DoG: Sniffing Out Overconfidence in LLM Agents via Post-hoc Trajectory Restructuring

Hyunjun Jeon* **Dongha Lim*** **Kunwoong Kim** **Daewon Choi** **Jinwoo Shin**
Korea Advanced Institute of Science and Technology
{hyunjunian, dongha.lim, kunwoong.kim, daeone0920, jinwoos}@kaist.ac.kr

Abstract

The overconfidence of Large Language Model (LLM) agents poses a critical challenge to their reliable deployment for complex, high-stakes tasks. Current confidence estimation methods predominantly treat agent execution as a flat, time-ordered sequence. This oversimplification fundamentally fails to capture the complex logical dependencies, branching paths, error propagation, and tool interactions inherent in agentic reasoning. To address this, we introduce **DAG of Grounds (DoG)**, a novel method for agent confidence estimation via post-hoc trajectory restructuring. By transforming linear trajectories into Directed Acyclic Graphs (DAGs), DoG explicitly maps the grounding of an agent's final answer to its intermediate reasoning steps and tool outputs. We evaluate DoG on three benchmarks (GAIA, GPQA, and HLE), which involve long-horizon reasoning and iterative tool use. DoG outperforms existing calibration methods designed for general LLM outputs, performs comparably to or better than methods specifically designed for agent trajectories, and remains well calibrated across different LLM backbones and tool settings when its calibrator is fit on each. Without refitting, its ranking of trajectories also transfers across benchmarks, backbones, and tool settings. Furthermore, we release an interactive diagnostic tool that offers interpretability for analyzing opaque failure modes. Together, these contributions establish a structured, graph-based paradigm for confidence estimation that "sniffs out" overconfidence in autonomous systems.

1 Introduction

Large Language Models (LLMs) are increasingly used as autonomous agents that plan, call external tools, and complete multi-step tasks [18, 24, 34]. A central issue for these agents is *overconfidence*: agents often report high confidence on incorrect or logically flawed answers, and this miscalibration is observed even in recent frontier models [1]. Yet existing calibration techniques developed for static LLM responses, such as Verbalized Confidence, self-consistency, and token-level log-probability aggregation [6, 13, 15, 17, 29, 31], face fundamental difficulties when applied to agents. As they are designed for short, single-turn outputs, they fail to account for the multi-step trial-and-error process induced by tool calls and noisy intermediate observations [10, 14, 32]. Moreover, sampling-based methods such as self-consistency are impractical in such setting, as each trajectory involves numerous tool calls and incurs substantial API costs, making repeated sampling prohibitively expensive.

In response to these limitations, a recent line of work has introduced agent-specific calibration techniques that estimate confidence by analyzing the trajectory itself [21]. Despite this progress, two structural gaps persist across these methods. First, most rely on direct access to token-level log-probabilities from the underlying model, or invoke an additional external embedding model to score trajectory components [28, 37, 39]. As a result, such requirements render them inapplicable to

*Equal contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

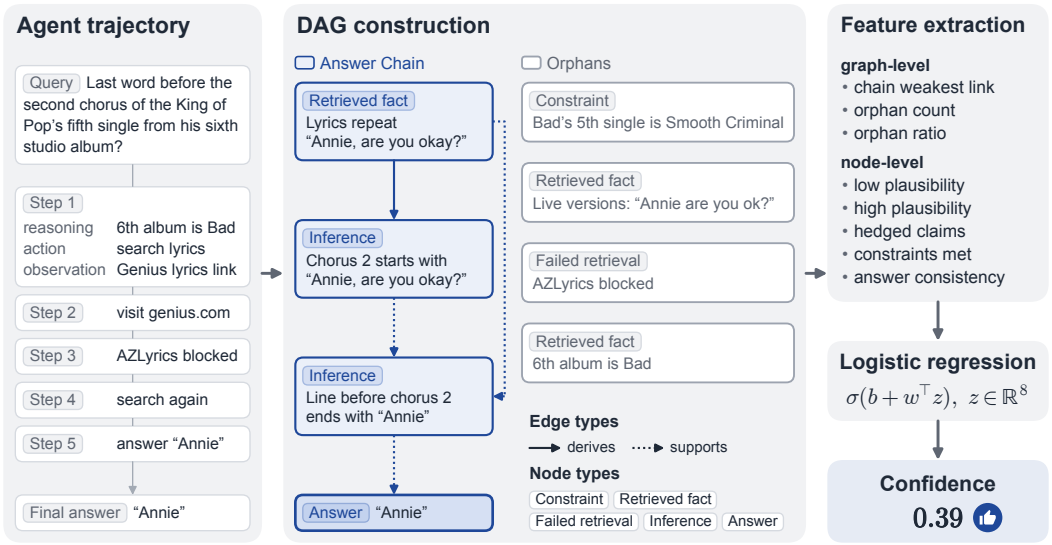


Figure 1: Overview of DoG on a real GAIA trajectory of GPT-5.4-nano with a wrong answer. DAG construction makes one post-hoc call to the same LLM to turn the trajectory into a DAG of grounds; the answer rests on the Answer Chain (blue), while the premises that identify the song are orphans (gray). Then, without any LLM call, a 9-parameter logistic regression on eight features extracted from the DAG gives an out-of-fold confidence of only 0.39. All contents are real; texts are shortened.

the increasingly dominant class of black-box commercial agents and incur extra latency and inference cost. Second, they treat the trajectory as a purely linear sequence ordered by time [28, 37, 39]. In practice, however, real agent behavior often involves branching, retries, error correction, and exploratory detours; a flat sequential view cannot distinguish steps that genuinely contributed to the final answer from those that did not. As a result, the confidence of pivotal steps is diluted by irrelevant noise, and the contribution signal, arguably the most informative cue for calibration, is lost.

We observe that an agent's reasoning is inherently graph-structured rather than linear. Recent work on structured reasoning, including Tree of Thoughts [33], Graph of Thoughts [2], and DAG-based formulations of chain-of-thought [38], has shown that exposing the non-linear structure of an LLM's thought process is essential for understanding how a model reaches its answer. The same holds for agentic execution, where tool calls selectively depend on prior steps and unproductive branches are discarded without contributing to the final answer. A linear trajectory can thus be retrospectively restructured into a DAG that captures these dependencies. The agent's own model is a natural choice here, since it needs no extra model, while other LLMs can also perform it (Appendix F).

Building on this insight, we propose **DAG of Grounds (DoG)**, a post-hoc calibration framework in which the agent's own model reflects on its completed trajectory in a single extra call to restructure it into a DAG and estimates confidence from this graph. Given a finished trajectory, DoG prompts this model to recover its underlying grounding structure as a DAG whose nodes are discrete grounds (pieces of evidence or intermediate conclusions produced during execution) and whose edges encode logical dependencies among them. Confidence is then estimated from two complementary types of signals: (i) node-level features that capture the reliability of each individual ground, and (ii) graph-level features that capture how these grounds combine to support the final answer [Section 3]. By explicitly modeling inter-step dependencies and the contribution path to the answer, DoG addresses the limitations of linear trajectory representations. Because it operates entirely on the completed trajectory using the agent's own outputs, DoG requires no additional rollouts, no fine-tuning of the base LLM, and no white-box access. This makes it directly applicable to black-box models and pluggable into any agent framework as a post-hoc calibrator with only small computational overhead [Appendix G]. It leaves the agent loop unchanged, and only its 9-parameter calibrator is fit, on labeled trajectories from the target setting (Section 3.3). Our main contributions are:

1. **Trajectory restructuring confidence estimation framework.** We propose DoG, a novel calibration framework in which an agent autonomously restructures its own reasoning trajectory into a DAG to estimate its confidence [Section 3]. This framework simultaneously

addresses two longstanding limitations: (i) existing approaches that treat trajectories as linear sequences, failing to capture the differential contributions of individual steps, and (ii) methods that rely on log-probability access or external embedding models, rendering them inapplicable to black-box language models.

2. **Comprehensive calibration evaluation.** We empirically demonstrate that our proposed method outperforms existing calibration baselines across three benchmarks. Furthermore, we quantitatively analyze the contribution of each DAG-derived feature to overall calibration performance and, finally, demonstrate that DoG stays calibrated across a wide range of agent environments when fit in each of them, and that its ranking transfers across them without refitting [Section 4].
3. **Test-time scaling application and open-source interactive diagnostic tool.** We show that the estimated confidence scores can be effectively leveraged for test-time scaling, improving the agent's final answer accuracy over plain majority voting when they are used to weight the votes of multiple rollouts [Section 4.4]. In addition, we release an open-source interactive diagnostic tool that allows users to visualize and explore the process of DAG trajectory construction and confidence estimation [Appendix A].

2 Related work

Confidence calibration for static LLMs. Confidence calibration for LLMs has been studied widely in the context of AI safety. Existing approaches for static, single-turn LLMs fall into three categories: logit-based methods that extract probabilities from the model's output distribution [9, 12], consistency-based methods that measure agreement across multiple sampled generations [6, 15, 30], and verbalization-based methods that prompt the LLM to state its certainty [17, 29]. Although logit-based methods are conceptually direct, the internal probabilities of RLHF-tuned models are often miscalibrated and can exacerbate overconfidence [13]. More broadly, these methods are designed to calibrate a single static output rather than the long-horizon process through which an agent produces its final answer. As a result, they do not transfer well to agentic settings, where answers emerge from iterative tool use, noisy intermediate observations, and dynamic environmental feedback. This limitation is especially pronounced for consistency-based methods, since repeated rollouts of full agent trajectories can be expensive and may trigger real-world side effects.

Confidence calibration in agents. A line of recent works estimates the reliability of LLM agents by analyzing their execution trajectories. SAUP [39] and UProp [5] propagate step-level uncertainty along the trajectory, weight each step by a learned situation-awareness factor, and accumulate a single scalar across the trace. TRACER [28] combines content-aware surprisal with behavioral signals such as repetition and action-observation mismatch, and aggregates them via a tail-focused risk functional. HTC [37] skips per-step modeling and extracts a fixed-dimensional process-level feature vector from the full execution, then passes it to a lightweight calibrator. A separate line calibrates specific components rather than full trajectories: MICE-for-CATs [26] scores individual tool calls from model-internal features, and the Confidence Dichotomy [32] separates evidence-providing tools from verification tools for calibration fine-tuning. Across this literature, trajectories are treated as temporal streams, without explicitly modeling the dependencies among steps.

Graph-based reasoning and confidence estimation. Representing LLM reasoning as a graph rather than a linear chain has been studied outside of confidence estimation. Tree of Thoughts [33] and Graph of Thoughts [2] construct branching structures during generation to improve task accuracy, DAG-Math [38] models chain-of-thought as a stochastic process over DAGs to study rule-consistency, and self-reflection methods such as Reflexion [25] analyze completed traces to produce corrective feedback. A few recent works extend this view to uncertainty quantification. Graph Uncertainty [11] uses centrality on a bipartite claim-response graph built from many sampled long-form generations, and All Roads Lead to Rome [36] applies Katz centrality and path-convergence scoring to a directed graph that aggregates multiple sampled reasoning chains for math and logic tasks. Topo-UQ [4] computes pairwise graph-edit distance between reasoning topologies elicited from multiple explanations, and EDTR [19] extracts topological risk features from the distribution of multiple chain-of-thought paths. However, these methods generally rely on multiple sampled generations, explanations, or reasoning paths, making them expensive and potentially unsafe in agent settings. DoG instead builds its graph from a single recorded trajectory, without extra rollouts or changes to the execution loop.

3 Method: DAG of Grounds (DoG)

3.1 Problem formulation: Post-hoc calibration for black-box agents

Black-box LLM agent and its trajectory. We consider a black-box LLM $\mathcal{M}$, accessible only through an API and with no access to internal token logits or hidden states. Given an input query x, the LLM agent solves the task by interacting with an external environment through a sequence of reasoning and tool use steps. Following standard agentic frameworks such as ReAct [34], we write the execution trajectory as $\tau = (s_1, s_2, \ldots, s_T)$, where T is the number of steps until the agent terminates. Each step $s_t = (r_t, a_t, o_t)$ is a triplet of reasoning r_t (*Thought*), tool call a_t (*Action*), and tool output o_t (*Observation*). The agent then produces a final answer $y = \mathcal{M}(x, \tau)$.

Post-hoc confidence calibration. The objective of confidence calibration is to estimate a score $c \in [0, 1]$ for the final answer y such that c approximates the probability that y matches the ground-truth answer y^*. Formally, a perfectly calibrated estimator satisfies $\mathbb{P}(y = y^* \mid c = p) = p$ for all $p \in [0, 1]$. We focus on the post-hoc setting, in which c is assigned after y has been produced, without modifying $\mathcal{M}$, its decoding procedure, or the trajectory generation process.

For proprietary black-box LLMs, white-box signals such as token-level sequence likelihoods $\prod_t \mathbb{P}(y_t \mid y_{<t}, x, \tau)$ [9] and internal self-evaluation probabilities [13] are not accessible. Therefore, a black-box estimator f should operate merely on text, as $f(x, \tau, y) \to c$.

3.2 DAG construction from trajectory

DoG operates as a two-stage pipeline: Stage 1 [Section 3.2] reorganizes τ into a structured DAG via a single LLM call, and Stage 2 [Section 3.3] reduces the DAG to a scalar with no further LLM calls. The complete end-to-end pseudocode of the DoG framework is provided in Appendix B.

After the agent produces its final answer y, the same LLM is invoked once more in a separate post-hoc call to reconstruct the grounding structure $\mathcal{G}$ from the completed execution record (x, τ, y). This call is isolated from the original tool-using rollout: it only reads the completed record, with no additional tool calls or interaction with the external environment. Within this post-hoc call, the model is instructed to identify the grounds that support the final answer and their logical dependencies, while also enumerating plausible alternative answers to reduce post-hoc anchoring on the agent's own answer. Appendix M gives the prompt, and Appendix F validates the resulting DAGs.

Taxonomy of nodes and edges. We define five node types based on the role each piece of evidence takes in the trajectory. The node set $\mathcal{V}$ is partitioned as follows:

- **Constraint.** Sub-questions or requirements extracted from the input query. These are the root nodes of the graph and form $\mathcal{V}_C$.
- **Retrieved Fact.** Observations produced by successful tool calls that return useful information, denoted by $\mathcal{V}_{RF}$.
- **Failed Retrieval.** Tool calls that return empty, erroneous, or irrelevant observations (e.g., 404 errors, paywalls). We keep these as $\mathcal{V}_{FR}$ to retain failure signals for calibration.
- **Inference.** Logical claims drawn from the agent's reasoning steps, without new external evidence, denoted by $\mathcal{V}_I$.
- **Answer.** The unique terminal node v_a representing the final answer y, giving $\mathcal{V}_A = \{v_a\}$.

Formally, $\mathcal{V} = \mathcal{V}_C \cup \mathcal{V}_{RF} \cup \mathcal{V}_{FR} \cup \mathcal{V}_I \cup \mathcal{V}_A$. For nodes $u, v \in \mathcal{V}$, a directed edge $e_{u \to v} \in \mathcal{E}$ represents a relationship between u and v, drawn from three edge types:

- **Supports.** Node u provides evidence confirming node v.
- **Refutes.** Node u provides evidence contradicting node v.
- **Derives.** Node v follows logically from node u.

We instruct the builder LLM to draw edges only along the temporal order of τ (i.e., u precedes v), so that $\mathcal{G}$ is acyclic. In practice, at least 98.2% of the DAGs are acyclic (Appendix F).

3.3 Confidence calibration via DAG

While existing approaches aggregate risk along a single linear trajectory, DoG estimates confidence in two steps based on the extracted graph $\mathcal{G}$. We first map $\mathcal{G}$ to an 8-dimensional feature vector $x \in \mathbb{R}^8$ that combines graph-level and node-level signals, and then pass x through a logistic regression to obtain the confidence score $c \in [0, 1]$.

Feature extraction. The feature vector x consists of two complementary parts.

First, **node-level signals** measure local properties of individual nodes, such as logical leaps between reasoning steps, hedging in the agent's reasoning, and the satisfaction of *Constraint* nodes.

