CONFIDENCE REASONING GRAPHS: STRUCTURED CONFIDENCE ESTIMATION FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

In consequential domains, aggregate agent success rates do not establish whether an individual execution can be trusted. Estimating confidence requires integrating heterogeneous evidence across interdependent trajectory steps, often without token-level probabilities, repeated roll-outs, or gold trajectories. We introduce **Confidence Reasoning Graphs (CRGs)**, an inference-time framework that estimates the probability an agent accomplished its task from a single trajectory, without privileged model access or training data. Rather than compressing an execution into one holistic judgment, a CRG recursively decomposes the claim that the task was accomplished into contextualized sub-claims, and grounds each in evidence from specific trajectory steps as supporting it, undermining it, or leaving it unverified. Confidence is then estimated for each leaf claim and aggregated into the root. Across three agentic benchmarks, three backbone models, and three frameworks, we find that CRG produces better-calibrated confidence than verbalized, sampling-based, and white-box surrogate baselines, and supports better risk-aware decisions. Our results also illustrate why calibration should not be evaluated in isolation: a surrogate baseline can achieve low calibration error while providing near-chance discrimination. Ablations attribute the gains to leaf-level confidence estimation and aggregation rather than graph construction alone. Finally, the resulting graph exposes the claims and trajectory evidence behind each estimate, allowing a confidence score to be audited rather than taken on trust.

1 INTRODUCTION

Increasingly, consequential real-world tasks are being delegated to LLM agents: autonomous systems that combine LLM reasoning with tools, memory, and skills to solve complex, multi-step tasks through interaction with an environment. Such agents are often used in domains where their decisions have real-world consequences, and users must correspondingly make inference-time decisions about whether to trust their output. An agent's average success rate offers little guidance for this decision: it does not reveal whether a particular output is correct, and checking that output can be difficult because it results from a chain of decisions, tool calls, and observations in which one mistake can affect everything that follows. What is needed is calibrated confidence for agents.

While confidence estimation for LLMs is well-studied (Geng et al., 2024), the agentic setting poses new challenges. Confidence in a single LLM output can often be estimated from sequence log-probabilities or from agreement across repeated samples (Wang et al., 2023). An agent trajectory, in contrast, is a heterogeneous trace that interleaves LLM-generated text, tool calls, skill, memory use, and observations from the environment. Estimating confidence in task success therefore requires reasoning jointly over a long trajectory with different sources of uncertainty: from the model, retrieval, tool failures, or environmental stochasticity (Liu et al., 2024; Zhao et al., 2025).

Real-world agent deployments further impose three constraints that make many existing confidence estimators for agents impractical. First, many agents use black-box LLMs that do not expose internal states, token probabilities, or even LLM reasoning text, making estimators that rely on these privileged signals unusable. Second, even a single agent run may be costly, making estimators requiring multiple roll-outs highly impractical. Finally, training a separate estimator requires labeled trajectories, which are costly to collect and can become stale as agents, models, tools, and environments evolve. Estimators reliant on token probabilities therefore cannot be applied to agents that

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

use black-box LLMs (Zhao et al., 2025; Zhang et al., 2026; Bouchard & Chauhan, 2026), and estimators based on agreement require an impractical number of runs for many real-world tasks (Duan et al., 2025). Verbalized confidence meets these constraints, but the resulting estimates are often substantially overconfident (Kaddour et al., 2026).

To overcome these limitations, we introduce **Confidence Reasoning Graphs (CRGs)**, a confidence estimation method for agents that evaluates a single trajectory without requiring privileged model access, repeated roll-outs, or labeled training data. CRG recursively decomposes the claim that *the agent successfully accomplished its task* into a set of contextualized sub-claims that define success for the task, then estimates confidence in each leaf sub-claim grounded in evidence from the trajectory, before finally aggregating the resulting leaf confidences into a root confidence. Together, these steps yields an overall confidence estimate and a structured argument supporting it.

We evaluate CRGs on trajectories produced by three strong agent LLMs across three challenging agent benchmarks: SWE-Bench Verified (Chowdhury et al., 2024), EnterpriseOps-Gym (Malay et al., 2026), and SkillsBench (Li et al., 2026). CRGs achieve the lowest adaptive ECE and Brier score and the highest decision utility on every benchmark, outperforming both verbalized and surrogate-probability baselines. We further show through ablations that leaf-level confidence estimation and aggregation, rather than graph construction alone, drive these gains, that our approach generalizes across agent frameworks. Ablations show that these gains come primarily from leaf-level confidence estimation and aggregation rather than graph construction alone, with product aggregation giving the best calibration among tested training-free rules. The gains persist across agent frameworks and task resampling, while CRG remains on the cost-calibration Pareto frontier.

Our contributions are as follows:

- We introduce **CRGs**, a structured, inspectable framework for training-free confidence estimation from a single black-box agent trajectory.
- We evaluate **CRGs** across diverse agentic benchmarks, agent frameworks, and backbone models, and show that they provide substantially better calibrated confidence than strong verbalized and surrogate-probability baselines.
- We validate each design component through ablations and show that CRGs lie on the cost–calibration Pareto frontier: the best-performing instantiation costs only \$0.04–\$0.07 per estimate, a fraction of the cost of generating a trajectory with a strong LLM.

2 Related Work

Confidence Estimation for LLMs and Agents Prior work estimates confidence in LLM outputs using either "white-box" signals such as token probabilities and internal representations (Vashurin et al., 2025; Malinin & Gales, 2021), "black-box" signals based on agreement across sampled outputs (Wang et al., 2023; Kuhn et al., 2023), or LLM verbalized confidence (Kadavath et al., 2022; Tian et al., 2023; Xiong et al., 2024). Extending these approaches to *LLM agents* requires estimating confidence over the heterogeneous, multi-step trajectories they produce. A few works have approached this challenge, but require learning from or aggregating "white-box" signals (Bouchard & Chauhan, 2026; Zhang et al., 2026; Zhao et al., 2025; Ma et al., 2026), using signals from repeated or branching executions (Mehta, 2026; Duan et al., 2025), or learning from labeled trajectories (Xuan et al., 2026). These approaches therefore cannot be applied directly to the real-world setting described in § 1, where agents use costly, black-box LLMs. While verbalized confidence has been explored for agents, findings suggest they are highly over-confident (Kaddour et al., 2026).

Structured Arguments and Belief Propagation. Other prior works have other areas have explored assessing a target proposition using a structured argument of claims and evidence. Assurance cases, a closely related framework widely used in safety engineering, justify a critical system property through a hierarchical argument grounded in evidence (Rushby, 2015). A few works have explored the construction or evaluation of assurance cases using LLMs (Ikhwantri & Marijan, 2025; 2026), however these tasks differ substantially from estimating agent confidence, and none use the structure of the argument to quantify truth in the root claim. Related forms of structure also appear in claim verification, where a target claim is expanded into graphs or trees of logically related subclaims, including entailed and contradictory premises (Hou et al., 2025). The Bayesian approach of

Hou et al. (2025) models claim truth values as latent variables in a hidden Markov tree and infers target truth from observed LLM confidence scores. Its emission probabilities, $p(S_u \mid Z_u)$, are estimated from held-out truth-labeled statements, requiring labeled calibration data before inference. We instead target a setting requiring neither fitted parameters nor labeled calibration data.

3 PRELIMINARIES

LLM Agents An LLM agent is an AI agent whose policy is governed by an LLM (Su et al., 2024). Given a task instruction q, the agent interacts with an environment Σ over a sequence of steps. At step t, the agent's policy π conditions on the interaction history $h_t = (q, a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ to sample an action $a_t \sim \pi(\cdot \mid h_t)$, with $a_t \in \mathcal{A}$. Applying a_t to Σ returns an observation $o_t \in \mathcal{O}$. Execution halts after T steps, either when the agent emits a terminal action or when a step budget is exhausted. The agent's execution is summarized by its *trajectory*:

$$\tau = (q,\, a_1, o_1,\, a_2, o_2,\, \ldots,\, a_T, o_T), \tag{1}$$

together with the *solution* A it produces, which may be an explicit final response within a_T, or a change the agent has made to Σ, depending on the task.

Confidence Estimation Let $y \in \{0, 1\}$ indicate whether the agent's solution A successfully completes task q, as determined by the environment's verifier. A *confidence estimator* maps an observable trajectory τ to a confidence score, $F_\theta : \mathcal{T} \to [0, 1]$, where $c = F_\theta(\tau)$ estimates the probability of success, given the trajectory $P(y = 1 \mid \tau)$. We study this problem under three practical constraints which together characterize the real-world deployment setting introduced in § 1:

1. **Black-box:** The estimator reads only the textual content of τ, not hidden representations, token-level probabilities, or reasoning tokens.
2. **Single-trajectory:** The estimator observes only one completed execution τ, and may not re-run the agent or interact with the environment Σ.
3. **Training-free:** The estimator requires no labeled trajectories, and hence no supervised fitting or post-hoc calibration.

An estimator satisfying all three applies to any agent task, framework, and backbone model without instrumentation, and supports deployments with proprietary models in state-changing agents.

Evaluating a Confidence Estimator Confidence estimates serve different purposes, so we report complementary metrics. Adaptive **Expected Calibration Error (ECE)** measures calibration by comparing mean confidence with empirical success rates in ten approximately equal-count bins. **Area Under the Receiver-Operating characteristic Curve (AUROC)** measures whether successful trajectories receive higher confidence than unsuccessful ones, while the **Brier score** measures squared probabilistic error. Because calibration and ranking alone do not capture the consequences of acting on an estimate, we also report the **Behavioral Alignment Score (BAS)** (Wu et al., 2026), which averages accept-or-abstain utility across risk thresholds and strongly penalizes confident failures. Formal definitions of ECE and Brier score appear in § A.2; the BAS decision model appears in § A.12.

4 METHODOLOGY

Given a trajectory τ, we re-frame confidence estimation as evaluating confidence in the claim that the agent accomplished its task *successfully.* CRG evaluates this claim using a structured argument inspired by *assurance cases* in four steps, illustrated in Figure 1. First, we use an LLM to recursively decompose it into simpler, contextualized claims which jointly define task success (§ 4.1). Second, an LLM grounds each resulting leaf claim in evidence extracted from the trajectory (§ 4.2). We then estimate the confidence in the truth of each claim using an LLM (§ 4.3), and finally aggregate these estimates into confidence in original claim (§ 4.4). As in assurance cases, our motivation is that evaluating a high-level, multifaceted claim is assessed more reliably when decomposed into concrete, evidence-grounded sub-claims. Formally, let G_0 denote the claim that *the agent successfully*

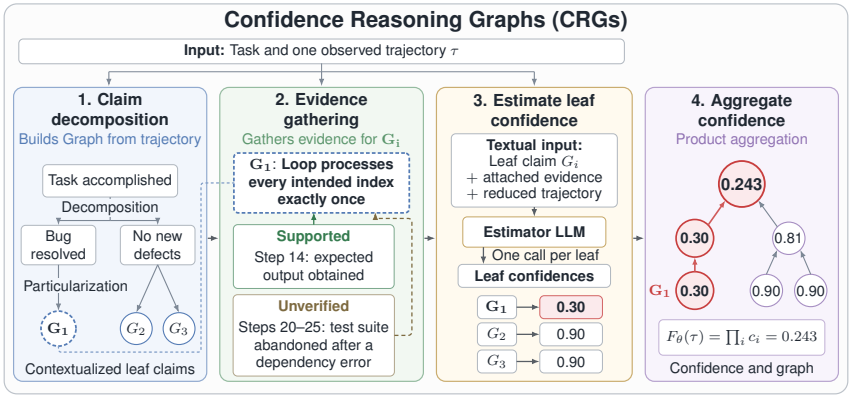


Figure 1: Overview of CRG confidence estimation. A constructor LLM (1) decomposes and particularizes the task-success claim and (2) attaches cited evidence as Supported, Undermined, or Unverified (two types shown). An estimator LLM (3) receives a text prompt containing a leaf claim, its attached evidence, and a reduced trajectory, and outputs one confidence score per leaf. (4) Product aggregation computes root confidence while retaining the graph for inspection. Red highlights the low-confidence G_1 path. Scores are illustrative; product aggregation uses a conditional-independence approximation.

completed its task. CRG estimates confidence in the probability that this claim is true given the trajectory:

$$F_\theta(\tau) \;\approx\; P(G_0 \mid \tau).$$

4.1 Claim Decomposition

Starting from $\mathcal{V} = \{G_0\}$, an LLM iteratively expands a leaf claim by applying one of two refinement strategies, both inspired by argument strategies in the Goal Structuring Notation assurance case formalism (Kelly & Weaver, 2004):

1. **Decomposition** splits a claim $G \in \mathcal{V}$ into individually necessary and jointly sufficient sub-claims $\mathrm{ch}(G) = \{G_1, \ldots, G_m\}$, such that $G \equiv \bigwedge_{j=1}^{m} G_j$ and the sub-claims are mutually independent given τ.
2. **Particularization** restates a claim $G \in \mathcal{V}$ as one contextualized for the specific task, yielding one equivalent child $\mathrm{ch}(G) = \{G'\}$ such that $G \equiv G'$.

