FINDING ROOT CAUSES, NOT SYMPTOMS: A MECHANISM AND METHOD FOR AGENT FAILURE ATTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

When a multi-agent LLM system fails over a long run, naming the responsible agent is often the easy part. Pinpointing the step where the failure began is hard, and hardest on the long trajectories that matter most. We find that on such runs the dominant error is an *omission*: a necessary action is silently skipped, so it leaves no local trace and surfaces only later as a downstream symptom. Attributors that score steps by local evidence are therefore pulled toward that late symptom, a systematic bias we call *symptom attraction*. But what the run needed to do is determined by the task, not by the trajectory. From the question and its correct answer, which we call the *goal*, the conditions that any successful run must meet can be listed. The condition the skipped action would have met is on that list, even though no step in the trajectory shows it. *GCCA* (Goal-Conditioned Causal Attribution) turns the goal into a chain of necessary preconditions and blames the first step that leaves one unmet. On Who&When, GCCA is the best zero-shot attributor at both naming the agent (70.1%) and pinpointing the step (23.0%) on the long, omission-heavy subset. It also cuts the late bias from as much as $+15.7$ steps to a near-zero $+1.03$. The bias that reveals the problem is thus the bias GCCA removes. GCCA stays competitive on short trajectories, uses the fewest tokens at 4.7k per sample against 8–67k, and holds up across five backbones.

1 INTRODUCTION

Multi-agent systems built from large language models (Guo et al., 2024) split a complex task across specialized agents and can run for tens to hundreds of steps. When such a system fails, one early wrong decision can derail everything after it, in a trajectory too long to inspect by hand. *Failure attribution* (Zhang et al., 2025b) is the task of finding that decision automatically: which agent is responsible (*Who*) and at which step the failure began (*When*). Naming the agent is often feasible; pinpointing the step is much harder, and hardest on the long runs that matter most.

Figure 1 shows a concrete failure. A student asks which pages to study, and the pages were announced only in a lecture recording. At step 5 an agent opens the audio file but never transcribes it. Every later step then proceeds normally on the missing information, until the final answer turns out to be wrong. The decisive mistake is step 5, yet existing methods blame the late steps where the answer visibly breaks down. This is not a one-off: across long trajectories, methods that score steps by local evidence are systematically drawn to the late *symptom* instead of the early root cause. We call this pattern *symptom attraction*.

Why does it happen? The errors that dominate long runs are *omissions*, that is, necessary actions that are simply skipped. A skipped step writes nothing into the trajectory, so a method that reads the trajectory alone sees only the downstream symptoms and is pulled late. But the signal is not gone. Whether an action was required is decided by the goal, not by the trajectory, so the goal still tells us that the skipped action had to happen even though no step shows it. This is exactly the information the trajectory lost.

We turn this observation into a method. *GCCA* (Goal-Conditioned Causal Attribution) does not ask which step looks most faulty. It reads the goal backward into a chain of necessary preconditions and asks which step is the first to leave one unmet, pulling attribution from the symptom back to the root

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.

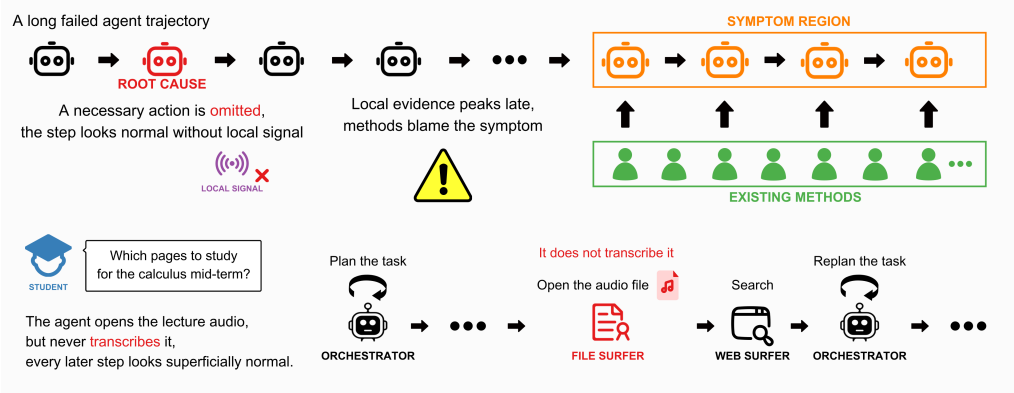


Figure 1: On a real Who&When failure, existing attributors locate the late *symptom*, not the early *root cause*: the agent opens the lecture audio but never transcribes it—an early omission that leaves no local signal—so every later step looks normal and methods blame the visible late steps.

cause. The gains land where the mechanism predicts: on long, omission-heavy trajectories, where the trajectory falls silent, GCCA improves on all three axes at once, while staying competitive on short ones.

We make three contributions. **(i) Phenomenon.** We give the first systematic characterization of *symptom attraction*, the omission-driven late bias by which attributors locate the symptom rather than the root cause. The bias is most pronounced on long trajectories. **(ii) Mechanism.** We give a stylized information-theoretic model: under a locally-silent root cause and local-evidence scoring, an attributor's bias is non-negative and equals the cause–symptom gap. We confirm empirically that this gap grows with trajectory length, and that the lost signal survives in the goal, conditional on the trajectory. **(iii) Method.** We propose GCCA, an attributor that checks the trajectory against a goal-derived necessary-precondition chain. Its gains concentrate on long, omission-heavy trajectories, where it is best on both Who and When and cuts the mechanism-predicted late bias to a near-zero $+1.03$. It is zero-shot on an accessible backbone, uses the fewest tokens, and is stable across five backbones.

2 Symptom Attraction: The Phenomenon and Why It Happens

2.1 Step Localization Collapses on Long Trajectories

Table 1 evaluates six attribution methods on four datasets, ordered by mean trajectory length. Two datasets are the Algorithm-Generated and Hand-Crafted subsets of Who&When (Zhang et al., 2025b); the other two are the Agentic and Math subsets of AFTraj, the corpus released with Agent-Foresight (Zhang et al., 2026). Two patterns stand out. First, step-level accuracy is far below agent-level accuracy almost everywhere. Naming the agent is possible from coarse cues, but pinpointing the step is not. Second, step-level accuracy is worst on the longest dataset: no method exceeds 17.8 on Hand-Crafted, while agent-level accuracy stays high. Pinpointing *when* a failure began is therefore the hard part of attribution, and the difficulty appears in two independent benchmark sources.

Step localization collapses on long trajectories because of the kind of error that dominates them. A *commission* error runs a wrong action, which leaves visibly bad output. An *omission* error skips a necessary action: it never runs, so it produces no output of its own and the steps around it look normal, which is precisely what a reader of the trajectory cannot catch. Omissions dominate long runs: only 4% of Algorithm-Generated failures are omissions against 72% of Hand-Crafted ones (κ=0.69; Appendix G), because a longer task requires more preconditions and each can be quietly dropped. We state this as an idealized limit. Let $R^\star$ be the true root-cause step and $o_{R^\star}$ what that step alone shows. If the omission is *perfectly* silent, that step shows no anomaly at all, so

$$I(R^\star; o_{R^\star}) \approx 0, \tag{1}$$

where $I(\cdot;\cdot)$ denotes mutual information. In words, the content of the step itself tells us almost nothing about whether it is the root cause. Real trajectories are only *partly* silent, since replanning

Table 1: Agent-level and step-level accuracy (%) of six baselines across four datasets, ordered by mean trajectory length $\bar{L}$. Best per column in bold.

	AFTraj-Agentic $\bar{L}$=8.0		Algorithm-Gen. $\bar{L}$=8.7		AFTraj-Math $\bar{L}$=14.3		Hand-Crafted $\bar{L}$=51.6	
Method	Agent↑	Step↑	Agent↑	Step↑	Agent↑	Step↑	Agent↑	Step↑
All-at-Once	**79.6**	28.6	**57.1**	14.8	**93.5**	17.9	58.0	6.9
Step-by-Step	72.1	**53.3**	30.7	21.2	75.3	36.5	**64.4**	17.2
Binary Search	20.7	16.0	50.3	20.6	15.9	2.5	23.0	6.9
A2P	78.2	51.9	56.1	**33.6**	74.3	35.5	59.8	**17.8**
ECHO	79.4	52.7	45.2	22.8	92.9	**74.6**	47.1	15.5
CHIEF	73.8	38.2	50.3	23.0	88.4	53.4	55.2	8.0

and later references leak some signal, so we treat Eq. (1) as an idealization. The phenomenon itself supports it: if the root cause were visible locally, readers of the trajectory would not miss it so consistently (Table 1).

