PASTA: PROVENANCE-GUIDED ABDUCTIVE SELECTION FOR TAILORED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-correction repairs a failed LLM agent from feedback on its final output, and a common design choice is to revise or regenerate that output as a whole. We show that, for agents that act on a user profile, this discards information that the trajectory carries in the provenance of its commitments. The dominant failure, which we call *inference promotion*, turns an inference about the user into a constraint the user never stated and builds later decisions on it; it accounts for 55% of 300 annotated natural failures. PASTA extracts a provenance-typed belief graph from the trajectory, scores each commitment that could explain a verifier-reported violation by a provenance prior and predicted repair, and regenerates from the candidate that best trades this score against downstream cost. With gold answers withheld from every model-visible stage, PASTA reaches 74% repair success against 66% for matched Reflexion and 68% for the strongest simple restart rule, and its margin grows with the number of faults. On five multi-hop and planning benchmarks with injected faults and identical feedback, it outperforms the strongest baseline on every dataset by 2.6 to 9.8 points. On the 300 natural failures, it repairs 76% against 64% for matched full-output retry, comes within one point of annotated-root restart, and cuts reuse of the unsupported belief on follow-up tasks from 30% to 10%. Provenance relative to the user profile is a repair signal, and where to restart is a different question from where the fault began.

1 INTRODUCTION

Agents that act on behalf of a specific user fail in a characteristic way. Consider a deployment agent whose user profile states a $500 monthly cloud budget and that the service processes payments. The agent infers that the service needs 99.99% availability, a target the profile never states, and treats it as a requirement: it provisions a multi-region database, deploys the service across three regions behind a global load balancer, and returns a plan that costs about $1,400 per month. Each step is consistent with the one before it, so when the verifier rejects the plan for exceeding the budget, the cause lies three steps back and every later choice depends on it. We call this failure *inference promotion*: an interpretation of a profile fact is promoted into an operative constraint and built upon. Under a strict four-part definition, it accounts for 165 of 300 natural failures (55%; §4.3).

Repairing such a failure raises three coupled questions: which belief explains the violation, where regeneration should restart, and whether the repair persists once the current answer passes. Self-Refine (Madaan et al., 2023) iteratively critiques and revises outputs, while Reflexion (Shinn et al., 2023) stores verbal reflections to improve subsequent trials. These mechanisms can help an agent learn from failure, but they do not by themselves establish whether a repaired state will reuse a particular unsupported inference. RAFFLES (Zhu et al., 2026a) and EDGE (Hou et al., 2026) study fault attribution, while CausalFlow (Bonagiri et al., 2026) connects attribution to counterfactual repair. RIR (Yu et al., 2026b) selects a restore checkpoint using its own failure diagnosis and retains reflection memory; dependency-guided rollback repair (Yu et al., 2026a) propagates an upstream memory diagnosis through runtime dependencies. PASTA studies a more specific repair signal: explicitly distinguishing user-stated facts from model-inferred commitments when ranking restart candidates.

PASTA (Provenance-guided Abductive Selection for Tailored Agents) answers the first two questions with two principles and tests the third directly. First, *provenance is a repair signal*: a model inference

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.

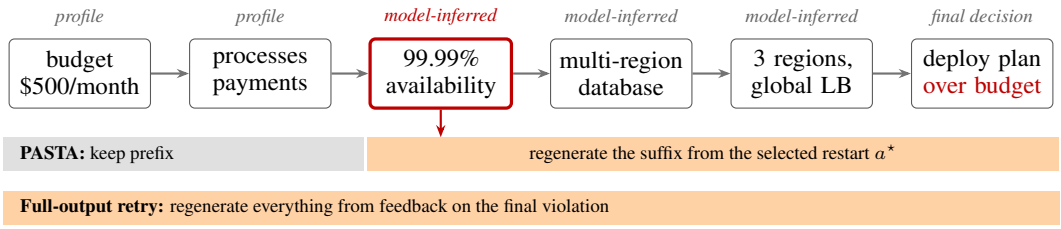


Figure 1: Repairing the deployment example of Section 1. Full-output retry regenerates everything from feedback on the final violation. PASTA constructs candidates from the violated step's ancestry, ranks them by provenance and predicted repair, and restarts from the candidate with the best trade-off against downstream cost (here, the promoted inference). It keeps the prefix before the selected node and regenerates only the suffix.

Algorithm 1 PASTA: abductive ranking and admissible restart selection.

Require: task x, profile P, trajectory τ, verifier V, decay α, at most three attempts

$G \leftarrow \textsc{Extract}(P, x, \tau)$
$\mathcal{V} \leftarrow V(P, x, \tau)$; **if** $\mathcal{V} = \emptyset$ **then return** τ
Take the first violation $v \in \mathcal{V}$ and construct $\mathcal{C}(v; G)$ ▷ Eq. 1
for each $a_i \in \mathcal{C}$ **do**
    Compute the prior π_i and proposal indicators $e_{i1}, \ldots, e_{iM}$
end for
Compute $q^{(0)}$ (Eq. 2) and costs δ_i (Eq. 3); $\mathcal{T} \leftarrow \emptyset$
for $r = 0, 1, 2$ **while** $\mathcal{C} \setminus \mathcal{T} \neq \emptyset$ **do**
    Compute $q^{(r)}$ by down-weighting $\mathcal{T}$ by α and renormalizing ▷ Eq. 4
    $a^\star \leftarrow \arg\max_{a_i \in \mathcal{C} \setminus \mathcal{T}} \left(q_i^{(r)} - \delta_i\right)$ ▷ Eq. 3
    Keep $\tau_{<t_{a^\star}}$ and regenerate the suffix from $a^\star$ to obtain τ'
    if $V(P, x, \tau') = \emptyset$ **then return** τ'; **else** $\mathcal{T} \leftarrow \mathcal{T} \cup \{a^\star\}$
end for
return the last attempted τ' ▷ scored as a failure

that cites a profile fact is a prime suspect, while the fact itself rarely is. Second, *the restart point is not the root cause*: repair needs only a point from which regeneration removes the violation, and a later such point preserves more of an otherwise correct trajectory. PASTA extracts the agent's beliefs with their provenance, abductively ranks those that could explain a violation, restarts from the candidate with the best plausibility–cost trade-off, keeps the prefix, and regenerates the suffix (Figure 1). To test persistence, we retain the repaired state and issue a follow-up task on which the same unsupported belief could cause another error.

With gold answers withheld from every model-visible stage, PASTA reaches 74% repair success against 66% for matched Reflexion and 68% for the strongest simple restart rule, and its margin grows with the number of injected faults. On five multi-hop and planning benchmarks with injected faults and identical feedback, it outperforms Reflexion on every dataset. On 300 human-annotated natural failures, it repairs 76% against 64% for matched full-output retry and cuts reuse of the unsupported belief on follow-up tasks from 30% to 10%. Our contributions are threefold: (i) we identify inference promotion, which accounts for most natural failures in our study, and show that provenance relative to the user profile is an effective repair signal; (ii) we formulate restart selection as a plausibility–cost trade-off distinct from root-cause attribution and show that effective restarts rarely coincide with annotated root causes; and (iii) we evaluate repair in single-model, multi-hop, and human-annotated experiments, including whether a repair changes the agent's later behavior.

2 PASTA

Algorithm 1 summarizes PASTA; the rest of this section defines each of its steps.

Setting. An agent receives a task x and a user profile P of stated facts and constraints, and produces a trajectory $\tau = (s_1, \ldots, s_T, y)$ whose steps are actions, observations, and reasoning summaries and whose final answer is y. A rule-based verifier $V(P, x, \tau)$ returns step-aligned violations of stated constraints; $V(P, x, \tau) = \emptyset$ means the trajectory is accepted. The repairer sees the task, profile, trajectory, tool observations, and constraint-only feedback, but never gold answers. Each commitment the agent acts on has a *provenance*: the profile, a tool, external evidence, world knowledge, or the model's own inference. A failure exhibits *inference promotion* when a supported source fact in P yields an unsupported inferred commitment that is promoted into an operative constraint and used downstream. For a commitment a at step t_a, let $R(a)$ keep $\tau_{<t_a}$ verbatim and regenerate from a; we call a *admissible* if $V(P, x, R(a)) = \emptyset$. Admissible points need not coincide with the root cause: restarting before the fault may regenerate it correctly, and restarting after it may compensate for it. Among admissible points, a later one regenerates less.

Belief graph and candidates. An LLM extracts a graph $G = (A, E)$ from the profile, the task, and the observable steps, without access to hidden chain-of-thought. Each node $a_i = (c_i, \rho_i, t_i, \ell_i)$ records a commitment c_i, its provenance ρ_i, its step t_i, and an extraction confidence $\ell_i \in [0, 1]$. Profile-derived nodes cite the profile spans they rest on, typed edges record dependencies (supports, depends-on, contradicts, downstream-of), and the extractor marks the final-decision node a_f that most directly supports y. PASTA repairs the first reported violation. For a violation at step t_v, with beliefs $B_v = \{a_i : t_i = t_v\}$, the candidates are the violated step's beliefs, their ancestors, and the final decision:

$$\mathcal{C} = B_v \cup \bigcup_{a \in B_v} \mathrm{Anc}(a; G) \cup \{a_f\}. \tag{1}$$

Ancestors admit a promoted inference that entered before the violated step, and a_f admits a repair at the decision itself.

Abductive ranking. Each candidate is a hypothesis about which belief explains the violation. It receives a prior π_i from its provenance and confidence, and a counterfactual generator proposes a minimal replacement and predicts whether it removes the violation ($e_{ij} \in \{0, 1\}$ for proposal $j = 1, \ldots, M$):

$$\pi_i = w_{\rho_i} \max(0.05,\, 1 - \ell_i), \qquad r_i = \frac{1}{M} \sum_{j=1}^{M} e_{ij}, \qquad q_i = \frac{\pi_i\, r_i}{\sum_{k \in \mathcal{C}} \pi_k\, r_k}. \tag{2}$$

If every $r_k = 0$, then $q_i = \pi_i / \sum_k \pi_k$. The weights encode the first principle: w_ρ is 1.0 for model inferences, 0.6 for world knowledge, 0.4 for external evidence, 0.25 for tool observations, and 0.15 for profile facts; they are fixed defaults, not tuned per benchmark. The generator returns three proposals in one call, and we use the first ($M = 1$).

Restart selection. PASTA trades predicted repair against a normalized downstream cost. With D_i the descendants of a_i in G and $\mathcal{T}$ the candidates already tried,

$$\delta_i = \frac{|D_i| + 1}{\max_{k \in \mathcal{C}} (|D_k| + 1)}, \qquad a^\star = \arg \max_{a_i \in \mathcal{C} \setminus \mathcal{T}} \big(q_i - \delta_i\big). \tag{3}$$

With equally scored candidates, Eq. 3 reduces to minimum-cost selection, and on stored first-round decisions the cost term changes the selected restart in 30% of multi-candidate cases.