Decomposition splits a broad claim into constituents that can be checked separately; particularization restates a generic claim as one that can be evaluated in the context of the given task q and trajectory τ. As an example, consider a coding agent tasked with fixing an off-by-one bug in a function `average`.[1] The root claim G_0 might be decomposed into "the reported bug has been resolved" and "no new defects were introduced," and the former particularized as "the modified loop processes every intended index exactly once." Both strategies aim to preserve the same invariant: each parent claim is equivalent to the conjunction of its children:

$$G \;\equiv\; \bigwedge_{G' \in \mathrm{ch}(G)} G'. \tag{2}$$

Applying this invariant recursively, the root claim G_0 is equivalent to the conjunction of the tree's leaves $\mathcal{L}$:

$$G_0 \;\equiv\; \bigwedge_{G \in \mathcal{L}} G. \tag{3}$$

We formalize this equivalence and its implications for confidence aggregation in § A.13. The LLM may refine any current leaf or stop early, subject to a maximum refinement depth k, where depth

[1] An off-by-one error is a common class of bug in which a loop or index boundary is set one position too high or too low — here, a summation loop that skips the final element, so that `average([1,2,3])` returns 1.5 rather than 2.0.

is the number of refinement edges on a root-to-leaf path. We set $k = 5$ throughout; § A.6 examines sensitivity to this choice. § A.10 evaluates how well constructed graphs satisfy Equation 2 in practice.

4.2 Evidence Gathering

To ground confidence estimation for the leaf claims, we enrich the claim tree induced by ch with evidence from the trajectory, forming a graph. In a single prompt conditioned on both τ and this tree, an LLM produces a set of evidence items $\mathcal{X}$ and attaches each item $x \in \mathcal{X}$ to one or more leaf claims in $\mathcal{L}$. Each evidence item x describes a salient event from the trajectory for evaluating claims. Each attachment cites a set of trajectory steps and receives one of three labels: `Supported`, `Undermined`, or `Unverified`. We represent these labeled attachments as

$$\mathcal{R} \subseteq \mathcal{X} \times \mathcal{L} \times 2^{\{1,\dots,T\}} \times \{\texttt{Supported}, \texttt{Undermined}, \texttt{Unverified}\},$$

where $(x, G, S, r) \in \mathcal{R}$ means that evidence item x is attached to leaf claim G citing trajectory steps S with label r. Our relations follow work on evidence-based factuality evaluation (Fatahi Bayat et al., 2025; Song et al., 2024), where each piece of evidence supports belief in claim, undermines belief in that claim, or indicates that a claim cannot be verified from the trajectory. Together, $\mathcal{V} \cup \mathcal{X}$ form the graph's nodes, while ch and $\mathcal{R}$ define its claim and evidence edges, respectively.

Returning to the running example, the claim "the modified loop processes every intended index exactly once" may be `Supported` by step 14, where the agent runs the repaired function and obtains the expected output, and `Unverified` by steps 20–25, where the agent abandons efforts to run the project's test suite after a dependency fails to install. `Unverified` evidence matters because a common failure mode of agents is not by acting incorrectly, but stopping short of confirming that an action succeeded. The prompt for claim decomposition and evidence gathering appears in § A.11.2.

4.3 Estimating Leaf Confidence

Let $\psi : \mathcal{L} \to [0, 1]$ assign each leaf claim $G \in \mathcal{L}$ a confidence $c_G = \psi(G)$, interpreted as an estimate of $P(G \mid \tau)$. We instantiate ψ by verbalized confidence, prompting an estimator LLM for the probability that G holds.

Each leaf is scored in a separate call using its attached evidence and a reduced trajectory. We retain the prefix through the last cited step, using T as the cutoff when no step is cited. Uncited actions are replaced by framework-provided summaries when available and truncated otherwise. Summarizing them preserves context, while keeping the cited steps the most salient content in the prompt. The reduction assumes that the latest relevant evidence was cited; it cannot recover evidence omitted during construction. Prompts appear in § A.11.

4.4 Aggregating Confidence to the Root

Given leaf confidences, we define a confidence $f(G)$ for every claim in $\mathcal{V}$ by recursion from the leaves:

$$f(G) = \begin{cases} \psi(G), & G \in \mathcal{L}, \\ \bigoplus_{G' \in \mathrm{ch}(G)} f(G'), & \text{otherwise}, \end{cases} \tag{4}$$

where $\bigoplus$ is an aggregation operator over the confidence values of a claim's children, and the estimator returns the confidence at the root, $F_\theta(\tau) = f(G_0)$. This formulation lets us compare aggregation rules over identical graphs, which we do in Appendix A.8.

Under Equation 3, the root claim holds exactly when all leaf claims hold, so its probability is their joint probability. Computing this joint probability generally requires modeling dependencies among the leaves. For aggregation, we make the simplifying approximation that the children introduced at each decomposition are conditionally independent given τ, yielding the approximation $P(G_0 \mid \tau) \approx \prod_{G \in \mathcal{L}} P(G \mid \tau)$. Replacing each marginal probability with its leaf estimate $c_G \approx P(G \mid \tau)$ gives:

$$F_\theta(\tau) = f(G_0) = \prod_{G \in \mathcal{L}} c_G. \tag{5}$$

Table 1: OpenHands success rates (%) by agent LLM. n is the number of problems.

Benchmark	Num. Problems	Total Trajectories	GPT-5.5	Gemini-3.5 Flash	MiniMax-M3	Avg.
SWE-Bench Verified	306	900	73.9	74.3	71.7	73.3
EnterpriseOps-Gym	325	975	41.8	44.3	31.4	39.2
SkillsBench	81	219	50.7	50.7	45.2	48.9

Thus, our final confidence estimate is the product of the leaf confidences. § A.13 gives the general dependence-aware factorization, marginal-only bounds, and derives the product rule under our local conditional-independence approximation.

5 Experimental Setup

Benchmarks and Trajectories. We evaluate confidence estimators on trajectories from three challenging benchmarks: SWE-Bench Verified (Chowdhury et al., 2024) for software engineering, EnterpriseOps-Gym (Malay et al., 2026) for tool-heavy enterprise workflows, and SkillsBench (Li et al., 2026) for long-horizon skill use. We focus on harder problems, where task success is less saturated and confidence estimation is more informative; § A.1 details the filtering criteria and splits. Table 1 summarizes the resulting data. For each problem, we collect one trajectory using the OpenHands agent framework (Wang et al., 2025), with benchmark-specific iteration caps (§ A.1), with three agent LLMs: GPT-5.5 (OpenAI, 2026a), Gemini-3.5 Flash (Google DeepMind, 2026), and the open-weight MiniMax-M3 (Lai et al., 2026). Each benchmark's verifier provides the success label y used to evaluate confidence estimates. We evaluate transfer beyond OpenHands in § 6.2.

CRG Settings. We run CRG with two estimator LLMs, which read each completed trajectory and output a confidence: the open-weight Qwen-3.8 27B and the closed-source GPT-5.6 Sol (OpenAI, 2026b), both set to medium reasoning effort. We use a maximum refinement depth of $k = 5$ across benchmarks without per-benchmark tuning; § A.6 shows that no graph exceeds this depth in a development-set ablation.

Baselines. We compare CRG against three black-box baselines and one surrogate-probability baseline. *Basic Verbalizer* directly asks the estimator LLM for its confidence that the agent succeeded (Zhang et al., 2026). *Reason-as-Graph* describes our claim-decomposition procedure in the estimator prompt, testing whether decomposition-oriented reasoning alone recovers the benefits of explicit graph construction and leaf aggregation. Following self-consistency work (Wang et al., 2023), *Verbal Consistency* samples ten true/false judgments of task success *with reasoning* and uses the fraction judged true as confidence.[2] Finally, *Surrogate LNSP* uses the length-normalized sequence probability of the agent's final action under a surrogate LLM (Shrivastava et al., 2023; Zhang et al., 2026; Bouchard & Chauhan, 2026). All black-box baselines use the same backbone LLM as the CRG constructor and estimator. § A.11 provides the complete prompts.

Metrics. We report adaptive ECE with 10 approximately equal-count bins, Brier score, AUROC, and BAS. Their definitions appear in § A.2 and § A.12; § A.5 reports equal-width ECE and sensitivity to bin count.

6 Results and Analyses

We compare CRG against black-box and surrogate probability baselines in Section 6.1, test transfer to two other frameworks in Section 6.2, and analyze the contribution of key components of the framework through ablations in Section 6.3.

6.1 Main Results

Overall Performance. Table 2 shows a consistent advantage for CRGs in calibration and decision quality across all three benchmarks. **CRGs achieve the lowest adaptive ECE and Brier score**

[2]Due to inference cost, we run Verbal Consistency only with Qwen-3.8 27B.

Table 2: Confidence estimation results on OpenHands trajectories, averaged over agent LLMs. We report adaptive ECE (10-bins), Brier Score, AUROC, and BAS. Best per column in **bold**, second underlined. †White-box: uses token probabilities from a surrogate LLM.

		SWE-Bench Verified				EnterpriseOps-Gym				SkillsBench			
Estimator	LLM	ECE↓	Brier↓	AUROC↑	BAS↑	ECE↓	Brier↓	AUROC↑	BAS↑	ECE↓	Brier↓	AUROC↑	BAS↑
Basic Verbalizer	GPT-5.6 Sol	0.20	0.23	0.57	−0.04	0.58	0.57	0.50	−2.48	0.46	0.46	0.54	−2.17
	Qwen-3.8 27B	0.18	0.22	**0.66**	0.27	0.49	0.47	0.60	−0.57	0.41	0.41	**0.65**	−0.33
Reason-as-Graph	GPT-5.6 Sol	0.17	0.22	0.57	0.06	0.57	0.56	0.47	−2.69	0.43	0.44	0.51	−2.09
	Qwen-3.8 27B	0.18	0.22	**0.66**	0.27	0.46	0.44	0.62	−0.50	0.38	0.38	0.63	−0.40
Verbal Consistency	Qwen-3.8 27B	0.24	0.25	0.54	−7.58	0.45	0.45	0.62	−13.72	0.45	0.44	0.60	−13.87
Surrogate LNSP†	Qwen-3.8 27B	0.10	0.21	0.47	0.36	0.22	0.28	0.61	0.04	0.16	0.28	0.46	0.09
CRG (ours)	GPT-5.6 Sol	0.16	0.23	0.62	0.36	0.24	0.29	**0.67**	0.06	0.22	0.29	**0.65**	0.09
	Qwen-3.8 27B	**0.09**	**0.20**	0.57	**0.37**	**0.13**	**0.24**	0.62	**0.08**	**0.11**	**0.24**	**0.65**	**0.14**

and highest BAS on every benchmark, with a CRG also attaining the best or second-best AUROC. CRGs with Qwen-3.8 27B are best calibrated, achieving adaptive ECEs of 0.09, 0.13, and 0.11 on SWE-Bench Verified, EnterpriseOps-Gym, and SkillsBench, respectively, compared with 0.18–0.49 for direct verbalization and 0.10–0.22 for the surrogate-probability baseline. CRGs with GPT-5.6 Sol substantially out-perform their verbalized counter-parts, indicating that improvements are not specific to a single CRG instantiation. We also observe divergence between calibration and discrimination: direct verbalizers reach AUROC 0.66 on SWE-Bench Verified despite substantial miscalibration, whereas Surrogate LNSP attains lower ECE but weak AUROC, performing near-chance on SWE-Bench Verified (0.47) and SkillsBench (0.46). **CRG is the only evaluated approach that consistently combines strong calibration (ECE ↓), competitive discrimination (AUROC ↑), and positive expected utility (BAS ↑) across all three benchmarks.** We validate the significance in these gains with bootstrap-resampling (§ A.14), and further find these gains are inexpensive: CRG with Qwen 3.8 27B costs $0.04–0.07 per trajectory and lies on the cost–calibration Pareto frontier, while verbalized baselines remain less calibrated even at higher reasoning effort and comparable cost (§ A.4).

Evaluating Reliability. To investigate the source of calibration differences in Table 2, Figure 2 plots reliability curves and empirical confidence CDF's for the best-calibrated verbalizer, surrogate, and CRG instantiation. The represented verbalizer (Reason-as-Graph) is systematically overconfident, assigning confidence scores over 90% to the majority of trajectories on all benchmarks despite substantially lower success rates. Figure 2 further reveals why calibration error alone is insufficient as a metric. On SWE-Bench Verified, Surrogate LNSP's adaptive ECE is close to CRG's (0.10 vs. 0.09), but its predictions are concentrated near the benchmark success rate (mean confidence 0.69 vs. 0.73 success rate). This yields low calibration error without reliably distinguishing individual successes from failures, as reflected in its poor AUROC. **CRGs instead distribute confidence estimates across a substantially wider range while remaining closer to the calibration diagonal on all benchmarks.**

Evaluating Decision Utility. To evaluate how confidence estimates inform decision-making under risk, Figure 3 reports selective utility across acceptance thresholds t, with BAS corresponding to the area under each curve. A utility below zero means that for the risk-reward ratio defined by t, it would have been better to abstain altogether than to use that estimator for decision making. At low thresholds (low-risk settings), all methods behave similarly. However, their behavior diverges as risk increases: Consistent with the overconfidence observed in Figure 2, Reason-as-Graph incurs *negative utility* on all benchmarks due to high-confidence failures, reaching -3.6 on EnterpriseOps-Gym. Surrogate LNSP avoids incurring costs of this magnitude, but achieves lower utility than CRGs at nearly every risk-threshold. **CRGs achieve the highest BAS on all three benchmarks, maintain positive utility for the widest span of risk thresholds, and are the only method that never falls substantially below total abstention.** CRGs therefore support the best decision-making in both low-risk and high-risk environments.

