

Grounding First: Auditing a Structured Recovery Wrapper for Local Open-Weight Reflexion Agents

Nimay Ballal and Badri Prasad V R

Department of Computer Science and Engineering

PES University

Karnataka, India

peslug24cs301@stu.pes.edu

badriprasad@pes.edu

Abstract

We audit how action interfaces and recovery mechanisms shape the evaluation of local open-weight Reflexion agents on 134 ALFWorld valid-unseen tasks. The analysis covers six primary admissible-action conditions and historical free-text runs across three model deployments. We organize the audit around four implementation contracts: action execution, recovery triggering, treatment changes, and outcome construction. Historical free-text runs show substantial invalid-output fallback, while designated admissible-parser fallback remains below 1.1 percent; this contrast does not by itself establish parser faithfulness. Structured Recovery Prompting (SRP) combines a recovery prompt with counter resets and relaxed termination limits. Its implementation-reported task-success differences range from 0.00 to +6.72 percentage points and are smaller than the baseline contrast between the two Qwen model sizes. The available evidence supports descriptive comparisons rather than an isolated recovery-prompt effect. The audit identifies incomplete repeat detection and termination-based success labeling, highlighting the importance of trigger coverage and outcome definitions when interpreting recovery results. The contribution is a focused diagnostic audit that connects aggregate performance to implementation behavior and provides concrete checks for evaluating recovery in the studied agents.

1 Introduction

Language agents turn generated text into environment transitions. ReAct and Reflexion combine reasoning, action, observation, and across-trial verbal memory (Yao et al., 2023; Shinn et al., 2023); in ALFWorld, each output is an executable household command (Shridhar et al., 2021). If the executor replaces an invalid output with `look`, however, the resulting trajectory no longer represents the model’s intended action, and later reflection operates on an altered history.

We audit that failure surface for local open-weight Reflexion agents. **Action-interface invalidity** is a single-step boundary event in which raw output is sanitized or replaced. **Trajectory stagnation** is a within-trial executable loop, such as repeated exploration without state change. The distinction is operational: it identifies where the wrapper records failure, not whether the underlying cause is formatting, reasoning, or both.

The wrapper first grounds decisions in the environment-provided action set $\mathcal{A}_t$ by requesting one integer index. It then optionally applies Structured Recovery Prompting (SRP): at an exact trailing-count threshold, the code prepares a block containing recent failures, visited receptacles, failed actions, and the next admissible list. Consuming that block also resets counters and persistently raises repeated-action and no-progress exhaustion limits. SRP is therefore a composite prompt-and-policy treatment, not a prompt-only intervention.

We ask three questions. **RQ1:** How much action substitution is recorded in the historical free-text and primary admissible pipelines? **RQ2:** What does the implemented recovery condition actually change and detect? **RQ3:** Which empirical conclusions survive an audit of the available code, summaries, traces, and missing artifacts?

Historical free-text baselines record invalid fallback on 18.8% of Qwen2.5-7B steps and 66.2% of Llama-3.1-8B steps; primary admissible parser fallback is below 1.1%. The recovered free-text count covers invalid prefixes and empty outputs, excluding timeouts and whitespace-only repairs; it is not a complete invalid-action measure. Section 3.2 details its scope and missing run-level provenance. Admissible fallback does not establish that other parsing routes preserve model intent.

In the primary table, the 57.5-point Qwen2.5-14B versus Qwen2.5-7B baseline contrast is larger than any within-model SRP difference. SRP is +6.72 points for Qwen2.5-7B, +2.99 for Qwen2.5-

14B, and 0.00 for Llama-3.1-8B, but these are descriptive because matched task outcomes are unavailable and the treatment is bundled.

The contribution is a code-and-artifact-grounded four-contract audit: the *action contract* from raw output to executed command, the *trigger contract* from trace predicates to event logging, the *treatment contract* covering every state or policy change after an event, and the *outcome contract* from environment signals to reported task success. We instantiate these contracts by formalizing the active trigger $\theta > 0 \wedge c_t = \theta$, separating invalid output and admissible-parser fallback, and distinguishing trigger-associated success from causal rescue. The audit identifies a repeat-detector defect, a done/won validity gap, incomplete structured artifacts, and an unexplained runtime anomaly. These findings bound the claim to the audited wrapper rather than a general recovery algorithm.

2 Background and Related Work

ALFWorld as a controlled long-horizon environment. ALFWorld aligns TextWorld-style text interaction with ALFRED-style embodied household tasks (Côté et al., 2019; Shridhar et al., 2020, 2021). The interface exposes step-level textual observations and actions, enabling trajectory-level inspection when the corresponding logs are retained. Its task families include picking and placing objects, examining objects under light, cleaning before placement, heating before placement, cooling before placement, and placing two instances together. These tasks stress a model’s ability to maintain a partial state estimate, preserve subgoals across object locations, and issue valid executor commands for many steps.

Analytical agent evaluation. AgentBoard argues that final success rates alone reveal little about multi-turn agent behavior and supplements them with progress, grounding, subskill, and trajectory analyses (Ma et al., 2024). Our audit shares that diagnostic motivation but targets a different layer: whether an existing wrapper’s parser, trigger, stopping rule, logger, and outcome label support the interpretation attached to its aggregate metrics.

