, which is invalid. This highlights the need for robust methods that can assess the soundness of each step without being adversely distracted by prior errors.

To address this issue, we draw inspiration from human reasoning. Humans typically review claims sequentially, and disregard previously unsound statements when evaluating subsequent ones (Johnson-Laird, 2010; Mukherjee et al., 2025). In contrast, LLMs struggle to ignore prior errors, which causes naive detection methods to fail at simultaneously identifying and localizing all errors in a reasoning chain (Wu et al., 2024a; Song and Tavanapong, 2024). To overcome this limitation, we develop Autoregressive Reasoning Entailment Stability (ARES), a probabilistic framework that evaluates the soundness of each reasoning step based on its expected entailment probability, conditioned only on previously-occurring, sound claims (Figure 2). We iteratively evaluate each claim as follows: *entailed* claims are retained as premises for subsequent steps, while *non-entailed* claims are discarded. For *uncertain* claims, retention is probabilistic based on the entailment model. This adaptation not only improves error detection but also enables us to give certified guarantees on the robustness of a reasoning chain.

Our contributions are highlighted as follows.

- We introduce Autoregressive Reasoning Entailment Stability (ARES), a novel probabilistic framework for evaluating claims in LLM reasoning chains. This framework uniquely assesses each step by conditioning only on previously verified sound claims, ensuring a robust and adaptable evaluation.
- We design a computationally and sample-efficient autoregressive algorithm for entailment estimation within this framework. Crucially, this algorithm provides sample-efficient certifications of entailment with rigorous statistical guarantees, a capability absent in prior methods.
- We demonstrate that ARES accurately certifies both sound and unsound reasoning steps, particularly excelling in long chains prone to error propagation. ARES significantly surpasses existing approaches and generalizes across diverse reasoning tasks.

2 Soundness in Reasoning Chains

We aim to identify and certify errors within LLM-generated chain-of-thought (CoT) reasoning. To this end, this section formalizes reasoning chains in terms of their constituent claims (Section 2.1), introduces the concept of probabilistic entailment between these claims (Section 2.2), and defines a notion of soundness that incorporates internal groundedness, validity, and the entailment of a final hypothesis (Section 2.3).

2.1 Claims and Sequences of Claims

A reasoning chain is conceptualized as a sequence of claims, where a claim is the assertion of a proposition. For instance, "The denominator is $3x - 7$" is a claim regarding a component of an algebraic expression, while "We know that $\frac{x}{3x-7} = \frac{2}{5}$" is a claim that synthesizes prior information about an equation. The granularity of claims is domain-dependent; it is permissible for a claim to range from an atomic statement or a single sentence (e.g., "We can simplify $\frac{x}{3x-7} = \frac{2}{5}$ to $5x = 6x - 14$.") to more extensive segments like entire theorems or proofs.

For a more formal discussion of our method, we let $\mathcal{C}$ denote the set of all possible claims, and $\mathcal{C}^\star$ represent the set of all possible sequences of claims. An example of such a sequence is as follows:

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

which consists of the following individual claims:

$$\begin{aligned}\text{"Let the numerator be } x\text{"} &\in \mathcal{C},\\ \text{"The denominator is } 3x-7\text{"} &\in \mathcal{C},\\ \text{"We know that } \tfrac{x}{3x-7} = \tfrac{2}{5}\text{"} &\in \mathcal{C}.\end{aligned}$$

This distinction between individual claims and sequences of claims is important for discussing the inclusion and exclusion of items from a premise during logical entailment, which we define next.

2.2 Probabilistic Entailment of Claims

To capture the notion of logical entailment between claims expressed in natural language, we introduce probabilistic entailment models. This approach is motivated by the inherent fuzziness and ambiguity often present in natural language reasoning (Zadeh, 2008; Yu et al., 2024). Formally, a probabilistic entailment model $\mathcal{E} : \mathcal{C}^\star \times \mathcal{C} \to [0, 1]$ accepts a sequence of claims as a premise, $P \in \mathcal{C}^\star$, and a single claim as a hypothesis, $H \in \mathcal{C}$. It then returns a scalar value representing the probability that the premise P entails the hypothesis H. For instance, consider the premise and hypothesis pair

$$\begin{aligned}P = \big(&\text{"Sarah put on her running shoes."},\\ &\text{"She stretched by the sidewalk."},\\ &\text{"The sun was setting."}\big)\\ H = &\text{ "Sarah is going for an evening run."}\end{aligned}$$

A probabilistic entailment model might output $\mathcal{E}(P, H) = 0.85$. This score reflects the linguistic and social ambiguity in inferring the certainty of an "evening run" from the actions of "donning running shoes and stretching". Such a fuzzy, probabilistic approach generalizes classical Boolean logic, where the output is strictly 1 for entailment and 0 for non-entailment. [2]

[2] We distinguish between a non-entailed claim (not logically following premises) and a provably false claim (factually incorrect). For instance, "Sarah lives in Philadelphia" is not entailed but not demonstrably false.

2.3 Reasoning Chains and Soundness

To analyze the step-by-step reasoning of LLMs, particularly in CoT processes, we conceptualize the output as a *reasoning chain*. This chain initiates with a set of provided statements or contextual information, designated as *base claims*. Following these, the LLM autoregressively produces a sequence of subsequent statements, which we term *derived claims*. This entire sequence is formally represented as:

$$(C_1, \ldots, C_n, C_{n+1}, \ldots, C_{n+m}) \in \mathcal{C}^\star \qquad (1)$$

where $C_1, \ldots, C_n$ are the base claims, and $C_{n+1}, \ldots, C_{n+m}$ are the derived claims.

