Confidence over Time: Confidence Calibration with Temporal Logic for Large Language Model Reasoning

Zhenjiang Mao[1*], **Anirudhh Venkat**[1*], **Artem Bisliouk**[2], **Akshat Kothiyal**[1]
Sindhura Kumbakonam Subramanian[1], **Saithej Singhu**[1], **Ivan Ruchkin**[1]

[1]University of Florida
[2]University of Mannheim

Abstract

Large Language Models (LLMs) increasingly rely on long-form, multi-step reasoning to solve complex tasks such as mathematical problem solving and scientific question answering. Despite strong performance, existing confidence estimation methods typically reduce an entire reasoning process to a single scalar score, ignoring how confidence evolves throughout the generation. As a result, these methods are often sensitive to superficial factors such as response length or verbosity, and struggle to distinguish correct reasoning from confidently stated errors. We propose to characterize the stepwise confidence signal using Signal Temporal Logic (STL). Using a discriminative STL mining procedure, we discover temporal formulas that distinguish confidence signals of correct and incorrect responses. Our analysis found that the STL patterns generalize across tasks, and numeric parameters exhibit sensitivity to individual questions. Based on these insights, we develop a confidence estimation approach that informs STL blocks with parameter hypernetworks. Experiments on multiple reasoning tasks show our confidence scores are more calibrated than the baselines.

1 Introduction

Large Language Models (LLMs) have demonstrated strong performance on long-form, multi-step reasoning involved in mathematical problem solving, scientific explanation, and legal analysis (Wei et al., 2022; Wang et al., 2023; Yao et al., 2023). Despite their fluency, LLMs remain prone to factually incorrect or logically flawed responses that are expressed with high apparent certainty (Kadavath et al., 2022). Recent work has also explored enforcing formal temporal constraints during LLM generation to improve compliance with safety specifications, for example via logic-based verification techniques (Kamath et al., 2025). Specifically, we estimate a sample-level probability that a generated response is correct conditioned on the input, and evaluate its calibration against empirical correctness. More broadly, the need for calibrated confidence estimates has also been highlighted in other safety-critical learning settings (Mao et al., 2024), including image-controlled autonomy and safety prediction for embodied systems (Sobolewski et al., 2025). This motivates studying calibration not only as a predictive score, but as a reliability signal for downstream decision making (Mao et al., 2025b).

Most existing approaches to confidence or uncertainty quantification for LLMs reduce the generation process to a single scalar score. Representative methods include aggregating token-level probabilities, computing entropy-based statistics, or estimating confidence through output consistency across multiple samples (Ott et al., 2018). While effective in certain settings, these share a *common limitation:* they collapse a sequential reasoning process into a scalar, discarding information about how confidence evolves over time. This makes the confidence depend heavily on how the response is verbalized (length, verbosity, and stylistic fillers), since token-level aggregates can be dominated by high-probability but low-semantic-content tokens. Figure 2 illustrates this failure mode on three real responses from Qwen3-8B on Big-Bench-Hard, where the third example is incorrect yet receives a deceptively high scalar score because a single-step confidence drop is washed out by the mean.

We posit that confidence in long-form LLM reasoning should be viewed not as a static scalar, but as a *temporal signal* (Dong et al., 2018; Ribeiro et al., 2020) that reflects the evolution of model certainty over reasoning steps. To capture such temporal patterns, we derive stepwise confidence signals from segmented reasoning responses and characterize them using *Signal Temporal Logic (STL)* (Maler and Nickovic, 2004), which provides a formal lan-

[*]Equal contribution.

Findings of the Association for Computational Linguistics: ACL 2026, pages 9952–9976

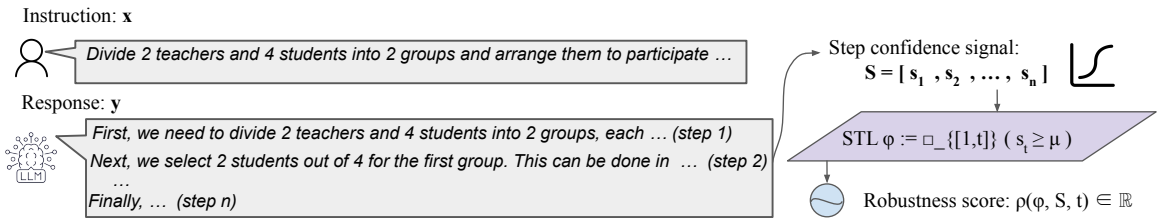


Figure 1: Overview of stepwise confidence modeling with STL. Token-level probabilities are aggregated into a stepwise confidence signal $\mathbf{S}$, on which STL formulas are evaluated to capture temporal confidence patterns via robustness scores.

guage for specifying temporal constraints over real-valued signals (Linard and Tumova, 2020; Nicoletti et al., 2024). STL yields a quantitative robustness score that indicates how well a confidence trajectory satisfies a given temporal specification (Figure 1). Guided by STL robustness, we use a discriminative STL mining procedure to automatically discover temporal formulas that best distinguish correct from incorrect LLM responses.

We analyze the structure and variability of temporal confidence patterns across tasks and instances, and use these insights to guide the design of our confidence estimation framework. Based on these insights, we design an interpretable confidence estimation framework that separates temporal structure from parameter instantiation. We fix STL structures to preserve interpretability, while enabling question-level adaptation of their continuous parameters via a hypernetwork (Ha et al., 2016; Garnelo et al., 2018).

Our contributions are as follows: (1) We introduce discriminative STL mining for modeling stepwise confidence signals in long-form LLM reasoning; (2) We show that effective temporal confidence structures exhibit consistent patterns across datasets and task types, suggesting reusable temporal signatures, particularly for incorrect reasoning; (3) Within a fixed task setting, we show that while a single STL structure can generalize across questions, its optimal parameters can exhibit significant question-level variability; (4) We propose an interpretable confidence estimation framework that combines STL robustness with a hypernetwork-based parameterization, improving standard calibration metrics while preserving temporal structure.

Practical use cases. Reliable per-response confidence enables several downstream behaviors that scalar estimators support poorly: *abstention* in high-stakes QA when reliability falls below a threshold, *selection* among candidates in Best-of-N or mixture-of-experts routing, and *early stopping* when stepwise failure patterns appear before generation finishes. Because our STL blocks expose interpretable failure modes (e.g., a sharp drop or oscillation), each of these decisions is also auditable. We focus on calibration quality in this paper and view the downstream uses as natural deployments enabled by it.

2 Background and Problem

2.1 Probability in LLMs

We consider a family of language tasks indexed by task type $\mathcal{T}$. Given an input prompt $\mathbf{x}$, a Large Language Model (LLM) $\mathcal{M}$ generates a response $\mathbf{y} = [y_1, \ldots, y_m]$ in an autoregressive manner. At each decoding step, the model assigns a probability to the generated token conditioned on the input and preceding tokens. We denote by $c_{\mathcal{T}}(\mathbf{x}, \mathbf{y}) \in \{0, 1\}$ a binary correctness predicate indicating whether the generated response $\mathbf{y}$ is correct for input $\mathbf{x}$ under task type $\mathcal{T}$, as determined by task-specific evaluation protocols. The goal of confidence estimation is to produce a scalar score reflecting the likelihood that a generated response is correct. A common baseline is to use the joint generation probability $P(\mathbf{y} \mid \mathbf{x})$ as confidence. However, this quantity suffers from severe length bias, assigning smaller values to longer responses regardless of correctness (Li et al., 2025b), making it unsuitable for long-form reasoning.

Problem Formulation. Our objective is to estimate a confidence score $\hat{P} \in [0, 1]$ that approximates the posterior probability $P(c_{\mathcal{T}}(\mathbf{x}, \mathbf{y}) = 1 \mid \mathbf{x})$. We assume access to labeled input–output pairs $\{(\mathbf{x}_i, \mathbf{y}_i, c_i)\}_{i=1}^{N}$ and learn a confidence model $\mathcal{S}$ mapping $(\mathbf{x}, \mathbf{y})$ to $\hat{P} = \mathcal{S}(\mathbf{x}, \mathbf{y})$.

2.2 Evaluation Metrics

We evaluate confidence estimation quality using standard calibration metrics, primarily *Expected*

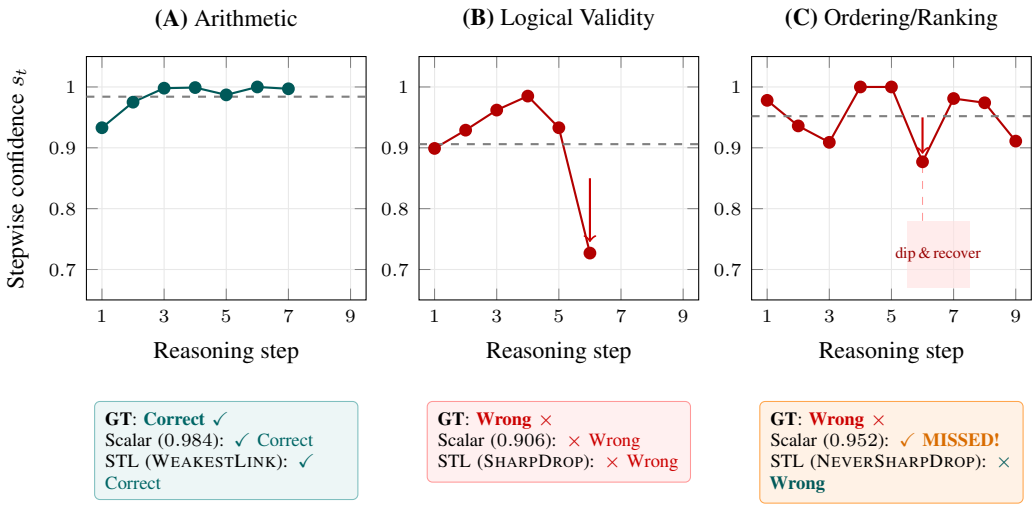


Figure 2: Three example confidence trajectories from Qwen3-8B on BBH. Solid lines show stepwise confidence; dashed lines show the scalar average. (A) A correct response with uniformly high confidence, correctly identified by both scalar and STL estimators. (B) An incorrect response with a sharp confidence drop, detected by both methods. (C) An incorrect "dip-and-recover" case: the scalar average remains high and misses the error, while STL detects it.

Calibration Error (ECE) (Guo et al., 2017) and the *Brier Score (BS)* (Brier, 1950), which quantify the alignment between predicted confidence and empirical correctness. Formal definitions are provided in the appendix.

2.3 Stepwise Confidence Signal and STL

We represent a generated response $\mathbf{y}$ as a sequence of n consecutive segments, where each segment corresponds to a contiguous subsequence of tokens. This segmentation induces a stepwise representation of the generation process and serves as the temporal axis for subsequent confidence modeling. In practice, such a decomposition may follow linguistic boundaries (e.g., sentences) or semantic reasoning steps (e.g., those produced by Chain-of-Thought prompting). Our formulation requires consistent segmentation within each task — but does not assume any specific method, which is treated as a design choice.

We define the *stepwise confidence signal* s_j as the aggregated confidence associated with segment u_j. Specifically, we compute s_j as the arithmetic mean of the probabilities of its constituent tokens:

$$s_j = \frac{1}{L_j} \sum_{k=0}^{L_j-1} P(y_{t_j+k} \mid \mathbf{y}_{<t_j+k}, \mathbf{x}), \qquad (1)$$

where L_j denotes the number of tokens in segment u_j and t_j is the index of its first token.

This process transforms the token-level probability sequence into a coarser-grained confidence signal $\mathbf{s} = [s_1, s_2, \ldots, s_n]$, which serves as the foundation for logic-based modeling.

STL Syntax. Signal Temporal Logic (STL) provides a formalism for specifying temporal constraints over real-valued signals (Maler and Nickovic, 2004). In this work, we use STL to describe temporal patterns over the stepwise confidence signal $\mathbf{S} = [s_1, \ldots, s_n]$.

An STL formula consists of a fixed logical structure φ together with continuous parameters $\boldsymbol{\theta}$, such as predicate thresholds and temporal bounds, and is denoted by $\varphi_{\boldsymbol{\theta}}$. The basic building block is a predicate μ, typically of the form $\mu \equiv s_t \geq c$, where larger values indicate higher confidence.

STL Robustness. STL formulas are evaluated under quantitative robustness semantics, yielding a score that summarizes how well a stepwise confidence trajectory satisfies a temporal pattern.