Regeneration and retry. Regeneration keeps the steps before $t_{a^\star}$ and regenerates the suffix from the selected node, given the task, profile, prefix, downstream-node information, and tool observations; it receives the identity of the selected node but not the proposed replacement text. If V rejects the result, $a^\star$ joins $\mathcal{T}$, and the first-round scores are down-weighted by $\alpha = 0.1$ for tried candidates and renormalized:

$$q_i^{(r)} = \frac{q_i^{(0)}\, \alpha^{\mathbf{1}[i \in \mathcal{T}_r]}}{\sum_{k \in \mathcal{C}} q_k^{(0)}\, \alpha^{\mathbf{1}[k \in \mathcal{T}_r]}}. \tag{4}$$

Because tried candidates are already excluded from selection, the update divides each untried score by $Z_r = 1 - (1 - \alpha) \sum_{k \in \mathcal{T}_r} q_k^{(0)} \leq 1$, which shifts weight from cost toward plausibility on later attempts. PASTA makes at most three attempts, each from the original trajectory; if all are rejected, it returns the last attempt, which counts as a failure. Appendix F proves the properties of this procedure, and Appendix G works through the deployment example.

3 EXPERIMENTAL SETUP

Single-model experiments. The matched evaluation withholds gold answers and target-bearing metadata from every model-visible stage (generation, extraction, proposal scoring, feedback, and regeneration) and uses them only for final scoring. All arms start from identical trajectories and share task information, observations, the repair model and prompt, constraint-only feedback, and at most three repair attempts. The baselines are matched Reflexion (Shinn et al., 2023), matched flat feedback, which re-prompts with the violated constraint, and selector controls that change only the restart policy (final decision, latest step, random candidate, or earliest candidate); three ablations remove the provenance prior, the graph, or proposal scoring. The nested-fault evaluation injects $N = 0$–4 faults into the same tasks with one injector. Both use Llama 3.1 8B (Grattafiori et al., 2024).

Multi-hop and additional benchmarks. For $N = 2$–5 faults, items from HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (2Wiki; Ho et al., 2020), LongMemEval (Wu et al., 2025), MuSiQue (Trivedi et al., 2022), and TravelPlanner (Xie et al., 2024) receive a context note suggesting a near-miss answer and $N-1$ flipped profile constraints, with hop corruptions at larger N; fields of real Amazon product and Food.com recipe records are corrupted in the same way. Profile-conditioned QA on TruthfulQA (Lin et al., 2022), 2Wiki, LongMemEval, and MuSiQue attaches 3–5 constraints from 28 templates, about half of them distractors, and requires a reference answer and a passing profile predicate; on PersonalWAB (Cai et al., 2025), the agent recommends one item from a 20-item slate. These experiments use Llama 3.1 8B, Llama 3.3 70B, Mistral NeMo 12B (Mistral AI, 2024), and Qwen 2.5 7B (Yang et al., 2024) and compare direct generation (Direct), Self-Refine (SR), Full-Regen (FR), which discards the trajectory and resamples, Reflexion (Rfx), ReAct (Yao et al., 2023b), SelfCheckGPT (SChk; Manakul et al., 2023), and Tree of Thoughts (ToT; Yao et al., 2023a). On the QA datasets, the harness's verifier feedback includes the expected answer, so every method receives identical, informative feedback. Cost, component, and scale analyses come from an earlier configuration without sequential retry, which also covers DeepSeek-V3 (Liu et al., 2024) and Gemini-2.5-Flash (Comanici et al., 2025) (Appendices B and K).

Human annotation and statistics. We collect 300 fresh natural failures, repair them from identical trajectories with a shared prompt and model, constraint-only feedback, and at most three attempts, and have blinded judges assess correctness; annotated-root restart receives the annotated fault as a privileged diagnostic, and annotators code inference promotion only when all four of its elements are present (Appendix J). For 150 confirmed promotion cases, each arm's repaired state is retained and a follow-up task is issued, and reuse of the unsupported belief is counted over all 150. Two annotators blinded to method identity marked the earliest faulty step of 85 failed TravelPlanner trajectories (category $\kappa = 0.745$); localization reports top-1 agreement, agreement within one step, and mean reciprocal rank (MRR). We use exact two-sided McNemar tests for paired outcomes, paired cluster-aware uncertainty for the matched evaluation, Holm correction within baseline families, and item-bootstrap intervals for MRR (Appendix M). Calls use temperature 0 (0.5 for the counterfactual generator) and are cached by a hash of their full signature.

4 RESULTS

4.1 SINGLE-MODEL EXPERIMENTS

The arms in Table 1a differ only in how they repair (Section 3), so the rows decompose PASTA's 8-point margin over matched Reflexion. Stating the violated constraint instead of a verbal reflection adds 1 point, restarting from the strongest simple rule adds 1 more point, and selecting the restart accounts for the remaining 6 points. The gain therefore comes from where PASTA restarts. A restart helps only when it is selected well, since a fixed restart rule is barely better than no restart.

The selection needs all three components. Removing any one of them leaves PASTA only 1 to 3 points above the strongest simple rule, so each removal forfeits at least half of the 6-point selection gain. The three losses (5, 4, and 3 points) sum to twice that gain, which shows that the components interact rather than add. The provenance prior carries the most weight. With a uniform prior, the graph and the counterfactual proposals together gain only 1 point over a simple rule. Predicted repair alone does not separate a promoted inference from the profile fact it cites, because regenerating from

Table 1: Single-model experiments (Llama 3.1 8B, answer-blind). Gap: PASTA minus the arm. Margin: PASTA minus matched Reflexion.

(a) Matched arms and ablations

Arm	Success (%)	Gap (pp)
Matched Reflexion	66	+8
Matched flat feedback	67	+7
Strongest simple restart rule	68	+6
PASTA, uniform provenance prior	69	+5
PASTA, flat candidates (no graph)	70	+4
PASTA, no proposal scoring	71	+3
PASTA	**74**	–

(b) Nested faults

Faults N	Margin (pp)
1	+2
2	+4
3	+7
4	+10

Table 2: Multi-hop and real-record experiments: success (%) pooled over fault counts and four LLMs, and PASTA/Reflexion by N (–: not run). †Output-coercion defect in Reflexion on Llama 3.3 70B.

	Pooled					PASTA / Rfx by N			
Dataset	Direct	SR	FR	Rfx	PASTA	N=2	N=3	N=4	N=5
HotpotQA	57.9	58.9	57.0	71.6	**81.4**	84.8/84.1	79.9/68.3	79.6/62.3	–
2Wiki	60.8	63.1	64.2	71.5	**78.6**	90.6/89.0	88.0/78.9	88.4/73.4	34.6/33.7
LongMemEval	45.8	49.1	46.6	57.7	**65.3**	73.9/71.7	66.0/57.9	56.1/43.6	–
MuSiQue	46.2	49.6	45.7	57.2	**62.4**	85.6/79.9	80.8/73.5	44.3/42.4	39.0/33.2
TravelPlanner	34.0	32.3	31.3	32.6	**36.6**	63.3/55.4	30.1/27.1	16.3/15.3	–
Amazon products	47.8	44.9	43.3	**62.5**	61.0	56.4/75.5	37.2/50.7	19.5/27.0	–
Food.com recipes†	45.6	43.3	42.8	55.4	**59.6**	88.5/82.6	54.2/53.2	14.0/9.2	18.3/8.1

either one can remove the violation. In the worked example (Appendix G), the generator predicts repair for both nodes, and only the prior ranks the inference first.

Table 1b varies only the number of faults. The margin grows by about 2.7 points per fault, from 2 points at one fault to 10 points at four faults, and it shows no sign of saturating. A global revision must undo every fault in one pass, whereas a restart before the earliest suspected fault regenerates all later faults together. Items and injector are fixed across fault levels, so the widening is attributable to fault count rather than to item difficulty or injection method.

4.2 Multi-Hop Experiments

Under identical feedback, PASTA exceeds Reflexion in all 17 benchmark–fault-count cells (Table 2, top). The margin is small at $N = 2$, where it ranges from 0.7 to 7.9 points, and its change with N separates the benchmarks. On HotpotQA and 2Wiki, PASTA keeps 94% and 98% of its $N = 2$ success at $N = 4$, while Reflexion keeps only 74% and 82%. At $N = 4$, PASTA therefore fails about half as often as Reflexion, with error ratios of 0.54 and 0.44. The margin reaches 17.3 and 15.0 points because Reflexion degrades, not because PASTA improves. LongMemEval shows the same pattern more weakly, with a margin of 12.5 points at $N = 4$. The margin closes wherever PASTA itself collapses. PASTA's success falls below 45% in five cells (MuSiQue at $N = 4$ and 5, 2Wiki at $N = 5$, and TravelPlanner at $N = 3$ and 4), and the margin in these cells is only 0.9 to 5.8 points, whereas PASTA stays above 56% in every other cell. These collapses do not track fault count alone. On MuSiQue, most of the loss falls between $N = 3$ and $N = 4$, where PASTA loses 36.5 points and Reflexion loses 31.1 points; at this step, a separate injector adds hop corruptions and the item subset changes. On 2Wiki, both methods lose 40 to 54 points between $N = 4$ and $N = 5$, which use the same injector but different item subsets. On TravelPlanner, the loss comes earlier, between $N = 2$ and $N = 3$. Each flipped constraint can add a violation elsewhere in the plan. PASTA builds candidates from the first violation and accepts only a plan with no violations, so its restart must precede all of them, which removes the advantage of restarting late.

On the four QA datasets, the feedback states the expected answer (Section 3), and the baselines show how much of the absolute gain this accounts for. Self-Refine and Full-Regen stay within 3.4 points of Direct on every benchmark and fall below Direct on both record sets, so neither self-critique nor

Table 3: Human annotation studies. In (c), top-1 and ± 1 are counts of 85, and brackets are item-bootstrap 95% CIs. †Given the annotated fault as a privileged diagnostic.

(a) Repair, 300 failures

Repair method	Success (%)
Matched full-output retry	64
Random restart	65
Latest-step restart	66
Final-decision restart	68
PASTA, uniform provenance	69
PASTA, no proposal scoring	71
PASTA	**76**
Annotated-root restart†	77

(b) Follow-up reuse, 150 promotion cases

Method	Repaired (%)	Reused (%)
Full-output retry	66	30
Final-decision restart	70	28
PASTA	**78**	**10**

(c) Localization, 85 TravelPlanner trajectories

Selector	Top-1	± 1	MRR
PASTA restart	7	17	0.21 [0.15, 0.28]
Random ancestor	10	29	0.22 [0.16, 0.28]
Most recent step	12	16	0.28 [0.21, 0.36]

resampling repairs injected faults. Reflexion gains 10.7 to 13.7 points over Direct on the QA datasets, but it loses 1.4 points on TravelPlanner, which is the one benchmark whose feedback carries no answer. PASTA gains 2.6 points over Direct on TravelPlanner. Absolute gains over Direct thus depend heavily on answer-bearing feedback. The controlled quantity in this table is the PASTA–Reflexion margin, which both methods earn under the same feedback.

Real-record corruption produces the one clear loss (Table 2, bottom). On Amazon products, Reflexion leads by 19.1, 13.5, and 7.5 points at $N = 2$, 3, and 4. The pooled deficit is only 1.5 points because the pooled rate also includes $N = 1$. Corruption of these records often removes every identity-defining field (Appendix D), so the fault lies in the evidence the agent reads rather than in an inference it draws. PASTA's prior weights tool observations at 0.25 and external evidence at 0.4, whereas model inferences receive 1.0, so the prior points away from such a fault. This is consistent with the loss and with the case for learned priors (Section 6). On Food.com, PASTA leads by 4.2 points pooled, but its lead shrinks to 1.2 points (58.2% against 57.0%) once the defect-affected Llama 3.3 70B runs are excluded. We treat this result as a tie.