This partition is methodologically crucial. Base claims $(C_1, \ldots, C_n)$ serve as the foundational premises for a given reasoning task; their factual accuracy is given and assumed to be validated by external mechanisms. Instead, we focus on assessing whether each derived claim (C_{n+i} for $i = 1, \ldots, m$) is soundly inferred from the set of preceding statements. To begin, we define a deterministic (i.e., "hard") version of soundness, where all derived claims are entailed with certainty.

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

$$\begin{aligned}\mathcal{E}((C_1, \ldots, C_n), C_{n+1}) &= 1\\ &\vdots\\ \mathcal{E}((C_1, \ldots, C_{n+m-1}), C_{n+m}) &= 1\end{aligned} \qquad (2)$$

The concept of hard soundness provides a precise, albeit strict, benchmark for evaluating the logical integrity of a reasoning chain: it requires every derived claim to be perfectly entailed by its predecessors. However, LLM-generated reasoning chains often deviate from this ideal. Therefore, while hard soundness serves as an important theoretical standard of correctness, it cannot give nuanced measures of error, particularly in long reasoning chains. This necessitates more flexible methods for measuring claim soundness even in the presence of errors, which we address next.

3 Soundness Checks via Autoregressive Reasoning Entailment Stability

We now consider the practical certification of LLM-generated reasoning chains. These chains are formed autoregressively: starting from an initial sequence of base claims $C_1, \dots, C_n$, the LLM iteratively generates the derived claims $C_{n+1}, \dots, C_{n+m}$ where each

$$C_{n+k} = \mathsf{LLM}(C_1, \dots, C_{n+k-1}),$$

for reasoning steps $k = 1, \dots, m$. We aim to quantify the reliability of this process using a sequence of *entailment stability scores*: $\tau_1, \dots, \tau_m \in [0, 1]$, where each τ_k denotes how reliably the k-th derived claim C_{n+k} is entailed with respect to its preceding claims $C_1, \dots, C_{n+k-1}$. The connection between entailment and error detection is straightforward: if τ_k is small, then C_{n+k} is likely an error.

However, a well-principled and computationally tractable formulation of τ_k is non-trivial when entailment is probabilistic. Critically, hard soundness is incompatible with non-binary outputs, and it is not immediately clear how uncertain premises should be evaluated. ARES addresses this: Section 3.1 motivates probabilistic entailment using insights from human psychology, LLM empirics, and mathematical logic. Subsequently, Section 3.2 formalizes our approach, defines ARES, and details its efficient Monte Carlo estimation.

3.1 Entailment with Probabilistic Premises

The key challenge lies in accurately assessing entailment when premises are probabilistically uncertain. Our main insight is to calculate an overall likelihood by averaging across various probable combinations of that uncertain information.

Our approach is motivated by several observations. In **human cognition**, people naturally discount or ignore dubious statements when reasoning (Johnson-Laird, 2010). Similarly, lengthy contexts are often filtered to remove irrelevant and erroneous claims to improve **LLM performance** on reasoning tasks (Mukherjee et al., 2025). These observations collectively motivate our development of a probabilistic entailment framework based on premise subsets.

To measure the reliability of a hypothesis H with respect to a premise P containing k claims with uncertain soundness, we consider all 2^k configurations of inclusion and exclusion for P's claims. Each configuration is represented by a binary vector $\alpha \in \{0, 1\}^k$, where $\alpha_i = 1$ indicates inclusion of claim C_i and $\alpha_i = 0$ indicates exclusion. This leads to the following natural measure of *stability* for H with respect to P and $\mathcal{E}$:

$$\tau(\mathcal{E}, P, H) = \sum_{\alpha \in \{0,1\}^k} \mathcal{E}(P(\alpha), H) \cdot \Pr[\alpha], \quad (3)$$

where $\Pr[\alpha]$ is the probability of this specific configuration of premise claim inclusions, and depends on the base and derived claims, as well as the entailment model $\mathcal{E}$, which we discuss next.

3.2 Autoregressive Reasoning Entailment Stability with Efficient Sampling

We previously established a method for calculating the entailment of a single hypothesis based on a set of premises that might be uncertain (Equation (3)). Now, we will extend this concept to evaluate an entire LLM-generated reasoning chain, which consists of multiple steps. Our goal is to compute a sequence of *entailment stability scores*, denoted as $\tau_1, \dots, \tau_m$, where each score τ_k quantifies the reliability of the k-th derived claim, C_{n+k}.

The core challenge remains: how to reliably judge a claim when the preceding claims it relies on are themselves not entirely trustworthy? Our approach, **ARES**, solves this by autoregressively assessing each claim while accounting for the soundness of previous claims. In particular, when we evaluate the k-th derived claim, we consider all possible combinations of soundness for the preceding $n + k - 1$ claims. The stability score, τ_k, is then the expected entailment of the current claim, averaged across all sound combinations.

To formalize this, we represent a particular combination of inclusion or exclusion of previous claims using a binary vector $\alpha \in \{0, 1\}^{n+k-1}$, where let $\alpha_i = 1$ denote the inclusion of claim C_i and let $\alpha_i = 0$ denote its exclusion. The probability of this combination $\Pr[\alpha]$ is calculated recursively as follows:

- **Base Case** ($k = 1$)**:** For the first derived claim, C_{n+1}, the premises are the initial base claims $C_1, \dots, C_n$. We assume that each base claim C_i is associated with a prior probability of soundness p_i that is given. Therefore, let:

$$\Pr[\alpha_{1:n}] = \prod_{i=1}^{n} p_i^{\alpha_i} (1 - p_i)^{\alpha_i} \quad (4)$$

- **Inductive Case** ($k > 1$)**:** For subsequent claims, the probability of a specific premise combination $\alpha_{1:n+k}$ depends on two factors: the probability of the preceding combination