3 Related Work

3.1 Uncertainty Quantification for LLMs

We study response-level *confidence estimation*: predicting the probability that a specific generated response is correct, rather than decomposing aleatoric/epistemic uncertainty. While uncertainty quantification (UQ) is well-studied in machine translation (MT) (Ott et al., 2018; Wang et al., 2021) and classification (Beck et al., 2016), free-form LLM generation is harder due to its effectively unbounded output space. Existing approaches can be broadly categorized into *four lines of research*:

logit-based, verbalized uncertainty, consistency-based, and internal-state-based.

Logit-based: Early autoregressive UQ approaches rely on output probabilities, including cumulative token-wise likelihoods, predictive entropy, and max probability (Kadavath et al., 2022; Jiang et al., 2021). Length-normalized variants have been proposed to mitigate sentence-length bias (Malinin and Gales, 2021). However, for long-form reasoning, they conflate semantic and auxiliary tokens, often causing miscalibration.

Verbalized Uncertainty: Verbalized uncertainty prompts or fine-tunes LLMs to express confidence in natural language (Lin et al., 2022; Tian et al., 2023), including prompt/sampling schemes (Xiong et al., 2024) and targeted fine-tuning (Amayuelas et al., 2024). Although promising, these methods often require additional training/tuning or domain-specific adaptation to perform well (Kadavath et al., 2022; Manakul et al., 2023).

Consistency-based: To address the intractable reasoning space and semantic equivalence, a prominent line of work estimates uncertainty from multiple sampled outputs in a semantic space. *Self-Consistency* (Wang et al., 2023) estimates confidence from agreement among diverse generations, while *Semantic Entropy* (Kuhn et al., 2023) formalizes this idea via semantic clustering and entropy-based scoring. Extensions use entailment-based black-box clustering (Farquhar et al., 2024), self-reflection (Kirchhof et al., 2025), paraphrased prompting (Li et al., 2025a), and clarification sequences (Hou et al., 2024). Despite their effectiveness, these methods typically require multiple model runs and often rely on additional components such as external encoder models, introducing non-trivial inference overhead.

Internals-based: Internals-based methods estimate uncertainty from hidden states or token attribution, e.g., SAR (Duan et al., 2024); related approaches exploit model internals (Beigi et al., 2024; Azaria and Mitchell, 2023) but can be expensive (e.g., factorial SAR) or require large in-domain validation sets. Despite progress, LLMs remain overconfident in zero-shot and out-of-domain settings (Liu et al., 2025; Desai and Durrett, 2020), and structured reasoning remains comparatively under-explored. Recent work has begun to model confidence as a temporal signal over multi-step reasoning, including post hoc analysis of confidence trajectories using Signal Temporal Logic and temporal-aware uncertainty quantification through recurrent confidence aggregation (Mao and Venkat, 2026).

3.2 Temporal Logics

Temporal logic, originally developed for formal verification (Pnueli, 1977), provides a principled framework for specifying temporal patterns over signals, and *Signal Temporal Logic* (STL) (Maler and Nickovic, 2004) extends it to real-valued signals with quantitative robustness semantics. In this work, we apply STL to discrete-time confidence trajectories derived from stepwise LLM reasoning, using robustness to summarize temporal confidence patterns. The quantitative nature of STL has enabled its integration into neuro-symbolic learning (Garcez et al., 2019), including differentiable constraint enforcement in robotics and time-series models (Raman et al., 2014; Yu et al., 2023), as well as logic structure mining methods such as TeLEx (Jha et al., 2019).

In NLP, temporal logic has mainly been used for controlled generation (Lu et al., 2021) and consistency/fact-checking (Durrett and Klein, 2014), focusing on text semantics. STL has also been used to analyze confidence trajectories post hoc (Mao et al., 2025a); in contrast, we learn STL structures and parameters as *discriminative* confidence estimators under calibration objectives, with question-adaptive parameters via hypernetworks.

4 Temporal Confidence Patterns

This section studies whether stepwise confidence trajectories exhibit distinct temporal "confidence signatures" for correct versus incorrect reasoning. We introduce a discriminative STL mining procedure that discovers STL formulas whose robustness separates correct and incorrect reasoning traces, yielding a formula set $\Phi = \Phi_{\text{pos}} \cup \Phi_{\text{neg}}$. We then use the mined formulas to investigate structural generalization across tasks (RQ1) and parameter sensitivity at the question level (RQ2).

4.1 Discriminative STL Mining

To discover interpretable temporal patterns that characterize LLM confidence, we adopt a discriminative STL mining procedure adapted from TELEX (Jha et al., 2019), aimed at separating correct and incorrect reasoning traces.

Incremental Structure Learning. Given a dataset of stepwise confidence signals $\mathcal{D} = \{(\mathbf{S}_i, c_i)\}_{i=1}^{N}$, where each stepwise confidence signal $\mathbf{S} =$

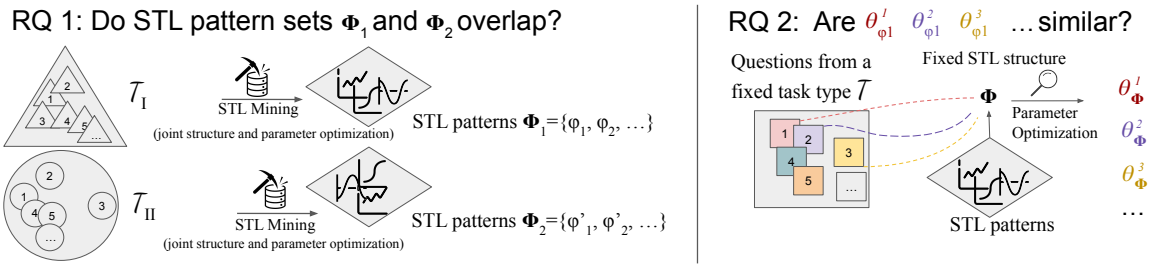


Figure 3: Conceptual illustration of Research Question 1 (RQ1) and Research Question 2 (RQ2). RQ1 examines whether the STL structures mined from different task types are shared or task-specific. RQ2 studies, under a fixed STL structure, whether the associated parameters θ learned during optimization generalize across questions.

$[s_1, \ldots, s_n]$ derived from an LLM response, we consider a predefined and shared set of atomic predicates $\mathcal{P}$ over the signal, each in the form of a simple inequality (e.g., $s_t \geq \mu$ or $s_t \leq \mu$), where μ is a learnable threshold parameter. The mining process then proceeds incrementally over this fixed predicate space:

Step 1: Base Template Instantiation. Given the stepwise confidence signal $\mathbf{S} = [s_1, \ldots, s_n]$ and atomic predicates $\mathcal{P}$, we initialize the search from a predefined set of primitive STL templates $\mathcal{T}_{\text{base}}$. Representative examples include $\Box_{[a,b]}(s_t \geq \mu)$ and $\Diamond_{[a,b]}(s_t \geq \mu)$, and for their full list, see Appendix C. For each template, we fit its parameters to maximize discriminative power between correct and incorrect responses, using a classification-based objective. This step yields a set of parameterized base templates $\{(\varphi_k, \theta_k)\}$ instantiated from predefined structures.

Step 2: Robustness-Based Signal Augmentation. To enable the construction of nested STL formulas without explicit enumeration, we adopt a robustness-based signal augmentation strategy (*signal lifting*) (Jha et al., 2019; Donzé and Maler, 2010; Kong et al., 2014), which uses robustness signals of mined templates as additional features for subsequent formula composition.

Step 3: Structural Composition. We construct more expressive STL formulas by composing discriminative base templates through temporal nesting and Boolean combinations, enabling the representation of diverse temporal confidence patterns.

Dual-Class Template Mining. Using the candidate STL formulas generated through Steps 1–3, we perform dual-class discriminative mining to identify two complementary categories of temporal confidence patterns.

Correct Patterns (Φ_{pos}): These formulas characterize temporal confidence signatures of correct reasoning, assigning higher scores to correct responses than to incorrect ones.

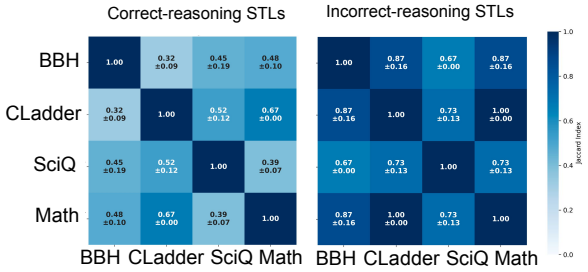


Figure 4: Pairwise similarity of mined STL formulas across four reasoning benchmarks. Left: similarity of STL patterns characterizing correct reasoning. Right: similarity of STL patterns characterizing incorrect reasoning. Similarity is measured using the Jaccard index over mined formula sets.

Incorrect Patterns (Φ_{neg}): These formulas capture confidence patterns associated with hallucinations or reasoning failures, where stronger satisfaction indicates a higher likelihood of incorrectness.

These dual sets of patterns operationalize the hypothesis that correct and incorrect reasoning traces exhibit distinct temporal confidence signatures.

Having performed STL mining, we study the generalizability of the mined patterns from two complementary analytical perspectives. Figure 3 provides a schematic overview of the two research questions, illustrating structural generalization across task types (RQ1) and parameter sensitivity under a fixed formula (RQ2).

4.2 RQ1: Generalizable Structure over Tasks

Do effective structural STL patterns generalize across different task types, or are they domain-specific? This question examines whether the discriminative STL structures governing temporal confidence are shared across task types or remain task-specific. We assess this by mining STL formulas on different datasets and measuring the structural overlap of the discovered patterns across tasks.

We empirically observe a clear asymmetry in the

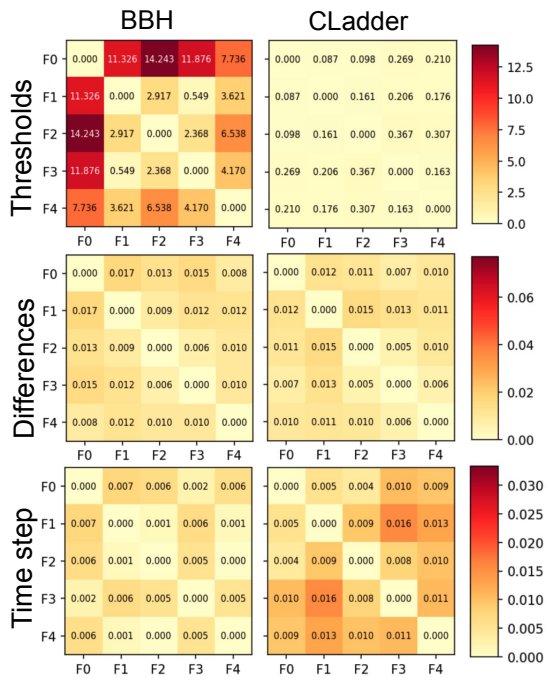


Figure 5: RQ2: Pairwise differences of optimized STL parameters across individual questions under a fixed STL structure. Parameters are grouped into three categories: predicate thresholds, difference-related parameters, and time-related parameters. Results are shown for BBH and CLadder using Qwen3-8B.

structural generalization of temporal confidence patterns. Across multiple reasoning benchmarks, STL formulas associated with incorrect reasoning exhibit consistently high structural similarity across tasks, whereas those associated with correct reasoning show substantially lower overlap. This trend holds across backbone models and persists at the subtask level within BBH (see Fig. 4 and Appendix). These findings indicate that failure-related confidence dynamics are structurally stable and largely reusable across tasks, while successful reasoning trajectories remain task-dependent. This asymmetry motivates treating Φ_{neg} as transferable structural templates, while allowing Φ_{pos} to adapt to task-specific characteristics.

4.3 RQ2: Parameter Sensitivity of Questions

Assuming a fixed STL structure φ for a given task type $\mathcal{T}$, do the associated parameters $\boldsymbol{\theta}$ generalize across questions, or are they question-specific?

Given a shared STL structure φ identified for a task type, we investigate whether its continuous parameters $\boldsymbol{\theta}$ can be shared across questions or require instance-level adaptation. We conceptually compare task-level parameter optimization with question-specific parameter fitting. If optimal parameters cluster tightly across questions, a single task-level parameterization would suffice; conversely, substantial variation would indicate the need for question-adaptive parameters.