Second, **graph-level signals** measure structural properties of $\mathcal{G}$ that a linear trajectory cannot expose. A challenge in extracting them is that exploratory dead-ends and steps that support the final answer coexist in $\mathcal{G}$. To separate the two, we define the *Answer Chain* as the ancestor set of the final answer node v_a:

$$\mathcal{A}(v_a) = \{v \in \mathcal{V} \mid v \rightsquigarrow v_a \text{ in } \mathcal{G}\}, \tag{1}$$

where $v \rightsquigarrow v_a$ denotes a directed path along parent links of any edge type, and $v_a \in \mathcal{A}(v_a)$. Concrete graph-level features include the observation bottleneck of $\mathcal{A}(v_a)$ and the number of orphan nodes outside it—neither of which admits a linear-trajectory analog.

The precise definitions of all eight features are given in Appendix H, and see Table 2 for the effect of each feature on the calibration performance [Section 4.3].

Calibration model. The raw feature vector x is first standardized using statistics $(\hat{\mu}, \hat{\sigma})$ computed on the training folds only, yielding $z = (x - \hat{\mu})/\hat{\sigma}$. The confidence score is then calculated via a simple logistic regression:

$$c \;=\; \sigma(b + w^\top z) \;=\; \frac{1}{1 + \exp(-(b + w^\top z))}, \tag{2}$$

where the learned weights $w \in \mathbb{R}^8$ dynamically adjust confidence based on the DAG's structural health, and the intercept $b \in \mathbb{R}$ absorbs the base-rate accuracy of the calibration pool. The calibrator has only $|\theta| = 9$ learnable scalars in total, by design.

Fitting protocol. We fit a dedicated calibrator per evaluation cell—a (dataset, base LLM, tool regime) triple—and report out-of-fold predictions. The results in the main paper use per-cell settings for the number of folds, seed repeats, and C, several of them chosen on the cell's own out-of-fold metrics. The appendix analyses of non-DAG features, transfer, and DAG builders use nested cross-validation, which keeps the choice of C out of the reported predictions (Appendix I).

4 Experiments

4.1 Experimental setup

Tasks and benchmark datasets. To evaluate DoG, we consider three public benchmarks that expose various failure modes. Each benchmark offers a distinct axis of reasoning and vulnerability: **GAIA** [18] evaluates long-horizon, tool-mediated problem solving, where failures often arise from how agents retrieve, combine, and rely on external evidence. **GPQA** [22] evaluates difficult scientific reasoning, where failures are more likely to stem from domain reasoning errors than from tool use failures. **HLE-Verified** [35] evaluates frontier-level question answering across diverse domains, where agents may produce plausible answers despite weak or incomplete support. Together, these benchmarks allow us to evaluate DoG across tool-intensive execution, domain-specialized reasoning, and broad knowledge uncertainty while keeping the agent harness and post-hoc calibration protocol fixed. Dataset splits, sample counts, and answer-format details are provided in Appendix C.

Agent setup. We use a unified tool-augmented agent harness across all benchmarks. For each instance, the agent alternates between reasoning and tool use before returning a final answer. The base LLM, decoding parameters, prompt template, tool interface, and stopping criterion are held fixed across calibration methods. Full details regarding these configurations, exact prompts, and

hyperparameters are provided in Appendix J. This setup ensures that calibration differences reflect the confidence estimator rather than changes in agent behavior.

Evaluation protocol and metrics. For each benchmark dataset, we first run the agent once to obtain a completed trajectory τ and final prediction y. All confidence estimators are then evaluated post-hoc on this fixed trajectory, hence the differences in calibration performance reflect the estimator, rather than variation in sampling or tool execution. At test time, confidence estimators may inspect the trajectory and final prediction, but may not modify the trajectory, re-run the agent, or access ground-truth responses. Given τ and y, each method outputs a scalar confidence score $\hat{p} \in [0, 1]$ estimating the probability that y is correct, which is evaluated against the binary correctness label derived from the answer.

To evaluate the quality of these confidence estimates, we use Expected Calibration Error (ECE) [9], Brier score [3], and AUROC [7]. ECE and Brier score are the primary calibration metrics: ECE measures the gap between predicted confidence and empirical accuracy across confidence bins, while Brier score measures the squared error of the probabilistic prediction. We compute ECE with the standard equal-width estimator using $M = 10$ bins. Results under other estimators are in Appendix D. We also consider AUROC as a discrimination metric, capturing whether confidence scores rank correct answers above incorrect ones rather than whether the scores are calibrated probabilities.

Baselines. We compare DoG against representative confidence estimation baselines. To ensure a fair comparison, all baselines are evaluated post-hoc and given the identical fixed trajectory τ as context, and we restrict our comparison to single-pass (sampling-free) methods that do not require additional agent rollouts; detailed prompts and hyperparameters are provided in Appendix K. We organize the baselines along the same two axes used to characterize the limitations of prior work in Section 1: whether they treat τ as a flat (linear) context or operate on its execution steps, and whether they require token-level log-probabilities or external models.

Static LLM baselines. Following the taxonomy of Tao et al. [27], we select one representative from each of the three families: P(True) [13] for token probability-based uncertainty (TPU), Verbalized Confidence [29] for numerical verbal uncertainty (NVU), and Linguistic Hedging [27] for linguistic verbal uncertainty (LVU). All three are adapted to the agentic setting by prepending the entire trajectory τ as a single flat textual context.

- **P(True)** prompts the model to judge whether the proposed answer y is correct given τ, and takes the probability assigned to the "True" token as the confidence score.
- **Verbalized Confidence** asks the agent to report a numerical confidence for y given τ.
- **Linguistic Hedging** prompts the agent to reflect on y given τ in natural language and uses an LLM judge to map implicit linguistic cues (e.g., hedging words such as "probably") to a numerical uncertainty score, rather than relying on explicitly reported probabilities.

Agent-specific baselines. We further compare against recent state-of-the-art methods explicitly designed to use the execution steps of agentic trajectories. Although these methods exploit the step-wise structure of τ, they still treat it as a one-dimensional, time-ordered sequence, and rely on token-level log-probabilities or auxiliary external scoring models to obtain uncertainty signals. They also require labeled trajectories from a held-out validation set to fit their internal parameters.

- **TRACER** [28] aggregates step-wise signals into a trajectory-level risk score via a tail-focused risk functional with a max-composite step risk, where the semantic repetition and coherence signals are computed using an external sentence-embedding model.
- **HTC** [37] extracts hand-crafted process-level features from the agent's token-level log-probability trajectory, and fits a regularized linear logistic calibrator on these features. Since its official implementation is not publicly available, we report the numbers from the original paper as a reference.

4.2 Main results: Calibration performance

Table 1 reports calibration performance on three benchmarks. DoG ranks first or second in all nine (benchmark, metric) cells, a level of consistency not achieved by any competing method. It obtains

Table 1: Main calibration results across benchmarks. **Bold** and underline denote the best and second-best values in each column. Base accuracies: 35.5% (GPQA), 64.2% (GAIA), 22.0% (HLE). DoG values are means over 5 runs (standard deviations and per-run data are in Appendix D). †HTC results are copied from the original paper, where base LLM and base accuracy differ from our setup.

Method	**GPQA**			**GAIA**			**HLE**		
	ECE ↓	Brier ↓	AUROC ↑	ECE ↓	Brier ↓	AUROC ↑	ECE ↓	Brier ↓	AUROC ↑
Zero-shot									
P(True)	0.406	0.394	0.666	0.173	0.223	0.715	0.453	0.434	0.547
Verbalized	0.282	0.287	0.729	0.150	0.235	0.655	0.332	0.290	0.637
Hedging	0.205	0.254	0.728	0.297	0.318	0.661	0.182	0.223	0.626
Supervised									
TRACER	0.058	0.221	0.616	0.054	0.230	0.547	0.088	0.167	0.632
HTC†	0.102	0.213	0.706	0.142	0.233	0.686	**0.031**	**0.090**	0.644
DoG (ours)	**0.045**	**0.195**	**0.731**	**0.041**	**0.194**	**0.727**	0.056	0.156	**0.709**

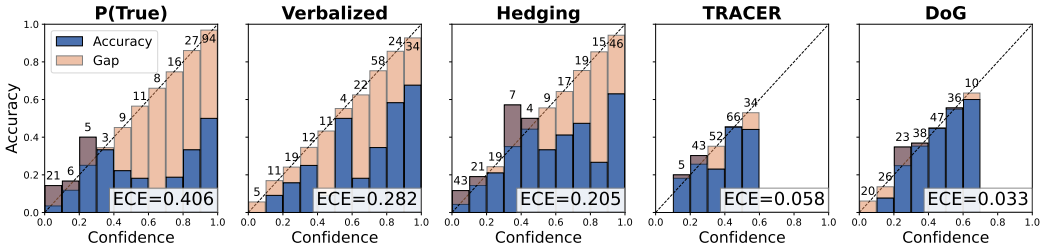


Figure 2: Reliability diagrams on GPQA (HLE and GAIA in Appendix D). DoG is shown for Run 2 in Table 7, since per-bin accuracies cannot be averaged across runs.

the best results on all three metrics for GPQA and GAIA, and trails only HTC on HLE ECE and Brier. DoG results are averaged over five independent DAG generations on the same trajectories; per-cell standard deviations (max $\sigma_{\mathrm{ECE}} = 0.017$, $\sigma_{\mathrm{Brier}} = 0.009$, $\sigma_{\mathrm{AUROC}} = 0.028$ [Appendix D]) preserve the ranking at $\pm 1\sigma$ in every cell except GPQA AUROC, where DoG leads Verbalized by 0.002, within run-to-run variation. The agent-specific methods (DoG, TRACER, HTC) are all supervised: DoG's calibrator is fit out-of-fold [Appendix I], TRACER is tuned via grid search, and HTC is fit as a supervised logistic calibrator on its log-probability features (see Appendix K for full description for the baselines). The static LLM baselines (P(True), Verbalized, Hedging) are zero-shot. DoG nonetheless leads TRACER on every cell and HTC on seven of nine, suggesting that its gains are not simply obtained from supervised tuning.

The static LLM baselines collapse as benchmark difficulty rises: P(True)'s ECE grows monotonically from 0.173 on GAIA to 0.453 on HLE, the confirmation-bias failure anticipated when a verifier judges from a flattened trajectory. The agent-specific baselines, TRACER and HTC, partially close this gap but still treat the trajectory as a one-dimensional sequence. TRACER pays the cost most visibly: the weakest GAIA AUROC of any method (0.547), unable to separate correct from incorrect rollouts despite competitive ECE. Figure 2 makes the failure mode concrete: TRACER's predictions never extend beyond mid confidence (~ 0.6), suppressing risk by avoiding confident claims rather than discriminating among them. DoG instead maps the topological dependencies among steps, preserving signals (such as the observation bottleneck on the Answer Chain or the orphan-node count) that have no direct analog in a linear trajectory. Section 4.3 reports a leave-one-out ablation in which every feature contributes positively to ECE; removing the chain-restricted weakest-link feature alone degrades AUROC by 0.060, three to four times larger than any other feature, indicating that DAG-derived signals are decisive for discrimination.

The two cells in which DoG does not lead are both HLE losses to HTC and should be carefully interpreted, since they are obtained under different base-agent accuracy regimes. HTC's results come from using GPT-4.1 at 5.4% HLE accuracy [23], while ours uses GPT-5.4-nano at 22%. Because the baseline Brier score for a naive constant predictor is $p(1-p)$, the two settings naturally induce different scales, making raw Brier scores difficult to compare directly (0.051 vs. 0.172). This regime can also make ECE less informative, since a constant predictor at $p = 0.054$ already attains $\mathrm{ECE} = 0$.

Table 2: Leave-one-out feature ablation on GAIA. Δ denotes the change relative to the full-feature baseline.

Dropped feature	ΔECE ↑	ΔBrier ↑	ΔAUROC ↓
Low-Plausibility Inferences	+0.0662	+0.0034	-0.0174
Constraint Satisfaction	+0.0312	+0.0020	-0.0136
Hedged Claims	+0.0298	+0.0027	-0.0157
High-Plausibility Inferences	+0.0219	-0.0009	+0.0080
Orphan Ratio	+0.0175	-0.0003	+0.0014
Orphan Count	+0.0135	-0.0010	+0.0021
Answer Consistency	+0.0112	-0.0009	-0.0018
Chain-Restricted Weakest Link	+0.0060	+0.0018	-0.0596

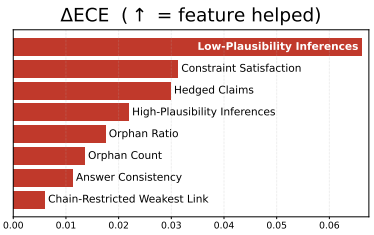


Figure 3: ΔECE per dropped feature on GAIA. Results for ΔBrier and ΔAUROC are in Table 2.

Normalizing via the Brier Skill Score, $\mathrm{BSS} = 1 - \mathrm{Brier}/p(1-p)$, yields (i) $\mathrm{BSS} = -0.76$ for HTC, indicating worse performance than the trivial base-rate predictor under this normalization, and (ii) $\mathrm{BSS} = +0.091$ for DoG, the highest among the compared HLE results (TRACER: $+0.027$; all static LLM baselines are negative). Compatibility analyses for various kinds of base LLMs and tool sets are shown in Section 4.5. DoG adds one post-hoc LLM call per task, at a bounded cost of about \$0.0023 regardless of trajectory length. This overhead is small, under 10% of the agent's rollout cost on GAIA. Verbalized Confidence can be cheaper on short trajectories (GPQA, HLE), but its cost grows with trajectory length, so DoG is already cheaper on long GAIA trajectories [Appendix G].

4.3 Analysis: Feature importance

To evaluate the contribution of each feature in DoG, we conduct a leave-one-out (LOO) ablation under the calibration protocol in Appendix I. We compare each change Δ with GAIA's run-to-run standard deviations ($\sigma_{\mathrm{ECE}} = 0.008$, $\sigma_{\mathrm{Brier}} = 0.009$, and $\sigma_{\mathrm{AUROC}} = 0.027$; [Appendix D]) to assess whether the effect is larger than the variation. On ECE, seven of eight features have Δ above one σ_{ECE}, with Low-Plausibility Inferences the largest single-feature contributor at $+0.0662$ (about 8σ). Constraint Satisfaction ($+0.0312$) and Hedged Claims ($+0.0298$) follow at $\sim 4\sigma$ each, and Figure 3 visualizes the full ECE ranking. The structurally correlated pair Orphan Ratio and Orphan Count ($\rho = 0.90$) retain distinct $\sim 2\sigma$ effects under L2 regularization, justifying their joint inclusion. By contrast, no ΔBrier exceeds σ_{Brier} (largest $+0.0034$, $\sim 0.4\sigma$). Hence, it does not support strong per-feature Brier claims on GAIA. The Brier improvement reported in Table 1 remains robust as a property of the full DoG feature suite.

On AUROC, only Chain-Restricted Weakest Link exceeds σ_{AUROC}: dropping this feature degrades AUROC by 0.0596 ($\sim 2.2\sigma$), three to four times larger than the effect of any other feature. This feature is also weakly correlated with the rest of the feature suite ($|\rho| \leq 0.09$) and suggesting that it captures a discrimination signal that is not well represented by the other seven features. Thus, this analysis identifies the answer-chain-level weakest-link signal as the main contributor to DoG's ranking accuracy on GAIA, consistent with the mechanism claimed in Section 4.2.

Appendix D also compares DoG with feature sets that need no graph, namely trajectory statistics, the same semantic judgments elicited as a flat list, and DoG without topology. DoG has the best Brier score and AUROC on every benchmark but not the lowest ECE, so node-level semantic judgments carry much of the calibration signal while the graph structure mainly adds discrimination.

4.4 Application: Test-time scaling with DoG

Beyond calibration metrics, we evaluate whether DoG provides a useful confidence signal for downstream decisions. We run the GPT-5.4-nano agent four more times on each of the 165 GAIA tasks, score every new rollout with a calibrator fit on the original GAIA trajectories, and select one answer per task. No labels from the new rollouts are used. Picking the rollout with the highest DoG score (DoG argmax) improves on a single rollout by 2.6 points but trails majority voting (0.491 vs. 0.504 in Table 3). Agreement is the stronger signal here. DoG separates correct from incorrect rollouts well when they are pooled across tasks (AUROC 0.74), but less well among the four rollouts of one task, which share the task's difficulty (within-task AUROC 0.60).