4.3 Human Annotation Studies

Of the 300 natural failures, 165 (55%) meet all four criteria of inference promotion (category agreement $\kappa \approx 0.78$), so the failure that PASTA targets is the majority mechanism in data we did not construct. PASTA repairs 76% of the failures, whereas matched full-output retry repairs 64%. Exactly one method repairs 66 of the failures, and PASTA is that method in 51 of them (exact McNemar $p = 1.01 \times 10^{-5}$). The asymmetry is large. PASTA repairs 47% of the failures that retry leaves unrepaired (51 of 108), while retry repairs 21% of those that PASTA leaves unrepaired (15 of 72). Annotated-root restart repairs 77% of the failures. Without access to the annotation, PASTA closes 12 of the 13 points between retry and this reference, and it regenerates 40% fewer tokens.

The rules in Table 3a separate where a repair restarts from how the restart is chosen. Simple rules gain only 1 to 4 points over retry, so restarting partway through the trajectory is not what helps. Final-decision restart (68%) is the rule that PASTA's selection reduces to under uniform scores (Proposition 2), so the 8 points between it and PASTA measure the value of the scores. Removing the provenance prior costs 7 points here and removing the proposals costs 5 points, compared with 5 and 3 points in the single-model experiments. This difference is consistent with a population in which most failures are promotions, which is the case the prior is built to rank first.

Repairing the task and removing the unsupported belief are separate outcomes (Table 3b). Full-output retry regenerates the entire trajectory, and final-decision restart regenerates only its last node, yet the belief resurfaces on the follow-up task after 30% and 28% of their repairs. The amount a repair regenerates therefore does not decide whether the belief survives. Final-decision restart repairs 6 more original tasks than retry (105 against 99), but it lowers reuse by only 3 cases. This pattern is the signature of a repair that satisfies the verifier downstream of the belief and keeps the belief in the prefix (Scenario (ii) of Appendix G). After PASTA, reuse falls to 10%. At least 102 of the 150 cases (68%) are both repaired and free of reuse under PASTA, compared with at least 54 (36%) under retry

Table 4: Additional benchmarks: success (%). Δ and p: PASTA against Reflexion, exact McNemar. Bold: best method. *Unparseable rows dropped. ‡Partial run of 27 pairs.

(a) Profile-conditioned QA (answer-informed), $n = 36$ (TruthfulQA) or 60 per cell

Dataset	LLM	Direct	SR	Rfx	FR	ReAct	SChk	ToT	PASTA	Δ	p
TruthfulQA	Llama 3.1 8B	91.7	88.9	94.4	91.7	91.7	88.9	88.9	**97.2**	+2.8	1.000
	Llama 3.3 70B	83.3	86.1	91.7	83.3	80.6	86.1	86.1	**97.2**	+5.6	0.500
	Mistral NeMo 12B	69.4	66.7	80.6	72.2	72.2	72.7*	56.5*	**88.9**	+8.3	0.250
2Wiki	Llama 3.1 8B	93.3	91.7	93.3	91.7	93.3	93.3	88.3	**96.7**	+3.3	0.500
	Llama 3.3 70B	96.7	96.7	96.7	96.7	96.7	96.7	93.3	**98.3**	+1.7	1.000
	Mistral NeMo 12B	95.0	91.7	95.0	93.3	95.0	89.8*	92.2*	**100.0**	+5.0	0.250
LongMemEval	Llama 3.1 8B	71.7	71.7	76.7	66.7	71.7	71.7	70.0	**88.3**	+11.7	0.016
	Llama 3.3 70B	70.0	66.7	70.0	70.0	70.0	70.0	66.7	**73.3**	+3.3	0.500
	Mistral NeMo 12B	68.3	68.3	73.3	65.0	68.3	70.0	67.9*	**81.7**	+8.3	0.063
MuSiQue	Llama 3.1 8B	88.3	86.7	90.0	81.7	88.3	86.7	88.3	**95.0**	+5.0	0.250
	Llama 3.3 70B	96.7	95.0	96.7	90.0	96.7	96.7	91.7	**98.3**	+1.7	1.000
	Mistral NeMo 12B	78.3	75.0	85.0	73.3	76.7	80.0	68.6*	**90.0**	+5.0	0.250

(b) PersonalWAB (answer-informed), 20-item-slate recommendation

	Llama 3.1 8B	Mistral NeMo 12B	Qwen 2.5 7B, run 1‡	Qwen 2.5 7B, run 2	Pooled, 567 pairs
Direct	96.7	91.1	92.6	91.7	–
Rfx	98.9	96.7	92.6	93.3	96.1
PASTA	**99.4**	**97.8**	**100.0**	**96.7**	**98.1**
Δ	+0.6	+1.1	+7.4	+3.3	+1.9
p	1.00	0.69	0.50	0.031	0.013

and 63 (42%) under final-decision restart. These are lower bounds that subtract reuse from repairs, since reuse is counted over all 150 cases.

PASTA's restart points agree with the annotated earliest faulty step no better than chance (Table 3c). They match it in 7 of 85 trajectories with an MRR of 0.21, whereas a random ancestor reaches an MRR of 0.22 and the most recent step reaches 0.28. This low agreement is the intended behavior of the rule. On 60 synthetic graphs with a known fault, the rule ranks the fault first whenever the fault is the only node whose regeneration resolves the violation (MRR 1.0). When several nodes resolve it, the rule never ranks the fault first but always places it in the top three (top-1 0.0, top-3 1.0), because the cost term prefers a cheaper admissible node. Agreement with annotated root causes, which is the target of attribution benchmarks, is therefore not a proxy for restart quality. The most recent step agrees with the annotation most often, yet as a restart rule on the 300 failures, which form a different set, it repairs 10 points fewer than PASTA.

4.4 Additional Benchmarks

Table 4a adds search- and sampling-based baselines on answer-informed QA with profile predicates. PASTA has the highest success in all 12 cells. It fails on 53 of 648 items, whereas Reflexion fails on 86, so PASTA has 38% fewer failures. Averaged over cells, none of the five ungated baselines, which never consult the verifier, improves on Direct. Their mean differences from Direct range from -3.7 to 0.0 points, and this holds even for ToT at about 49 calls per task. Only Reflexion (+3.4 points) and PASTA (+8.5 points) improve on Direct. The margin over Reflexion tracks headroom. It averages 7.3 points in the five cells where Reflexion is below 90% and 3.6 points in the seven cells where Reflexion is at or above 90%. No cell survives Holm correction at these sample sizes. In the nine cells with a net difference of at least two items, however, the exact p-values imply that Reflexion solves no item that PASTA misses, so in those cells PASTA's successes contain Reflexion's. Baseline runs predate PASTA's runs by four to six months on hosted endpoints, so provider-side model updates are not controlled in this comparison. The single-model experiments (Section 4.1) are the controlled counterpart.

PersonalWAB (Table 4b) builds profiles from real purchase, rating, and review histories, so it is the benchmark closest to deployed personalization and the only one here that is not QA. Every method exceeds 91% on it, which leaves little headroom. PASTA still leads Reflexion in all four cells. Pooled

over 567 pairs, PASTA reaches 98.1% and Reflexion reaches 96.1%, so PASTA's failures fall from 22 to 11. The pooled difference rests on 14 pairs that only PASTA solves and 3 pairs that only Reflexion solves (exact $p = 0.013$). The largest cell gain, 7.4 points, comes from a partial run of 27 pairs. The complete Qwen 2.5 7B run gains 3.3 points ($p = 0.031$), and no cell survives Holm correction.

4.5 Cost, Robustness, and Scale

Cost. An accepted trajectory costs two LLM calls, and a repaired one costs $2 + |\mathcal{C}| + k$ calls for $k \leq 3$ attempts, which is typically six to eight. Earlier token counts record only agent and extraction calls and are therefore lower bounds; on that basis, PASTA uses 0.8 to 1.8 times Reflexion's tokens and 11 to 14 times fewer tokens than ToT (Appendix K).

Robustness. Extraction is the one stage the verifier never checks, and a blind audit finds provenance macro-F1 of 0.88, dependency precision of 0.85, and usable repair-candidate coverage of 0.93. PASTA's advantage falls from 6 to 4 points when profile templates and one domain are held out. Across 100 re-draws of every profile, PASTA leads Reflexion in 92, with a mean effect of +2.07 points (95% CI [+1.06, +3.18]). The preregistered confirmatory tier, however, has a bottom-decile mean of -7.21 points and fails its bottom-decile criterion, so the advantage is consistent on average but not uniform across profiles (Appendix L).

Scale. The earlier configuration does not carry over to larger models. With DeepSeek-V3 and Gemini-2.5-Flash, Reflexion leads in 7 of 8 cells (Table 13, Appendix K), and PASTA recovers only 6% and 8% of Reflexion's answer-informed gains on TruthfulQA (44.4 and 33.3 points over Direct). The final configuration has not been run at this scale.

5 Related Work

Self-correction. Self-Refine (Madaan et al., 2023) revises outputs using self-generated feedback, Reflexion (Shinn et al., 2023) retains verbal reflections from task feedback across trials, and CRITIC (Gou et al., 2024) uses external tools to verify and revise outputs. Tree of Thoughts, LATS, RAP, and Graph of Thoughts organize search or refinement over intermediate reasoning (Yao et al., 2023a; Zhou et al., 2024; Hao et al., 2023; Besta et al., 2024). Search can revisit earlier decisions: Tree of Thoughts explicitly supports backtracking, and LATS combines Monte Carlo tree search with reflection and environment feedback. Studies of intrinsic self-correction find that improvement is not reliable without appropriate feedback (Huang et al., 2024; Kamoi et al., 2024). PASTA's distinction is therefore not the ability to revisit a trajectory, but its explicit provenance-weighted ranking of belief-level restart candidates after a verifier-reported violation.

Fault attribution. Tyen et al. (2024) find that supplying error locations improves correction on their tested reasoning tasks, while ProcessBench measures identification of the earliest erroneous mathematical reasoning step (Zheng et al., 2025). Agent research provides failure taxonomies (Cemri et al., 2025) and trace-attribution benchmarks such as TraceElephant and TRAIL (Chen et al., 2026; Deshpande et al., 2025). Attribution methods include StepFinder's temporal-semantic framework (Zhu et al., 2026b), EDGE's error dependency graphs (Hou et al., 2026), Abduct–Act–Predict's causal scaffolding (West et al., 2025), VerifyMAS's hypothesis verification (Qiao et al., 2026), and multi-granularity causal inference (Ma et al., 2025). EDGE uses repair interventions and counterfactual rollouts to validate dependencies; its evaluated detector outputs error attributions rather than an end-to-end recovery policy. These attribution objectives differ from choosing an effective restart point, although attribution can support repair. PASTA evaluates restart quality separately from agreement with the annotated originating fault (§4.3).

Repair and rollback. CausalFlow (Bonagiri et al., 2026) evaluates interventions at candidate steps and selects successful repairs with a minimality criterion. GA-Rollback, WebRollback, and AgentRewind restore earlier agent or environment states (Li et al., 2025; Zhang et al., 2026; Zhuang et al., 2026). RIR (Yu et al., 2026b) generates a failure diagnosis, identifies a causally relevant interval, selects the latest checkpoint within that interval that removes the diagnosed obstruction, and carries reflection memory across the rollback. Dependency-guided rollback repair (Yu et al.,

2026a) instead takes an upstream diagnosis of faulty memories, preserves independently supported state, and replays affected answer-relevant computation; its recurrence evaluation re-runs the last user input and is conditioned on successful recovery. MemoRepair (Zhao et al., 2026) addresses cascades in agent memory. PASTA receives a verifier-reported violation rather than a supplied faulty-node diagnosis, ranks candidates by user-relative provenance and predicted repair, and evaluates reuse of the unsupported belief on a new task over all cases.