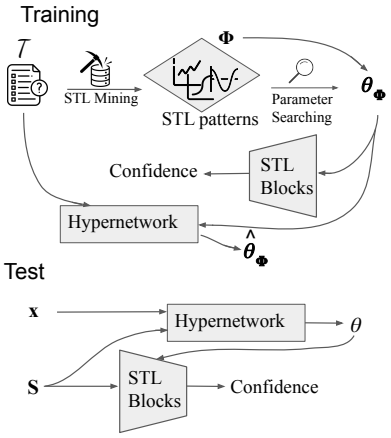


Figure 6: Confidence quantification pipeline using STL blocks. During training, discriminative STL mining identifies temporal formula structures, which are instantiated as STL blocks with learnable parameters. At test time, the STL blocks are fixed and applied to stepwise confidence signals to compute scalar confidence.

Under a fixed STL structure, we observe pronounced variability in the optimized parameters across individual questions. As shown in Fig. 5, predicate thresholds and difference-related parameters vary significantly across instances, whereas temporal parameters remain comparatively stable. This effect is dataset-dependent: BBH exhibits substantially higher parameter dispersion than CLadder, particularly for threshold-related parameters. Further analysis indicates that this sensitivity is not explained by lexical or semantic similarity between questions: as reported in Appendix E (Table 13), datasets with higher intra-dataset question similarity do not necessarily yield more stable parameter configurations, and BBH in particular combines high semantic similarity with high parameter variability. These results suggest that a single global parameterization is insufficient even when the underlying temporal structure is shared, motivating instance-adaptive parameterization mechanisms that preserve structural interpretability while allowing question-level flexibility, which we explore in Section 5.

5 Confidence Estimation with STL Blocks

5.1 STL Blocks Design

Given a stepwise confidence signal $\mathbf{S} = [s_1, \ldots, s_T]$, an STL block evaluates a fixed STL

formula φ under quantitative robustness semantics to produce a scalar confidence estimate $\hat{p} \in [0, 1]$. Specifically, the block computes an aggregated robustness score $\rho(\varphi, \mathbf{S}) \in \mathbb{R}$ and maps it to a probabilistic confidence via a monotonic, differentiable transformation: $\hat{p} = \omega(\alpha\, \rho(\varphi, \mathbf{S}) + \beta)$, where $\omega(\cdot)$ denotes the sigmoid function and α, β are learnable parameters. This mapping preserves robustness ordering while producing a bounded output suitable for probabilistic training.

Aggregation across patterns. Our design explicitly separates *interpretable temporal structure*, encoded by fixed STL formulas, from *continuous confidence calibration*, controlled by a small set of learnable parameters. When multiple STL formulas are used, each STL block produces an individual confidence score. Scores from *positive* patterns (higher indicates correctness) and *negative* patterns (higher indicates incorrectness) are aggregated separately, with negative-pattern scores converted via $1 - \hat{p}$ so that all components consistently indicate correctness. The final confidence estimate is obtained by a weighted aggregation of these components into a single scalar prediction. Implementation details are deferred to the appendix.

As illustrated in Fig. 6, our confidence estimation pipeline instantiates mined STL formulas as differentiable blocks and applies them to stepwise confidence signals to produce scalar confidence scores. Our primary approach predicts *question-adaptive parameters* for these STL blocks using a hypernetwork conditioned on the input question. To assess the necessity of instance-level adaptation, we also consider an ablated variant in which both the STL structure and its parameters are fixed after mining and shared across all test inputs.

5.2 Question-Adaptive Hypernetworks

A central challenge in question-level confidence estimation is balancing adaptability with interpretability. While different questions may require different parameter values, adapting the temporal structure itself would undermine the interpretability of the learned STL formulas. We therefore fix the STL structure discovered during mining and allow only its continuous parameters to vary across instances.

Given a fixed STL formula structure φ, we adopt a hypernetwork-based parameterization to instantiate the parameters of each STL block on a per-question basis. This enables question-specific adaptation to varying reasoning complexity and confidence dynamics while preserving the interpretable temporal structure encoded by the STL formulas.

Hypernetwork Architecture. We fix the STL formula structure φ discovered during the mining stage and use a hypernetwork $\mathcal{H}_\psi$ to predict its parameters on a per-instance basis. The hypernetwork takes as input the original prompt $\mathbf{x}$ and the corresponding stepwise confidence signal $\mathbf{S}$, and outputs a parameter vector: $\theta = \mathcal{H}_\psi(\mathbf{x}, \mathbf{S})$, where θ includes all continuous parameters associated with the STL block, including temporal bounds, predicate thresholds, and the parameters α and β of the robustness-to-confidence mapping. In practice, the input prompt $\mathbf{x}$ is encoded using a fixed text encoder, while the confidence signal $\mathbf{S}$ is summarized using simple temporal statistics (e.g., mean, variance, and slope) or a lightweight sequence encoder. The resulting representations are concatenated and fed into a multi-layer perceptron to produce θ_i.

Training Objective. The hypernetwork is trained end-to-end using the same discriminative objective as in the mining stage. For each question, the predicted parameters θ_i are instantiated into the fixed STL structure φ, and the resulting robustness score $\rho(\varphi_{\theta_i}, \mathbf{S}_i)$ is mapped to a confidence estimate P_i. The hypernetwork parameters ψ are learned by minimizing the discriminative loss over the training set: $\min_\psi \quad \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(c_i, \hat{P}_i\big)$, where $\hat{P}_i$ is obtained by instantiating the predicted parameters $\theta_i = \mathcal{H}_\psi(\mathbf{x}_i, \mathbf{S}_i)$ into the fixed STL structure φ, evaluating the resulting robustness, and applying the robustness-to-confidence mapping. The loss $\mathcal{L}$ corresponds to the discriminative objective.

6 Experiments

We evaluate our method on four reasoning benchmarks: GAOKAO-Math, CLadder, SciQ, and Big-Bench-Hard (BBH). All tasks are formulated as multiple-choice or binary decision problems, enabling consistent correctness labeling. Experiments are conducted using three large language models: Qwen3-8B, Gemma-3-12B, and Llama-3-8B. Unless otherwise specified, all methods operate in a single-sample inference regime; only self-consistency baselines draw multiple samples.

We compare our STL-based confidence estimator against representative baselines: logit-based (Lin et al., 2022), consistency-based (Portillo Wightman et al., 2023), self-evaluation (Kadavath et al., 2022) approaches, and internals-based method (Duan et al., 2024; Beigi et al., 2024). All

Table 1: Ablation results on uncertainty calibration across base models and datasets. We report ECE (↓) and Brier Score (↓) as mean ± std over 5 folds; best results are in **bold**. **A1**: cross-task reuse of both positive and negative STL patterns (mined on BBH); **A2**: cross-task reuse of negative (failure-mode) patterns only; **A3**: in-domain STL mining with fixed parameters; **Ours**: in-domain STL with instance-adaptive parameters via a hypernetwork. Since A1 and A2 use the mined pattern from BBH, "–" indicates that the A1 and A2 results of BBH are the same with A3.

Base Model	Method	SciQ		CLadder		Math		BBH	
		ECE ↓	Brier ↓	ECE ↓	Brier ↓	ECE ↓	Brier ↓	ECE ↓	Brier ↓
Qwen3	AveLogit	$.055_{\pm.006}$	$.028_{\pm.004}$	$.200_{\pm.007}$	$.230_{\pm.006}$	$.150_{\pm.023}$	$.177_{\pm.020}$	$.339_{\pm.011}$	$.354_{\pm.009}$
	SAR	$.244_{\pm.039}$	$.272_{\pm.012}$	$.406_{\pm.010}$	$.421_{\pm.009}$	$.373_{\pm.085}$	$.382_{\pm.071}$	$.372_{\pm.011}$	$.382_{\pm.010}$
	Self-Eval	$.010_{\pm.005}$	$.040_{\pm.009}$	$.222_{\pm.005}$	$.250_{\pm.005}$	$.146_{\pm.016}$	$.165_{\pm.018}$	$.197_{\pm.013}$	$\mathbf{.217}_{\pm.012}$
	Self-Consistency	$.046_{\pm.009}$	$.045_{\pm.007}$	$.223_{\pm.016}$	$.239_{\pm.008}$	$.091_{\pm.021}$	$\mathbf{.143}_{\pm.020}$	$.248_{\pm.183}$	$.227_{\pm.195}$
	InternalInspector	$.368_{\pm.026}$	$.248_{\pm.057}$	$.236_{\pm.043}$	$.305_{\pm.077}$	$\mathbf{.064}_{\pm.016}$	$.168_{\pm.016}$	$.337_{\pm.039}$	$.518_{\pm.050}$
	A1	$.506_{\pm.005}$	$.283_{\pm.003}$	$.258_{\pm.007}$	$.254_{\pm.001}$	$.354_{\pm.018}$	$.271_{\pm.004}$	–	–
	A2	$.176_{\pm.006}$	$.057_{\pm.004}$	$.084_{\pm.004}$	$.183_{\pm.003}$	$.146_{\pm.015}$	$.168_{\pm.008}$	–	–
	A3	$.005_{\pm.007}$	$.025_{\pm.005}$	$.057_{\pm.006}$	$.178_{\pm.004}$	$.082_{\pm.009}$	$.158_{\pm.017}$	$.057_{\pm.009}$	$.228_{\pm.003}$
	Ours	$\mathbf{.005}_{\pm.006}$	$\mathbf{.025}_{\pm.005}$	$\mathbf{.035}_{\pm.003}$	$\mathbf{.172}_{\pm.004}$	$.109_{\pm.014}$	$.154_{\pm.014}$	$\mathbf{.052}_{\pm.015}$	$.224_{\pm.003}$
Gemma3	AveLogit	$.396_{\pm.030}$	$.404_{\pm.023}$	$.127_{\pm.012}$	$.181_{\pm.009}$	$.153_{\pm.028}$	$.171_{\pm.038}$	$.209_{\pm.006}$	$.249_{\pm.004}$
	SAR	$.246_{\pm.039}$	$.265_{\pm.032}$	$.388_{\pm.009}$	$.413_{\pm.009}$	$.381_{\pm.098}$	$.393_{\pm.089}$	$.379_{\pm.011}$	$.400_{\pm.005}$
	Self-Eval	$\mathbf{.044}_{\pm.020}$	$.059_{\pm.018}$	$.204_{\pm.011}$	$.222_{\pm.010}$	$.136_{\pm.031}$	$\mathbf{.135}_{\pm.043}$	$.220_{\pm.007}$	$.228_{\pm.006}$
	Self-Consistency	$.052_{\pm.011}$	$\mathbf{.049}_{\pm.011}$	$.096_{\pm.024}$	$.228_{\pm.008}$	$.148_{\pm.046}$	$.182_{\pm.036}$	$.186_{\pm.233}$	$.195_{\pm.232}$
	InternalInspector	$.831_{\pm.159}$	$.055_{\pm.064}$	$.172_{\pm.043}$	$.214_{\pm.015}$	$\mathbf{.060}_{\pm.040}$	$.167_{\pm.028}$	$.995_{\pm.005}$	$\mathbf{.001}_{\pm.001}$
	A1	$.142_{\pm.027}$	$.264_{\pm.008}$	$.064_{\pm.008}$	$.171_{\pm.004}$	$.203_{\pm.052}$	$.154_{\pm.018}$	–	–
	A2	$.059_{\pm.022}$	$.246_{\pm.001}$	$.038_{\pm.010}$	$.167_{\pm.006}$	$.157_{\pm.054}$	$.149_{\pm.024}$	–	–
	A3	$.056_{\pm.016}$	$.247_{\pm.001}$	$.042_{\pm.013}$	$.165_{\pm.006}$	$.122_{\pm.057}$	$.151_{\pm.026}$	$.054_{\pm.013}$	$.196_{\pm.003}$
	Ours	$.048_{\pm.012}$	$.245_{\pm.003}$	$\mathbf{.018}_{\pm.005}$	$\mathbf{.160}_{\pm.006}$	$.131_{\pm.046}$	$.141_{\pm.024}$	$\mathbf{.039}_{\pm.004}$	$.189_{\pm.004}$
Llama	AveLogit	$.546_{\pm.041}$	$.511_{\pm.031}$	$.402_{\pm.008}$	$.413_{\pm.006}$	$.550_{\pm.070}$	$.530_{\pm.052}$	$.510_{\pm.009}$	$.502_{\pm.006}$
	SAR	$.205_{\pm.016}$	$.258_{\pm.025}$	$.333_{\pm.001}$	$.365_{\pm.005}$	$.292_{\pm.071}$	$.323_{\pm.049}$	$.309_{\pm.012}$	$.330_{\pm.005}$
	Self-Eval	$.056_{\pm.021}$	$\mathbf{.066}_{\pm.023}$	$.381_{\pm.011}$	$.383_{\pm.011}$	$.593_{\pm.041}$	$.585_{\pm.042}$	$.454_{\pm.007}$	$.454_{\pm.007}$
	Self-Consistency	$\mathbf{.043}_{\pm.025}$	$.097_{\pm.018}$	$.179_{\pm.025}$	$.273_{\pm.010}$	$.469_{\pm.078}$	$.461_{\pm.059}$	$.242_{\pm.169}$	$.265_{\pm.154}$
	InternalInspector	$.641_{\pm.059}$	$.132_{\pm.039}$	$.026_{\pm.001}$	$.250_{\pm.000}$	$.315_{\pm.046}$	$.231_{\pm.025}$	$.887_{\pm.082}$	$\mathbf{.039}_{\pm.008}$
	A1	$.064_{\pm.045}$	$.216_{\pm.011}$	$.125_{\pm.009}$	$.266_{\pm.002}$	$\mathbf{.077}_{\pm.038}$	$.228_{\pm.020}$	–	–
	A2	$.059_{\pm.020}$	$.214_{\pm.016}$	$.012_{\pm.004}$	$.284_{\pm.002}$	$.106_{\pm.045}$	$.229_{\pm.025}$	–	–
	A3	$.058_{\pm.018}$	$.213_{\pm.016}$	$.010_{\pm.004}$	$.285_{\pm.002}$	$.104_{\pm.059}$	$.231_{\pm.027}$	$.014_{\pm.010}$	$.233_{\pm.002}$
	Ours	$.054_{\pm.020}$	$.211_{\pm.016}$	$\mathbf{.009}_{\pm.003}$	$\mathbf{.233}_{\pm.001}$	$.103_{\pm.040}$	$\mathbf{.225}_{\pm.023}$	$\mathbf{.012}_{\pm.007}$	$.232_{\pm.002}$