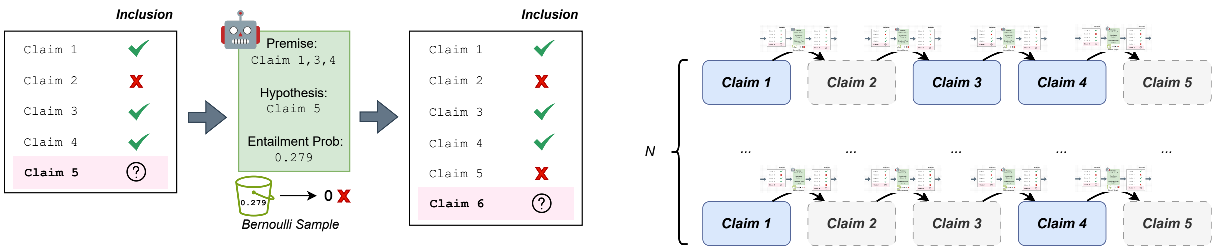


Figure 3: **(Estimating ARES)** (Left) The entailment rate of each derived claim is autoregressively computed. We first randomly initialize a premise (denoted by α) according to the base priors $p_1, \ldots, p_n$. Then, for each derived claim, we compute its entailment rate with respect to the premise set. Finally, we add this derived claim to the premise set with probability equal to its entailment rate. (Right) This is run in parallel across N instances.

($\Pr[\alpha_{1:n+k-1}]$) and the entailment probability of the new claim given that preceding combination. That is, a claim is added to our set of "sound" premises based on how strongly the current set entails it, where let $\Pr[\alpha_{1:n}] =$

$$\Pr[\alpha_{1:n+k-1}] \cdot \mathcal{E}(C(\alpha_{1:n+k-1}), C_{n+k}) \quad (5)$$

where $C(\alpha_{1:n+k-1})$ are the claims indexed by $\alpha_{1:n+k-1} \in \{0,1\}^{n+k-1}$.

Using the above definition for $\Pr[\alpha]$, we may quantify how likely each combination of previous claims may affect the current entailment. In particular, we naturally define the *entailment stability score* τ_k for the k-th derived claim as a marginalization over all combinations of its predecessors:

$$\tau_k = \sum_{\alpha \in \{0,1\}^{n+k-1}} \mathcal{E}(C(\alpha), C_{n+k}) \cdot \Pr[\alpha] \quad (6)$$

However, directly computing τ_k is highly inefficient, as it requires summing over 2^{n+k-1} possible combinations of premise entailment. Instead, we estimated it by sampling the premise combinations:

$$\hat{\tau}_k = \frac{1}{N} \sum_{i=1}^{N} \mathcal{E}(C(\alpha^{(i)}), C_{n+k}), \quad (7)$$

where let $\alpha^{(1)}, \ldots, \alpha^{(N)} \sim \{0,1\}^{n+k-1}$ be i.i.d. sampled according to Algorithm 1 and in Figure 3. Additionally, note that $\hat{\tau}_k$ converges rapidly to τ as the number of samples N grows, allowing us to obtain a rigorous statistical guarantee on our stability scores as a function of the number of samples.

Theorem 3.1 (Estimating Entailment Stability). *Let $N \geq \frac{\log(2m/\delta)}{2\varepsilon^2}$ for any $\varepsilon > 0$ and $\delta > 0$. For any entailment model $\mathcal{E}$ and reasoning chain $(C_1, \ldots, C_{n+m})$, define $\tau_1, \ldots, \tau_m$ as in Equation (7). Then, with probability at least $1 - \delta$, this estimate has error $|\hat{\tau}_k - \tau_k| \leq \varepsilon$ for all k.*

Proof. See Appendix A. □

Algorithm 1 Estimating ARES

```
Require: Reasoning chain $(C_1, \ldots, C_{n+m})$, tolerance $(\varepsilon, \delta)$, base priors $p_1, \ldots, p_n$, and entailment model $\mathcal{E}$.
 1: $N \leftarrow \frac{\log(2m/\delta)}{2\varepsilon^2}$
 2: for $i = 1, \ldots, N$ do
 3:     $\alpha_1^{(i)} \sim \text{Bernoulli}(p_1), \ldots, \alpha_n^{(i)} \sim \text{Bernoulli}(p_n)$
 4:     for $k = 1, \ldots, m$ do
 5:         $p_{n+k}^{(i)} \leftarrow \mathcal{E}(C(\alpha_{1:n+k-1}^{(i)}), C_{n+k})$
 6:         $\alpha_{n+k}^{(i)} \sim \text{Bernoulli}(p_{n+k}^{(i)})$
 7:     end for
 8: end for
 9: for $k = 1, \ldots, m$ do
10:     $\hat{\tau}_k = \frac{1}{N} \sum_{i=1}^{N} p_{n+k}^{(i)}$
11: end for
```

Error Detection. Recall the connection between entailment stability and error detection: the lower a claim's entailment stability τ_k, the greater its error. Consider a simple thresholding mechanism: if some estimate $\hat{\tau}_k$ falls below a prescribed error threshold, then we mark the derived claim C_{n+k} as erroneous. In the following, we demonstrate the empirical effectiveness of this procedure.

4 Evaluating ARES for Estimating Probabilistic Soundness

ARES performs error detection by estimating the entailment stability of each derived claim and applying a thresholding mechanism. We next run experiments to validate the performance of ARES against multiple baselines on diverse benchmarks.