Diagnosis and personalization. Model-based diagnosis derives explanations from a system model and observations (Reiter, 1987; de Kleer & Williams, 1987; Console & Torasso, 1991); spectrum-based localization uses execution spectra from passing and failing tests (Jones & Harrold, 2005; Abreu et al., 2006; Wong et al., 2016); and delta debugging repeatedly tests reduced inputs (Zeller & Hildebrandt, 2002). GenProg searches for code changes validated by a test suite (Le Goues et al., 2012). These methods motivate evaluating an intervention by its effect rather than treating localization agreement as the sole objective. PASTA applies this distinction to restart selection from a failed agent trajectory, using an extracted belief graph and a verifier rather than a formal program model. Personalization research documents profile-induced hallucinations (Sun et al., 2026) and evaluates personalized outputs and preference following (Salemi et al., 2024; Zhao et al., 2025); PASTA targets the more specific mechanism of inference promotion. Appendix N extends this discussion.

6 Future Work and Conclusion

Personalized agents fail when they promote inferences about their users into operative constraints. PASTA repairs such failures by using provenance and predicted repair to choose where to restart, trading plausibility against downstream cost. Across single-model, multi-hop, and human-annotated experiments, it repairs more failures than matched baselines, approaches annotated-root restart at 40% lower regeneration cost, and reduces reuse of the unsupported belief on later tasks from 30% to 10%.

Learned provenance priors. The provenance weights are fixed defaults. Learning them per domain or model would let the prior adapt to settings where tool outputs or world knowledge, rather than the model's own inferences, are the likelier source of error.

Interventional ranking and cost. Counterfactual execution could supply interventional evidence in place of predicted repair, and token-level costs could replace descendant counts so that the trade-off tracks actual regeneration cost.

From restart to state repair. Characterizing when the latest admissible restart also removes the promoted belief would connect restart selection with state repair and carry the reduction in belief reuse to longer interactions.

Prevention and training. Agents could catch promotion at commit time by asking the user before an inference becomes a constraint, and pairs of promoted and repaired trajectories could serve as preference data for training agents that promote less.

Ethics Statement

Profiles are synthetic or derived from public benchmark data, and no human-subject data beyond annotator labels were collected. Better repair of personalized agents could also make profile-conditioned systems more persuasive or more effective at acting on inferred preferences. PASTA's provenance labels make inferred commitments explicit, which can support user review, but they do not guarantee that stated preferences are appropriate to act on.

Reproducibility Statement

All LLM calls use hosted APIs, and no local GPU inference is used. Every model call is cached by a SHA-256 hash of its full call signature, example indices are drawn with a fixed seed, and

pairing across methods is checked by automated unit tests. Appendix A maps every result to its configuration and analysis; Appendices B–E describe the configurations, models, data, and baselines; Appendices F–H give proofs, a worked example, and a reference implementation; Appendices J–M give the protocols, additional results, robustness details, and statistical procedures; and Appendices P and Q give the prompts and the providers, caching, sampling, and run windows.

AI USE STATEMENT

We used an AI assistant (Claude) to restructure, draft, and copy-edit the manuscript text, check its arithmetic and internal consistency, draw the overview figure, draft appendix material (proofs, the worked example, the reference implementation, and descriptive sections), format tables, and check bibliography keys. All experimental design, results, analyses, and claims are the authors'; the authors verified all numbers and citations and take full responsibility for the content.

REFERENCES

Samir Abdaljalil, Hasan Kurban, Khalid Qaraqe, and Erchin Serpedin. Theorem-of-Thought: A multi-agent framework for abductive, deductive, and inductive reasoning in language models. arXiv preprint arXiv:2506.07106, 2025. URL https://arxiv.org/abs/2506.07106.

Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. An evaluation of similarity coefficients for software fault localization. In *Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 39–46, 2006. doi: 10.1109/PRDC.2006.18.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Akash Bonagiri, Devang Borkar, Gerard Janno Anderias, Setareh Rafatirad, and Houman Homayoun. CausalFlow: Causal attribution and counterfactual repair for LLM agent failures. arXiv preprint arXiv:2605.25338, 2026.

Hongru Cai, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua. Large language models empowered personalized web agents. In *Proceedings of the ACM Web Conference 2025 (WWW)*, pp. 198–215, 2025. doi: 10.1145/3696410.3714842.

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 *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2025.

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 *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19888–19905, 2026. doi: 10.18653/v1/2026.acl-long.912.

Gheorghe Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261, 2025. URL https://arxiv.org/abs/2507.06261.

Luca Console and Pietro Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3):133–141, 1991. doi: 10.1111/j.1467-8640.1991.tb00388.x.

Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1): 97–130, 1987. doi: 10.1016/0004-3702(87)90063-4.

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. URL https://arxiv.org/abs/2505.08638.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024. URL https://arxiv.org/abs/2407.21783.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8154–8173, 2023.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pp. 6609–6625, 2020.

Jun Hou, Priya Pitre, Yi Fang, and Xuan Wang. EDGE: Error dependency graph-guided multi-error attribution in multi-agent LLM systems. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026. arXiv:2609.01360.

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. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

James A. Jones and Mary Jean Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 273–282, 2005. doi: 10.1145/1101908.1101949.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12, 2024.

Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer. GenProg: A generic method for automatic software repair. *IEEE Transactions on Software Engineering*, 38(1):54–72, 2012. doi: 10.1109/TSE.2011.104.

Xingzuo Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yong Xu, and Min Zhang. Generator-assistant stepwise rollback framework for large language model agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 17694–17711, 2025.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, 2022.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. arXiv preprint arXiv:2412.19437, 2024. URL https://arxiv.org/abs/2412.19437.

Guoqing Ma, Jia Zhu, Hanghui Guo, Weijie Shi, Jiawei Shen, Jingjiang Liu, and Yidan Liang. Automatic failure attribution and critical step prediction method for multi-agent systems based on causal inference. arXiv preprint arXiv:2509.08682, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9004–9017, 2023.

Meta. Llama 3.3 70B instruct model card. https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct, 2024.

Mistral AI. Mistral NeMo. https://mistral.ai/news/mistral-nemo, 2024. Model announcement.

Hezhe Qiao, Hanghang Tong, Ee-Peng Lim, Bing Liu, and Guansong Pang. VerifyMAS: Hypothesis verification for failure attribution in LLM multi-agent systems. arXiv preprint arXiv:2605.17467, 2026.

Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi: 10.1016/0004-3702(87)90062-2.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. LaMP: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhongxiang Sun, Yi Zhan, Chenglei Shen, Weijie Yu, Xiao Zhang, Ming He, and Jun Xu. When personalization misleads: Understanding and mitigating hallucinations in personalized LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8041–8060, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13894–13908, 2024.

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. arXiv preprint arXiv:2509.10401, 2025.

W. Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. A survey on software fault localization. *IEEE Transactions on Software Engineering*, 42(8):707–740, 2016. doi: 10.1109/TSE.2016.2521368.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54590–54613, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024. URL https://arxiv.org/abs/2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2369–2380, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023b.

Caili Yu, Yiqi Wang, Jiaqi Zhang, Yiqun Duan, Mingkai Zheng, Zhangkai Wu, Kaize Shi, and Taotao Cai. From faulty memories to corrected actions: Dependency-guided rollback repair for memory-augmented agents. arXiv preprint arXiv:2608.10502, 2026a. URL https://arxiv.org/abs/2608.10502.

Yi Yu, Liuyi Yao, Yaliang Li, Enshu Wang, and Libing Wu. Rollback the world, keep the reflection: Rollback-induced reflection for long-horizon LLM agents. arXiv preprint arXiv:2609.18304, 2026b. URL https://arxiv.org/abs/2609.18304.

Andreas Zeller and Ralf Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Transactions on Software Engineering*, 28(2):183–200, 2002. doi: 10.1109/32.988498.

Zhisong Zhang, Tianqing Fang, Kaixin Ma, Wenhao Yu, Hongming Zhang, Haitao Mi, and Dong Yu. WebRollback: Enhancing web agents with explicit rollback mechanisms. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 187–197, 2026.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. Do LLMs recognize your preferences? Evaluating personalized preference following in LLMs. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Yang Zhao, Chengxiao Dai, Mengying Kou, and Yue Xiu. MemoRepair: Barrier-first cascade repair in agentic memory. arXiv preprint arXiv:2605.07242, 2026.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160, 2024.

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 *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7659–7688, 2026a.

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. doi: 10.1145/3770855.3817991. arXiv:2606.03467.

Yu Zhuang, Kefei Chen, Yitong Duan, Shuxin Zheng, Jian Li, and Xu-Yao Zhang. AgentRewind: Recoverable execution for long-horizon LLM agents. arXiv preprint arXiv:2608.14380, 2026. URL https://arxiv.org/abs/2608.14380.

A Reproducibility Index

This index maps each reported result to its configuration, data, and analysis.

- **Single-model experiments** (§4.1, Table 1). Llama 3.1 8B; answer-blind serialization (Appendix J); identical initial trajectories and matched repair information; three-attempt configuration (Table 6); paired, cluster-aware uncertainty (Appendix M); nested faults from one injector on shared items (Appendix D).
- **Injected faults** (§4.2, Table 2). Five benchmarks and two real-record datasets with four LLMs; injection procedure in Appendix D; baselines in Table 7.
- **Human annotation studies** (§4.3, Table 3). 300 natural failures with blinded correctness judgments and the criteria of Table 10; 150 follow-up cases; 85 annotated TravelPlanner trajectories; exact McNemar tests and item-bootstrap intervals (Appendix M); a synthetic check on 60 graphs without LLM calls.
- **Additional benchmarks** (§4.4, Table 4). Profile-conditioned QA and PersonalWAB, with flat-feedback controls in Appendix K; exact McNemar tests with Holm correction (Appendix M).
- **Cost, robustness, and scale** (§4.5; Appendix K, Tables 11–13). The earlier configuration without sequential retry (Appendix B); regenerated tokens relative to annotated-root restart; blind graph audit; profile dependence, held-out templates, and 100 profile re-draws (Appendix L).

Shared artifacts are the PASTA configuration (Table 6), the baseline settings (Table 7), a reference implementation of the selection rule (Appendix H), the prompts (Appendix P), and the caching, seeding, and pairing procedures (Appendix Q).

B Experimental Configurations

The results come from two PASTA configurations and two evaluation regimes, collected over several studies.

Configurations. The answer-blind evaluation, its ablations, and the natural-failure study use up to three repair attempts with sequential retry (Eq. 4, $\alpha = 0.1$). An earlier configuration without sequential retry underlies the component analyses and proposal-count sweep (Table 12), the answer-informed cost measurements (Table 11), and the model-scale runs (Table 13); its success on the four component-analysis cells is 86.1%, 77.8%, 96.7%, and 78.3%.