baselines are implemented per their original descriptions and evaluated under identical data splits and inference settings. We evaluate confidence estimation quality using standard calibration metrics, including Expected Calibration Error (ECE), Brier Score, and AUROC. All metrics are computed on held-out test sets and averaged across folds.

We include Self-Consistency (Wang et al., 2023) strictly as a calibration baseline (its consistency rate has been adopted as a confidence estimator in calibration studies (Xiong et al., 2024)), not as an accuracy baseline; our method is orthogonal to it and the two could in principle be combined. For segmentation, we use a regex cascade aligned with CoT step markers: a primary matcher for `Step` N: (yielding 8.7 segments on average and covering the majority of responses), with bullet (`\n-`, `\n*`) and newline (> 20 chars) fallbacks. A segmentation-strategy ablation is reported in Appendix D.3.

6.1 Results and Analysis

Table 1 summarizes the main experimental results, analyzing both cross-task transferability of STL patterns (RQ1) and the effect of instance-level parameter adaptation (RQ2).

Across backbone models and datasets, reusing only negative (failure-mode) STL patterns (A2) consistently outperforms full cross-task reuse (A1), confirming that incorrect-reasoning confidence dynamics transfer more reliably than correct-reasoning patterns. However, both cross-task variants are clearly inferior to in-domain STL mining (A3 and Ours), indicating that task-specific temporal structures and parameters remain crucial for accurate calibration. Comparing A3 with Ours isolates the impact of question-level parameter adaptation. While in-domain fixed STL blocks already yield strong performance, the proposed hypernetwork-based parameterization further improves calibration in most settings, with particularly consistent gains on CLadder and BBH. This demonstrates that, even under a shared temporal structure, confidence dynamics vary substantially across questions, supporting RQ2.

Standard logit-based, consistency-based, and self-evaluation baselines exhibit higher calibration

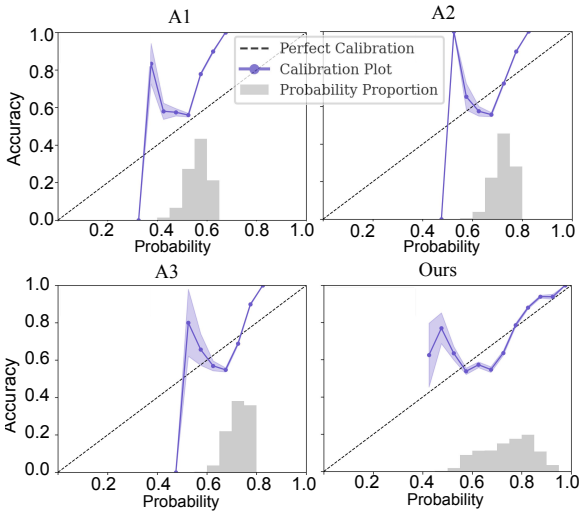


Figure 7: Reliability diagrams for three ablated variants (A1–A3) and our method on CLadder with Qwen3. The dashed diagonal denotes perfect calibration. Our method aligns more closely with the diagonal across confidence ranges, indicating improved calibration.

error across all benchmarks. In contrast, STL-based methods (A3 and Ours) achieve more reliable calibration by explicitly modeling temporal confidence evolution, with Ours providing the most stable performance under single-sample inference. A controlled comparison against generic sequence classifiers (MLP, Bi-LSTM) with and without STL-derived features is reported in Appendix D.4, showing that the calibration gains come from the STL representation rather than from a particular architecture. Table 7 shows that confidence estimation is sensitive to the number of STL templates. Too few templates underfit temporal patterns, while too many introduce redundancy and degrade calibration. An intermediate configuration (10 balanced templates) achieves the best overall trade-off across ECE, Brier score, and AUROC.

Cost vs. calibration trade-off. Table 9 reports inference time per example alongside average ECE across all base models and datasets from Table 1. Our method runs at $0.55\,\mathrm{s}$ per example, on the same order as the cheapest internals-based baseline (InternalInspector, $0.078\,\mathrm{s}$) and one to four orders of magnitude faster than self-evaluation, SAR, and self-consistency, while attaining an average ECE of 0.051 — an 83% relative reduction over AveLogit and a substantial improvement over every competing approach. The only faster method, InternalInspector, in fact *degrades* calibration relative to the logit baseline. The offline cost is also modest: our hypernetwork has 2.67M parameters ($\approx 0.03\%$ of the LLM) and trains on a single GPU, in contrast to InternalInspector, which requires 12–57M parameters and multi-GPU training.

7 Conclusion

We proposed a temporal confidence estimation framework that models stepwise LLM confidence as a signal evaluated with Signal Temporal Logic. The resulting confidence scores are both well calibrated and interpretable. Experiments show that temporal structures are task-dependent, while instance-level parameter adaptation via hypernetworks is crucial for capturing question-specific confidence dynamics. Overall, our results highlight the role of temporal structure in LLM confidence estimation and offer an efficient alternative to sampling-based approaches.

8 Limitations

Our approach relies on stepwise segmentation of reasoning responses; while Appendix D.3 shows that the method remains competitive under alternative segmentation strategies, step-based segmentation works best and may not be readily available for unstructured outputs. Effective STL structures are also partly task-dependent: failure-mode templates transfer across tasks, but the most accurate calibration still benefits from in-domain mining. Finally, our evaluation focuses on structured reasoning benchmarks with unambiguous correctness labels; extending the framework to open-ended generation and to dedicated hallucination detection (which would require span-level annotations) is a natural direction for future work.

Acknowledgements

This work was supported by the NSF Grant CNS 2513076. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF) or the US Government.

References

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, and 1 others. 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297.

Alfonso Amayuelas, Kyle Wong, Liangming Pan, Wenhu Chen, and William Yang Wang. 2024. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6416–6432.

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Daniel Beck, Lucia Specia, and Trevor Cohn. 2016. Exploring prediction uncertainty in machine translation quality estimation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 208–218, Berlin, Germany. Association for Computational Linguistics.

Mohammad Beigi, Ying Shen, Runing Yang, Zihao Lin, Qifan Wang, Ankith Mohan, Jianfeng He, Ming Jin, Chang-Tien Lu, and Lifu Huang. 2024. InternalInspector i^2: Robust confidence estimation in LLMs through internal states. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12847–12865, Miami, Florida, USA. Association for Computational Linguistics.

Glenn W. Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. *arXiv preprint arXiv:1805.04604*.

Alexandre Donzé and Oded Maler. 2010. Robust satisfaction of temporal logic over real-valued signals. In *International conference on formal modeling and analysis of timed systems*, pages 92–106. Springer.

Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. 2024. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5050–5063, Bangkok, Thailand. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.

Artur d'Avila Garcez, Marco Gori, Luis C Lamb, Luciano Serafini, Michael Spranger, and Son N Tran. 2019. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *arXiv preprint arXiv:1905.06088*.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. 2018. Conditional neural processes. In *International conference on machine learning*, pages 1704–1713. PMLR.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Bairu Hou, Yujian Liu, Kaizhi Qian, Jacob Andreas, Shiyu Chang, and Yang Zhang. 2024. Decomposing uncertainty for large language models through input clarification ensembling. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Susmit Jha, Ashish Tiwari, Sanjit A Seshia, Tuhin Sahai, and Natarajan Shankar. 2019. Telex: learning signal temporal logic from positive examples using tightness metric. *Formal Methods in System Design*, 54(3):364–387.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Adharsh Kamath, Sishen Zhang, Calvin Xu, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. 2025. Enforcing temporal constraints for llm agents. *arXiv preprint arXiv:2512.23738*.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30.

Michael Kirchhof, Luca Füger, Adam Golinski, Eeshan Gunesh Dhekane, Arno Blaas, and Sinead Williamson. 2025. Self-reflective uncertainties: Do llms know their internal answer distribution? In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*.

Zhaodan Kong, Austin Jones, Ana Medina Ayala, Ebru Aydin Gol, and Calin Belta. 2014. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 273–282.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Jiawei Li, Akshayaa Magesh, and Venugopal V Veeravalli. 2025a. Principled detection of hallucinations in large language models via multiple testing. *arXiv preprint arXiv:2508.18473*.

Yinghao Li, Rushi Qiang, Lama Moukheiber, and Chao Zhang. 2025b. Language model uncertainty quantification with attention chain. In *Second Conference on Language Modeling*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*.

Alexis Linard and Jana Tumova. 2020. Active learning of signal temporal logic specifications. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 779–785. IEEE.

Xiaoou Liu, Tiejin Chen, Longchao Da, Chacha Chen, Zhen Lin, and Hua Wei. 2025. Uncertainty quantification and confidence calibration in large language models: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6107–6117.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.

Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *International symposium on formal techniques in real-time and fault-tolerant systems*, pages 152–166. Springer.

Andrey Malinin and Mark Gales. 2021. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*.

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.

Zhenjiang Mao, Artem Bisliouk, Rohith Nama, and Ivan Ruchkin. 2025a. Temporalizing confidence: Evaluation of chain-of-thought reasoning with signal temporal logic. In *Proceedings of the 20th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2025)*, pages 882–890, Vienna, Austria. Association for Computational Linguistics.

Zhenjiang Mao, Carson Sobolewski, and Ivan Ruchkin. 2024. How safe am i given what i see? calibrated prediction of safety chances for image-controlled autonomy. In *6th Annual Learning for Dynamics & Control Conference*, pages 1370–1387. PMLR.

Zhenjiang Mao, Mrinall Eashaan Umasudhan, and Ivan Ruchkin. 2025b. How safe will i be given what i saw? calibrated prediction of safety chances for image-controlled autonomy. *arXiv preprint arXiv:2508.09346*.

Zhenjiang Mao and Anirudhh Venkat. 2026. Recurrent confidence chain: Temporal-aware uncertainty quantification in large language models. *arXiv preprint arXiv:2601.13368*.

Daniele Nicoletti, Samuele Germiniani, and Graziano Pravadelli. 2024. Mining signal temporal logic specifications for hybrid systems. In *2024 Forum on Specification & Design Languages (FDL)*, pages 1–8. IEEE.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*, pages 3956–3965. PMLR.

Amir Pnueli. 1977. The temporal logic of programs. In *18th annual symposium on foundations of computer science (sfcs 1977)*, pages 46–57. ieee.

Gwenyth Portillo Wightman, Alexandra Delucia, and Mark Dredze. 2023. Strength in numbers: Estimating confidence of large language models by prompt agreement. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 326–362, Toronto, Canada. Association for Computational Linguistics.

Vasumathi Raman, Mehdi Maasoumy, and Alexandre Donzé. 2014. Model predictive control from signal temporal logic specifications: A case study. In *Proceedings of the 4th ACM SIGBED international workshop on design, modeling, and evaluation of cyber-physical systems*, pages 52–55.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*.