6.2 Generalization to Other Agent Frameworks

In this section, we evaluate whether CRG transfers to agent frameworks beyond OpenHands. We evaluate on 609 SWE-Bench Verified trajectories collected with two additional framework–model

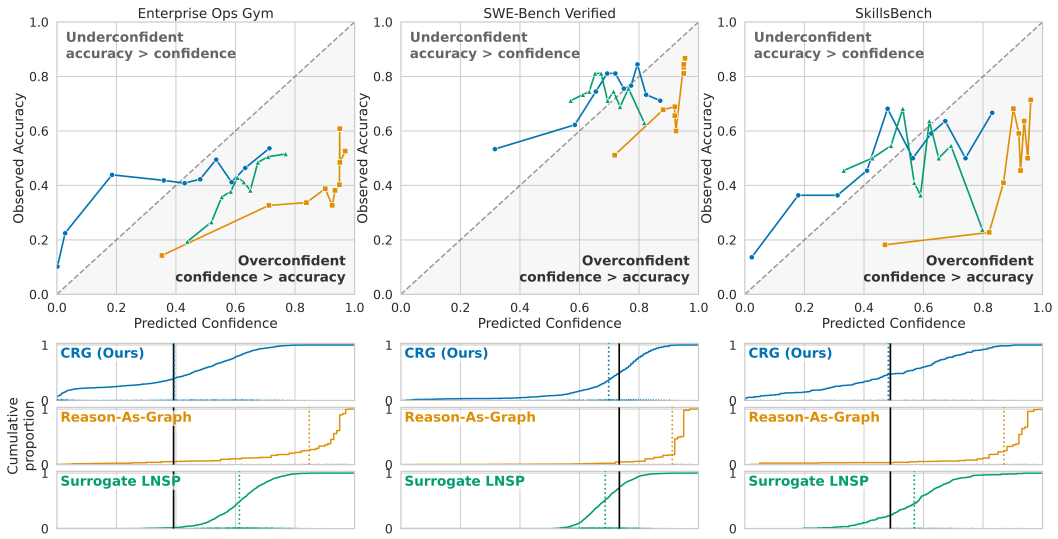


Figure 2: Reliability diagrams (top) and empirical confidence CDFs (bottom) for CRG (Ours), Reason-as-Graph, and Surrogate LNSP, using 10 equal-mass bins. Perfect calibration lies on the diagonal; under-confidence is above and overconfidence below (shaded). Dashed and solid vertical black lines indicate mean confidence and benchmark success rate, respectively. CRG is best calibrated on all three benchmarks, with estimates distributed across the confidence range.

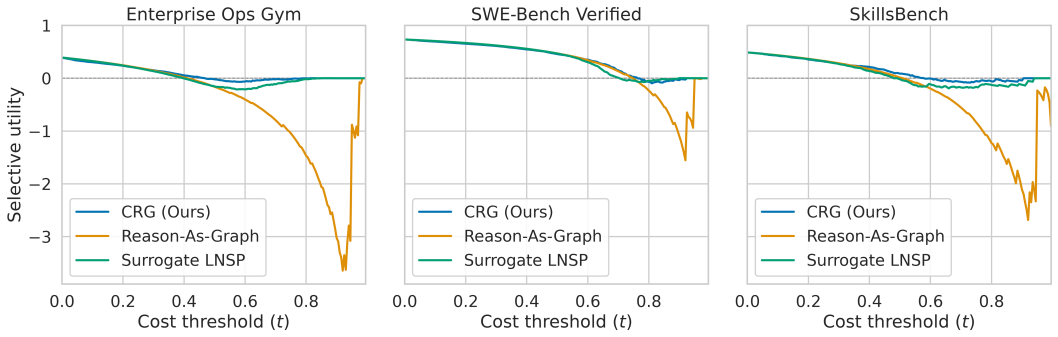


Figure 3: Selective utility across cost thresholds t for the best method from each class (all Qwen-3.8 27B). The dashed zero line corresponds to abstaining from all solutions, so values below it are worse than abstention. All estimators coincide at low t, where accepting all solutions is optimal. As t increases, Reason-As-Graph incurs losses as large as -3.6 on EnterpriseOps-Gym, while CRG (Ours) remains above Surrogate LNSP through the mid range.

combinations: Codex with GPT-5.5 and Claude Code with Claude Opus 4.7. We compare against the black-box verbalized baselines; Surrogate LNSP is omitted because computing its token probabilities requires reconstructing the exact LLM context at each agent step.[3]

Table 3 shows that the calibration and decision-utility gains transfer across agent frameworks **CRGs reduce adaptive ECE from 0.19 to 0.05, while achieving the lowest Brier score (0.20) and highest BAS (0.38).** For discrimination, CRG is comparable to both verbalizers on Codex trajectories (AUROC 0.61 vs. 0.63) but weaker on Claude Code trajectories (0.54 vs. 0.64–0.67). This reinforces the distinction between calibration and discrimination: in this transfer setting, CRGs produce substantially better probability estimates and acceptance decisions despite weaker ranking quality.

[3]This context may not be recoverable post hoc, for cloud-hosted or closed agent frameworks, making surrogate-based methods difficult to apply in such settings.

Table 3: Transfer to other agent frameworks on SWE-Bench Verified. Codex (GPT-5.5) has $n = 305$ trajectories and 72.5% success; Claude Code (Claude Opus 4.7) has $n = 304$ and 72.0% success. All estimators use Qwen-3.8 27B. ECE is adaptive ECE with 10 bins.

	Codex (GPT 5.5)				Claude Code (Opus 4.7)			
Estimator	ECE↓	Brier↓	AUROC↑	BAS↑	ECE↓	Brier↓	AUROC↑	BAS↑
Qwen 3.8 27B								
Basic Verbalizer	0.19	0.24	**0.63**	0.22	0.19	0.23	0.64	0.25
Reason-as-Graph	0.19	0.23	**0.63**	0.23	0.19	0.23	**0.67**	0.26
CRG (ours)	**0.05**	**0.20**	0.61	**0.38**	**0.11**	**0.22**	0.54	**0.34**

Table 4: Ablating graph construction and confidence propagation on development data (Qwen-3.8 27B, $n = 495$). Reason-as-Graph decomposes the task in the prompt and reports one confidence; Reason-with-Graph builds the graph but scores the root directly rather than propagating from leaves.

Method	ECE↓	Brier↓	AUROC↑	BAS↑
Reason-as-Graph (no explicit graph)	0.38	0.37	**0.73**	−0.23
Reason-with-Graph (no propagation)	0.35	0.36	0.61	−0.16
CRG (graph + propagation)	**0.10**	**0.23**	0.68	**0.17**

6.3 Component Ablation

We examine the contribution of each component of our method in Table 4. *Reason-as-Graph* is our assurance-case inspired verbal baseline. *Reason-with-Graph* constructs the same claim and evidence graph as a CRG, but directly verbalizes confidence in the root claim G_0 conditioned on the graph, ablating our propagation method. We find *Reason-with-Graph* provides only a modest calibration improvement over Reason-as-Graph (adaptive ECE $0.38 \rightarrow 0.35$; Brier $0.37 \rightarrow 0.36$). Using our full method with leaf-level scoring and aggregation substantially improves both calibration and decision utility: ECE drops to 0.10, Brier to 0.23, and BAS increases from -0.16 to 0.17. **Thus, the main gain does not come from simply exposing the model to a structured decomposition; it arises from using that decomposition to make localized confidence judgments and combine them into the final estimate.** The improvement is primarily in confidence quality rather than ranking: CRG achieves AUROC 0.68, below Reason-as-Graph (0.73) but above Reason-with-Graph (0.61). This is consistent with our main results, where improved calibration need not imply improved pairwise discrimination. Finally, § A.8 compares alternative aggregation rules on the same graphs — arithmetic and geometric means, maximum, the Fréchet bounds, and the learned belief propagation of Hou et al. (2025) — and finds that product aggregation gives the strongest calibration and competitive discrimination among training-free rules.

6.4 Additional Findings

In § A.3, we analyze case studies, demonstrating how our claim graph might be audited by a human and over-ruled at inference time. Further analyses show that our gains are also robust in post-hoc calibration with temperature scaling (§ A.9) and our approach is insensitive to LLM reasoning effort (§ A.7). In § A.10, we use textual entailment to test the refinement invariant in Equation 2 and diagnose violations of the independence approximation behind Equation 5; We find CRG remains effective despite imperfect adherence to these assumptions.

7 Conclusion

We introduced **Confidence Reasoning Graphs (CRGs)**, which estimates task-success confidence from a single black-box agent trajectory by decomposing success into evidence-grounded claims and aggregating their confidences. **CRGs** improve calibration and decision utility across benchmarks over evaluated baselines while retaining strong discrimination. This enables practical, auditable confidence estimates for black-box agents without training.

AI USE STATEMENT

In this work, we used generative AI tools for refinement of the writing, formatting help, literature search, figure conversion and implementation, and in the assistance of writing for proofs. We have not used generative AI tools for producing our ideas, designing our pipelines or experiments, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for coding and creation of plots, both of which we carefully review and improve manually. We have reviewed all AI-assisted content used anywhere in the production of this work. We checked LLM literature review by by carefully reviewing each paper, and conducting further literature review manually. We reviewed LLM edits to our text with using a text diff to compare the before and after. We used LLMs for the writing in an atomic fashion: to refine the phrasing of individual points in the text, rather than their semantics. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work aims to support safer use of LLM agents by providing confidence estimates for individual executions. However, these estimates should not be treated as guarantees of correctness. An inaccurate or overconfident estimate could create false reassurance, especially in high-stakes applications.

Our method also inherits limitations from the LLMs used to construct and evaluate the reasoning graph: decompositions may be incomplete, dependencies between claims may be missed, and evidence may be misinterpreted. CRG confidence should therefore be used as a decision-support signal alongside appropriate verification and human oversight.

REPRODUCIBILITY STATEMENT

We support reproducibility by describing our method in § 4 and the benchmarks, estimator settings, baselines, and metrics in § 5. § A.1 documents benchmark filtering, trajectory collection, and success-label assignment; § A.11 provides the full prompts; and § A.13 states the assumptions and proofs for confidence aggregation. The implementation repository contains the prompt templates, experiment configurations, evaluation scripts, and dependency lockfile used in our experiments. § A.14 describes the resampling procedure for the uncertainty analysis. Framework versions, randomization settings, and their scope are documented in § A.15.

REFERENCES

Dylan Bouchard and Mohit Singh Chauhan. Beyond Single-Turn Confidence: Trajectory-Adapted Uncertainty Quantification for LLM Agents, August 2026. URL http://arxiv.org/abs/2608.11552. arXiv:2608.11552 [cs.CL].

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench Verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Jinhao Duan, James Diffenderfer, Sandeep Madireddy, Tianlong Chen, Bhavya Kailkhura, and Kaidi Xu. UProp: Investigating the Uncertainty Propagation of LLMs in Multi-Step Agentic Decision-Making, June 2025. URL http://arxiv.org/abs/2506.17419. arXiv:2506.17419 [cs.CL].

Farima Fatahi Bayat, Lechen Zhang, Sheza Munir, and Lu Wang. FactBench: A dynamic benchmark for in-the-wild language model factuality evaluation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33090–33110, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1587. URL https://aclanthology.org/2025.acl-long.1587/.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. A Survey of Confidence Estimation and Calibration in Large Language Models. In Kevin Duh,

Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6577–6595, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.366. URL https://aclanthology.org/2024.naacl-long.366/.

Google. Gemini 3.6 flash: Model card, July 2026. URL https://deepmind.google/models/model-cards/gemini-3-6-flash/.

Google DeepMind. Gemini 3.5 Flash: Model Card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-5-flash/.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1321–1330. JMLR.org, 2017.

Bairu Hou, Yang Zhang, Jacob Andreas, and Shiyu Chang. A Probabilistic Framework for LLM Hallucination Detection via Belief Tree Propagation. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3076–3099, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.158. URL https://aclanthology.org/2025.naacl-long.158/.

Minghui Huang. Decmetrics: Structured claim decomposition scoring for factually consistent llm outputs, 2025. URL https://arxiv.org/abs/2509.04483.

Fariz Ikhwantri and Dusica Marijan. Explainable Compliance Detection with Multi-Hop Natural Language Inference on Assurance Case Structure, July 2025. URL http://arxiv.org/abs/2506.08713. arXiv:2506.08713 [cs.CL].

Fariz Ikhwantri and Dusica Marijan. Evaluating Assurance Cases as Text-Attributed Graphs for Structure and Provenance Analysis, April 2026. URL http://arxiv.org/abs/2604.20577. arXiv:2604.20577 [cs.SE].

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.

Jean Kaddour, Srijan Patel, Gbètondji Dovonon, Leo Richter, Pasquale Minervini, and Matt J. Kusner. Agentic Uncertainty Reveals Agentic Overconfidence, 2026. URL https://arxiv.org/abs/2602.06948. Version Number: 1.

Tim Kelly and Rob Weaver. The goal structuring notation–a safety argument notation. *Proc Dependable Syst Networks Workshop Assurance Cases*, 01 2004.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu (eds.), *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-3250/.

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.

Xunhao Lai, Weiqi Xu, Yufeng Yang, Qiaorui Chen, Yang Xu, Lunbin Zeng, Xiaolong Li, Haohai Sun, Haichao Zhu, Vito Zhang, Jinkai Hu, Jiayao Li, Rui Gao, Zekun Li, Songquan Zhu, Jingkai Zhou, and Pengyu Zhao. Minimax sparse attention, 2026. URL https://arxiv.org/abs/2606.13392.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han-chung Lee, and Dawn Song. SkillsBench: Benchmarking How Well Agent Skills Work Across Diverse Tasks, June 2026. URL http://arxiv.org/abs/2602.12670. arXiv:2602.12670 [cs.AI].

Hao Liu, Zi-Yi Dou, Yixin Wang, Nanyun Peng, and Yisong Yue. Uncertainty calibration for tool-using language agents. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16781–16805, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.978. URL https://aclanthology.org/2024.findings-emnlp.978/.

Zhengzhao Ma, Boxi Cao, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. From sequence to structure: Relational uncertainty propagation for llm agents, 2026. URL https://arxiv.org/abs/2608.16002.

