CONSISTENT PLAN-ACT FOR LONG-HORIZON AGENTIC TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

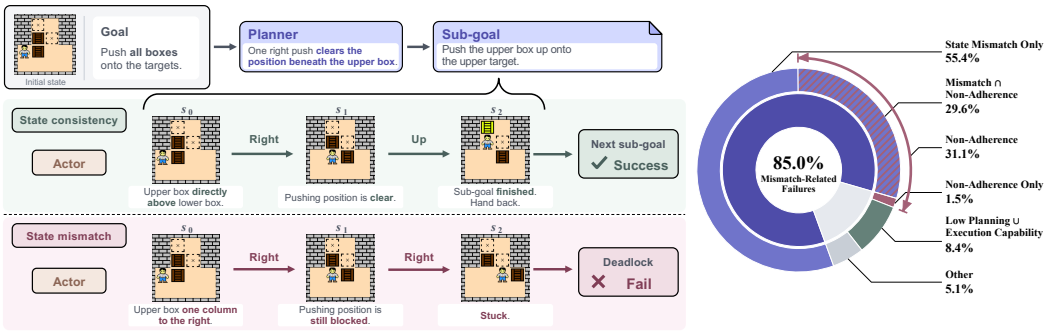
Long-horizon agentic tasks demand strong reasoning and efficient execution across successive interactions with dynamic environments. A common approach decouples high-level planning from low-level execution through separate planner and actor roles. To investigate coordination failures in these tasks, we prompt both agents for structured state assertions and compare their reports programmatically to detect explicit contradictions. Our analyses reveal systematic disagreement about the same task-relevant state facts, a phenomenon we term ***planner-actor state mismatch***. We further find that providing agents with task-relevant state information reduces mismatch and improves coordination and task performance. Based on the systematic analysis of the state mismatch, we propose **Con**sistent **P**lan-**Act** (**ConPAct**), which feeds detected contradictions back to both agents to form consistent state interpretations and fine-tunes them on curated consistent interactions for better coordination. ConPAct improves performance across various environments and model configurations, e.g., increasing MiniGrid success rate from 38.6% to 54.4% with GPT-5.6-sol/terra as planner and actor respectively, demonstrating that state consistency can guide both inference-time correction and coordination training.

1 INTRODUCTION

Large language models (LLMs) (OpenAI, 2026a; DeepSeek-AI, 2026) increasingly serve as agents in interactive environments, including games (Paglieri et al., 2025b), web navigation (Yao et al., 2022; Zhou et al., 2024), GUI automation (Rawles et al., 2023; Xie et al., 2024; Rawles et al., 2025), and embodied tasks (Shridhar et al., 2021; Ahn et al., 2023; Liu et al., 2023). These tasks require agents to combine long-term planning (Chen et al., 2025b; Li et al., 2026a) with action selection guided by observations and environmental feedback (Wang et al., 2025a). Alongside frameworks that interleave reasoning and action (Yao et al., 2023) or incorporate reflection on task feedback (Shinn et al., 2023), plan-act architectures assign high-level planning and low-level execution to separate agents (Erdogan et al., 2025; Jin et al., 2026; Molinari & Ciravegna, 2026). A planner translates the overall objective into sub-goals (Hu et al., 2025b), while an actor selects and adapts actions to achieve them (Erdogan et al., 2025; Peng et al., 2026). This division reduces the burden of simultaneously managing planning and execution (Erdogan et al., 2025) and allows models with different capabilities to serve the two roles (Tan et al., 2026). Their coordination through sub-goals and execution feedback requires common ground (Clark & Brennan, 1991): compatible interpretations of the task-relevant information referenced in sub-goals and execution feedback.