Carson Sobolewski, Zhenjiang Mao, Kshitij Maruti Vejre, and Ivan Ruchkin. 2025. Generalizable image repair for robust visual control. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9971–9978. IEEE.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *Preprint*, arXiv:2305.14975.

Ke Wang, Yangbin Shi, Jiayi Wang, Yuqi Zhang, Yu Zhao, and Xiaolin Zheng. 2021. Beyond glass-box features: Uncertainty quantification enhanced quality estimation for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4687–4698, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Jingjin Yu, Ming C. Lin, Karen Leung, Nikos Aréchiga, and Marco Pavone. 2023. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *Int. J. Rob. Res.*, 42(6):356–370.

A Relationship Between Confidence and Uncertainty

In the literature, the terms *uncertainty* and *confidence* are often used interchangeably, but they refer to distinct concepts. Uncertainty (aleatoric or epistemic) typically characterizes ambiguity or lack of knowledge (Kendall and Gal, 2017; Abdar et al., 2021) in the data distribution or the model. In contrast, confidence refers to a sample-level estimate of the reliability or correctness of a specific generated response. In this work, we focus on confidence estimation rather than explicit modeling of aleatoric or epistemic uncertainty. Uncertainty in deep learning, and specifically in Large Language Models (LLMs), is typically categorized into two primary types: *Aleatoric* and *Epistemic*. Often referred to as data uncertainty, *aleatoric uncertainty* arises from the inherent stochasticity within the data distribution itself. In the context of generative models, this typically manifests when an input prompt $\mathbf{x}$ plausibly maps to multiple valid responses $\mathbf{y}$. Mathematically, this corresponds to a dispersed or multi-modal posterior distribution $p(\mathbf{y} \mid \mathbf{x})$. Crucially, aleatoric uncertainty is considered *irreducible*: even with infinite training data and a perfect model, this uncertainty persists as it is an intrinsic property of the underlying task. In this work, we focus on confidence estimation for individual model outputs, rather than modeling inherent data ambiguity.

Known as model uncertainty, *epistemic uncertainty* stems from the model's lack of knowledge or ignorance regarding the underlying data-generating process. In the context of LLMs, this typically arises when the input prompt falls into "knowledge blind spots" or *Out-of-Distribution* (OOD) regions where the training data coverage is sparse or non-existent (Malinin and Gales, 2021; Desai and Durrett, 2020). A paradigmatic example is a query regarding future events, such as *"Where will the 2060 Olympics be held?"*. Since this information is absent from the training corpus, the model is forced to guess, exhibiting high epistemic uncertainty. Mathematically, this corresponds to uncertainty over the model parameters $\boldsymbol{\theta}$. Epistemic uncertainty arises from the model's lack of knowledge about the data-generating process, for instance, when inputs fall into out-of-distribution regions or knowledge blind spots. While epistemic uncertainty characterizes the model's lack of knowledge at a distributional level, our work does not aim to explicitly estimate or decompose epistemic uncertainty. Instead, we focus on confidence estimation at the instance level, i.e., predicting the probability that a specific generated response is correct. Epistemic uncertainty may influence confidence implicitly, but it is not modeled as a separate uncertainty component.

A.1 Evaluation Metrics

To rigorously quantify the quality of the estimated confidence P, we compare it against the empirical correctness using standard calibration metrics. We primarily utilize the following three metrics:

Expected Calibration Error (ECE): ECE (Guo et al., 2017) measures the expected difference between the predicted confidence and the empirical accuracy. It is computed by partitioning samples into B bins based on their confidence scores:

$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{|B_b|}{N} \left|\mathrm{acc}(B_b) - \mathrm{conf}(B_b)\right|, \quad (2)$$

where N is the total number of samples, $|B_b|$ is the number of samples in bin b, and $\mathrm{acc}(B_b)$ and $\mathrm{conf}(B_b)$ represent the average accuracy and average confidence within that bin, respectively.

Brier Score (BS): The Brier Score (Brier, 1950) is a strictly proper scoring rule that measures the mean squared error between the predicted probability P_i and the binary ground truth outcome $c_i \in \{0, 1\}$ (where 1 indicates a correct response):

$$\mathrm{BS} = \frac{1}{N} \sum_{i=1}^{N} (P_i - c_i)^2. \quad (3)$$

B Signal Temporal Logic Preliminaries

B.1 STL Syntax and Semantics

Signal Temporal Logic (STL) provides a rigorous formalism for specifying and monitoring the temporal behavior of real-valued signals (Maler and Nickovic, 2004). We use STL to describe logical constraints on the evolution of the stepwise confidence signal $\mathbf{S} = [s_1, \ldots, s_n]$.

An STL formula is defined by a fixed logical structure φ together with a set of continuous parameters $\boldsymbol{\theta}$, such as predicate thresholds and temporal bounds. We denote a parameterized STL formula as $\varphi_{\boldsymbol{\theta}}$. The basic building block is a *predicate* μ, typically taking the form of an inequality $\mu \equiv s_t \geq c$, where c is a constant threshold. The inequality direction is chosen such that larger (better) values of s_t correspond to higher confidence.

Complex formulas are constructed using Boolean operators (negation $\neg$, conjunction $\wedge$) and temporal operators (Always $\Box$, Eventually $\Diamond$). Our chosen fragment of the STL syntax is given by:

$$\varphi ::= \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \Box_{[a,b]}\varphi \mid \Diamond_{[a,b]}\varphi,$$

where $[a, b]$ denotes a time interval with parametric start time a and end b. Intuitively, $\Box_{[a,b]}\varphi$ requires φ to hold at *all* steps within the interval, while $\Diamond_{[a,b]}\varphi$ requires φ to hold at *some* step within the interval. We focus on the $\Box$ and $\Diamond$ operators in this work, as they are sufficient to capture the temporal confidence patterns of interest, and omit the Until operator for simplicity.

STL Robustness. Unlike standard Boolean logic, which yields binary outcomes (True/False), STL offers *quantitative semantics*, also known as robustness degree ρ. The robustness $\rho(\varphi, \mathbf{S}, t)$ measures *how strongly* the signal $\mathbf{S}$ satisfies the formula φ at time t. A positive robustness indicates satisfaction, while a negative value implies violation. For a predicate $\mu \equiv s_t \geq c$, the robustness is simply $\rho(\mu, \mathbf{S}, t) = s_t - c$. Crucially, the robustness for temporal operators is computed using min-max operations:

$$\rho(\Box_{[a,b]}\varphi, \mathbf{S}, t) = \min_{k \in [t+a, t+b]} \rho(\varphi, \mathbf{S}, k), \quad (4)$$

$$\rho(\Diamond_{[a,b]}\varphi, \mathbf{S}, t) = \max_{k \in [t+a, t+b]} \rho(\varphi, \mathbf{S}, k). \quad (5)$$

This quantitative property is particularly valuable for LLMs, as it transforms logical constraints into differentiable scalar values that can serve as uncertainty metrics or training objectives. As illustrated in Figure 1, STL robustness offers a quantitative way to summarize whether a stepwise confidence trajectory matches a specified temporal pattern.

C Primitive STL Templates and Atomic Predicates

This section summarizes the complete set of primitive STL templates $\mathcal{T}_{\text{base}}$ and atomic predicates $\mathcal{P}$ used in the discriminative STL mining procedure described in Section 4. These templates define the base hypothesis space from which more complex temporal formulas are constructed through Boolean composition, temporal nesting, and parameterization.

We emphasize that these primitives are *not* used as fixed rules. Instead, they serve as a compact, interpretable basis for defining a large search space of candidate temporal confidence patterns, from which optimal formulas are selected via data-driven discriminative mining.

C.1 Primitive STL Templates

We group the primitive templates into two categories: *positive templates* φ^+, which are intended to capture temporal confidence patterns characteristic of correct reasoning, and *negative templates* φ^-, which describe failure modes such as instability, confidence collapse, or oscillation.

Positive Templates (φ^+). These templates describe temporal confidence behaviors commonly observed in correct or well-structured reasoning trajectories.

Template	STL Formula
WeakestLink	$\Box_{[0,T]}(s_t \geq \mu)$
EndHigh	$\Box_{[T-k,T]}(s_t \geq \mu)$
StartHigh	$\Box_{[0,k]}(s_t \geq \mu)$
NeverSharpDrop	$\Box_{[0,T]}(\Delta_t > -\varepsilon)$
ConfidenceGain	$\bar{s}_{\text{end}} > \bar{s}_{\text{start}} + \varepsilon$
LowVarOverall	$\text{Var}(s) \leq \nu$

Table 2: Primitive positive STL templates used as base structures for discriminative mining.

Negative Templates (φ^-). Negative templates are designed to capture temporal confidence patterns associated with incorrect reasoning, hallucination, or unstable inference.

Template	STL Formula
EventuallyLow	$\Diamond_{[0,T]}(s_t \leq \mu)$
EndLow	$\Diamond_{[T-k,T]}(s_t \leq \mu)$
ConfidenceLoss	$\bar{s}_{\text{start}} > \bar{s}_{\text{end}} + \varepsilon$
SharpDrop	$\Diamond_{[0,T]}(\Delta_t \leq -\varepsilon)$
FinalDecline	$\Box_{[T-k,T]}(\Delta_t \leq 0)$
Recovery	$\Diamond(s_t \leq \mu_l) \wedge \Diamond(s_t \geq \mu_h)$

Table 3: Primitive negative STL templates capturing failure-related confidence patterns.

For each template, we optimize its parameters θ (in the above examples, $\theta = (a, b, \mu)$) by minimizing a classification-based objective that separates correct and incorrect responses. Specifically, we compute the robustness score $\rho(\varphi, \mathbf{S})$ for each response and minimize negative log-likelihood (NLL) loss with respect to the correctness labels. This objective is identical in form to the loss used later for confidence estimation, but is applied here solely for evaluating the discriminative utility of candidate STL templates. This step yields a set

of *parameterized base templates* $\{(\varphi_k, \theta_k)\}$, each instantiated from a predefined template and fitted to the data to maximize its discriminative power.

Robustness-Based Signal Augmentation. To enable the construction of nested STL formulas without explicitly enumerating complex parse trees, we adopt a robustness-based signal augmentation strategy, commonly referred to as *signal lifting* in the STL mining literature (Jha et al., 2019; Donzé and Maler, 2010; Kong et al., 2014). For each parameterized base template (φ_k, θ_k) discovered in Step 1, we compute the robustness signal $r_j \in \mathbb{R}^n$ defined by $r_j[t] = \rho(\varphi_j, \mathbf{S}, t)$ for all time steps t.

We then construct an augmented multi-dimensional signal $\mathbf{S}' \in \mathbb{R}^{n\times(d+k)}$ by concatenating these robustness signals along the feature dimension:

$$\mathbf{S}' = [\mathbf{S} \mid r_1 \mid r_2 \mid \ldots \mid r_k]. \tag{6}$$

Based on this augmented signal, we define new atomic predicates indicating the satisfaction of the inner formula: $\mu_j^+ := (r_j \geq 0)$ (formula satisfied) and $\mu_j^- := (r_j < 0)$ (formula violated).

Structural Composition. Using the augmented predicates μ_j^+ and μ_j^-, we generate complex structures via two mechanisms:

- **Temporal Nesting:** We apply temporal templates to the new predicates (e.g., $\Diamond_{[a,b]}\mu_j^+$), which is equivalent to applying the same temporal operator to the original formula ϕ_j, i.e., $\Diamond_{[a,b]}\phi_j$, under the robustness semantics.
- **Boolean Combination:** To cover diverse error patterns, we combine formulas using logic operators. Negation is implicitly handled via the complementary predicates μ_j^+ and μ_j^-, corresponding to satisfaction and violation of the underlying formula, respectively. Specifically, if a set of formulas $\{\phi_k\}$ effectively captures distinct subsets of errors, we combine them via disjunction $\bigvee \phi_k$. Conversely, to tighten the criteria for correctness, we employ conjunction $\bigwedge \phi_k$.

Dual-Class Template Mining. Using the candidate STL formulas generated through Steps 1–3, we explicitly perform dual-class discriminative mining to identify two complementary categories of temporal patterns. Here, $z_i(\phi)$ is obtained by applying a simple monotonic transformation to the scalar robustness $\rho(\phi, \mathbf{S}_i)$, so that larger robustness yields a larger logit. Building on the discriminative objective used for parameterizing base templates in Step 1, we explicitly search for two complementary categories of patterns. We aim to find formulas that not only recognize their target class but also suppress the non-target class:

Correct Patterns (Φ_{pos}): These formulas characterize correct reasoning flows (e.g., "confidence consistently increases"). They are optimized to assign high confidence to correct responses and low confidence to the incorrect ones. In this setting, $z_i(\phi)$ denotes a discriminative logit for the target class of ϕ: larger values indicate stronger evidence that instance i belongs to the target class.