Shiva Krishna Reddy Malay, Shravan Nayak, Jishnu Sethumadhavan Nair, Sagar Davasam, Aman Tiwari, Sathwik Tejaswi Madhusudhan, Sridhar Krishna Nemala, Srinivas Sunkara, and Sai Rajeswar. EnterpriseOps-Gym: Environments and Evaluations for Stateful Agentic Planning and Tool Use in Enterprise Settings, March 2026. URL http://arxiv.org/abs/2603.13594. arXiv:2603.13594 [cs.AI].

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jN5y-zb5Q7m.

Aman Mehta. When agents disagree with themselves: Measuring behavioral consistency in llm-based agents, 2026. URL https://arxiv.org/abs/2602.11619.

Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

OpenAI. GPT-5.5 System Card, 2026a. URL https://openai.com/index/gpt-5-5-system-card/.

OpenAI. GPT-5.6 System Card. https://deploymentsafety.openai.com/gpt-5-6, July 2026b. Published July 9, 2026.

Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C. Mozer. Mitigating bias in calibration error estimation. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 4036–4054. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/roelofs22a.html.

John M. Rushby. The interpretation and evaluation of assurance cases. 2015. URL https://api.semanticscholar.org/CorpusID:67274133.

Vaishnavi Shrivastava, Percy Liang, and Ananya Kumar. Llamas Know What GPTs Don't Show: Surrogate Models for Confidence Estimation, November 2023. URL http://arxiv.org/abs/2311.08877. arXiv:2311.08877 [cs.CL].

Yixiao Song, Yekyung Kim, and Mohit Iyyer. VeriScore: Evaluating the factuality of verifiable claims in long-form text generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9447–9474, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.552. URL https://aclanthology.org/2024.findings-emnlp.552/.

Yu Su, Diyi Yang, Shunyu Yao, and Tao Yu. Language agents: Foundations, prospects, and risks. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pp. 17–24, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.emnlp-tutorials.3.

OpenHands Team. Openhands index: A comprehensive leaderboard for ai coding agents. https://index.openhands.dev, 2025.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.330. URL https://aclanthology.org/2023.emnlp-main.330/.

Roman Vashurin, Ekaterina Fadeeva, Artem Vazhentsev, Lyudmila Rvanova, Daniil Vasilev, Akim Tsvigun, Sergey Petrakov, Rui Xing, Abdelrahman Sadallah, Kirill Grishchenkov, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, Maxim Panov, and Artem Shelmanov. Benchmarking Uncertainty Quantification Methods for Large Language Models with LM-Polygraph. *Transactions of the Association for Computational Linguistics*, 13:220–248, March 2025. ISSN 2307-387X. doi: 10.1162/tacl_a_00737. URL https://doi.org/10.1162/tacl_a_00737.

Xingyao Wang, Simon Rosenberg, Juan Michelini, Calvin Smith, Hoang Tran, Engel Nyst, Rohit Malhotra, Xuhui Zhou, Valerie Chen, Robert Brennan, and Graham Neubig. The openhands software agent sdk: A composable and extensible foundation for production agents, 2025. URL https://arxiv.org/abs/2511.03690.

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.

Nathaniel Weir, Kate Sanders, Orion Weller, Shreya Sharma, Dongwei Jiang, Zhengping Jiang, Bhavana Dalvi Mishra, Oyvind Tafjord, Peter Jansen, Peter Clark, and Benjamin Van Durme. Enhancing systematic decompositional natural language inference using informal logic. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9458–9482, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.531. URL https://aclanthology.org/2024.emnlp-main.531/.

Sean Wu, Fredrik K. Gustafsson, Edward Phillips, Boyan Gao, Anshul Thakur, and David A. Clifton. Bas: A decision-theoretic approach to evaluating large language model confidence, 2026. URL https://arxiv.org/abs/2604.03216.

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.

Weihao Xuan, Qingcheng Zeng, Heli Qi, Yunze Xiao, Junjue Wang, and Naoto Yokoya. The confidence dichotomy: Analyzing and mitigating miscalibration in tool-use agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11325–11349, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.520. URL https://aclanthology.org/2026.acl-long.520/.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents, 2025. URL https://arxiv.org/abs/2504.21798.

Jiaxin Zhang, Caiming Xiong, and Chien-Sheng Wu. Agentic confidence calibration. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=B1ISNZQHuI.

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 (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 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 Appendix

A.1 Benchmarks and Trajectory Collection

Test Set: Main Experiments

We construct our test set for evaluating confidence estimators using three agentic benchmarks: SWE-Bench Verified (Chowdhury et al., 2024), SkillsBench (Li et al., 2026), and EnterpriseOps-Gym (Malay et al., 2026). Because confidence estimation is most informative when agent success is not saturated, we focus on the more challenging tasks from each benchmark. We describe the problems chosen and number of trajectories sampled below.

SWE-Bench Verified. SWE-Bench Verified evaluates software-engineering agents on real-world repository issues. We select the `test` split of `princeton-nlp/SWE-bench_Verified` and retain instances whose `difficulty` annotation is greater than 15 minutes: `15 min - 1 hour`, `1-4 hours`, or `>4 hours`, yielding **306 problems**.

SkillsBench. SkillsBench contains 87 long-horizon tasks organized into three difficulty levels: 6 *Core* tasks requiring less than one hour, 53 *Extended* tasks requiring 1–4 hours, and 28 *Extreme* tasks requiring more than four hours. We retain the Extended and Extreme tasks and use the *with-skills* condition, giving **81 eligible problems**. The with-skills condition loads the task's supplied skills into the agent context.

EnterpriseOps-Gym. EnterpriseOps-Gym evaluates agents on tool-heavy workflows spanning multiple enterprise application domains. Unlike the other two benchmarks, it does not provide the same human-effort difficulty annotation. We use **50% of the problems within each domain**, yielding **325 problems** while preserving domain composition. The eight domains are Calendar, CSM, Drive, Email, HR, Hybrid, ITSM, and Teams.

Trajectory collection. For each selected problem, we attempt one OpenHands trajectory (Wang et al., 2025) using GPT-5.5 (OpenAI, 2026a), Gemini-3.5 Flash (Google DeepMind, 2026), or MiniMax-M3 (Lai et al., 2026). The saved run configurations use *high* reasoning effort for all three agent LLMs. SkillsBench and EnterpriseOps-Gym trajectories used OpenHands SDK 1.31.0 with a 100-iteration cap. The SWE-Bench trajectories instead use a 500-iteration cap as they are

sourced from the OpenHands Index (Team, 2025), rather then recomputed. We omit any runs trajectories which fail, crash, or cannot be represented in the 262K context-window length of Qwen 3.8 27B for either CRGs or verbalized baselines. This yields the following evaluation set counts: 900 SWE-Bench trajectories across 305 distinct problems (303 GPT-5.5, 300 Gemini-3.5 Flash, and 297 MiniMax-M3), 219 SkillsBench trajectories across 74 problems (73 per agent LLM), and 975 EnterpriseOps-Gym trajectories across 325 problems (325 per agent LLM). These OpenHands trajectories constitute our main evaluation set. We separately evaluate transfer to Codex and Claude Code and describe that evaluation set in § 6.2.

Success labels. For each scored trajectory, we use the benchmark's native verifier to obtain the binary success label $y \in \{0, 1\}$. For SWE-Bench Verified and EnterpriseOps-Gym this is the `resolved` status in the benchmark evaluation report. For SkillsBench, numeric verifier reward counts as success only when it equals one; partial reward is a failure for our binary evaluation.

Development Set

For ablations (§ 6.3, § A.6, § A.8), we construct a development set from two benchmarks:

1. **SWE-Smith** (Yang et al., 2025): a dataset of *synthetic* repository-level SWE problems. We sample 64 random problems.
2. **EnterpriseOps Gym** (Malay et al., 2026): the tool-intensive benchmark used in our main evaluation, from which we sample 64 problems (8 per domain) which do not overlap with problems in the test set described above.

We generate trajectories with a maximum of 100 steps, using the same agent LLMs: GPT 5.5, Gemini 3.5 Flash, and Minimax M3. While we do *no hyper-parameter tuning* on this development set, it was used in the initial development of our prompts.[4] On the test set, we therefore demonstrate generalization of our method to harder, real problems from SWE-Bench Verified, and a new domain in SkillsBench.

A.2 Calibration and Brier Score

For n scored trajectories, let $c_i \in [0, 1]$ be the estimated probability of success and $y_i \in \{0, 1\}$ the observed outcome. Given a partition of the trajectory indices into bins $B_1, \ldots, B_M$, define

$$\mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} c_i, \qquad \mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} y_i.$$

Expected Calibration Error (ECE) (Guo et al., 2017) is

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left| \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \right|. \tag{6}$$

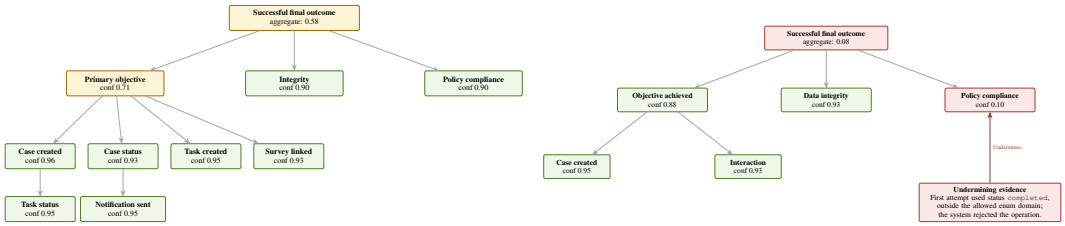
For our main results, we use adaptive ECE (Nixon et al., 2019): we sort trajectories by confidence and divide them into $M = 10$ approximately equal-count bins. In our implementation, bin sizes differ by at most one, and tied confidences retain their original order according to a stable sort, following Nixon et al. (2019). Equal-count binning avoids sparsely populated bins, especially on smaller datasets: reducing the variance of per-bin accuracy estimates, and has been reported to have lower bias (Roelofs et al., 2022). § A.5 examines other bin counts and equal-width bins.

We also report the binary Brier score,

$$\mathrm{Brier} = \frac{1}{n} \sum_{i=1}^{n} (c_i - y_i)^2. \tag{7}$$

Lower values indicate better probabilistic predictions for both metrics. The Brier score summarizes squared prediction error, whereas ECE specifically measures the gap between confidence and observed success frequency within bins.

[4] While our method has a maximum refinement depth hyper-parameter k, we find depths greater than 5 are never achieved, and therefore do not tune this parameter per-benchmark or to optimize performance on development data



(a) Successful CRG Estimation **(b) Failed CRG Estimation**

Figure 4: CRGs for two successful EnterpriseOps-Gym trajectories. (a) A correct estimate: an HR-enrollment task decomposes into verifiable subtasks (case created, status set, survey linked, notification sent), each with high confidence. (b) An incorrect estimate, caused by incomplete graph construction: the agent's first attempt used a status outside the allowed enum domain and was rejected, but no evidence item records the subsequent correction. The policy-compliance claim is therefore scored against the violation alone, receiving 0.10 and driving the root to 0.08. Because the graph exposes which claim and which cited step produced the estimate, a user can inspect this reasoning and overrule it.

A.3 Case Studies

We examine two EnterpriseOps-Gym trajectories to illustrate how CRG connects task-specific claims and trajectory evidence to the final confidence estimate.

Aggregating high-confidence conditions. In Figure 4(a), a health-insurance enrollment task requires creating an HR case, setting its status, creating and linking a survey task, updating the task status, and notifying the employee. The corresponding leaf claims receive confidences of 0.93–0.96. Their aggregation gives a primary-objective confidence of 0.71 and, after accounting for integrity and policy-compliance claims, a root confidence of 0.58. **The graph makes explicit why several individually high-confidence conditions can still yield a more cautious estimate of overall task success.**

Tracing a low-confidence estimate. Figure 4(b) shows a trajectory that also succeeded, but on which CRG estimation is wrong. The agent initially attempts to create a support case using the status `completed`, which is outside the allowed value set and is rejected by the system. CRG attaches this event as undermining evidence to the policy-compliance claim, which receives confidence 0.10; although the case-creation and interaction claims receive 0.95 and 0.93, the root falls to 0.08. The agent subsequently corrected the value and completed the task, but no evidence item records the correction, so the claim was scored against the violation alone — a failure of graph construction rather than of the confidence estimate it supports. **Unlike a single scalar confidence, the graph identifies the specific claim and trajectory evidence responsible for the low estimate**, so a user deciding whether to accept this solution can check that step, find the correction, and overrule the estimate.

A.4 Cost

We compare the inference cost and calibration of each estimator in Figure 5. Because methods differ in model size and in prompt and completion length, we convert inference usage to an estimated cost per trajectory in USD.[5]

Across all three benchmarks, CRG lies on the cost–calibration Pareto frontier, together with the cheaper but less calibrated Surrogate LNSP. CRG costs approximately $0.04–$0.07 per trajectory and achieves lower adaptive ECE than the surrogate on every benchmark, with the largest gap on EnterpriseOps-Gym (0.13 vs. 0.22). Thus, CRG trades additional inference cost for substantially improved calibration in settings where the cheaper surrogate is insufficient.

[5]We run Qwen-3.8 27B locally and estimate its cost using the rates reported by OpenRouter.