ReAct and Reflexion. ReAct interleaves reasoning traces with actions and observations, creating a direct interface between language reasoning and environment transition feedback (Yao et al., 2023). Reflexion adds verbal memory across trials, allowing

failed traces to be converted into natural-language feedback for the next attempt (Shinn et al., 2023). Positive Experience Reflection, implemented as Sweet&Sour, adds positive subgoal reflections and managed short- and long-term memory, primarily evaluated in ScienceWorld with supplementary ALFWorld results (Lippmann et al., 2025). This study does not propose positive reflection or a new memory system. It keeps the inherited across-trial Reflexion mechanism fixed and audits action-interface handling plus a within-trial recovery wrapper. Each task can run for three trials; solved tasks are skipped in later trials, failed tasks receive generated reflections, and the memory cap is three reflections per task.

Failure recovery in agent execution. Recent REALM work studies recovery in complementary settings. ToolReflection uses self-generated errors and real-time API feedback to train models to correct tool calls (Polyakov et al., 2025). Conditional Multi-Stage Failure Recovery uses zero-shot chain prompting across execution-time and post-execution stages for embodied agents in TEACH (Farag et al., 2025). The present study differs in both evidence and objective: it audits an untrained local ALFWorld wrapper whose trigger, parser, and termination policy can be inspected, and it treats implementation confounds as part of the result rather than attributing aggregate gains to recovery alone.

Closed-source API agents and local open-weight agents. Published Reflexion results use high-capability closed-source API models for ALFWorld (Shinn et al., 2023). The project assets instead use local Qwen2.5 and Llama 3.1 deployments through Ollama’s OpenAI-compatible endpoint (Qwen et al., 2024; Grattafiori et al., 2024; Ollama, 2024). The highest reported local condition in the supplied primary summary is Qwen2.5-14B+SRP at 73.9% TSR with three trials. Because model class, trial budget, prompts, and runtime differ, this paper does not make a numerical parity claim against the original API-based Reflexion experiments.

Runtime wrappers versus heavier repair. Some agent-repair methods use substantially heavier within-episode machinery. ReflexGrad routes between continuous refinement and causal diagnosis and reports multi-seed, compute-matched comparisons (Kadu and Krishnan, 2025). The present wrapper is simpler and should not be read as a com-

peting state-of-the-art method. Its recovery prompt sees recent actions and observations, visited receptacles, failed actions, and admissible commands; it has no explicit symbolic inventory or subgoal state. The contribution here is the audit of this lightweight implementation, including where its detector and evidence fail.

3 Problem Formulation and Methodology

3.1 ALFWorld as a POMDP

We treat each ALFWorld task as a finite-horizon partially observable Markov decision process (Kaelbling et al., 1998),

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, \Omega, R, \rho_0, H \rangle, \quad H = 49. \quad (1)$$

Here T , Ω , and R are the transition, observation, and reward functions; ρ_0 is the initial-state distribution; and H is the implemented horizon. At decision t , the environment has hidden state $s_t \in \mathcal{S}$, emits text observation $o_t \in \mathcal{O}$, and exposes a finite command set $\mathcal{A}_t \subset \mathcal{A}$. Operationally, the agent observes the pair $\tilde{o}_t = (o_t, \mathcal{A}_t)$; we retain separate notation because the parser consumes $\mathcal{A}_t$ directly. The model receives a prompt derived from the history h_t , emits raw text z_t , and a parser P produces the executed action $a_t = P(z_t, \mathcal{A}_t)$. ALFWorld then returns o_{t+1} , the environment termination flag d_t , and an `info["won"]` value $w_t \in \{0, 1\}$. The visible trial loop stores w_t in its `reward` field but operationally returns episode success whenever $d_t = 1$; it does not separately test w_t at that return point.

For task i and trial k , define the step record and trace as

$$\xi_{i,k,t} = (o_{i,k,t}, \mathcal{A}_{i,k,t}, z_{i,k,t}, a_{i,k,t}, o_{i,k,t+1}, w_{i,k,t}, d_{i,k,t}), \quad (2)$$

$$\tau_{i,k} = (\xi_{i,k,t})_{t=1}^{L_{i,k}}, \quad (3)$$

where $L_{i,k} \leq H$. Following the implementation, let $\hat{S}_{i,k} = 1$ if trial k returns its success flag on the $d_{i,k,t} = 1$ code path. This operational label does not test $w_{i,k,t}$ separately. Retaining the CSV’s name, we define implementation-reported TSR over n_{task} unique tasks as

$$\text{TSR} = \frac{1}{n_{\text{task}}} \sum_{i=1}^{n_{\text{task}}} \mathbf{1}[\exists k \leq 3 : \hat{S}_{i,k} = 1]. \quad (4)$$

The denominator is the number of unique tasks, not episode rows. Tasks marked successful are logged

as skipped in later trials. Because completed structured primary episode records are absent, the reported TSR cannot be independently reclassified using $w_{i,k,t}$.

3.2 Admissible-Action Grounding

In free-text mode, the sanitizer accepts the prefixes `go to`, `open`, `close`, `take`, `put`, `examine`, `use`, `look`, `inventory`, and `think`. Empty outputs and invalid prefixes execute `look`; whitespace repair and LLM-call fallback are also marked as changes in the step record. The aggregate code reports

$$\text{IAR} = \frac{\sum_r n_r^{\text{invalid}}}{\sum_r L_r}, \quad (5)$$

using the logged `n_invalid_fallbacks` count. The supplied CPython 3.13 logger bytecode increments this count only for `invalid_prefix` and `empty_output`; it excludes LLM-call timeouts and whitespace-only repairs. No run-level hash ties that bytecode to the primary CSV, so IAR remains a run-reported operational statistic despite the recoverable definition.

