CAN FAILED TRAJECTORIES REVEAL WHERE AGENTS GO WRONG? AGENTIC FAILURE ATTRIBUTION WITHOUT STEP-LEVEL LABELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Failure attribution for LLM agents aims to identify decisive steps responsible for task failure, which is critical for debugging and improving agentic systems. Existing methods rely on costly LLM prompting, step-level error annotations, or verified successful trajectories that lack failure signals for assistance. In this paper, we study failure attribution from *step-unlabeled failure trajectories*, where each trajectory is known to have failed but its decisive step is unknown. We propose SOAP (**S**pectral sc**O**ring with **A**ttention-guided **P**ropagation), an effective framework requiring neither step-level training labels nor a separate success-only reference set. SOAP first constructs a spectral reference space from LLM representations and assigns each step a base error score according to its projection behavior. Importantly, later steps may exhibit stronger failure signals because they inherit an earlier mistake, causing independent scorers to favor downstream consequences over the decisive error. To address this *downstream contamination*, SOAP derives soft step dependencies from attention patterns and propagates downstream error evidence toward its likely upstream sources. Extensive experiments show that SOAP consistently improves attribution performance over competitive baselines.

1 INTRODUCTION

When an LLM-based agent fails on a long-horizon task, identifying *where the failure began* is essential for debugging, auditing, and improving the agent. An agent trajectory may contain dozens or even hundreds of reasoning, communication, and tool-use steps, while only one early decision introduces the error that ultimately causes task failure. Manually tracing such trajectories is prohibitively expensive and prone to inconsistency, especially as agents are increasingly deployed for software engineering, scientific discovery, and other complex tasks (Yao et al., 2023; Wang et al., 2024). This motivates the problem of *failure attribution*: given a failed trajectory, identify the decisive step that introduced the failure (Zhang et al., 2025c).

Failure attribution is fundamentally challenging. Existing methods typically rely either on prompting powerful LLM judges, which incurs substantial inference cost and can be unreliable over long trajectories (Zhang et al., 2025c; Banerjee et al., 2025; Wang et al., 2026; Rafi et al., 2026; Yu et al., 2025), or on post-training attribution models using failure trajectories with step-level annotations (Zhang et al., 2025a;b). Such annotations are costly and often ambiguous, since identifying the decisive step may require substantial domain expertise. OAT (Yeh et al., 2026) alleviates this burden by learning the latent dynamics of successful trajectories and detecting deviations from the learned flow. However, it has no access to failure trajectories during training, and therefore *models only what successful behavior looks like*. Moreover, its success-only reference set must be verified and may become stale as agents and task distributions evolve. This motivates a complementary question:

Can failed trajectories themselves enable failure attribution without step-level annotations?

Failed trajectories provide a complementary source of supervision. In many agentic systems, whether a task has failed can be determined from execution outcomes or task evaluators, while identifying the step responsible for that failure remains costly. We therefore consider a collection of *step-unlabeled failure trajectories*, i.e., every trajectory is known to have failed, but no step is annotated as decisive or non-decisive. Each trajectory consequently contains an unlabeled mixture

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.

of ordinary steps and the decisive error. Unlike OAT (Yeh et al., 2026), such a failure-conditioned dataset directly exposes the behaviors that arise when agents fail. Yet harnessing this information is non-trivial: without step-level membership labels, the learner must extract a meaningful failure signal without knowing which step introduced the failure. The first challenge is therefore to *identify decisive-error evidence from a step-level unlabeled mixture within failed trajectories*.

A second challenge is that the step exhibiting the strongest failure signal need not be the step that introduced the failure. Agent trajectories are autoregressive because each step is generated from preceding observations and actions. Once an error occurs, later steps operate on a corrupted context and may produce failed tool calls, contradictory reasoning, or repeated recovery attempts. These downstream consequences can be more conspicuous than the original error. For example, a plausible but incorrect tool argument may trigger a sequence of visible error messages and unsuccessful retries, causing a conventional per-step scorer to select the final retry rather than the initial incorrect action. We refer to this phenomenon as *downstream contamination* (Figure 1). Recent methods reconstruct dependencies through LLM prompting or specialized training, then perform graph search or backtracking (Rafi et al., 2026; Wang et al., 2026; Zhang et al., 2025b), which introduce substantial inference or training overhead. This motivates a lightweight alternative that extracts soft dependencies from the model to separate root causes from downstream effects.

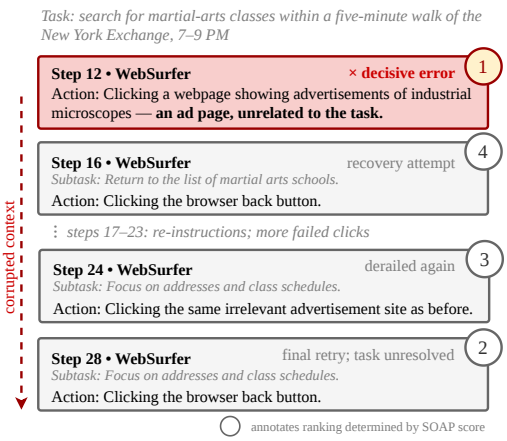

Figure 1: **Downstream contamination** in a failed WHO&WHEN trajectory (Zhang et al., 2025c). An early wrong click (step 12) lands the browser on an irrelevant page; later steps are spent on recovery attempts that derail again.

In this paper, we propose SOAP (**S**pectral sc**O**ring with **A**ttention-guided **P**ropagation), a framework for decisive-error localization from step-unlabeled failure trajectories. At a high level, SOAP consists of two components. **First**, it constructs a *spectral reference space* from step representations extracted from the failure trajectories. Each step receives a base error score according to its projection onto a selected band of singular directions. This produces a step-level signal without requiring decisive-error annotations or a separate collection of verified successful trajectories. **Second**, SOAP addresses downstream contamination through *dependency-aware rescoring*. We derive a soft dependency graph from models' attention patterns, where each edge approximates how strongly a later step draws on an earlier one. Rather than treating downstream failure signals as independent and competing evidence, SOAP propagates them backward toward the preceding steps on which they depend. An earlier step is therefore promoted when anomalous downstream behavior is systematically built upon it.

We evaluate SOAP on three failure-attribution benchmarks comprising five subsets—WHO&WHEN, CORRECT-ERROR, and TRACEELEPHANT using Qwen3.5-9B and DeepSeek-R1-Distill-Llama-8B. Across different evaluation settings, SOAP consistently improves decisive-step localization over competitive prompt-based and training-based baselines. In particular, on TRACEELEPHANT with DeepSeek-R1-Distill-Llama-8B, SOAP improves step-level Accuracy by 24.66% over the strongest baseline. Ablation studies further show that both spectral step scoring and attention-guided rescoring contribute to the improvements (Section 4.3). Our main contributions are summarized as follows:

- We formulate decisive-error localization from *step-unlabeled trajectories* within failed trajectories. This setting exploits trajectory-level failure signals while requiring neither decisive-error annotations nor carefully verified successful trajectories.
- We identify *downstream contamination* as a central obstacle to decisive-error localization, i.e., later steps may exhibit stronger failure signals because they inherit an earlier mistake, causing independent per-step scorers to systematically localize failures too late.
- We propose SOAP, an effective framework performing spectral step scoring with attention-derived dependency propagation. Extensive experiments and ablations on diverse benchmarks demonstrate when and why SOAP improves attribution.

2 PROBLEM SETUP

We formalize the task of localizing the decisive error in a failed agent trajectory.

Agent trajectory. Let x denote a task description and $\tau = (s_1, \ldots, s_T)$ the resulting trajectory of a multi-agent system. Each step s_t records the action or message produced at turn t and is associated with an agent $a_t \in \{1, \ldots, K\}$. For evaluation, each failed trajectory is paired with a gold decisive-error index $t^\star \in \{1, \ldots, T\}$, defined as the earliest step at which the error responsible for the eventual task failure is introduced (Zhang et al., 2025c). The corresponding responsible agent is $a_{t^\star}$.

Our framework extracts model-internal signals, including hidden representations, using a frozen language model $\mathcal{M}$. When the language model underlying the agents exposes its internal states, $\mathcal{M}$ may coincide with that model. Otherwise, $\mathcal{M}$ can be a separate open-weight model, enabling the analysis of trajectories generated by proprietary agents whose internal states are inaccessible. In either case, $\mathcal{M}$ is used without fine-tuning on failure-attribution data.

Failure attribution. Given a task description x and a failed trajectory τ, the goal is to estimate its decisive-error index $t^\star$. We operationalize this prediction by assigning each step a scalar attribution score $\widetilde{S}(s_t) \in \mathbb{R}$, where a larger value indicates stronger evidence that s_t introduced the failure. The predicted decisive-error step and responsible agent are $\hat{t} = \arg\max_{t \in \{1,\ldots,T\}} \widetilde{S}(s_t), \hat{a} = a_{\hat{t}}$. Sorting these scores additionally produces a ranked list of candidate steps for failure triage. We evaluate both top-1 localization and top-k performance in Section 4.2 and Appendix C.2, respectively.

Step-unlabeled failure trajectories. In addition to the test trajectory, we assume access to a disjoint reference collection of failed trajectories $\mathcal{D}_{\text{fail}} = \left\{\left(x^{(n)}, \tau^{(n)}\right)\right\}_{n=1}^{N}, \tau^{(n)} = \left(s_1^{(n)}, \ldots, s_{T_n}^{(n)}\right)$. The task-level outcome of every reference trajectory is known to be a failure, but no step-level annotation identifies where the failure was introduced or which agent was responsible. We model the marginal distribution of reference steps as an unlabeled contamination mixture (Huber, 1964):

$$P_{\text{unlabeled}} = (1 - \phi)P_{\text{ord}} + \phi P_{\text{err}}, \tag{1}$$

where P_{ord} denotes the distribution of ordinary steps, P_{err} denotes that of failure-related steps, and $\phi \in (0, 1)$ is the unknown contamination proportion. Failure-related steps include the decisive error and any downstream steps affected by it. Formally, each reference trajectory has an unobserved nonempty set $\mathcal{E}^{(n)} \subseteq \{1, \ldots, T_n\}$ of failure-related steps, whose earliest member is the decisive-error step: $t^{\star(n)} = \min \mathcal{E}^{(n)}$. At the marginal level, $s_t^{(n)} \sim P_{\text{err}}$ when $t \in \mathcal{E}^{(n)}$ and $s_t^{(n)} \sim P_{\text{ord}}$ otherwise. This setting is *practical* because failed trajectories can often be collected automatically using task-level evaluators, whereas localizing the decisive error and assigning responsibility require costly expert annotation.

Our method uses $\mathcal{D}_{\text{fail}}$ only to construct the reference representation space, without observing $\mathcal{E}^{(n)}$ or any responsible-agent annotations. Gold step-level labels are reserved for evaluation and, where applicable, hyperparameter selection on a disjoint validation split.

3 SOAP: SPECTRAL SCORING WITH ATTENTION-GUIDED PROPAGATION

We introduce SOAP, a framework for localizing decisive errors from step-unlabeled failure trajectories. At a high level, SOAP consists of two stages. *First*, we construct a spectral reference space from step representations extracted by $\mathcal{M}$ on $\mathcal{D}_{\text{fail}}$ and derive a base error score for each step (§3.1). *Second*, we estimate soft inter-step dependencies from the attention patterns of $\mathcal{M}$ and propagate error evidence from anomalous downstream steps back to their likely upstream sources (§3.2). The resulting score $\widetilde{S}(s_t)$ measures how likely step s_t is to have introduced the failure. Figure 2 summarizes the framework.

3.1 SPECTRAL STEP SCORING FROM UNLABELED FAILURE TRAJECTORIES

Although every trajectory in $\mathcal{D}_{\text{fail}}$ ends in failure, most individual steps remain locally valid, while failure-related steps constitute only a small fraction of each trajectory (Zhang et al., 2025c). Consequently, the representation space induced by $\mathcal{M}$ is dominated by recurring patterns of ordinary agent

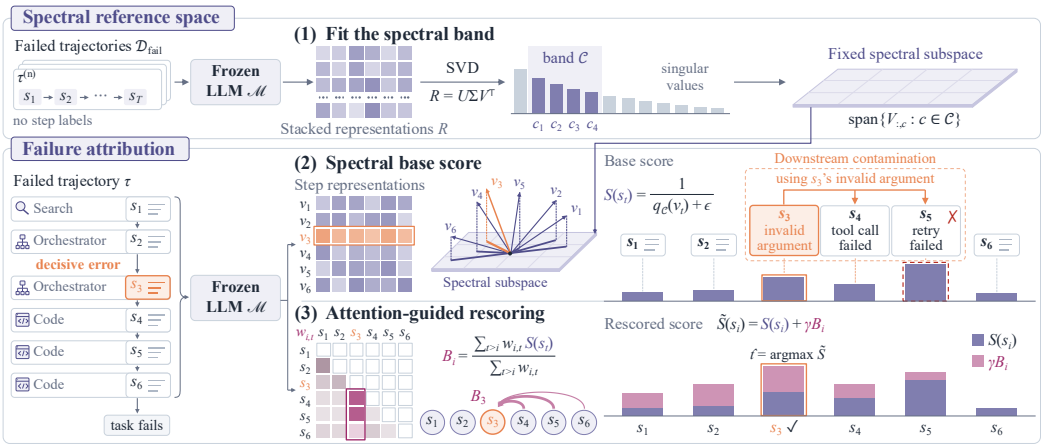

Figure 2: **Overview of SOAP for agentic failure attribution.** SOAP constructs a spectral reference space from step-unlabeled failure trajectories and assigns each step a base error score. To address downstream contamination, SOAP derives soft dependencies from attention patterns and propagates downstream error evidence toward likely upstream sources to localize the decisive error.

behavior, with failure-related steps forming a sparse, unlabeled component. We exploit this imbalance by spectrally decomposing the reference representations and measuring how strongly each step aligns with selected spectral directions.

For a trajectory $\tau = (s_1, \ldots, s_T)$, we encode each step using the task description x and selected trajectory context associated with that step. Let $v_t \in \mathbb{R}^d$ denote the resulting representation extracted by $\mathcal{M}$ for step s_t (cf. Appendix A.3).

Spectral reference space. We apply the representation extraction procedure to every step in $\mathcal{D}_{\text{fail}}$, obtaining $v_t^{(n)} \in \mathbb{R}^d$ for the t-th step of the n-th reference trajectory. Let $M_{\text{ref}} = \sum_{n=1}^{N} T_n$ denote the total number of reference steps. Stacking these representations row-wise gives $R \in \mathbb{R}^{M_{\text{ref}} \times d}$. Let $r = \min\{M_{\text{ref}}, d\}$. We compute the compact singular value decomposition

$$R = U\Sigma V^\top, \qquad V \in \mathbb{R}^{d \times r}. \tag{2}$$

The columns of V define orthogonal directions in the reference representation space, ordered by non-increasing singular value.

Because ordinary steps dominate R, the leading singular directions may primarily encode frequent shared structure, such as response style, trajectory format, and common tool-use patterns. Conversely, directions at the extreme tail may capture irrelevant variation. We therefore do not restrict scoring to either end of the spectrum. Instead, we select a contiguous spectral band

$$\mathcal{C} = \{c_{\text{begin}}, \ldots, c_{\text{end}}\}, \qquad 1 \le c_{\text{begin}} \le c_{\text{end}} \le r, \tag{3}$$

whose boundaries are selected as described in §A.3.