We use GPT-5.6-sol (OpenAI, 2026c) to examine sub-goal execution in failed plan-act trajectories across tasks and model configurations. Categorizing these failures reveals recurring disagreements between the planner and actor about the same task-relevant environment state. These disagreements can lead the actor to deviate from the planner's intent or treat feasible sub-goals as infeasible, contributing to task failure. This failure mode differs from limited planning (Li et al., 2026c; Tan et al., 2026; Dou et al., 2026) or execution (Hu et al., 2025b; Zhen et al., 2026; Wang et al., 2025a) capabilities, and sub-goal non-adherence (Jia et al., 2025; Kim et al., 2026; Venkatesan et al., 2026). We term it ***planner-actor state mismatch***. Figure 1a illustrates how individually coherent decisions can fail to support coordinated progress. State mismatch occurs at least once in **85%** of the analyzed failed trajectories (Figure 1b). Improving planning (Erdogan et al., 2025; Li et al., 2026c), execution (Hu et al., 2025b), or adherence (Jeong et al., 2026) does not by itself require the two roles to

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.



(a) Planner-Actor state mismatch and consistency. (b) Failure cases.

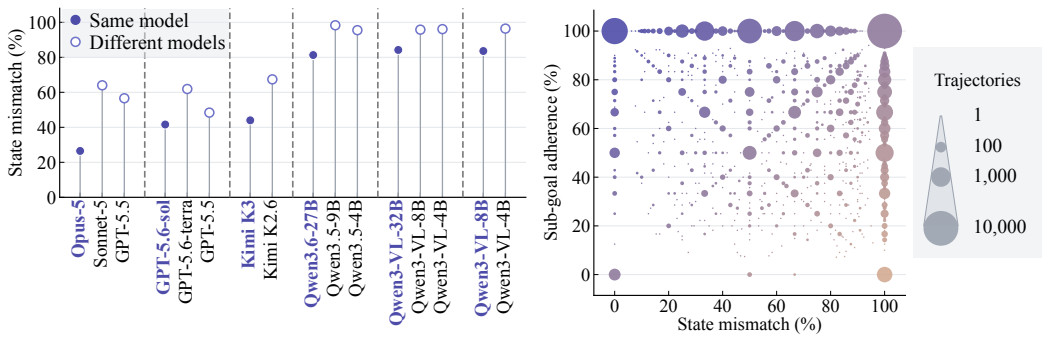
Figure 1: **Planner-actor state mismatch**. (a) Misreading the upper box as one column farther right, the actor pushes `Right` twice, causing deadlock; the correct sequence is `Right` → `Up`, with consistent interpretation as the planner. (b) Mismatch occurs in **85.0%** of failed trajectories, far exceeding non-adherence (31.1%) and low planning/execution capability (8.4%).

agree on the state facts underlying their decisions. Prior work examines inter-agent failures (Cemri et al., 2025), explicit belief-state representations (Li et al., 2023), and belief alignment through dialogue (Qiu et al., 2024; Dongre & Hakkani-Tür, 2026); here we focus on agreement about the same observed state facts at planner-actor handoffs.

To measure state mismatch, we use state assertions inspired by explicit belief-state representations (Li et al., 2023). Each agent reports its perceived task-relevant state in a shared structured format alongside its sub-goal or action. These reports serve as proxies for state interpretations, enabling programmatic detection of explicit contradictions without ground truth, or external verifiers. Our analysis is summarized as follows. First, mismatch occurs across environments and models and increases with the capability gap between roles. Controlled comparisons show that it persists under both visual and textual observations, with cross-role disagreement exceeding within-role sampling variability. Second, controlled interventions establish its causal impact on coordination: correcting the actor's state interpretation restores coordinated behavior while holding the sub-goal and observation fixed. Finally, our mismatch metric enables us to examine whether providing reliable state information improves agreement between the roles. Supplying state assertions from a stronger model or environment oracle reduces the measured disagreement while increasing task success.

Building on these findings, we propose **Con**sistent **P**lan-**Act** (**ConPAct**), which feeds programmatically detected state contradictions back to both agents. Unlike model-generated critiques (Madaan et al., 2023; Du et al., 2024), this feedback identifies mutual conflicting fields in their state assertions without any external information. The agents then discuss these disagreements and revise their interpretations, sub-goals, or actions before subsequent execution. There are three variants: **ConPAct-I**nference applies this intervention during inference without training. **ConPAct-S**ampling trains on sampled collaboration and correction sequences selected for state consistency, whereas **ConPAct-R**eplaying uses a teacher model to reconstruct such supervision from existing ReAct (Yao et al., 2023) trajectories. Experiments across various environments demonstrate benefits from both inference-time intervention and training. On MiniGrid (Chevalier-Boisvert et al., 2023; 2019), ConPAct-I raises success from 38.6% to 54.4% with a GPT-5.6-sol planner and GPT-5.6-terra actor (OpenAI, 2026c). ConPAct-S reaches 39.7% with Qwen3-VL-32B (Bai et al., 2025), exceeding the strongest evaluated training baseline by a large margin (29.1%). Ablations support the contributions of explicit conflict feedback, participation by both roles, and supervision of correction sequences. Our contributions are summarized as follows:

- We identify and characterize ***planner-actor state mismatch***, a relational failure mode. We establish its prevalence, impact on coordination, and mitigation potential through systematic analyses.
- We develop **ConPAct**, which uses programmatically detected state contradictions to guide joint revision during inference and trains agents on consistent collaboration and correction sequences obtained through sampling or replaying from existing trajectories.
- Experiments across environments and model configurations demonstrate improved coordination and task performance, with ablations supporting the contributions of each components.



(a) Prevalence of planner–actor state mismatch. (b) State mismatch despite high sub-goal adherence.

Figure 2: **Prevalence of planner-actor state mismatch.** (a) Mismatch occurs across various model configurations, with even homogeneous configurations have notable state mismatch; (b) State mismatch coexists with high sub-goal adherence, with marker size indicating trajectory frequency. Model sources are provided in Appendix C.2.

2 Characterizing Planner-Actor State Mismatch

2.1 Problem Setup and Measurement

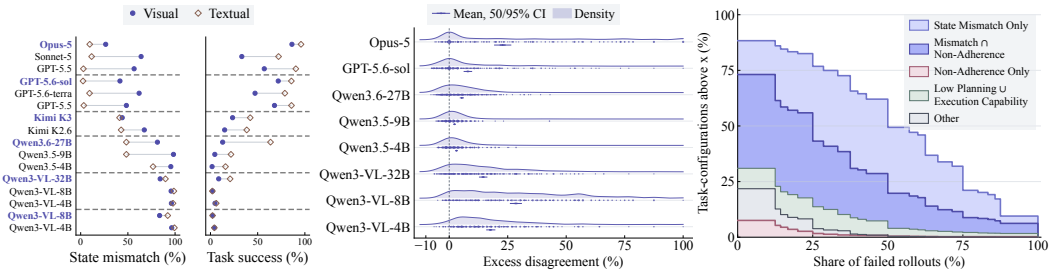
Long-Horizon Agentic Tasks. We study planner-actor state mismatch in Sokoban (Hu et al., 2025a) and MiniGrid (Chevalier-Boisvert et al., 2023; 2019), which require agents to pursue goals over extended sequences of interdependent actions and coordinate planning and execution throughout the task. Their underlying grid states are also readily accessible, providing ground truth for evaluating agents' state interpretations. We evaluate 200 Sokoban levels and 150 MiniGrid levels, with eight rollouts per level. Our evaluation covers Claude (Anthropic, 2026a;b), GPT (OpenAI, 2026c;b), Kimi (Kimi Team, 2026; Moonshot AI, 2026), and Qwen (Qwen Team, 2026b;a; Bai et al., 2025), using homogeneous/heterogeneous model configurations.

Plan-Act Agents. Given task instruction x_i, trajectory τ_i consists of model-call turns t, each with observation $o_{i,t}$ and response $y_{i,t}$. The planner generates a sub-goal $g_{i,t}$ from its observation; the actor uses the latest sub-goal and its own observation to select action $a_{i,t}$ and report execution status $z_{i,t}$ (Erdogan et al., 2025; Peng et al., 2026). Consecutive actor turns under one sub-goal form an *execution segment*. Completion or execution difficulties trigger *dynamic replanning*, where the planner revises the sub-goal using the latest observation and actor feedback (Erdogan et al., 2025; Lin et al., 2023; Paglieri et al., 2025a). Coordination through sub-goals and feedback thus depends on the roles' interpretations of task-relevant state.