In admissible mode, the prompt lists $\mathcal{A}_t$ with zero-based integer IDs. The parser accepts a bare integer, numbered prefix, exact text match, or partial text match. If all fail, it chooses the first admissible command other than `look` or `inventory`, falling back to `look` only as a last resort. Single-token outputs are accepted with `_min_len=1`. We therefore report admissible parser fallback separately:

$$\text{AdmFb} = \frac{\sum_r n_r^{\text{adm-fallback}}}{\sum_r n_r^{\text{adm-step}}}. \quad (6)$$

Whenever $\mathcal{A}_t$ is nonempty, admissible parsing guarantees that the executed command belongs to that list. Membership does not establish faithfulness to model intent: partial matching can resolve an output without entering a counted fallback path. `AdmFb` does not measure all heuristic interpretation or action-selection mismatches. The supplied logger bytecode counts `smart_fallback` and `last_resort_fallback` decisions in the `AdmFb` numerator. As with IAR, the absence of a run-linked source snapshot prevents proving that the supplied bytecode is identical to the primary-run logger.

The admissible set is read from `info["admissible_commands"][0]`

after `env.reset()` and every `env.step([action])`. The set returned after step t is used for decision $t + 1$ and for any recovery block prepared after step t . The logged field `admissible_actions_available` is assigned after `env.step`; it stores the next-state list rather than the list from which the current `selected_index` was chosen.

3.3 Structured Recovery Prompting

SRP operates inside a trial. The implementation computes two trailing counts after executing step t :

$$f_{i,k,t} = F(\tau_{i,k}[1:t]), \quad (7)$$

$$g_{i,k,t} = G(\tau_{i,k}[1:t]), \quad (8)$$

$$c_{i,k,t} = \max(f_{i,k,t}, g_{i,k,t}). \quad (9)$$

Here F counts a suffix of observations containing one of ten fixed failure phrases. The intended G predicate includes a failure phrase, sanitization, repeated action, repeated observation, or an empty `look`. Recovery is active only when $\theta > 0$. Define the logged-event indicator

$$e_{i,k,t} = \mathbf{1}[\theta > 0 \wedge c_{i,k,t} = \theta]. \quad (10)$$

The equality is exact, not $c_{i,k,t} \geq \theta$. If $e_{i,k,t} = 1$ and the trial continues, the next model call receives one recovery block containing the task, visited receptacles, failed actions, the last six action-observation pairs, and $\mathcal{A}_{t+1}$ with failed actions marked. Its final instruction requests only an integer. Writing Φ for the recovery-block constructor and Ψ for the normal prompt constructor,

$$x_{i,k,t+1} = \begin{cases} \Phi(h_{i,k,t+1}, \mathcal{A}_{i,k,t+1}), & e_{i,k,t} = 1, \\ \Psi(h_{i,k,t+1}, \mathcal{A}_{i,k,t+1}), & e_{i,k,t} = 0, \end{cases} \quad (11)$$

where $x_{i,k,t+1}$ is the actual prompt for the next decision. When the block is consumed, `set_recovery_mode(True)` resets repeated-action, repeated-observation, and no-progress counters. Recovery mode is never disabled within the trial: the repeated-action exhaustion threshold remains three steps higher and the no-progress threshold remains two steps higher until termination. The reported SRP condition therefore bundles prompt injection with termination-policy relaxation.

The implementation logs a recovery event after `env.step` but before checking done or exhaustion. Consequently, a terminal step can in principle contain a logged event even though no subsequent

recovery prompt is consumed. The structured step records needed to count this case are absent for the completed primary conditions.

The implemented G function does not fully realize its stated repetition logic. It scans backward with the comparison values initialized to `None`; a newest step that is only a pure action or observation repeat fails before a comparison is established. Figure 1 includes the direct synthetic checks. Consequently, the evidence does not support describing SRP as a general repeated-action detector.

3.4 Trace-Level Metrics

Final reported TSR hides whether an agent fails by invalid commands, logged mistake streaks, action-observation loops, or planning-depth exhaustion. We therefore report trace metrics and termination diagnostics separately.

For attempted episode r , let n_r^{seq} be the logged `n_mistake_sequences` value and let M be the number of non-skipped attempted episodes. The supplied logger bytecode defines each sequence as a maximal streak of at least two steps whose observation contains a fixed failure phrase or whose `sanitization_changed` flag is true. The aggregation code defines Multi-Mistake Sequence Rate as

$$\text{MMSR} = \frac{1}{M} \sum_{r=1}^M \mathbf{1}[n_r^{\text{seq}} > 0]. \quad (12)$$

MMSR is therefore not a measure of all valid-action loops. Repeated-action and max-step terminations are reported separately, and the missing run-level source snapshot still limits exact reproducibility.

Let $q_r = 1$ if attempted episode r contains at least one logged recovery event, and let $s_r = 1$ if that episode is marked successful. The aggregation code's RR field is therefore more accurately named triggered-episode success rate:

$$\text{TESR} = \frac{\sum_{r=1}^M q_r s_r}{\sum_{r=1}^M q_r}, \quad \sum_r q_r > 0. \quad (13)$$

If no event is logged, TESR is not applicable, although the aggregation code encodes it as 0.0. TESR is associational, not counterfactual: it does not show that a recovery prompt was consumed before success, caused success, or made the next step exit a loop. The visible step record contains an `is_recovery_step` flag, but the aggregate CSV does not expose it and the completed primary step records needed to analyze it are absent.