Spectral base score. For a step representation v_t, we compute its average squared projection onto the selected spectral band and invert this quantity:

$$q_{\mathcal{C}}(v_t) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \langle v_t, V_{:,c} \rangle^2, \quad S(s_t) = \frac{1}{q_{\mathcal{C}}(v_t) + \epsilon}, \tag{4}$$

where $\epsilon > 0$ ensures numerical stability. Because the reference spectrum is learned primarily from ordinary steps, decisive-error steps tend to exhibit lower projection energy within the selected band (Figure 3). The reciprocal therefore assigns larger base scores to steps exhibiting stronger decisive-error evidence. Constructing R and its spectral decomposition *uses no decisive-error or responsible-agent annotations.*

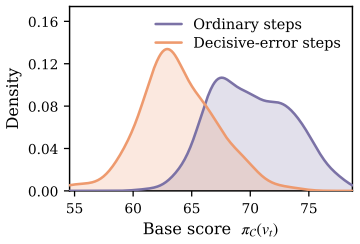

Figure 3: **Base-score distributions** on a CORRECT-ERROR test split (Yu et al., 2025). Decisive-error steps exhibit lower projection energy than ordinary steps.

3.2 DEPENDENCY-AWARE RESCORING

Attention-guided step dependencies. The base score in Eq. 4 evaluates each step independently and therefore cannot distinguish an error source from its downstream consequences. Once a decisive error occurs, subsequent steps may inherit the corrupted context and exhibit stronger failure signals than the original error. An earlier step is therefore a more plausible root cause when anomalous downstream steps depend strongly on it. We capture this structure through a soft dependency graph over the target trajectory.

For each step s_t, let $\mathcal{P}_t \subseteq \{1, \dots, t-1\}$ denote the index set of eligible predecessor steps retained by our context construction. For each $i \in \mathcal{P}_t$, we aggregate the post-softmax attention routed from tokens in s_t to tokens in s_i, yielding a non-negative attention mass $m_{i,t}$. We normalize these masses over the retained predecessors:

$$w_{i,t} = \begin{cases} \dfrac{m_{i,t}}{\sum_{j \in \mathcal{P}_t} m_{j,t}}, & i \in \mathcal{P}_t, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Thus, $\sum_{i \in \mathcal{P}_t} w_{i,t} = 1$ whenever $\mathcal{P}_t$ is nonempty, and $w_{i,t}$ represents the relative dependency of the later step s_t on the earlier step s_i. The construction of $\mathcal{P}_t$ and the attention aggregation procedure are detailed in Appendix A.3.

These weights provide model-internal evidence about which earlier steps contribute to the representation of a later step. Because they are obtained directly from the frozen LLM model, their construction requires neither repeated generative prompting, which introduces substantial inference cost (Zhang et al., 2025c; Yeh et al., 2026), nor explicit dependency reconstruction and traversal through LLM-based graph reasoning (Rafi et al., 2026; Wang et al., 2026; Bakish et al., 2026), nor task-specific post-training on attributed failure trajectories (Zhang et al., 2025a;b).

Rescoring. We treat the error signal of a downstream step as additional evidence for the earlier steps on which it depends. For each step s_i, we compute the dependency-weighted average of its downstream base scores:

$$B_i = \frac{\sum_{t>i} w_{i,t}\, S(s_t)}{\sum_{t>i} w_{i,t}}, \tag{6}$$

In particular, $B_T = 0$ because the final step has no downstream dependents. We combine the local error signal with the downstream evidence attributed to the step:

$$\widetilde{S}(s_i) = S(s_i) + \gamma B_i, \qquad \gamma \in [0, 1]. \tag{7}$$

Here, γ controls the strength of dependency-aware rescoring, with $\gamma = 0$ recovering the spectral base score. The normalization in Eq. 6 prevents an early step from receiving a large correction merely because it has many downstream dependents. Instead, downstream steps with larger dependency weights exert greater influence on its correction. Thus, an earlier step is promoted when the downstream steps that depend strongly on it exhibit large base error scores.

Finally, we predict the decisive-error step and corresponding responsible agent as

$$\hat{t} = \arg\max_{i \in \{1, \dots, T\}} \widetilde{S}(s_i), \qquad \hat{a} = a_{\hat{t}}. \tag{8}$$

4 EXPERIMENTS

4.1 SETUP

Datasets. We evaluate our method SOAP on 3 failure-attribution benchmarks comprising 5 subsets: WHO&WHEN (Zhang et al., 2025c), with algorithm-generated (WW-AG, 126 trajectories) and hand-crafted (WW-HC, 58 trajectories) subsets; CORRECT-ERROR (Yu et al., 2025) (CE, 2,226 trajectories); and TRACEELEPHANT (Chen et al., 2026), with Captain-Agent (TE-CAP, 85 trajectories) and Magentic-One (TE-MAG, 91 trajectories) subsets. They cover trajectories of varying lengths and diverse QA, coding, and tool-use tasks. Each trajectory is annotated with the decisive-error step and responsible agent. For each subset, we randomly split trajectories into unlabeled-reference, validation, and test sets using a $30/20/50\%$ ratio. The reference annotations are hidden

Table 1: **Main comparison in the *without-ground-truth* setting.** Results are step-level attribution accuracy (%), averaged over three seeds (std in Appendix C.1). No method receives the gold task answer during inference. **Supervised** marks methods that consume step-level error annotations. Best per column within each block in bold, second best underlined.

Method	Supervised	WW-AG	WW-HC	CE	TE-CAP	TE-MAG
GPT-4o						
Prompt-based methods						
All-at-Once (Zhang et al., 2025c)	✗	14.81	6.90	28.39	16.28	5.80
Step-by-Step (Zhang et al., 2025c)	✗	20.63	13.79	44.47	13.95	13.04
Binary Search (Zhang et al., 2025c)	✗	22.22	4.60	9.70	9.30	6.52
CORRECT (Yu et al., 2025)	✓	15.34	4.60	35.12	10.85	13.77
CHIEF (Wang et al., 2026)	✗	25.93	14.94	38.56	13.95	8.70
ErrorProbe (Li et al., 2026)	✗	**38.62**	**21.84**	**59.52**	**27.13**	**28.26**
Qwen3.5-9B						
Prompt-based methods						
All-at-Once (Zhang et al., 2025c)	✗	17.46	11.49	33.47	4.65	1.45
Step-by-Step (Zhang et al., 2025c)	✗	10.58	16.09	31.15	14.73	10.87
Binary Search (Zhang et al., 2025c)	✗	2.65	13.79	59.92	0.78	5.07
CORRECT (Yu et al., 2025)	✓	29.10	3.45	36.35	7.75	2.90
CHIEF (Wang et al., 2026)	✗	28.57	3.45	29.39	3.88	2.17
ErrorProbe (Li et al., 2026)	✗	39.68	21.84	55.03	23.26	20.29
Training-based methods						
StepFinder (Zhu et al., 2026b)	✓	15.87	13.33	30.03	18.29	6.67
OAT (Yeh et al., 2026)	✗	16.72	10.11	56.50	15.35	18.84
SOAP (ours)	✗	**47.62**	**34.48**	**61.78**	**35.66**	**23.19**
DeepSeek-R1-Distill-Llama-8B						
Prompt-based methods						
All-at-Once (Zhang et al., 2025c)	✗	7.41	5.75	21.95	4.65	2.90
Step-by-Step (Zhang et al., 2025c)	✗	16.93	9.20	9.04	11.63	4.35
Binary Search (Zhang et al., 2025c)	✗	7.94	12.64	34.15	10.08	13.77
CORRECT (Yu et al., 2025)	✓	20.11	11.49	32.82	3.88	0.72
CHIEF (Wang et al., 2026)	✗	17.46	2.30	8.40	7.75	9.42
ErrorProbe (Li et al., 2026)	✗	24.34	10.34	35.97	17.05	8.70
Training-based methods						
StepFinder (Zhu et al., 2026b)	✓	18.94	10.80	36.91	14.73	4.20
OAT (Yeh et al., 2026)	✗	12.70	15.86	52.50	17.98	13.04
SOAP (ours)	✗	**45.50**	**28.74**	**64.68**	**42.64**	**30.43**

and used only to construct the spectral matrix R; validation labels are used for hyperparameter selection, and test labels only for final evaluation. Splitting is performed at the trajectory level. Full statistics and preprocessing details are provided in Appendix A.1.

Baselines. We compare SOAP against two families of baselines. First, *prompt-based* methods include All-at-Once, Step-by-Step, and Binary Search (Zhang et al., 2025c), the schema-retrieval method CORRECT (Yu et al., 2025), the causal-graph method CHIEF (Wang et al., 2026), and the analyzer–verifier diagnoser ErrorProbe (Li et al., 2026). These methods ask a judge LLM to name the decisive step. Second, *training-based* methods include OAT (Yeh et al., 2026), which learns successful-trajectory dynamics through one-class training, and StepFinder (Zhu et al., 2026b), which trains a step classifier on synthetically corrupted trajectories with step-level error labels.

Implementation details. We use Qwen3.5-9B (Qwen Team, 2025) and DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025) (DeepSeek-8B) as backbones $\mathcal{M}$. We obtain step representations by mean-pooling token hidden states. We select the representation layer, the spectral band $\mathcal{C}$, the attention layer band, and the propagation strength γ on the validation set. All training-based baselines use the same two backbones and test data as SOAP, with the hyperparameters recommended in their papers. Following Yeh et al. (2026), we additionally compare with prompt-based baselines with GPT-4o as the judge (results with GPT-5 are in Appendix C.3). Following Zhang et al. (2025c), we report *Step-Level Accuracy* (additional metrics, including *Agent-level Accuracy* and *top-k Accuracy* are in Appendix C.1 and C.2). Further implementation details are provided in Appendices A.3 and A.4.

4.2 MAIN RESULTS

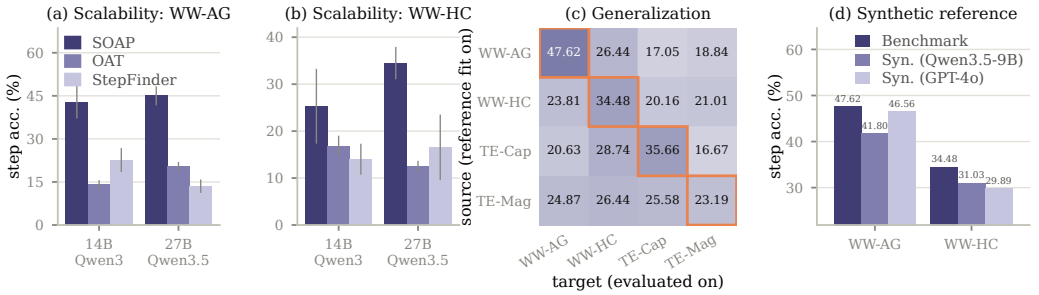

Figure 4: (a–b) **Scalability to larger backbones**, which show the step-level accuracy (%) on WW-AG and WW-HC using Qwen3-14B and Qwen3.5-27B; error bars denote standard deviations. (c) **Cross-distribution generalization**, with source datasets as rows and target datasets as columns; outlined diagonal entries correspond to the in-distribution results in Table 1. (d) **Results with synthetic trajectories** generated by Qwen3.5-9B and GPT-4o. Panels (c) and (d) use Qwen3.5-9B as LLM $\mathcal{M}$ for embedding extraction and rescoring.

Comparison with competitive baselines. Table 1 compares SOAP with prompt-based and training-based baselines across all five subsets, without access to the gold task answer at inference. With both open-weight backbones, SOAP achieves the highest step-level accuracy on every subset. For example, on WW-AG with Qwen3.5-9B, SOAP attains 47.62%, compared with 39.68% for ErrorProbe (Li et al., 2026), the strongest prompt-based baseline, and 16.72% for OAT (Yeh et al., 2026), the strongest training-based baseline. Its advantage over prompt-based methods using closed-source judges also persists with both GPT-4o and GPT-5; the GPT-5 results are in Appendix C.3. These results demonstrate that step-unlabeled failure trajectories provide an effective source of localization signal, outperforming the success-only trajectories used by OAT and the step-level supervision used by StepFinder. Agent-level results are provided in Appendix C.1.

Table 2: **Comparison in the *with-ground-truth* setting.** Results are step-level accuracy (%) on WHO&WHEN, with the gold task answer provided. All methods use Qwen3.5-9B as the backbone. Marks follow Table 1.

Method	Sup.	WW-AG	WW-HC
Prompt-based methods			
All-at-Once	✗	21.16	11.49
Step-by-Step	✗	17.46	18.39
Binary Search	✗	2.12	13.79
CORRECT	✓	32.28	4.60
CHIEF	✗	30.69	4.60
ErrorProbe	✗	<u>40.74</u>	<u>19.54</u>
Training-based methods			
StepFinder	✓	18.52	12.87
OAT	✗	22.12	8.05
SOAP (ours)	✗	**43.39**	**34.48**

Results for the with-GT setting. To assess the effectiveness of SOAP under the with-GT setting, we provide the gold task answer to all methods and compare them in Table 2. When provided with the ground-truth answer, SOAP remains the best method on both subsets, ahead of the strongest prompt-based baseline ErrorProbe (Li et al., 2026) (43.39 vs. 40.74 on WW-AG and 34.48 vs. 19.54 on WW-HC), while every training-based baseline trails by more than 20 points (e.g., 22.12 for OAT (Yeh et al., 2026) on WW-AG). Results on DeepSeek-8B appear in Appendix A.5.

Scalability to larger backbones. To assess scalability, we evaluate SOAP on two larger backbones, Qwen3-14B and Qwen3.5-27B. As shown in Figure 4(a-b), SOAP achieves stable accuracy on both backbones and outperforms the SOTA baselines (e.g., 44.97 vs. 22.65 for StepFinder (Zhu et al., 2026b) on WW-AG). These results confirm that SOAP remains effective on larger backbones.

Generalization across data distributions. We construct the reference on a source subset and directly evaluate SOAP on a different target subset, for every pair among {WW-AG, WW-HC, TE-CAP, TE-MAG}. As shown in Figure 4(c), SOAP transfers across subsets: for example, constructing on WW-HC and testing on TE-MAG achieves 21.01, compared with the in-domain result of 23.19. These results indicate that the spectral reference captures failure structure that can be generalizable in practice. Results on DeepSeek-8B appear in Appendix C.5.