Regimes. In the answer-informed regime, the verifier's violation message states the expected answer on 2Wiki, HotpotQA, LongMemEval, and MuSiQue, and the task record passed to every method contains reference-answer metadata. In the answer-blind regime, gold answers and target-bearing metadata are removed from every model-visible stage, and references are used only for scoring. Our primary claims rest on the answer-blind regime.

How the studies relate. Several studies refine earlier ones. The earlier 85-case TravelPlanner annotation used broad fault categories (profile misuse, faulty assumption, reasoning); the 300-case study codes strict inference promotion with the four criteria of Table 10. The answer-informed multi-fault study draws different item subsets at each fault count and switches injectors at $N \geq 4$; the nested-fault evaluation shares items and one injector across fault levels to remove these confounds. The profile re-draw analysis adds a preregistered confirmatory tier, restricted after a paraphrase audit to template types whose paraphrases all preserved meaning. Finally, the follow-up evaluation tests whether a repair changes later behavior, which verifier acceptance alone cannot show.

Run windows. Baseline runs for the profile-conditioned matrix date from March–May 2026 and PASTA runs from September 2026. All calls go through hosted APIs (Appendix Q).

C Models

All models are accessed through hosted APIs (Appendix Q).

- **Llama 3.1 8B** (Grattafiori et al., 2024), an open-weight dense model from Meta, is the model for the answer-blind evaluation, the natural-failure and follow-up studies, and the nested-fault evaluation. It also appears in the multi-fault study, the profile-conditioned matrix, the flat-feedback controls, the component analyses, and PersonalWAB.
- **Llama 3.3 70B** (Meta, 2024), a larger open-weight model from the same family, is the largest model in the profile-conditioned matrix and the multi-fault study.
- **Mistral NeMo 12B** (Mistral AI, 2024), an open-weight model from Mistral AI and NVIDIA, appears in the multi-fault study, the profile-conditioned matrix, the flat-feedback controls, the component analyses, and PersonalWAB.
- **Qwen 2.5 7B** (Yang et al., 2024), an open-weight model from Alibaba, appears in the multi-fault study, PersonalWAB (two runs), the component analyses, and the scale study.
- **DeepSeek-V3** (Liu et al., 2024), a mixture-of-experts model with 671B total and 37B activated parameters per token, and **Gemini-2.5-Flash** (Comanici et al., 2025), a proprietary Google model, appear only in the scale study (Table 13).

The set spans dense open-weight models from 7B to 70B parameters, one mixture-of-experts model, and one proprietary model. PASTA calls use temperature 0 for the agent, extraction, and re-execution and 0.5 for the counterfactual generator, with at most 8192 tokens per call (Table 6).

D Benchmarks and Data

Table 5 summarizes the benchmarks.

Table 5: Benchmarks and their use in this paper.

Benchmark	Content	Use in this paper
HotpotQA	multi-hop QA over Wikipedia (Yang et al., 2018)	multi-fault injection
2Wiki	multi-hop QA over Wikipedia and Wikidata (Ho et al., 2020)	multi-fault injection; profile-conditioned QA
LongMemEval	long-term memory of chat assistants (Wu et al., 2025)	multi-fault injection; profile-conditioned QA (oracle mode)
MuSiQue	multi-hop questions composed from single-hop questions (Trivedi et al., 2022)	multi-fault injection; profile-conditioned QA
TravelPlanner	multi-constraint travel planning for language agents (Xie et al., 2024)	multi-fault injection; 85-case annotation study
TruthfulQA	questions that elicit common misconceptions (Lin et al., 2022)	profile-conditioned QA
PersonalWAB	profiles synthesized from real purchase, rating, and review histories (Cai et al., 2025)	recommendation from a 20-item slate
Amazon, Food.com	real product and recipe records	real-record corruption

Multi-hop QA. HotpotQA (Yang et al., 2018) contains about 113k Wikipedia-based questions whose answers require reasoning over multiple supporting documents, annotated with sentence-level supporting facts. 2WikiMultiHopQA (Ho et al., 2020) builds multi-hop questions from Wikipedia and Wikidata and supplies evidence paths for the reasoning steps. MuSiQue-Ans (Trivedi et al., 2022) contains about 25k questions of two to four hops composed from single-hop questions to require connected reasoning; MuSiQue-Full additionally includes unanswerable contrast questions.

Memory and planning. LongMemEval (Wu et al., 2025) tests the long-term memory of chat assistants with 500 questions covering information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention; we use its oracle setting, which retains only the evidence sessions. TravelPlanner (Xie et al., 2024) contains 1,225 travel-planning queries with a tool sandbox and hard and commonsense constraints; it supplies multi-fault items and the 85 naturally failed trajectories of the localization study.

Truthfulness and recommendation. TruthfulQA (Lin et al., 2022) contains 817 questions in 38 categories on which models tend to reproduce human misconceptions. PersonalWAB (Cai et al., 2025) provides user profiles synthesized from real purchase, rating, and review histories; our adaptation asks the agent to recommend one item from a 20-item slate, counts success when it picks the item the user interacted with, and passes item identifiers in the task metadata. Amazon product and Food.com recipe records supply the real-record corruption study, in which fields of real records are corrupted; corruption of Amazon records often removes every identity-defining field.

Profile construction. Profile-conditioned QA attaches a synthetic profile of 3–5 constraints drawn from 28 templates. About half the constraints are relevant and half are distractors, and six further TruthfulQA templates push toward popular misconceptions. Each 2Wiki, LongMemEval, and MuSiQue cell has 60 items drawn with replacement, and each TruthfulQA cell has 36 paired (question, seed) observations under three dataset seeds. Success requires a reference answer in the output and a passing profile predicate.

Fault injection. For $N = 2$–5 faults, each multi-fault item receives one context note suggesting a near-miss answer and $N-1$ flipped profile constraints, with hop corruptions at larger N. Different N use different item subsets, and $N \geq 4$ uses a separate injector. The nested-fault evaluation instead injects $N = 0$–4 faults into the same tasks with one injector, nesting faults as N grows.

E PASTA AND BASELINE CONFIGURATIONS

Table 6 lists every PASTA setting used in the reported runs; the provenance weights and the decay are fixed defaults and are not tuned per benchmark. Table 7 lists the baselines of the multi-hop harness. Gated methods use the verifier in the loop and stop once it passes; because the harness's verifier states the expected answer on the QA datasets, ungated baselines are shown for reference. Rows that error or fail to parse are dropped rather than scored as failures. Matched baselines in the single-model and human-annotation studies share PASTA's repair model, repair prompt, constraint-only feedback, and three-attempt maximum.

Table 6: PASTA configuration used in the reported runs.

Parameter	Value
Provenance weights w_ρ	model-inferred 1.0; world knowledge 0.6; external evidence 0.4; tool 0.25; profile 0.15
Unrecognized provenance	treated as model-inferred
Confidence floor	0.05
Proposals	three per generator call; default $M = 1$ retains the first
Edit cost	normalized descendant count δ_i; equal weight on q and δ
Repair attempts	at most 3; decay $\alpha = 0.1$
Temperatures	agent, extraction, re-execution 0.0; counterfactual generator 0.5
Max tokens	8192 for all PASTA calls
Verifier	benchmark rule checks; LLM verification layer disabled

F METHOD DETAILS AND PROOFS

Table 8 collects the notation.

Candidate construction and cost. $\mathrm{Anc}(a; G)$ and D_i are computed by breadth-first search over the dependency edges of G, upstream and downstream respectively, so constructing $\mathcal{C}$ and every δ_i takes $O(K(|A| + |E|))$ time, which is negligible next to the LLM calls. Because $a_f \in \mathcal{C}$, the candidate set is never empty.

Call accounting. An accepted trajectory costs two LLM calls: the agent call and the extraction call. A repair adds one counterfactual-generator call per candidate, each returning three replacements (Appendix P), and one re-execution call per attempt. A repaired task therefore costs $2 + K + k$ calls, where $k \leq 3$ is the number of attempts. The verifier is rule-based and issues no LLM calls.

Table 7: Answer-informed baselines. *Gated*: uses the verifier in the loop. *Calls*: LLM calls per task. ToT loses rows to parse errors from unescaped template braces (23 of 36 TruthfulQA×Mistral rows survive).

Method	Description	Gated	Calls
Direct	single agent call, $T = 0$	–	1
Self-Refine (SR)	one critique-and-revise round	–	2
Reflexion (Rfx)	verbal reflection on verifier violations, then retry (at most 2 retries)	✓	1–5
Full-Regen (FR)	discard the trajectory and resample once, $T = 0.7$	–	1
ReAct	thought–action–observation format; single call, no environment interaction	–	1
SelfCheckGPT (Manakul et al., 2023) (SChk)	5 samples, sentence-consistency check, regenerate if flagged	–	7–8
ToT	BFS with $k = 3$, beam 2, depth 4	–	∼49
VRP	re-prompt with the violated constraint (1 round)	✓	2
Iter-VRP	VRP with at most 2 retries	✓	2–3
PASTA	Algorithm 1	✓	2 / ∼6–8

Table 8: Notation.

Symbol	Meaning
x, P	task; user profile of stated facts and constraints
$\tau = (s_1, \ldots, s_T, y)$	trajectory with steps s_t (actions, observations, reasoning summaries) and final answer y
$V(P, x, \tau)$	rule-based verifier; returns step-aligned violations, $\emptyset$ if the trajectory is accepted
v, t_v	first reported violation; its step
$R(a)$	trajectory that keeps $\tau_{<t_a}$ verbatim and regenerates from a
$G = (A, E)$	provenance-typed belief graph with nodes A and dependency edges E
$a_i = (c_i, \rho_i, t_i, \ell_i)$	node: commitment, provenance, step, extraction confidence
a_f	final-decision node
B_v	nodes at the violated step, $\{a_i : t_i = t_v\}$
$\mathrm{Anc}(a; G)$, D_i	ancestors of a; descendants of a_i
$\mathcal{C}$, K	candidate set (Eq. 1); $K = \lvert\mathcal{C}\rvert$
w_ρ, π_i	provenance weight; prior of a_i (Eq. 2)
e_{ij}, M, r_i	predicted removal of the violation by proposal j; number of proposals; mean of e_{ij} over j
q_i, $q_i^{(r)}$	ranking score; score at attempt r (Eq. 4)
δ_i	normalized downstream cost (Eq. 3)
$\mathcal{T}$, $\mathcal{T}_r$	tried candidates; tried set before attempt r
α, Z_r	decay for tried candidates; renormalizer at attempt r (Proposition 3)
$a^\star$	selected restart

Properties. The following properties hold for the ranking, selection, and retry rules of Section 2.

Proposition 1 (Well-defined scores)**.** *Every prior satisfies* $0.0075 \le \pi_i \le 1$. *The first-round scores* $q^{(0)}$ *of Eq. 2, including the fallback, are nonnegative and sum to one over* $\mathcal{C}$.

Proof. Every weight lies in $[0.15, 1]$, and $\max(0.05, 1 - \ell_i) \in [0.05, 1]$ because $\ell_i \in [0, 1]$; hence $\pi_i \in [0.0075, 1]$. If some $r_k > 0$, the denominator $\sum_k \pi_k r_k$ is positive, and the scores are nonnegative and sum to one by construction. Otherwise, the fallback normalizes the strictly positive priors. □

Proposition 2 (Uniform scores select the cheapest candidate)**.** *If all untried candidates have equal scores, Eq. 3 selects an untried candidate of minimum cost, that is, one with the fewest descendants. In the first round this holds whenever* $q_i = 1/K$ *for every candidate.*

Proof. If $q_i = c$ for every untried a_i, then $\arg\max_i (c - \delta_i) = \arg\min_i \delta_i$, and δ_i increases with $\lvert D_i\rvert$. □

When the final-decision node has no descendants, it attains the minimum cost, so uniform scores recover the final-decision restart control up to ties.