Using the empirical correctness label $c_i \in \{0, 1\}$, we minimize the standard Negative Log-Likelihood (NLL):

$$\mathcal{L}_{\text{pos}} = -\frac{1}{N}\sum_{i=1}^{N}\left[c_i \log \sigma(z_i(\phi)) + (1-c_i)\log\left(1-\sigma(z_i(\phi))\right)\right], \tag{7}$$

Incorrect Patterns (Φ_{neg}): These formulas capture hallucinations or structural errors (e.g., "confidence oscillates"). For such patterns, a high robustness score should indicate a high probability of incorrectness. For Φ_{neg}, the target class is incorrectness; equivalently, we flip the label and reuse the same logistic mapping on $z_i(\phi)$. Accordingly, we invert the target label and optimize:

$$\mathcal{L}_{\text{neg}} = -\frac{1}{N}\sum_{i=1}^{N}\left[(1-c_i)\log \sigma(z_i(\phi)) + c_i \log\left(1-\sigma(z_i(\phi))\right)\right], \tag{8}$$

We hypothesize that correct and incorrect reasoning traces exhibit distinct temporal confidence signatures. We capture such signatures using STL formulas evaluated under quantitative robustness semantics, using both a scalar robustness score $\rho(\varphi, \mathbf{S}_i)$ for discriminative scoring and a time-resolved robustness signal $\rho(\varphi, \mathbf{S}_i, t)$ for signal lifting in Step 2. Following prior STL mining work (Jha et al., 2019), we adapt these techniques to the LLM confidence setting and learn discriminative STL patterns that separate correct from incorrect responses.

Formally, given a dataset $\{(\mathbf{S}_i, c_i)\}_{i=1}^N$, we mine STL formulas ϕ whose robustness separates the two classes. For a candidate formula ϕ, we compute a robustness-derived logit $z_i(\phi)$ and optimize a NLL objective on $\sigma(z_i(\phi))$ to quantify its discriminative utility. This can be expressed as the following optimization problem:

$$(\varphi^*, \theta^*) = \arg\min_{\varphi,\theta} \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\big(c_i, \hat{P}_i\big), \tag{9}$$

where $\hat{P}_i$ denotes the predicted confidence for instance i and $\mathcal{L}$ corresponds to the discriminative loss defined in Eq. (7)–(8).

This formulation allows the temporal logic structure φ and its parameters θ to be learned under a calibration-oriented objective, while treating STL robustness as a structured confidence signal rather than a probability itself.

Having performed STL mining, we study the generalizability of the mined patterns from two complementary analytical perspectives. Figure 3 provides a schematic overview of the two research questions, illustrating the distinction between structural generalization across task types (RQ1) and parameter sensitivity under a fixed formula (RQ2).

C.2 Atomic Predicates

The primitive templates are instantiated over a shared set of atomic predicates $\mathcal{P}$, defined over the stepwise confidence signal $\mathbf{S} = [s_1, \ldots, s_n]$ and its first-order difference $\Delta_t = s_t - s_{t-1}$.

Category	Predicate
Level	$s_t \geq \mu, \quad s_t \leq \mu$
Deviation	$s_t \geq \bar{s} - k\sigma$
Derivative	$\Delta_t > -\varepsilon, \quad \Delta_t \leq -\varepsilon, \quad \Delta_t \leq 0$
Aggregate	$\mathrm{Var}(s) \leq \nu, \quad \bar{s}_{\text{end}} > \bar{s}_{\text{start}}$

Table 4: Atomic predicates used to instantiate STL templates.

C.3 Compositional Search Space

Although the base template set $\mathcal{T}_{\text{base}}$ contains only 14 primitive templates, it induces a large hypothesis space through systematic composition: (i) Boolean combination (conjunction and disjunction), (ii) temporal nesting of operators, and (iii) parameterized time windows and thresholds.

For example, pairwise Boolean composition alone yields $\binom{14}{2} = 91$ conjunctions and 91 disjunctions, and higher-order compositions further expand the space. When combined with multiple temporal windows and continuous parameter ranges, the effective search space quickly reaches hundreds or thousands of candidate formulas.

This combinatorial growth motivates the use of discriminative STL mining to automatically identify a small set of informative temporal confidence patterns from data.

D Implementation Details

D.1 STL Blocks and Robustness-to-Confidence Mapping

This appendix provides implementation-level details for the STL blocks introduced in Section 5.1. These details are omitted from the main text for clarity and reproducibility.

Robustness computation. Each STL block evaluates a fixed STL formula φ on a stepwise confidence signal $\mathbf{S} = [s_1, \ldots, s_T]$ and produces both an aggregated robustness value $\rho(\varphi, \mathbf{S})$ and, optionally, a per-step robustness trace $\rho(\varphi, \mathbf{S}, t)$. Temporal operators are implemented using standard soft approximations of STL robust semantics, such as soft-min and soft-max, to ensure differentiability.

Numerical stabilization. In practice, robustness values may attain large magnitudes, which can lead to numerical instability when passed through nonlinear mappings. To mitigate this, aggregated robustness values are clipped to a bounded range prior to further processing. After probability mapping, outputs are additionally clipped to avoid saturation at 0 or 1, ensuring stable optimization under binary cross-entropy loss.

Parameterized sigmoid mapping. The robustness-to-confidence transform is implemented as a learnable sigmoid:

$$\hat{p} = \omega(\alpha\, \tilde{\rho} + \beta), \tag{10}$$

where $\tilde{\rho}$ denotes the clipped robustness value. The scale parameter α controls the sharpness of the decision boundary, while the bias parameter β controls its location. Both parameters are optimized jointly with the rest of the model.

Interpretation of mapping parameters. Smaller values of α yield smoother transitions between low and high confidence, making the estimator less sensitive to small robustness variations. Larger values of α produce sharper transitions, approaching a hard threshold on robustness. The bias parameter β determines the robustness level at which the predicted confidence crosses 0.5.

Aggregation over multiple STL formulas. When multiple STL formulas are active, each produces an individual confidence score. Scores from formulas associated with correct reasoning patterns (Φ_{pos}) are treated as positive evidence, while scores

from incorrect reasoning patterns (Φ_{neg}) are inverted to represent negative evidence. The final confidence estimate is obtained by a weighted aggregation of these components, as described in Section 5.1.

D.2 Hypernetwork Architecture

The hypernetwork $\mathcal{H}_\psi$ used in Section 5.2 maps the input prompt $\mathbf{x}$ and stepwise confidence signal $\mathbf{S}$ to a per-instance parameter vector θ that instantiates the STL blocks. We describe the sequence-encoder component referenced in Section 5.2 in detail here.

The encoder consists of two parallel branches that are fused before being passed to the parameter head:

Transformer branch. A 3-layer Pre-LayerNorm Transformer encoder with model dimension $d = 256$ and 4 attention heads processes a token-level confidence projection together with learned positional embeddings. A learnable `[CLS]` token is prepended, and its final representation is taken as a 256-dimensional summary of the trajectory.

Statistical-feature branch. In parallel, we compute a fixed vector of 16 hand-crafted temporal statistics of the stepwise signal $\mathbf{S}$, including its mean, variance, minimum, maximum, slope, monotonicity score, and several local-difference statistics. These global descriptors give the encoder direct access to coarse trajectory shape independent of the Transformer attention pattern.

Fusion and parameter head. The Transformer `[CLS]` embedding and the statistical-feature vector are concatenated and passed through a 2-layer MLP with LayerNorm and GELU activations, producing a 256-dimensional encoding. This encoding, together with the encoded prompt $\mathbf{x}$, is then passed to a small parameter head that emits the per-instance STL parameters θ (temporal bounds, predicate thresholds, and the robustness-to-confidence mapping coefficients α, β).

The full sequence encoder contains approximately 1.2M trainable parameters, and the entire hypernetwork (including the parameter head) has roughly 2.67M parameters — about 0.03% of the underlying 8B-parameter base LLMs. This architecture is intentionally lightweight: it allows instance-adaptive parameterization while contributing only $\sim$0.55 seconds of total inference overhead per example.

Table 5: Effect of segmentation strategy on calibration (MATH, Qwen3-8B). Step-based segmentation achieves the best ECE, but all alternative strategies remain competitive with logit-based baselines.

Segmentation Strategy	ECE ↓
Step-based (default)	**0.085**
Sentence-level	0.139
Fixed-window (20 tokens)	0.151
Fixed-window (50 tokens)	0.109

D.3 Segmentation Robustness

To complement the regex-based segmentation specified in Section 6, we examine how the choice of segmentation strategy affects calibration. Resegmenting reasoning traces requires reprocessing the raw response tokens, which is expensive at the scale of our full benchmark suite, so we conduct this ablation on a smaller-scale slice (MATH with Qwen3-8B). We compare four strategies: the original step-based regex cascade, sentence-level splitting (via a standard sentence tokenizer), and two fixed-window strategies that partition the token sequence into equal-length windows of 20 and 50 tokens, respectively.

Table 5 reports ECE for each variant. The semantically aligned step-based segmentation yields the best ECE (0.085), as expected: STL templates were designed to capture transitions between coherent reasoning steps, and mechanical splits dilute that structure. At the same time, even the worst alternative (Fixed-window-20, ECE $= 0.151$) remains competitive with most baselines reported in our main results (Table 1). This suggests that step-based segmentation is preferable when available, but the temporal-confidence framework as a whole is not critically dependent on a specific segmentation scheme, which we view as an encouraging robustness property for future deployments where step markers are not guaranteed.

D.4 Are STL Formulas Necessary, or Do We Just Need a Sequence Model?

A natural question is whether a generic sequence-to-scalar classifier trained directly on the same stepwise confidence trace could match our results without the STL machinery. To probe this, we trained two standard neural classifiers on the raw stepwise signal under our exact data splits and evaluation protocol: a 3-layer MLP (128 hidden, ReLU, dropout 0.3) operating on the flattened signal padded to 50 steps plus 16 summary statis-

Table 6: Effect of STL-derived features on simple sequence classifiers (CLadder, Qwen3-8B). Without STL features, generic sequence models do not improve over the AveLogit baseline; with STL features, even an MLP becomes competitive with our full hypernetwork.

Method	ECE ↓
AveLogit	.201
MLP (raw signal)	.331
Bi-LSTM (raw signal)	.182
MLP + STL features	.041
Bi-LSTM + STL features	.037
Ours (STL + hypernetwork)	**.035**

tics, and a 2-layer Bi-LSTM (64 hidden) followed by a linear head. We then trained the same two architectures on the *robustness signals* produced by our mined STL templates instead of the raw confidence trace. On CLadder with Qwen3-8B, the comparison (Table 6) is informative: a Bi-LSTM trained on the raw signal does no better than the AveLogit baseline, while injecting STL-derived features alone is enough to drop ECE to 0.041 for the MLP and 0.037 for the Bi-LSTM. Our full instance-adaptive method achieves 0.035, only marginally better in ECE than these classifiers-with-STL-features, but it retains an advantage that black-box classifiers cannot offer: the temporal structure of the prediction is exposed as an auditable STL formula, so the model's failure modes are inspectable rather than implicit in the weights. This experiment confirms that the calibration gain comes from *the STL representation itself*, not from a particular classifier architecture.

E Additional Results

E.1 Analysis of Question Similarity and Parameter Variability

To investigate whether parameter variability can be explained by the intrinsic similarity of questions within a dataset, we analyze the intra-dataset question similarity using both TF-IDF and sentence-transformer-based embeddings. As reported in Table 13, datasets with higher question similarity do not necessarily exhibit lower parameter variance, and vice versa. For example, BBH exhibits relatively high semantic similarity between questions, yet shows substantial parameter variability, whereas CLadder displays lower question similarity but more stable parameter configurations.

Table 7: Hyperparameter study on the number of STL templates. All configurations use balanced positive and negative templates.