2.2 SYMPTOM ATTRACTION

When these methods pick the wrong step, they do not miss at random. They miss in one direction: they pick a step that is too *late*. An omitted action does not cause a visible failure at once; the failure appears later, where the missing action was needed. A method that scores each step by its local evidence is therefore pulled toward that late *symptom* and away from the early root cause. We call this *symptom attraction*.

Figure 2 measures the effect with prediction bias, the signed distance from the predicted step to the true step, where a positive value means the prediction is late. We report it for the four baselines whose bias is meaningful; the other two pick a step by search order rather than by evidence. The bias is near zero on short trajectories and sharply positive on the long Hand-Crafted set, where the two most thorough readers, CHIEF and ECHO, reach $+15.72$ and $+15.21$. The more closely a method follows the visible trajectory, the more strongly it is drawn late.

This direction follows from Eq. (1). The consequences of the missing action appear downstream, so the evidence about $R^\star$ is not absent but *displaced*. Write $R^\star \to r_1 \to \cdots \to r_m$ for the chain of steps from the root cause to the symptom, and o_{r_i} for what step r_i shows. The information that a step carries about $R^\star$ then rises as we move toward the symptom,

$$I(R^\star; o_{R^\star}) \approx 0 \;<\; I(R^\star; o_{r_1}) \;<\; \cdots \;<\; I(R^\star; o_{r_m}), \tag{2}$$

An attributor that ranks steps by local evidence, $\hat{r} = \arg\max_i P(\text{mistake} \mid \text{step}_i, \tau)$, is therefore pulled toward the symptom r_m and away from the root cause. This is why its signed error runs late. Appendix A states this as a stylized identity; it is not a law that holds for every attributor. Whether the late pull grows with trajectory length is an empirical question, and Figure 2 shows that it does.

2.3 THE SIGNAL IS NOT GONE—IT IS IN THE GOAL

If the trajectory cannot reveal an omission locally, what can? The task goal. Write it as $G = (q, a^\star)$, where q is the question and $a^\star$ is its correct answer. The goal implies a set of necessary conditions $\{C_1, \ldots, C_n\}$ that can be listed without looking at the trajectory at all, so a condition is named even when no step ever addressed it, exactly the case for an omission. The goal alone is not enough, since many executions satisfy the same goal. What identifies the root cause is the *mismatch* between what G requires and what the trajectory τ did: the first condition left unmet. Because G does not depend on the agents' actions, this mismatch survives the local silence of Eq. (1). Conditioning on the goal therefore adds information about $R^\star$ that the trajectory alone does not localize:

$$I(R^\star; G \mid \tau) \;>\; 0, \tag{3}$$

where $I(R^\star; G \mid \tau)$ is the mutual information between the root-cause step and the goal, given the trajectory. It measures the extra localizing power that the goal adds on top of τ.

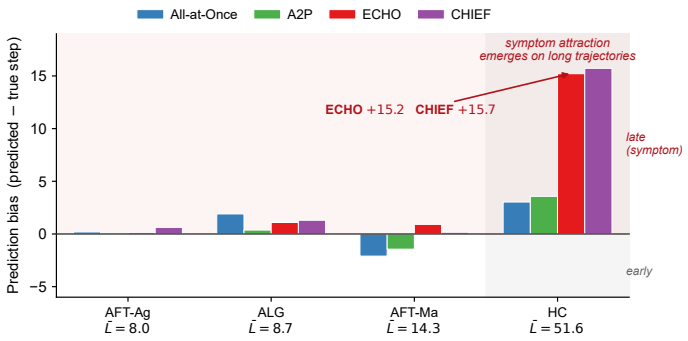


Figure 2: Prediction bias (predicted − true step) across four datasets ordered by mean trajectory length. Only the four baselines with meaningful bias are shown; Step-by-Step and Binary Search fix the step by search order, not evidence scoring (see text).

This is directly measurable. Eq. (3) makes a falsifiable prediction: adding the goal should help most where the trajectory is silent, that is, on omissions. We test this with a probe that predicts the root-cause step under two matched conditions, scored against the human labels: (A) the trajectory only, (B) the trajectory plus the goal G. Both share the same trajectory and target step, so their difference isolates the conditional term in Eq. (3). On the Hand-Crafted omissions, adding the goal doubles root-cause accuracy, from 5.6% to 11.1% (one-sided bootstrap p=0.055), whereas on commissions, where the trajectory already carries local signal, it does not help (18.8% to 12.5%). The gain therefore appears exactly where omissions silence the trajectory. The subset is small (n=42), so we read this as directional evidence rather than a precise effect size (Appendix C).

This is the opening that GCCA exploits, replacing the symptom-seeking rule $\arg\max_i P(\text{mistake} \mid \text{step}_i, \tau)$, whose evidence peaks late, with a goal-derived rule that peaks at the root cause (Section 3).

3 GCCA: Goal-Conditioned Causal Attribution

GCCA puts Section 2 into practice, turning the goal-side signal into an explicit structure that can be checked. We use "causal" in a diagnostic sense: we regress from the failure back to the unmet precondition. We do not use it in an interventional sense, since GCCA never re-executes the run and never perturbs it counterfactually. It recovers the cause implied by the goal without validating that cause by intervention (Section 5).

3.1 Overview and the necessary-precondition chain

GCCA takes two inputs: a failed trajectory $\tau = (\text{step}_1, \ldots, \text{step}_{|\tau|})$ and the task goal $G = (q, a^\star)$, where q is the question and $a^\star$ is its ground-truth answer when one is available. It returns the root-cause step $r^\star$ and the responsible agent $i^\star$. Section 2 described a symptom-seeking rule that ranks steps by local evidence. GCCA replaces it with a *goal-conditioned* rule,

$$r^\star = \min\{\, i : \text{step}_i \text{ is the first to violate some } C_j \in \text{NPC}(G) \,\}, \tag{4}$$

Here the *necessary-precondition chain* $\text{NPC}(G) = \{C_1, \ldots, C_n\}$, defined below, lists what the goal requires. Where a symptom-seeking rule maximizes evidence that peaks late, GCCA returns the *earliest* condition left unmet. For an omission no step performs the violation, so the earliest violating step is the first one that moves on as though C_j held, step 5 in Figure 1.

How the rule is applied depends on trajectory length. A short trajectory fits within a model's effective attention span, so the earliest violation can be read off in one pass. A long trajectory dilutes attention (Liu et al., 2024) and drifts late (Section 2), so GCCA narrows the search hierarchically first:

$$\text{route}(\tau) = \begin{cases} \text{direct}, & |\tau| \le k, \\ \text{hierarchical}, & |\tau| > k, \end{cases} \tag{5}$$

as detailed in Section 3.2 and depicted in Figure 3. Accuracy is robust to the threshold k over a broad range, and we defer its setting to Appendix H.

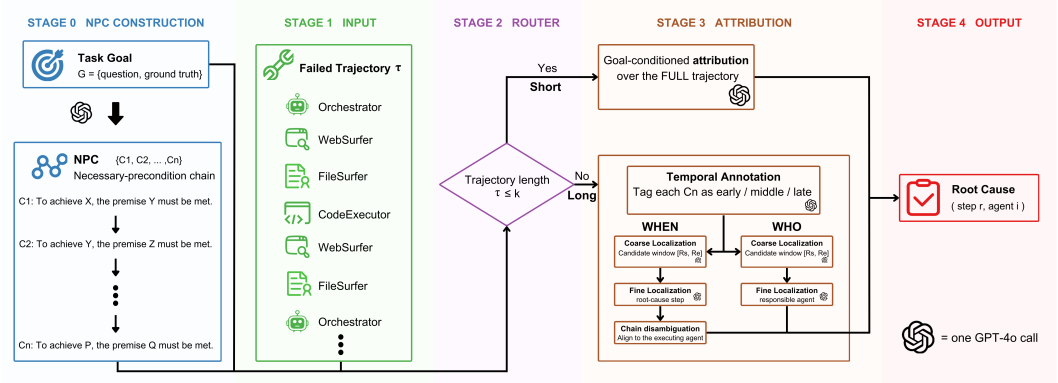


Figure 3: The GCCA framework: a necessary-precondition chain (NPC) reverse-decomposed from the goal and checked against the trajectory—a direct pass for short trajectories, a hierarchical search for long ones—blaming the first step to leave a condition unmet.