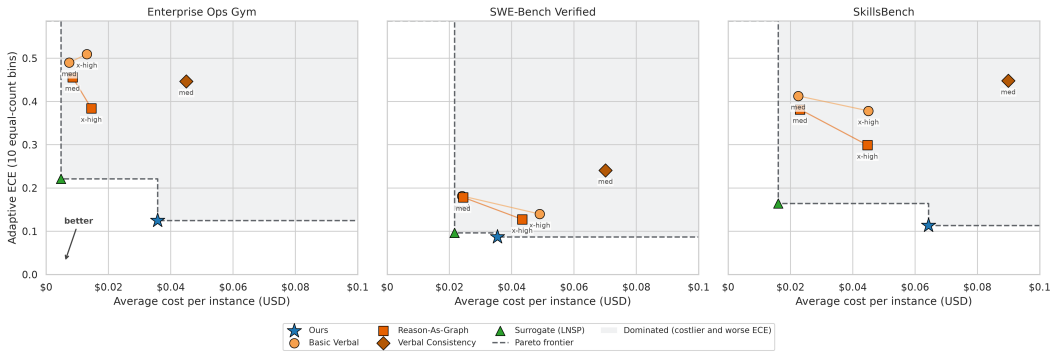


Figure 5: Cost–calibration tradeoff of each estimator: average cost per trajectory (USD) against adaptive ECE (↓), so lower-left is better. CRGs are the best-calibrated estimators on every benchmark, sharing the Pareto frontier with only the cheaper but worse-calibrated Surrogate LNSP. Importantly, CRGs are cost-efficient in absolute terms: \$0.04–0.07 per trajectory, compared to the cost of generating the trajectory being judged, which averages \$0.15, \$1.06, and \$0.95 on Enterprise Ops Gym, SWE-Bench, and SkillsBench respectively.

Table 5: Sensitivity of confidence-estimation calibration error to the number of bins. ECE uses equal-width bins and Adaptive ECE uses equal-mass bins. Best per column in **bold**, second underlined. †White-box: uses token probabilities from a surrogate LLM.

		SWE-Bench Verified					EnterpriseOps-Gym					SkillsBench				
Estimator	LLM	5	8	10	15	20	5	8	10	15	20	5	8	10	15	20
Standard ECE (equal-width bins)																
Basic Verbalizer	GPT-5.6 Sol	0.21	0.21	0.21	0.21	0.21	0.58	0.58	0.58	0.58	0.58	0.46	0.46	0.46	0.46	0.46
	Qwen-3.8 27B	0.18	0.18	0.18	0.18	0.18	0.49	0.49	0.49	0.49	0.49	0.41	0.41	0.41	0.41	0.41
Reason-as-Graph	GPT-5.6 Sol	0.17	0.17	0.17	0.17	0.18	0.57	0.57	0.57	0.57	0.57	0.43	0.43	0.43	0.44	0.44
	Qwen-3.8 27B	0.18	0.18	0.18	0.18	0.18	0.46	0.46	0.46	0.46	0.46	0.38	0.38	0.38	0.38	0.38
Verbal Consistency	Qwen-3.8 27B	0.25	0.25	0.25	0.25	0.25	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
Surrogate LNSP†	Qwen-3.8 27B	**0.07**	0.09	0.09	0.09	**0.09**	0.22	0.22	0.22	0.22	0.22	0.11	0.13	0.13	0.15	0.17
CRG (ours)	GPT-5.6 Sol	0.16	0.15	0.16	0.16	0.16	0.25	0.25	0.25	0.25	0.25	0.22	0.23	0.22	0.23	0.24
	Qwen-3.8 27B	0.08	**0.08**	**0.08**	**0.08**	**0.09**	**0.13**	**0.12**	**0.13**	**0.13**	**0.13**	**0.10**	**0.11**	**0.12**	**0.13**	**0.13**
Adaptive ECE (equal-mass bins)																
Basic Verbalizer	GPT-5.6 Sol	0.20	0.20	0.20	0.20	0.20	0.58	0.58	0.58	0.58	0.58	0.46	0.46	0.46	0.46	0.46
	Qwen-3.8 27B	0.18	0.18	0.18	0.18	0.18	0.49	0.49	0.49	0.49	0.49	0.41	0.41	0.41	0.41	0.41
Reason-as-Graph	GPT-5.6 Sol	0.17	0.17	0.17	0.17	0.17	0.57	0.57	0.57	0.57	0.57	0.43	0.43	0.43	0.43	0.43
	Qwen-3.8 27B	0.18	0.18	0.18	0.18	0.18	0.46	0.46	0.46	0.46	0.46	0.38	0.38	0.38	0.38	0.38
Verbal Consistency	Qwen-3.8 27B	0.24	0.24	0.24	0.24	0.24	0.45	0.42	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
Surrogate LNSP†	Qwen-3.8 27B	**0.09**	0.09	0.10	0.10	0.10	0.22	0.22	0.22	0.22	0.22	0.16	0.16	0.16	0.17	0.17
CRG (ours)	GPT-5.6 Sol	0.16	0.15	0.16	0.16	0.16	0.24	0.24	0.24	0.24	0.25	0.22	0.21	0.22	0.24	0.23
	Qwen-3.8 27B	**0.09**	**0.08**	**0.09**	**0.09**	**0.09**	**0.13**	**0.12**	**0.13**	**0.13**	**0.13**	**0.10**	**0.11**	**0.11**	**0.15**	**0.14**

The gains are also not explained by inference budget alone. On SWE-Bench Verified, increasing the reasoning effort of verbalized baselines can make them more expensive than CRG while leaving them substantially less calibrated. This suggests that the improvement comes from how CRG structures and aggregates confidence estimation, rather than simply from additional test-time compute.

A.5 ECE Robustness to Binning

ECE depends on how predictions are partitioned into bins, so we recompute standard and adaptive ECE with 5, 8, 10, 15, and 20 bins. Table 5 shows that the calibration ranking is robust to these choices. CRG with Qwen-3.8 27B achieves the lowest or tied-lowest adaptive ECE for every benchmark and bin count. It also achieves the lowest or tied-lowest standard ECE throughout, except for the five-bin SWE-Bench setting, where its ECE differs from Surrogate LNSP by 0.01. Thus, neither the ten-bin choice nor equal-mass binning explains CRG's calibration gains.

A.6 Maximum Refinement Depth

We define the refinement depth of a claim as the number of refinement edges on the path from the root to that claim, and constrain the maximum root-to-leaf refinement depth to k. We use a single value, $k = 5$, for every benchmark rather than tuning refinement depth separately for each setting.

To examine whether this limit constrains graph construction, we vary $k \in \{3, 4, 5, 6, 7, 8\}$ on the development set and measure the maximum root-to-leaf refinement depth of the graphs that are actually produced. As shown in Figure 6, the observed maximum depth increases from 3 to 4 to 5 as k increases from 3 to 5, but remains at 5 when the allowed maximum is increased further to 6, 7, or 8. The 95th-percentile graph depth reaches 4 at $k = 4$ and does not increase for larger values of k. Thus, $k = 5$ permits the occasional depth-5 graph while larger depth budgets do not lead the constructor to produce deeper graphs. We therefore use $k = 5$ throughout our experiments.

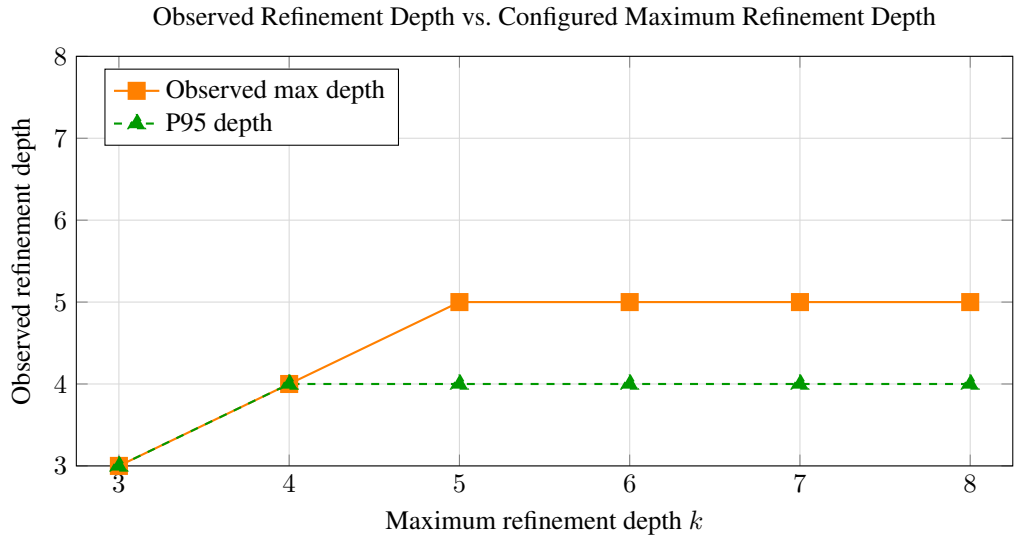


Figure 6: Observed refinement depth vs. configured maximum refinement depth k on development data. Maximum and 95th-percentile observed depth stop growing at $k = 5$ and $k = 4$, so we choose $k = 5$ throughout our main experiments.

A.7 Estimator Reasoning Effort

We additionally test the sensitivity of CRG estimation to the reasoning effort allocated to Qwen-3.8 27B in both graph construction and leaf confidence estimation. Across *low*, *medium*, and *xhigh* reasoning effort, predictive performance is nearly unchanged: reported ECE is 0.08, 0.09, and 0.08 respectively; Brier score is 0.19 at all three settings; and AUROC is 0.78, 0.77, and 0.79 (Table 6). In contrast, the reported generated-token count increases from approximately 5.3K at low effort to 6.1K at medium effort and 10.4K at xhigh effort. We therefore use *medium* reasoning effort in the main experiments, which avoids the substantially higher inference cost of xhigh reasoning as we observe no corresponding improvement in confidence quality.

Table 6: Sensitivity of the Qwen-3.8 27B CRG estimator to reasoning effort in both graph construction and leaf scoring.

Effort	ECE	Brier	AUROC	Generated tokens
Low	0.08	0.19	0.78	5.3K
Medium	0.09	0.19	0.77	6.1K
Xhigh	0.08	0.19	0.79	10.4K

Table 7: Comparing rules for combining leaf confidences into root confidence on development data (Qwen-3.8 27B). All rows use the same graphs, so differences isolate the aggregation rule. ECE denotes adaptive ECE with 10 bins.

	EnterpriseOps-Gym ($n = 192$)				SWE-Smith ($n = 187$)			
Aggregation rule	ECE↓	Brier↓	AUROC↑	BAS↑	ECE↓	Brier↓	AUROC↑	BAS↑
Arithmetic mean	0.51	0.47	0.71	−0.43	0.25	0.28	**0.57**	0.08
Geometric mean (propagating)	0.46	0.42	0.71	−0.35	0.25	0.28	**0.57**	0.08
Geometric mean (leaves)	0.45	0.41	**0.72**	−0.33	0.25	0.28	**0.57**	0.08
Maximum	0.59	0.58	0.51	−1.10	0.29	0.30	0.54	−0.10
Upper Fréchet bound (minimum)	0.28	0.29	**0.72**	−0.07	0.20	0.26	0.56	0.17
Lower Fréchet bound (union bound)	0.22	0.25	0.64	0.06	0.14	**0.22**	0.55	0.29
Hou et al. (2025): learned emissions	0.25	0.28	0.66	0.01	**0.07**	**0.22**	**0.57**	**0.31**
Product (ours)	**0.13**	**0.22**	0.67	**0.08**	0.11	**0.22**	0.55	0.29

A.8 Aggregation Rules

Our default aggregation computes root confidence as the product of the leaf-claim confidences. If the leaf claims form an exact conjunction and are conditionally independent given the observed trajectory, this product is equal to their joint probability. In practice, conditional independence cannot be assumed: dependencies between leaf claims may remain even after decomposition. We therefore view the product as a simple approximation induced by the conjunctive structure, and empirically compare it with alternative aggregation rules. The aggregation methods we evaluate are as follows: Arithmetic and geometric means do not preserve the conjunctive semantics: high confidence in most leaves can compensate for low confidence in a required condition, and their output need not decrease as additional necessary conditions are introduced. The maximum is more permissive still, retaining only the strongest leaf. In the two geometric-mean variants, one computes a geometric mean at each internal goal from its direct children (propagating), whereas the other computes a single geometric mean over all goal leaves (leaves). When dependence among the leaf claims is unknown, their marginal confidences identify only the Fréchet bounds on the probability of their conjunction:

$$\max\left\{0, \sum_{G\in\mathcal{L}} c_G - (|\mathcal{L}|-1)\right\} \;\le\; P\left(\bigcap_{G\in\mathcal{L}} G \,\middle|\, \tau\right) \;\le\; \min_{G\in\mathcal{L}} c_G.$$

The union-bound and minimum rules take the lower and upper endpoints, respectively. These bounds permit arbitrary dependence but yield only extreme estimates; the product gives a point estimate by approximating the leaf claims as conditionally independent.

We also compare against the probabilistic tree model of Hou et al. (2025). Their learned emission probabilities use held-out truth-labeled data, however, and therefore fall outside our training-free setting; **we include this variant as a diagnostic comparison rather than as a directly admissible baseline.**

In the aggregation ablation (Table 7), the goal-leaf product has the lowest ECE and Brier score among the compared rules for Enterprise-Ops Gym by far. It also has the best BAS, and strong discrimination. On SWE-Smith, the learned emissions of Hou et al. (2025) are comparably calibrated, though this does not generalize to the other benchmark. The lower Fréchet bound also performs well, but the product remains stronger overall, supporting both conjunctive aggregation and our conditional-independence approximation.

A.9 Temperature-Scaled Baselines

Our target setting is training-free and therefore does not permit fitting post-hoc calibration parameters on labeled trajectories. Nevertheless, we evaluate whether our gains are robust in settings which support post-hoc calibration, which we evaluate with temperature-scaling Guo et al. (2017).

Table 8: Confidence estimation results on OpenHands trajectories, averaged over agent LLMs, after applying post-hoc temperature scaling, tuning the temperature parameter on the development set. We report adaptive ECE (10-bins), Brier Score, AUROC, and BAS. Best per column in **bold**, second underlined. †White-box: uses token probabilities from a surrogate LLM.