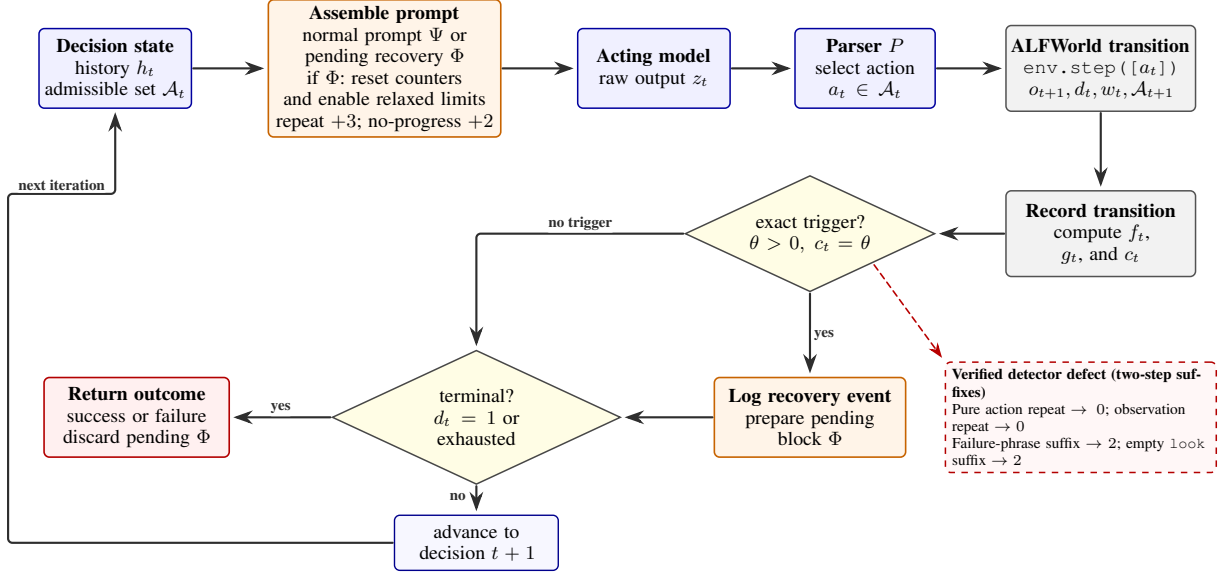


Figure 1: Code-faithful execution order for the implemented wrapper. The upper lane performs action selection and the environment transition; the lower lane records the step, applies the exact equality trigger, and then checks termination. A pending recovery block is consumed only on the next decision if the trial continues. The red callout reports direct tests of `count_consecutive_stagnation`.

4 Experimental Setup

4.1 Environment and Runtime

Primary admissible evaluations use ALFWorld `valid_unseen` with 134 tasks and three trials. Because tasks marked successful are skipped in later trials, C1–C6 contain 2,053 attempted episodes. Family counts are 18 `look_at_obj`, 31 `pick_clean_then_place`, 24 `pick_and_place`, 17 `pick_two_obj`, 21 `pick_cool_then_place`, and 23 `pick_heat_then_place`. Few-shot ReAct, Reflexion memory, and reflection generation are enabled. Temperature 0.0 disables deliberate sampling but does not guarantee cross-run determinism.

The command list is state-dependent. A command that was valid at one step may disappear after an object is taken, opened, cooled, cleaned, or placed. Aggregate metrics and figures are produced by the project scripts using NumPy, pandas, SciPy, and Matplotlib (Harris et al., 2020; McKinney, 2010; Virtanen et al., 2020; Hunter, 2007).

4.2 Parsed Hyperparameters

The captured primary pipeline header records 134 tasks, three trials, temperature 0.0, admissible actions, a 49-step hard budget, repeated-action limit 3, and no-progress limit 8. The visible current CLI defaults are 2 and 4, while the trial-function defaults are 3 and 5; the primary launch script is absent. We

therefore treat 3 and 8 as run-recorded effective values rather than reconstructed source defaults. The run/configuration ledger appears in Appendix A. The workspace contains `base_config.yaml` but no `eval_config.yaml`; no settings are inferred from the missing file. The configuration’s `random_seed: 42` is distinct from the later runtime `-seed` option used for task-order reruns.

4.3 Conditions

The primary six-condition admissible design is:

- C1/C2: Qwen2.5-14B, without/with SRP at $\theta = 2$.
- C3/C4: Qwen2.5-7B, without/with SRP at $\theta = 2$.
- C5/C6: Llama-3.1-8B, without/with SRP at $\theta = 2$.

The primary admissible pipeline ran on June 8–9, 2026. C2 is the completed rerun after an initial environment-reset failure. The earlier free-text pipeline ran on June 2, 2026. Because these pipelines were executed separately and the code evolved between them, their TSR difference is a historical comparison, not a controlled action-mode ablation.

5 Quantitative Results and Evaluation

5.1 Implementation-Reported Task Success Across Conditions

Table 1 reports the supplied primary summary with task-level Wilson intervals (Wilson, 1927). These are heuristic intervals under an independent, identically distributed Bernoulli model for tasks. Because `valid_unseen` is a fixed, heterogeneous benchmark split and each condition has only one pipeline execution, the intervals do not quantify run-to-run uncertainty and should not be used to infer an SRP effect.