Results with synthetic trajectories. Our main results construct R from existing benchmark trajectories. We further examine whether alternative synthetic trajectories can serve as the reference while retaining Qwen3.5-9B as the attribution backbone. We generate the same number of trajectories using either Qwen3.5-9B itself or GPT-4o as an external generator, while keeping the original validation and test splits unchanged. As shown in Figure 4(d), both synthetic references remain effective. For example, the GPT-4o-generated reference achieves 46.56 on WW-AG, close to 47.62

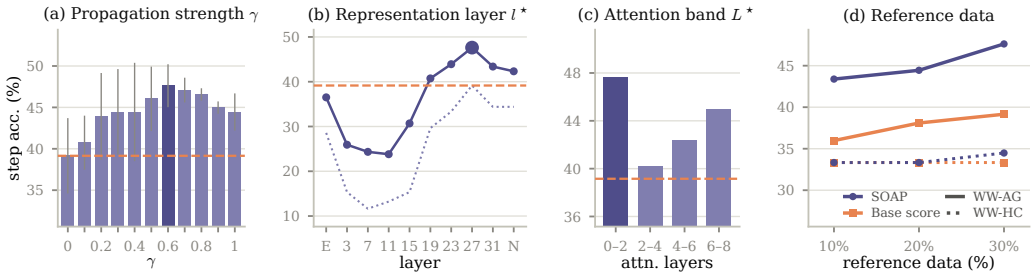

Figure 5: **Ablation studies.** Sensitivity to (a) propagation strength γ, (b) representation layer $l^\star$, (c) attention layer band $L^\star$, and (d) the amount of unlabeled reference data. All experiments use Qwen3.5-9B. Panels (a–c) vary one hyperparameter at a time from the selected WW-AG configuration used in Table 1. Dark bars and enlarged markers denote the selected values; the orange dashed line denotes the base score without rescoring ($\gamma = 0$), and the dotted curve in (b) reports the base score at each layer. Panel (d) constructs R using 10%, 20%, or 30% of the reference corpus and reports results on WW-AG and WW-HC. Best viewed in color.

using the original training trajectories. Thus, SOAP can construct its reference space from synthetic trajectories, either when generated by the same or a different model. Generation details are provided in Appendix A.7.

4.3 ANALYSIS

Comparison with alternative scoring functions. We compare the spectral base score (Eq. 4) with seven alternatives in Table 3: step perplexity, projection to random subspace, top subspace, tail subspace, full spectrum, and the ℓ_1 and ℓ_2 norms of LLM step embeddings (details in Appendix A.3). Our spectral band (Eq. 3) remains the best on both subsets, and the best alternative reaches only 18.52 on WW-AG, compared with 39.15 for ours. These results indicate that the decisive-error signal lies more likely in a selected spectral band than in LLM predictive likelihood or embedding magnitude.

Table 3: **Comparison with alternative scoring functions.** Results are step-level accuracy (%) on backbone Qwen3.5-9B, with no rescoring.

Scoring function	WW-AG	WW-HC
Perplexity	3.70	8.05
Random subspace	13.23	9.20
Top subspace	12.70	**33.33**
Tail subspace	6.35	6.90
Full spectrum	12.17	31.03
Norm ℓ_1	18.52	18.39
Norm ℓ_2	14.29	26.44
Spectral band (ours)	**39.15**	**33.33**

Sensitivity to the hyperparameters. We vary the propagation strength γ (Eq. 7), representation layer $l^\star$ for base score calculation (Eq. 4), and layer band $L^\star$ for attention mass calculation (Eq. 5) one at a time. Here, $L^\star$ specifies the group of attention blocks averaged to estimate step dependencies (cf. Appendix A.3 for details). As shown in Figure 5, propagation consistently improves over the base score and peaks at $\gamma = 0.6$, while later representation layers perform best. The earliest attention band performs best, suggesting that lower attention blocks provide clearer dependency signals, although all bands improve over the base score. Extended ablations on subsets WW-HC, TE-CAP, TE-MAG, backbone DeepSeek-8B are in Appendix B.

Importance of attention-guided rescoring. Table 4 keeps the spectral base score fixed and replaces attention-guided rescoring with simpler alternatives (details in Appendix A.3). *Uniform, normalized* averages all successor scores equally, whereas *uniform, unnormalized* sums them, favoring early steps with more successors. *Temporal bias* subtracts a penalty $\lambda t/T$ that grows with the step position t, either from the base score $S(s_t)$ directly (*base*) or after standardizing the base scores within each trajectory (*standardized*). *Earliest of top-5* selects the earliest among the five highest-scoring steps. Attention-guided rescoring performs best on both subsets, reaching 47.62 on WW-AG compared with 40.74 for the strongest alternative. This shows that the gain arises from selectively weighting downstream evidence through attention, rather than uniform aggregation or positional bias.

Table 4: **Alternatives to attention-guided rescoring.** Step-level accuracy using Qwen3.5-9B; all methods share the same base score.

Rescoring design	WW-AG	CE
Base score (no rescoring)	39.15	61.38
Content-agnostic		
Uniform, unnormalized	16.93	57.85
Uniform, normalized	40.21	60.78
Position heuristics		
Temporal bias, std.	40.74	61.62
Temporal bias, base	22.22	41.65
Earliest of top-5	21.16	4.51
Attention-guided (ours)	**47.62**	**61.78**

Quantity of unlabeled reference data. We construct R on random subsamples covering $\{10, 20, 30\}\%$ of the corpus, with the validation and test splits fixed. As shown in Figure 5(d), SOAP needs little reference data: with a third of our 30% train split (twelve trajectories on WW-AG and six on WW-HC), accuracy is within a few points of the full-data result (43.39 vs. 47.62 on WW-AG). These results indicate that a handful of unlabeled trajectories suffices to construct the reference, showcasing the practicality of our approach.

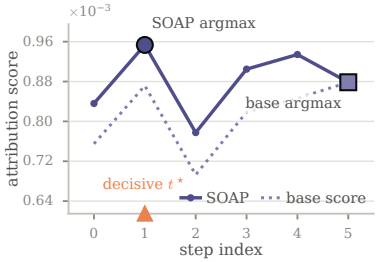

Figure 6: **Qualitative example.** Per-step scores on a TE-CAP trajectory with backbone Qwen3.5-9B. The base score (dotted) peaks late; SOAP (solid) lifts the decisive error $t^\star$ (orange marker) to the top. The large square and circle mark each score's argmax.

Qualitative example. In the TE-CAP trajectory of Figure 6, the base score peaks on the final Orchestrator turn, which only restates the wrong answer. Rescoring routes that late signal back to the decisive error four steps earlier, where the extraction expert misreads the program's structure and proposes the wrong fix, and corrects the prediction. More textual examples are in Appendix C.7.

Additional analysis. Due to space limitations, we defer additional experiments to the Appendix: cost analysis (A.6), context window w in constructing $\mathcal{P}_t$ (B.1), representation pooling (B.3), hyperparameter sensitivity (B.2), trivial baselines (B.5), effect of the reference corpus (B.6), a supervised probe comparison (B.7), and extended results on larger backbones (C.4), synthetic references (C.6), and reference data quantity (B.4).

5 RELATED WORK

Agentic failure attribution has gained interest recently for ensuring LLM-based agents' reliability (Qi et al., 2026; Barke et al., 2026). Zhang et al. (2025c) formulate automated failure attribution, introduce the WHO&WHEN benchmark, and evaluate All-at-Once, Step-by-Step, and Binary Search LLM judges; Who&When Pro scales the benchmark to 12K failed trajectories across modalities (Liu et al., 2026a), and complementary work categorizes common multi-agent failure modes (Cemri et al., 2026; Deshpande et al., 2025; Raj et al., 2026; Zhu et al., 2025). Subsequent methods improve inference-time prompting: RAFFLES iterates a judge–evaluator loop (Zhu et al., 2026a), ErrorProbe pairs an analyzer with a self-improving verifier (Li et al., 2026), CHIEF constructs causal dependency graphs through multi-stage LLM reasoning (Wang et al., 2026), and others restructure the context, retrieve error schemata, or reconstruct step dependencies (Banerjee et al., 2025; Yu et al., 2025; Bakish et al., 2026; Rafi et al., 2026). Another line trains dedicated attributors: AgenTracer and StepFinder learn from labeled or synthetically corrupted trajectories by reinforcement learning or temporal-semantic modeling (Zhang et al., 2025a; Zhu et al., 2026b), and GraphTracer learns to trace root causes over information-flow graphs (Zhang et al., 2025b). These approaches, however, rely on *computationally intensive inference or step-level error annotations*, limiting their practicality; we compare with representatives of both lines in Table 1. OAT avoids both by one-class learning over clean successful trajectories (Yeh et al., 2026), but its success-only data *carry no decisive-error signal*, unlike our step-unlabeled failure trajectories, and it underperforms SOAP (Table 1). Du et al. (2024) detects hallucinations using unlabeled LLM generations but trains a classifier on top of a different projection score without the dependency propagation compared to our approach. Liu et al. (2026b) also use attention signals, but score them on steps with high negative log-likelihood for two-stage localization; MASC targets online self-correction via prototype-guided next-execution reconstruction rather than post-hoc error attribution (Shen et al., 2025).

6 CONCLUSION

We propose SOAP, an effective framework for failure attribution in LLM-based multi-agent systems. Our framework scores each step against a spectral reference constructed on unlabeled failure trajectories, and routes late contamination back to the decisive error with attention-guided propagation. SOAP is lightweight, requires no step-level annotation, and can be readily applied to off-the-shelf LLMs. Extensive experiments show that SOAP consistently outperforms prompt-based and training-based baselines across benchmarks, backbones, and settings. We hope our work can inspire future research on failure attribution and more broadly on building reliable multi-agent systems.

AI USE STATEMENT

We used Qwen3.5-9B and GPT-4o to generate synthetic agentic trajectories for the experiments described in Appendix A.7. Generative AI tools also assisted with result interpretation, and manuscript polishing. The use of language models for representation extraction and prompt-based attribution is described in Section 4.1 and Appendices A.3–A.4. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with generative AI assistance.

ETHICS STATEMENT

This work aims to improve reliability of LLM agents by identifying steps associated with task failure. Our experiments use existing agent-trajectory benchmarks and synthetic reference trajectories. Attribution errors may arise from ambiguity in benchmark annotations. Applications to private agent logs should protect sensitive information and respect the access restrictions of the underlying data.

REPRODUCIBILITY STATEMENT

Section 3 specifies the spectral scoring and attention-guided rescoring procedures. Section 4.1 and Appendices A.1–A.5 describe the benchmarks, data splits, input construction, context truncation, hyperparameter search spaces, baseline implementations, and evaluation settings. Appendix A.6 reports the software and hardware environment and timing protocols. Appendix A.7 describes synthetic reference generation, with corresponding results in Appendix C.6. Additional ablations, sensitivity analyses, and variability across random seeds are reported in Appendices B and C.

REFERENCES

Yarden Bakish, Amir Dudai, Roy Ganz, Oren Nuriel, Elad Ben Avraham, Mor Shpigel Nacson, and Ron Litman. Adaptive influence graphs for failure attribution in multi-agent systems. *arXiv preprint arXiv:2608.24361*, 2026.

Chaithanya Bandi, Razvan-Gabriel Dumitru, Ben Hertzberg, Divyansh Agarwal, Geobio Boo, Tejas Polakam, Sami Hassaan, Jeff Da, HiJae Kim, Vipul Gupta, Manasi Sharma, Andrew Park, Martin Dimakis, Ernesto Gabriel Hernandez Montoya, Dan Rambado, Ivan Salazar, Rafael Cruz, MohammadHossein Rezaei, Chetan Rane, Ben Levin, Daniel Yue Zhang, Brad Kenstler, and Bing Liu. MCP-Atlas: A large-scale benchmark for tool-use competency with real MCP servers. *arXiv preprint arXiv:2602.00933*, 2026.

Adi Banerjee et al. Where did it all go wrong? a hierarchical look into multi-agent error attribution. *arXiv preprint arXiv:2510.04886*, 2025.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=fAjbYBmonr.

Mengzhuo Chen, Junjie Wang, Fangwen Mu, Yawen Wang, Zhe Liu, Huanxiang Feng, and Qing Wang. Seeing the whole elephant: A benchmark for failure attribution in LLM-based multi-agent systems. 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. 19888–19905, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.912. URL https://aclanthology.org/2026.acl-long.912/.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. Trail: Trace reasoning and agentic issue localization. *arXiv preprint arXiv:2505.08638*, 2025.

Xuefeng Du, Chaowei Xiao, and Yixuan Li. Haloscope: Harnessing unlabeled llm generations for hallucination detection. *Advances in Neural Information Processing Systems*, 37:102948–102972, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. doi: 10.1214/aoms/1177703732.

Jiazheng Li, Emine Yilmaz, Bei Chen, and Thu Le. Towards self-improving error diagnosis in multi-agent systems. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 2063–2077, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.98. URL https://aclanthology.org/2026.findings-acl.98/.

Jiale Liu, Huajun Xi, Shaokun Zhang, Yifan Zeng, Tianwei Yue, Chi Wang, Jian Kang, Qingyun Wu, and Huazheng Wang. Who&when pro: Can LLMs really attribute failures in AI agents? *arXiv preprint arXiv:2607.09996*, 2026a.

Yang Liu, Hongjiang Feng, Junsong Pu, and Zhuangbin Chen. MASPrism: Lightweight failure attribution for multi-agent systems using prefill-stage signals. *arXiv preprint arXiv:2605.07509*, 2026b.

Yunjia Qi, Zehua Yin, Xintong Shi, Hao Peng, Songyuanyi Lu, Yixian Liu, Richeng Xuan, Yuhong Liu, Zhichao Hu, Xiaozhi Wang, Lei Hou, Bin Xu, and Juanzi Li. TRAJDEBUG: Tracing error lifecycle to identify critical failures in long-horizon agent trajectories. *arXiv preprint arXiv:2608.06346*, 2026.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Md Nakhla Rafi, Md Ahasanuzzaman, Dong Jae Kim, Zhijie Wang, and Tse-Hsun Chen. Falat: Tracing failures in llm agent trajectories via dependency-guided search, 2026. URL https://arxiv.org/abs/2606.00765.

Harsh Raj, Vipul Gupta, Anas Mahmoud, Razvan-Gabriel Dumitru, Darvin Yi, Aakash Sabharwal, and Yunzhong He. Model or harness? an interaction-centric taxonomy for localizing agent failures. *arXiv preprint arXiv:2607.28802*, 2026.

Xu Shen, Qi Zhang, Song Wang, Zhen Tan, Xinyu Zhao, Laura Yao, Vaishnav Tadiparthi, Hossein Nourkhiz Mahjoub, Ehsan Moradi Pari, Kwonjoon Lee, and Tianlong Chen. Metacognitive self-correction for multi-agent system via prototype-guided next-execution reconstruction. *arXiv preprint arXiv:2510.14319*, 2025.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024. doi: 10.1007/s11704-024-40231-1.

Yawen Wang, Wenjie Wu, Junjie Wang, and Qing Wang. From flat logs to causal graphs: Hierarchical failure attribution for llm-based multi-agent systems, 2026. URL https://arxiv.org/abs/2602.23701.

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

Samuel Yeh, Yiwen Zhu, Shaleen Deep, and Sharon Li. Tracing agentic failure from the flow of success. *arXiv preprint arXiv:2607.12747*, 2026.

Yifan Yu, Moyan Li, Shaoyuan Xu, Jinmiao Fu, Xinhai Hou, Fan Lai, and Bryan Wang. CORRECT: COndensed eRror RECognition via knowledge transfer in multi-agent systems. *arXiv preprint arXiv:2509.24088*, 2025.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the llm agentic systems? *arXiv preprint arXiv:2509.03312*, 2025a.

Heng Zhang, Yuling Shi, Xiaodong Gu, Haochen You, Zijian Zhang, Lubin Gan, Yilei Yuan, and Jin Huang. Graphtracer: Graph-guided failure tracing in llm agents for robust multi-turn deep search, 2025b. URL https://arxiv.org/abs/2510.10581.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. In *International Conference on Machine Learning (ICML)*, 2025c.