Experiment Setup. We consider comparisons with LLM-Judge, which takes the whole reasoning chain as input and makes a judgment for each step together, Entail-Prev and Entail-Base, which judge the entailment of a claim based on all preceding claims and only base claims respectively, and two ROSCOE (Golovneva et al., 2023) and two ReCEval (Prasad et al., 2023) correctness methods that are based on pairwise comparisons.

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 1: **(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.

For LLMs, we use GPT-4o-mini (OpenAI, 2024) and Qwen3-4B (Yang et al., 2025). For a PRM, we used Qwen2.5-Math-PRM-7B (Zhang et al., 2025). We evaluate on four datasets: PRMBench (Song et al., 2025), DeltaBench (He et al., 2025), ClaimTrees (our synthetic data), and CaptainCookRecipes (graph-based recipe dataset derived from CaptainCook4D (Peddi et al., 2024)). We evaluate using Macro-recall, Macro-precision and Macro-F1 following the literature (He et al., 2025). To compute the error threshold for the entailment scores, we first sweep over all the values that occur in the training split and select the one that maximizes Macro-F1. We repeat this process in a 5-fold cross-validation where each time we use one fold for validation and four folds for testing and report the average and standard deviation. Additional details and analyses can be found in Appendix C.

4.1 RQ1: Does ARES work better than baseline methods on Benchmarks?

We measure ARES's ability to identify errors in natural reasoning chains using PRMBench and DeltaBench. With GPT-4o-mini backbone entailment model, we find that ARES achieves the best Macro-F1 scores on both datasets, shown in Table 1. LLM-Judge performs poorly on DeltaBench while Entail-Base underperforms on PRMBench. DeltaBench's long reasoning chains appear to confuse LLM-Judge when making holistic judgments. For Qwen3-4B, Entail-Base performs slightly better, while all other methods lag behind. Our inspection reveals that Qwen3-4B-based entailment models frequently classify next claims as entailed, suggesting limited capability for judging complex reasoning. Additional experiments in Appendix C.9 show that ARES can also achieve further improve-

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 2: In this ClaimTrees example, after two correct steps (**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 steps.

ments on top of a PRM backbone.

4.2 RQ2: In what setting does ARES identify more errors than baselines?

To pinpoint where ARES most effectively outperforms other methods, we needed to test it in settings with long reasoning chains and clear error propagation. Since existing benchmarks often lack these specific features, we constructed two controllable datasets designed to isolate these challenges:

- **ClaimTrees:** A synthetic logical reasoning dataset involving proofs over abstract graphs.
- **CaptainCookRecipes:** A graph-rule-based dataset adapted from the cooking task graphs in CaptainCook4D (Peddi et al., 2024).

We designed these datasets specifically to test the core reasoning capabilities of each method, controlling for confounding variables. For example, ClaimTrees uses abstract symbols and shuffled rules to mitigate ordering bias. In both datasets, we represent the underlying rules (e.g., logical rules, recipe actions) as base claims. We then intentionally remove a key base claim—like a rule in a proof or an ingredient in a recipe—to create unsound derivations, allowing us to precisely track how the initial error propagates through the reasoning chain. Further details can be found in Appendix C.5.

Experiments on these controlled datasets confirm that ARES excels at identifying propagated errors, especially in long chains. As demonstrated in Figure 4, ARES maintains a high Macro-F1 score even as chains become very long, whereas the performance of all baseline methods deteriorates sharply after just a few steps. For example, ARES sustains a Macro-F1 score of at least 89% on chains up to 50 steps long, while other methods fall into the 30–40% range. The results in Table 1 further highlight this robust performance across our synthetic datasets, with an example shown in Table 2.

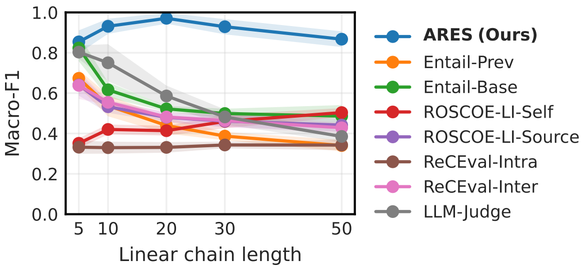


Figure 4: **(ClaimTrees) GPT-4o-mini.** ARES can robustly identify error propagations in long reasoning chains, whereas other methods fail.

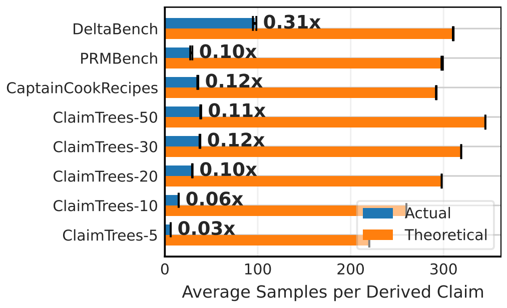


Figure 5: **(Per-Claim Samples)** ARES in practice only uses 0.03x to 0.31x the number of samples required by the theoretical bound.

We further discuss in Appendix B that only ARES satisfies all important desiderata for error detection while other methods fail to.

4.3 RQ3: Is ARES computationally efficient?

While ARES samples multiple combinations of previous claims to check soundness, it is implemented efficiently to avoid redundant LLM calls for the same premise-hypothesis pairs. Figure 5 shows the theoretical versus actual samples used for each derived claim and complete reasoning chain, respectively. For ClaimTrees, the average total samples per example increases with chain length. With shorter chains (ClaimTrees-5), we achieve extreme efficiency at only 0.03x of theoretical samples needed. DeltaBench uses the most samples but still achieves 0.31x of theoretical samples

Method	**PRMBench**	**DeltaBench**	**ClaimTrees-10**	**CaptainCookRecipes**
ARES-ε0.1	**0.640**	**0.708**	**0.931**	0.633
ARES-ε0.2	0.599	0.697	0.926	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 3: **(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.

needed. DeltaBench needing more samples indicates greater uncertainty in the entailment model's outputted probability for this dataset. In ideal cases where the entailment model outputs only 1 or 0 for every derived claim, we need just one sample per claim each step.