The largest primary difference is the 57.5-point baseline contrast between Qwen2.5-14B and Qwen2.5-7B. Within-model SRP differences are smaller and heterogeneous, but no valid paired uncertainty estimate is available. Llama-3.1-8B has identical aggregate reported task success. These observations do not isolate the recovery prompt because SRP also changes termination policy. Logged `AdmFb` is low but nonzero; it does not quantify the use or faithfulness of other parsing routes, including partial matching. Figure 2 visualizes the aggregate and task-family results without converting the descriptive differences into treatment effects; Appendix B provides the remaining run-level diagnostics.

5.2 Task Archetypes

The category analysis is descriptive and exploratory; no multiplicity-adjusted category tests are available. In the historical free-text runs, all five multi-step categories have 0.0% TSR, while `look_at_obj` is nonzero.

In the primary admissible run, Qwen2.5-7B is higher with SRP on `look_at_obj`, `pick_and_place`, `pick_two_obj`, and `pick_heat_then_place`, but lower on cleaning and cooling. Qwen2.5-14B also has mixed category differences. Llama-3.1-8B has identical category marked-success counts in C5 and C6. The aggregate artifacts do not determine whether these changes arise from recovery prompting, the relaxed exhaustion limits, task-order effects, or their interaction.

5.3 Historical Pipeline Comparison

Table 2 compares the earlier free-text baseline pipeline with the later admissible baseline pipeline. Free-text execution is strongly affected by outputs that are sanitized to `look`. The comparison moti-

vates action-interface control, but it does not isolate action mode from all intervening code and run changes.

The directly supported boundary diagnosis is narrower than a TSR causal claim: the baseline free-text IAR values are 18.8% and 66.2%, whereas reported primary admissible steps use the environment list and invoke heuristic parser fallback on fewer than 1.1% of steps. The supplied optional Qwen2.5-14B free-text rerun directory is absent and is excluded from the analysis.

5.4 Uncertainty and Triggered-Episode Success

All baseline/SRP conditions evaluate the same 134 tasks, so a matched-task analysis is preferable to an independent-samples Fisher test. The primary episode JSON files and paired task outcomes are absent from the supplied bundle, preventing McNemar or paired-bootstrap analysis. We therefore treat +6.72 and +2.99 points as descriptive differences, not confirmatory effects.

Forensic notes report later Qwen2.5-7B reruns, but the referenced directories are absent and the recorded unpaired analysis ignores task matching and repeated-task clustering. These reruns are therefore excluded from the abstract, primary table, and conclusions.

Triggered-episode success is also descriptive. The primary CSV reports TESR values of 44.4% for Qwen2.5-14B, 28.0% for Qwen2.5-7B, and 0.0% for Llama-3.1-8B. Forensic notes report 27, 25, and 29 event-containing episodes, respectively, but the primary CSV does not expose these denominators and the completed structured records are absent. The counts and prompt-consumption sequence therefore cannot be independently reconstructed. Event-containing episodes are a selected subset, and TESR does not establish prompt consumption, rescue, loop exit, or the counterfactual outcome without SRP.

5.5 Logged Mistake Sequences

Historical free-text MMSR is high: 88.6% for the Qwen2.5-7B baseline, 90.2% for Qwen2.5-7B SRP, and 97.5% for the Llama-3.1-8B baseline. Primary admissible MMSR ranges from 1.1% to 7.2%. Low admissible MMSR does not imply high TSR or absence of loops; it only means that the logger rarely marked the specific sequence type counted by `n_mistake_sequences`.

Cond.	Model	SRP	Marked	Reported TSR	95% Wilson CI	AdmFb	Δ vs. base
C1	Qwen2.5-14B	none	95/134	70.9%	[62.7, 77.9]	0.87%	–
C2	Qwen2.5-14B	$\theta = 2$	99/134	73.9%	[65.9, 80.6]	1.01%	+2.99 pp
C3	Qwen2.5-7B	none	18/134	13.4%	[8.7, 20.2]	0.41%	–
C4	Qwen2.5-7B	$\theta = 2$	27/134	20.1%	[14.2, 27.7]	0.23%	+6.72 pp
C5	Llama-3.1-8B	none	7/134	5.2%	[2.6, 10.4]	0.62%	–
C6	Llama-3.1-8B	$\theta = 2$	7/134	5.2%	[2.6, 10.4]	0.67%	0.00 pp

Table 1: Primary admissible C1–C6 results from the supplied summary CSV. Reported TSR follows `done`, not a separate `won` check. AdmFb counts designated parser-fallback paths, not all heuristic parsing. Differences are descriptive; Wilson intervals do not establish an SRP effect.