Chenyang Zhu, Spencer Hong, Jingyu Wu, Kushal Chawla, Yuhui Tang, Youbing Yin, Nathan Wolfe, Erin Babinsky, and Daben Liu. RAFFLES: Reasoning-based attribution of faults for LLM systems. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7659–7688, Rabat, Morocco, March 2026a. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.359. URL https://aclanthology.org/2026.eacl-long.359/.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. Where LLM agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

Taiyu Zhu, Yifan Wu, Weilin Jin, Ying Li, and Gang Huang. StepFinder: A temporal semantic framework for failure attribution in multi-agent systems. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2026b.

A ADDITIONAL EXPERIMENTAL DETAILS

A.1 DATASETS AND STATISTICS

We evaluate on three failure-attribution benchmarks that together provide five reported subsets. Every trajectory is a failed run annotated with one decisive-error step and its responsible agent.

WHO&WHEN. We use the two subsets of Zhang et al. (2025c): WW-AG holds 126 trajectories generated by CaptainAgent, and WW-HC holds 58 trajectories of Magentic-One.

CORRECT-ERROR. The corpus of Yu et al. (2025) spans seven question pools (ARC, GAIA, HotpotQA, MATH-500, MMLU-Pro, MuSiQue, and 2WikiMultihopQA), 2,226 trajectories in total. We report the column CE as the unweighted mean over the seven per-subset accuracies.

TRACEELEPHANT. From Chen et al. (2026) we use the CaptainAgent subset TE-CAP (85 trajectories) and the Magentic-One subset TE-MAG (91 trajectories). TRACEELEPHANT also releases, for every step, the context the agent saw when it produced that step. We discard this context and keep only the step outputs, so its trajectories take the same form as those of WHO&WHEN.

Trajectory statistics. Table 5 reports the size and shape of every subset. Trajectory length differs by an order of magnitude: a WW-AG trajectory has at most ten turns, while a WW-HC trajectory averages 52 turns and reaches 130, which makes exact localization hardest there.

Table 5: Corpus statistics per subset: number of trajectories, turns per trajectory (mean, median, maximum), distinct agents per trajectory, and the mean position of the decisive error (0-indexed). The CE row is the unweighted mean over its seven subsets.

Subset	Agent system	Traj.	Turns			Agents	Decisive step
			mean	med.	max		mean
WW-AG	CaptainAgent	126	8.7	10	10	3.6	3.0
WW-HC	Magentic-One	58	51.6	32	130	5.7	15.2
TE-CAP	CaptainAgent	85	20.5	13	59	2.7	6.0
TE-MAG	Magentic-One	91	29.3	28	50	2.1	11.4
CE-ARC	CORRECT	304	5.3	2	22	2.3	2.0
CE-GAIA	CORRECT	50	10.9	10	22	2.9	7.7
CE-HOTPOTQA	CORRECT	578	12.8	10	22	3.0	5.2
CE-MATH-500	CORRECT	157	6.3	5	21	3.1	3.3
CE-MMLU-PRO	CORRECT	92	6.1	2	22	2.6	2.0
CE-MUSIQUE	CORRECT	312	14.7	21	22	3.0	6.1
CE-2WIKIMQA	CORRECT	733	14.5	14	22	3.0	6.7
CE (mean)	CORRECT	2,226	10.1	9.1	22	2.8	4.7

A.2 INPUT FORMAT AND CONTEXT TRUNCATION

Input prompts. To score step s_t, the LLM $\mathcal{M}$ reads a single user turn of its chat template. The turn holds the task description followed by the steps $s_1, \ldots, s_t$. Each step is written as `[{agent}] - Step {i}: {content}`, and consecutive steps are separated by a blank line. No system prompt is used. The tokens of s_t are used in both extracting its representation v_t and the attention mass $m_{i,t}$ (Appendix A.3). In simplified form, the input to score s_t reads:

```
[{agent_1}] - Step 1:  {task description and first action}
[{agent_2}] - Step 2:  {content}
...
[{agent_t}] - Step t:  {content}                          ← scored step
```

Context truncation. The context budget is 8,192 tokens for every subset and backbone. The task description always opens the context. Without the ground truth it is the question alone; with the ground truth it is the question followed by the gold answer (Appendix A.5). When the task descrip-

tion, the preceding steps, and the scored step together exceed the budget, we drop the oldest steps one at a time until the input fits. The task description and the scored step are never dropped.

A.3 Implementation details of SOAP

Step representations. v_t is the mean of the hidden states over the tokens of the scored step at one layer. We store the output of every full-attention block, the embedding layer, and the last block after the final norm; for architectures that interleave linear-attention and full-attention blocks (e.g., Qwen3.5), only the full-attention blocks are kept. The representation layer $l^\star$ is selected over these positions. Ablations on the pooling design are in Appendix B.3.

Attention mass. Consider the input for step s_t as the token sequence of $[s_1, \ldots, s_t]$, and let $\mathcal{T}_i$ denote the token positions of step s_i. Let $A^{(l,h)}$ be the post-softmax attention matrix of layer l and head h: row p holds the distribution of query token p over the key tokens it may attend to. The mass that step s_t routes to an earlier step s_i is the probability that a typical query token of s_t places on the tokens of s_i, averaged over the heads and over the layers of the selected band $L^\star$:

$$m_{i,t}^{(l,h)} = \frac{1}{|\mathcal{T}_t|} \sum_{p \in \mathcal{T}_t} \sum_{q \in \mathcal{T}_i} A_{p,q}^{(l,h)}, \qquad m_{i,t} = \frac{1}{|L^\star|\, H} \sum_{l \in L^\star} \sum_{h=1}^{H} m_{i,t}^{(l,h)}, \tag{9}$$

where H is the number of heads and $L^\star$ is one of four equal bands of the full-attention blocks. Averaging over the query tokens treats the step as a whole; summing over the key tokens keeps the total probability that lands in the predecessor. Every $m_{i,t}$ for step s_t comes from the forward pass that also produces v_t:

Input: steps $s_1, \ldots, s_t$; layer band $L^\star$. **Output:** $m_{i,t}$ for every $i < t$.
1. Run one forward pass of $\mathcal{M}$ on $[s_1, \ldots, s_t]$ and record the token span $\mathcal{T}_i$ of every step.
2. **For** each layer $l \in L^\star$ and head h: take the attention rows of the query tokens in $\mathcal{T}_t$.
   For each predecessor $i < t$: sum the probabilities over the key tokens in $\mathcal{T}_i$, then average over the query tokens; this is $m_{i,t}^{(l,h)}$.
3. Average $m_{i,t}^{(l,h)}$ over the heads and the layers in $L^\star$ to obtain $m_{i,t}$.
4. Normalize $m_{i,t}$ over the predecessor set $\mathcal{P}_t$ by Eq. 5.

Spectral base score. The reference matrix R stacks the representations of every step of the reference split, uncentered. We keep the 20 leading right-singular vectors and consider every contiguous band $[c_{\text{begin}}, c_{\text{end}})$ within this range. The score is $S(s_t) = 1/(q_{\mathcal{C}}(v_t) + \epsilon)$ with $\epsilon = 10^{-12}$. Ranking is within a trajectory, and ties resolve to the earliest step.

Construction of the predecessor set during rescoring. Section 3.2 normalizes the attention mass over an index set $\mathcal{P}_t$ of retained predecessors. We build $\mathcal{P}_t$ in three steps. First, context truncation (Appendix A.2) fixes which earlier steps the LLM reads when it encodes s_t; only those can receive attention from s_t. Second, among them we keep the w predecessors with the largest mass $m_{i,t}$. The window $w \in \{1, 2, 3, 4, 5, \text{all}\}$ is selected per cell, and $w =$ all keeps every retained predecessor. Third, we remove any turn that is never a candidate step: the user's question turn of WW-HC and, with the ground truth, the pinned answer block. The weights $w_{i,t}$ of Eq. 5 are then normalized over the surviving set. Because $w_{i,t} = 0$ outside $\mathcal{P}_t$, the sums in Eq. 6 run over the successors t whose $\mathcal{P}_t$ contains i. A step that no successor retains keeps its base score, as does the last step. The propagation strength is selected from $\gamma \in \{0, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$, and $\gamma = 0$ recovers the base score. Appendix B.1 varies w.

Hyperparameter selection. To keep the search space tractable, SOAP's hyperparameters are selected in two stages on the validation split. Stage 1 selects the representation layer $l^\star$ and the spectral band $\mathcal{C}$ by the validation step-level accuracy of the base score alone. Stage 2 freezes $(l^\star, \mathcal{C})$ and selects the attention band $L^\star$, propagation strength γ, and window w by the validation step-level accuracy of the rescored prediction. Ties in both stages are broken by agent-level accuracy. Test labels are never consulted during selection. Baselines use the hyperparameters recommended in their respective papers.

Alternative scoring functions. The seven alternatives of Table 3 share the selected layer and band width $|\mathcal{C}|$ of the spectral score; only the score itself changes. *Perplexity* is the mean negative log-likelihood of the step's tokens under the LLM in context, read as higher = error. Four projection

variants replace the spectral band with a different subspace of the same dimension: *random subspace* uses a random orthonormal basis of dimension $|\mathcal{C}|$, *top subspace* the leading band $[0, |\mathcal{C}|)$, *tail subspace* the last $|\mathcal{C}|$ of the 20 computed components, and *full spectrum* all 20. The ℓ_1 and ℓ_2 norms score $\|v_t\|_1$ and $\|v_t\|_2$. Every projection and norm score is read as lower = error, the same orientation as the spectral score. When the selected band starts at component 0, the top subspace *is* the spectral band, which explains the ties in Table 3.

Alternatives to attention-guided rescoring. Every row of Table 4 starts from the same base score S and changes only how the correction is formed. The two content-agnostic rows keep the form $\widetilde{S}(s_i) = S(s_i) + \gamma B_i$ of Eq. 7, with the selected γ, and replace the dependency-weighted average of Eq. 6 by a rule that ignores the attention:

$$B_i^{\text{norm}} = \frac{1}{T-i} \sum_{t>i} S(s_t), \qquad B_i^{\text{unnorm}} = \sum_{t>i} S(s_t). \tag{10}$$

The normalized form gives every successor the same weight; the unnormalized form sums them, so an early step with many successors collects more.

The three position heuristics drop the propagation and act on the base score alone. *Temporal bias* subtracts a penalty that grows with the position of the step,

$$\widetilde{S}^{\text{base}}(s_t) = S(s_t) - \lambda \frac{t}{T}, \qquad \widetilde{S}^{\text{std}}(s_t) = z\big(S(s_t)\big) - \lambda \frac{t}{T}, \tag{11}$$

where $z(\cdot)$ standardizes the base scores within the trajectory to zero mean and unit variance, and λ is selected per column from $\{0.1, 0.2, \ldots, 1.0\}$ by the same rule as every other hyperparameter. *Earliest of top*-5 predicts the smallest index among the five steps with the largest base score. Each heuristic pushes blame toward earlier steps without asking which earlier step the later ones depend on, which is the comparison Table 4 makes.

A.4 Baselines

Prompt-based methods. All-at-Once, Step-by-Step, and Binary Search run the code released with Who&When (Zhang et al., 2025c). CORRECT (Yu et al., 2025) and CHIEF (Wang et al., 2026) are reproduced from their released code with the prompts kept unchanged. ErrorProbe (Li et al., 2026) runs the pipeline of its paper: a failure-mode tagger and a dependency graph over chunks of the trajectory, then a Strategist, an Investigator, and an Arbiter that names the decisive step. For every method we adopt the decoding settings its paper recommends. The with-GT and without-GT runs differ in one line of the prompt, which states the gold answer. We did not run the GPT-5 judge on the Correct-Error corpus, whose 2,226 trajectories and multi-call protocols made it the most expensive cell of the sweep; Appendix C.3 therefore omits it.

OAT. OAT models the latent dynamics of *successful* trajectories and scores each step of a failed trajectory by how far it departs from them. A neural controlled differential equation predicts each step's latent state from the trajectory so far; the anomaly score is the squared distance between the predicted and observed states, and the predicted decisive step is the argmax. Training is one-class on the successful tool-use trajectories released with the paper (MCP-Atlas), following the paper's own out-of-distribution protocol; one trained model serves every subset. In the with-GT setting the gold answer is appended to the task description with the same phrasing the prompt-based baselines use; the training corpus carries no answers, so the model is shared between settings.

StepFinder. StepFinder is a supervised step classifier. Each step's text and agent name are embedded separately, and the sequence passes through a BiLSTM, an agent-aware attention layer, and a scoring head with a position prior, trained with a per-trajectory cross-entropy loss plus a self-supervised next-step objective. Training uses the step-labeled failure corpora released with the paper. To place StepFinder on the same backbones as Soap, we swap its original encoder for the backbone's last-layer hidden states. In the with-GT setting the gold answer is appended to the first step, since StepFinder encodes no separate task description; the models are shared between settings.

Evaluation. OAT and StepFinder score steps but name no agent; their responsible-agent prediction is the agent of the predicted step. The prompt-based judges name an agent directly, as the Who&When prompts request, and that name is scored. We train five models per backbone; a reported cell averages over the subset's three test splits and, within each split, the five training seeds. All methods follow the same scoring rules as Table 1.

A.5 WITH-GROUND-TRUTH SETTING

How SOAP receives the answer. In the with-GT setting, the LLM's context opens with a pinned block, *"The problem is: {question}. The answer for the problem is: {answer}"*, phrased as in the prompting baselines so the LLM reads the answer in the same register. The block precedes every turn and survives context truncation. OAT appends the answer to the task description and StepFinder to the first step, as described in Appendix A.4.

Full results. Table 6 extends Table 2 to all five subsets, the GPT-4o judge, and DeepSeek-8B. SOAP is best on every column of both open-backbone blocks, tied only by Step-by-Step on TE-MAG with Qwen3.5-9B (21.01). The answer helps the judges more than it helps SOAP: Step-by-Step gains 21 points on CE with Qwen3.5-9B and 15 with GPT-4o, while SOAP moves by under 5 points everywhere except DeepSeek-8B TE-MAG (+6.5). The closest a judge gets is on CE with the answer, where Binary Search and ErrorProbe with Qwen3.5-9B reach 59.89 and 59.57 against SOAP's 60.59.

Table 6: Step-level accuracy (%) in the with-ground-truth setting on all five subsets. The prompt-based rows use the judge named in the block; the training-based rows and SOAP use the backbone named in the block. The best result in each column within a block is highlighted in **bold**.