To have a more direct analysis of efficiency, we conduct another analysis on the performance vs. sample size trade-off across all datasets on GPT-4o-mini in Table 3. We find that in practice ARES's performance is stable even with fewer samples, indicating potential further computational savings. On synthetic benchmarks ClaimTrees and CaptainCookRecipes, there is no significant difference for $\varepsilon = 0.1$ to 0.4, while more differences are shown for PRMBench and DeltaBench.

4.4 RQ4: Is ARES useful for selecting Best-of-N generations?

To test if ARES is useful for downstream tasks, we run a best-of-N experiment–selecting the generation scoring the highest in soundness among multiple generations, and see which methods' selected generations have better accuracies. We perform the experiment on PRMBench, which contains both original and modified process. We use both as the two generations, with the original process as the correct generation and the modified process as the incorrect generation. We select the best of two candidate chains by either averaging all step scores or using only the final step's score.

Results in Table 4 show that when using the score of the final step—a stricter and often more decisive measure—ARES significantly outperforms all other methods. Notably, the performance of simpler methods like Entail-Prev collapses on this stricter metric. This highlights ARES's unique strength in maintaining a sound evaluation of steps throughout the entire reasoning chain, making its final assessment particularly reliable. Therefore, ARES is a strong and robust predictor of downstream task performance.

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

Table 4: **(PRMBench Best-of-N)** ARES is a robust predictor for downstream task performance. The table shows the selection accuracy (higher is better) for choosing the correct reasoning chain from two options. Using the final step's score is a stricter evaluation, where ARES's performance stands out. **Bold** indicates the best performance within bootstrap standard error.

4.5 Ablations

We conduct ablations on ClaimTrees to examine the robustness and design choices of ARES.

Robustness to Irrelevant Claims. We tested our method on reasoning trees with varying widths (irrelevant sources) and depths (path lengths), where an error was introduced by removing a single rule. As shown in Table A9, ARES remains stable across all configurations. In contrast, baseline methods degrade, suffering more from increased depth than width. ARES is thus capable of filtering irrelevant claims, and error propagation in long chains is the primary reason other approaches fail.

Base Claim Inclusion Probability. We also vary the probability p of including base claims and compare probabilistic vs. binary entailment models. Results in Table A11 show that $p = 1$ with a probabilistic model consistently performs best, while binary models sometimes benefit from $p < 1$. Choosing $p = 1$ is therefore often both accurate and efficient, as it avoids resampling base claims and reduces variance. However, as ClaimTrees has a clearer cut in soundness, the case can be different when entailment contains ambiguity.

Additional Results. We further examine benign errors (inserted rules that do not affect downstream steps) in Appendix C.8. All methods perform equally well, unlike the irrelevant-claim setting where baselines degrade.

4.6 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.

5 Related Work

Guarantees for Single-Step Explanations. Research in interpretability has shifted from heuristic evaluation toward formal guarantees for individual predictions. One major branch is inherently interpretable models, which provide guarantees such as optimality (Angelino et al., 2018; Ustun and Rudin, 2019), monotonicity (Gupta et al., 2016; Milani Fard et al., 2016), or faithfulness by construction in deep learning models (Bassan and Katz, 2023; You et al., 2025). A second branch focuses on post-hoc explanations for black-box models, including conservation guarantees (Bach et al., 2015; Shrikumar et al., 2017; Montavon et al., 2017), local accuracy, missingness, and consistency (Lundberg and Lee, 2017; Wu et al., 2024b), precision (Ribeiro et al., 2018), minimality (Ferreira et al., 2022; Bassan and Katz, 2023), sufficiency (Bassan et al., 2025), or certified interventions via recourse methods (Ustun et al., 2019; Karimi et al., 2020). A third branch of work provides certified robustness guarantees, particularly in the form of *stability certificates* (Xue et al., 2023; Kim et al., 2024; Jin et al., 2025), which have been applied to model explainability in medicine (Achara et al., 2025). Our work extends stability guarantees to LLM reasoning chains.

Guarantees and Verification for Multi-Step Reasoning. While single-step methods are well-studied, LLMs often generate multi-step reasoning chains prone to *hallucinations* and *error propagation* (Huang et al., 2025; Lyu et al., 2024). A significant body of work focuses on practical error detection without formal guarantees. Common approaches include self-consistency checkers (Manakul et al., 2023; Dhuliawala et al., 2024) and automated verifiers such as LLM Judges (Tyagi et al., 2024; He et al., 2024, 2025), Process Reward Models (PRMs) (Lightman et al., 2024; Zhang et al., 2025), and specialized entailment models (Dalvi et al., 2021; Havaldar et al., 2025).

To provide more rigor, logic-based verifiers assess *soundness*, though often limiting to pairwise checks (Golovneva et al., 2023; Prasad et al., 2023) or taking a brittle approach to propagated errors (Mukherjee et al., 2025). An early formal guarantee, Faithful Chain-of-Thought, ensures reasoning traces deterministically yield the final answer (Lyu et al., 2023). While a suite of evaluation benchmarks exists (Tyagi et al., 2024; Jacovi et al., 2024; Song et al., 2025; Zheng et al., 2025; He et al., 2025), a unified standard for error definition is still emerging (Lee and Hockenmaier, 2025; Mukherjee et al., 2025). Recent statistical methods provide calibrated step-level reliability for generation but focus on isolated predictions within the chain (Feng et al., 2025; Quach et al., 2024; Cherian et al., 2024). In contrast, our work introduces *propagation-aware guarantees* that certify *entire reasoning chains*, ensuring upstream errors do not corrupt downstream judgments.