The necessary-precondition chain. We reverse-decompose the goal into an ordered set of conditions

$$\mathrm{NPC}(G) = \{C_1, C_2, \ldots, C_n\}, \tag{6}$$

Here C_1 restates the goal, each C_{j+1} must hold for C_j to be attainable, and C_n is the most basic prerequisite. This backward decomposition follows fault-tree analysis (Vesely et al., 1981) and goal regression (Waldinger, 1977). It names, in order, everything the trajectory *had to* do, independently of what it actually did. It is driven mainly by the question q; the answer $a^\star$, when available, only sharpens the conditions, so GCCA also runs without a reference answer (Section 4.3).

Reaching the goal requires every C_j, whether the corresponding action is taken or skipped. Checking the trajectory against $\mathrm{NPC}(G)$ therefore catches exactly the omissions that a step-by-step scan misses. A counterfactual test is anchored on the steps actually taken (e.g., A2P (West et al., 2025)), so an action that was never taken leaves it nothing to test; the top-down goal constraint flags such an action as a violated condition. The argument assumes that the generated chain really contains the responsible condition, and we verify that it does in 81–86% of failures (Appendix C). What the goal recovers is therefore the failed precondition itself; pinpointing its exact step is a separate and harder problem (Section 4.2).

Running example. Consider the failure of Figure 1, where the pages a student needs were announced only in a lecture recording. Reading the goal backward gives three conditions: C_1, report the recommended pages, which is the goal itself; C_2, transcribe the recording, since the pages exist only inside it; and C_3, open the audio file. The agent opens the file, satisfying C_3, but never transcribes it. C_2 is therefore the first unmet condition, and GCCA blames that step, early in the trajectory and long before the failure becomes visible. Constructing the chain costs a single LLM call. Further chains appear verbatim in Appendix C.

3.2 Attribution: Direct and Hierarchical Paths

Both paths apply Eq. (4), differing only in how much of the trajectory a single call must attend to at once. The full prompt templates are given in Appendix B.

Direct path ($|\tau| \le k$). The whole trajectory fits in one context window, so attribution takes two calls: one builds the NPC, and one makes a single goal-conditioned pass over τ and $\mathrm{NPC}(G)$, returning $r^\star$ and $i^\star$ together. This path covers the short-trajectory benchmark entirely.

Hierarchical path ($|\tau| > k$). This path narrows the search in three calls. (i) *Temporal-annotated NPC:* the construction call also tags each C_j as early, middle, or late. (ii) *Coarse localization:* from a compact per-step summary, the model selects the violated condition and a narrow window $[R_s, R_e]$. (iii) *Fine localization:* within that window, the model identifies the first step that fails the condition. Narrowing the window is what holds agent identification steady over a long context; the step itself is carried by the chain (Section 4.3).

Table 2: Main results on Who&When, GPT-4o, 3-run mean±std (%). Tokens/sample in thousands (k), sample-weighted over both subsets. Best per column in bold among zero-shot methods. †Supervised, hence not directly comparable; its released training set shares 11.9%/3.4% of Algorithm-Generated/Hand-Crafted test questions verbatim.

Type	Method	kTok/samp↓	Hand-Crafted Dataset		Algorithm-Generated Dataset	
			Agent-level↑	Step-level↑	Agent-level↑	Step-level↑
Holistic	All-at-Once	7.8	58.0±1.6	6.9±1.4	**57.1±1.3**	14.8±1.3
	A2P	8.1	59.8±0.8	17.8±3.3	56.1±1.0	33.6±1.9
Iterative	Step-by-Step	13.8	64.4±0.8	17.2±2.4	30.7±0.7	21.2±1.3
	Binary Search	16.5	23.0±1.6	6.9±0.0	50.3±4.7	20.6±3.6
	ECHO	14.0	47.1±2.2	15.5±0.0	45.2±2.3	22.8±1.3
Specialized	StepFinder†	—	—	23.0	—	29.6
	CHIEF	67.1	55.2±7.8	8.0±2.2	50.3±3.8	23.0±2.6
	GCCA (ours)	**4.7**	**70.1±1.6**	**23.0±3.3**	51.1±3.0	**33.9±2.6**

Who and When. The agent and the exact step call for different granularities, so the two come from separate runs of the hierarchical path. The benchmark's annotation guideline requires the decisive step to be one taken by the responsible agent (Zhang et al., 2025b), so a final alignment step reconciles the two: where localization returns the orchestrator's dispatching step, the prediction moves to the step that carries the action out. The alignment is deterministic and uses no ground truth.

Cost. GCCA queries the goal sparsely: two calls per sample on the direct path and six on the hierarchical one, averaging 3.1 over the benchmark.

4 Experiments

4.1 Experimental setup

Benchmark and metrics. We evaluate on the Who&When benchmark (Zhang et al., 2025b), which pairs each failed trajectory with a ground-truth responsible agent and root-cause step. Its two subsets stress attribution differently. Hand-Crafted contains long manual failures: 58 trajectories, mean length 51.6, and 72% omissions by two-rater human annotation (Appendix G). Algorithm-Generated contains short automatic ones: 126 trajectories, mean length 8.7, and 4% omissions. We report agent-level (Who) and step-level (When) accuracy; step-level accuracy on Hand-Crafted is the hardest and most discriminative metric.

Backbone and baselines. Unless noted, all methods share **GPT-4o** (OpenAI et al., 2024), run zero-shot, and report mean±std over three runs, with GCCA at temperature 0. Following the Who&When protocol, every method receives the same inputs: the question, the trajectory, and the ground-truth answer $a^\star$, so GCCA holds no input advantage; withholding $a^\star$ is ablated separately (Section 4.3). We fix a widely accessible, non-reasoning backbone so that the method is not confounded with an expensive reasoning model, and Section 4.3 confirms the results are not tied to GPT-4o. We compare against six attributors in three families (Table 2): *holistic* All-at-Once (Zhang et al., 2025b) and A2P (West et al., 2025); *iterative* Step-by-Step, Binary Search (Zhang et al., 2025b), and ECHO (Banerjee et al., 2025); and *specialized* StepFinder (Zhu et al., 2026b) and CHIEF (Wang et al., 2026). We also report per-sample token cost. StepFinder is a supervised model trained on Who&When-derived data, so we quote its published numbers and keep it outside the zero-shot comparison.

4.2 Main results

The shape of Table 2 matches what the mechanism predicts. GCCA's advantage concentrates on the long Hand-Crafted set, where omissions silence the trajectory and every baseline is weakest.

Table 3: Localization quality and ablations on Who&When (GPT-4o, 3-run mean, %). HC: Hand-Crafted; ALG: Algorithm-Generated; NPC: necessary-precondition chain; Hier.: hierarchical path; Disambig.: step alignment; GT: ground-truth answer. In (a), best per column in bold; in (b), the full model in bold.

(a) Step tolerance and signed bias on HC (steps).

Method	Step tolerance↑					signed bias	
	Exact	±1	±2	±3	±5	ALG	HC
A2P	17.8	27.6	**33.9**	35.6	47.1	**+0.37**	+3.57
ECHO	15.5	24.1	27.6	32.2	47.7	+1.09	+15.21
CHIEF	8.0	17.2	27.0	34.5	42.5	+1.30	+15.72
GCCA	**23.0**	**28.7**	31.6	**36.2**	**56.3**	+0.65	**+1.03**

(b) Ablations on GCCA.

	HC		ALG	
Setting	Agent	Step	Agent	Step
GCCA (full)	**70.1**	**23.0**	**51.1**	**33.9**
w/o NPC	54.6	14.4	54.8	32.5
w/o Hier.	63.8	22.4	50.5	30.7
w/o Disambig.	69.5	18.4	53.2	33.6
w/o GT	62.6	20.7	51.3	31.7

There it posts the best score on both dimensions, 70.1 against 64.4 on Who and 23.0 against 17.8 on When. On the short Algorithm-Generated set the trajectory still carries local signal, and GCCA is competitive rather than dominant: it ties A2P on the step (33.9 against 33.6) and scores 51.1 against All-at-Once's 57.1 on the agent. There the responsible agent is a single verification-style role that single-pass baselines name by default, whereas GCCA follows the chain to whichever agent broke a condition.