State Assertions. We prompt each role to report its perceived state as `JSON` alongside its sub-goal or action. For each environment, we specify a compact schema $\mathcal{F}$ shared by both roles, with fixed fields and a predefined value domain $\mathcal{V}_f$ for each field f. Both roles use the shared schema $\mathcal{F}$ describing task-relevant properties inferable from a single observation, such as object identities, action prerequisites, and goal conditions. Each field f has a predefined set $\mathcal{V}_f$ of canonical, mutually exclusive values. Parsing response $y_{i,t}$ yields the *state assertion* $\mathbf{v}_{i,t} = \text{parse}_{\mathcal{F}}(y_{i,t})$, where $\mathbf{v}_{i,t}[f] \in \mathcal{V}_f$. At planner-actor handoffs, we compare assertions from consecutive role calls on the same observation ($o_{i,t} = o_{i,t+1}$), without either role seeing the other's assertion. For a valid pair, we detect *state mismatch* if any corresponding field differs:

$$d_{i,t} = \mathbf{1}[\exists f \in \mathcal{F} : \mathbf{v}_{i,t}[f] \neq \mathbf{v}_{i,t+1}[f]]\,. \tag{1}$$

Metrics for analyzing state mismatch. (1) *State mismatch rate* (D) is the fraction of valid handoff pairs with $d_{i,t} = 1$; (2) *State correctness* (A^{plan}, A^{act}) is each role's exact-match assertion accuracy against environment ground truth on the same paired observations; (3) *Sub-goal adherence* (H) is the fraction of execution segments in which an LLM judge (Zheng et al., 2023) determines that the actor pursues the assigned sub-goal; (4) *Task success* (S) is the fraction of trajectories with environment-confirmed task completion. The detailed definitions are placed in Appendix B.



(a) Visual vs. textual mismatch. (b) Excess cross-role disagreement. (c) Failure patterns.

Figure 3: Planner-actor state mismatch is systematic and prevalent in failed trajectories. (a) Mismatch persists under both visual and textual observations; (b) mean cross-role disagreement exceeds within-role sampling variability; (c) mismatch-only failures dominate across task configurations.

2.2 Prevalence of Planner-Actor State Mismatch

Prevalence of state mismatch. As shown in Figure 2a, mismatch occurs in all combinations, reaching 26.5% and 41.6% for homogeneous Opus-5 and GPT-5.6-sol, respectively. Figure 2b illustrates that high sub-goal adherence coexists with substantial state mismatch. Specifically, 82.9% of those with measured adherence $H = 100\%$ contain at least one detected mismatch.

State mismatch across observation modalities. In Figure 3a, replacing screenshots with structured textual states reduces mismatch in stronger configurations, but mismatch persists in all configurations. Even homogeneous Opus-5 and GPT-5.6-sol still exhibit rates of 9.6% and 2.5%, respectively. Therefore, planner-actor state mismatch extends beyond visual perception difficulties.

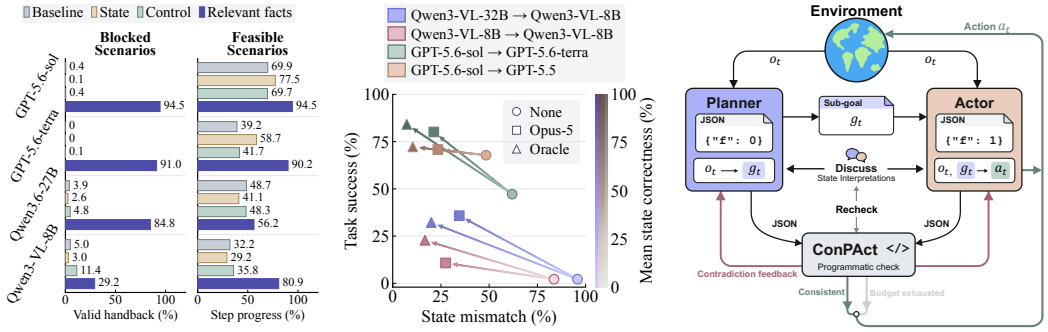
State mismatch beyond sampling variability. To test whether sampling variability explains mismatch, we repeatedly sample on 300 fixed observations while keeping each role's model and context fixed. We measure disagreement between state assertions from repeated sampling within each role and average the planner's and actor's rates to obtain a sampling baseline. Cross-role disagreement exceeds this baseline for all eight models under visual observation, with all 95% confidence intervals for the excess excluding zero (Figure 3b). For Opus-5, the excess is 22.8 percentage points, indicating that planner-actor disagreement contains systematic differences beyond sampling variability.