6 Conclusion

Current methods cannot reliably detect propagation errors in LLM reasoning chains. To address this limitation, we introduce ARES, a novel framework for certifying the soundness of an LLM's reasoning chain. By quantifying the soundness of each claim through autoregressive sampling, ARES provides a fine-grained inductive guarantee on the chain's overall reliability that is useful in error detection. Empirically, ARES demonstrates superior performance, robustly identifying errors in lengthy and complex reasoning chains where existing methods fail due to error propagation.

Limitations

ARES's performance is tied to the quality of the entailment model; poor calibration can lead to unreliable scores. However, our model-agnostic approach allows for easily substituting better-calibrated components via techniques like temperature scaling to improve performance.

While our efficient sampling algorithm mitigates computational overhead, ARES is more intensive than simpler approaches. Additionally, our approach assumes that the claims are already decomposed and, therefore, cannot detect errors at the sub-claim level. We leave this for future work, noting it would increase computational costs.

Finally, our evaluation on four datasets with two models demonstrates effectiveness across different domains, but it is not exhaustive. Performance could also be improved with better LLM prompts.

Ethical Considerations

Our research framework for detecting reasoning errors raises several ethical considerations. While ARES can improve the reliability of AI reasoning, it may create false confidence in underlying models when they consistently make undetected errors. Implementation requires careful evaluation across diverse domains to prevent biases from propagating through certified reasoning chains. Additionally, computing resource requirements for probabilistic sampling may limit accessibility to well-resourced institutions. We acknowledge the importance of transparent reporting of ARES's limitations and recommend human oversight when used in high-stakes domains such as healthcare or legal applications to ensure responsible deployment.

Potential Risks

While ARES offers significant advantages over existing methods, there are several potential risks to consider. First, the probabilistic sampling approach introduces computational overhead, though our efficient algorithm mitigates this. Second, ARES may create false confidence in underlying LLM reasoning when it consistently fails to detect certain types of errors. Implementation requires careful evaluation across diverse domains to prevent biases from propagating through certified reasoning chains. Additionally, computing resources for probabilistic sampling may limit accessibility to well-resourced institutions. Human oversight remains essential when deployed in high-stakes domains like healthcare or legal applications to ensure responsible use and reliable reasoning verification.

Acknowledgments

We thank Mayank Keoliya for discussion about this work. This research was partially supported by a gift from AWS AI to Penn Engineering's ASSET Center for Trustworthy AI, by ASSET Center Seed Grant, ARPA-H program on Safe and Explainable AI under the award D24AC00253-00, by NSF award CCF 2442421, by the AI2050 program at Schmidt Sciences (Grant G-25-67983), and by funding from the Defense Advanced Research Projects Agency's (DARPA) SciFy program (Agreement No. HR00112520300). The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Akshit Achara, Esther Puyol Anton, Alexander Hammers, and Andrew P King. 2025. Invisible attributes, visible biases: Exploring demographic shortcuts in mri-based alzheimer's disease classification. *arXiv preprint arXiv:2509.09558*.

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

Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. 2018. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18(234):1–78.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Shahaf Bassan, Yizhak Yisrael Elboher, Tobias Ladner, Matthias Althoff, and Guy Katz. 2025. Explaining, fast and slow: Abstraction and refinement of provable explanations. In *Forty-second International Conference on Machine Learning*.

Shahaf Bassan and Guy Katz. 2023. Towards formal xai: formally approximate minimal explanations of neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 187–207. Springer.

Jiuhai Chen and Jonas Mueller. 2024. Quantifying uncertainty in answers from any language model and enhancing their trustworthiness. In *Proceedings* *of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5186–5200, Bangkok, Thailand. Association for Computational Linguistics.

John Cherian, Isaac Gibbs, and Emmanuel Candes. 2024. Large language model validity via enhanced conformal prediction methods. *Advances in Neural Information Processing Systems*, 37:114812–114842.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Yu Feng, Ben Zhou, Weidong Lin, and Dan Roth. 2025. BIRD: A trustworthy bayesian inference framework for large language models. In *The Thirteenth International Conference on Learning Representations*.

João Ferreira, Manuel de Sousa Ribeiro, Ricardo Gonçalves, and João Leite. 2022. Looking Inside the Black-Box: Logic-based Explanations for Neural Networks. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, pages 432–442.

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

Maya Gupta, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, and Alexander Van Esbroeck. 2016. Monotonic calibrated interpolated look-up tables. *Journal of Machine Learning Research*, 17(109):1–47.

Shreya Havaldar, Hamidreza Alvari, John Palowitch, Mohammad Javad Hosseini, Senaka Buthpitiya, and Alex Fabrikant. 2025. Entailed between the lines: Incorporating implication into NLI. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32274–32290, Vienna, Austria. Association for Computational Linguistics.

Hangfeng He, Hongming Zhang, and Dan Roth. 2024. SocREval: Large language models with the socratic method for reference-free reasoning evaluation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2736–2764, Mexico City, Mexico. Association for Computational Linguistics.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Z.y. Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and Bo Zheng. 2025. Can large language models detect errors in long chain-of-thought reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18468–18489, Vienna, Austria. Association for Computational Linguistics.

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

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. 2024. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4615–4634, Bangkok, Thailand. Association for Computational Linguistics.

Helen Jin, Anton Xue, Weiqiu You, Surbhi Goel, and Eric Wong. 2025. Probabilistic stability guarantees for feature attributions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. 2020. Model-agnostic counterfactual explanations for consequential decisions. In *International conference on artificial intelligence and statistics*, pages 895–905. PMLR.