Coarse localization is accurate; exact pinpointing is hard for everyone. The 23.0% exact-step figure understates what GCCA recovers. Its chain *contains* the true root-cause condition in 81–86% of failures (Appendix C), and it lands within ±5 steps far more often than any baseline (Table 3a), so it reliably identifies *which* precondition failed. What remains open is the exact step on long trajectories, unsolved field-wide: no zero-shot baseline exceeds 17.8 on Hand-Crafted. GCCA leads there, and it carries the lowest positive bias of any method (Table 3a).

The advantage is concentrated on omissions. Two annotators independently label each of the 58 Hand-Crafted failures as an omission or a commission, giving 42 against 16 with κ=0.69 (Appendix G). On the omissions, the thorough trajectory readers are pulled onto the downstream symptom. ECHO and CHIEF post signed biases of +18.9 and +22.1 steps, and where the symptom lies strictly after the root cause, they land nearer the symptom than the cause 74% and 75% of the time. GCCA behaves differently. It is the most accurate on omissions, at 22.2% exact, and stays near the cause with a bias of +5.8.

Cost. The token column places GCCA on the accurate-and-cheap frontier. At 4.7k tokens and 3.1 calls per sample it is the cheapest method in the table, a third of ECHO's 14.0k and an order of magnitude below CHIEF's 67.1k. A goal-conditioned query sends a compact chain and summary rather than the full trajectory (Appendix D).

Tolerance. Exact-step accuracy is demanding for every method, so Table 3a also reports tolerant matches, which count a prediction as correct if it falls within a few steps. On Hand-Crafted, GCCA has the best exact accuracy and the largest margin at wide tolerances, reaching 56.3 against 47.1 at ±5; at ±2 A2P is slightly ahead. It lands in the root-cause neighborhood even when it misses the exact step.

Bias closes the loop. The rightmost columns of Table 3a report signed prediction bias. Section 2 used this number to expose the problem; here the same number shows that GCCA fixes it. GCCA's Hand-Crafted bias is only +1.03, below even the mildest baseline, A2P at +3.57, and an order of magnitude under CHIEF's +15.72 and ECHO's +15.21 (significant; Appendix G). The aggregate mixes error types: +5.8 on omissions and −11.4 on commissions. On the omission subset that matters here, +5.8 is the smallest positive bias of any trajectory reader (Table 7). CHIEF lands late two-thirds of the time, whereas GCCA's misses fall about as often early as late, with a median of 0. This is genuine localization rather than mere balance: GCCA has both the best wide-tolerance accuracy and the lowest bias, so it lands nearer the root cause than any baseline.

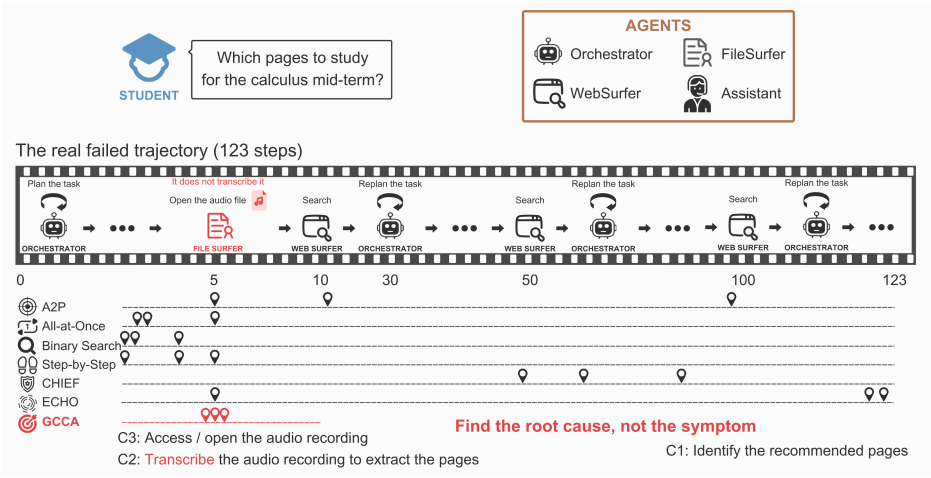


Figure 4: Case study on a real Who&When failure (calculus audio task, 123 steps): each method's three runs on the step axis. GCCA localizes the step-5 omission on all three runs; no baseline does so more than once.

4.3 Ablations and robustness

Component and input ablations. Table 3b removes one piece at a time. The necessary-precondition chain matters most: attributing directly from the trajectory drops accuracy from $70.1/23.0$ to $54.6/14.4$, the largest fall on both metrics, while barely touching the short set. The other components act on different dimensions. Forcing the single-pass path costs mostly agent accuracy (70.1 to 63.8); removing the step alignment costs mostly step accuracy (18.4). Withholding the ground-truth answer lowers accuracy only slightly, to $62.6/20.7$, and even then GCCA's step accuracy tops every zero-shot baseline that *is* given the answer, all at or below 17.8: the signal lives in the goal *structure* rather than in the answer key.

Cross-backbone robustness. We run GCCA unchanged on five backbones from four vendors (OpenAI et al., 2024; OpenAI, 2026; Yang et al., 2025; Anthropic, 2025; DeepSeek-AI et al., 2026). Hand-Crafted step-level accuracy stays within a narrow band of 19.5 to 25.3 with no collapse (Appendix E), so goal-conditioned attribution does not depend on one expensive, strong-reasoning model. It also transfers to an *independent second source*: on the Agentic subset of AFTraj (Zhang et al., 2026), GCCA is best at naming the agent (81.4), though on those 8-step trajectories its step-level lead disappears (Appendix F).

4.4 Case study

Figure 4 returns to the step-5 audio-transcription omission of Section 3.1, showing every method's three runs. GCCA returns step 5 all three times: its chain finds that obtaining the contents of the recording is the first unmet condition. The baselines scatter from step 0 to step 97; four of them touch step 5 on one run out of three, none on more, and CHIEF and ECHO never reach it. The figure is the paper in miniature: baselines drawn to the late symptom, GCCA following the goal back to the root cause.

5 Related Work

Reading the signal off the trajectory. We review prior work along the axis that organizes ours: *where the attribution signal comes from*. The Who&When benchmark (Zhang et al., 2025b) introduced the task over trajectories such as Magentic-One (Fourney et al., 2024) on GAIA (Mialon et al., 2023), reporting that pinpointing the step is far harder than naming the agent. Subsequent methods differ mainly in *how* they extract evidence. *Holistic* methods judge in one pass, including the abduct–act–predict simulation of A2P (West et al., 2025). *Iterative* methods search over many passes, from step-by-step and binary search to the consensus panel of ECHO (Banerjee et al., 2025).

Specialized methods add structure or training: CHIEF's causal graph (Wang et al., 2026), the supervised StepFinder (Zhu et al., 2026b), and trained, replay, schema, and check-based variants (Zhang et al., 2025a; Zhu et al., 2026a; Ge et al., 2025; Yu et al., 2026; Xia et al., 2026; Feng et al., 2026; Wu et al., 2026). Complementary lines catalogue failure modes (Cemri et al., 2025; Ma et al., 2026) and enlarge evaluation with trace-level benchmarks (Fan et al., 2026; Zhang et al., 2026; Deshpande et al., 2025), arguing that output-only logs leave some failures unattributable (Chen et al., 2026). These differ in how they search, but each ranks the responsible step by evidence *inside the trajectory*. That includes CHIEF, which derives conditions from the task yet still scores steps by their trace. Such evidence is what an omission fails to leave.

Deriving the signal from goals and causal structure. When the trajectory is uninformative, the classical alternative is to reason from what the task *required*. GCCA's chain is a form of goal regression, which posts the preconditions of goal-achieving actions as subgoals (Waldinger, 1977; Russell & Norvig, 2021), the backward decomposition that also underlies fault-tree analysis (Vesely et al., 1981). A related tradition attributes failures through explicit causal structure (Baah et al., 2010; Xin et al., 2023; Chraim et al., 2026; Pearl, 2009; Shapley, 1953), but assumes either an observable signature or a re-executable system, and a recorded omission offers neither. GCCA keeps the goal-regression backbone but drops the intervention requirement, so a silent omission still breaks a named condition.

What the method rests on. Two findings make our approach necessary and possible. Models attend unevenly to long inputs: lost-in-the-middle (Liu et al., 2024), positional bias (Zhang et al., 2024), degradation as context grows (Hong et al., 2025; Hsieh et al., 2024). A single pass therefore drifts to salient late steps, which is why GCCA narrows the search hierarchically. And LLMs cannot reliably self-correct without an external signal (Huang et al., 2024; Shinn et al., 2023; Lightman et al., 2023; Wang et al., 2023; Zheng et al., 2023); GCCA supplies one, the goal, and queries it sparsely.