2.3 Behavioral Implications of State Mismatch

State mismatch dominates failure patterns across task configurations. Figure 3c plots the fraction of level-configuration pairs exceeding each within-pair failure-share threshold, considering only pairs with failed rollouts. Mismatch-only failures exceed these thresholds in more pairs than non-adherence-only or low-capability failures. In 49.4% of these pairs, mismatch without detected non-adherence occurs in more than half of all failures. Thus, conflicting state interpretations characterize most failures in a broad range of task configurations.

Controlled interventions. We keep the observation, assigned sub-goal, and actor fixed within each event and vary only the supplementary information. We compare four conditions: no supplementary information, a state assertion with random field values to control for structured formatting, freeform control context, and task-relevant state facts from Opus-5. Freeform control context consists of character-matched irrelevant text for blocked sub-goals and non-critical state facts for feasible sub-goals. Valid handback rate measures handoff signals in the first response; first-step progress rate measures first actions that reduce distance to the sub-goal.

Task-relevant state information improves feedback and execution. The results are displayed in Figure 4a. For blocked sub-goals, task-relevant facts increase GPT-5.6-sol's valid handback rate from 0.4% to 94.5%, compared with 0.1% for random state assertions and 0.4% for control context. For feasible sub-goals, they increase Qwen3-VL-8B's first-step progress rate from 32.2% to 80.9%, compared with 29.2% for random state assertions and 35.8% for non-critical state facts. The gains over both controls show that task-relevant state information helps actors adapt their coordination decisions to sub-goal feasibility, i.e., returning control for replanning when execution is blocked and selecting a progress-making action when execution is feasible. Full results for mismatch prevalence, observation and sampling controls, and state interventions appear in Appendix D.



(a) Behavioral implications. (b) Injecting state assertions. (c) **ConPAct-I** framework.

Figure 4: State information and consistent plan-act coordination. (a) Task-relevant state facts improve the actor's decisions. (b) Injecting reliable state assertions reduces mismatch and improves task success. (c) ConPAct feeds detected contradictions back for coordination reconciliation.

3 Consistent Plan-Act

3.1 Task-Level Implications and Consistency Interventions

To evaluate the task-level effects of reliable state information, we inject oracle assertions from environment states and assertions generated by Opus-5. Relative to no injection, both sources improve assertion correctness, reduce mismatch, and increase task success consistently, shown in Figure 4b. For GPT-5.6-sol paired with GPT-5.6-terra, oracle injection reduces mismatch from 61.9% to 7.3% and raises task success from 47.2% to 84.1%. Building on these gains and the behavioral results in Section 2.3, we propose **Con**sistent **P**lan-**Act** (**ConPAct**), which aligns planner-actor state interpretations through task-relevant information exchange. Detected contradictions identify the state facts that require reassessment. ConPAct feeds these contradictions back to both agents, directing their discussion toward the disputed facts and how they should inform planning and execution.

3.2 Feeding Back Detected Contradictions for State Consistency

As shown in Figure 4c, **ConPAct-I** compares state assertions about the same observation at planner–actor handoffs. When definite values conflict, it feeds the disputed fields and both roles' judgments back to the agents. Environment actions remain paused so that the discussion concerns a fixed state. The planner and actor exchange their interpretations, reassess the disputed facts against their observations, and revise their state judgments and associated sub-goals or execution decisions. We recheck their assertions after each exchange until all fields have definite, matching values or the discussion budget is exhausted. Execution then resumes with the updated discussion context. This process uses the agents' observations and exchanged messages, requiring neither environment ground truth nor an additional verifier model.

3.3 Learning State-Consistent Coordination

We further train agents to coordinate and resolve state disagreements using consistent interactions and corrective discussions. **ConPAct-S** collects these interactions by sampling trajectories with ConPAct-I. From both successful and failed trajectories, we retain consistent collaboration segments and correction sequences that reach full agreement across all assertion fields. We supervise only valid replies without detected state contradictions, while keeping earlier inconsistent replies as context for learning how to revise conflicting judgments. Each example supervises the role's complete response given its task, observation, and available interaction history.