Combining the two works best. Weighting each vote by its DoG score raises accuracy to 0.533 ($+3.0$ points over majority voting) and recovers 41% of the gap between the mean single rollout and the oracle, compared with 23% for majority voting alone. In our runs, the weights never overturn a strict plurality, so DoG changes the outcome only on the 39 tasks where the plurality is tied. On these tasks, DoG-weighted voting picks a correct answer in 31% of cases, versus 18% expected under random tie-breaking. Because DoG needs only one auxiliary LLM call per rollout for DAG construction, it provides a lightweight selection signal without extra agent rollouts or external reward models. Appendix D gives the setup details and a breakdown by vote pattern.

Table 3: Best-of-4 trajectory selection accuracy on all 165 GAIA tasks.

Selection Method	Accuracy	Δ vs. Mean
Mean Single Rollout	0.465	—
Majority Vote [30]	0.504	+0.038
DoG Argmax	0.491	+0.026
DoG-Weighted Vote	**0.533**	**+0.068**
Oracle (Any Correct)	0.630	+0.165

4.5 Compatibility: DoG across agent environments

A confidence estimator that is useful in practice must remain effective when the underlying agent stack changes, either by swapping the language model driving the agent or by reconfiguring the inventory of tools available to it. We therefore stress-test DoG on the GAIA benchmark along these two axes and summarize the results in Tables 4 and 5.

Compatibility across base models. Table 4 evaluates DoG when the underlying agent is instantiated with seven different LLMs spanning a wide range of vendors, capacity tiers, and architectures: frontier closed-weight models (GPT-5.4, Claude-opus-4.7, Gemini-3.1-pro), their efficient counterparts (GPT-5.4-nano, Claude-haiku-4.5, Gemini-3.1-flash-lite), and an open-weight mixture-of-experts model (Qwen-3.5-35B-A3B). Across all seven backbones, DoG produces well-calibrated and discriminative estimates: ECE remains below 0.09, AUROC stays above 0.67, and no model exhibits a degenerate failure mode along any of the three metrics. The narrow range across vendors and capacity tiers suggests that DoG does not strongly depend on a particular model family's output style. This stability is consistent with the use of structural signals from DoG, such as grounding edges, branching structure, and tool-call dependencies, rather than surface-level token patterns.

Table 4: Compatibility across base models.

Model	ECE ↓	Brier ↓	AUROC ↑
GPT-5.4	0.057	0.197	0.694
GPT-5.4-nano	0.041	0.194	0.727
Claude-opus-4.7	0.056	0.158	0.719
Claude-haiku-4.5	0.069	0.198	0.677
Gemini-3.1-pro	0.042	0.157	0.675
Gemini-3.1-flash-lite	0.087	0.216	0.698
Qwen-3.5-35B-A3B	0.061	0.196	0.698

Compatibility across tool sets. Table 5 fixes the backbone and instead varies the action space available to the agent across three regimes: *No Tool* (pure chain-of-thought reasoning), *Basic Tool* (only web available), and *Full Tool* (the complete suite used in our main experiments). Calibration error (ECE) remains low across all three regimes (≤ 0.041). Part of this reflects the low base accuracy without tools (6.4%), where a constant prediction at the base rate would already have near-zero ECE and a Brier score of 0.060, so AUROC is the more informative comparison there. AUROC and Brier do shift across regimes, with discrimination becoming harder as more tools are introduced. We emphasize that this shift is not a degradation of DoG itself but a reflection of intrinsic task difficulty: a richer toolset unlocks longer-horizon, multi-step problems that are inherently harder to predict correctly, with deeper DAGs, more branching, and more tool-induced failure modes. Even under this most demanding regime, DoG surpasses the strongest baselines reported in our main experiments [Section 4.2] under the same Full Tool conditions.

Table 5: Compatibility across tool sets (base model: GPT-5.4-nano).

Tool Set	ECE ↓	Brier ↓	AUROC ↑
No Tool	0.027	0.032	0.908
Basic Tool	0.041	0.143	0.825
Full Tool	0.041	0.194	0.727

Three levels of robustness. We separate three levels at which robustness can hold. (i) *Per-setting calibration.* With a calibrator fit in each setting, DoG stays calibrated and discriminative across all seven backbones and three tool regimes (Tables 4 and 5). (ii) *Zero-shot ranking.* A calibrator fit on

one benchmark, backbone, or tool regime and applied unchanged to the others keeps its ranking. Mean AUROC changes by only -0.007, -0.017, and -0.008 across benchmarks, backbones, and tool regimes, whereas the transfer results reported in HTC's original paper, under a different base LLM, show its mean AUROC falling from 0.679 to 0.567 (Appendix E). (iii) *Zero-shot calibration.* Absolute calibration does not transfer. Zero-shot ECE rises to 0.198, 0.121, and 0.247, following the gap in base accuracy between source and target. Re-estimating only the intercept b on labeled target data closes 93%, 79%, and 79% of this gap, but that is target adaptation rather than zero-shot calibration. We therefore claim robustness at levels (i) and (ii) and treat (iii) as a limitation (Section 5).

5 Conclusion

We presented **DAG of Grounds (DoG)**, a post-hoc calibration framework that restructures a black-box agent's trajectory into a DAG and maps it to a calibrated scalar over node- and graph-level features. Across GPQA, GAIA, and HLE-Verified, DoG ranks first or second on every (benchmark, metric) cell, remains calibrated across seven base LLMs and three tool regimes when fit in each setting, and keeps its ranking without refitting across benchmarks, backbones, and tool regimes—all without logits, external models, or additional rollouts. Used to weight votes over four rollouts, it also recovers 41% of the oracle gap in Best-of-4 selection. DoG has three main limitations. First, its DAGs are built by an LLM, by default the agent's own model, and contain errors. Human validation finds 8.0% of randomly sampled edges unsupported, based on a single annotator, although other LLMs used as builders show no systematic self-construction bias (Appendix F). Second, the calibrator is fit on labeled trajectories from the target setting. Its ranking transfers without refitting, but its absolute calibration does not, and refitting the intercept recovers it only with target labels (Appendix E). Third, DoG adds one post-hoc LLM call per trajectory (Appendix G), which stays off the agent's critical path but delays the confidence score by one call. Our lightweight calibrator may also compress higher-order topological signals. Future work includes label-free calibration under distribution shift, richer GNN-based readouts over the DAG, and integration into online agent control loops for early stopping and tool-budget allocation. We release an interactive diagnostic tool [Appendix A] to facilitate these directions and hope that explicitly modeling the graph structure of agent reasoning becomes a standard component of future calibration research.

Acknowledgments and Disclosure of Funding

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)). We also sincerely thank Daehee Han, HyunKyu Kim, Yeeun Yoo, and Youngjun Kwak of the KakaoBank Financial Tech Lab for their helpful discussions. The authors declare no competing interests.

References

[1] Casey O. Barkan, Sidney Black, and Oliver Sourbut. Do large language models know what they are capable of? In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=EO6WtJ0q6G.

[2] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL http://dx.doi.org/10.1609/aaai.v38i16.29720.

[3] Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1 – 3, 1950. doi: 10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2. URL https://journals.ametsoc.org/view/journals/mwre/78/1/1520-0493_1950_078_0001_vofeit_2_0_co_2.xml.

[4] Longchao Da, Xiaoou Liu, Jiaxin Dai, Lu Cheng, Yaqing Wang, and Hua Wei. Understanding the uncertainty of LLM explanations: A perspective based on reasoning topology. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=p4wZfBFgyI.

[5] Jinhao Duan, James Diffenderfer, Sandeep Madireddy, Tianlong Chen, Bhavya Kailkhura, and Kaidi Xu. UProp: Investigating the uncertainty propagation of LLMs in multi-step decision-making, 2026. URL https://openreview.net/forum?id=NnlelrGapm.

[6] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625 – 630, 2024. URL https://api.semanticscholar.org/CorpusID:270615909.

[7] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[8] Center for AI Safety and Scale AI. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025. URL https://arxiv.org/abs/2501.14249.

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

[10] Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. Towards uncertainty-aware language agent. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6662–6685, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.398. URL https://aclanthology.org/2024.findings-acl.398/.

[11] Mingjian Jiang, Yangjun Ruan, Prasanna Sattigeri, Salim Roukos, and Tatsunori Hashimoto. Graph-based uncertainty metrics for long-form language model generations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=YgJPQW0lkO.

[12] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 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, 2021. doi: 10.1162/tacl_a_00407. URL https://aclanthology.org/2021.tacl-1.57/.

[13] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

[14] Michael Kirchhof, Gjergji Kasneci, and Enkelejda Kasneci. Position: Uncertainty quantification needs reassessment for large language model agents. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025. URL https://openreview.net/forum?id=Lrv20S5RZV.

[15] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

[16] Ananya Kumar, Percy S. Liang, and Tengyu Ma. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems*, 2019.

[17] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=8s8K2UZGTZ.

[18] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

[19] Abhishek More, Anthony Zhang, Nicole Bonilla, Ashvik Vivekan, Kevin Zhu, Parham Sharafoleslami, and Maheep Chaudhary. Optimizing chain-of-thought confidence via topological and dirichlet risk analysis. In *AAAI 2026 Workshop on Assessing and Improving Reliability of Foundation Models in the Real World*, 2026. URL https://openreview.net/forum?id=ZcJyoPZAAM.

[20] Jeremy Nixon, Mike Dusenberry, Ghassen Jerfel, Linchuan Zhang, and Dustin Tran. Measuring calibration in deep learning, 2020. URL https://openreview.net/forum?id=r1la7krKPS.

[21] Changdae Oh, Seongheon Park, To Eun Kim, Jiatong Li, Wendi Li, Samuel Yeh, Xuefeng Du, Hamed Hassani, Paul Bogdan, Dawn Song, and Sharon Li. Uncertainty quantification in LLM agents: Foundations, emerging challenges, and opportunities. In *Agentic AI in the Wild: From Hallucinations to Reliable Autonomy*, 2026. URL https://openreview.net/forum?id=eMptSxXg9g.

[22] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

[23] Scale AI and Center for AI Safety. Humanity's last exam leaderboard, 2025. URL https://scale.com/leaderboard/humanitys_last_exam. GPT-4.1 accuracy: 5.40%. Last updated December 17, 2025.

[24] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

[25] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

[26] Nishant Subramani, Jason Eisner, Justin Svegliato, Benjamin Van Durme, Yu Su, and Sam Thomson. Mice for cats: Model-internal confidence estimation for calibrating agents with tools. *CoRR*, abs/2504.20168, April 2025. URL https://doi.org/10.48550/arXiv.2504.20168.

[27] Linwei Tao, Yi-Fan Yeh, Minjing Dong, Tao Huang, Philip Torr, and Chang Xu. Revisiting uncertainty estimation and calibration of large language models, 2025. URL https://arxiv.org/abs/2505.23854.

[28] Sina Tayebati, Divake Kumar, Nastaran Darabi, Davide Ettori, Ranganath Krishnan, and Amit Ranjan Trivedi. Tracer: Trajectory risk aggregation for critical episodes in agentic reasoning, 2026. URL https://arxiv.org/abs/2602.11409.

[29] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=g3faCfrwm7.

[30] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

[31] Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gjeQKFxFpZ.

[32] Weihao Xuan, Qingcheng Zeng, Heli Qi, Yunze Xiao, Junjue Wang, and Naoto Yokoya. The confidence dichotomy: Analyzing and mitigating miscalibration in tool-use agents, 2026. URL https://arxiv.org/abs/2601.07264.

[33] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h.

[34] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

[35] Weiqi Zhai, Zhihai Wang, Jinghang Wang, Boyu Yang, Xiaogang Li, Xander Xu, Bohan Wang, Peng Wang, Xingzhe Wu, Anfeng Li, et al. Hle-verified: A systematic verification and structured revision of humanity's last exam. *arXiv preprint arXiv:2602.13964*, 2026.

[36] Caiqi Zhang, Chang Shu, Ehsan Shareghi, and Nigel Collier. All roads lead to Rome: Graph-based confidence estimation for large language model reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31814–31824, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1620. URL https://aclanthology.org/2025.emnlp-main.1620/.

[37] Jiaxin Zhang, Caiming Xiong, and Chien-Sheng Wu. Agentic confidence calibration, 2026. URL https://arxiv.org/abs/2601.15778.

[38] Yuanhe Zhang, Ilja Kuzborskij, Jason D. Lee, Chenlei Leng, and Fanghui Liu. DAG-math: Graph-of-thought guided mathematical reasoning in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ylr6WArKQN.

[39] Qiwei Zhao, Dong Li, Yanchi Liu, Wei Cheng, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Huaxiu Yao, Chen Zhao, Haifeng Chen, and Xujiang Zhao. Uncertainty propagation on LLM agent. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6064–6073, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.302. URL https://aclanthology.org/2025.acl-long.302/.

A Release: an interactive diagnostic tool

We provide a web-based interactive diagnostic tool for constructing reasoning DAGs from LLM trajectories and measuring confidence calibration. The diagnostic tool is publicly accessible at:

https://calibration-playground.vercel.app/

This platform offers a range of functionalities to facilitate experimentation with DAG-based confidence estimation. The core features include, but are not limited to, the following:

1. **Task authoring and dataset import.** Creating tasks consisting of a question and ground-truth answer, either manually or by importing benchmark splits (e.g., GPQA) from the Hugging Face Hub.
2. **Trajectory generation.** Producing a reasoning trajectory and final answer for each task using a configurable LLM under the three tool regimes of Appendix J: no tools, basic tools (web search and webpage visits), or all tools (adding Python execution, image inspection, and audio transcription).
3. **DAG construction.** Decomposing a trajectory into a DAG of atomic claims, where each node carries a textual claim, parent dependencies, and more.
4. **Confidence estimation.** Extracting the eight node- and graph-level features from the DAG and mapping them to a calibrated confidence with the logistic regression calibrator (Section 3.3), alongside a baseline verbal self-confidence elicited directly from the model.
5. **Automated scoring.** Grading the final answer against the ground truth with benchmark-specific rules, namely normalized exact match for numeric, list, and string answers following the official GAIA scorer and option-letter matching for multiple-choice questions, or with an optional LLM judge that uses the official HLE grading prompt for open-ended answers.
6. **Calibration analytics.** Visualizing the relationship between predicted confidence (DAG-derived and verbal) and empirical accuracy across tasks through calibration plots.
7. **Configuration and customization.** Selecting the model provider, API endpoint, model, and tool regime, and overriding the prompts of the trajectory and DAG construction stages.
8. **Data portability.** Persisting all tasks, trajectories, DAGs, and scores locally in the browser, with import and export support in JSON format.

B Algorithm for DAG of Grounds (DoG)

Algorithm 1 provides a comprehensive, step-by-step pseudo-code for the DoG framework. As a post-hoc calibrator, DoG operates on a completed trajectory to estimate its reliability. The algorithm explicitly consists of the three core phases: restructuring the 1D trace into an epistemic DAG while isolating the genuine Answer Chain, extracting topological and semantic features, and computing the final calibrated confidence by applying the fitted logistic regression model.