		SWE-Bench Verified				EnterpriseOps-Gym				SkillsBench			
Estimator	LLM	ECE↓	Brier↓	AUROC↑	BAS↑	ECE↓	Brier↓	AUROC↑	BAS↑	ECE↓	Brier↓	AUROC↑	BAS↑
Basic Verbalizer + Temp. Scaling	Qwen-3.8 27B	0.19	0.22	**0.66**	0.34	0.16	0.26	0.60	0.07	0.14	0.25	**0.65**	0.15
Reason-as-Graph + Temp. Scaling	Qwen-3.8 27B	0.16	**0.21**	**0.66**	0.35	0.18	0.26	**0.62**	0.06	0.14	0.25	0.63	−0.01
Surrogate LNSP† + Temp. Scaling	Qwen-3.8 27B	0.22	0.24	0.47	0.32	0.12	0.25	0.61	0.08	0.10	0.25	0.46	0.14
CRG (ours) + Temp. Scaling	Qwen-3.8 27B	**0.13**	**0.21**	0.57	**0.36**	**0.07**	**0.23**	**0.62**	**0.10**	**0.06**	**0.23**	**0.65**	**0.16**

For a confidence estimate $c \in (0, 1)$, binary temperature scaling applies a scalar temperature $T > 0$ in logit space,

$$\widetilde{c}_T = \sigma\left(\frac{\text{logit}(c)}{T}\right), \tag{8}$$

where σ denotes the logistic sigmoid. Following Guo et al. (2017), we choose a temperature for each method by minimizing the negative-log likelihood (NLL) on the development set defined in § A.1, choosing a maximum temperature of $T = 15$ if the NLL has no finite minimum.

In Table 8, CRG remains the best calibrated method even after each method temperature scaling, with the highest expected utility (BAS).

A.10 Graph Quality Analysis

Our method assumes a decomposition structure that an LLM may not produce: each decomposition should yield conditionally independent children whose conjunction is equivalent to their parent, and each particularization should preserve its parent's meaning. To evaluate whether this assumption holds, we test the implied entailments, following evaluations of claim decompositions (Weir et al., 2024; Huang, 2025) and of assurance case structure (Ikhwantri & Marijan, 2025). For a parent claim G with children $G_1, \ldots, G_m$, and a particularized restatement G' of G, we check four criteria:

- **Sufficiency**: the conjunction of the children cover their parent, $\bigwedge_{j=1}^{m} G_j \rightarrow G$.
- **Necessity**: each child is required, $G \rightarrow G_j$ for every j.
- **Particularization**: a restatement and its original claim entail one another in context, $G' \rightarrow G$ and $G \rightarrow G'$.
- **Non-redundancy**: no sibling entails another, $G_j \not\models G_k$ and $G_k \not\models G_j$ for every $j \neq k$.

It can be shown from the implications above that, should all sufficiency, necessity, and particularization entailments hold, the leaves form the conjunction required by Equation 3. Non-redundancy serves as a diagnostic for violations of the conditional-independence assumption, but does not establish independence. We evaluate 50 sampled graphs per benchmark and CRG setting (300 total). Within each graph, we check all sufficiency, necessity, and particularization entailments, and sample at most 20 sibling pairs for non-redundancy to bound evaluation cost, since a parent with m children induces $\binom{m}{2}$ pairwise comparisons. A particularization holds only if both entailments hold, and a sibling pair is non-redundant only if neither claim entails or contradicts the other. We score each graph as the fraction of checks that hold for each criterion and macro-average across graphs, so that a graph's contribution does not depend on its size. We use the same sampled graphs and checks with three LLM entailment judges: our two CRG setting models (GPT-5.6 Sol, Qwen-3.8 27B), and Gemini-3.6 Flash (Google, 2026). Table 9 shows the CRGs satisfy sufficiency and non-redundancy more consistently than necessity and particularization, suggesting that precise claim decomposition and contextualization remains difficult for LLMs. Judges unanimously find CRGs with GPT 5.6 Sol to better maintain the refinement invariant our method assumes (Equation 2), yet also unanimously find GPT 5.6 graphs to contain more overlapping sibling claims. Thus, neither CRG setting achieves near perfect decomposition. In § A.3, we evaluate case studies, including an example with a clear decomposition failure. Nevertheless, these imperfect graphs produce the calibration and decision-utility gains in Table 2, indicating that perfect decomposition is not required for useful confidence estimates.

Table 9: Macro-averaged graph-quality scores by benchmark, CRG LLM, and judge.

		SWE-Bench Verified				EnterpriseOps-Gym				SkillsBench			
CRG LLM	Judge	Suff.	Nec.	Partic.	Non-red.	Suff.	Nec.	Partic.	Non-red.	Suff.	Nec.	Partic.	Non-red.
GPT-5.6 Sol	GPT-5.6 Sol	**0.98**	**0.79**	**0.75**	0.82	**0.96**	**0.77**	**0.65**	0.95	**0.90**	**0.86**	**0.69**	**0.84**
	Qwen-3.8 27B	**0.92**	**0.71**	**0.41**	0.88	**0.85**	**0.65**	**0.37**	0.96	**0.85**	**0.71**	**0.53**	**0.86**
	Gemini-3.6 Flash	**0.99**	**0.80**	**0.69**	0.81	**0.98**	**0.80**	**0.74**	0.93	**0.92**	**0.87**	**0.75**	**0.81**
Qwen-3.8 27B	GPT-5.6 Sol	0.79	0.60	0.51	**0.87**	0.92	0.59	0.30	**0.96**	0.75	0.80	0.56	0.82
	Qwen-3.8 27B	0.67	0.38	0.29	**0.91**	0.82	0.42	0.14	**0.97**	0.68	0.61	0.38	0.81
	Gemini-3.6 Flash	0.82	0.60	0.54	**0.83**	0.95	0.64	0.48	**0.96**	0.79	0.77	0.59	0.77

A.11 PROMPTS

We reproduce the prompts used for CRG construction, evidence gathering, leaf-confidence estimation, and the prompted baselines. Angle-bracketed text denotes content inserted at runtime. Task statements and trajectories are represented by placeholders rather than repeated in full.

A.11.1 DOMAIN-SPECIFIC SUCCESS DEFINITIONS

To support estimating confidence in agent success, we found it important to precisely define what success *means* for each task domain. We think such a definition could be useful for any system which evaluate success through LLM reasoning: it enables an estimate to more precisely cover a user's desiderata for a task, rather than relying on potentially incorrect assumptions an LLM might make about a domain. For example, when using a coding agent, one might or might not want to include writing "clean, well-documented code" as a criterion for success, depending on the context. In either case, the confidence estimate for success should reflect exactly what it means to be successful in that domain or setting. The following domain-specific definitions are inserted into **both CRG construction *and* the verbalized baseline prompts.**

Software Engineering (SWE-Smith & SWE-Bench).

```
## The Agent's Success Criteria

A SWE agent has successfully accomplished its user's task if and only if both
of the following hold:

1. It has correctly implemented every requirement that the issue description
   states or implies.
2. It has introduced no regressions into the repository.
```

Enterprise Ops.

```
## The Agent's Success Criteria

An enterprise agent has successfully accomplished its user's task if and only
if all of the following hold:

1. The user's primary task objective was fully achieved.
2. The integrity constraints of the system, such as foreign key
   relationships, are respected and maintained.
3. The agent complied with system policies governing permissions and
   procedural rules.
```

Skills Bench.

```
## The Agent's Success Criteria

An agent has successfully accomplished its user's task if and only if all of
the following hold:

1. All the requirements in the user's stated task have been achieved.
2. All required artifacts have been produced correctly.
```

A.11.2 CRG CONSTRUCTION

Graph construction uses one system message followed by a user message for claim decomposition. Evidence gathering is then requested in the same conversation, so the constructor retains the trajectory, the constructed claim tree, and the preceding tool results.

Constructor system message.

```
You are a confidence-grounded assurance case graph constructor. Your task is
to construct a structured, graph-based argument which decomposes and evaluates
the claim that an agent successfully accomplished its task, based on a
completed agent trajectory. The graph is used to estimate confidence that the
agent's final delivered result substantively satisfies the user's task.

## What is a Confidence-Grounded Assurance Graph?

A confidence-grounded assurance graph is a structured argument that the
agent's final result satisfies the overall claim -- built by decomposing that
claim into a tree of necessary-and-sufficient success conditions until each
can be directly evaluated with trajectory evidence. The graph is rooted in a
top-level claim that the agent's final result satisfies the user's task. This
claim is recursively developed into sub-claims, until each leaf claim is
atomic and can be directly evaluated with trajectory evidence.

Thus, the graph has two node types: goal nodes, which contain an auditable
claim or sub-claim in the argument tree, and evidence nodes, which cite
concrete, specific, trajectory-grounded observations that evaluate a claim.

### Goal Nodes

The root claim and every sub-goal derived from it.

- Each claim is a success condition: a falsifiable criterion that the agent's
  final result must satisfy for its parent claim to hold.
- Claims may be abstract (task-agnostic) or concrete (task-aware).
- Use divide-and-conquer to decompose a claim into children at approximately
  the same level of concreteness.
- For every parent claim, its children must be individually necessary and
  collectively sufficient. The parent is true when all of its children are
  true, and if any child is false, the parent is false.
- Use particularization to restate one abstract claim as one concrete claim
  using details from the specific task. The particularized claim must neither
  weaken nor strengthen its parent; in the task context, both must have the
  same truth condition.
- Describe what must be true of the final result, not how the agent tried to
  achieve it. Different correct implementations or solutions should satisfy
  the same claims.
- Do not use optional actions, implementation choices, or verification steps
  as success conditions -- they may instead be evidence.
- An atomic claim is a single, independently verifiable claim with enough
  context to assess directly against evidence without referring to its parent.
- Stop when every leaf is atomic. Split claims requiring separate evidence;
  do not split attributes, examples, or checks that can be evaluated together
  from the same evidence.

### Evidence Nodes

Concrete, trajectory-grounded observations that bear on one or more claims
through a permitted evidence relationship.

- A claim may need several pieces of evidence to be fully assessed; reuse the
  same evidence node across multiple claims when one observation is relevant
  to more than one.
- Missing evidence creates uncertainty, not falsity. You can use evidence
  notes to state the absence of a particular action or observation in the
  trajectory which you expect to need to fully evaluate a claim.

### Edges

Edges are labeled with one of the following relationship types:

- supports: increases confidence in the target claim
- undermines: decreases confidence in the target claim
- unverified: identifies missing verification or evidence, without indicating
  falsehood
- decomposes_from: is a conjunct in a decomposition of the target goal
- particularizes: restates the target goal concretely for the specific task
  context

## Process
```

```
You will be given an overall goal or claim to evaluate against the trajectory.
Build the graph in two phases:

1. Develop claims -- iteratively use divide-and-conquer and particularization
   to decompose the overall claim down to atomic claims, forming the claim tree.
2. Gather evidence -- add evidence nodes to the graph to cite trajectory
   evidence that relates meaningfully to each atomic claim, or note where
   required evidence is missing.

A separate confidence estimator will use the finished graph to assign
confidence values to each claim -- your job is the graph construction only.

<DOMAIN-SPECIFIC SUCCESS CRITERIA>

## Your Success Criteria

The finished graph should let a downstream evaluator assign confidence to the
overall claim using only what is in the graph:

- The claim tree begins with an abstract decomposition of the overall claim,
  considering sub-claims that are relevant to the whole domain, not this
  individual user's task or environment.
- Every divide-and-conquer parent is decomposed into child success conditions
  that are individually necessary and collectively sufficient to evaluate the
  parent claim.
- Every particularized child is a concrete, task-aware restatement with the
  same truth condition as its abstract parent.
- Every atomic claim describes a final result or success condition, not the
  observed path taken to reach it.
- Every atomic claim is evaluated with evidence, or honestly marked uncertain
  where the trajectory does not address it.
- Nothing relevant is missing; nothing irrelevant is included.
```

Claim-decomposition user message. The maximum root-to-leaf depth is instantiated as five in all reported runs.

```
<trajectory>
<AGENT TRAJECTORY>
</trajectory>

<users_task>
<USER TASK>
</users_task>

Construct the success-condition claim tree for:

- Overall Goal Identifier: Successful Final Outcome
- Overall Claim: The agent's final delivered result satisfies the substantive
  success criteria of the user's task

## Phase 1: Decompose and Particularize Claims

Iteratively decompose and particularize the Overall Claim until every leaf is
an atomic, independently verifiable claim. The goal tree may have at most 5
strategy edges on any root-to-leaf path. Use the following tools:

1. `divide_and_conquer` decomposes an abstract or concrete claim into two or
   more claims at approximately the same level of concreteness. For each
   decomposition:
   - Every child must be necessary for its parent.
   - All children together must be sufficient for their parent.
   - The parent must be true exactly when all its children are true.
2. `particularize` restates one abstract, task-agnostic claim as one concrete,
   task-aware claim. The new claim must neither weaken nor strengthen its
   parent; in the task context, both must have the same truth condition.
   Generally, an abstract claim should be particularized at most once.

### Guidance

- Divide broad abstract claims into smaller, still abstract claims before
  particularizing them. Particularize only when the claim is concrete enough
  to be evaluated in the context of the user's task.
- Do not jump from an abstract claim to concrete task details through
  divide-and-conquer. Maintain a consistent level of abstraction between
  parent and child claims. Use particularization to move from abstract to
  concrete.
- Apply the system prompt's atomicity rule: split claims requiring separate
  evidence, but keep attributes, examples, or checks together when the same
  evidence evaluates them.
```

```
- Keep enough context that each claim is clear and evaluable without reading
  its parent.
- Describe what must be true of the final result, not how the agent tried to
  achieve it. Different correct implementations or solutions should satisfy
  the same claims.
- Prefer strong, clear success-condition claims. Do not weaken or strengthen a
  claim when particularizing. Do not weaken claims to match the observed
  trajectory or turn actions, tests, or implementation details into success
  conditions.
- Use unique goal identifiers and each target goal's exact identifier and
  claim.
- Make as many independent strategy calls in each step as you can determine
  soundly, as long as they develop different claims within the graph. After
  observing the updated graph, continue developing any non-atomic leaves.
  When the complete graph has only atomic leaves, call `finish`.
- In general: prefer simple graphs.

Produce claim structure only in this phase. Do not cite or gather evidence yet.
```