STL Configuration	ECE↓	Brier↓	AUROC↑
2 templates (1+1, total=2)	$0.024_{\pm.009}$	$0.235_{\pm.001}$	$0.610_{\pm.007}$
10 templates (5+5, total=10)	$\mathbf{0.020}_{\pm.005}$	$\mathbf{0.234}_{\pm.001}$	$\mathbf{0.617}_{\pm.006}$
16 templates (8+8, total=16)	$0.023_{\pm.007}$	$0.234_{\pm.001}$	$0.614_{\pm.008}$

E.2 Representative Mined STL Patterns

This appendix presents representative Signal Temporal Logic (STL) formulas *discovered by the mining procedure* described in Section 4. Rather than listing the full search space, we report the most frequently occurring and consistently generalizable patterns observed across datasets and backbone models.

These formulas correspond to empirical confidence dynamics that repeatedly emerge during mining, particularly those associated with strong cross-task structural reuse (RQ1). They are reported here to complement the quantitative analyses in the main text and to improve interpretability.

Dataset statistics. We evaluate on four public reasoning benchmarks: Big-Bench Hard (BBH), CLadder, SciQ, and GAOKAO-Math. BBH contains 23 challenging reasoning tasks (6511 samples). SciQ contains 13,679 multiple-choice science questions. CLadder contains over 10,000 samples. For GAOKAO-Math, we use the subset released by the benchmark source, containing 213 examples in our experiments. All experiments are conducted under the same evaluation protocol, and results are reported as mean $\pm$ standard deviation over 5 folds.

Acknowledgement: We used AI-assisted tools for language polishing and editing of the manuscript. All technical content, experimental design, analysis, and conclusions were developed and verified by the authors.

Potential Risks: Although this work aims to improve confidence calibration for long-form reasoning, miscalibrated confidence estimates may still occur, especially under distribution shift, noisy step segmentation, or open-ended tasks. In deployment, over-trusting such confidence scores could lead to inappropriate abstention or acceptance decisions in high-stakes settings. Therefore, the proposed method should be used as an assistive reliability signal rather than a standalone guarantee of correctness.

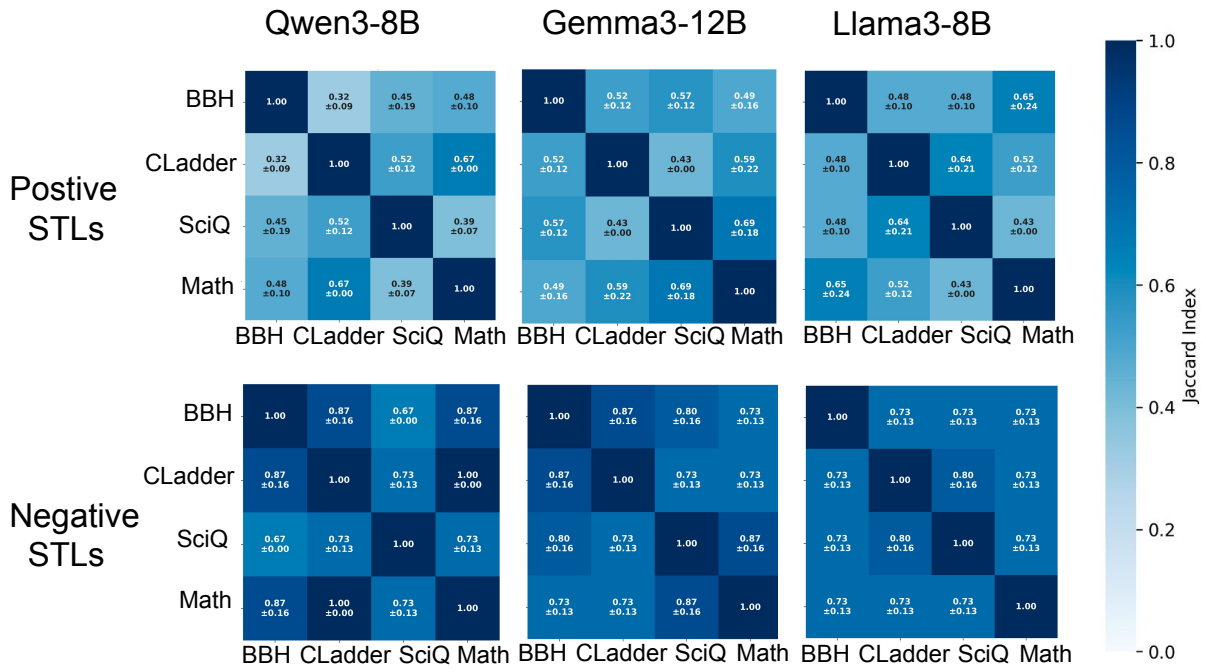


Figure 8: RQ1: Pairwise similarity of mined STL formulas across four reasoning benchmarks for different backbone models. Results are shown separately for correct (top) and incorrect (bottom) STL patterns, using Qwen3-8B, Gemma3-12B, and Llama3-8B. Across all models, incorrect STL patterns consistently exhibit higher cross-task similarity than correct patterns, confirming that the asymmetric generalization behavior observed in the main text is not model-specific.

Table 8: RQ1: Average Jaccard similarity of mined STL templates across task types. Results are reported as mean $\pm$ standard deviation over folds, separately for correct and incorrect STL templates and for different backbone models. Incorrect templates consistently exhibit higher cross-task similarity than correct templates, indicating stronger structural reuse.

Model	**Correct Templates**	**Incorrect Templates**
Qwen	0.473 ± 0.040	0.811 ± 0.045
Gemma3	0.549 ± 0.050	0.789 ± 0.108
Llama	0.532 ± 0.057	0.744 ± 0.027

Table 9: Inference time and average calibration error across baselines, averaged over all base models and datasets used in Table 1. Our method achieves the lowest ECE while remaining within the same order of magnitude as the fastest baseline.

Method	**Inference Time (s)**	**Avg ECE** $\downarrow$
InternalInspector	0.078	.411
Self-Consistency	> 10	.169
Self-Eval	4.73	.222
SAR	2.64	.327
Ours	0.55	**.051**

Table 10: Representative STL formulas discovered by discriminative mining. Positive patterns characterize confidence dynamics associated with correct reasoning, while negative patterns capture common failure modes. Here, s_t denotes the stepwise confidence at reasoning step t, $\Delta_t = s_t - s_{t-1}$ denotes the confidence change between steps, T is the total number of reasoning steps, and μ, ε are learned thresholds.

Category	**Pattern Name**	**Discovered STL Formula**
Positive	WeakestLink	$\square_{[0,T]}(s_t \geq \mu)$
	EndHigh	$\square_{[T-k,T]}(s_t \geq \mu)$
	StartHigh	$\square_{[0,k]}(s_t \geq \mu)$
	NeverSharpDrop	$\square_{[0,T]}(\Delta_t > -\varepsilon)$
Negative	EventuallyLow	$\diamond_{[0,T]}(s_t \leq \mu)$
	EndLow	$\diamond_{[T-k,T]}(s_t \leq \mu)$
	SharpDrop	$\diamond_{[0,T]}(\Delta_t \leq -\varepsilon)$
	Recovery	$\diamond(\text{low}) \wedge \diamond(\text{high})$

Table 11: Ablation study on uncertainty quantification: AUROC (↑) with mean ± std over 5 folds. Higher is better.

Base Model	**Variant**	**SciQ**	**CLadder**	**Math**	**BBH**
Qwen3	AveLogit	$.505_{\pm.068}$	$.718_{\pm.005}$	$.800_{\pm.024}$	$.536_{\pm.006}$
	Self-Eval	$.525_{\pm.072}$	$.534_{\pm.016}$	$.540_{\pm.045}$	$.540_{\pm.017}$
	Self-Consistency	$.789_{\pm.042}$	$.726_{\pm.023}$	$.765_{\pm.047}$	$.595_{\pm.210}$
	InternalInspector	$.514_{\pm.058}$	$.576_{\pm.085}$	$.503_{\pm.009}$	$.548_{\pm.002}$
	A1	$.502_{\pm.150}$	$.685_{\pm.004}$	$.737_{\pm.035}$	–
	A2	$.509_{\pm.171}$	$.713_{\pm.003}$	$.731_{\pm.035}$	–
	A3	$.501_{\pm.139}$	$.719_{\pm.013}$	$.749_{\pm.033}$	$.643_{\pm.003}$
	Ours	$.521_{\pm.150}$	$.735_{\pm.007}$	$.777_{\pm.030}$	$.650_{\pm.001}$
Gemma3	AveLogit	$.551_{\pm.024}$	$.644_{\pm.014}$	$.821_{\pm.066}$	$.548_{\pm.014}$
	Self-Eval	$.601_{\pm.111}$	$.599_{\pm.022}$	$.854_{\pm.073}$	$.629_{\pm.020}$
	Self-Consistency	$.586_{\pm.065}$	$.597_{\pm.018}$	$.797_{\pm.138}$	$.675_{\pm.215}$
	InternalInspector	$.768_{\pm.403}$	$.527_{\pm.034}$	$.501_{\pm.041}$	$.663_{\pm.337}$
	A1	$.583_{\pm.029}$	$.612_{\pm.012}$	$.832_{\pm.075}$	–
	A2	$.584_{\pm.026}$	$.626_{\pm.015}$	$.825_{\pm.077}$	–
	A3	$.580_{\pm.028}$	$.646_{\pm.019}$	$.832_{\pm.078}$	$.657_{\pm.026}$
	Ours	$.581_{\pm.032}$	$.667_{\pm.013}$	$.836_{\pm.077}$	$.661_{\pm.025}$
Llama	AveLogit	$.529_{\pm.045}$	$.507_{\pm.014}$	$.511_{\pm.090}$	$.567_{\pm.015}$
	Self-Eval	$.573_{\pm.106}$	$.526_{\pm.014}$	$.585_{\pm.111}$	$.523_{\pm.013}$
	Self-Consistency	$.628_{\pm.077}$	$.553_{\pm.032}$	$.554_{\pm.100}$	$.554_{\pm.178}$
	InternalInspector	$.504_{\pm.197}$	$.507_{\pm.016}$	$.527_{\pm.039}$	$.715_{\pm.024}$
	A1	$.571_{\pm.060}$	$.504_{\pm.015}$	$.549_{\pm.092}$	–
	A2	$.563_{\pm.058}$	$.512_{\pm.012}$	$.547_{\pm.096}$	–
	A3	$.554_{\pm.046}$	$.513_{\pm.011}$	$.555_{\pm.094}$	$.581_{\pm.016}$
	Ours	$.557_{\pm.035}$	$.519_{\pm.012}$	$.569_{\pm.087}$	$.586_{\pm.019}$