6 Conclusion and Limitations

We identified *symptom attraction*, the tendency of attribution to land on the late symptom rather than the early omission where a failure began. We traced it, in information-theoretic terms, to two facts: the trajectory is silent about omissions, and the lost signal survives in the goal. GCCA turns this into a method. It reads the goal backward as a necessary-precondition chain and blames the first step that leaves one unmet. GCCA reduces the Hand-Crafted late bias to a near-zero $+1.03$, so the bias that reveals the problem is the bias the method removes. This closes the loop from phenomenon to mechanism to method. The gains land where the mechanism predicts: on long, omission-heavy trajectories GCCA is best on both Who and When, zero-shot, at the lowest token cost, and stably across five backbones.

Limitations. Five limits temper these results. First, on the short Algorithm-Generated set GCCA is not first on agent-level accuracy, because there the responsible agent is a single verification-style role that is nameable from coarse cues; its strength is step localization on long trajectories. Second, exact-step accuracy on the longest trajectories stays low in absolute terms, a difficulty shared across methods, and chain faithfulness of 81–86% (Appendix C) indicates that what remains open is exact-step disambiguation rather than chain quality. Third, the method rests on the quality of the precondition chain, and how that quality varies across backbones is untested. Fourth, the attribution is diagnostic rather than interventional, and separately the chain is a goal-regression heuristic that can include non-necessary conditions (Appendix C) rather than a verified minimal set. Fifth, the advantage is specific to omissions: on commissions GCCA reaches 25.0% against A2P's 37.5% (Appendix G), which is what the mechanism predicts, since a visibly wrong action is exactly what reading the trajectory already catches.

Outlook. The goal-conditioned view suggests three directions: extending necessary-precondition reasoning beyond omission-dominated failures, learning the chain instead of generating it zero-shot, and using it to drive an agent's self-correction online.

REPRODUCIBILITY STATEMENT

All experiments use the public Who&When benchmark (Zhang et al., 2025b) and the public AFTraj benchmark (Zhang et al., 2026); no data are collected from human subjects. GCCA is entirely zero-shot: the full set of prompt templates for every stage (chain construction, coarse and fine localization) is given verbatim in Appendix B, the routing rule and threshold in Section 3.2 and Appendix H, and worked necessary-precondition chains in Appendix C. All main numbers are 3-run means with the backbone, decoding settings, and call/token budgets stated in Section 4.1 and Appendix D. The human error-type annotation protocol, inter-annotator agreement, and adjudication are described in Appendix G. Source code and the annotation labels are provided as anonymized supplementary material.

AI USE STATEMENT

We used generative AI tools only as general-purpose writing and coding assistants: to polish prose, and to help draft and debug analysis and plotting scripts. Generative AI was not used to generate research ideas, experimental results, or claims. All AI-assisted code was verified and re-run by the authors, and all reported numbers come from our own executions of the released code. We have reviewed all AI-assisted text and take full responsibility for the final content of this work.

REFERENCES

Anthropic. System card: Claude sonnet 4.5. https://www.anthropic.com/claude-sonnet-4-5-system-card, September 2025.

George K. Baah, Andy Podgurski, and Mary Jean Harrold. Causal inference for statistical fault localization. In *Proceedings of the 19th International Symposium on Software Testing and Analysis (ISSTA '10)*, pp. 73–84. Association for Computing Machinery, 2010. ISBN 9781605588230. doi: 10.1145/1831708.1831717. URL https://doi.org/10.1145/1831708.1831717.

Adi Banerjee, Anirudh Nair, and Tarik Borogovac. Where did it all go wrong? a hierarchical look into multi-agent error attribution, 2025. URL https://arxiv.org/abs/2510.04886.

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?, 2025. URL https://arxiv.org/abs/2503.13657.

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, 2026. URL https://arxiv.org/abs/2604.22708.

Fabien Chraim, Jian Zhang, Dominik Janzing, Xiang Song, Christos Faloutsos, and John Evans. Netcause: Counterfactual learning for root cause analysis in large-scale networks, 2026. URL https://arxiv.org/abs/2606.13543.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, et al. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. Trail: Trace reasoning and agentic issue localization, 2025. URL https://arxiv.org/abs/2505.08638.

Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, Xin Cong, and Yankai Lin. Agentprocessbench: Diagnosing step-level process quality in tool-using agents, 2026. URL https://arxiv.org/abs/2603.14465.

Naihe Feng, Yi Sui, Shiyi Hou, Ga Wu, and Jesse C. Cresswell. Conformal agent error attribution, 2026. URL https://arxiv.org/abs/2605.06788.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. Magentic-one: A generalist multi-agent system for solving complex tasks, 2024. URL https://arxiv.org/abs/2411.04468.

Yu Ge, Linna Xie, Zhong Li, Yu Pei, and Tian Zhang. Who is introducing the failure? automatically attributing failures of multi-agent systems via spectrum analysis, 2025. URL https://arxiv.org/abs/2509.13782.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges, 2024. URL https://arxiv.org/abs/2402.01680.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma, July 2025. URL https://research.trychroma.com/context-rot.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2024. URL https://arxiv.org/abs/2310.01798.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics (TACL)*, 2024. URL https://arxiv.org/abs/2307.03172.

Xuyan Ma, Xiaofei Xie, Yawen Wang, Junjie Wang, Boyu Wu, Mingyang Li, and Qing Wang. Demystifying the lifecycle of failures in platform-orchestrated agentic workflows, 2026. URL https://arxiv.org/abs/2509.23735.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

OpenAI. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, UK, 2nd edition, 2009. ISBN 9780521895606.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2021.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, Princeton, NJ, 1953.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. Fault tree handbook (nureg-0492). Technical Report NUREG-0492, U.S. Nuclear Regulatory Commission, 1981.

Richard Waldinger. Achieving several goals simultaneously. *Machine Intelligence*, 8, 1977.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

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.

Alva West, Yixuan Weng, Minjun Zhu, Zhen Lin, Zhiyuan Ning, and Yue Zhang. Abduct, act, predict: Scaffolding causal inference for automated failure attribution in multi-agent systems, 2025. URL https://arxiv.org/abs/2509.10401.

Hua Wu, Wanhao Zheng, Minghao Zhang, Xiaojing Bai, Mengyang Pu, Li Sun, Junchen Liu, and Yuhan Xu. Intention-behavior consistency-based automated failure attribution for llm-driven multi-agent systems. *Expert Systems with Applications*, 331(Part A):133044, 2026. ISSN 0957-4174. doi: 10.1016/j.eswa.2026.133044. URL https://www.sciencedirect.com/science/article/pii/S095741742601955X.

Yufei Xia, Anjun Gao, Yueyang Quan, Zhuqing Liu, and Minghong Fang. Who broke the system? failure localization in llm-based multi-agent systems, 2026. URL https://arxiv.org/abs/2607.07989.

Ruyue Xin, Peng Chen, and Zhiming Zhao. Causalrca: Causal inference based precise fine-grained root cause localization for microservice applications. *Journal of Systems and Software*, 203: 111724, 2023. ISSN 0164-1212. doi: 10.1016/j.jss.2023.111724.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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, 2026. URL https://arxiv.org/abs/2509.24088.

Boxuan Zhang, Jianing Zhu, Zeru Shi, Dongfang Liu, and Ruixiang Tang. Agentforesight: Online auditing for early failure prediction in multi-agent systems, 2026. URL https://arxiv.org/abs/2605.08715.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems?, 2025a. URL https://arxiv.org/abs/2509.03312.

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, 2025b. URL https://arxiv.org/abs/2505.00212.

Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu, and Zhangyang Wang. Found in the middle: How language models use long contexts better via plug-and-play positional encoding, 2024. URL https://arxiv.org/abs/2403.04797.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Chenyang Zhu, Spencer Hong, Jingyu Wu, Kushal Chawla, Charlotte Tang, Youbing Yin, Nathan Wolfe, Erin Babinsky, and Daben Liu. Raffles: Reasoning-based attribution of faults for llm systems, 2026a. URL https://arxiv.org/abs/2509.06822.