Algorithm 1 Overall framework of DAG of Grounds (DoG)

Require: Query x, completed agent trajectory τ, and final answer y
Require: An auxiliary Calibrator LLM $\mathcal{M}_{cal}$
Require: Standardization pool statistics $(\hat{\mu}, \hat{\sigma})$, Learned LR weights $w \in \mathbb{R}^8$ and intercept $b \in \mathbb{R}$
Ensure: Calibrated confidence score $c \in [0, 1]$

// Stage 1: DAG Construction & Structural Isolation [Section 3.2]

$\mathcal{G}(\mathcal{V}, \mathcal{E}) \leftarrow$ PARSEZEROSHOTDAG$(\mathcal{M}_{cal}, x, \tau, y)$ ▷ Distill into taxonomy
$v_a \leftarrow$ Unique Answer node in $\mathcal{V}_A$
$\mathcal{A}(v_a) \leftarrow \{v \in \mathcal{V} \mid v \rightsquigarrow v_a \text{ in } \mathcal{G}\}$ ▷ Isolate genuine load-bearing Answer Chain
$\mathcal{V}_{orphan} \leftarrow \mathcal{V} \setminus \mathcal{A}(v_a)$ ▷ Isolate exploratory dead-ends

// Stage 2-1: Topological Feature Extraction [Section 3.3]

$\mathbf{x}_{graph} \leftarrow$ EXTRACTGRAPHSIGNALS$(\mathcal{G}, \mathcal{A}(v_a), \mathcal{V}_{orphan})$ ▷ e.g., chain-restricted bottleneck
$\mathbf{x}_{node} \leftarrow$ EXTRACTNODESIGNALS$(\mathcal{G})$ ▷ e.g., logical leaps, hedges, constraints
$\mathbf{x} \leftarrow \text{Concat}(\mathbf{x}_{graph}, \mathbf{x}_{node}) \in \mathbb{R}^8$ ▷ Form the fixed-dimensional feature vector

// Stage 2-2: Parametric Calibration Model [Section 3.3]

$\mathbf{z} \leftarrow (\mathbf{x} - \hat{\mu}) / \hat{\sigma}$ ▷ Standardize with training-fold statistics
$c \leftarrow \frac{1}{1+\exp(-(b+w^\top \mathbf{z}))}$ ▷ Compute probability via logistic regression
return c

C Benchmark datasets

To comprehensively evaluate the calibration performance of the DoG framework, we utilize three highly challenging, frontier-level benchmarks. To ensure a reproducible and balanced evaluation, we sample subsets from the official validation or test splits using a seeded Fisher-Yates shuffle (seed = 42). Below, we detail the characteristics, sample sizes, and scoring protocols for each dataset.

GAIA. GAIA [18] is designed to evaluate general-purpose AI assistants on long-horizon tasks that require multi-step reasoning, tool orchestration, and web browsing.

- **Setup & sampling:** We use all $N = 165$ questions in the `2023_all` validation split, which encompasses a comprehensive mixture of all difficulty tiers (Levels 1, 2, and 3).
- **Scoring protocol:** The dataset naturally expects concise, definitive answers (e.g., specific numbers, entities, or short strings). We utilized a strict, normalized exact-match (EM) scoring function as defined in the official GAIA evaluation scripts to determine task success.

GPQA. GPQA [22] is a challenging dataset comprising PhD-level scientific questions spanning physics, biology, and chemistry. It specifically tests deep domain expertise.

- **Setup & sampling:** We evaluate agents on the `gpqa_main` split. To significantly increase the reasoning difficulty and enforce rigorous generative reasoning, we evaluate GPQA in an *open-ended* configuration by entirely dropping the four multiple-choice options.
- **Offline filtering:** Removing the options leaves some question stems ill-posed (e.g., stems that say "Which of the following..."). For each of the 448 questions in `gpqa_main`, we ask `GPT-5.4-nano` whether the stem needs the hidden choices to be answered, and take a 3-call majority vote. This flags 126 stems (28%) and leaves 322 open-ended questions, from which we sample $N = 200$ tasks with seed 42.
- **Scoring protocol:** Since the output format is open-ended, we employed an LLM-as-a-judge (`GPT-5.4-nano`) to evaluate whether the agent's derived answer semantically matched the ground-truth string.

HLE-Verified. HLE-Verified [8, 35] represents the extreme frontier of reasoning benchmarks, designed to be highly resistant to current scaling paradigms. It involves complex, multi-disciplinary question answering where even state-of-the-art models frequently fail despite producing highly confident, plausible-sounding rationales.

- **Setup & sampling:** We utilize the open-ended configuration from the `gold` split of the `HLE-Verified` subset, ensuring high-quality and unambiguous ground truths. We sample $N = 200$ queries.
- **Scoring protocol:** We strictly adhere to the official HLE evaluation protocol. The agent's final output was parsed using the official LLM judge prompt routed through `GPT-5.4-nano` on the Responses API. The judge extracted the final answer from the agent's prose and returned a strict boolean verdict (`yes`/`no`) by comparing it against the ground truth.

D Additional experimental results

Reliability diagrams. Figure 4 presents the reliability diagrams for all evaluated methods across the GPQA, GAIA, and HLE benchmarks. Each plot illustrates the calibration of the models by comparing confidence levels against empirical accuracy. We observe that baseline methods such as P(True) and Verbalized often exhibit significant overconfidence, as indicated by the large gaps between the bars and the identity line, resulting in higher ECE values. In contrast, TRACER and the proposed DoG method consistently show much tighter alignment with the identity line across all three benchmarks. Notably, DoG achieves the lowest ECE values in almost all cases, demonstrating superior calibration performance and more reliable confidence estimation compared to other baselines.

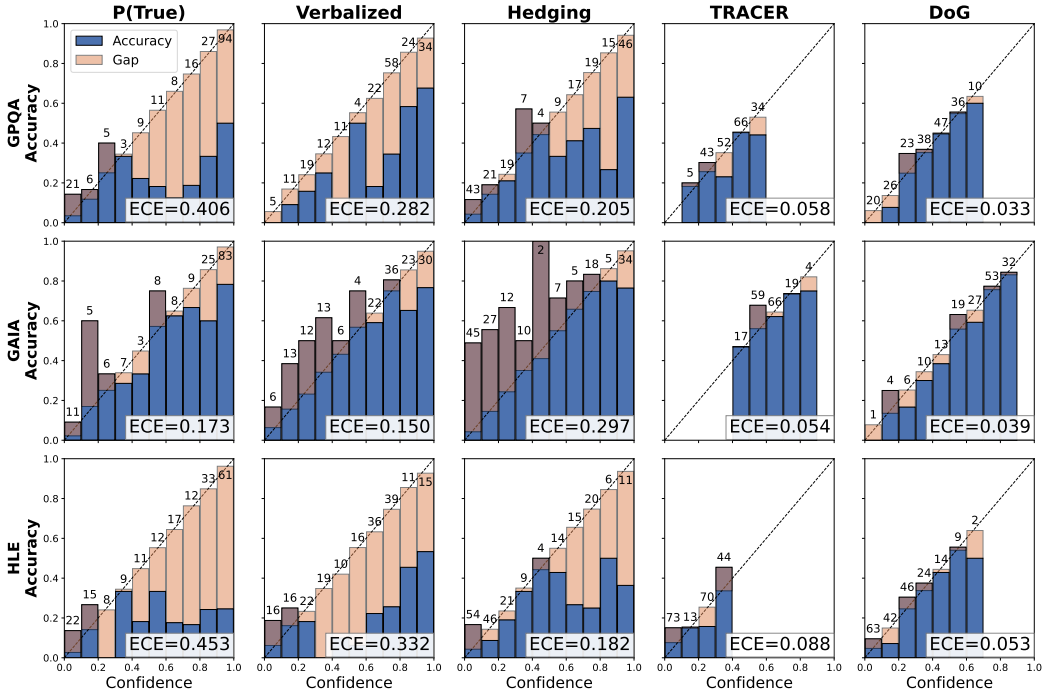


Figure 4: Comparison of methods in terms of reliability diagrams on GPQA, GAIA, and HLE benchmarks. DoG is shown for Run 2 in Table 7.

Standard deviations. To assess the stability and reproducibility of our results, we perform five independent DAG generation runs using the *GPT-5.4-nano* agent for the three datasets. Table 6 summarizes the mean and standard deviation for the ECE, Brier score, and AUROC, with detailed per-run results provided in Table 7. The low standard deviations indicate that DoG maintains consistent calibration performance despite the stochastic nature of the DAG generation process. While there are some variations, particularly in the AUROC and ECE metrics for the HLE dataset, the overall consistency suggests that the reported performance does not strongly depend on specific random seeds or executions.

Table 6: Per-dataset calibration metrics (mean $\pm$ std) over five independent DAG generations on GPT-5.4-nano with full tools.

Dataset	ECE	Brier	AUROC
GPQA	0.0450 ± 0.0119	0.1954 ± 0.0020	0.7307 ± 0.0080
HLE	0.0560 ± 0.0171	0.1560 ± 0.0053	0.7086 ± 0.0281
GAIA	0.0408 ± 0.0080	0.1935 ± 0.0090	0.7269 ± 0.0272

Table 7: Per-run calibration metrics across five independent DAG generations on GPT-5.4-nano.

Dataset	Metric	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GAIA	ECE	0.0348	0.0390	0.0461	0.0327	0.0518	0.0408	0.0080
	Brier	0.1955	0.1888	0.1988	0.1807	0.2036	0.1935	0.0090
	AUROC	0.7200	0.7414	0.7179	0.7638	0.6914	0.7269	0.0272
GPQA	ECE	0.0538	0.0326	0.0528	0.0544	0.0315	0.0450	0.0119
	Brier	0.1960	0.1954	0.1941	0.1984	0.1931	0.1954	0.0020
	AUROC	0.7312	0.7232	0.7387	0.7219	0.7383	0.7307	0.0080
HLE	ECE	0.0560	0.0528	0.0389	0.0479	0.0843	0.0560	0.0171
	Brier	0.1564	0.1506	0.1547	0.1539	0.1646	0.1560	0.0053
	AUROC	0.7203	0.7401	0.7076	0.7115	0.6636	0.7086	0.0281

Robustness of ECE to the estimator. Table 1 uses the standard equal-width ECE with $M = 10$ bins. To check that our conclusions do not depend on this choice, we recompute ECE from per-task predictions with equal-width binning (5, 10, 15, and 20 bins), equal-mass (adaptive) binning (10 and 15 bins) [20], and the debiased ℓ_1 estimator [16, Appendix G] with 15 equal-mass bins (Gaussian approximation, 4,000 Monte Carlo samples). As in Table 1, DoG values are means over 5 runs, and baselines are single runs. HTC is taken from its original paper without per-task scores, so we cannot recompute it.

Table 8: ECE under different estimators. EW-k: equal-width with k bins (EW-10 is used in Table 1); EM-k: equal-mass with k bins; ℓ_1-deb.: debiased ℓ_1 ECE with 15 equal-mass bins. DoG values are means $\pm$ standard deviations over 5 runs. **Bold**: best; underline: second best.

Dataset	Method	EW-5	EW-10	EW-15	EW-20	EM-10	EM-15	ℓ_1-deb.
GPQA	P(True)	0.405	0.406	0.414	0.414	0.394	0.396	0.391
	Verbalized	0.282	0.282	0.282	0.282	0.288	0.282	0.269
	Hedging	0.196	0.205	0.205	0.212	0.188	0.200	0.178
	TRACER	**0.036**	0.058	**0.065**	0.088	0.086	0.089	**0.035**
	DoG (ours)	0.040$\pm$.007	**0.045**$\pm$.011	**0.065**$\pm$.015	**0.066**$\pm$.006	**0.053**$\pm$.012	**0.080**$\pm$.009	0.038$\pm$.015
GAIA	P(True)	0.165	0.173	0.176	0.189	0.161	0.162	0.136
	Verbalized	0.137	0.150	0.167	0.164	0.158	0.171	0.153
	Hedging	0.297	0.297	0.299	0.310	0.296	0.294	0.282
	TRACER	0.053	0.054	**0.066**	**0.064**	0.082	0.108	0.062
	DoG (ours)	**0.030**$\pm$.007	**0.041**$\pm$.007	0.067$\pm$.010	0.094$\pm$.018	**0.081**$\pm$.010	**0.095**$\pm$.025	**0.049**$\pm$.032
HLE	P(True)	0.453	0.453	0.463	0.463	0.447	0.444	0.438
	Verbalized	0.332	0.332	0.332	0.336	0.335	0.331	0.322
	Hedging	0.154	0.182	0.179	0.200	0.198	0.219	0.208
	TRACER	0.036	0.088	0.084	0.099	0.104	0.105	0.074
	DoG (ours)	**0.022**$\pm$.004	**0.056**$\pm$.015	**0.061**$\pm$.006	**0.080**$\pm$.015	**0.069**$\pm$.016	**0.093**$\pm$.020	**0.064**$\pm$.027

Across all 21 (benchmark, estimator) cells, DoG's ECE is lower than that of every zero-shot baseline. DoG ranks first in 17 cells and second in the other 4, each time behind TRACER only. In 3 of these 4 cells, the gap is within DoG's 5-run standard deviation. Thus, the ranking in Table 1 is not an artifact of the 10-bin choice.

Comparison of calibration performance for generally optimized methods. While all three agent-specific confidence calibration methods (TRACER, HTC, and DoG) can achieve high calibration performance on a specific target domain through careful hyperparameter tuning, a crucial metric for real-world applicability is the ability to maintain this performance across a wide range of domains without requiring domain-specific adjustments. Table 9 compares how well these methods rank trajectories (AUROC) across three diverse benchmarks (GAIA, GPQA, and HLE) when configured with a single, universally optimized set of hyperparameters (denoted as "general" weights).

As shown in the results, existing baselines exhibit significant inconsistencies across domains. TRACER performs poorly on GAIA (0.538), while HTC suffers severe performance drops on GPQA (0.581) and HLE (0.521). In contrast, DoG outperforms the baselines across all evaluated

Table 9: Calibration results across benchmarks of generally optimized confidence estimation methods.

Benchmark	Method	AUROC ↑
GAIA	TRACER (general)	0.538
	HTC (general)	0.643
	DoG (general)	**0.704**
GPQA	TRACER (general)	0.603
	HTC (general)	0.581
	DoG (general)	**0.719**
HLE	TRACER (general)	0.605
	HTC (general)	0.521
	DoG (general)	**0.642**

benchmarks using the exact same general weights. By achieving the highest AUROC scores on GAIA (0.704), GPQA (0.719), and HLE (0.642), DoG ranks trajectories better than the baselines in every domain with a single set of weights. This table covers ranking only. Absolute calibration does not carry over across domains in the same way, as Appendix E shows.

Comparison with non-DAG features. Unlike the LOO ablation in Section 4.3, which keeps the graph, Table 10 compares DoG with feature sets that need no graph, all fit with the same calibrator under nested cross-validation. They are eight trajectory statistics such as step count and failed retrievals, the same five semantic judgments elicited as a flat list, and DoG without its three topology features. DoG has the best Brier score and AUROC on every benchmark. Under a paired bootstrap over tasks, its AUROC lead is significant against trajectory statistics on all three benchmarks ($p \leq 0.003$), against the flat judgments on GAIA, and against DoG without topology on GAIA and GPQA ($p < 0.02$). The graph does not lower ECE, however. Withholding the three topology features together lowers mean ECE on all three benchmarks, unlike the one-at-a-time drops in Table 2, which use a different protocol. Trajectory statistics reach the lowest ECE on HLE only by predicting the base rate (AUROC 0.50). Node-level semantic judgments thus carry much of the calibration signal, while the graph structure mainly adds discrimination.

Table 10: DoG against feature sets that need no graph, all fit with the same calibrator under the nested cross-validation of Appendix E (means over five Stage-1 runs). The DoG row therefore differs from Table 1. **Bold** and underline denote the best and second-best values in each column.

Features	GPQA			GAIA			HLE		
	ECE ↓	Brier ↓	AUROC ↑	ECE ↓	Brier ↓	AUROC ↑	ECE ↓	Brier ↓	AUROC ↑
Trajectory statistics	0.044	0.225	0.622	0.106	0.229	0.546	**0.009**	0.174	0.503
Flat judgments	**0.037**	0.201	0.695	0.059	0.212	0.651	0.047	0.167	0.632
DoG w/o topology	0.048	0.207	0.676	**0.036**	0.201	0.669	0.042	0.163	0.658
DoG (ours)	0.053	**0.197**	**0.728**	0.052	**0.195**	**0.721**	0.046	**0.160**	**0.686**

Best-of-4 selection details. We run the GPT-5.4-nano all-tools agent four more times on each of the 165 GAIA tasks, which gives 660 new trajectories. We build their DAGs with the same prompt as in the main experiments (no parse failures) and score them with one calibrator fit on the 165 original GAIA trajectories with $C = 0.07$, which is also the median value that nested cross-validation selects on GAIA. No label from the new rollouts is used to fit or tune this calibrator. The new rollouts cover the same tasks as the calibration pool, because the 165 GAIA validation tasks are the only ones with public answers. They are, however, shorter (7.5 vs. 9.7 steps on average) and less accurate (0.465 vs. 0.642) than the original trajectories. For voting, we lowercase each answer and strip non-alphanumeric characters. Three rollouts with an empty answer are scored as incorrect and do not vote.