Chaehyeon Kim, Weiqiu You, Shreya Havaldar, and Eric Wong. 2024. Evaluating groups of features via consistency, contiguity, and stability. In *The Second Tiny Papers Track at ICLR 2024*.

Jinu Lee and Julia Hockenmaier. 2025. Evaluating step-by-step reasoning traces: A survey. *arXiv preprint arXiv:2502.12289*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. 2024. Towards faithful model explanation in NLP: A survey. *Computational Linguistics*, 50(2):657–723.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Mahdi Milani Fard, Kevin Canini, Andrew Cotter, Jan Pfeifer, and Maya Gupta. 2016. Fast and flexible monotonic functions with ensembles of lattices. *Advances in neural information processing systems*, 29.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. 2017. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern recognition*, 65:211–222.

Sagnik Mukherjee, Abhinav Chinta, Takyoung Kim, Tarun Anoop Sharma, and Dilek Hakkani Tur. 2025. Premise-augmented reasoning chains improve error identification in math reasoning with LLMs. In *Forty-second International Conference on Machine Learning*.

OpenAI. 2024. Gpt-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. Accessed: 2025-05-19.

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. 2024. Captaincook4d: A dataset for understanding errors in procedural activities. In *Advances in Neural Information Processing Systems*, volume 37, pages 135626–135679. Curran Associates, Inc.

Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. 2023. ReCEval: Evaluating reasoning chains via correctness and informativeness. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10066–10086, Singapore. Association for Computational Linguistics.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. 2024. Conformal language modeling. In *The Twelfth International Conference on Learning Representations*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMlR.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25299–25346, Vienna, Austria. Association for Computational Linguistics.

Seok Hwan Song and Wallapak Tavanapong. 2024. How much do prompting methods help llms on quantitative reasoning with irrelevant information? In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2128–2137.

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

Nemika Tyagi, Mihir Parmar, Mohith Kulkarni, Aswin Rrv, Nisarg Patel, Mutsumi Nakamura, Arindam Mitra, and Chitta Baral. 2024. Step-by-step reasoning to solve grid puzzles: Where do LLMs falter? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19898–19915, Miami, Florida, USA. Association for Computational Linguistics.

Berk Ustun and Cynthia Rudin. 2019. Learning optimized risk scores. *Journal of Machine Learning Research*, 20(150):1–75.

Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable recourse in linear classification. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 10–19.

Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. 2024a. How easily do irrelevant inputs skew the responses of large language models? In *First Conference on Language Modeling*.

Yinjun Wu, Mayank Keoliya, Kan Chen, Neelay Velingker, Ziyang Li, Emily J Getzen, Qi Long, Mayur Naik, Ravi B Parikh, and Eric Wong. 2024b. Discret: Synthesizing faithful explanations for treatment effect estimation. *Proceedings of machine learning research*, 235:53597.

Anton Xue, Rajeev Alur, and Eric Wong. 2023. Stability guarantees for feature attributions with multiplicative smoothing. *Advances in Neural Information Processing Systems*, 36:62388–62413.

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, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Weiqiu You, Helen Qu, Marco Gatti, Bhuvnesh Jain, and Eric Wong. 2025. Sum-of-parts: Self-attributing neural networks with end-to-end learning of feature groups. In *Forty-second International Conference on Machine Learning*.

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

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

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516, Vienna, Austria. Association for Computational Linguistics.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024, Vienna, Austria. Association for Computational Linguistics.

A Proofs

Theorem 3.1 (Estimating Entailment Stability). *Let $N \geq \frac{\log(2m/\delta)}{2\varepsilon^2}$ for any $\varepsilon > 0$ and $\delta > 0$. For any entailment model $\mathcal{E}$ and reasoning chain $(C_1, \dots, C_{n+m})$, define $\tau_1, \dots, \tau_m$ as in Equation (7). Then, with probability at least $1 - \delta$, this estimate has error $|\hat{\tau}_k - \tau_k| \leq \varepsilon$ for all k.*

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

$$\Pr\left(\bigcap_{i=n+1}^{n+m} \mathcal{A}_i\right) = 1 - \Pr\left(\bigcup_{i=n+1}^{n+m} \bar{\mathcal{A}}_i\right) \geq 1 - \delta. \tag{8}$$

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

$$\begin{aligned}
&\Pr\left(\bigcup_{i=n+1}^{n+m} \bar{\mathcal{A}}_i\right) \leq \sum_{i=n+1}^{n+m} \Pr\left(\bar{\mathcal{A}}_i\right) && \text{(Boole's)} \\
&= \sum_{i=n+1}^{n+m} \Pr\left(|\hat{\tau}_i - \tau_i| \geq \varepsilon\right) && (9) \\
&\leq \sum_{i=n+1}^{n+m} 2\exp(-2N\varepsilon^2) && \text{(Hoeffding's)} \\
&= 2m\exp(-2N\varepsilon^2) && (10) \\
&\leq \delta \quad \text{when } N \geq \frac{\log(2m/\delta)}{2\varepsilon^2}, && (11)
\end{aligned}$$

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.

Table A5 shows that only ARES satisfies all desiderata while none of the baseline methods does.

C Experiments

C.1 Entailment Model

We instantiate an LLM judge to assess whether a hypothesis h is supported by a premise P that may contain multiple claims. We use two output modes: (i) *binary* YES/NO, mapped to $\{1, 0\}$; and (ii) a *7-point Likert scale* where the LLM must output exactly one label from {*Very Likely*, *Likely*, *Somewhat Likely*, *Neutral*, *Somewhat Unlikely*, *Unlikely*, *Very Unlikely*}. We convert the label to a probability via

$$\begin{aligned}
\text{Very Likely} &\mapsto 1.0 \\
\text{Likely} &\mapsto 0.8 \\
\text{Somewhat Likely} &\mapsto 0.6 \\
\text{Neutral} &\mapsto 0.5 \\
\text{Somewhat Unlikely} &\mapsto 0.4 \\
\text{Unlikely} &\mapsto 0.2 \\
\text{Very Unlikely} &\mapsto 0.0
\end{aligned}$$

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

Table A5: (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.

which fits in a double-column layout.