Taiyu Zhu, Yifan Wu, Weilin Jin, Ying Li, and Gang Huang. Stepfinder: A temporal semantic framework for failure attribution in multi-agent systems, 2026b. URL https://arxiv.org/abs/2606.03467.

A A STYLIZED MODEL OF SYMPTOM ATTRACTION

This appendix states the stylized model of symptom attraction summarized in Section 2.2, expanding the evidence ordering of Eq. (2); the premises—the locally-silent root cause (Eq. 1) and the goal-side signal (Eq. 3)—are stated in the main text. We stress at the outset that assumptions (A1)–(A3) below are idealizations: what follows is a mechanism we then verify empirically, not a theorem holding verbatim for every attributor or every trajectory.

Proposition 1 (Symptom attraction under a stylized model) *Under* (A1) *a locally-silent root cause (Eq. 1),* (A2) *local evidence about $R^\star$ non-decreasing toward the symptom (Eq. 2), and* (A3) *an attributor that returns the step of maximal local evidence, $\hat{r} = \arg\max_i P(\textit{mistake} \mid \mathrm{step}_i, \tau)$, the estimate coincides with the symptom, $\hat{r}{=}r_m$, so the expected signed error equals the mean cause–symptom gap, $\mathbb{E}[\hat{r} - R^\star] = \mu_{\mathrm{prop}} \geq 0$ with $\mu_{\mathrm{prop}} = \mathbb{E}[r_m - R^\star]$. With only stochastically monotone evidence this weakens from an identity to a positive tendency.*

Propagation of the omission. After the omission at the true root cause $R^\star$, its consequences propagate,

$$R^\star \to r_1 \to \cdots \to r_m,$$

until the failure becomes visible at the *symptom* step r_m. The local evidence o_r that a step carries—an anomalous output, a stalled tool call, a contradiction—is absent at the normal-looking cause and accumulates as the missing result is repeatedly needed downstream, so the information each step's local observation reveals about $R^\star$ increases toward the symptom, giving the ordering of Eq. (2).

Direction of the bias. Under (A3) a local-evidence attributor returns $\hat{r} = \arg\max_i P(\text{mistake} \mid \mathrm{step}_i, \tau)$; by the monotone ordering of (A2) this argmax is the symptom, $\hat{r} = r_m$ in the noiseless limit. The expected signed error is then exactly the mean cause–symptom gap,

$$\mathbb{E}[\hat{r} - R^\star] \;=\; \mu_{\mathrm{prop}} \;=\; \mathbb{E}[r_m - R^\star] \;\geq\; 0,$$

non-negative because the symptom never precedes the cause. This is an identity of the idealized model, not a bound needing an unspecified constant. Two relaxations weaken it to a *tendency* without reversing its sign: if the evidence is only stochastically monotone (A2), the argmax lands near but not exactly at r_m; and if the trajectory is only partially silent (A1), part of the mass returns to $R^\star$. Both shrink the expected bias, consistent with the smaller but still positive biases on shorter sets.

Dependence on length. The model does not by itself *prove* that μ_{prop} grows with trajectory length: a long run only *admits* a larger cause–symptom gap, and a short run may hide an omission until the end. Whether the gap grows on average is an empirical question, which Figure 2 answers directly—the bias rises from near zero on the short sets to $+15.72$ on the long Hand-Crafted set.

B PROMPTS

GCCA is entirely zero-shot; every step is a single call with one of the templates below, where {`task`} is the question q, {`answer`} the ground-truth answer $a^\star$ (omitted when unavailable), and {`trajectory`} the formatted steps. The direct path ($|\tau| \leq k$) uses P1–P2; the hierarchical path ($|\tau| > k$) uses P3–P5, each run once for the responsible agent and once for the exact step.

Inside the prompt text below, extttCk names the last element of the chain; this k is the chain length, not the routing threshold k of Eq. (5).

(P1) Necessary-precondition chain, direct path.

```
Task: {task}
Correct answer: {answer}
List the NECESSARY PRECONDITIONS to reach this answer, from the final
goal (C1) back to the earliest prerequisite (Ck). Each condition must be
a specific, verifiable action or information-retrieval step. Include
OMISSION conditions: things the agent MUST do/read/access.
Reply as a numbered list: C1: <final goal> ... Ck: <earliest prerequisite>
```

(P2) Goal-conditioned attribution, direct path.

```
Task: {task}  Correct answer: {answer}
To reach the correct answer, the following conditions MUST be satisfied
in order: {conditions}
Trajectory: {trajectory}
The task FAILED. Based on the required conditions, find the ROOT CAUSE:
the EARLIEST step whose action PREVENTED one of the above conditions
from being satisfied. Think about which condition was NEVER satisfied
and which step first caused that.
Reply EXACTLY: FAILED_CONDITION: <...> AGENT: <name> STEP: <int> REASON: <...>
```

(P3) Temporal-annotated chain, hierarchical path.

```
Task: {task}  Correct answer: {answer}
List the NECESSARY PRECONDITIONS from final goal (C1) back to earliest
(Ck). For each condition, estimate WHEN: EARLY, MIDDLE, or LATE.
Reply as: C1: <condition> | WHEN: LATE ... Ck: <condition> | WHEN: EARLY
```

(P4) Coarse localization.

```
Task: {task}  Correct answer: {answer}
Required conditions (with expected timing): {conditions}
Trajectory summary (one line per step): {summary}
The task FAILED. Which condition was first violated? What step range
should we investigate?
Reply EXACTLY: FAILED_CONDITION_TEXT: <...> RANGE_START: <int> RANGE_END: <int>
```

Both {summary} and {window} are rendered programmatically, with no extra model call, so the call and token counts in Table 4 cover the whole pipeline. Each renders one line per step as `Step i [agent]: content`, with newlines stripped and the content truncated: to 60 characters for {summary}, which spans the full trajectory, and to 500 for {window}, which covers only the steps inside the coarse range.

(P5) Fine localization.

```
Task: {task}  Correct answer: {answer}
Violated condition: {condition}
Steps {start}--{end}: {window}
Find the FIRST EXECUTION step (not orchestration) that failed to satisfy
the condition.
Reply EXACTLY: STEP: <int> AGENT: <name> REASON: <one sentence>
```

(P5′) Fine localization, agent pass. The two hierarchical runs are not the same call issued twice. P3–P5 above produce the step; the agent comes from a second run whose fine-localization prompt is worded for agent responsibility rather than step boundaries, shown below. Only its `AGENT` field is kept, and the alignment of Section 3.2 is not applied to it, which is why the alignment moves the step prediction without touching the agent prediction.

```
Task: {task}  Correct answer: {answer}
Violated condition: {condition}
Relevant steps (Steps {start} to {end}): {window}
Find the FIRST step in this range where the agent's action failed to
satisfy the violated condition (wrong action, incomplete execution,
missing necessary step). The step must be where the agent PRODUCED
OUTPUT (not an orchestration/instruction step).
Reply EXACTLY: STEP: <int> AGENT: <agent name> REASON: <one sentence>
```

C EXAMPLE NECESSARY-PRECONDITION CHAINS

To show that the necessary-precondition chains GCCA elicits are concrete and not cherry-picked, we list the chains it constructs for two Hand-Crafted failures, verbatim from the model's output. In each, the goal is decomposed backward into conditions C_1 (the goal) down to the earliest prerequisite, each tagged with a coarse position; GCCA then reports the first condition the trajectory leaves unmet and localizes the step that violates it.

Example 1 (the running example, id 51; 123 steps). A student asks which pages to study; the pages were announced only in an audio recording.

> C_1 [late]: Identify the recommended reading for the Calculus mid-term.
> C_2 [middle]: Listen to / transcribe the audio recording from the professor.
> C_3 [early]: Obtain a working way to access the audio.

GCCA flags C_2 as the first unmet condition and localizes step 5 (FileSurfer opens the audio but never transcribes it)—the true root cause.

Example 2 (id 1; 29 steps). A user asks where to take martial-arts classes within a five-minute walk of the NYSE, open 7–9 pm.

> C_1 [late]: Attend classes at a school meeting all constraints.
> C_2 [middle]: Confirm the class schedule and availability for evening (7–9 pm) sessions.
> C_3 [early]: Verify the school is within a five-minute walk of the NYSE.
> C_4–C_5 [early]: Ensure attire/equipment and a compatible work schedule.

GCCA flags C_2 (schedule never confirmed) and localizes step 12 (WebSurfer), matching the ground truth. The chains are specific and checkable: each condition names a concrete action or fact, so a skipped one surfaces as a violated condition rather than as a diffuse anomaly.