Method	WW-AG	WW-HC	CE	TE-CAP	TE-MAG
GPT-4o					
All-at-Once	15.87	3.45	24.50	9.30	6.52
Step-by-Step	22.75	18.39	59.29	18.60	19.57
Binary Search	28.57	4.60	18.44	9.30	6.52
CORRECT	15.87	4.60	37.09	10.85	10.14
CHIEF	24.34	11.49	42.10	14.73	13.77
ErrorProbe	**34.92**	**28.74**	**62.26**	**30.23**	**26.81**
Qwen3.5-9B					
All-at-Once	21.16	11.49	33.32	6.98	1.45
Step-by-Step	17.46	18.39	52.04	24.81	**21.01**
Binary Search	2.12	13.79	59.89	0.00	10.14
CORRECT	32.28	4.60	36.43	10.08	8.70
CHIEF	30.69	4.60	30.78	6.98	10.87
ErrorProbe	40.74	19.54	59.57	24.81	20.29
StepFinder	18.52	12.87	28.52	13.64	8.99
OAT	22.12	8.05	56.21	17.98	20.00
SOAP (w/o rescoring)	41.27	33.33	60.30	32.56	15.94
SOAP	**43.39**	**34.48**	**60.59**	**36.43**	**21.01**
DeepSeek-R1-Distill-Llama-8B					
All-at-Once	21.16	5.75	24.16	8.53	4.35
Step-by-Step	15.34	3.45	16.51	10.85	5.07
Binary Search	6.35	12.64	40.01	4.65	7.25
CORRECT	19.58	11.49	36.19	14.73	1.45
CHIEF	14.29	11.49	10.95	4.65	0.00
ErrorProbe	18.52	14.94	40.39	21.71	11.59
StepFinder	18.94	9.89	24.99	15.04	6.23
OAT	20.85	17.01	52.39	18.14	13.77
SOAP (w/o rescoring)	40.21	29.89	64.81	41.09	35.51
SOAP	**44.97**	**29.89**	**65.19**	**42.64**	**36.96**

A.6 COMPUTE RESOURCES AND COST

We compare execution time on the frozen Qwen3.5-9B backbone. All experiments use Python 3.10, PyTorch 2.13, and Transformers 5.15 on one NVIDIA H200 (141 GB), processing one trajectory at a time.

Fitting and inference time. Table 7 reports fitting time, forward-pass time, and CPU scoring time per trajectory. SOAP fits its SVD on the unlabeled reference set in 0.08 s on WW-AG and 0.29 s on WW-HC, two to three orders of magnitude faster than OAT and StepFinder (51–115 s per training seed). One forward pass costs 0.38 s on WW-AG and 0.76 s on WW-HC, on par with OAT (0.42

and 2.13 s) and StepFinder (0.23 and 0.19 s). Scoring the stored vectors then takes 0.7 and 5.5 ms per trajectory, the fastest of the three.

Table 7: Fitting and inference time on WHO&WHEN with Qwen3.5-9B. Fitting is per training seed (mean of five) for OAT and StepFinder, one CPU SVD fit for SOAP. Extraction is one forward pass of the backbone.

	Fitting (s/fit)		Embedding extraction (s/forward pass)		CPU scoring (ms/trajectory)	
Method	WW-AG	WW-HC	WW-AG	WW-HC	WW-AG	WW-HC
OAT	51.10	51.10	0.42	2.13	2.4	10.4
StepFinder	51.79	115.43	0.23	0.19	1.8	5.0
SOAP	**0.08**	**0.29**	0.38	0.76	**0.7**	**5.5**

Comparison with prompt-based methods. Table 8 compares all methods on the same 40 WW-AG and 17 WW-HC trajectories with Qwen3.5-9B, without the ground-truth answer. Each judge is timed over all of its requests per trajectory, prompt processing and decoding included; SOAP is timed over one forward pass plus CPU scoring. SOAP takes 0.38 s on WW-AG and 0.76 s on WW-HC, faster than every judge except Binary Search (0.2 and 0.4 s). The multi-call judges are the slowest: CHIEF takes 24 s and ErrorProbe 6 s per trajectory.

A.7 SYNTHETIC REFERENCE TRAJECTORIES

The synthetic references of Figure 4(d) come from re-running the agent systems behind the two WHO&WHEN subsets, CaptainAgent for WW-AG and Magentic-One for WW-HC, once with Qwen3.5-9B and once with GPT-4o as the backbone of their agents. The systems run on the same question pools the benchmarks draw from (GAIA and AssistantBench), and we keep the trajectories whose question belongs to the target subset. The runs are unsupervised: the corpus mixes successes and failures, carries no step-level annotation, and serves only to construct R. The existing benchmarks and our synthetic trajectories share a question pool but never a trajectory.

Table 8: Time per trajectory (s) on the shared WHO&WHEN subset.

Method	AG	HC
All-at-Once	2.0	2.1
Step-by-Step	2.0	3.6
Binary Search	0.2	0.4
CORRECT	1.9	1.6
CHIEF	23.9	23.5
ErrorProbe	5.9	6.5
SOAP	0.38	0.76

The evaluation follows Table 1; only the reference changes. Our GPT-4o corpus differs from the WHO&WHEN trajectories despite the shared model name for two reasons: WHO&WHEN was generated with the `2024-08-01-preview` version of GPT-4o, whereas ours uses `2024-08-06`, and we rebuilt the generation pipeline from the paper's description because the benchmark does not release its generation code. Appendix C.6 reports the full results for both backbones.

B ADDITIONAL ABLATION STUDIES

The main text ablates SOAP on WW-AG and WW-HC with backbone Qwen3.5-9B (Section 4.3). This appendix extends the ablations to the remaining cells, TE-CAP and TE-MAG on Qwen3.5-9B and all four subsets on DeepSeek-8B, under the same protocol: start from the configuration selected for Table 1, vary one axis, hold everything else fixed.

B.1 CONTEXT WINDOW OF THE DEPENDENCY WEIGHTS

Table 9 varies the window w, which controls the number of predecessors each step distributes its blame over, with everything else at the selected configuration. The preference is dataset-dependent. On WHO&WHEN both backbones favor the sharpest window: accuracy peaks at w=1–2 and declines toward w=all (47.62 vs. 40.21 on Qwen3.5-9B WW-AG), because restricting each step to its strongest dependencies keeps the correction targeted rather than diluting it toward the uniform weights of Table 4. On TE-CAP the pattern reverses: wider windows win on both backbones, up to w=all on DeepSeek-8B. A poorly chosen w can fall below the base score (Qwen3.5-9B TE-MAG at w=1: 15.94 vs. 21.01), confirming that w is worth selecting rather than fixing.

Table 9: Context window w of the dependency weights. Step-level accuracy (%) when each step keeps its w strongest predecessors, everything else at the selected configuration; the selected w is highlighted in **bold**. DeepSeek-8B's selected γ on WW-HC is 0, so that row is constant by construction.

Backbone	Subset	$w = 1$	2	3	4	5	all
Qwen3.5-9B	WW-AG	**47.62**	41.80	40.74	40.21	40.21	40.21
	WW-HC	33.33	**34.48**	32.18	32.18	32.18	32.18
	TE-CAP	28.68	29.46	34.88	34.88	**35.66**	34.88
	TE-MAG	15.94	20.29	18.84	**23.19**	18.84	18.12
DeepSeek-8B	WW-AG	**45.50**	38.62	33.33	34.39	37.57	37.57
	WW-HC	28.74	28.74	28.74	28.74	28.74	**28.74**
	TE-CAP	35.66	38.76	41.09	41.09	41.86	**42.64**
	TE-MAG	**30.43**	30.43	30.43	29.71	28.99	28.99

B.2 SENSITIVITY TO THE HYPERPARAMETERS ON ALL CELLS

Figures 7 and 8 repeat Figure 5 on all eight (backbone, subset) cells and add the window w as a panel. Two regularities hold across backbones. First, γ has the same two regimes: WW-AG climbs toward large γ on DeepSeek-8B as on Qwen3.5-9B (38.62 at $\gamma = 0$ to 45.50 at $\gamma = 1.0$), while every other cell peaks at $\gamma \in \{0.1, 0.2\}$ and decays beyond, so the correction is a nudge where the base score is already strong and carries most of the signal where it is not. Second, the selected layer is the per-layer optimum of post-rescoring accuracy in seven of the eight cells (the exception is DeepSeek-8B TE-MAG, where block 9 edges the selected block 10 by 1.4 points); WW-HC concentrates in the last block on both backbones, and DeepSeek-8B consistently prefers the late attention band (24–32). These results suggest that the selected configurations sit on broad optima rather than on isolated peaks.

B.3 POOLING OF THE STEP REPRESENTATION

SOAP averages the hidden states over a step's tokens. We compare this with using the last-token representation. As shown in Table 10, mean pooling is clearly better: on WW-AG the base score drops from 39.15 to 29.10 on Qwen3.5-9B and from 38.62 to 29.10 on DeepSeek-8B, and on TE-CAP from 40.31 to 24.81 on DeepSeek-8B; only TE-MAG is close. Rescoring still helps on last-token representations (e.g., 29.10 to 30.69 on DeepSeek-8B WW-AG), but from a much lower base. These results indicate that the spectral signal lives in the step's tokens as a whole rather than in the state at its final token, which mostly reflects what comes next.

Table 10: Mean vs. last-token pooling of the step representation. We compare the step-level accuracy (%); the best result in each column is highlighted in **bold**.

		Qwen3.5-9B				DeepSeek-8B			
Pooling	Method	WW-AG	WW-HC	TE-Cap	TE-Mag	WW-AG	WW-HC	TE-Cap	TE-Mag
Last token	Base	29.10	22.99	24.03	21.01	29.10	22.99	24.81	24.64
	SOAP	29.63	24.14	25.58	23.19	30.69	22.99	24.81	26.09
Mean (Ours)	Base	39.15	33.33	33.33	21.01	38.62	28.74	40.31	29.71
	SOAP	**47.62**	**34.48**	**35.66**	**23.19**	**45.50**	**28.74**	**42.64**	**30.43**

B.4 QUANTITY OF UNLABELED REFERENCE DATA

Table 11 completes Figure 5(d) with the remaining cells and DeepSeek-8B. With a third of the reference corpus (6–12 trajectories) every cell is within about four points of its full-data accuracy, and most within one or two. Because the configuration is re-selected at each fraction on the validation split, a small fit can occasionally score above the full one (DeepSeek-8B WW-HC and TE-MAG at 10%); this is selection noise on tiny fits, not a real gain. These results confirm that a handful of unlabeled trajectories suffices to construct the reference.

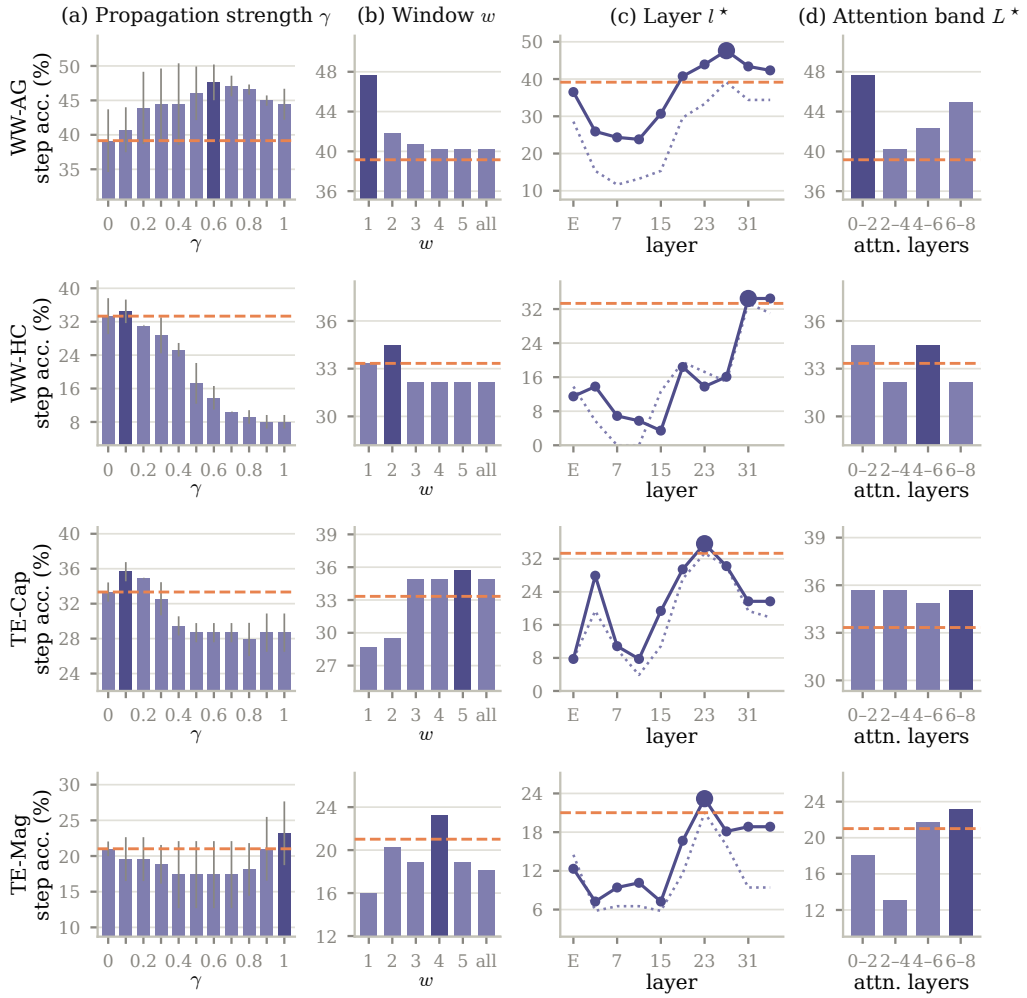

Figure 7: Sensitivity of SOAP on Qwen3.5-9B, all four subsets. As Figure 5, with the window w added: (a) propagation strength γ, (b) window w, (c) representation layer $l^\star$ (solid: after rescoring; dotted: base score at that layer; large marker: selected layer), (d) attention band $L^\star$. The dashed orange line denotes the base score of the selected configuration.

Table 11: Quantity of unlabeled reference data, all cells. Step-level accuracy (%) of SOAP when the reference is fit on $\{10, 20, 30\}\%$ of the corpus, validation and test fixed, full configuration re-selected per fraction. The best result in each column within a block is highlighted in **bold**.