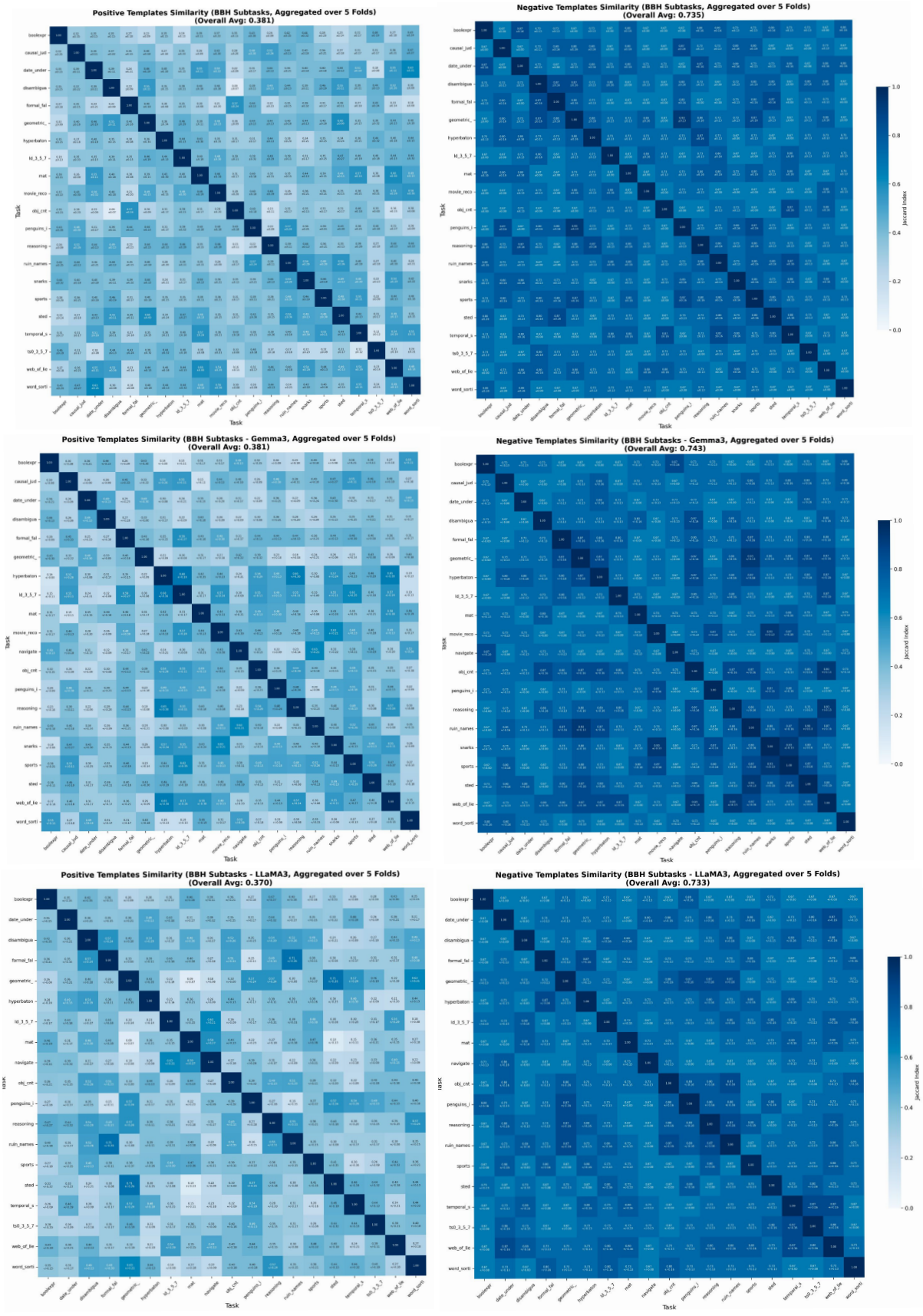
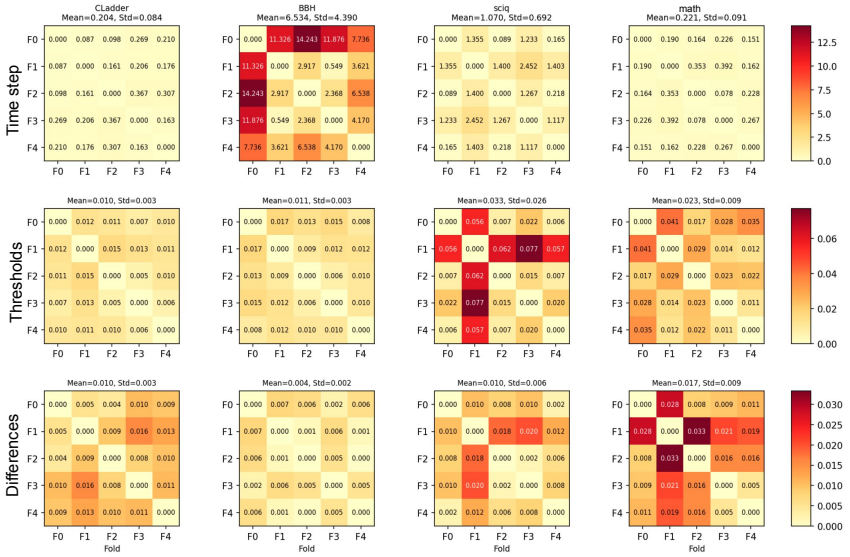


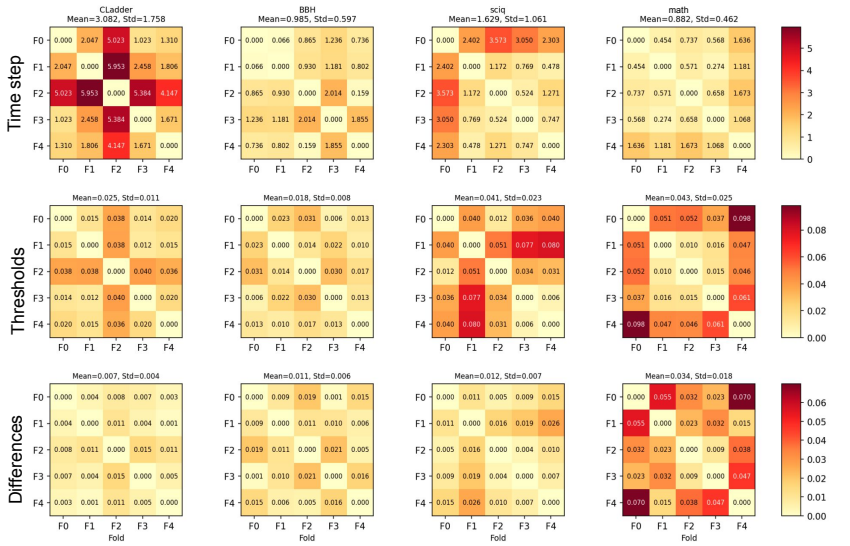
Figure 9: Pairwise similarity of mined STL formulas across BBH subtasks. Results are aggregated over five folds and shown separately for correct (left) and incorrect (right) STL patterns, across different backbone models. Correct STL patterns exhibit low similarity across subtasks, while incorrect patterns remain highly consistent. This trend mirrors the dataset-level analysis and suggests asymmetric generalization behavior at finer task granularity.

Table 12: Cross-Fold Parameter MAE (Mean $\pm$ Std) for Different Models and Datasets

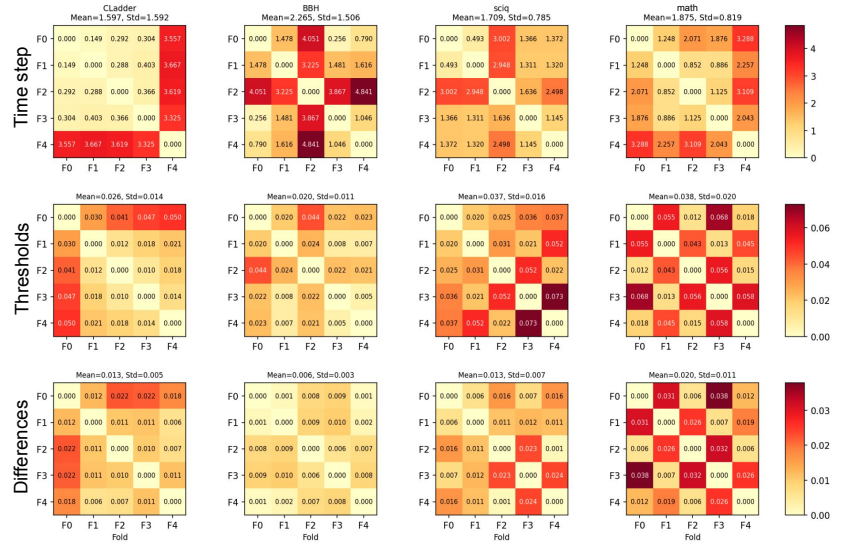
Type	Model	CLadder	BBH	sciq	math
Time step	Qwen3	0.204 ± 0.084	6.534 ± 4.390	1.070 ± 0.692	0.221 ± 0.091
	Llama	3.082 ± 1.758	0.985 ± 0.597	1.629 ± 1.061	0.882 ± 0.462
	Gemma	1.597 ± 1.592	2.265 ± 1.506	1.709 ± 0.785	1.875 ± 0.819
Threshold	Qwen3	0.010 ± 0.003	0.011 ± 0.003	0.033 ± 0.026	0.023 ± 0.009
	Llama	0.025 ± 0.011	0.018 ± 0.008	0.041 ± 0.023	0.043 ± 0.025
	Gemma	0.026 ± 0.014	0.020 ± 0.011	0.037 ± 0.016	0.038 ± 0.020
Difference	Qwen3	0.010 ± 0.003	0.004 ± 0.002	0.010 ± 0.006	0.017 ± 0.009
	Llama	0.007 ± 0.004	0.011 ± 0.006	0.012 ± 0.007	0.034 ± 0.018
	Gemma	0.013 ± 0.005	0.006 ± 0.003	0.013 ± 0.007	0.020 ± 0.011



(a) Qwen3

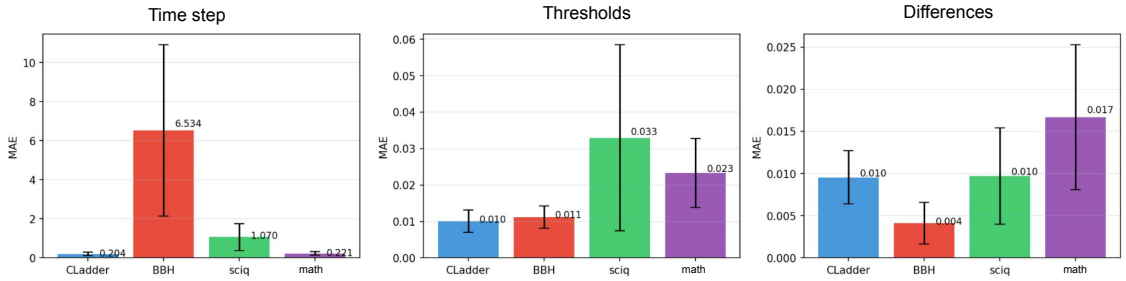


(b) Gemma3

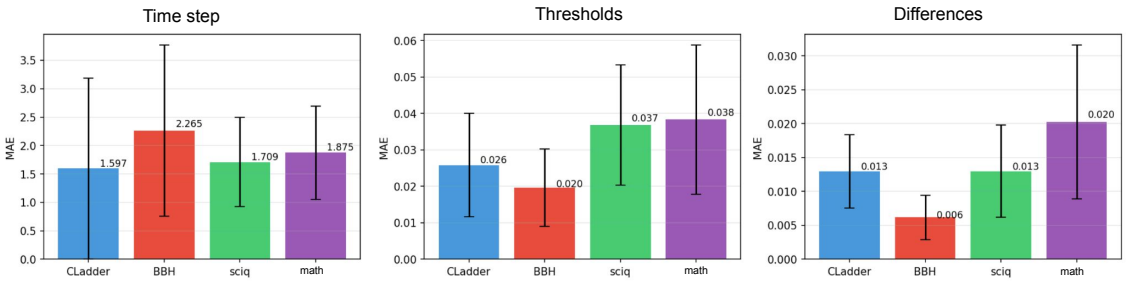


(c) Llama3

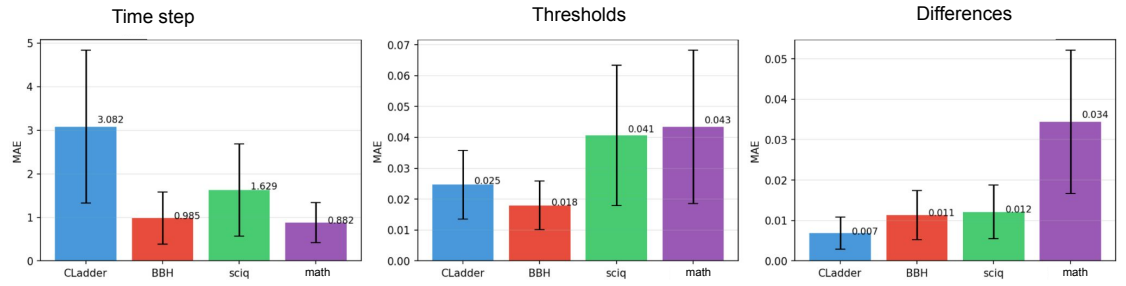
Figure 10: Instance-level STL parameter variability under a fixed temporal structure. Each row corresponds to a different parameter category.



(a) Qwen3



(b) Gemma3



(c) Llama3

Figure 11: RQ2: Question-level sensitivity of optimized STL parameters under a fixed formula structure. We report the mean absolute difference (MAE) of learned parameters across individual questions, grouped by parameter type: temporal bounds (left), predicate thresholds (middle), and difference-related parameters (right). Results are shown for four datasets across three backbone models (Qwen3, Gemma3, Llama3). While temporal parameters remain relatively stable, threshold and difference parameters exhibit substantially higher variability across questions, particularly on BBH and CLadder.

Table 13: Intra-dataset question similarity measured using TF-IDF and sentence-transformer embeddings. Values are reported as mean and standard deviation of pairwise cosine similarity within each dataset. Higher similarity does not necessarily correspond to lower STL parameter variability, suggesting that parameter sensitivity is not solely explained by surface-level or semantic similarity among questions.

Dataset	TF-IDF		Sentence Transformers	
	Mean	Std	Mean	Std
CLadder	0.054	0.125	0.276	0.172
BBH	0.181	0.136	0.767	0.197
sciq	0.565	0.093	0.716	0.062
math	0.103	0.168	0.405	0.144