This example also exposes a failure mode of the elicitation: C_4–C_5 (attire/equipment and a compatible work schedule) are *not* strictly necessary to the stated goal—the model sometimes appends plausible but non-essential preconditions. Over-generation of this kind does not derail attribution here, because Eq. (4) minimizes over the *step* index rather than scanning the condition list in order: a spurious condition changes the answer only if the model both judges it violated and places that violation earlier than the genuine one, which does not happen here; but it means $\mathrm{NPC}(G)$ is a goal-regression heuristic, not a verified minimal precondition set (Limitations, Section 6).

Chain faithfulness. GCCA rests on a premise worth testing directly: that the elicited chain *contains* the condition whose violation is the true root cause—not that the chain is complete or minimal, only that the responsible condition is present for the attributor to find. We measure this on both subsets with an LLM judge (GPT-4o) that, given the human-annotated root-cause step and the elicited $\mathrm{NPC}(G)$, decides whether any listed condition corresponds to that step. The elicited chain contains the root-cause condition in 85.7% of Algorithm-Generated failures and 81.4% of Hand-Crafted ones (the latter among chains that parse; a further 16% of long chains degraded to a placeholder through a formatting-parse fault we have since fixed, so the unconditional Hand-Crafted rate is about 68%; the reported accuracies come from those same runs and are unaffected). The premise is thus not a blanket assumption but an empirically mostly-satisfied condition. Two points follow. First, chain faithfulness (81–86%) sits well above GCCA's exact-step accuracy (23% on Hand-Crafted): the responsible condition is usually present, so the residual error is dominated by *localizing* it to the exact step, not by *eliciting* it—consistent with the mechanism, which places the difficulty in mapping a recovered goal-signal onto a precise step. Second, this measurement is an automatic (LLM-judge) estimate rather than a human one, and reproduces within a few points across runs; we report the more conservative figures.

The goal adds information exactly where the trajectory is silent (probe for Eq. 3). Appendix A argues the recoverable signal lives in the goal; here we test the conditional-information claim of Eq. (3) directly. Using the same GPT-4o judge and the human root-cause labels, we ask it to name the root-cause step under two matched conditions: **(A)** given the trajectory alone, and **(B)** given the trajectory *plus* the goal $G = (q, a^\star)$. The two conditions share the same trajectory and the

same target step, so the accuracy difference $B - A$ isolates the extra localizing power the goal contributes—precisely the conditional term $I(R^\star; G \mid \tau)$. Seeing $a^\star$ is of course an advantage in itself, but it is granted identically on omissions and on commissions, so it cannot explain a gain that appears on omissions alone. On Hand-Crafted *omissions*, where the trajectory is locally silent (Eq. 1), adding the goal raises root-cause accuracy from 5.6% to 11.1%—a doubling ($+5.6$ points; paired bootstrap over items, $20{,}000$ resamples, 95% CI $[-0.8, +13.5]$, one-sided $p{=}0.055$; a more conservative majority-vote-over-runs aggregation gives $+7.1$, $p{=}0.046$). On *commissions*, where the trajectory already carries local evidence, the goal adds nothing ($18.8\% \to 12.5\%$, $n{=}16$). The direction and the localization of the effect match Eq. (3)—the goal's contribution is concentrated on exactly the omissions that defeat trajectory-only reading—while the magnitude sits at the edge of significance on this small subset (42 omissions, 16 commissions) with an LLM-judge estimate; we therefore read it as directional evidence for the mechanism, not a precise effect size.

D COST BREAKDOWN

Table 4 reports the per-method cost behind the cost column (kTok/samp) of Table 2, split by subset and broken into LLM calls and total tokens per sample (GPT-4o, 3-run mean; call counts are deterministic given the routing, so they do not vary across runs). Two points stand out. First, GCCA's cost scales with the routing: on the short Algorithm-Generated set every trajectory takes the direct path (2 calls), while on the long Hand-Crafted set most take the hierarchical path, averaging 5.5; the sample-weighted mean is 3.1. Second, and more strikingly, GCCA uses the *fewest tokens per sample of any method*—$4{,}686$ weighted, a third of ECHO's $14{,}032$, an order of magnitude below CHIEF's $67{,}116$, and below even the single-call holistic baselines—because a goal-conditioned query sends the model a compact chain and a summary rather than the full trajectory repeatedly. Cheap in calls and cheaper still in tokens, GCCA sits alone on the accurate-and-cheap frontier.

Table 4: Per-sample cost by subset (GPT-4o, 3-run mean). ALG: Algorithm-Generated (126 samples); HC: Hand-Crafted (58); Wtd: sample-weighted mean. Lowest weighted cost in each block in bold.

Method	LLM calls / sample↓			Tokens / sample↓		
	ALG	HC	Wtd	ALG	HC	Wtd
All-at-Once	1.0	1.0	**1.0**	3,490	17,176	7,804
A2P	1.0	1.0	**1.0**	3,726	17,659	8,118
Step-by-Step	2.7	10.4	5.1	3,687	35,810	13,813
Binary Search	3.5	5.6	4.2	7,210	36,596	16,473
ECHO	6.0	6.0	6.0	7,493	28,237	14,032
CHIEF	6.0	6.0	6.0	37,064	132,399	67,116
GCCA (ours)	2.0	5.5	3.1	**1,501**	**11,606**	**4,686**

E CROSS-BACKBONE RESULTS

Table 5 reports GCCA run unchanged across five backbones from four vendors, supporting the robustness claim of Section 4.3.

F GENERALIZATION TO A SECOND BENCHMARK SOURCE

The main evaluation uses Who&When, the standard failure-attribution benchmark and the only one that pairs *long* trajectories with root-cause labels. To test whether GCCA transfers to an independent source, we evaluate it on AFTraj-Agentic (Zhang et al., 2026) (470 multi-agent failures). Among the available extended benchmarks this is the one closest in character to Hand-Crafted—multi-agent and omission-heavy (72% omissions)—differing chiefly in that its trajectories are *short* (mean length 8.0). We therefore read it as a short-trajectory analogue of our target regime rather than a second long-trajectory test, of which none currently exists.

Table 5: GCCA is largely insensitive to the choice of backbone, 3-run mean (%). Across five backbones from four vendors, accuracy varies little and Hand-Crafted step-level accuracy stays within 19.5–25.3 with no collapse; best per column in bold.

	Hand-Crafted Dataset		Algorithm-Generated Dataset	
Backbone	Agent-level↑	Step-level↑	Agent-level↑	Step-level↑
GPT-4o (main)	**70.1**	23.0	51.1	**33.9**
GPT-5.2	67.2	24.7	47.1	29.9
Qwen3-235B-a22b-instruct (thinking)	63.8	23.6	49.7	27.2
Claude-Sonnet-4.5	63.2	19.5	**51.3**	33.1
DeepSeek-V4-flash (thinking)	59.8	**25.3**	49.5	33.6

Table 6 reports every method at 3-run mean. Two effects from the main results carry over. First, part of the omission-driven weakness reappears: All-at-Once and CHIEF, which also fail at Hand-Crafted step localization, again land at only 28.6 and 38.2% on When, far below the top group. Second, GCCA generalizes at the agent level: it is the *best* method at naming the agent (Who 81.4). On the step it is no longer first. Four methods sit within 1.5 points: Step-by-Step 53.3, ECHO 52.7, A2P 51.9, GCCA 51.8. The clear margin GCCA holds on the long Hand-Crafted set (23.0 vs. 17.8/17.2 over A2P/Step-by-Step) disappears here. This is the length dependence the mechanism predicts, and we state it as a limit rather than a win: on 8-step trajectories the whole run already fits the model's attention, so reading it directly works, and the hierarchical contraction GCCA adds for long runs buys nothing. ECHO makes the point most sharply. It is among the best here and reaches 74.6% on the 14-step AFTraj-Math set (Table 1), yet falls to 15.5% with a +15.2 late bias on the 52-step Hand-Crafted set. GCCA's step-level advantage is, by design, a long-trajectory effect.

Table 6: Generalization to AFTraj-Agentic (Zhang et al., 2026) (470 multi-agent failures, mean length 8.0; GPT-4o, 3-run mean, %). GCCA is best on Who; on When four methods sit within 1.5 points and GCCA is not first, as the length dependence of the mechanism predicts. All-at-Once and CHIEF, which collapse on Hand-Crafted, collapse here too. Best per column in bold.