Prior work derives plan supervision from execution traces (Erdogan et al., 2025) or relabels their goals in hindsight (Andrychowicz et al., 2017; Li et al., 2026b). When only existing ReAct (Yao et al., 2023) trajectories are available, **ConPAct-R** converts them into planner-actor interactions while preserving the recorded action sequence. A teacher segments each trace into sub-goals and adds state assertions and role exchanges. Around nonoptimal actions, it constructs hypothetical errors in state judgments or sub-goals, followed by feedback and revision, to provide correction examples. We validate the reconstructed interactions against the recorded execution and review

Table 1: Inference-time performance of Plan-Act methods. **Bold**/underlining mark the best/second-best values, including ties. The ReAct (Actor) uses only the actor and is unranked.

	Opus-5 ↓ *Sonnet-5*				*GPT-5.6-sol* ↓ *GPT-5.6-terra*				*Kimi K3* ↓ *Kimi K2.6*				*Qwen3.6-27B* ↓ *Qwen3.5-9B*			
Method	Sok.	Cra.	Proc.	Mini.	Sok.	Cra.	Proc.	Mini.	Sok.	Cra.	Proc.	Mini.	Sok.	Cra.	Proc.	Mini.
ReAct (Actor)	70.9	4.74	20.55	33.4	66.2	5.10	17.21	31.4	45.7	3.39	11.23	21.9	6.2	3.14	2.29	7.9
Plan-Act	84.2	5.26	23.49	33.1	75.1	5.13	17.20	38.6	57.7	**4.23**	10.85	28.3	12.0	3.27	3.19	**13.9**
HiPlan	91.3	5.23	25.26	31.6	76.7	5.13	17.95	47.7	65.5	3.29	11.85	27.1	13.3	3.32	4.76	10.6
TAPE	90.5	5.17	21.25	29.2	75.9	**5.24**	21.12	31.2	53.6	3.82	12.15	21.5	10.3	3.01	2.11	8.6
ConPAct-I	**95.4**	**5.91**	**25.75**	**36.7**	**86.6**	5.14	**22.83**	**54.4**	**65.8**	4.16	**12.79**	**34.2**	**15.7**	**3.46**	**4.85**	13.5

each candidate target using only the information available to the student at that turn. We then apply the same consistency filtering and supervision rules as ConPAct-S, keeping hypothetical errors and nonoptimal actions as context only. Algorithms and prompt templates for all three variants are provided in Appendix E.

4 Evaluation

4.1 Experimental Setup

Environments. We evaluate on Sokoban from GamingAgent (Hu et al., 2025a), Crafter (Hafner, 2022), Procgen (Cobbe et al., 2020), and MiniGrid/BabyAI (Chevalier-Boisvert et al., 2023; 2019). These environments cover box-pushing, survival and crafting, procedurally generated games, and instruction-following grid-world tasks. The evaluation sets contain 200 Sokoban puzzles balanced between one and two boxes, 200 Crafter seeds, 140 levels from seven Procgen games in easy mode, and 150 instances spanning 27 MiniGrid and BabyAI task IDs. Each instance is evaluated with eight rollouts. The action and total model-call limits are both 60 for Sokoban, 160 for Crafter and Procgen, and 60–200 for MiniGrid, depending on the task. We additionally evaluate desktop task performance on 361 OSWorld (Xie et al., 2024) tasks spanning ten application categories, with three rollouts per task.

Configurations. ConPAct-I is evaluated with four planner–actor pairs from the Claude, GPT, Kimi, and Qwen families (Table 1). For supervised training, we use Qwen3-VL-8B and Qwen3-VL-32B (Bai et al., 2025), with either the same model size in both roles or a 32B planner paired with an 8B actor. GPT-5.6-sol (OpenAI, 2026c), with its default thinking effort, serves as the teacher for data construction, and the training-token budget is matched to three epochs of ReAct SFT. ConPAct-S/R are evaluated with model-initiated discussion and without system-triggered ConPAct-I.