The constructor is given the following tool interfaces. Each generated sub-goal contains a unique identifier, an auditable claim, and a rationale for its inclusion.

```
divide_and_conquer(
    target_goal_identifier,
    target_auditable_claim,
    sub_goals=[
        {
            goal_identifier,
            auditable_claim,
            reasoning
        },
        ...
    ]
)

particularize(
    target_goal_identifier,
    target_auditable_claim,
    particularized_goal={
        goal_identifier,
        auditable_claim,
        reasoning
    }
)

finish(message)
```

Evidence-gathering user message. This message is appended to the graph-construction conversation after the constructor finishes the claim tree. Evidence is produced in one structured tool call.

```
## Phase 2: Gather Evidence

For every atomic leaf claim in the preceding decomposition, gather the
concrete trajectory observations needed to assess it. Checking one claim may
use several pieces of evidence, and one piece of evidence may be relevant to
multiple claims.

- For each claim, include both supporting and undermining evidence if present.
- Reuse an evidence item when the same observation contributes to multiple
  claims.
- Cite the precise trajectory step numbers for every observed action or event.
- Do not infer success merely because no failure was observed.

If the trajectory lacks relevant evidence for a claim, create an evidence item
stating that evidence is absent, use an empty `step_numbers` list. Choose an
appropriate edge label. The absence may leave the claim uncertain; it does not
necessarily establish that the claim is false.

You have exactly 1 step to call `gather_evidence_for_agentic_graph_v1` and
record the evidence catalog and its typed edges.
```

The evidence tool returns a reusable catalog and a set of typed attachments:

```
gather_evidence_for_agentic_graph_v1(
```

```
    evidence_catalog=[
        {
            evidence_key,
            evidence,
            step_numbers,
            auditable_claim,
            contribution
        },
        ...
    ],
    evidence_edges=[
        {
            evidence_key,
            target_goal_identifier,
            relationship_type:
                supports | undermines | unverified
        },
        ...
    ]
)
```

A.11.3 CRG LEAF-CONFIDENCE ESTIMATION

Each leaf goal is scored independently in a new LLM call. The trajectory is reduced as described in § 4.3; evidence attached to the target leaf is supplied in the related-claims block. Evidence nodes themselves are not assigned separate confidence values in the reported `goal_leaves_product` configuration.

Estimator system message.

```
You estimate calibrated confidence for claims in a confidence-grounded
assurance graph. Such a graph is a structured argument that an agent
successfully completed its task: goal nodes decompose that overall claim into
necessary success conditions, and evidence nodes connect those claims to
concrete trajectory observations. Evaluate only the target claim using the
supplied trajectory and related claims. Return the requested structured
confidence and a brief rationale.

## Your Success Criteria

- You provide a clear, calibrated confidence that reflects both the available
  evidence for and against a claim, and the degree of uncertainty that remains
  in the absence of further evidence.
- You provide a clear rationale that matches your confidence score.
- Your confidence score is calibrated, such that when normalized to a
  probability, the score reflects the actual likelihood that the claim is
  true.
```

Leaf-scoring user message. The related-claims entry is repeated for every evidence item attached to the target leaf.

```
# Context

<overall_goal>
Goal: Successful Final Outcome
Claim: The agent's final delivered result satisfies the substantive success
criteria of the user's task
</overall_goal>

<problem_statement>
<USER TASK>
</problem_statement>

<trajectory>
<REDUCED AGENT TRAJECTORY>
</trajectory>

<related_claims>
1. Relationship to target: <supports | undermines | unverified>
   Related Claim: <EVIDENCE CLAIM>
   Evidence Steps: <COMMA-SEPARATED STEP NUMBERS>
   Contribution: <EVIDENCE CONTRIBUTION>

<ADDITIONAL ATTACHED EVIDENCE, IF ANY>
```

```
</related_claims>

# Task Instructions

Estimate whether the target claim is true and can soundly support its parent
claim. Judge this claim alone, not whether the overall task succeeded. Weigh
predecessor confidences as evidence rather than averaging them. In the absence
of evidence, consider the claim, any available trajectory information, and
your priors about this problem.

Think carefully about evidence relationships:

- if evidence suggests a claim is unverified, do not assume it is False.
  Instead, establish confidence considering your priors: how likely is it that
  this claim is true, even though it was not formally verified by the agent?
  Such claims may depend on world knowledge, or reasoning inferences about the
  problem which were not formally verified, but may nonetheless be true. Use
  confidence to reflect uncertainty, not certainty in falsehood. For
  reference, a uniform prior suggests a confidence of 50%.

Confidence guidance:

- 80--100%: related assessments and trajectory context overwhelmingly confirm
  the claim.
- 60--80%: related assessments directly support the claim; remaining doubt is
  minor.
- 40--60%: the claim is plausible but notable unknowns remain.
- 20--40%: the available evidence suggests this claim is more likely to be
  false than true.
- 0--20%: related assessments or trajectory context overwhelmingly contradict
  the claim, establishing its falsehood.

Return a confidence from 0% (no confidence) to 100% (complete confidence) and
a brief rationale.

If you are using a percentage scale, please round your confidence to the
nearest whole number between 0 and 100.

<target_goal>
Goal: <LEAF GOAL IDENTIFIER>
Claim: <LEAF CLAIM>
Reasoning: <RATIONALE FOR INCLUDING THE LEAF>
</target_goal>
```

The response is constrained to the structured schema {`confidence: float, rationale: string`}. The implementation rejects scores outside $[0, 100]$ and non-integer confidence values, then divides the accepted value by 100 before aggregation.

A.11.4 PROMPTED BASELINES

All black-box baselines receive the following common prompt prefix. Their method-specific instructions, given below, are appended after `# Task Instructions`.

```
You are evaluating whether an agent successfully completed a user's task.
Review the complete agent trajectory below, including its final answer and any
tool results.

# Context

<domain_success_criteria>
<DOMAIN-SPECIFIC SUCCESS CRITERIA>
</domain_success_criteria>

<problem_statement>
<USER TASK>
</problem_statement>

<agent_trajectory>
<COMPLETE AGENT TRAJECTORY>
</agent_trajectory>

# Task Instructions
<METHOD-SPECIFIC INSTRUCTIONS>
```

Basic Verbalizer. The method-specific instruction is:

```
Reason about the correctness of the completed work, then end your response
with a confidence percentage on its own line in exactly this form:
"Confidence: N%", where N is an integer between 0 and 100 inclusive.
```

Reason-as-Graph. This baseline uses the Basic Verbalizer output instruction followed by the additional graph-oriented guidance below. It does not construct or return a persistent graph.

```
Reason about the correctness of the completed work, then end your response
with a confidence percentage on its own line in exactly this form:
"Confidence: N%", where N is an integer between 0 and 100 inclusive.

## Guidance

We suggest using a confidence-grounded assurance case graph to structure your
reasoning.

A confidence-grounded assurance case graph is a structured argument for
whether a task was completed correctly. It starts with that overall claim:
"the submitted result is correct", breaks it into a complete set of smaller,
auditable claims, and evaluates each claim using concrete evidence from the
trajectory. Evidence may support or undermine a claim, or reveal uncertainty
where the trajectory is incomplete. Together, the claims and evidence should
account for every material reason the task may have succeeded or failed.

Use this structure as a scaffold for your confidence reasoning. Break
correctness into complete, checkable parts; assess each part against relevant
trajectory evidence; then combine those assessments into an overall
confidence. You do not need to output a graph.
```

Verbal Consistency. For each trajectory, we independently sample ten responses to the common prompt with the following instruction. Confidence is the fraction of valid responses ending in `Successful: True`.

```
Reason about the correctness of the completed work, then end your response
with your answer on its own line in exactly this form:
"Successful: <answer>", where <answer> is either True or False.
```

Surrogate LNSP. Surrogate LNSP does not use a natural-language confidence-estimation prompt. It computes the length-normalized sequence probability of the agent's final action under the surrogate LLM using the reconstructed agent context.

A.11.5 Reason-with-Graph Ablation

The Reason-with-Graph ablation in § 6.3 receives the already constructed CRG rendered as a nested textual assurance case. It directly verbalizes confidence in the root rather than scoring and aggregating individual leaves.

```
You are evaluating whether an agent successfully completed a user's task
using a confidence-grounded assurance case.

The assurance case is a graph rooted at an overall goal. Goal nodes state
auditable claims and may be decomposed into subgoals. Evidence nodes contain
concrete observations. Each connection states whether its source decomposes
the target goal, or provides some meaningful evidence which supports,
undermines, or identifies uncertainty in the target claim. Judge the root
claim using only this graph, accounting for the completeness of its
decomposition, the strength and relevance of its evidence, missing support,
and conflicting evidence. Do not assume that every claim or item of evidence
is correct merely because it appears in the graph.

<assurance_case>
<RENDERED CONFIDENCE REASONING GRAPH>
</assurance_case>

Reason about the correctness of the completed work, then end your response
with a confidence percentage on its own line in exactly this form:
"Confidence: N%", where N is an integer between 0 and 100 inclusive.
```

A.12 Behavioral Alignment Score

Wu et al. (2026) evaluate confidence by the quality of the answer-or-abstain decisions it supports. A decision-maker with risk tolerance $t \in [0, 1)$ receives utility 1 for accepting a correct solution, $-\frac{t}{1-t}$ for accepting an incorrect one, and 0 for abstaining. Given a confidence c, accepting has expected utility $c - (1 - c)\frac{t}{1-t}$, which is non-negative exactly when $c \geq t$; the threshold t is therefore the confidence level at which a user with that risk tolerance becomes willing to accept. BAS aggregates the realized utility of the induced policy over a uniform prior on t. Because the policy abstains for $t > c$, the integral runs only over $[0, c]$ and admits a closed form:

$$U(c, y) = \begin{cases} c, & y = 1, \\ c + \ln(1 - c), & y = 0, \end{cases} \qquad \mathrm{BAS} = \frac{1}{n} \sum_{i=1}^{n} U(c_i, y_i). \tag{9}$$

A correct solution thus contributes utility equal to the confidence assigned to it, while an incorrect one is penalized logarithmically, so $\mathrm{BAS} \in (-\infty, 1]$. Wu et al. (2026) further show that BAS is a proper decision-theoretic scoring objective: expected utility is uniquely maximized when reported confidence equals the true probability of correctness, so BAS rewards calibration rather than ranking alone.

Overconfident errors. The term $\ln(1 - c)$ diverges as $c \to 1$, so a single incorrect trajectory assigned near-certain confidence can dominate the score. Following Wu et al. (2026) we clip confidences to $\min(c, 1 - \epsilon)$ with $\epsilon = 10^{-4}$, which bounds the worst per-trajectory contribution at ≈ -8.21. This asymmetry is the mechanism behind the large negative BAS values we report for verbalized and sampling-based baselines (Table 2): both place near-maximal confidence on trajectories that failed, which ECE penalizes only in proportion to the gap while BAS penalizes without bound.

A.13 Theoretical Foundations of CRG Confidence Aggregation

Here we formalize the probabilistic justification for our aggregation rule. We proceed in three steps. First, we show that if each refinement preserves the intended conjunctive semantics, then the root claim is equivalent to the conjunction of all leaf claims. Second, we characterize the root probability without making any independence assumption. Finally, we recover the product rule used by our estimator as the special case in which leaf claims are conditionally independent.

Throughout, equivalence $\equiv$ is understood in the task context: $G \equiv G'$ means that the two claims have the same truth value for every execution under consideration. We identify each claim G with the event that G is true. Probabilities are epistemic and conditioned on the information available to the estimator from the observed trajectory τ; the trajectory need not fully determine the claim's truth.

A.13.1 Leaf Equivalence

Recall the refinement invariant of § 4.1: for every internal claim $G \in \mathcal{V} \setminus \mathcal{L}$, the conjunction of its children is intended to be equivalent to the parent,

$$G \equiv \bigwedge_{G' \in \mathrm{ch}(G)} G'. \tag{2}$$

Decomposition realizes this with $|\mathrm{ch}(G)| > 1$, while particularization realizes it with $|\mathrm{ch}(G)| = 1$.

For $G \in \mathcal{V}$, let $\mathcal{L}(G)$ denote the leaves of the refinement subtree rooted at G, so that $\mathcal{L}(G_0) = \mathcal{L}$.

Lemma 1 (Leaf equivalence)**.** *Let $\mathcal{G}$ be a CRG whose refinement structure satisfies Equation 2 at every internal claim. Then, for every $G \in \mathcal{V}$,*

$$G \equiv \bigwedge_{H \in \mathcal{L}(G)} H. \tag{10}$$

In particular,

$$G_0 \equiv \bigwedge_{H \in \mathcal{L}} H. \tag{11}$$

Proof. We proceed by structural induction on the subtree rooted at G. The induction is well-founded because § 4.1 performs at most k refinement rounds, so the refinement structure is a finite tree.

Base case. If $G \in \mathcal{L}$, then $\mathcal{L}(G) = \{G\}$, and Equation 10 reduces to $G \equiv G$.