Method	Agent-level (Who)↑	Step-level (When)↑
All-at-Once	79.6	28.6
A2P	78.2	51.9
Step-by-Step	72.1	**53.3**
Binary Search	20.7	16.0
ECHO	79.4	52.7
CHIEF	73.8	38.2
GCCA	**81.4**	51.8

G LOCALIZATION BY ERROR TYPE (HUMAN-ANNOTATED)

The two error-type claims—that omissions defeat trajectory readers, and that GCCA is strongest exactly there—rest on a human annotation of the 58 Hand-Crafted failures. Two annotators independently labelled each failure as an *omission* (a skipped-but-necessary action that looks normal in the step output) or a *commission* (a visibly wrong action), and marked the first step at which the failure becomes observable—the *symptom* step R_{sym}. On error type the annotators agree on 50 of 58 cases (raw agreement 86%; Cohen's κ=0.69, substantial). On the symptom step they agree exactly on 60% of cases and within ± 2 steps on 67%, with a mean absolute difference of 4.9 steps. The 24 cases that differ on either dimension—8 on error type, 16 on the symptom step (by more than 3 steps)—were adjudicated to a consensus; the remaining 34 agree on both. The consensus has 42 omissions and 16 commissions and places R_{sym} downstream of or at the root cause in *every* case (38 strictly after, 20 equal, none before; mean $R_{\text{sym}} - R^{\star} = +7.1$); moreover *neither* annotator, independently, ever placed the symptom before the root cause, so the premise of symptom attraction holds prior to any adjudication. These labels are used only to slice results and never enter GCCA or any baseline. An LLM auto-classifier (GPT-4o-mini), released for reproducibility, recovers a compatible omission-dominant split (67% omissions, vs. the human 72%).

Table 7 reports, per method on the human omission subset, exact step accuracy, accuracy within ± 5 steps of the root cause, and the signed prediction bias; the final column gives commission accuracy for contrast. Three patterns emerge. **(i) Most reading-based baselines fall off on omissions:** their exact accuracy drops sharply from commissions to omissions (A2P $37.5 \rightarrow 10.3$, ECHO $25.0 \rightarrow 11.9$, Binary Search $20.8 \rightarrow 1.6$), the error type on which the trajectory is silent. Step-by-Step is the exception: its forward scan halts early rather than drifting late, so it rises on omissions ($10.4 \rightarrow 19.8$) and carries a negative bias (-2.0). The pattern is therefore specific to readers that weigh the whole trajectory, not to trajectory reading as such. **(ii) The thorough readers drift onto the symptom:** ECHO and CHIEF reach signed biases of $+18.9$ and $+22.1$ steps, and on the 38 gap cases (symptom strictly after the root cause) their predictions land nearer the symptom than the root cause 74% and 75% of the time—a direct measurement of symptom attraction, isolated to the methods that read the whole trajectory. **(iii) GCCA does not:** it is the most accurate on omissions (22.2% exact) and localizes them to the root-cause neighborhood far better than any baseline (61.1% within ± 5 vs. at most 52.4%), staying near the cause (bias $+5.8$; nearer the root than the symptom on 62% of gap cases). This is the empirical signature of drawing the attribution signal from the goal rather than the trajectory.

Table 7: Hand-Crafted localization by *human-annotated* error type (GPT-4o, 3-run mean, %). Omission n=42, commission n=16; inter-annotator κ=0.69. Best in each omission column in bold.

	Omission			Commission
Method	exact↑	± 5 ↑	bias	exact↑
All-at-Once	5.6	39.7	$+6.5$	10.4
A2P	10.3	46.8	$+7.1$	37.5
Step-by-Step	19.8	52.4	-2.0	10.4
Binary Search	1.6	32.5	-3.7	20.8
ECHO	11.9	45.2	$+18.9$	25.0
CHIEF	6.3	38.9	$+22.1$	12.5
GCCA	**22.2**	**61.1**	$+5.8$	25.0

The mechanism effects are statistically significant. We test the two effects the mechanism predicts—the symptom-ward direction of the error, and its removal—with a paired bootstrap over the 58 items (20,000 resamples; 95% CIs). Because the root cause sits early (mean $R^\star/L$=0.43) and the symptom follows it, a *uniformly random* step is already more likely to fall nearer the symptom than the root; on the 38 gap cases the expected rates for a random predictor are 57.2% (nearer symptom) and 41.8% (nearer root), and its expected signed bias is $+10.1$ steps. We therefore test against this random-predictor null rather than 50%. The symptom *direction* is significant: ECHO and CHIEF land closer to the symptom on 74% and 75% of gap items (binomial p=0.027 and p=0.012 against 57.2%), whereas GCCA lands closer to the *root* on 62% (p=0.006 against 41.8%) and A2P is indistinguishable from random (50%, p=0.41). Relative to the random-predictor bias of $+10.1$, ECHO ($+15.21$) and CHIEF ($+15.72$) are pulled *later* than chance while GCCA ($+1.03$) is pulled earlier. The late-bias reduction is likewise significant against the symptom-attracted readers: GCCA's Hand-Crafted bias is 14.0 and 14.5 steps below ECHO's and CHIEF's (95% CI $[6.5, 22.7]$ and $[8.1, 21.4]$; Cohen's d=0.44, 0.57), and the same holds on the omission subset.

H SENSITIVITY TO THE ROUTING THRESHOLD k

GCCA has a single hyperparameter, the routing threshold k that separates the direct path ($|\tau| \leq k$) from the hierarchical one ($|\tau| > k$); the main experiments fix $k = 15$. Table 8 varies it over $\{10, 15, 20, 25, 30\}$ (3-run mean, GPT-4o). The choice is not delicate. Step-level accuracy is essentially flat throughout (Hand-Crafted 22.4–24.1, Algorithm-Generated 33.1–34.1), and agent-level accuracy holds a stable plateau for $k \leq 20$ before easing by a few points at $k = 25, 30$, as progressively longer trajectories are forced through the single-pass path and begin to drift toward the symptom—the same effect isolated by the "w/o hierarchical" ablation of Table 3b, which is the $k \to \infty$ limit. The Algorithm-Generated set is unaffected by design: its trajectories are at most 10 steps, so every $k \geq 10$ routes all of them through the direct path and the small variation seen there is run-to-run noise. The default $k = 15$ thus sits safely inside the stable region rather than at a cliff.

Table 8: Sensitivity to the routing threshold k, GPT-4o, 3-run mean (%). $k = 15$ is the default used throughout the paper; its row reproduces the main-table GCCA numbers.

	Hand-Crafted Dataset		Algorithm-Generated Dataset	
Threshold k	Agent-level↑	Step-level↑	Agent-level↑	Step-level↑
$k = 10$	**70.7**	**24.1**	**53.4**	33.1
$k = 15$ (default)	70.1	23.0	51.1	33.9
$k = 20$	69.0	23.0	50.0	33.1
$k = 25$	65.5	23.6	51.3	33.9
$k = 30$	66.1	22.4	**53.4**	**34.1**

I FAILURE CASES

GCCA's remaining Hand-Crafted errors fall into two modes tied to the two hierarchical stages. Most are local—its tolerant accuracy leads (Section 4.2)—but a tail are gross mislocalizations, and both modes are illustrated below.

Coarse-localization miss. The chain and the violated condition are identified correctly, but the candidate window returned by coarse localization does not contain the true step. For id 5 ("last word before the second chorus" of a Michael Jackson single, 20 steps), GCCA correctly flags that the lyric was never verified, but returns the window $[14, 19]$, whereas the ground-truth step is 12; fine localization can then only pick within the window, landing at step 16. Here the temporal prior placed the condition slightly too late.

Fine-localization drift. The window does contain the true step, but within it the model selects the wrong one—typically drawn to a later, more salient step, the same symptom-attraction pull the method is designed to resist, now acting inside a narrowed window. For id 8 (monday.com C-suite question, 129 steps), coarse localization returned $[1, 128]$, which narrows nothing at all, and fine localization then returns step 58 against a ground truth of step 8. This case is really a coarse-stage failure: with the window left degenerate, the fine stage faces the whole trajectory and the same late pull as a single pass. A cap on window width would prevent it.

These misses are not all small—id 8's prediction sits 50 steps from the true step. What goal-conditioning removes is their systematic *lateness*, not their size: GCCA's signed bias stays near zero (Section 4.2) because its residual errors no longer pile up late as the baselines' do, even though some remain large.