Contradiction scoring. For contradiction judgments (e.g., in ROSCOE and ReCEVal), we apply the same labels but invert the mapping so that higher scores indicate stronger contradiction:

$$\begin{aligned}\text{Very Unlikely} &\mapsto 1.0\\ \text{Unlikely} &\mapsto 0.8\\ \text{Somewhat Unlikely} &\mapsto 0.6\\ \text{Neutral} &\mapsto 0.5\\ \text{Somewhat Likely} &\mapsto 0.4\\ \text{Likely} &\mapsto 0.2\\ \text{Very Likely} &\mapsto 0.0\end{aligned}$$

with the binary case mapped analogously (YES/NO $\mapsto 1/0$ for "is contradiction?").

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. The hyperparameters $\delta = 0.1$ and $\varepsilon = 0.1$ are chosen following previous work (Jin et al., 2025). The prompts are tuned by manually examining the results and seeing that they are able to produce reasonable results for entailment on each dataset. We observe that DeltaBench prefers simpler and more natural prompts while for other datasets the prompts need to have more specifications for what entail vs. not entail mean. The prompts are released in the codebase https://github.com/fallcat/ares.

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. The standard deviation is reported for the 5 folds. Specific packages used can be found in our codebase https://github.com/fallcat/ares.

C.4 Licenses for Artifacts

All datasets, models, and code used in this work follow the licenses and terms specified by their original authors, as cited in the corresponding papers. We do not redistribute these artifacts under different terms. For the artifacts we release (https://github.com/fallcat/ares) code and evaluation scripts), we provide them under the MIT license.

Our use of existing artifacts is consistent with their intended use as specified by the original authors (e.g., datasets accessed for research purposes were used only in research contexts). For the artifacts we create, we specify that they can be freely used, modified, and redistributed under the MIT license.

C.5 Controllable Datasets

ClaimTrees. ClaimTrees is a synthetic dataset in which the reasoning chain reasons starts from a state such as AZ, and reason all the way to another state, say VD. 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 A6. In this example, rule H3 -> VD does not actually exist, and thus the reasoning steps starting from the third derived step onward are unsound claims. We can construct reasoning chains with arbitrary length and errors occurring at different

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	0.310±0.043
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 underline is the second best.

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 A6: Long Chain Example for ClaimTrees

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 A8. With propagated errors present, only ARES is able to capture all errors.

C.6 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.7 Best-of-N Results

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

C.8 Ablations

On ClaimTrees, we also construct two other types of trees to inspect the strengths of ARES: wide trees with more sources, imitating the behavior of inserting irrelevant claims, and trees with inserted outgoing edges that are not in the base claims, imitating the case of benign errors in the reasoning chains that do not result in error propagation.

Inserting Irrelevant Claims. We investigated whether ARES better identifies errors in long versus wide chains with the same number of nodes. We constructed wide reasoning trees with multiple sources and one sink, with a rule removed in the middle. Starting from one source, we derive to the sink node, where the rule error can be in the path from source to sink or in other paths.

Table A9 shows that for reasoning chains about trees of depth 5 with 3 sources, other methods show more significant performance drops while ARES maintains high performance. For wide reasoning trees, other methods don't drop as much, but in trees with greater depth, their error rates increase significantly while ARES remains stable.

Inserting Benign Errors. We also examined cases where non-existing rules are inserted in reasoning chains but don't affect later steps. Table A10 shows that all methods perform equally well when errors don't cause downstream propagation.

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	0.722 ± 0.016	0.725 ± 0.017	0.715 ± 0.011
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	0.546 ± 0.008	**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 underline is the second best.

How do different choice of p for base claims and granularity of the entailment model affect the performance? We allow using different p_i for the flexibility to not include all the base claims in the premise, and we want to see the impact of different design choices. Table A11 shows ablations for using $p = 1$ vs. $p = 0.95$ as well as using granular vs binary entailment models (which use strict $\{0, 1\}$). $p = 1$ consistently performs better with a probabilistic entailment model, while a binary entailment model sometimes benefits from $p = 0.95$. Thus we can effectively choose $p = 1$, including all base claims, which can greatly reduce the actual computation cost as the effective sample size now only depends on entailment of derived claims.

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.

D AI Assistants in Research

We used AI assistants (e.g., ChatGPT, Cursor) to support coding frameworks, generate visualizations, and revise writing for clarity and readability.

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 A8: **(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.

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.

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	<u>0.898</u>	<u>0.913</u>	<u>0.891</u>
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	<u>0.947</u>	<u>0.951</u>	<u>0.948</u>
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	<u>0.971</u>	<u>0.971</u>	<u>0.971</u>
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**	<u>0.972</u>	**0.971**
ARES-0.95	0.931	0.934	0.929
ARES-bin-1	<u>0.967</u>	**0.973**	<u>0.969</u>
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**	<u>0.899</u>	**0.890**
ARES-0.95	0.871	<u>0.871</u>	0.867
ARES-bin-1	0.887	**0.904**	0.886
ARES-bin-0.95	<u>0.892</u>	0.892	<u>0.888</u>
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 A11: **GPT-4o-mini (ClaimTrees)** ARES consistently identifies errors in long reasoning chains while other methods gradually fail.