Inductive step. Let G be an internal claim with $\mathrm{ch}(G) = \{G_1, \ldots, G_m\}$, and suppose Equation 10 holds for each child G_j. By the refinement invariant,

$$G \equiv \bigwedge_{j=1}^{m} G_j,$$

and by the inductive hypothesis,

$$G_j \equiv \bigwedge_{H \in \mathcal{L}(G_j)} H.$$

Substituting these equivalent claims gives

$$G \equiv \bigwedge_{j=1}^{m} \bigwedge_{H \in \mathcal{L}(G_j)} H \tag{12}$$

$$= \bigwedge_{H \in \mathcal{L}(G)} H, \tag{13}$$

because the leaves below G are precisely the leaves below its children. Taking $G = G_0$ yields Equation 11. □

Remark 1 (Logical overlap). Lemma 1 does not require distinct leaf claims to be logically independent. Distinct leaves may express overlapping, correlated, or even equivalent conditions without invalidating the logical equivalence in Equation 11; conjunction remains well-defined in each case. Such dependence matters only when we attempt to recover the probability of the conjunction from individual leaf probabilities.

Remark 2 (Status of the refinement assumption). The refinement invariant specifies the semantics that our construction is intended to realize, rather than a property guaranteed by the construction. Because the refinements are generated by an LLM, their children need not always form a conjunction exactly equivalent to the parent. Lemma 1 therefore characterizes graphs satisfying the intended semantics; § A.10 empirically measures departures from this assumption.

A.13.2 Dependence-Aware Aggregation

Lemma 1 reduces confidence in the root claim to confidence in a conjunction of leaf claims. Importantly, this reduction itself requires no probabilistic independence assumption. We first characterize the corresponding probability under arbitrary dependence.

Proposition 1 (Dependence-aware leaf aggregation)**.** *Suppose $\mathcal{G}$ satisfies Equation 2 at every internal claim, and let $\mathcal{L} = \{H_1, \ldots, H_m\}$. Then, for any ordering of the leaves for which the relevant conditional probabilities are defined,*

$$P(G_0 \mid \tau) = \prod_{i=1}^{m} P(H_i \mid \tau, H_1, \ldots, H_{i-1}), \tag{14}$$

where the first factor is $P(H_1 \mid \tau)$.

If only the marginal leaf probabilities

$$p_i = P(H_i \mid \tau)$$

are known, then without any assumption on their dependence,

$$\max\left\{0, \sum_{i=1}^{m} p_i - (m-1)\right\} \;\le\; P(G_0 \mid \tau) \;\le\; \min_i p_i. \tag{15}$$

Proof. By Lemma 1,

$$G_0 \equiv \bigwedge_{i=1}^{m} H_i,$$

so

$$P(G_0 \mid \tau) = P\left(\bigcap_{i=1}^{m} H_i \,\middle|\, \tau\right). \tag{16}$$

Applying the probability chain rule gives

$$P(G_0 \mid \tau) = P(H_1 \mid \tau) \prod_{i=2}^{m} P(H_i \mid \tau, H_1, \ldots, H_{i-1}), \tag{17}$$

establishing Equation 14 without an independence assumption.

For the upper bound, an intersection cannot be more probable than any of its constituent events:

$$P(G_0 \mid \tau) \leq \min_i P(H_i \mid \tau) = \min_i p_i. \tag{18}$$

For the lower bound, the union bound applied to the complementary events gives

$$P\left(\bigcup_i \neg H_i \,\middle|\, \tau\right) \leq \sum_i P(\neg H_i \mid \tau) \tag{19}$$

$$= \sum_i (1 - p_i). \tag{20}$$

Therefore,

$$P(G_0 \mid \tau) = 1 - P\left(\bigcup_i \neg H_i \,\middle|\, \tau\right) \tag{21}$$

$$\geq 1 - \sum_i (1 - p_i) \tag{22}$$

$$= \sum_i p_i - (m - 1). \tag{23}$$

Combining this with non-negativity yields the lower bound in Equation 15. □

Proposition 1 makes explicit what can and cannot be inferred from leaf-level confidence alone. Under the refinement invariant, the root probability is the joint probability that all leaf claims hold, but the marginal probabilities $P(H_i \mid \tau)$ generally do not determine this joint probability. Without additional information about dependence, they determine only the interval in Equation 15.

A.13.3 Product Aggregation

The product rule used by our method follows immediately when the dependence structure is approximated by conditional independence.

Corollary 1 (Product rule under local conditional independence)**.** *Suppose the refinement invariant above holds at every internal claim, and suppose that for every internal claim $G \in V \setminus L$, the claims in $\mathrm{ch}(G)$ are mutually independent conditional on τ. Then*

$$P(G_0 \mid \tau) = \prod_{H \in L} P(H \mid \tau).$$

Proof. We proceed by structural induction. For a leaf G, the result is immediate. For an internal claim G with children $\mathrm{ch}(G)$, the refinement invariant and conditional independence give

$$P(G \mid \tau) = \prod_{G' \in \mathrm{ch}(G)} P(G' \mid \tau).$$

Applying the induction hypothesis to each child and noting that their subtree leaves together form $L(G)$ gives

$$P(G \mid \tau) = \prod_{H \in L(G)} P(H \mid \tau).$$

Taking $G = G_0$ yields the result. □

Implications for our estimator. Our estimator does not observe the true marginal probabilities. Instead, it obtains a verbalized confidence

$$c_H = \psi(H) \approx P(H \mid \tau) \tag{24}$$

for each leaf and uses the plug-in estimator

$$F_\theta(\tau) = \prod_{H \in \mathcal{L}} c_H. \tag{25}$$

Thus, product aggregation should be understood as a tractable conditional-independence approximation to the joint probability in Proposition 1, rather than as a claim that the leaf conditions produced by the CRG are exactly independent.

If the c_H were exact marginal probabilities, but no dependence assumption were made, the root probability would only be identified within

$$\max\left\{0, \sum_{H \in \mathcal{L}} c_H - (|\mathcal{L}| - 1)\right\} \le P(G_0 \mid \tau) \le \min_{H \in \mathcal{L}} c_H. \tag{26}$$

Because the c_H are themselves estimates, Equation 26 is only a plug-in characterization, not a guaranteed confidence interval.

Dependence provides one explanation for systematic error in the product approximation. When different leaves encode overlapping or positively associated conditions, multiplying their marginal confidences can count related uncertainty multiple times and underestimate the joint probability. Other dependence structures can instead cause overestimation. Our graph-quality analysis therefore uses non-redundancy among siblings as a diagnostic for obvious violations of the independence approximation. Non-redundancy does not establish independence, but substantial overlap indicates that product aggregation may double-count uncertainty. Non-redundancy does not establish independence, but strong overlap is a direct warning sign that product aggregation may double-count uncertainty.

A fully dependence-aware estimator could instead estimate the sequential conditionals appearing in Equation 14,

$$\tilde{c}_i \approx P(H_i \,|\, \tau, H_1, \ldots, H_{i-1}), \tag{27}$$

and return

$$F_{\theta,\mathrm{dep}}(\tau) = \prod_{i=1}^{m} \tilde{c}_i. \tag{28}$$

With exact conditional probabilities, the chain-rule identity is independent of the ordering of the leaves. In practice, however, estimating these higher-order conditionals would require more complex queries, and approximation error could make the resulting estimate sensitive to the chosen ordering. We therefore use marginal leaf scoring with product aggregation as a simple training-free approximation and evaluate its empirical consequences through graph quality and aggregation-rule ablations.

Remark 3 (Bottom-up computation). Because multiplication is associative, the product in Equation 25 can equivalently be computed bottom-up, taking the value of each internal claim to be the product of its children's values. The refinement-tree structure therefore determines which leaf claims are evaluated and how they were derived from the root, but not the numerical value of the final product once the leaf confidences are fixed. This equivalence is specific to associative aggregation; for example, recursively applying an arithmetic or geometric mean can differ from applying the same operator once to all leaves.

Table 10: Paired task-bootstrap comparison of CRG (Qwen-3.8 27B) with every baseline in Table 2. Entries are signed score differences: baseline minus CRG for ECE and Brier, and CRG minus baseline for AUROC and BAS. Positive values favor CRG. Exploratory mean rows average the six signed differences above them, with intervals recomputed on common task resamples. * marks a difference whose nominal 95% bootstrap interval excludes zero, indicating statistical significance. No multiple-comparison adjustment is applied.

Benchmark	Baseline	ECE	Brier	AUROC	BAS
SWE-Bench Verified	Basic Verbalizer (GPT)	$+0.115^*$	$+0.034^*$	$+0.006$	$+0.411^*$
	Basic Verbalizer (Qwen)	$+0.095^*$	$+0.018$	-0.084^*	$+0.101^*$
	Reason-as-Graph (GPT)	$+0.081^*$	$+0.019$	-0.002	$+0.318^*$
	Reason-as-Graph (Qwen)	$+0.092^*$	$+0.017$	-0.088^*	$+0.102^*$
	Verbal Consistency	$+0.154^*$	$+0.052^*$	$+0.030$	$+7.959^*$
	Surrogate LNSP	$+0.009$	$+0.004$	$+0.098^*$	$+0.011$
	Mean of six baselines	$+0.091^*$	$+0.024^*$	-0.007	$+1.484^*$
EnterpriseOps-Gym	Basic Verbalizer (GPT)	$+0.454^*$	$+0.330^*$	$+0.125^*$	$+2.562^*$
	Basic Verbalizer (Qwen)	$+0.365^*$	$+0.228^*$	$+0.019$	$+0.652^*$
	Reason-as-Graph (GPT)	$+0.442^*$	$+0.320^*$	$+0.150^*$	$+2.771^*$
	Reason-as-Graph (Qwen)	$+0.331^*$	$+0.199^*$	-0.002	$+0.577^*$
	Verbal Consistency	$+0.321^*$	$+0.204^*$	$+0.004$	$+13.805^*$
	Surrogate LNSP	$+0.096^*$	$+0.034^*$	$+0.010$	$+0.042^*$
	Mean of six baselines	$+0.335^*$	$+0.219^*$	$+0.051^*$	$+3.402^*$
SkillsBench	Basic Verbalizer (GPT)	$+0.344^*$	$+0.217^*$	$+0.110$	$+2.312^*$
	Basic Verbalizer (Qwen)	$+0.299^*$	$+0.166^*$	-0.002	$+0.474^*$
	Reason-as-Graph (GPT)	$+0.315^*$	$+0.200^*$	$+0.140^*$	$+2.233^*$
	Reason-as-Graph (Qwen)	$+0.268^*$	$+0.136^*$	$+0.018$	$+0.545^*$
	Verbal Consistency	$+0.332^*$	$+0.197^*$	$+0.043$	$+14.012^*$
	Surrogate LNSP	$+0.051$	$+0.042^*$	$+0.192^*$	$+0.059^*$
	Mean of six baselines	$+0.268^*$	$+0.160^*$	$+0.084^*$	$+3.273^*$

Remark 4 (Conservatism relative to leaf estimates). Since every $c_H \in [0, 1]$,

$$F_\theta(\tau) \leq \min_{H \in \mathcal{L}} c_H. \tag{29}$$

Thus, under product aggregation, one low-confidence leaf cannot be offset by high-confidence values on other leaves, and introducing additional leaves weakly decreases the estimate if existing confidences are held fixed. This is an algebraic property of the estimator; when conditional independence fails, it does not imply that the product is necessarily a lower bound on the true root probability.

A.14 Paired Bootstrap Analysis

We check whether CRG's gains in Table 2 persist under task resampling. For each benchmark–baseline–metric comparison, we create 10,000 bootstrap resamples by drawing the benchmark's original number of task IDs with replacement and retaining their available OpenHands trajectories. We compute the metric for Qwen CRG and the baseline on the same resample (paired bootstrap; Koehn, 2004). This uses saved predictions without new inference. Table 10 shows signed differences; asterisks mark nominal 95% bootstrap intervals that exclude zero.

Qwen CRG has lower adaptive ECE in all 18 comparisons, and the nominal 95% intervals exclude zero in 16: every verbalized baseline on every benchmark, plus Surrogate LNSP on EnterpriseOps-Gym. Its smaller ECE gains over Surrogate LNSP on SWE-Bench Verified and SkillsBench remain uncertain. Brier and BAS gains are supported in 14 of 18 and 17 of 18 comparisons, respectively. AUROC includes the SWE-Bench Verified losses to two Qwen verbalizers already visible in Table 2. Thus, task resampling supports the principal calibration and decision-utility gains, but does not establish that every per-benchmark difference is robust. As an exploratory summary, the mean of

the six paired differences has a nominal 95% interval above zero for ECE, Brier, and BAS on all three benchmarks, and for AUROC on EnterpriseOps-Gym and SkillsBench.

Overall, the paired bootstrap shows that **CRG's main calibration and decision-utility gains persist under test-task resampling.**

A.15 Software Versions and Randomization

SkillsBench and EnterpriseOps-Gym trajectories used OpenHands SDK v1.31.0. For SWE-Bench Verified, trajectories are source from the published runs on https://index.openhands.dev/. Therefore, the precise OpenHands SDK versions vary by model: v1.18.1 with GPT-5.5, v1.28.0 with Gemini-3.5 Flash, and v1.24.0 with MiniMax-M3. The Codex and Claude Code transfer runs used the `codex-acp` v0.11.1 and `claude-agent-acp` v0.30.0 adapters, respectively, with OpenHands SDK v1.19.1 and v1.18.0.

The 306 SWE-Bench Verified tasks were selected by difficulty without random sampling. For EnterpriseOps-Gym, we independently shuffled the tasks in each of eight domains with seed 42. Confidence-estimation runs shuffled their fixed datasets with seed 42; this affected processing order, not task membership. The paired task bootstrap used 10,000 resamples per comparison and We did not set an explicit inference-time seed for agent or estimator LLM calls.