Proposition 3 (Decay acts through renormalization). *Let $S_r = \sum_{k\in\mathcal{T}_r} q_k^{(0)}$ and $Z_r = 1-(1-\alpha)S_r$. Then $\alpha \le Z_r \le 1$, every untried candidate has $q_i^{(r)} = q_i^{(0)}/Z_r$, and the selection at attempt r is*

$$a^\star = \arg\max_{a_i\in\mathcal{C}\setminus\mathcal{T}_r} \big(q_i^{(0)} - Z_r\,\delta_i\big).$$

Consequently, the order of untried candidates by score is the same at every attempt, and the effective weight on cost, Z_r, never increases from one attempt to the next.

Proof. The denominator of Eq. 4 equals $\sum_{k\notin\mathcal{T}_r} q_k^{(0)} + \alpha\sum_{k\in\mathcal{T}_r} q_k^{(0)} = (1-S_r)+\alpha S_r = Z_r$, using $\sum_k q_k^{(0)} = 1$ (Proposition 1). Since $0 \le S_r \le 1$, we have $\alpha \le Z_r \le 1$, and in particular $Z_r > 0$. For untried a_i the numerator is $q_i^{(0)}$, which gives $q_i^{(r)} = q_i^{(0)}/Z_r$. Multiplying the objective $q_i^{(0)}/Z_r - \delta_i$ by $Z_r > 0$ preserves its argmax. Finally, $\mathcal{T}_r$ only grows with r, so S_r is nondecreasing and Z_r is nonincreasing. □

Proposition 4 (Termination). *PASTA performs at most $\min(3, K)$ regenerations, never regenerates twice from the same candidate, and always returns a trajectory.*

Proof. Each rejected attempt adds $a^\star$ to $\mathcal{T}$, the argmax ranges over $\mathcal{C}\setminus\mathcal{T}$, and the loop of Algorithm 1 ends after three attempts or once every candidate has been tried. Its last line returns the last attempt, and an accepted trajectory is returned directly. □

Scale of the trade-off. Because the scores sum to one while each δ_i lies in $(0, 1]$, spreading plausibility over many comparable candidates makes each q_i small (about $1/K$), and cost differences then decide the selection. Appendix I discusses this choice.

G Worked Example

We apply Eqs. 1–4 to the deployment example of Figure 1. The confidences and proposal outcomes are illustrative values chosen to show the mechanics; they are not outputs of our runs.

Graph, candidates, and costs. The nodes are a_1 ("budget $500/month") and a_2 ("processes payments"), both from the profile; the model-inferred commitments a_3 ("needs 99.99% availability"), a_4 ("multi-region database"), and a_5 ("three regions, global load balancer"); and the final-decision node $a_6 = a_f$ ("deployment plan"), at steps 1–6. Dependency edges run $a_2 \to a_3$, $a_3 \to a_4$, $a_3 \to a_5$, and $a_1, a_4, a_5 \to a_6$. The verifier reports that the plan exceeds the budget at step 6, so $B_v = \{a_6\}$ and Eq. 1 gives $\mathcal{C} = \{a_1, \ldots, a_6\}$. The descendant counts are $|D_i| = (1, 4, 3, 1, 1, 0)$, so $\max_k(|D_k| + 1) = 5$ and $\delta = (0.4, 1.0, 0.8, 0.4, 0.4, 0.2)$.

Priors. With the weights of Table 6 and confidences $\ell = (0.95, 0.90, 0.40, 0.70, 0.70, 0.80)$, the priors are $\pi = (0.0075, 0.015, 0.60, 0.30, 0.30, 0.20)$. The profile facts receive the smallest priors, as Principle 1 prescribes.

Scenario (i): only the inference and its source are predicted to repair. Suppose the generator ($M = 1$) predicts removal for a_2 and a_3 only. Then $q_2 = 0.015/0.615 = 0.024$ and $q_3 = 0.60/0.615 = 0.976$, and $a^\star = a_3$ with $q_3 - \delta_3 = 0.176$ (Table 9). PASTA keeps steps 1–2 and regenerates from the promoted inference, as in Figure 1. If the verifier rejects this attempt, Proposition 3 gives $Z_1 = 1 - 0.9 \times 0.976 = 0.122$, and the only other candidate with a positive score, a_2, rises to $q_2^{(1)} = 0.024/0.122 = 0.20$. Its objective, $0.20 - 1.0 = -0.80$, still trails that of the final-decision node, $0 - 0.2 = -0.20$, so the second attempt restarts from a_6.

Scenario (ii): several candidates are predicted to repair. Suppose instead that the generator predicts removal for a_3, a_4, and a_6, since a single-region database or smaller instance classes in the final plan could also fit the budget. Then $q = (0, 0, 0.545, 0.273, 0, 0.182)$. Although a_3 has the highest score, the cost term selects a_6 ($0.182 - 0.2 = -0.018$ against $0.545 - 0.8 = -0.255$), an instance of $a^\star \neq \arg\max_i q_i$ (Section 2). This restart is admissible if the regenerated plan fits the budget, yet it keeps the unsupported belief in the prefix, which is why we measure reuse on follow-up tasks (Section 4.3).

Uniform scores. If every candidate had $q_i = 1/6$, Proposition 2 would select a_6, the cheapest candidate, which reproduces the final-decision restart control.

Table 9: Eqs. 1–4 on the deployment example of Figure 1, with illustrative values. e_i: predicted removal of the violation ($M = 1$). The selected restart in each scenario is in bold.

						Scenario (i)			Scenario (ii)		
Node	Commitment	Provenance	ℓ_i	π_i	δ_i	e_i	q_i	$q_i - \delta_i$	e_i	q_i	$q_i - \delta_i$
a_1	budget $500/month	profile	0.95	0.0075	0.4	0	0	-0.400	0	0	-0.400
a_2	processes payments	profile	0.90	0.015	1.0	1	0.024	-0.976	0	0	-1.000
a_3	needs 99.99% availability	model-inferred	0.40	0.60	0.8	1	0.976	$\mathbf{+0.176}$	1	0.545	-0.255
a_4	multi-region database	model-inferred	0.70	0.30	0.4	0	0	-0.400	1	0.273	-0.127
a_5	three regions, global LB	model-inferred	0.70	0.30	0.4	0	0	-0.400	0	0	-0.400
a_6	deployment plan (a_f)	model-inferred	0.80	0.20	0.2	0	0	-0.200	1	0.182	$\mathbf{-0.018}$

H Reference Implementation

The listing restates Eqs. 1–4 and Algorithm 1 in Python. It covers the selection logic only: the LLM-backed steps (extraction, counterfactual proposals, and regeneration, specified by the prompts in Appendix P) and the rule-based verifier are passed in as functions. Run on the graph of Appendix G, it reproduces the selections of both scenarios, the retry to a_6, and the minimum-cost choice under uniform scores.

```python
# Reference implementation of PASTA's restart selection (Eqs. 1-4, Alg. 1).
# LLM-backed steps (extraction, counterfactual proposals, regeneration) and
# the rule-based verifier are passed in as functions. Ties are broken by
# step order.
from dataclasses import dataclass, field

PROVENANCE_WEIGHTS = {
    "model_inferred": 1.0,
    "world_knowledge": 0.6,
    "external_evidence": 0.4,
    "tool": 0.25,
    "profile": 0.15,
}
CONFIDENCE_FLOOR = 0.05
DECAY = 0.1  # alpha in Eq. 4
MAX_ATTEMPTS = 3

@dataclass
class Node:
    node_id: str
    commitment: str
    provenance: str
    step: int
    confidence: float  # extraction confidence
    parents: list = field(default_factory=list)  # upstream dependencies

def _closure(start, neighbors):
    seen, stack = set(), list(start)
    while stack:
        n = stack.pop()
        if n not in seen:
            seen.add(n)
            stack.extend(neighbors(n))
    return seen

def ancestors(graph, node_id):
    return _closure(graph[node_id].parents, lambda n: graph[n].parents)
```

```python
def descendants(graph, node_id):
    children = {n: [] for n in graph}
    for n, node in graph.items():
        for p in node.parents:
            children[p].append(n)
    return _closure(children[node_id], lambda n: children[n])

def candidates(graph, violation_step, final_id):
    # Eq. 1: beliefs at the violated step, their ancestors, and a_f.
    b_v = [n for n, node in graph.items() if node.step == violation_step]
    cands = set(b_v) | {final_id}
    for n in b_v:
        cands |= ancestors(graph, n)
    return sorted(cands, key=lambda n: graph[n].step)

def prior(node):
    w = PROVENANCE_WEIGHTS.get(node.provenance,
                               PROVENANCE_WEIGHTS["model_inferred"])
    return w * max(CONFIDENCE_FLOOR, 1.0 - node.confidence)

def first_round_scores(graph, cands, indicators):
    # Eq. 2; indicators[n] lists e_nj over the M proposals for candidate n.
    pi = {n: prior(graph[n]) for n in cands}
    r = {n: sum(indicators[n]) / len(indicators[n]) for n in cands}
    z = sum(pi[n] * r[n] for n in cands)
    if z > 0:
        return {n: pi[n] * r[n] / z for n in cands}
    total = sum(pi.values())
    return {n: pi[n] / total for n in cands}  # fallback

def costs(graph, cands):
    # Eq. 3, left: normalized downstream cost delta.
    size = {n: len(descendants(graph, n)) + 1 for n in cands}
    largest = max(size.values())
    return {n: size[n] / largest for n in cands}

def retry_scores(q0, tried, alpha=DECAY):
    # Eq. 4: down-weight tried candidates, then renormalize.
    raw = {n: q * (alpha if n in tried else 1.0) for n, q in q0.items()}
    z = sum(raw.values())
    return {n: v / z for n, v in raw.items()}

def select(q, delta, tried):
    # Eq. 3, right: argmax of q - delta over untried candidates.
    untried = [n for n in q if n not in tried]
    return max(untried, key=lambda n: q[n] - delta[n])

def pasta(graph, trajectory, final_id, verify, propose, regenerate):
    # Algorithm 1 after extraction. verify(traj) returns a list of
    # (step, message) violations; propose(node) returns the M indicators;
    # regenerate(node) keeps the prefix before node.step and regenerates.
    violations = verify(trajectory)
    if not violations:
        return trajectory, None
    cands = candidates(graph, violations[0][0], final_id)
    q0 = first_round_scores(graph, cands,
                            {n: propose(graph[n]) for n in cands})
    delta = costs(graph, cands)
    tried, attempt = set(), trajectory
    for _ in range(MAX_ATTEMPTS):
        if len(tried) == len(cands):
            break
        a_star = select(retry_scores(q0, tried), delta, tried)
        attempt = regenerate(graph[a_star])
        if not verify(attempt):
            return attempt, a_star
        tried.add(a_star)
    return attempt, None  # last attempt, scored as a failure
```

I Design Choices

Provenance weights. The weights order provenance by suspicion, following Principle 1: model inferences are most suspect and profile facts least, with world knowledge, external evidence, and tool observations in between. Unrecognized provenance defaults to model-inferred, so an unlabeled commitment is treated as maximally suspect. The weights are fixed defaults and are not tuned per benchmark; Tables 1 and 12 report the effect of replacing them with a uniform prior.