Reference data	WW-AG	WW-HC	TE-CAP	TE-MAG
Qwen3.5-9B				
10%	43.39	33.33	34.88	**23.19**
20%	44.44	33.33	34.88	**23.19**
30%	**47.62**	**34.48**	**35.66**	**23.19**
DeepSeek-R1-Distill-Llama-8B				
10%	37.04	**29.89**	41.86	**31.88**
20%	**46.03**	28.74	**42.64**	30.43
30%	45.50	28.74	**42.64**	30.43

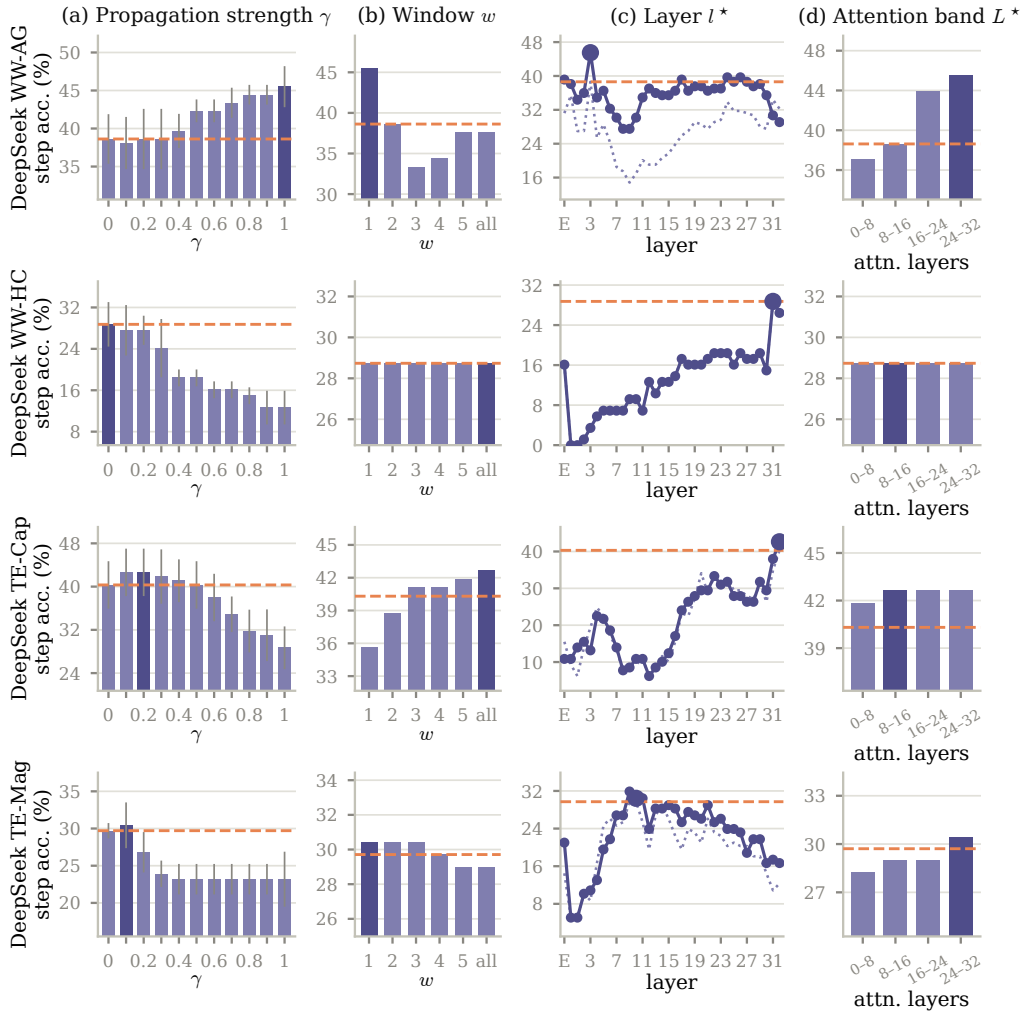

Figure 8: Sensitivity of SOAP on DeepSeek-8B, all four subsets; panels as in Figure 7. DeepSeek-8B sweeps every block in panel (c). On WW-HC the selected γ is 0, so the w and attention-band panels are flat by construction.

B.5 LENGTH AND NORM BASELINES

A natural concern is that the base score only tracks surface properties of a step, such as its length or the norm of its representation. To test this, we compare SOAP with two trivial baselines, which predict the longest or the shortest step, and with a unit-norm control that scales every step representation to $\|v_t\|_2 = 1$ in both the reference and test sets. The control either keeps the spectral band selected on the original representations (*anchor band*) or re-selects it on the normalized ones (*re-selected*), since the best band may shift once the norm is removed. As shown in Table 12, length alone does not locate the decisive step: the longest step scores only 8–20%, 9–32 points below the base score, and the shortest step falls below random on every cell. Removing the norm lowers accuracy but leaves a clear signal: with the re-selected band, the unit-norm score reaches 22–35%, above every length and random row, and trails the base score by 3–13 points on seven of the eight cells, even exceeding it on Qwen3.5-9B TE-MAG (23.19 vs. 21.01). WW-AG loses under 5 points on both backbones, because its selected band skips the top singular vector, which carries most of the norm variation. SOAP achieves the best accuracy on every cell (e.g., 47.62 vs. 35.45 for the best unit-norm score on Qwen3.5-9B WW-AG), tied only by the base score on DeepSeek-8B WW-HC, where the selected γ is 0. These results confirm that the base score captures a directional signal beyond step length and representation norm, and that rescoring builds on this signal.

Table 12: Trivial baselines and the unit-norm control. Step-level accuracy (%) on the test split; the best result in each column is **bold**.

	WW-AG	WW-HC	TE-CAP	TE-MAG
Qwen3.5-9B				
Longest step	19.05	19.54	10.08	11.59
Shortest step	0.53	0.00	4.65	2.90
Random	12.10	5.08	8.58	5.16
Unit-norm, anchor band	35.45	16.09	24.81	16.67
Unit-norm, re-selected	35.45	22.99	30.23	23.19
Base score	39.15	33.33	33.33	21.01
SOAP	**47.62**	**34.48**	**35.66**	**23.19**
DeepSeek-R1-Distill-Llama-8B				
Longest step	15.87	19.54	8.53	12.32
Shortest step	0.53	0.00	4.65	1.45
Random	12.10	5.08	8.58	5.16
Unit-norm, anchor band	29.10	6.90	25.58	10.87
Unit-norm, re-selected	33.86	22.99	27.13	22.46
Base score	38.62	28.74	40.31	29.71
SOAP	**45.50**	**28.74**	**42.64**	**30.43**

B.6 EFFECT OF THE REFERENCE CORPUS

SOAP constructs its spectral reference from the step-unlabeled failure trajectories of the benchmark's reference split. We further examine how the choice of this corpus affects attribution by replacing it with five alternatives. *Random basis* replaces the reference directions with a random orthonormal basis of the same dimension as the selected band, which encodes no textual structure. MCP-ATLAS *successes* and MCP-ATLAS *failures* each draw 50 runs from MCP-ATLAS (Bandi et al., 2026), a corpus of 198 single-agent tool-use runs by Qwen3.5-27B, and therefore serve as out-of-distribution references. *Steps before* $t^\star$ fits R only on the reference steps that precede the annotated decisive step, the closest approximation to a success-only reference that the benchmarks allow. *Random subset* draws the same number of steps uniformly from the full reference split, which separates the effect of using fewer steps from that of using earlier, cleaner steps. The last two references read step labels and serve only as diagnostics. For every reference, we re-select the configuration on the validation split and report the step-level accuracy on the test split.

Table 13: Effect of the reference corpus on WHO&WHEN. Step-level accuracy (%) on the test split. †Diagnostic references that read step labels on the reference split. The best result in each column within a block is highlighted in **bold**.

Reference	WW-AG	WW-HC
Qwen3.5-9B		
Random basis	33.33	25.29
MCP-ATLAS successes	38.10	28.74
MCP-ATLAS failures	42.86	28.74
Steps before $t^{\star\dagger}$	36.51	33.33
Random subset†	39.68	33.33
Benchmark reference split	**47.62**	**34.48**
DeepSeek-R1-Distill-Llama-8B		
Random basis	36.51	27.59
MCP-ATLAS successes	40.21	29.89
MCP-ATLAS failures	44.44	**32.18**
Steps before $t^{\star\dagger}$	44.97	31.03
Random subset†	39.15	27.59
Benchmark reference split	**45.50**	28.74

As shown in Table 13, the benchmark reference performs best on WW-AG with both backbones, outperforming every alternative by up to 14.29 points with Qwen3.5-9B and 8.99 points with

DeepSeek-8B. Moreover, failed trajectories provide a more informative reference than successful ones: MCP-ATLAS failures outperform MCP-ATLAS successes on WW-AG with both backbones (e.g., 42.86 vs. 38.10 with Qwen3.5-9B) and match or exceed them on WW-HC. Removing the decisive step and its downstream steps also hurts with Qwen3.5-9B, lowering WW-AG accuracy from 47.62 to 36.51, below even a random subset of the same size (39.68); with DeepSeek-8B, the steps before $t^\star$ nearly match the full reference (44.97 vs. 45.50). The random basis performs worst on WW-AG, 9–14 points below the benchmark reference. On WW-HC the differences narrow, and with DeepSeek-8B two alternatives even exceed the benchmark reference (32.18 for MCP-ATLAS failures and 31.03 for the steps before $t^\star$, vs. 28.74). These results indicate that failed trajectories carry more informative signal for decisive-error localization.

B.7 COMPARISON WITH A SUPERVISED PROBE

SOAP uses the step-level labels of the 20% validation split only to select its configuration. We further examine whether a supervised classifier trained directly on these labels can localize the decisive error more accurately. The probe is a logistic-regression classifier on the 4096-dimensional mean-pooled hidden state, with one example per step and balanced class weights. It is trained on the validation split alone, and its layer and regularization strength are selected by leave-one-trajectory-out cross-validation within that split. Like SOAP, the probe predicts the step with the largest score in each trajectory, and we report the step-level accuracy on the test split.

Table 14: Comparison with a supervised probe trained on validation labels. Step-level accuracy (%) on the test split. The best result in each column within a block is highlighted in **bold**.

	WW-AG	WW-HC	TE-CAP	TE-MAG
Qwen3.5-9B				
Probe	43.92	20.69	18.60	17.39
SOAP	**47.62**	**34.48**	**35.66**	**23.19**
DeepSeek-R1-Distill-Llama-8B				
Probe	35.45	19.54	20.93	20.29
SOAP	**45.50**	**28.74**	**42.64**	**30.43**

As shown in Table 14, SOAP outperforms the probe on every subset with both backbones, by 3.70–17.06 points with Qwen3.5-9B and 9.20–21.71 points with DeepSeek-8B. The probe comes closest on WW-AG with Qwen3.5-9B (43.92 vs. 47.62), the subset with the largest validation split. The largest gap arises on TE-CAP with both backbones (e.g., 20.93 vs. 42.64 with DeepSeek-8B), suggesting that a probe fitted on a few labeled trajectories overfits them. These results indicate that, when only a few labeled trajectories are available, using them to select the configuration of SOAP is more effective than training a classifier on them.

C ADDITIONAL ANALYSIS

C.1 AGENT-LEVEL ACCURACY AND VARIANCE ACROSS SEEDS

Agent-level accuracy. Table 15 reports the responsible-agent accuracy of every row of Table 1 on WHO&WHEN, i.e., how often the agent of the predicted step is the annotated agent. The metric is far more forgiving than step-level accuracy because a trajectory involves 3.6 agents on WW-AG and 5.7 on WW-HC on average (Table 5), so a method that names the right agent at a wrong step still scores. SOAP is best on both subsets with both open-weight backbones (60.32 on WW-AG and 67.82 on WW-HC with Qwen3.5-9B; 61.90 and 65.52 with DeepSeek-8B). The prompt-based judges close much of their step-level gap here: All-at-Once with GPT-4o reaches 59.77 on WW-HC while locating the step in 6.90% of cases, which says that a judge often blames the right agent for the wrong reason. These results indicate that agent-level accuracy rewards the coarse verdict a judge produces naturally, whereas step-level accuracy measures localization.

Variance across seeds. Table 16 gives the standard deviation over the three seeds of every step-level number in Table 1. On the small subsets one trajectory is worth 1.6 points (WW-AG, 63 test trajectories) to 3.4 points (WW-HC, 29), so standard deviations of 2–5 points are the norm for every

Table 15: Agent-level accuracy (%) in the without-ground-truth setting on WHO&WHEN, for every row of Table 1. Mean over the seed triple. The best result in each column within a block is highlighted in **bold**.

Method	WW-AG	WW-HC
GPT-4o		
All-at-Once	53.44	**59.77**
Step-by-Step	30.16	49.43
Binary Search	**58.73**	12.64
CORRECT	54.50	54.02
CHIEF	40.21	48.28
ErrorProbe	56.08	55.17
Qwen3.5-9B		
All-at-Once	56.61	48.28
Step-by-Step	19.05	47.13
Binary Search	38.10	62.07
CORRECT	59.79	47.13
CHIEF	50.26	54.02
ErrorProbe	53.44	65.52
StepFinder	39.26	37.01
OAT	40.53	53.10
SOAP	**60.32**	**67.82**
DeepSeek-R1-Distill-Llama-8B		
All-at-Once	42.33	45.98
Step-by-Step	44.97	37.93
Binary Search	28.57	48.28
CORRECT	44.97	32.18
CHIEF	37.04	52.87
ErrorProbe	43.39	47.13
StepFinder	48.99	41.15
OAT	42.33	56.32
SOAP	**61.90**	**65.52**

method, and differences below that on a single subset should not be read; CE, with over a thousand test trajectories, is stable to under one point. SOAP's margin over the strongest prompt-based baseline on WW-AG (47.62 vs. 39.68 for ErrorProbe with Qwen3.5-9B) is about three standard deviations, and its margins over the training-based methods and over every judge on DeepSeek-8B are larger still.

C.2 ACCURACY AT k

SOAP, OAT, and StepFinder each produce a per-step score, so the top-k steps form a ranked shortlist for triage; the prompt-based judges return a single verdict and are excluded. Table 17 reports step-level accuracy at $k \in \{1, 3, 5\}$ and mean reciprocal rank (MRR) on the test splits of Table 1. For OAT and StepFinder we rank their stored scores; a step that OAT never scored counts as a miss at every k.

SOAP holds its lead as the shortlist grows. On WW-AG with Qwen3.5-9B, it locates the decisive step in 74.60% of trajectories at k=3, against 56.72 for OAT and 55.24 for StepFinder, and in 91.01% at k=5. On WW-HC, whose trajectories average 52 turns, the k=5 gap is 12 points over OAT (50.57 vs. 38.39). The baselines close in on CORRECT-ERROR, where five candidates already cover most of a trajectory (Table 5). MRR confirms the ranking: 63.92 on WW-AG with Qwen3.5-9B, against 41.46 for OAT and 41.23 for StepFinder.

C.3 RESULTS WITH GPT-5 AS THE JUDGE

Table 18 repeats the prompt-based rows of Tables 1 and 2 with GPT-5 as the judge on WHO&WHEN and TRACEELEPHANT; the SOAP rows are those of the main tables. We leave out CORRECT-ERROR: its 2,226 trajectories and the multi-call protocols make it too expensive to run with GPT-5. GPT-5 lifts every judge, Binary Search most (41.80 on WW-AG without the answer) and ErrorProbe

Table 16: Step-level accuracy (%) in the without-ground-truth setting as mean ± standard deviation over the three seeds of the frozen triple, for every row of Table 1. OAT and StepFinder are first averaged over their five training seeds within each split.