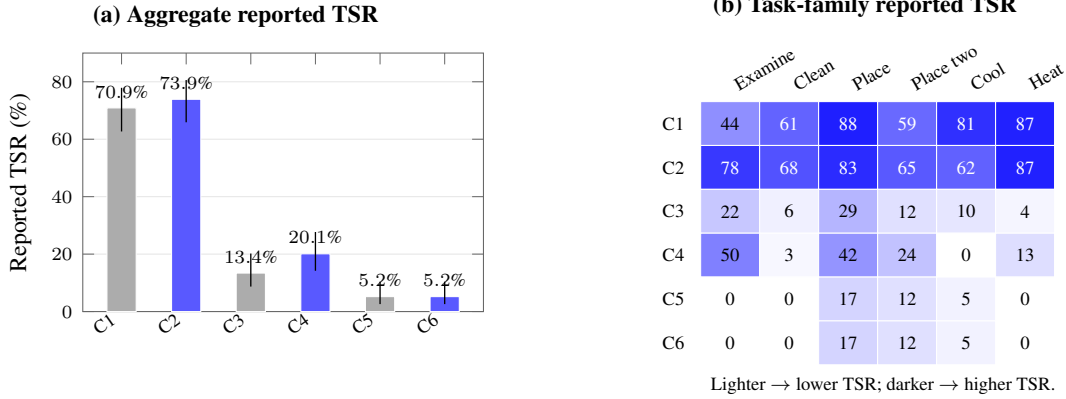


Figure 2: Primary admissible-action values transcribed from the supplied summary CSV; condition IDs match Table 1. Panel (a) shows implementation-reported task-level TSR with heuristic 95% Wilson intervals under an independent-task model; panel (b) shows rounded descriptive task-family reported TSR. The figure does not isolate a causal SRP effect because the treatment also changes termination policy and paired task outcomes are unavailable.

Model	Free	Admissible	Difference
Qwen2.5-7B	3.73%	13.4%	+9.70 pp
Llama-3.1-8B	2.24%	5.2%	+2.99 pp

Table 2: Historical baseline comparison using the runtime’s reported TSR from supplied CSV rows. The runs were separate pipelines, so “Difference” is not a causal grounding effect.

6 Failure Taxonomy and Architectural Discussion

Action-interface invalidity. In the historical free-text runs, Qwen2.5-7B invalid-fallback rates range from 18.8% to 30.3%, and Llama-3.1-8B ranges from 66.2% to 68.8%. These rates mean that many executed steps are sanitizer- or fallback-mediated rather than faithful executions of the raw model output. The resulting `look` actions and repeated observations confound any attempt to interpret low TSR solely as a planning failure.

Detector-limited trajectory stagnation. On steps where the environment exposes a nonempty admissible list, admissible parsing prevents execution of

commands outside that list, but it does not remove valid-action loops. The termination logger detects repeated actions independently, whereas the SRP trigger’s backward scan misses pure repeat suffixes as shown in Figure 1. This explains why low MMSR can coexist with high repeated-action termination. Both Qwen model pairs have positive descriptive TSR differences and nonzero marked successes among event-containing episodes. For Qwen2.5-7B, however, 72% of event-containing episodes still fail, and neither comparison isolates the treatment mechanism.

Long-horizon failure in Llama-3.1-8B. The primary CSV implies that 364/390 baseline episodes and 382/390 SRP episodes reach the 49-step limit after accounting for reported-success and repeated-action terminations. This establishes persistent non-completion under the wrapper, but it does not distinguish planning failure from state tracking, action selection, parser behavior, or another deficit. The wrapper supplies no explicit inventory, completed-subgoal, or remaining-plan state; whether adding such structure would improve these episodes is not tested.

Qwen2.5-14B does not establish saturation. Qwen2.5-14B has the strongest baseline and 44.4% TESR, but its baseline/SRP difference is only +2.99 points and no valid paired uncertainty estimate is available. The aggregate data cannot distinguish saturation, trigger-selection bias, category mix, or a genuinely small treatment effect. We therefore avoid a mechanistic saturation claim.

Model-size contrast dominates the primary table. The +57.5-point Qwen2.5 baseline contrast is larger than either within-model SRP difference. Because these are two separately deployed model sizes rather than a controlled scaling law, the defensible claim is descriptive: model choice is more strongly associated with TSR in this table than the implemented wrapper is.

7 Conclusion

The available evidence does not isolate a recovery-prompt effect in this ALFWorld wrapper. The four-contract audit explains why: SRP bundles prompt and termination-policy changes, misses pure repeat suffixes, logs events without separately confirming prompt consumption, and labels terminal `done` as success without checking `won`. Missing structured records prevent win-based relabeling and paired analysis. Historical free-text invalid fallback complicates interpretation of planning behavior; admissible-list membership prevents out-of-list execution when the list is nonempty, but low designated fallback does not establish faithful action selection. The primary model-size contrast exceeds the within-model SRP differences, which remain descriptive. The contribution is a bounded case study of action execution, trigger coverage, the complete treatment bundle, and outcome construction: these contracts need validation before aggregate gains can be attributed to recovery prompting.

Limitations

Composite treatment and detector defect. The SRP condition changes both the next prompt and, after block consumption, the exhaustion thresholds, so the observed differences cannot be attributed to prompting alone. Event logging also precedes the terminal checks, and the absent structured primary step records prevent counting how many logged events were followed by an actual recovery call. In addition, the stagnation scan does not reliably detect pure repeated actions or observations. A clean

follow-up requires separate prompt-only, threshold-only, and combined conditions after fixing and unit-testing the detector.

Success-label validity. The runtime returns success whenever `done=True` and does not separately require `info["won"]=True`. Because the two fields are recorded separately by the environment, their equivalence is not established by the audited code. The absent structured episode records prevent measuring any divergence or correcting the task outcomes. All TSR values in this paper are therefore implementation-reported and may differ from a win-validated TSR.

Statistical evidence. The primary table is one pipeline execution per condition. Conditions were executed sequentially in a fixed order rather than randomized or counterbalanced, so time-dependent serving or hardware effects cannot be excluded. Temperature 0.0 does not establish cross-run determinism or estimate run-level variance. Matched task outcomes are absent, so paired inference is unavailable. The later two-run numbers and $p = 0.004$ value survive only in forensic notes and use an independence assumption that does not match repeated tasks; they are provisional rather than confirmatory.