Table 11 breaks the results down by vote pattern. Agreement carries most of the signal. A unanimous answer is correct in 80% of tasks, a 3–1 majority in 52%, and a 2–1–1 plurality in 29%. DoG argmax

can override the majority, and this costs accuracy on 3–1 and 2–1–1 splits. DoG-weighted voting never overrides a strict plurality in these runs, so it matches majority voting on every untied task. On the 39 tied tasks, it picks a correct answer in 12, against 7.1 expected at random. A correct answer is among the tied ones in 18 of these tasks, so 6 remain unrecovered.

DoG scores rank rollouts well across tasks but bunch together within a task. Pooled over the new rollouts, their AUROC is 0.743. Within a task, the standard deviation of the four scores is 0.085 on average, against 0.180 over all rollouts, and a correct rollout outscores an incorrect rollout of the same task in only 107 of the 177 such pairs (60%). Because the new rollouts are less accurate than the calibration pool, the scores are also too high in level (mean score 0.62 against an accuracy of 0.47, with ECE 0.153). Selection uses only the ranking within each task, so this shift in level does not affect it.

Table 11: Best-of-4 accuracy on GAIA by vote pattern, i.e., the number of rollouts that share each distinct answer. "All distinct" includes 2 tasks with empty answers. Majority vote breaks ties at random (scored in expectation).

Vote pattern	**Tasks**	**Single**	**DoG Argmax**	**Majority**	**DoG-Weighted**	**Oracle**
4 (unanimous)	64	0.797	0.797	0.797	0.797	0.797
3–1	31	0.419	0.355	0.516	0.516	0.645
2–1–1	31	0.194	0.226	0.290	0.290	0.484
2–2 (tie)	13	0.346	0.462	0.346	0.385	0.692
All distinct (tie)	26	0.087	0.231	0.099	0.269	0.346
All	165	0.465	0.491	0.504	0.533	0.630

E Robustness under distribution shift

Tables 4 and 5 fit a separate calibrator in each setting. Here we fit it on one setting and apply it unchanged to another, along three axes, namely benchmark, backbone, and tool regime. For each source, we choose C by 5-fold cross-validation on the source alone, fit one calibrator on all source trajectories, and apply it to the target without using any target label. Diagonal cells are in-domain out-of-fold estimates under nested cross-validation (outer 10-fold, repeated twice). As in Table 1, the benchmark axis averages five DAG regenerations. The backbone and tool axes use one DAG set per agent, so their GPT-5.4-nano, GAIA, all-tools cell differs slightly (in-domain ECE 0.039 vs. 0.052). Trajectories without a correctness label or a parsable DAG are excluded. Because of these choices, the in-domain values here differ from those in Tables 1, 4, and 5.

The ranking transfers. Mean AUROC changes from the in-domain value by only -0.007, -0.017, and -0.008 on the three axes (Tables 12–14). Absolute calibration does not. Mean ECE rises from 0.050 to 0.198, from 0.071 to 0.121, and from 0.045 to 0.247, and the rise follows the gap in base accuracy between source and target. GPQA and HLE, the closest pair in accuracy (0.355 and 0.221), transfer with the lowest ECE (0.061 and 0.085). The worst cell is all tools to no tools (0.483), the pair furthest apart in accuracy (0.642 and 0.064). The transfer results reported for HTC in its original paper show a similar rise in ECE across benchmarks (from 0.092 to 0.242), but HTC loses far more ranking, with mean AUROC falling from 0.679 to 0.567, against 0.712 to 0.705 for DoG.

Because the intercept b absorbs the base rate of the calibration pool (Section 3.3), we also freeze the source's scaler and feature weights and refit only b on the target, fitting it on training folds and scoring held-out folds. This single parameter closes 93%, 79%, and 79% of the zero-shot ECE gap on the three axes and leaves AUROC essentially unchanged. The fitted shift in b tracks the gap in logit base accuracy (correlations of 0.91, 0.68, and 0.97), and 8 target labels already cut mean ECE from 0.205 to 0.099 across benchmarks. The refit is not free, since 5 of the 42 backbone cells get worse by up to 0.029. More importantly, it uses target labels, so it is a lightweight adaptation rather than zero-shot calibration. We therefore claim robustness only for per-setting calibration and zero-shot ranking, and treat zero-shot calibration as a limitation (Section 5).

Table 12: Transfer across benchmarks (GPT-5.4-nano, all tools). Rows are the source the calibrator is fit on, and columns are the target. Diagonal cells (italic) are in-domain. DoG values are means over five DAG regenerations, with standard deviations for the zero-shot cells. †HTC values are copied from its original paper [37], whose base LLM and base accuracy differ from ours.

Fit ↓ / Eval →	GAIA	GPQA	HLE
Base accuracy	0.642	0.355	0.221
DoG, ECE ↓, zero-shot			
GAIA	*0.052*±.010	0.220±.055	0.195±.079
GPQA	0.313±.030	*0.053*±.020	0.061±.015
HLE	0.317±.027	0.085±.028	*0.046*±.011
DoG, ECE ↓, after refitting b			
GAIA	*0.052*	0.056	0.054
GPQA	0.080	*0.053*	0.050
HLE	0.048	0.079	*0.046*
DoG, AUROC ↑, zero-shot			
GAIA	*0.721*±.028	0.692±.024	0.703±.036
GPQA	0.703±.032	*0.728*±.006	0.699±.041
HLE	0.720±.027	0.711±.016	*0.686*±.023
HTC†, ECE ↓, zero-shot			
GAIA	*0.142*	0.135	0.387
GPQA	0.134	*0.102*	0.233
HLE	0.297	0.268	*0.031*

Table 13: Zero-shot transfer across backbones (GAIA, all tools). Rows are the source and columns the target. Diagonal cells (italic) are in-domain, and the last column averages the six off-diagonal cells of each row. Columns abbreviate GPT-5.4-nano (nano), Claude-opus-4.7 (Opus), Claude-haiku-4.5 (Haiku), Gemini-3.1-pro (Pro), Gemini-3.1-flash-lite (Flash-lite), and Qwen-3.5-35B-A3B (Qwen).

Fit ↓ / Eval →	GPT-5.4	nano	Opus	Haiku	Pro	Flash-lite	Qwen	Off-diag.
Base accuracy	0.636	0.642	0.733	0.681	0.755	0.561	0.600	
ECE ↓								
GPT-5.4	*0.094*	0.035	0.055	0.079	0.057	0.171	0.145	0.090
nano	0.088	*0.039*	0.160	0.099	0.088	0.153	0.093	0.113
Opus	0.047	0.054	*0.054*	0.097	0.049	0.231	0.087	0.094
Haiku	0.072	0.083	0.086	*0.081*	0.079	0.207	0.161	0.115
Pro	0.112	0.138	0.075	0.167	*0.046*	0.158	0.117	0.128
Flash-lite	0.200	0.170	0.182	0.207	0.165	*0.096*	0.131	0.176
Qwen	0.137	0.118	0.149	0.153	0.082	0.126	*0.087*	0.128
AUROC ↑								
GPT-5.4	*0.685*	0.681	0.728	0.679	0.651	0.708	0.618	0.677
nano	0.742	*0.696*	0.725	0.709	0.553	0.561	0.715	0.667
Opus	0.715	0.686	*0.709*	0.680	0.658	0.670	0.689	0.683
Haiku	0.719	0.690	0.718	*0.648*	0.493	0.516	0.521	0.610
Pro	0.734	0.659	0.699	0.652	*0.690*	0.713	0.701	0.693
Flash-lite	0.717	0.637	0.731	0.637	0.665	*0.697*	0.679	0.678
Qwen	0.730	0.698	0.681	0.694	0.581	0.624	*0.670*	0.668

Table 14: Transfer across tool regimes (GPT-5.4-nano, GAIA). Rows are the source and columns the target. Diagonal cells (italic) are in-domain.

Fit ↓ / Eval →	All tools	Basic tools	No tools
Base accuracy	0.642	0.280	0.064
ECE ↓, *zero-shot*			
All tools	*0.039*	0.326	0.483
Basic tools	0.228	*0.067*	0.079
No tools	0.296	0.073	*0.029*
ECE ↓, *after refitting b*			
All tools	*0.039*	0.137	0.070
Basic tools	0.078	*0.067*	0.038
No tools	0.150	0.054	*0.029*
AUROC ↑, *zero-shot*			
All tools	*0.696*	0.813	0.877
Basic tools	0.705	*0.831*	0.886
No tools	0.682	0.840	*0.899*

F Fidelity of the constructed DAGs

DoG's features depend on a DAG built by an LLM. We therefore check how reliable these DAGs are at three levels: structural validity, stability across construction choices, and semantic correctness judged by a human. Agent trajectories are held fixed throughout.

Structural validity. For each benchmark, we build 15 DAG sets that vary the construction (repeated calls, prompt templates, and DAG builders). Table 15 reports three checks: the output parses into a graph, the graph is acyclic, and the answer node resolves to a valid node. All three checks pass in at least 96.9% of cases.

Table 15: Structural checks of the constructed DAGs (min–max over 15 DAG sets per benchmark).

Check	GPQA	GAIA	HLE
Valid parse	98.5–100%	98.2–100%	99.5–100%
Acyclic graph	99.5–100%	98.2–100%	99.5–100%
Answer node resolves	97.4–100%	98.8–100%	96.9–100%

Stability of the Answer Chain. Free-form nodes cannot be matched one-to-one across DAGs. We therefore map each Answer Chain back to the trajectory steps it comes from, and report the mean pairwise Jaccard similarity between these step sets. We vary one factor at a time: repeated calls (5 runs), the prompt template (original vs. redesigned), and the DAG builder (the agent's own model and five external models). Table 16 shows the results.

Table 16: Stability of the Answer Chain: mean pairwise Jaccard similarity between trajectory-step sets.

Variation	GPQA	GAIA	HLE
Repeated calls (5 runs)	0.927	0.723	0.881
Prompt template (original vs. redesigned)	0.916	0.693	0.850
DAG builder (self + 5 external)	0.794	0.617	0.721

Answer Chains are most stable across repeated calls and least stable across builders. GAIA, which has the longest trajectories, is the least stable benchmark.

Human validation. Structural checks cannot tell whether an edge reflects a real dependency in the trajectory. We therefore ask a human to judge the edges directly. Specifically, one of the authors labeled the edges of 54 tasks (18 per benchmark), blind to correctness labels and DoG confidence. For each benchmark, 9 tasks are *contested*: those with the lowest mean pairwise Jaccard between the Answer Chains (as trajectory-step sets) of the six DAG builders. The other 9 tasks are drawn at random. This gives 176 edges from random tasks and 127 edges from contested tasks.

The annotator judges whether the trajectory record supports each edge, not whether the claims are scientifically correct. Thus, an edge between two wrong claims is still justified if the record shows the dependency. Each edge receives one of four labels. *Justified*: the record supports the dependency. *Wrong relation*: the nodes are related, but the edge type is wrong (e.g., *Supports* where the parent contradicts the child). *Unsupported*: the record shows no such dependency. *Not verifiable*: the shown excerpt is not enough to decide (e.g., the observation is truncated). An edge without a type label is justified if any dependency holds.

Table 17: Human validation of DAG edges (% of edges in each sample).

Sample	Edges	Justified	Wrong relation	Unsupported	Not verifiable
Random	176	88.6	1.1	8.0	2.3
Contested	127	75.6	4.7	14.2	5.5

The annotator judges 88.6% of random edges and 75.6% of contested edges as justified. Contested tasks score lower, as expected, since they are chosen for builder disagreement. Wrong relations are rare (1.1% and 4.7%), so most errors are unsupported links rather than wrong edge types. The annotator also corrects each Answer Chain by hand. Matching the original chains to these corrected chains node by node gives a mean Jaccard similarity of 0.85, 0.89, and 0.89 on GPQA, GAIA, and HLE.

Choice of DAG builder. By default, the agent's own model builds the DAG, so it might simply confirm its own reasoning. To test this, Table 18 keeps the GPT-5.4-nano trajectories fixed and replaces the builder with five other LLMs. Self-construction shows no systematic advantage or disadvantage: the agent's own model ranks between first and fourth of the six builders on every metric. Changing the builder can still improve discrimination. Gemini-3.1-pro gives the highest AUROC and the lowest Brier score on all three benchmarks, although its ECE is higher than that of self-construction. Thus, self-construction is a practical default, and the builder can be replaced when more discrimination is needed.

Table 18: DoG with different DAG builders on fixed GPT-5.4-nano trajectories, fit under the nested cross-validation of Appendix E. **Bold** and underline denote the best and second-best values in each column.

DAG builder	**GPQA**			**GAIA**			**HLE**		
	ECE ↓	Brier ↓	AUROC ↑	ECE ↓	Brier ↓	AUROC ↑	ECE ↓	Brier ↓	AUROC ↑
GPT-5.4-nano (self)	<u>0.053</u>	0.197	0.728	**0.052**	0.195	0.721	0.046	0.160	0.686
GPT-5.4	0.056	<u>0.186</u>	<u>0.763</u>	0.071	0.208	0.689	<u>0.036</u>	0.158	0.707
Claude-opus-4.7	**0.043**	0.189	0.744	<u>0.054</u>	<u>0.192</u>	<u>0.726</u>	0.038	<u>0.151</u>	<u>0.737</u>
Claude-haiku-4.5	0.059	0.211	0.675	0.067	0.198	0.719	**0.033**	0.168	0.628
Gemini-3.1-pro	0.062	**0.154**	**0.828**	0.079	**0.182**	**0.772**	0.061	**0.119**	**0.849**
Gemini-3.1-flash-lite	0.080	0.201	0.717	0.071	0.218	0.594	0.048	0.166	0.637

Overall, the DAGs are not error-free: 8.0% of random edges are unsupported, and our validation uses a single annotator. However, DoG is computed on these same DAGs and still outperforms TRACER in all nine cells and HTC in seven of nine (Table 1). Thus, reconstruction errors do not prevent the DAGs from giving a useful calibration signal. We discuss this as a limitation in Section 5.

G Cost analysis

In this section, we provide a detailed cost-performance analysis of the DoG framework compared to the trajectory-level Verbalized Confidence baseline. The evaluation is conducted on the `GPT-5.4-nano` agent under the `Full Tool` regime across GAIA, GPQA, and HLE datasets.

Cost calculation. Because both the baseline and DoG operate post-hoc on the exact same frozen trajectories, the rollout cost is identical across methods (e.g., $\sim$ \$0.029 per task for GAIA). Therefore, we strictly compare the *calibration overhead*, i.e., the cost of the LLM calls required to estimate confidence. Costs are calculated using the official API pricing for `GPT-5.4-nano`: \$0.20 per 1M input tokens, \$0.02 per 1M cached input tokens, and \$1.25 per 1M output tokens. Crucially, the cost for DoG represents the one-time Stage-1 DAG extraction cost, since the Stage-2 logistic regression inference incurs zero additional LLM API costs. DoG's metrics in Table 19 are the five-run means of Table 1, and its cost is averaged over the same five runs.

Bounded cost vs. length-dependent scaling. As shown in Table 19, the calibration cost for the Verbalized Confidence fluctuates by a factor of $\sim 5.6\times$ (from \$0.0005 on GPQA to \$0.0028 on GAIA) because the entire trajectory must be passed into the prompt. In contrast, DoG's cost stays nearly constant at \$0.0023–\$0.0024 per task across datasets, because the DAG parsing prompt truncates overly long observations and enforces a bounded output schema. Consequently, the relative cost advantage of DoG grows rapidly as the trajectory becomes longer.