Confidence floor. Without the floor, a commitment extracted with confidence 1 would receive a zero prior and no plausibility score. The floor of 0.05 keeps every prior positive (Proposition 1).

One proposal. The generator returns three replacements per call, and the default $M = 1$ retains the first. In the proposal-count sweep (Table 12b), $M = 1$ matches or exceeds $M = 3$ in every column.

Equal weights on plausibility and cost. Eq. 3 weights q_i and δ_i equally. Because the scores sum to one, the balance between the two terms depends on the number of candidates (Appendix F); on stored first-round decisions, the cost term changes the selected restart in 30% of multi-candidate cases.

Regenerating from the selected node. Regeneration receives the identity of the selected node but not the proposed replacement text. The repair therefore acts through the choice of restart point and the preserved prefix, and the proposals serve only to score candidates.

Restarting from the original trajectory. Every attempt restarts from the original trajectory rather than from the previous attempt, so a rejected attempt cannot carry its own errors into the next one.

Candidate set. Ancestors admit a promoted inference that entered before the violated step, and the final-decision node admits a repair at the decision itself; when that node has no descendants, it is the cheapest candidate (Proposition 2).

First violation. PASTA builds candidates from the first reported violation and accepts a repair only when the verifier reports no violations at all.

J Evaluation and Annotation Protocols

Table 10 lists the criteria for coding inference promotion.

Table 10: Criteria for inference promotion, illustrated with the deployment example.

Element	Criterion	Deployment example
Source fact	a fact or constraint stated in the profile	"processes payments"
Unsupported inference	a commitment derived from the source fact that the profile does not state or support	"needs 99.99% availability"
Promotion	the inference becomes a constraint on later choices	every component must survive a regional outage
Downstream use	later steps rely on the promoted constraint	database, regions, and load balancer chosen around it

Answer-blind serialization. Every model-visible serialization in the answer-blind protocol excludes gold answers and target-bearing fields, including the inputs to initial generation, belief extraction, proposal scoring, feedback, and regeneration. References are used only for independent final scoring.

Natural-failure study. All arms start from the same failed trajectories, use a shared repair prompt and repair model, receive constraint-only feedback, and have a maximum of three repair attempts. Gold answers and target-bearing metadata are excluded from all model-visible stages, and correctness is assessed with blinded judgments. The strict promotion code requires evidence for all four criteria of Table 10: a supported source fact, an unsupported commitment, promotion into an operative constraint, and downstream use.

Follow-up evaluation. The same 150 confirmed promotion cases are used in every arm. After the original task is repaired, the retained state is preserved exactly as the agent would retain it, and a new task is issued on which the unsupported belief could lead to another error. Reuse is measured across all 150 cases in each arm.

Earlier 85-case annotation. Two independent annotators, blinded to method identity, labeled a seven-way fault category and the earliest faulty step of each of 85 naturally failed TravelPlanner trajectories. Consensus follows a preregistered rule; 12 items were adjudicated by the authors. Each annotator also received five attention checks built from one template. The stored answer key (category 1 at step 5) was corrected, after annotations were reviewed, to category 2 at step 4: step 4 of each check introduces a restaurant that appears nowhere in the task, profile, or earlier steps, which the codebook labels a faulty assumption. Both annotators gave this answer on all five checks. On the 73 items resolved by the consensus rule, without the 12 author-adjudicated ones, the top-1 counts are 7, 9, and 12 for the PASTA restart point, a random ancestor, and the most recent step, and the MRRs are 0.21 [0.15, 0.29], 0.22 [0.16, 0.28], and 0.29 [0.22, 0.38], so the orderings of Table 3c are unchanged.

Profile re-draws. Each profile is re-drawn 100 times with template substitution and paraphrase. An automatic paraphrase check audited 30 constraints, and the confirmatory tier keeps only the template types whose paraphrases all passed review.

Synthetic graphs. The sanity check in §4.3 uses 60 synthetic graphs with a known faulty node, a deterministic verifier, and a rule-based repair sampler, with no LLM calls, in a single-fault mode and in a multi-valid mode in which several nodes can each resolve the violation.

K Additional Results

This appendix reports results that Sections 4.4 and 4.5 summarize. The flat-feedback controls come from the final configuration, and the cost, component, and scale analyses come from the earlier configuration without sequential retry (Appendix B).

Table 11: Earlier answer-informed cost: recorded tokens per task ($\times 10^3$) and LLM calls per task. ‡Agent and extraction calls only. §Accepted / repaired tasks.

Dataset	Direct	Rfx	SChk	ToT	PASTA‡
TruthfulQA	0.8	2.4	5.9	34.3	3.0
2Wiki	1.3	2.8	10.3	58.3	4.9
LongMemEval	1.8	6.3	12.2	69.1	5.0
MuSiQue	1.4	4.0	11.3	71.9	5.2
Calls/task	1	1–5	7–8	∼49	2 / 6–8§

Flat-feedback controls. In separate paired runs against VRP and Iter-VRP, which re-prompt with the violated constraint as text without a dependency graph, PASTA reaches 88.9% against Iter-VRP's 73.3% on LongMemEval with Llama 3.1 8B (task-level sign test 14 vs. 2, $p = 0.004$) and 99.4% against 95.0% on 2Wiki with Mistral NeMo 12B ($p = 0.63$); VRP's revision prompt omits the task metadata that PASTA's prompts include. The answer-blind counterpart is matched flat feedback in Table 1.

Cost. In the earlier answer-informed runs (Table 11), PASTA's recorded agent and extraction tokens are above Reflexion's on three of four datasets, below SelfCheckGPT's, and an order of magnitude below ToT's, with median wall-clock times of 7 s on accepted and 28–32 s on repaired tasks. Because proposal and regeneration calls are not recorded, these counts are lower bounds on PASTA's repair cost.

Components. In the earlier configuration (Table 12a), removing the graph, the provenance prior, counterfactual proposals, or the edit cost lowers mean success by 2.8, 3.1, 1.5, and 0.6 pp; on LongMemEval, removing proposals or the edit cost raises success by 3.3 pp and the uniform prior by 1.7 pp. In the proposal-count sweep (Table 12b), $M = 1$ matches or exceeds $M = 3$ in every column; nominal $M = 3$ and $M = 5$ retain the same three-proposal batch, so their differences reflect run variation.

Table 12: Earlier-configuration component analyses on four cells ($n = 36$ for TruthfulQA, 60 otherwise). (a) Change in success (pp) when one component is removed, relative to $M = 1$. (b) Success (%) by nominal proposal count M; nominal $M = 3$ and $M = 5$ retain the same three-proposal batch.

	TruthfulQA Mistral	TruthfulQA Qwen	2Wiki Mistral	LongMemEval Llama 3.1 8B	Mean
(a) Removing one component (pp)					
No graph (flat candidates)	−8.3	−2.8	0.0	0.0	−2.8
Uniform provenance prior	−5.6	−8.3	0.0	+1.7	−3.1
No counterfactual proposals	−2.8	−8.3	+1.7	+3.3	−1.5
No edit cost	−2.8	−2.8	0.0	+3.3	−0.6
(b) Proposal count (% success)					
$M = 1$ (default, reference)	86.1	77.8	96.7	78.3	
$M = 3$	80.6	66.7	96.7	75.0	
$M = 5$	72.2	66.7	96.7	78.3	

Table 13: Model scale in the earlier answer-informed configuration: success (%) as Direct / Reflexion / PASTA; bold marks the higher of Reflexion and PASTA.

Model	TruthfulQA	2Wiki	LongMemEval	MuSiQue
Qwen 2.5 7B	63.9 / 69.4 / **77.8**	81.7 / **91.7** / 90.0	70.0 / **76.7** / 71.7	80.0 / 83.3 / **85.0**
DeepSeek-V3	38.9 / **83.3** / 41.7	93.2 / **96.6** / 93.2	36.2 / 43.1 / **45.6**	56.7 / **71.7** / 61.7
Gemini-2.5-Flash	41.7 / **75.0** / 44.4	80.0 / **90.0** / 83.3	28.3 / **33.3** / 30.0	51.7 / **68.3** / 60.0

Model scale. With Qwen 2.5 7B, PASTA leads Reflexion on TruthfulQA and MuSiQue; with DeepSeek-V3 and Gemini-2.5-Flash, Reflexion leads in 7 of 8 cells (Table 13).

L Robustness Details

Profile re-draws. Over 100 re-draws of every profile with template substitution and paraphrase (five datasets, two LLMs), PASTA is ahead of Reflexion in 92 draws, with a mean effect of +2.07 pp (task-clustered 95% CI [+1.06, +3.18]) and a bottom-decile mean of -0.34 pp over 40,800 outcomes. An automatic paraphrase check preserved meaning in 17 of 30 audited constraints, so the preregistered confirmatory tier keeps only template types whose paraphrases all passed review (2,944 paired outcomes); there, the mean effect is +1.59 pp and the bottom-decile mean is -7.21 pp, which does not meet the preregistered bottom-decile criterion.

Profile dependence. Success is 12 pp higher with the correct profile than with no profile, and valid preference swaps change PASTA's choices appropriately in 90% of cases. With profile templates and one domain held out, PASTA's advantage is +4 pp, against +6 pp in development.

Graph audit and verification. The blind graph audit reports provenance macro-F1 of 0.88, dependency precision of 0.85, and usable repair-candidate coverage of 0.93. Comparing in-loop acceptance with independent assessment, 3% of verifier-accepted outputs fail the independent check.

M Statistical Procedures

Exact McNemar and sign tests. For paired binary outcomes, let b count pairs in which only PASTA succeeds and c pairs in which only the comparator succeeds. The exact two-sided test treats $\min(b, c)$ as a binomial draw with $b + c$ trials and success probability $1/2$:

$$p = \min\Big(1,\ 2\sum_{i=0}^{\min(b,c)} \binom{b+c}{i} 2^{-(b+c)}\Big).$$

The 300-case comparison has $b = 51$ and $c = 15$, which gives $p = 1.01 \times 10^{-5}$; the pooled PersonalWAB comparison has $b = 14$ and $c = 3$, which gives $p = 0.013$. Task-level sign tests apply the same computation to the tasks on which the two methods differ.

Holm correction. Within each baseline family of m comparisons, the ordered p-values $p_{(1)} \leq \cdots \leq p_{(m)}$ are tested in sequence, and the j-th hypothesis is rejected only if $p_{(k)} \leq 0.05/(m-k+1)$ for every $k \leq j$. Against Reflexion, the smallest per-cell p is $0.016 > 0.05/12$ in the profile-conditioned matrix ($m = 12$) and $0.031 > 0.05/4$ on PersonalWAB ($m = 4$), so no cell is rejected in either family.

Differences and pooled rates. Differences are computed from unrounded rates. One paired outcome moves a rate by $1/60 = 1.67$ pp in a 60-item cell and by $1/36 = 2.78$ pp in a TruthfulQA cell, so every Δ in Table 4 is a multiple of one of these steps. The counts behind that table's Reflexion and PASTA columns (Reflexion/PASTA) are, for Llama 3.1 8B, Llama 3.3 70B, and Mistral NeMo 12B in turn: TruthfulQA ($n = 36$) 34/35, 33/35, and 29/32; 2Wiki ($n = 60$) 56/58, 58/59, and 57/60; LongMemEval ($n = 60$) 46/53, 42/44, and 44/49; MuSiQue ($n = 60$) 54/57, 58/59, and 51/54. Their totals give the pooled 562/648 and 595/648.