Runtime anomaly. C4 took 413 minutes, versus about 40 minutes for C3 and about 40 minutes in rerun notes. The forensic documents attribute this to GPU contention, but no resource-monitoring trace is supplied. We exclude the anomalous duration from performance claims and treat the cause as unverified.

Artifact completeness. See Appendix A for the parsed runtime configuration ledger. The bundle contains the primary summary CSV, pipeline stdout, historical free-text summaries and world logs, visible source for the parser, trigger, termination wrapper, CLI, and aggregation, and recoverable CPython 3.13 logger bytecode. The stdout preserves printed action–observation traces for C1 and C3–C6 plus the failed initial C2 attempt, but not the completed C2 rerun or the structured fields needed to audit `won`, prompt consumption, and task pairing. The primary episode JSON files, later rerun directories, `eval_config.yaml`, `prompts/alfworld_3prompts.json`, and source for `main.py`, `utils.py`, and `metrics_logger.py` are absent. No run-level commit hash links the summary, bytecode, and

visible sources to one immutable snapshot. Model tags are recorded, but immutable model digests and exact ALFWorld, Ollama, and dependency versions are not. Exact rerun reproducibility and prompt reconstruction therefore remain incomplete.

Historical comparison and external validity. The free-text and admissible pipelines ran on different dates with evolving code, so their TSR difference is not a controlled grounding effect. ALFWorld also exposes a complete legal-command list; many web, tool-use, and robotic environments do not. Transfer beyond ALFWorld is untested.

Metric and model scope. MMSR is recoverable from supplied logger bytecode but not a run-linked source file, and TESR is a selected-subset statistic rather than rescue. Only three local model deployments are evaluated, with no additional model family or commercial baseline. The results are diagnostic, not a leaderboard or scaling-law claim. Overstating incomplete artifacts or associative results could mislead subsequent agent research; we address this risk by making the evidence gaps, implementation defects, and non-causal interpretation explicit.

Ethical Considerations

The experiments use simulated household tasks and local model inference; no human participants or personal data are involved. More capable agents deployed outside simulation would require separate safety analysis for unintended actions and human oversight.

References

- Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2019. TextWorld: A Learning Environment for Text-Based Games. In *Computer Games: 7th Workshop, CGW 2018, Revised Selected Papers*, volume 1017 of *Communications in Computer and Information Science*, pages 41–75. Springer International Publishing. https://doi.org/10.1007/978-3-030-24337-1_3.
- Younna Farag, Svetlana Stoyanchev, Mohan Li, Simon Keizer, and Rama Doddipatla. 2025. Conditional Multi-Stage Failure Recovery for Embodied Agents. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pages 200–227, Vienna, Austria. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2025.realm-1.15>.
- Aaron Grattafiori et al. 2024. The Llama 3 Herd of Models. Computing Research Repository, arXiv:2407.21783, version 3. <https://doi.org/10.48550/arXiv.2407.21783>.
- Charles R. Harris et al. 2020. Array Programming with NumPy. *Nature*, 585:357–362. <https://doi.org/10.1038/s41586-020-2649-2>.
- John D. Hunter. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95. <https://doi.org/10.1109/MCSE.2007.55>.
- Ankush Kadu and Aswanth Krishnan. 2025. ReflexGrad: Within-Episode Failure Recovery in LLM Agents via Progress-Gated Dual-Process Routing. Computing Research Repository, arXiv:2511.14584, version 4, revised 2026. <https://doi.org/10.48550/arXiv.2511.14584>.
- Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1–2):99–134. [https://doi.org/10.1016/S0004-3702\(98\)00023-X](https://doi.org/10.1016/S0004-3702(98)00023-X).
- Philip Lippmann, Matthijs T. J. Spaan, and Jie Yang. 2025. Positive Experience Reflection for Agents in Interactive Text Environments. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pages 131–142, Vienna, Austria. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2025.realm-1.10>.
- Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. In *Advances in Neural Information Processing Systems*, volume 37, pages 74325–74362. <https://doi.org/10.52202/079017-2365>.
- Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56–61. <https://doi.org/10.25080/Majora-92bf1922-00a>.
- Ollama. 2024. OpenAI Compatibility. Ollama Blog, February 8. Accessed August 5, 2026. <https://ollama.com/blog/openai-compatibility>.
- Gregory Polyakov, Ilseyar Alimova, Dmitry Abulkhanov, Ivan Sedykh, Andrey Bout, Sergey Nikolenko, and Irina Piontkovskaya. 2025. Tool-Reflection: Improving Large Language Models for Real-World API Calls with Self-Generated Data. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pages

184–199, Vienna, Austria. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2025.realm-1.14>.

Qwen, An Yang, Baosong Yang, Beichen Zhang, et al. 2024. Qwen2.5 Technical Report. Computing Research Repository, arXiv:2412.15115, version 2. <https://doi.org/10.48550/arXiv.2412.15115>.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. <https://doi.org/10.52202/075280-0377>.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10737–10746. <https://doi.org/10.1109/CVPR42600.2020.01075>.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=0IOX0YcCdTn>.

Pauli Virtanen et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272. <https://doi.org/10.1038/s41592-019-0686-2>.

Edwin B. Wilson. 1927. Probable Inference, the Law of Succession, and Statistical Inference. *Journal of the American Statistical Association*, 22(158):209–212. <https://doi.org/10.1080/01621459.1927.10502953>.

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

A Parsed Runtime Configuration

This ledger records only values visible in the supplied configuration and runtime code. The missing `eval_config.yaml` remains an artifact gap; no values are inferred from it.