Table 19: Cost-performance trade-off between Verbalized Confidence and DoG. The table explicitly details the absolute cost difference (Δ Cost) alongside the resulting absolute and relative performance improvements (Δ). ECE and Brier score are lower-is-better ($\downarrow$). AUROC is higher-is-better ($\uparrow$).

Dataset	Calibration Cost ($/task)			ECE ↓			Brier ↓			AUROC ↑		
	Verb.	DoG	Δ	Verb.	DoG	Δ (%)	Verb.	DoG	Δ	Verb.	DoG	Δ
GAIA	\$0.0028	\$0.0023	**-\$0.0005**	0.150	0.041	**-72.7%**	0.235	0.194	**-0.041**	0.655	0.727	**+0.072**
GPQA	\$0.0005	\$0.0023	**+\$0.0018**	0.282	0.045	**-84.0%**	0.287	0.195	**-0.092**	0.729	0.731	**+0.002**
HLE	\$0.0008	\$0.0024	**+\$0.0016**	0.332	0.056	**-83.1%**	0.290	0.156	**-0.134**	0.637	0.709	**+0.072**

Cost-performance trade-offs. The Δ columns in Table 19 clearly illustrate two distinct regimes:

- **Cost reduction with performance gains on long trajectories (GAIA):** For tasks requiring extensive tool use, DoG costs less than the baseline. This saves \$0.0005 per task while reducing ECE by 72.7% (-0.109), lowering the Brier score, and increasing AUROC by 0.072.
- **Small premium on short trajectories (GPQA / HLE):** On datasets with shorter traces, DoG costs about \$0.0016–\$0.0018 more per task, while reducing ECE by 83.1% to 84.0% and lowering the Brier score.

H Detailed formulation of DAG features

In Section 3.3, we introduce the 8-dimensional feature vector $x \in \mathbb{R}^8$ extracted from the DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here, we provide the exact mathematical formulations and extraction criteria for these features. Let $\mathcal{A}(v_a)$ denote the ancestor set of the answer node v_a (i.e., the Answer Chain), which includes v_a itself. If the answer node does not resolve to a valid node, we treat all nodes as orphans.

H.1 Graph-level features

These features capture the topological geometry and the global state of the agent's exploration.

x_1**: Chain-Restricted Weakest Link.** This feature measures the true informational bottleneck by finding the shortest tool observation strictly within the Answer Chain. Let $\text{obs}(v)$ be the set of observation strings associated with node v, and $|o|$ be the character length of the observation o.

$$x_1 = \min_{v \in \mathcal{A}(v_a)} \min_{o \in \text{obs}(v)} \log(1 + |o|) \tag{3}$$

Note: If the Answer Chain contains no tool observations, it falls back to the trajectory-wide minimum.

x_2**: Orphan Count.** The absolute number of nodes that do not contribute to the final answer. These represent exploratory dead-ends.

$$x_2 = |\mathcal{V}| - |\mathcal{A}(v_a)| \tag{4}$$

x_3**: Orphan Ratio.** The proportion of the graph that consists of orphan nodes, quantifying the agent's disorientation.

$$x_3 = \frac{|\mathcal{V}| - |\mathcal{A}(v_a)|}{|\mathcal{V}|} \tag{5}$$

H.2 Node-level features

These features aggregate the localized epistemic health evaluations across the nodes. We use the indicator function $\mathbf{1}[\cdot]$ to count semantic conditions evaluated by the calibrator LLM.

x_4**: Low-Plausibility Inferences.** The count of inference nodes ($v \in \mathcal{V}_I$) whose internal logical derivation is evaluated as a "leap of faith" or lacks sound mathematical/logical backing.

$$x_4 = \sum_{v \in \mathcal{V}_I} \mathbf{1}[v.\text{plausibility} = \text{low}] \tag{6}$$

x_5**: High-Plausibility Inferences.** The count of inference nodes evaluated as having highly sound independent plausibility.

$$x_5 = \sum_{v \in \mathcal{V}_I} \mathbf{1}[v.\text{plausibility} = \text{high}] \tag{7}$$

x_6**: Hedged Claims.** The count of nodes across the entire DAG where the claim is expressed using hedged or uncertain language (e.g., "might," "appears to be," "probably").

$$x_6 = \sum_{v \in \mathcal{V}} \mathbf{1}[v.\text{hedged} = \text{True}] \tag{8}$$

x_7**: Constraint Satisfaction.** A boolean indicator, judged by the builder LLM, of whether the final answer addresses all constraints in the query.

$$x_7 = \mathbf{1}[\text{all constraints are addressed}] \tag{9}$$

x_8**: Answer Consistency.** A scalar value, judged by the builder LLM, mapping the logical and numerical consistency between the final answer v_a and its supporting ancestors $\mathcal{A}(v_a)$.

$$x_8 = \begin{cases} 1.0 & \text{if consistent} \\ 0.5 & \text{if minor inconsistency} \\ 0.0 & \text{if major inconsistency} \end{cases} \tag{10}$$

I Calibration model optimization and protocols

To compute the final confidence score, DoG employs a lightweight parametric calibrator. This section details the per-cell calibration pools, the loss formulation, and the out-of-fold evaluation protocol used to fit the 9 learnable parameters (8 feature weights $\mathbf{w}$ and 1 intercept b).

Per-cell calibration pools and standardization. We define an evaluation cell as a triple (d, m, r) with dataset $d \in \{\text{GAIA}, \text{GPQA}, \text{HLE}\}$, base LLM m is one of the seven base LLMs in Table 4, and tool regime $r \in \{\text{all-tools}, \text{basic-tools}, \text{notools}\}$. Each cell maintains its own calibration pool $\mathcal{D}_{cal}^{(d,m,r)}$ of pre-collected trajectories, sized by the dataset task count: $N_{\text{GAIA}} = 165$, $N_{\text{GPQA}} = 200$ (filtering in Appendix C), $N_{\text{HLE}} = 200$. Before optimization, raw feature vectors $\mathbf{x}^{(i)} \in \mathbb{R}^8$ are standardized via z-score normalization, $z_j^{(i)} = (x_j^{(i)} - \hat{\mu}_j)/\hat{\sigma}_j$. To prevent leakage during K-fold evaluation, scaler statistics $(\hat{\mu}, \hat{\sigma})$ are fit exclusively on the training folds and applied to the held-out fold.

Model formulation and objective. For each cell the calibrator is a binary logistic regression model optimized by L-BFGS, minimizing the L_2-regularized logistic loss:

$$\mathcal{L}(\theta) = \frac{1}{2C}\|\mathbf{w}\|_2^2 - \sum_{i=1}^{n}\Big[\, y^{(i)} \log c^{(i)} + (1 - y^{(i)}) \log(1 - c^{(i)}) \,\Big] \tag{11}$$

where $c^{(i)} = \sigma(b + \mathbf{w}^\top \mathbf{z}^{(i)})$ and C is the inverse regularization strength. The intercept b is left unregularized so it can absorb the cell-specific base-rate accuracy. We use a maximum of 2,000 iterations, and C depends on the protocol below.

Protocol 1: per-cell settings (main paper). The results in the main paper (Tables 1, 2, 4, and 5, the reliability diagrams, and Tables 6–8) use stratified out-of-fold predictions with the per-cell settings in Table 20. For each fold, the scaler and model are fit on the other folds and predict the held-out fold. With several seed repeats, the predictions are averaged over repeats, and each task is still held out once per repeat. Some of these settings were selected on the cell's own out-of-fold metrics. The HLE calibrator is fit on the pooled GPQA and HLE trajectories and scored on HLE only. The general calibrator of Table 9 and the examples in Appendix L use one calibrator fit on the pooled GAIA and GPQA trajectories with $C = 0.5$.

Table 20: Per-cell settings of Protocol 1. The Claude and Gemini models are Claude-opus-4.7, Claude-haiku-4.5, Gemini-3.1-pro, and Gemini-3.1-flash-lite. All cells use the agent's own model as the DAG builder.

Cell	**Training pool**	**Folds × repeats**	C
GPT-5.4-nano, GAIA, all tools	GAIA	20 × 20	0.07
GPT-5.4-nano, GPQA, all tools	GPQA	5 × 1	0.1
GPT-5.4-nano, HLE, all tools	GPQA + HLE	5 × 1	0.3
GPT-5.4, GAIA	GAIA	8 × 20	0.07
Qwen-3.5-35B-A3B, GAIA	GAIA	20 × 20	0.2
Claude and Gemini models, GAIA	GAIA	5 × 1	0.5
GPT-5.4-nano, GAIA, basic or no tools	GAIA	5 × 1	0.5
General calibrator	GAIA + GPQA	5 × 1	0.5

Protocol 2: nested cross-validation (appendix analyses). The analyses in Appendices D, E, and F use nested cross-validation. The outer loop is a stratified 10-fold split repeated with two seeds, and the out-of-fold predictions are averaged over the two repeats. Inside each outer training split, C is chosen from $\{0.02, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.3, 0.5, 1.0\}$ by 5-fold cross-validation with the lowest log loss, and the model is refit on the whole training split with that C. Each cell is fit on its own trajectories only, and no reported prediction influences the choice of C. Over the five DAG regenerations, the selected C has a median of 0.07 on GAIA and 0.1 on GPQA and HLE. DoG's in-domain ECE, Brier score, and AUROC under this protocol are 0.053, 0.197, and 0.728 on GPQA, 0.052, 0.195, and 0.721 on GAIA, and 0.046, 0.160, and 0.686 on HLE.

For cells with a non-reference base LLM, the Stage-1 DAG is inherently rebuilt with the judge fixed to the same model class as the agent (self-prediction); the Stage-2 calibrator is then fit per-cell exactly as above. In the DAG-builder analysis (Table 18), we instead keep the GPT-5.4-nano trajectories fixed and replace the builder with other LLMs.

J Agent setup and implementation details

To generate the execution trajectories for the DoG calibration framework, we develop a robust, reproducible, tool-augmented agent harness. The generation process is structurally decoupled from the DAG construction and relies on a LangGraph-based workflow. This section details the agent architecture, tool environment, base models, and dataset evaluation protocols.

Agent architecture and execution loop. The agent operates using a bounded ReAct-style loop [34]. At each step, the model is provided with the system prompt, the input query, the execution history, and the available tool schemas. It must emit either a native tool call or a synthetically parsed JSON tool call. To prevent infinite loops and manage API costs, we enforce a strict budget of `MAX_TOOL_STEPS = 20`. If the agent exhausts this budget without yielding an answer, a `provide_final_answer` fallback forces a terminal tool call. This ensures every task ultimately yields a committed answer. The entire trace consisting of Thoughts, Actions, Observations, and step latencies, is recorded as the trajectory τ.

Tool environment and regimes. To rigorously test calibration across varying degrees of external grounding, the agent is equipped with a suite of tools, which are dynamically toggled based on the experimental regime:

- **Web search:** Retrieves the top organic search results and answer-box snippets via the Serper API.
- **Visit webpage:** A proxy-based crawler that fetches and converts main-content HTML, JS-rendered pages, and PDFs into Markdown.
- **Execute `Python`:** An in-browser Pyodide sandbox pre-loaded with scientific libraries (e.g., `numpy`, `pandas`, `sympy`) to execute data processing and calculations.
- **Inspect image:** A vision sub-call routed to a local Qwen-3.5-35B-A3B model, enabling text-only reasoning models to ground on image inputs without consuming native vision-model tokens.
- **Transcribe Audio:** An audio parsing tool utilizing a local Whisper instance.

In our experiments, we utilize three tool regimes: (1) **All-tools**: Access to the complete toolset; (2) **Basic-tools**: Restricted to Web Search and Visit Webpage to simulate standard web-browsing agents; and (3) **No-tools**: A pure-knowledge baseline restricted solely to the internal knowledge.

Base models and reasoning extraction. To evaluate DoG across diverse frontier models, we integrate multiple LLM families via their respective APIs. Our pool includes **OpenAI** (GPT-5.4, GPT-5.4-nano), **Anthropic** (Claude-opus-4.7, Claude-haiku-4.5), **Google** (Gemini-3.1-pro, Gemini-3.1-flash-lite), and **Qwen** (Qwen-3.5-35B-A3B). Crucially, modern frontier models employ diverse mechanisms for internal reasoning (e.g., OpenAI Responses API summaries, Anthropic's adaptive thinking blocks, and Gemini's reasoning content). Our harness standardizes these diverse surfaces, extracting them uniformly into the trajectory's `Thought` field. This ensures that DoG has faithful access to the genuine internal reasoning traces, preserving the integrity of the DAG extraction process across different model families.

Dataset processing and evaluation protocol. To guarantee reproducibility, task sampling across datasets was executed using a seeded Fisher-Yates shuffle (seed = 42). For datasets requiring open-ended evaluation (e.g., HLE and the open-ended variant of GPQA), answers were scored using a dedicated LLM judge (GPT-5.4-nano) guided by official evaluation prompts. For open-ended GPQA, we remove the 126 stems that depend on the hidden choices, following the procedure in Appendix C. For standardized datasets (e.g., GAIA), we utilize strict normalized exact-match scoring. To ensure robustness against transient API failures, all network calls were wrapped in an exponential backoff protocol.

K Detailed implementation of baseline methods

We compare DoG against two distinct families of baselines: static LLM baselines (which evaluate the trajectory as a flat textual context) and agent-specific baselines (which exploit step-wise structure but still aggregate signals sequentially). To ensure a fair and controlled comparison, all applicable baselines are evaluated post-hoc and provided with the identical fixed trajectory τ as context.

K.1 P(True) [13]

We adapt the prompt of Kadavath et al. [13] $P(\text{True})$ for agent trajectories, requiring a single LLM call per task. Each task consists of a question q_i, trajectory τ_i, and answer a_i. The judge is the same model used for generation.

Prompting. The prompt structure is as follows:

```
Question: {q_i}
Reasoning: {tau_i}
Here are some brainstormed ideas: {ideas}
Possible Answer: {a_i}
Is the possible answer: (A) True (B) False
The possible answer is:
```

The `Reasoning` block is omitted if τ_i is empty. Trajectory steps are formatted as: `Step <i>: Thought, Tool, Input, Observation`. Up to five unique `final_answer` candidates are extracted from τ_i as *ideas*. To ensure output reliability, the system instruction is set to: *"Respond with exactly one word: either True or False."* We do not use few-shot demonstrations to prevent cross-task contamination.

Confidence estimation. We utilize the OpenAI API with `top_logprobs=5`. Confidence $P(\text{True})$ is calculated by extracting the log-probabilities $\ell(t)$ for tokens corresponding to True/False branches (T_+, T_-) at the first token position containing such tokens:

$$\text{conf}_i = \frac{\sum_{t \in T_+} e^{\ell(t)}}{\sum_{t \in T_+} e^{\ell(t)} + \sum_{t \in T_-} e^{\ell(t)}}$$

Tokens are normalized by removing leading punctuation and whitespace and converting to lowercase. If no valid branch token is identified, the result is marked as missing.

K.2 Verbalized confidence [29]

To evaluate agent confidence, we adapt the Verb-1S top-k method [29] to an agent-trajectory setting. We set $k = 4$ to elicit the model's top guesses and their associated probabilities in a single forward pass.

Methodology. For each task i, the judge model (identical to the agent m_i) is provided with the original question q_i, the reasoning trajectory τ_i, and the committed answer a_i. The model is instructed to output its confidence in a_i as P_1, while also providing three alternative guesses (G_2, G_3, G_4) and their respective probabilities. By anchoring G_1 to the original answer a_i, P_1 serves as the model's calibrated confidence for that specific trajectory.

Prompting. We use a zero-shot template that forces a strict output format:

```
G1: <answer verbatim>
P1: <probability>
G2: <alternative>
P2: <probability> ...
```

The trajectory τ_i is provided as context evidence to ensure the judge evaluates the same reasoning process that produced the original answer. The prompt explicitly prohibits chain-of-thought, preamble, or any extra commentary to ensure parseability.

Confidence Extraction. We parse the output using regex to extract the value P_1. Responses provided as percentages are rescaled to $[0, 1]$ and clipped; values that fail to parse are treated as missing and excluded from evaluation.

Hyperparameters.

- **Model:** Same as the agent (m_i).
- **Sampling:** $k = 4$, single forward pass per task, temperature set to model defaults.
- **Budget:** Exactly one LLM call per task.