Method	WW-AG	WW-HC	CE	TE-CAP	TE-MAG
GPT-4o					
All-at-Once	14.81 ±2.70	6.90 ±0.00	28.39 ±0.36	16.28 ±3.29	5.80 ±2.05
Step-by-Step	20.63 ±1.30	13.79 ±5.63	44.47 ±1.25	13.95 ±6.85	13.04 ±3.55
Binary Search	22.22 ±3.43	4.60 ±1.63	9.70 ±0.82	9.30 ±1.90	6.52 ±0.00
CORRECT	15.34 ±0.75	4.60 ±1.63	35.12 ±1.93	10.85 ±4.78	13.77 ±2.71
CHIEF	25.93 ±3.96	14.94 ±1.63	38.56 ±0.86	13.95 ±1.90	8.70 ±3.07
ErrorProbe	**38.62** ±1.98	**21.84** ±3.25	**59.52** ±1.49	**27.13** ±2.19	**28.26** ±3.07
Qwen3.5-9B					
All-at-Once	17.46 ±2.59	11.49 ±4.30	33.47 ±0.53	4.65 ±1.90	1.45 ±2.05
Step-by-Step	10.58 ±1.50	16.09 ±1.63	31.15 ±2.46	14.73 ±2.90	10.87 ±4.70
Binary Search	2.65 ±0.75	13.79 ±5.63	59.92 ±0.88	0.78 ±1.10	5.07 ±3.69
CORRECT	29.10 ±2.99	3.45 ±0.00	36.35 ±0.46	7.75 ±1.10	2.90 ±2.71
CHIEF	28.57 ±2.24	3.45 ±0.00	29.39 ±0.65	3.88 ±1.10	2.17 ±1.77
ErrorProbe	39.68 ±2.24	21.84 ±5.86	55.03 ±1.21	23.26 ±3.80	20.29 ±2.71
StepFinder	15.87 ±0.78	13.33 ±3.39	30.03 ±1.07	18.29 ±3.90	6.67 ±0.54
OAT	16.72 ±0.79	10.11 ±6.33	56.50 ±0.82	15.35 ±1.52	18.84 ±2.76
SOAP	**47.62 ±2.59**	**34.48 ±2.82**	**61.78 ±0.73**	**35.66 ±1.10**	**23.19 ±4.47**
DeepSeek-R1-Distill-Llama-8B					
All-at-Once	7.41 ±3.26	5.75 ±1.63	21.95 ±0.85	4.65 ±1.90	2.90 ±1.02
Step-by-Step	16.93 ±1.98	9.20 ±1.63	9.04 ±1.74	11.63 ±1.90	4.35 ±1.77
Binary Search	7.94 ±1.30	12.64 ±3.25	34.15 ±0.94	10.08 ±5.80	13.77 ±2.71
CORRECT	20.11 ±1.98	11.49 ±3.25	32.82 ±1.12	3.88 ±1.10	0.72 ±1.02
CHIEF	17.46 ±3.43	2.30 ±1.63	8.40 ±0.42	7.75 ±1.10	9.42 ±2.71
ErrorProbe	24.34 ±1.98	10.34 ±2.82	35.97 ±0.41	17.05 ±2.90	8.70 ±1.77
StepFinder	18.94 ±2.84	10.80 ±3.10	36.91 ±0.22	14.73 ±0.79	4.20 ±1.08
OAT	12.70 ±2.99	15.86 ±8.29	52.50 ±1.13	17.98 ±0.22	13.04 ±1.98
SOAP	**45.50 ±2.70**	**28.74 ±4.30**	**64.68 ±0.42**	**42.64 ±4.39**	**30.43 ±3.07**

Table 17: Step-level accuracy at k (%) and mean reciprocal rank (MRR, %) in the without-ground-truth setting for the three methods that rank every step. The best result in each column within a block is highlighted in **bold**.

	WW-AG			WW-HC			CE			TE-CAP			TE-MAG			MRR
Method	@1	@3	@5	@1	@3	@5	@1	@3	@5	@1	@3	@5	@1	@3	@5	WW-AG
Qwen3.5-9B																
OAT	16.72	56.72	86.46	10.11	25.75	38.39	56.50	74.60	82.18	15.35	53.02	73.80	18.84	32.17	40.29	41.46
StepFinder	15.87	55.24	75.87	13.33	29.66	34.48	30.03	59.04	68.58	18.29	37.21	50.70	6.67	17.10	28.12	41.23
SOAP	**47.62**	**74.60**	**91.01**	**34.48**	**42.53**	**50.57**	**61.78**	**79.61**	**85.74**	**35.66**	**63.57**	**75.97**	**23.19**	**32.61**	**42.75**	**63.92**
DeepSeek-R1-Distill-Llama-8B																
OAT	12.70	45.19	83.92	15.86	37.01	42.30	52.50	74.08	83.17	17.98	47.13	75.04	13.04	31.01	41.45	37.62
StepFinder	18.94	60.32	78.94	10.80	21.61	29.20	36.91	63.05	74.12	14.73	41.24	58.60	4.20	11.01	21.45	43.75
SOAP	**45.50**	**76.19**	**93.12**	**28.74**	**42.53**	**59.77**	**64.68**	**80.30**	**87.41**	**42.64**	**62.02**	**78.29**	**30.43**	**39.13**	**44.20**	**63.71**

on WW-AG (49.21), so the judge's capacity rather than the prompting protocol sets these methods' accuracy. Even so, SOAP on a 9B or 8B backbone stays ahead of every GPT-5 judge on WW-HC, TE-CAP, and (on DeepSeek-8B) TE-MAG without the answer; on WW-AG, ErrorProbe with GPT-5 overtakes it (49.21 vs. 47.62 without the answer, 53.44 vs. 43.39 with it). These results indicate that a stronger judge narrows the gap on the short algorithm-generated trajectories and in the answer-aided setting, while SOAP remains the most accurate method on the long agentic trajectories with no judge at all.

C.4 LARGER BACKBONES

Table 19 reports the numbers behind Figure 4(a–b) with seed standard deviations and baselines at every size. Within the Qwen3.5 family, SOAP holds its level from 9B to 27B: 47.62 vs. 44.97 on WW-AG (a gap of about one standard deviation) and 34.48 on WW-HC at both sizes. The Qwen3-

Table 18: Step-level accuracy (%) of the prompt-based methods with GPT-5 as the judge, without and with the ground-truth answer, on WHO&WHEN and TRACEELEPHANT. CORRECT-ERROR was not run with GPT-5. The SOAP rows repeat Tables 1 and 2. The best result in each column within a block is highlighted in **bold**.

Method	WW-AG	WW-HC	TE-CAP	TE-MAG
Without GT				
All-at-Once	18.52	4.60	10.85	10.87
Step-by-Step	24.34	18.39	17.83	6.52
Binary Search	41.80	21.84	24.03	20.29
CORRECT	24.87	13.79	20.93	23.91
CHIEF	27.51	25.29	22.48	16.67
ErrorProbe	**49.21**	16.09	26.36	20.29
SOAP, Qwen3.5-9B	47.62	**34.48**	35.66	23.19
SOAP, DeepSeek-8B	45.50	28.74	**42.64**	**30.43**
With GT				
All-at-Once	17.99	9.20	10.85	13.77
Step-by-Step	28.04	24.14	20.16	15.22
Binary Search	47.09	24.14	31.01	25.36
CORRECT	32.80	21.84	27.13	20.29
CHIEF	29.63	18.39	14.73	22.46
ErrorProbe	**53.44**	17.24	29.46	19.57
SOAP, Qwen3.5-9B	43.39	**34.48**	36.43	21.01
SOAP, DeepSeek-8B	44.97	29.89	**42.64**	**36.96**

14B point is competitive on WW-AG (42.86) but 9 points lower on WW-HC; because it also crosses model generations, this gap reflects a family effect as much as a size effect. Neither baseline scales with model size. OAT stays within 14–20 on WW-AG and 10–17 on WW-HC, and StepFinder moves without a trend, with training-seed standard deviations of 1–6 points. The margin of Table 1 holds at every size we tested.

Table 19: Step-level accuracy (%) by backbone size on WHO&WHEN, mean ± standard deviation (over the seed triple for SOAP; over the five training seeds for OAT and StepFinder). Qwen3-14B belongs to the previous model generation. The best result in each column is highlighted in **bold**.

Subset	Method	Qwen3.5-9B	Qwen3-14B	Qwen3.5-27B
WW-AG	OAT	$16.72_{\pm 1.56}$	$13.97_{\pm 1.48}$	$20.42_{\pm 1.40}$
	StepFinder	$15.87_{\pm 2.99}$	$22.65_{\pm 3.72}$	$13.54_{\pm 2.08}$
	SOAP	$\mathbf{47.62}_{\pm \mathbf{2.59}}$	$\mathbf{42.86}_{\pm \mathbf{4.67}}$	$\mathbf{44.97}_{\pm \mathbf{2.70}}$
WW-HC	OAT	$10.11_{\pm 1.13}$	$16.78_{\pm 2.00}$	$12.41_{\pm 1.13}$
	StepFinder	$13.33_{\pm 4.80}$	$14.02_{\pm 2.94}$	$16.55_{\pm 6.24}$
	SOAP	$\mathbf{34.48}_{\pm \mathbf{2.82}}$	$\mathbf{25.29}_{\pm \mathbf{6.50}}$	$\mathbf{34.48}_{\pm \mathbf{2.82}}$

C.5 CROSS-DISTRIBUTION TRANSFER

Figure 9 gives the complete transfer grids for both backbones; the Qwen3.5-9B grid repeats Figure 4(c). For each source–target pair, the reference R is fitted on the source's reference split, and the configuration is re-selected on the target's validation split, which for the cross setting pools the target's reference and validation files, since the target's reference split is not used for fitting; the reported number is the resulting test accuracy. The diagonal of each grid is the in-distribution number of Table 1. The diagonal wins every column on both backbones, so a reference fitted on the target's own distribution remains the best choice. The degradation off the diagonal, however, is graded, not catastrophic: DeepSeek-8B WW-HC→WW-AG keeps 39.68 of the in-distribution 45.50, and WW-HC→TE-CAP keeps 35.66 of 42.64.

C.6 SYNTHETIC REFERENCES, FULL RESULTS

Table 20 extends Figure 4(d) to DeepSeek-8B (generation details in Appendix A.7). On WW-AG, the existing benchmark reference remains strongest (45.50), while both synthetic corpora stay within

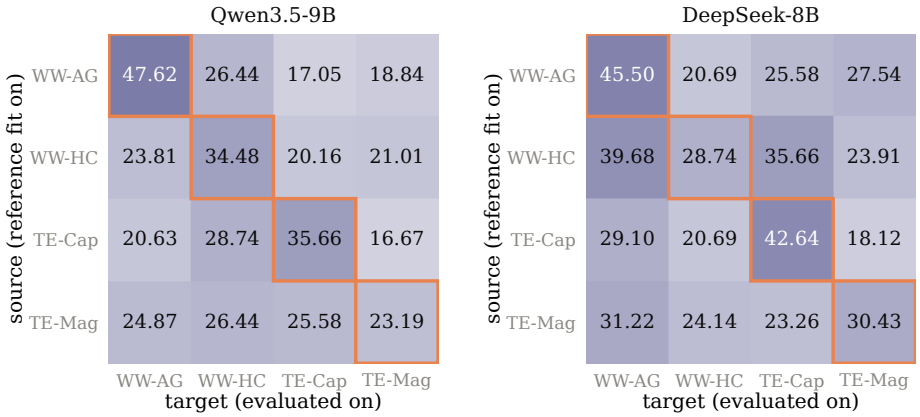

Figure 9: Cross-distribution transfer, all grids. Step-level accuracy (%) of SOAP with the reference fitted on the source subset (rows) and evaluated on the target subset (columns), for both backbones. The outlined diagonal is the in-distribution number of Table 1 in every grid.

Table 20: **Synthetic references with DeepSeek-8B.** Step-level accuracy (%) of SOAP; only the reference corpus changes.

Reference	WW-AG	WW-HC
Existing benchmarks	**45.50**	28.74
Synthetic, Qwen3.5-9B	39.68	**29.89**
Synthetic, GPT-4o	38.10	25.29

6–7 points (39.68 and 38.10). On WW-HC the Qwen3.5-9B-generated corpus slightly exceeds the existing benchmark reference (29.89 vs. 28.74). Neither generator consistently dominates: GPT-4o wins on WW-AG and Qwen3.5-9B wins on WW-HC, matching the pattern observed with the Qwen3.5-9B backbone in Figure 4(d). All synthetic rows remain well above the training-based baselines of Table 1.

C.7 QUALITATIVE EXAMPLES

Figures 10–13 show four textual trajectories in the test set, all scored with Qwen3.5-9B under the configuration selected for Table 1. We denote step i as S_i. Each step shows its agent, $[S \to \widetilde{S}]$ (base and rescored score, $\times 10^{-3}$), and its content condensed by us; tags mark the annotated decisive step and each score's argmax. The third example (Figure 12) is the trajectory whose score plot appears in Figure 6. Agent names and step indices are as released. The WW-HC trajectory (Figure 11) uses Magentic-One, whose orchestrator alternates between private ledger entries and dispatches: (thought) marks a ledger entry where the orchestrator reasons about progress, and $\to$ marks an instruction sent to an agent.

The four trajectories fail in different ways, but the base score misreads them in the same way: it flags a late symptom rather than the step that caused it. SOAP rescores each step by accounting for the attention its successors pay to it, correcting this downstream contamination. For each example we give the task, the annotated decisive step and what went wrong, where the base score peaks, and where SOAP lands.

- **WW-AG** (Figure 10). *Task:* identify the command clicked in the last video of a 2018 VSCode blog post on replit.com. *Decisive step, S_1:* the video-analysis expert submits code that loops over a `None` return from `perform_web_search`; the loop crashes immediately. *Base score:* peaks at S_2, the `TypeError` traceback. SOAP*:* lifts S_1, the buggy call that produced the crash.
- **WW-HC** (Figure 11). *Task:* price mailing a DVD to Colombia with three carriers. *Decisive step, S_8:* the WebSurfer ignores the instruction to look up DHL, clicks a FedEx rate tool, and returns an empty form with no rate. The orchestrator records FedEx as done, and every later turn builds on that belief. *Base score:* peaks at S_{73}, the final search dump after

65 turns of loops and replans, a symptom at the far end of the contamination. SOAP*:* lifts S_8 above it, because the turns that stalled on the FedEx form attend to S_8.

- **TE-CAP** (Figure 12; score plot in Figure 6). *Task:* identify the missing character in an Unlambda program that should output "For penguins". *Decisive step, S_1:* the extraction expert misanalyzes the code's combinator structure, concludes a dot is needed before `s`, and answers "." (the correct answer is "backtick"). *Base score:* peaks at S_5, the orchestrator's final restatement of the wrong answer. SOAP*:* moves the top-1 to S_1, because S_3–S_5 attend to the analysis at S_1.
- **TE-MAG** (Figure 13). *Task:* solve a game-show puzzle by code. *Decisive step, S_6:* the orchestrator fixes a conservative strategy, the same guess for every box. A `NameError` follows at S_8, and at S_9 the orchestrator re-issues the script with the bug fixed and the strategy unchanged. *Base score:* peaks at S_9, the bug fix. SOAP*:* lifts S_6, on which both the crash and the fix depend.

WW-AG · CaptainAgent

Task. In the 2018 VSCode blog post on replit.com, what was the command they clicked on in the last video to remove extra lines?
Gold answer. Format Document.
Annotation. S_1, VideoContentAnalysis_Expert: "The code is incorrect for the task."
Prediction. Base score S_2; SOAP S_1.

$\mathbf{S}_0$ VSCode_Expert [12.32 → 19.94] Restates the manager's brief.
$\mathbf{S}_1$ VideoContentAnalysis_Expert [17.44 → 30.42] **decisive** **SOAP top-1** Submits code that loops over `perform_web_search`'s `None` return. The loop fails.
$\mathbf{S}_2$ Computer_terminal [21.64 → 21.64] **base top-1** `TypeError: 'NoneType' not iterable`.
$\mathbf{S}_3$ VideoContentAnalysis_Expert [15.76 → 28.63] Asks the group to open the link.
$\mathbf{S}_4$ VideoContentAnalysis_Expert [21.45 → 21.45] Repeats the instructions.
$\mathbf{S}_5$ VideoContentAnalysis_Expert [4.87 → 4.87] TERMINATE. No answer.

Figure 10: WW-AG CaptainAgent trajectory, scored with Qwen3.5-9B.