Dataset paths. `base_config.yaml` specifies `json_2.1.1/train`, `valid_seen`, and `valid_unseen` for training, in-distribution evaluation, and out-of-distribution evaluation,

respectively. The train and evaluation game caps are both `-1`.

Environment. The `AlfredTWE` environment has domain randomization disabled. Task types 1–6 are enabled, the expert timeout is 150, and the expert type is hardcoded.

Controller and general settings. The controller is `oracle`; receptacles are loaded; the configuration seed is 42; CUDA is enabled; the observation-pool capacity is 3; and initial receptacles are not hidden.

Training and evaluation defaults. Evaluation is enabled with batch size 10. The configured RL action space is `admissible`, maximum target length is 20, beam width is 10, and maximum step count is 50. This is an inherited configuration value; the visible trial loop’s 49-step budget is the operative truncation limit for the reported runs.

Primary runtime CLI. The primary runs use 134 environments, three trials, and temperature 0.0, with memory, reflection, and few-shot examples enabled.

Action and recovery modes. Primary C1–C6 use `admissible` actions; the historical grounding runs use free text. Recovery thresholds represented in the workspace are 0, 2, and the optional value 3.

Termination policy. The visible loop has a 49-step hard budget. The current CLI defaults are repeated-action limit 2 and no-progress limit 4, while the trial-function defaults are 3 and 5. The captured primary pipeline header reports effective values 3 and 8; its launch script is absent. Recovery mode adds three repeated-action allowances and two no-progress allowances to the effective values.

Local inference API. The runtime uses an OpenAI-compatible local endpoint. Its short-output retry loop makes up to six attempts before fallback. The separate utility-layer source is absent from the supplied source bundle.

Model stack. The local Ollama model tags are:

- `qwen2.5:14b-instruct`,
- `qwen2.5:7b-instruct`, and
- `llama3.1:8b`.

The acting model also handles recovery prompting.

Cond.	TSR	MMSR	TESR	MSL	SPT	N	IAR	LL	Rep.	NP	AdmFb
C1	70.90	2.68	–	2.33	10.95	261	0.00	0.87	44.83	0.00	0.87
C2	73.88	2.33	44.44	2.00	12.74	258	0.00	0.33	41.47	0.00	1.01
C3	13.43	1.31	–	2.67	10.94	383	0.00	0.37	54.05	0.00	0.41
C4	20.15	1.08	28.00	2.00	11.33	371	0.00	0.26	45.28	0.00	0.23
C5	5.22	6.41	–	2.74	13.86	390	0.00	0.00	4.87	0.00	0.62
C6	5.22	7.18	0.00	2.00	13.86	390	0.00	0.00	0.26	0.00	0.67

Table 3: Run-level primary aggregate metrics used in the analysis. Rates are percentages; N counts non-skipped attempted episodes. TESR relabels the CSV column RR.

Condition	TSR	MMSR	TESR	MSL	SPT	N	IAR	LL	Rep.	NP
Qwen2.5-7B baseline	3.73	88.61	–	2.60	5.60	395	18.81	5.16	89.11	6.33
Qwen2.5-7B SRP $\theta = 2$	9.70	90.18	1.90	2.44	12.69	387	30.27	11.96	75.71	2.07
Qwen2.5-7B SRP $\theta = 3$	6.72	89.54	3.33	2.72	13.78	392	25.70	10.06	83.16	1.28
Llama-3.1-8B baseline	2.24	97.47	–	2.08	4.00	396	66.20	29.74	98.74	0.51
Llama-3.1-8B SRP $\theta = 2$	2.24	97.22	0.00	2.61	4.00	396	68.81	36.59	99.24	0.00

Table 4: Run-level aggregate metrics used from the historical free-text summary CSV. Rates are percentages; TESR relabels the CSV column RR.

B Additional Aggregate Metrics

Table 3 collects the run-level diagnostic columns used in the analysis. TSR and SPT inherit the run-time’s `done`-based success label; SPT is the mean number of steps among attempted episodes marked successful. MSL is the mean length of logged mistake sequences. IAR is the free-text invalid-output fallback rate. The supplied logger bytecode defines LL as the number of consecutive `look` actions after the first, divided by total steps; it does not require the observations to match. Rep. is the attempted-episode repeated-action termination rate, NP is the no-progress termination rate, and AdmFb is the admissible-parser fallback rate. Baseline TESR is shown as not applicable rather than the encoded value 0.0 because no recovery event can occur. The source CSV also contains task counts, Wilson limits, and per-family TSR/MMSR columns; task counts and family TSR appear in the main paper.

The source loop returns only reported success, repeated-action exhaustion, no-progress exhaustion, or the 49-step limit. Table 5 therefore reconstructs terminal counts from N , tasks marked successful, and the two explicit termination rates. Because marked tasks are skipped in later trials, each such task contributes one attempted episode labeled successful. The max-step counts are residuals, not independently logged columns in the supplied CSV.

C Historical Free-Text Metrics

Table 4 preserves all aggregate columns used from the separately executed June 2 free-text pipeline. These runs are not pooled with the primary admissible pipeline because the execution date and code path differ.

Cond.	Marked success	Repeat	No prog.	Max step
C1	95	117	0	49
C2	99	107	0	52
C3	18	207	0	158
C4	27	168	0	176
C5	7	19	0	364
C6	7	1	0	382

Table 5: Attempted-episode terminal outcomes reconstructed from the primary aggregate summary and trial-loop return paths. “Marked success” inherits the run-time’s `done`-based label.