K.3 Linguistic Hedging [27]

We adapt the *Linguistic Hedging* method proposed by Tao et al. [27] to the agent-trajectory setting. Linguistic Hedging is a single-call, black-box, post-hoc procedure: an LLM judge reads the model's response, detects hedging language (e.g., "probably", "might be", "I'm not sure…"), and outputs a scalar confidence score $\hat{c} \in [0, 1]$. This approach requires no access to logits or token probabilities, and involves no training or fine-tuning. By consuming exactly one judge call per task, the method remains well within our budget of ≤ 2 LLM calls per task.

Response construction. Rather than sampling a fresh response from the agent using the MMLU-Pro prompt as in the original paper [27, Appx. F.1], we utilize the existing recorded trajectory $\tau = (s_1, \ldots, s_T)$ and the final answer a for each task. Each step is defined as $s_t = (\mathit{thought}_t, \mathit{tool}_t, \mathit{input}_t, \mathit{obs}_t)$. The input provided to the judge, $R(\tau, a)$, is constructed by concatenating the parsed steps and the final answer:

$$R(\tau, a) = \texttt{strip}\left(\left[\bigoplus_{t=1}^{T} \phi(s_t)\right] \,\|\, \text{"Final answer: "}\, a\right),$$

where $\|$ denotes string concatenation and $\phi(s_t)$ formats step t as follows:

```
Step t --- Thought: <thought>
  Tool: <toolName>(<input>)
  Observation: <observation>
```

The `Tool` and `Observation` fields are omitted if they are empty. Because the step-by-step thought stream naturally contains the hedging language during agentic execution, we use it as the primary basis for Linguistic Hedging. Auxiliary fields such as `reasoningTrace`, `dag`, and `coarseDag` are intentionally excluded.

Numerical-confidence stripping. The $\texttt{strip}(\cdot)$ function applies a set of nine regular expressions [adopted verbatim from 27, Appx. F.3] to the assembled response. This step removes any explicit numerical confidence expressions (e.g., "Confidence: 73%", "Confidence (0-100): [80]", "Confidence Level: 0.9") before the text is passed to the judge. This effectively prevents the judge from being biased by previously verbalized numerical confidence scores leaked from within the trajectory.

Judge prompt. The judge receives the question q and the stripped response $R(\tau, a)$ using the exact Linguistic Hedging prompt from Tao et al. [27, Appx. F.2], including all four few-shot exemplars. The prompt instructs the judge to focus solely on the provided answer and any hedging modifiers, to assign a score of 0.0 to evasive or abstained answers, and to conclude with a single line: "`Confidence score:` p", where $p \in [0, 1]$.

Parsing. We parse the judge's output using the following regular expression:

```
[Cc]onfidence\s*score\s*[:=]\s*([0-9]*\.?[0-9]+).
```

Parsed values v are clipped to the $[0, 1]$ interval; any $v > 1$ is treated as a percentage and divided by 100. Parse failures, empty responses, and API exceptions result in a missing value ($\hat{c} = \perp$) and are dropped during evaluation, rather than being silently mapped to 0.

Judge model. While the original paper employed an external judge model (LLaMA-4-Maverick-17B-128E-Instruct), we use GPT-5.4-nano, the exact same model that generated the agent trajectories. This aligns with the same-model self-evaluation setup used for other baseline methods in our study (e.g., P-TRUE, VERBALIZED-CONFIDENCE).

K.4 TRACER [28]

For the TRACER baseline, we tune only the hyperparameters defined by the original TRACER formulation, together with a single binary polarity choice for the deterministic risk-to-confidence mapping. All other components including (i) the per-step risk formula $\text{risk}_i = \max(U_i, \alpha \cdot \text{Da}_i, \beta \cdot \text{Do}_i^{\text{agent}}, \gamma \cdot \text{Do}_i^{\text{user}})$, (ii) the top-$k$/max ensemble aggregation, (iii) the deterministic clip-to-$[0, 1]$ mapping, and (iv) the embedding-based definitions of Da and Do are kept fixed.

Because the trajectories are single-turn and contain no token log-probs, U_i and Do^{user} cannot be recovered, and we accordingly fix $\gamma = 0$ throughout the sweeps. Embeddings used inside Da and Do^{agent} are produced by `text-embedding-3-large` with inputs truncated to 8,000 tokens. Observations that are longer than this limit are kept tail-aligned.

The grid is searched over the following ranges:

- $\alpha \in \{0,\ 0.1,\ 0.25,\ 0.5,\ 0.75,\ 1.0,\ 1.25,\ 1.5,\ 1.75,\ 2.0,\ 2.5,\ 3.0,\ 4.0,\ 5.0,\ 7.0,\ 10.0\}$
- $\beta \in \{0,\ 0.1,\ 0.25,\ 0.4,\ 0.5,\ 0.6,\ 0.75,\ 0.9,\ 1.0,\ 1.25,\ 1.5,\ 2.0,\ 3.0,\ 5.0\}$
- top-k percentile $\in \{0.05,\ 0.1,\ 0.15,\ 0.2,\ 0.25,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1.0\}$
- ensemble weight $w \in \{0,\ 0.05,\ 0.1,\ 0.15,\ 0.2,\ 0.25,\ 0.3,\ 0.4,\ 0.5,\ 0.7,\ 1.0\}$

excluding the degenerate combination $(\alpha, \beta) = (0, 0)$. A single binary polarity bit selects between the canonical TRACER mapping $c = 1 - \text{clip}(\text{score}, 0, 1)$ and its reversed form $c = \text{clip}(\text{score}, 0, 1)$, giving roughly 63,778 candidate configurations per search.

Per-Dataset optimized hyperparameters. For each target benchmark $D \in \{\text{GAIA}, \text{GPQA}, \text{HLE}\}$, hyperparameters are selected by exhaustive grid search to minimize the Brier score on only the samples of D. The configuration with the lowest Brier score on the target benchmark is selected, and the same configuration is then applied (without retuning) to all four evaluation sets reported below (GAIA, GPQA, HLE, and the pooled set). The selected hyperparameters for each dataset are summarized in Table 21.

Table 21: Per-dataset optimized hyperparameters for TRACER.

Target	α	β	γ	**top-k**	**ensemble w**	**polarity (mapping)**
GAIA	1.25	1.0	0	1.0	0.2	$c = \text{clip}(\text{score}, 0, 1)$
GPQA	0.0	0.75	0	1.0	0.0	$c = \text{clip}(\text{score}, 0, 1)$
HLE	0.0	0.5	0	0.8	0.0	$c = \text{clip}(\text{score}, 0, 1)$

Pooled optimized hyperparameters. In addition to the per-dataset tuning, we also evaluate all configurations on the pooled set of trajectories to find a globally robust setting, without using dataset identity, cross-validation, or per-dataset fitting. Here, the selection objective is the pooled Brier score across all three benchmarks. The globally chosen configuration is $\alpha = 0.25$, $\beta = 0.75$, $\gamma = 0$, top-$k = 1.0$, $w = 0.5$, with the reversed polarity $c = \text{clip}(\text{score}, 0, 1)$.

K.5 Holistic Trajectory Calibration (HTC) [37]

In this study, we adopt the results from the HTC framework as our performance baseline, categorizing them into two distinct approaches: a dataset-specific optimized baseline and a general confidence estimator baseline.

First, as detailed in the original paper, the *HTC-Reduced* variant serves as our dataset-specific baseline. This baseline represents the performance achieved when weights are individually optimized for each dataset using Lasso regularization to select a compact subset of features. This approach effectively denoises spurious signals, leading to superior calibration in specific high-stakes environments. The performance metrics for these optimized results are summarized in Table 22.

Furthermore, to establish a baseline for a general confidence estimator that utilizes an identical set of weights across different tasks, we analyze the weight-specific calibration performance graphs provided in Tables 7-10 of the original paper. For this general-purpose baseline, we select the

weight configuration that was optimized on the SimpleQA dataset using the GPT-4o model, as it demonstrated the best generalization capability. This specific configuration achieved the lowest average Brier of 0.146 across the GAIA, GPQA, and HLE benchmarks. By adopting this as our general baseline, we ensure a more rigorous point of comparison for cross-domain reliability without dataset-specific tuning.

Table 22: Performance of the HTC-Reduced (Best Performing Variant) from Zhang et al. [37], provided in Table 5 of Appendix A.3.1.

Dataset	ECE ↓	Brier Score ↓	AUROC ↑
GAIA	0.1420	0.2332	0.6860
GPQA	0.1022	0.2134	0.7060
HLE	0.0305	0.0897	0.6439

These figures represent the mean performance of the HTC-Reduced variant. As highlighted in the original study, this model achieves the most robust calibration by isolating universal uncertainty signals while mitigating overfitting in small-data regimes.

L Qualitative results

To illustrate how DoG overcomes the limitations of linear trajectory evaluation, we present visual case studies from the GAIA benchmark (`GPT-5.4-nano`, *Full Tool*). In the DAG visualizations, valid Answer Chains are highlighted with a glowing border, while exploratory dead-ends are isolated as borderless orphans. Edge styles denote epistemic relationships: solid lines for *derives*, dotted lines for *supports*, and dashed red lines for *refutes*.

L.1 Exposing structural flaws in fluent trajectories

The most significant advantage of DoG is its ability to detect logical disconnections in trajectories that appear superficially flawless (i.e., zero execution errors, successful tool retrievals).

Case 1: The Missing Epistemic Link

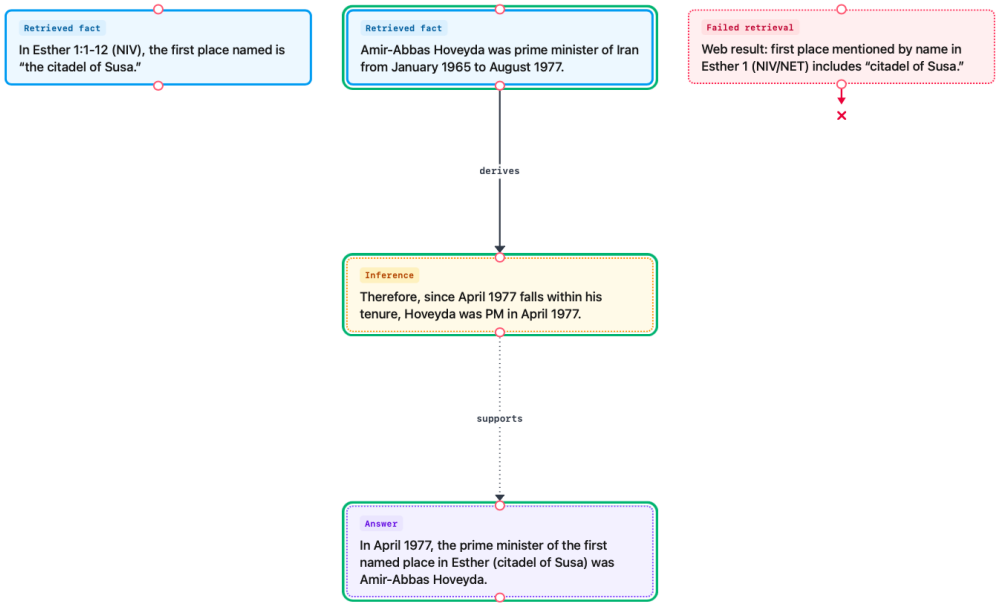


Figure 5: DoG isolates the critical premise as an orphan node, exposing the ungrounded answer.

- **Query:** "In April of 1977, who was the Prime Minister of the first place mentioned by name in the Book of Esther?"
- **Scores:** $c_{verb} = 0.70$ vs. $c_{DoG} = 0.39$
- **Analysis:** The raw trajectory reveals a mix of successful web searches and a failed webpage visit across 5 steps. Evaluating this flat sequence, the baseline struggles to isolate the noise and is heavily anchored by the successful retrieval of Iran's Prime Minister, yielding a confident 0.70. However, DoG's topological extraction reveals that the fundamental premise, mapping the "first place mentioned" to Susa (Iran) rather than India, was never logically connected to the Answer Chain (remaining as an orphan node). Recognizing this missing link, DoG appropriately penalizes the confidence.

Case 2: High-Quality Facts on a Weak Premise

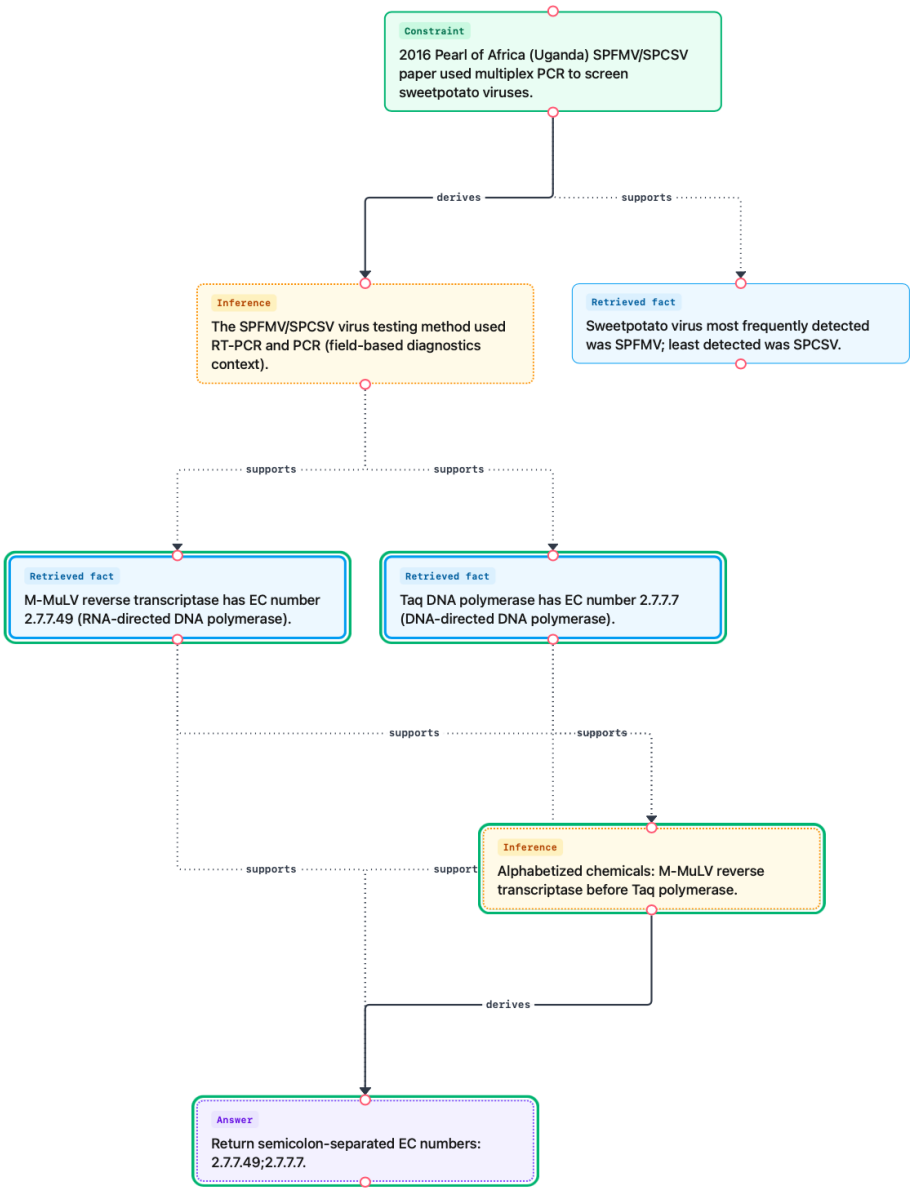


Figure 6: A flawed root inference invalidates the high-quality retrievals built upon it.

- **Query:** "What are the EC numbers of the two most commonly used chemicals for the virus testing method in the paper about SPFMV and SPCSV...?"
- **Scores:** $c_{verb} = 0.86$ vs. $c_{DoG} = 0.14$
- **Analysis:** The agent executes a rigorous, 7-step deep retrieval process across multiple scientific documents without a single execution error. Misled by this apparent rigor, the baseline reports an overconfident 0.86. DoG, however, maps the epistemic dependencies and identifies that this retrieval chain is built upon a single, unverified root inference (assuming the testing method without verifying the paper). DoG therefore lowers the confidence, since a deep chain built on an unverified root is only as reliable as that root.