Intervals and localization metrics. The answer-blind comparison uses paired, cluster-aware uncertainty over repeated tasks or users; profile re-draw intervals are clustered by task; and localization MRR intervals resample items with replacement. Top-1 counts trajectories in which a selector's first choice is the annotated earliest faulty step, ± 1 step counts first choices within one step of it, and MRR averages the reciprocal rank of the annotated step in each selector's ordering.

Code. The listing below implements both tests; it reproduces every p-value reported in this appendix.

```python
# Exact McNemar test and Holm correction used in the statistical analyses.
from math import comb

def mcnemar_exact(b, c):
    # b: pairs where only PASTA succeeds; c: pairs where only the comparator does.
    n, k = b + c, min(b, c)
    if n == 0:
        return 1.0
    return min(1.0, 2 * sum(comb(n, i) for i in range(k + 1)) / 2 ** n)

def holm(pvalues, alpha=0.05):
    # Step-down procedure; returns one reject flag per hypothesis.
    order = sorted(range(len(pvalues)), key=lambda i: pvalues[i])
    reject = [False] * len(pvalues)
    for rank, i in enumerate(order):
        if pvalues[i] > alpha / (len(pvalues) - rank):
            break
        reject[i] = True
    return reject
```

N Extended Related Work

Table 14 positions PASTA against the closest methods.

Self-correction and search. Self-Refine (Madaan et al., 2023) has one model generate an output, critique it, and revise it iteratively without additional training. Reflexion (Shinn et al., 2023) converts task feedback into verbal reflections stored in episodic memory and conditions later trials on them. CRITIC (Gou et al., 2024) verifies and revises outputs with external tools such as search engines and code interpreters. Tree of Thoughts (Yao et al., 2023a) searches over intermediate thoughts with self-evaluation and explicitly supports backtracking; LATS (Zhou et al., 2024) combines Monte Carlo tree search with language-model value estimates, reflection, and environment feedback; RAP (Hao et al., 2023) uses a language model as both world model and reasoning agent; and Graph of Thoughts (Besta et al., 2024) organizes thoughts as a graph with aggregation and refinement. Huang et al. (2024) find that intrinsic self-correction without external feedback can fail to improve reasoning, and Kamoi et al. (2024) survey the conditions under which self-correction succeeds. These methods

Table 14: Selected related mechanisms and their relation to PASTA. This is a comparison of the cited formulations, not a claim that other methods are incapable of using provenance or selecting earlier states.

Method	Primary mechanism	Relation to PASTA's restart objective
Self-Refine (Madaan et al., 2023)	Iterative self-feedback and output revision	Revision does not itself establish which user-relative belief should determine a restart.
Reflexion (Shinn et al., 2023)	Verbal reflection retained across trials	Can learn from failures; differs from an explicit provenance–cost restart score.
ToT / LATS (Yao et al., 2023a; Zhou et al., 2024)	Thought search / Monte Carlo tree search	Already explore alternatives; ToT explicitly supports backtracking.
RAFFLES (Zhu et al., 2026a)	Judge–evaluator fault attribution	Evaluates fault identification rather than PASTA's restart-selection objective.
EDGE (Hou et al., 2026)	Error dependency graph and multi-error detection	Uses repair interventions to validate dependencies, not an evaluated end-to-end recovery policy.
CausalFlow (Bonagiri et al., 2026)	Counterfactual intervention and repair	Uses downstream re-execution and repair minimality.
Dependency-guided rollback (Yu et al., 2026a)	Memory lineage, independent support, selective replay	Starts from faulty-memory identifiers supplied by an upstream detector.
RIR (Yu et al., 2026b)	Diagnosis, checkpoint selection, reflection memory	Already separates coarse causal localization from selection of the latest viable restore state.
PASTA (ours)	Belief graph and suffix regeneration	Combines user-relative provenance, predicted repair, and downstream cost after a verifier-reported violation.

already revise decisions or explore alternative trajectories. PASTA instead focuses on the specific decision rule of combining user-relative provenance, predicted repair, and downstream cost to select a belief-level restart.

Error localization. Tyen et al. (2024) show that supplying ground-truth mistake locations improves downstream performance on five tested reasoning tasks, while ProcessBench (Zheng et al., 2025) evaluates identification of the earliest erroneous step in mathematical reasoning. For agents, Cemri et al. (2025) develop a taxonomy of failure modes, and TRAIL (Deshpande et al., 2025) and TraceElephant (Chen et al., 2026) provide benchmarks for trace-based failure attribution. EDGE (Hou et al., 2026), StepFinder (Zhu et al., 2026b), RAFFLES (Zhu et al., 2026a), Abduct–Act–Predict (West et al., 2025), VerifyMAS (Qiao et al., 2026), and the causal-inference framework of Ma et al. (2025) infer error categories, responsible agents, or error locations using different forms of evidence. VerifyMAS verifies failure hypotheses before attributing entailed errors to agents; it is not a program-analysis method. Separately, Theorem-of-Thought (Abdaljalil et al., 2025) combines abductive, deductive, and inductive reasoning agents for general reasoning tasks, rather than constituting an agent-failure-attribution benchmark. Attribution and recovery are related but distinct objectives: our localization study (§4.3) compares PASTA's selected restarts with annotated fault locations without treating agreement as the only measure of repair quality.

Rollback and repair. CausalFlow (Bonagiri et al., 2026) tests counterfactual interventions at candidate steps, re-executes downstream computation, and selects successful repairs using a minimality criterion. GA-Rollback (Li et al., 2025), WebRollback (Zhang et al., 2026), and AgentRewind (Zhuang et al., 2026) provide mechanisms for restoring earlier agent or environment states. RIR (Yu et al., 2026b) jointly decides when to roll back, where to restore, and what information to retain. Its own reviewer generates a failure diagnosis; a coarse selector identifies a causally relevant interval, and a fine selector chooses the latest checkpoint in that interval whose restored state escapes the diagnosed obstruction. Reflection memory distilled from the abandoned suffix survives the rollback. Dependency-guided rollback repair (Yu et al., 2026a) takes faulty-memory identifiers from an up-

stream detector, builds a typed memory-to-action graph from runtime provenance, preserves state with independent trusted support, and replays answer-relevant affected computation. Its recurrence metric re-runs the last user input and is computed over successfully recovered cases. MemoRepair (Zhao et al., 2026) repairs cascades in agent memory. PASTA does not require a supplied faulty-node diagnosis, but does require a verifier-reported violation; its particular ranking combines user-relative provenance, predicted repair, and downstream cost, and its follow-up evaluation uses a new task over all cases.

Diagnosis and program repair. Consistency-based diagnosis (Reiter, 1987) derives minimal diagnoses from conflicts between a system model and observations, the General Diagnostic Engine (de Kleer & Williams, 1987) handles multiple faults and selects measurements, and Console & Torasso (1991) relate consistency-based and abductive definitions of diagnosis. Spectrum-based fault localization ranks program elements using execution spectra from failing and passing tests (Jones & Harrold, 2005; Abreu et al., 2006; Wong et al., 2016), and delta debugging (Zeller & Hildebrandt, 2002) isolates failure-inducing inputs through repeated tests. GenProg (Le Goues et al., 2012) searches for program variants validated by test suites, using fault localization to guide the patch search. These are precedents for separating diagnostic evidence from intervention validation, not evidence that all such methods implement PASTA's restart objective. PASTA constructs its candidates from a failed trajectory and a belief graph, then validates regenerated suffixes with its verifier.

Personalization. LaMP (Salemi et al., 2024) benchmarks personalized classification and generation conditioned on user profiles, PrefEval (Zhao et al., 2025) evaluates preference following, including under long conversational contexts, and Sun et al. (2026) study hallucinations induced by personalization. These works measure or mitigate profile-related errors; PASTA targets the specific mechanism of inference promotion and evaluates repair after a violation is reported.

O GLOSSARY

Inference promotion. Promoting an unsupported inference about the user into an operative constraint that later steps build on (criteria in Table 10).

Commitment. A proposition the agent acts on; a node of the belief graph.

Provenance. The source of a commitment relative to the profile: profile, tool, external evidence, world knowledge, or model inference.

Admissible restart. A commitment from which regeneration, keeping the preceding prefix, yields a verifier-accepted trajectory.

Root cause. The commitment at which the fault originated; it need not be admissible or selected.

Answer-blind. A protocol in which no model-visible stage sees gold answers or target-bearing metadata.

Answer-informed. A protocol in which the verifier's feedback states the expected answer.

Gated method. A method that consults the verifier in the loop and stops once it passes.

Matched baseline. A baseline that shares PASTA's repair model, prompt, feedback, and attempt budget.

Belief reuse. Reliance on the originally promoted, unsupported belief when solving a follow-up task after repair.

Full-output retry. Regenerating the whole output from feedback on the final violation.

Final-decision restart. Regenerating from the final-decision node.

Annotated-root restart. Regenerating from the annotated earliest fault, a privileged diagnostic.

P PROMPTS

We list the prompts verbatim. Braces denote runtime fields. In the re-execution prompt, the field labelled "Repaired assumption node" identifies the selected restart node; as stated in §2, the proposed replacement text is not passed to regeneration.

Belief-graph extraction.

```
Extract assumption nodes and their dependencies.
Profile: {profile}
Task: {task}
Trajectory: {observable_steps}
Classify each node by provenance: profile, tool, external_evidence,
model_inferred, world_knowledge. Profile-derived nodes must cite
profile_span_ids.
Edge types: supports, depends_on, contradicts, downstream_of.
Mark the final-decision node that most directly supports the answer.
Do not include hidden chain-of-thought.
Return JSON.
```

Counterfactual generator.

```
You are testing whether an assumption is the root cause of a
profile-grounded error. Given a user profile, a task, the agent's
original answer, the violated constraint, and one assumption from the
agent's reasoning, generate 3 counterfactual repairs. For each, provide
a minimal replacement and state whether the violation is expected to be
removed and whether the output improves.
Return JSON.
```

Downstream re-execution.

```
Inputs: Task; User profile; Repaired assumption node; Downstream nodes
to regenerate; Original trajectory prefix; Available tool observations.
Regenerate the downstream trajectory from the repaired assumption
onward. Keep upstream decisions unless they depend on the repaired
node. Use available tool observations exactly as given. Do not invent
new external facts. Satisfy the user profile. Return only the final
answer.
```

Q Infrastructure, Caching, and Sampling

Providers. All LLM calls use hosted APIs, and no local GPU inference is used. Llama 3.3 70B runs through Together AI, and Llama 3.1 8B, Mistral NeMo 12B, and Qwen 2.5 7B run through OpenRouter, with a second Qwen 2.5 7B run through Together AI for PersonalWAB.

Caching. Every model call is cached by a SHA-256 hash of (provider, model, temperature, system prompt, user prompt, max tokens, seed). Cache hits return identical bytes.

Sampling and pairing. Example indices are drawn with `Random` seeded by $42 \times 1{,}000{,}037 + idx$. Pairing across methods is checked by 28 automated unit tests before results are committed.

Run windows. Baseline runs for the primary answer-informed matrix date from March–May 2026, and PASTA runs from September 2026.