WW-HC · Magentic-One

Task. What is the cheapest option to mail a DVD to Colombia from Hartford, Connecticut using FedEx, DHL, or USPS? (Answer as JSON with keys `sender` and `price (usd)`.) **Gold answer.** USPS, 41.75 USD.
Annotation. S_8, WebSurfer: "did not return the useful information needed to answer the question." **Prediction.** Base score S_{73}; SOAP S_8.
Reading. S_0 is the unscored user question. (thought) marks the orchestrator's ledger to itself, → its instruction to an agent.

$\mathbf{S}_1$ Orchestrator (thought) [1.74 → 1.88] Initial plan: fact sheet (a DVD, Hartford to Colombia, three carriers); rates to look up per carrier; team Assistant, ComputerTerminal, FileSurfer, WebSurfer.
$\mathbf{S}_2$ Orchestrator (thought) [1.55 → 1.69] Ledger: nothing gathered; start with FedEx.
$\mathbf{S}_3$ Orchestrator → WebSurfer [1.49 → 1.49] Look up FedEx rates.
$\mathbf{S}_4$ WebSurfer [1.71 → 1.90] Types the query into Bing; returns a screenshot and page metadata.
$\mathbf{S}_5$ Orchestrator (thought) [1.82 → 2.00] Ledger: no rates yet; progress.
$\mathbf{S}_6$ Orchestrator → WebSurfer [1.57 → 1.57] Look up DHL rates.
$\mathbf{S}_7$ Orchestrator (thought) [0.92 → 0.92] Next speaker: WebSurfer.
$\mathbf{S}_8$ WebSurfer [2.59 → 2.74] **decisive** **SOAP top-1** Ignores the DHL instruction; clicks "Calculate Shipping Rates – FedEx". Returns the FedEx rate-tool landing page: screenshot, schema.org metadata, an empty form, no rate.
$\mathbf{S}_9$ Orchestrator (thought) [1.82 → 1.95] Ledger: "we have gathered information for FedEx"; DHL and USPS pending. No FedEx rate was ever read; every later turn builds on this belief.
$\mathbf{S}_{10}$ Orchestrator → WebSurfer [1.56 → 1.56] Look up DHL rates.
$\mathbf{S}_{11}$ Orchestrator (thought) [0.93 → 0.93] Next speaker: WebSurfer.
$\mathbf{S}_{12}$ Orchestrator (thought) [1.51 → 1.68] Ledger: in a loop, errors on FedEx and DHL, no progress; hand to Assistant.
$\mathbf{S}_{13}$ Orchestrator → Assistant [1.51 → 1.51] Get USPS rates, then fix FedEx and DHL.
$\mathbf{S}_{14}$ Orchestrator (thought) [1.59 → 1.71] Next speaker: Assistant.
$\mathbf{S}_{15}$ Assistant [1.45 → 1.61] Writes a two-step plan ("I'll visit the USPS website") but has no browser; returns nothing.
$\mathbf{S}_{16}$ Orchestrator (thought) [1.69 → 1.82] Ledger: rates still missing; try USPS next.
$\mathbf{S}_{17}$ Orchestrator → WebSurfer [1.56 → 1.56] Look up USPS rates.
$\mathbf{S}_{18}$ Orchestrator (thought) [0.91 → 1.02] Next speaker: WebSurfer.
$\mathbf{S}_{19}$ WebSurfer [1.44 → 1.61] Still on the FedEx tool: types "Hartford, Connecticut" into *From*.
$\mathbf{S}_{20}$ Orchestrator (thought) [1.63 → 1.78] Ledger: FedEx partially explored, DHL and USPS still pending.
$\mathbf{S}_{21}$ Orchestrator → WebSurfer [1.47 → 1.47] Continue with USPS.
$\mathbf{S}_{22}$ Orchestrator (thought) [0.94 → 0.94] Next speaker: WebSurfer.
$\mathbf{S}_{23}$ WebSurfer [1.62 → 1.76] Clicks the "Hartford, Connecticut, USA" suggestion.
$\mathbf{S}_{24}$ Orchestrator (thought) [1.57 → 1.71] Ledger: FedEx form partially filled; USPS and DHL incomplete.
$\mathbf{S}_{25}$ Orchestrator → WebSurfer [1.46 → 1.46] USPS rates, then revisit DHL.
$\mathbf{S}_{26}$ Orchestrator (thought) [0.95 → 0.95] Next speaker: WebSurfer.
$\mathbf{S}_{27}$ WebSurfer [1.43 → 1.60] Types "Colombia" into the FedEx *To* field.
$\mathbf{S}_{28}$ Orchestrator (thought) [1.78 → 1.93] Ledger: USPS rates still needed; FedEx form in progress.
$\mathbf{S}_{29}$ Orchestrator → WebSurfer [1.51 → 1.51] Continue with USPS.
$\mathbf{S}_{30}$ Orchestrator (thought) [0.96 → 1.07] Next speaker: WebSurfer.
$\mathbf{S}_{31}$ WebSurfer [1.65 → 1.82] Clicks "Colombia".
$\mathbf{S}_{32}$ Orchestrator (thought) [1.79 → 1.93] Ledger: USPS and DHL rates still missing.
$\mathbf{S}_{33}$ Orchestrator → WebSurfer [1.58 → 1.58] Look up USPS rates.
$\mathbf{S}_{34}$ Orchestrator (thought) [0.96 → 0.96] Next speaker: WebSurfer.
$\mathbf{S}_{35}$ WebSurfer [1.76 → 1.76] Clicks the browser back button; the FedEx form reads From Hartford 06103, destination Colombia; no rate.
$\mathbf{S}_{36}$ Orchestrator (thought) [1.58 → 1.69] Ledger: cheapest option not yet determined.
$\mathbf{S}_{37}$ Orchestrator (thought) [1.67 → 1.79] Stalled; replanning.
$\mathbf{S}_{38}$ Orchestrator (thought) [1.55 → 1.72] New plan: use the USPS International Shipping Calculator.
$\mathbf{S}_{39}$ Orchestrator (thought) [1.80 → 1.96] Ledger: request not yet addressed; proceed with new plan.
$\mathbf{S}_{40}$ Orchestrator → WebSurfer [1.79 → 1.79] Open the USPS calculator, enter the DVD details.
$\mathbf{S}_{41}$ Orchestrator (thought) [1.06 → 1.16] Next speaker: WebSurfer.
$\mathbf{S}_{42}$ WebSurfer [1.87 → 2.02] Bing search "USPS International Shipping Calculator"; results-page metadata.
$\mathbf{S}_{43}$ Orchestrator (thought) [1.72 → 1.86] Ledger: on the right track; click the calculator link.
$\mathbf{S}_{44}$ Orchestrator → WebSurfer [1.50 → 1.50] Click the calculator link, enter the details.
$\mathbf{S}_{45}$ Orchestrator (thought) [1.05 → 1.05] Next speaker: WebSurfer.
$\mathbf{S}_{46}$ WebSurfer [1.52 → 1.71] Clicks "Calculate a Price": the USPS Retail Postage Price Calculator opens, with the destination defaulting to United States.
$\mathbf{S}_{47}$ Orchestrator (thought) [2.00 → 2.18] Ledger: loop, but on the right page; instruction: ZIP 06101, Colombia, $7.5 \times 5.3 \times 0.6$ in, 0.2 lb.
$\mathbf{S}_{48}$ Orchestrator → WebSurfer [1.83 → 1.83] Enter ZIP 06101, Colombia, DVD dimensions and weight.
$\mathbf{S}_{49}$ Orchestrator (thought) [1.01 → 1.01] Next speaker: WebSurfer.
$\mathbf{S}_{50}$ WebSurfer [1.53 → 1.72] Clicks "International".
$\mathbf{S}_{51}$ Orchestrator (thought) [1.80 → 1.97] Ledger: USPS rates not yet obtained.
$\mathbf{S}_{52}$ Orchestrator → WebSurfer [1.76 → 1.76] Same instruction.
$\mathbf{S}_{53}$ Orchestrator (thought) [1.09 → 1.09] Next speaker: WebSurfer.
$\mathbf{S}_{54}$ WebSurfer [2.04 → 2.04] Clicks "International" again.
$\mathbf{S}_{55}$ Orchestrator (thought) [1.81 → 1.99] Ledger: USPS rates not yet retrieved.
$\mathbf{S}_{56}$ Orchestrator → WebSurfer [1.79 → 1.79] Same instruction.
$\mathbf{S}_{57}$ Orchestrator (thought) [1.11 → 1.11] Next speaker: WebSurfer.
$\mathbf{S}_{58}$ WebSurfer [2.24 → 2.46] Types 06101 into the from-ZIP field; destination still United States.
$\mathbf{S}_{59}$ Orchestrator (thought) [1.98 → 2.16] Ledger: USPS rates still pending.
$\mathbf{S}_{60}$ Orchestrator → WebSurfer [1.88 → 1.88] Select Colombia, enter dimensions and weight.
$\mathbf{S}_{61}$ Orchestrator (thought) [1.14 → 1.14] Next speaker: WebSurfer.
$\mathbf{S}_{62}$ WebSurfer [1.72 → 1.89] Clicks the destination-country field.
$\mathbf{S}_{63}$ Orchestrator (thought) [1.79 → 1.96] Ledger: USPS rates not yet obtained.
$\mathbf{S}_{64}$ Orchestrator → WebSurfer [1.71 → 1.71] Select Colombia, enter dimensions and weight.
$\mathbf{S}_{65}$ Orchestrator (thought) [1.08 → 1.08] Next speaker: WebSurfer.
$\mathbf{S}_{66}$ WebSurfer [1.70 → 1.70] Clicks the field again; nothing selected.
$\mathbf{S}_{67}$ Orchestrator (thought) [1.83 → 1.83] Ledger: loop, no progress.
$\mathbf{S}_{68}$ Orchestrator (thought) [1.23 → 1.23] Stalled; replanning.
$\mathbf{S}_{69}$ Orchestrator (thought) [1.49 → 1.67] New plan: retry the USPS calculator from scratch.
$\mathbf{S}_{70}$ Orchestrator (thought) [1.86 → 2.03] Ledger: full answer not yet provided; proceed.
$\mathbf{S}_{71}$ Orchestrator → WebSurfer [1.70 → 1.70] Open the USPS calculator, fill every field.
$\mathbf{S}_{72}$ Orchestrator (thought) [0.97 → 0.97] Next speaker: WebSurfer.
$\mathbf{S}_{73}$ WebSurfer [2.65 → 2.65] **base top-1** Types "USPS Retail Postage Price Calculator" into Bing once more; returns a 13k-character dump of result metadata. The run ends with no rate from any carrier.

Figure 11: WW-HC Magentic-One trajectory, scored with Qwen3.5-9B.

TE-CAP · CaptainAgent

Task. In Unlambda, what exact character or text needs to be added to correct the following code to output "For penguins"? If what is needed is a character, answer with the name of the character. Code: `` `r```````````.F.o.r. .p.e.n.g.u.i.n.s `` **Gold answer.** backtick.
Annotation. S_1, InformationExtraction_Expert: "a missing backtick disrupted function application, preventing the code from correctly constructing and outputting 'For penguins'."
Prediction. Base score S_5; SOAP S_1.

$\mathbf{S}_0$ Orchestrator [0.75 → 0.84] Assembles the team: InformationExtraction_Expert (web search and result parsing), CSVHandling_Expert (Python over local files), Verification_Expert (re-runs checks against cited evidence).
$\mathbf{S}_1$ InformationExtraction_Expert [0.87 → 0.95] **decisive** **SOAP top-1** Analyzes the Unlambda code. Identifies the dot operators and backtick-based function application, but miscounts the applications: concludes a dot is needed before `s` to include it in the output string. Answers "." (dot).
$\mathbf{S}_2$ Orchestrator [0.69 → 0.78] Next speaker: Verification_Expert.
$\mathbf{S}_3$ Verification_Expert [0.82 → 0.90] Agrees with the dot answer. Repeats the reasoning without independent verification of the combinator structure.
$\mathbf{S}_4$ Orchestrator [0.85 → 0.93] Summarizes: the task required identifying the missing character, determined to be a dot before `s`.
$\mathbf{S}_5$ Orchestrator [0.88 → 0.88] **base top-1** Final ANSWER "." with the same justification. TERMINATE. The correct answer is "backtick".

The final answer (S_5) and the summary (S_4) are restatements; the correction lifts the flawed analysis that produced them.

Figure 12: TE-CAP CaptainAgent trajectory (traj-59), scored with Qwen3.5-9B. This is the trajectory whose score plot appears in Figure 6.

TE-MAG · Magentic-One

Task. A game-show host hides 30 coins ($1,000 each) in three boxes; one box holds at least 2 coins and one holds 6 more than another. Bob guesses a count for each box and wins his guess whenever it does not exceed the box, otherwise nothing. With optimal play, what is the minimum amount Bob can win?
Gold answer. 16,000.
Annotation. S_6, Orchestrator: "assuming Bob should make the same conservative guess for each box, which ignored optimizing guesses based on varying coin distributions."
Prediction. Base score S_9; SOAP S_6 (on all three seeds of the triple).

$\mathbf{S}_0$ Orchestrator [0.72 → 1.37] Fact sheet: the givens (30 coins, three boxes, the two placement rules, the payoff rule); nothing to look up.
$\mathbf{S}_1$ Orchestrator [0.61 → 1.22] Plan: enumerate the valid distributions, derive the guess that maximizes the worst case, implement it with Coder, report.
$\mathbf{S}_2$ Coder [0.55 → 1.09] Ledger: proceed to the distributions; not in a loop; progress is being made.
$\mathbf{S}_3$ Orchestrator [0.75 → 1.44] Writes `find_valid_distributions()`: loops over (a, b, c) summing to 30 with $a \geq 2$ and some pair differing by 6; prints the list.
$\mathbf{S}_4$ Coder [0.50 → 1.05] Ledger: run the script.
$\mathbf{S}_5$ Coder [0.52 → 1.10] Ledger: distributions obtained; next, the strategy code.
$\mathbf{S}_6$ Orchestrator [0.70 → 1.44] **decisive** **SOAP top-1** Writes `calculate_min_winnings()`: for every distribution sets all three guesses to `min(a,b,c)` "as a conservative strategy", sums them, and takes the minimum over distributions. The guesses ignore that Bob chooses per box, so the answer collapses to three times the smallest box.
$\mathbf{S}_7$ Coder [0.48 → 1.04] Ledger: run the script.
$\mathbf{S}_8$ Coder [0.56 → 1.15] Ledger: execution failed, `NameError`: `valid_distributions` is undefined; carry the earlier result over; progress: false.
$\mathbf{S}_9$ Orchestrator [0.89 → 1.41] **base top-1** Explains the `NameError` and re-issues both functions as one combined script; the strategy is unchanged.
$\mathbf{S}_{10}$ Coder [0.52 → 1.08] Ledger: execute the combined script.
$\mathbf{S}_{11}$ Coder [0.56 → 1.27] Ledger: "Bob can ensure winning 3 coins"; provide the answer.
$\mathbf{S}_{12}$ Coder [0.71 → 0.71] ANSWER $3,000: the conservative guess always wins at least 3 coins. TERMINATE.

The crash report (S_8) and the fix (S_9) are the symptoms of the strategy fixed in S_6.

Figure 13: TE-MAG Magentic-One trajectory, scored with Qwen3.5-9B.