L.2 Rewarding clean chains

DoG does not merely penalize; it also accurately boosts confidence when a multi-step reasoning trace is structurally sound, countering the agent's natural hesitation on complex tasks.

Case 3: Clean Multi-Step Arithmetic

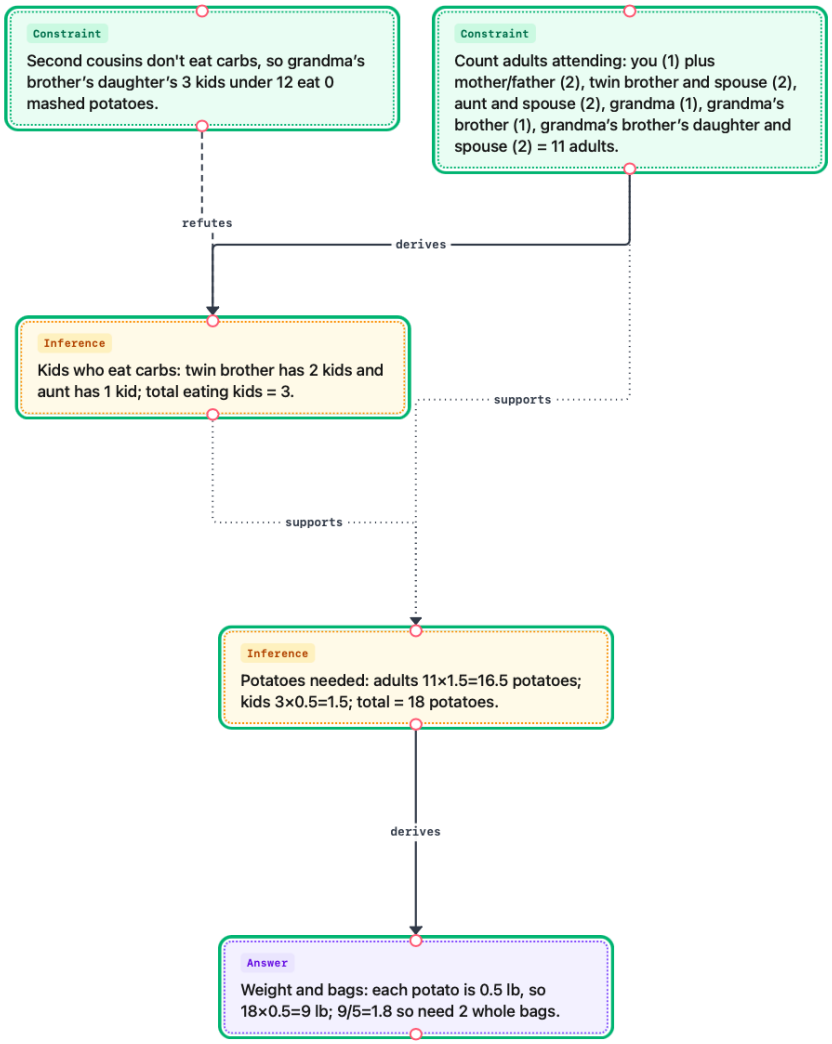


Figure 7: A clean, linear DAG with zero orphans strongly grounds the arithmetic reasoning.

- **Query:** [Lengthy word problem regarding family reunion mashed potatoes.]
- **Scores:** $c_{verb} = 0.12$ vs. $c_{DoG} = 0.80$
- **Analysis:** The trajectory for this task consists of only a single `final_answer` step with zero external tool invocations. Intimidated by the lengthy internal arithmetic and the lack of external grounding, the baseline agent reports a heavily underconfident 0.12. DoG parses this single block of pure reasoning text into a well-structured 5-node graph. It verifies that every internal inference securely belongs to the load-bearing Answer Chain with zero orphans. Rewarding this structured internal logic, DoG accurately restores the confidence to 0.80.

M Prompts

This appendix provides the exact system instructions and user prompts used across the two phases of our framework: (1) generating the sequential agent trajectories, and (2) extracting the DAG of Grounds (DoG) for confidence calibration.

M.1 Phase 1: Agent Trajectory Generation

To ensure reproducibility and rigorously evaluate the agent's generative reasoning, we utilize a standard ReAct-style main agent. The agent is strictly instructed to avoid looking up benchmark-specific answers (preventing data contamination) and is provided with dataset-specific answer formatting rules.

System Prompt for the Main Agent. The base structure of the system prompt provided to the agent is as follows:

System Prompt: Main Agent

```
You are an expert assistant who can solve any task using tool calls.

{DATASET_SPECIFIC_ANSWER_RULES}

You only have access to these tools:
- web_search: Search the web for current information. Use when you need
up-to-date facts, statistics, or verification of claims.
  Takes inputs: {"query": {"type": "string", "description": "Search query"}}
  Returns an output of type: string
- visit_webpage: Visits a webpage at the given URL and returns its full content
as markdown. Use this when a search result URL looks directly relevant - visit
it immediately to read the full content rather than doing more searches.
  Takes inputs: {"url": {"type": "string", "description": "The URL of the
webpage to visit"}}
  Returns an output of type: string
- execute_python: Execute Python code in a sandboxed Pyodide runtime.
Preloaded: numpy, pandas, scipy, sympy. Auto-installed on literal `import
X`: pypdf, openpyxl, bs4, lxml, docx, pptx, PIL, matplotlib, networkx, sklearn,
nltk, Bio, xlrd, requests. Task attachments are at /attachments/<name>. NOT
available: `subprocess`, `ssl`. For HTTP(S) requests use the preloaded
async helpers: `await fetch_text(url)`, `await fetch_bytes(url)`, `await
fetch_json(url)`. Top-level await is supported. Prefer this tool over
approximating computation from search snippets.
  Takes inputs: {"code": {"type": "string", "description": "Python code to
execute. Use print() to emit output."}}
  Returns an output of type: string
- inspect_image: Inspect an image and answer a question about it using a vision
model. Use when the task mentions an image attachment or a diagram is needed to
answer.
  Takes inputs: {"image": {"type": "string", "description": "http(s)
URL or ... URI"}, "question": {"type": "string",
"description": "What to ask about the image"}}
  Returns an output of type: string
- final_answer: Provides a final answer to the given problem. You must include
your complete reasoning trajectory.
  Takes inputs: {"reasoning": {"type": "string", "description": "Your
complete chain of thought - what you knew, what you inferred, how you connected
facts, any intermediate conclusions, and how it all led to the answer. Include
reasoning steps that did not involve tool calls."}, "answer": {"type":
"string", "description": "The final answer to the problem"}}
  Returns an output of type: string

Here are the rules you should always follow to solve your task:
1. ALWAYS provide a tool call, else you will fail.
2. Always use the right arguments for the tools. Never use variable names as
the action arguments, use the value instead.
```

```
3.  Call a tool only when needed:  try to solve the task yourself first.  If
no external information is needed, call final_answer directly.  If you found
relevant information in a search result, visit that URL with visit_webpage
before searching again.
4.  Never re-do a tool call that you previously did with the exact same
parameters.
5.  When calling final_answer, the reasoning field must contain your complete
reasoning trajectory - not just what you searched for, but your full chain of
thought:  what you already knew, what assumptions you made, what you inferred
from each piece of information, how you connected different facts together, any
intermediate conclusions you drew, and how all of this led to the final answer.
Reconstruct the entire reasoning process, including steps that did not involve
tool calls.
6.  ALWAYS express your action as a tool call.  NEVER write your final
answer as plain text.  If you have your final answer, you MUST call
final_answer({"answer":  "<your answer>", "reasoning":  "<your reasoning>"}).
Plain-text replies are not accepted.
7.  Reason substantively before every action.  Treat each step as a decision
that must be justified.  Before any tool call, work through:
  - What you already know for certain vs.  what is still uncertain.
  - The candidate hypotheses or answers you are currently considering, and which
piece of evidence would distinguish between them.
  - Why this specific tool call (with these specific arguments) is the most
informative next step, and what result would confirm or falsify each hypothesis.
  - Whether the evidence gathered so far is already sufficient to commit to an
answer, or whether one more check is warranted.
Do this reasoning thoroughly on every turn, even for seemingly obvious
continuations - do not skip reflection just because a next step looks natural.
Shallow or missing reasoning is a failure mode.
8.  Do not look up the benchmark itself.  You must solve the question from
primary sources, not from leaked answer keys.  The following are forbidden and
count as failure regardless of correctness:
  - Searching for the verbatim question text, or distinctive phrases from it, in
order to locate the originating dataset.
  - Visiting dataset mirrors, leaderboards, or evaluation dumps -- including but
not limited to:  huggingface.co/datasets/* pages that contain the question,
paperswithcode.com, patronus-ai/trail-benchmark, any repo or page whose
title contains "gpqa", "gaia", "hle", "mmlu", "benchmark", "eval", "diamond",
"validation set", "test split", or similar dataset identifiers.
  - Reading any page that displays a "ground truth", "solution", "correct
answer", or labeled answer column alongside the question.
If a search result looks like a benchmark mirror or solution dump (dataset
cards, HuggingFace dataset viewers with the question rendered as a row, GitHub
dumps of eval data), do NOT visit it - pick a different result, or refine the
query toward primary sources (papers, official docs, Wikipedia, source code,
the actual subject matter).  If you accidentally land on such a page, ignore
its labeled answers and rely only on independently-derivable facts (e.g.  a
referenced paper) to continue.
```

Dataset-Specific Answer Rules. Depending on the benchmark being evaluated, the following rules were injected into the `{DATASET_SPECIFIC_ANSWER_RULES}` slot:

- **GAIA and GPQA-open:** *"The answer must be a number, OR as few words as possible, OR a comma-separated list of numbers and/or strings. No articles (a, an, the), no abbreviations..."*
- **HLE-Verified:** *"Your response should be in the following format:*
  Explanation: {your explanation}
  Answer: {your chosen answer}"

User Message and Attachment Hints. Along with the system prompt, the first-turn user message strictly wraps the raw question text. If the task includes file attachments (e.g., in GAIA), an automatically generated hint is appended to guide the agent on the appropriate tool and code snippet to open the file:

Format: User Message with Attachment Hints

```
{question_text}

This task has the following attachments:
- {filename1} → {hint1}
- {filename2} → {hint2}
```

For instance, a PDF attachment generates the hint: `execute_python: from pypdf import PdfReader; text = "".join(p.extract_text() for p in PdfReader("/attachments/{name}").pages); print(text).`

Vision Sub-call Prompt (`inspect_image`). When the main agent invokes the vision tool, a secondary vision-capable LLM (Qwen 3.5 35B-A3B) is queried using the following prompt, ensuring text-only models can ground reasoning on image inputs without burning native vision tokens:

Prompt: Vision Sub-call

```
[System] You are a vision assistant. Inspect the image and answer the user's
question concisely. If the image contains text, transcribe it verbatim when
relevant. Reply with plain text only.

[User] {question text}
{image content block}
```

M.2 Phase 2: DoG Stage-1 DAG Extraction

Once the main agent yields a completed trajectory, a secondary calibrator LLM (e.g., `gpt-5.4-nano` for self-prediction) parses the trajectory into the epistemic DAG structure in a strict zero-shot, single-call process.

Trajectory Serialization and Truncation. To ensure the calibration cost remains robust and bounded regardless of the agent's exploration length, the trajectory is serialized with strict truncation logic. Observations are truncated at 1,500 characters, and tool inputs at 600 characters. The question is truncated at 4,000 characters, and the agent's reasoning trace at 3,000 characters:

Format: Trajectory Serialization

```
Step {step_index}
Tool: {tool_name}
Input: {tool_input} [Truncated if > 600 chars]
Observation: {tool_observation} [Truncated if > 1500 chars]
```

DAG Extraction Prompt. The calibrator LLM is instructed to output a strict JSON schema via the following prompt. The calibrator uses only the fields in Appendix H; other outputs, such as the support strength and its caps, are not used but kept as part of the prompt used for all results:

System & User Prompt: DoG DAG Extraction

```
[System] You convert agent trajectories into structured ground-DAGs that are
used later for confidence calibration. Output strict JSON only.

[User] You are given an agent trajectory that attempted to answer a question.
Convert it into a directed acyclic graph (DAG) of grounds whose terminal node
is the agent's final answer. PRESERVE failure signals - keep failed retrievals,
hedging, retries, dead ends.

# Part A - DAG of grounds
Node types: "constraint" / "retrieved_fact" / "failed_retrieval" / "inference"
/ "answer"

Per-node fields (REQUIRED):
```

```
- id, claim (≤ 25 words), node_type, parents
- "in_answer_chain": bool - TRUE iff there is a directed path of
supports/derives_from edges from this node to the answer node. Nodes
explored but NOT used to derive the answer are orphans and must have
in_answer_chain=false.

Top-level fields (REQUIRED):
- n_orphan_nodes: count of nodes with in_answer_chain=false.
- chain_min_evidence_quality: among nodes IN the answer chain, the LOWEST
evidence_quality.
- chain_has_low_plausibility_inference: TRUE if any inference node in the chain
has independent_plausibility="low".

# Part B - Multi-candidate enumeration with chain-bounded support_strength
Enumerate up to 4 PLAUSIBLE answer candidates. For each candidate emit its
"support_strength" (bet-calibrated probability in [0, 1]).

NEW RULE - CHAIN BOUND:
The agent's candidate's support_strength MUST NOT exceed what the chain
weakest-link supports:
- If chain_min_evidence_quality is "none" → cap at 0.40.
- If chain_min_evidence_quality is "low" → cap at 0.55.
- If chain_has_low_plausibility_inference is true → multiply by 0.75 after cap.
- If n_orphan_nodes ≥ 0.5× total_nodes → multiply by 0.85.

Question: {task}
Agent's final answer: {answer}
Trajectory: <trajectory> {trajectory} </trajectory>
Output the JSON now, nothing else.
```

N Broader impacts

DoG aims to improve the reliability of LLM agents by producing better-calibrated confidence estimates. Better calibration can support safer deployment of agents in high-stakes settings, as users and downstream systems can defer to humans or fall back to safer actions when the agent reports low confidence.

A potential negative impact is overreliance: even a well-calibrated estimate can be wrong on individual queries, and treating any single confidence value as a hard guarantee may give a false sense of safety. We therefore recommend using DoG as one signal alongside other reliability checks rather than as a stand-alone guarantee.

This work uses public benchmarks and standard LLM APIs and does not introduce new datasets or models with notable misuse risk.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state the proposed method (DoG), the black-box and post-hoc setting, and the empirical scope. The claims are mainly supported by Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 discusses three limitations (errors in the LLM-built DAGs, the need for labeled target data to keep absolute calibration under distribution shift, and the overhead and latency of one extra post-hoc LLM call per trajectory) and lists future directions.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 describes the agent harness, dataset splits, evaluation protocol, and baselines, and Appendices J and K provide prompts, hyperparameters, and implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: No code release during the review stage. We will release source code upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 4 and Appendix J report details such as the base LLMs, prompts, tool interface, dataset splits, and baselines.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: DoG results in Table 1 are means over five independent DAG-extraction runs, with standard deviations reported in Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: API cost for DoG and all baselines is reported in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work uses public benchmarks and standard LLM APIs, does not involve human-subject data collection, and conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix N discusses both the potential positive impacts (more reliable LLM agents in high-stakes deployments through better calibration) and the negative impacts (e.g., overreliance on a single confidence score) of this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk pretrained model, image generator, or scraped dataset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The benchmarks (GAIA, GPQA, HLE-Verified) and base LLMs used in the experiments are publicly available and properly cited in Section 4 and the Appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The interactive diagnostic tool released is documented in Appendix A.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or human-subject studies. The DAG edges in Appendix F were annotated by one of the authors, following the codebook described there.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research; the only human annotation was done by one of the authors (Appendix F).

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM agents are the subject of calibration and an LLM is used to extract the DAG from each trajectory through standard zero-shot prompting, but LLMs are not used as a novel component of the core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.