Baselines. Inference baselines include ReAct (Yao et al., 2023) using either the planner or actor model, Plan-Act (Erdogan et al., 2025), HiPlan (Li et al., 2026c), and TAPE (Jeong et al., 2026). Training baselines include ReAct SFT, Plan-Act SFT, PAA (Erdogan et al., 2025), HSL (Li et al., 2026b), WebSTAR (He et al., 2026), and ECoT (Zawalski et al., 2025).

Evaluation metrics. We focus on Sokoban avg@8, Crafter Score (Hafner, 2022), Procgen interquartile mean (IQM) (Agarwal et al., 2021), and MiniGrid success rate (SR). MiniGrid SR is averaged equally across the 27 task IDs; Procgen normalized-return metrics are multiplied by 100 for display. For OSWorld, we report mean evaluation score and full success rate as Score/SR (%). State analyses use mismatch rate D and assertion correctness A^{plan} and A^{act} from Section 2.1. Ground-truth states serve only to evaluate correctness in these analyses. Experimental configurations and metric definitions are provided in Appendices C and B.3, respectively.

4.2 Main Results

The main tables report Sokoban avg@8, Crafter Score, Procgen IQM, and MiniGrid SR (all ↑). Complete inference and training results are provided in Appendix Tables 15 and 16.

Table 2: Training results. **Bold**/underlining mark the best/second-best values, including ties.

	Qwen3-VL-8B ↓ *Qwen3-VL-8B*				*Qwen3-VL-32B* ↓ *Qwen3-VL-32B*				*Qwen3-VL-32B* ↓ *Qwen3-VL-8B*			
Method	Sok.	Cra.	Proc.	Mini.	Sok.	Cra.	Proc.	Mini.	Sok.	Cra.	Proc.	Mini.
ReAct SFT	44.4	4.24	4.50	8.8	60.1	4.63	6.88	18.5	—	—	—	—
Plan-Act SFT	35.6	4.06	3.50	7.8	50.7	3.10	6.35	15.8	45.4	2.85	5.60	11.9
PAA	43.8	4.33	3.15	10.8	65.4	3.76	8.27	11.6	49.4	3.40	5.89	8.7
HSL	44.4	4.39	4.82	8.0	61.3	4.23	5.85	16.5	—	—	—	—
WebSTAR	44.7	4.28	5.41	8.7	66.2	4.96	7.26	13.1	—	—	—	—
ECoT	34.4	4.77	2.56	10.2	57.2	5.23	5.25	29.1	41.2	4.70	2.62	12.7
ConPAct-S	48.6	**5.16**	3.22	**27.0**	**68.4**	5.05	**9.12**	**39.7**	48.5	**5.53**	4.56	**28.2**
ConPAct-R	**49.1**	4.72	**6.38**	24.8	66.9	**5.55**	6.35	33.9	**54.8**	5.26	**7.36**	27.8

ConPAct-I improves task performance across model families. Table 1 shows that ConPAct-I improves Sokoban avg@8 and Procgen IQM over Plan-Act for all four model pairs. Sokoban gains range from 3.7 to 11.5 percentage points. With GPT-5.6-sol and GPT-5.6-terra, MiniGrid SR increases from 38.6% to 54.4%, exceeding both HiPlan and TAPE. These results support using detected state contradictions to improve plan-act coordination without training.

Both sampling and replay provide effective coordination supervision. In Table 2, ConPAct-S and ConPAct-R outperform both ReAct SFT and Plan-Act SFT on Sokoban avg@8 and MiniGrid SR at both model sizes. For Qwen3-VL-32B, ConPAct-S achieves 39.7% MiniGrid SR, compared with 29.1% for the strongest baseline. The two data routes also yield leading results on the other environments: ConPAct-S attains the highest Procgen IQM (9.12), while ConPAct-R attains the highest Crafter Score (5.55) among the 32B methods.

The gains extend to heterogeneous model sizes. With a 32B planner and an 8B actor (Table 2, rightmost column group), both variants improve Sokoban avg@8, Crafter Score, and MiniGrid SR over Plan-Act SFT. MiniGrid SR rises from 11.9% to 28.2% with ConPAct-S and 27.8% with ConPAct-R. Thus, consistency-oriented supervision also benefits coordination between models of different sizes.

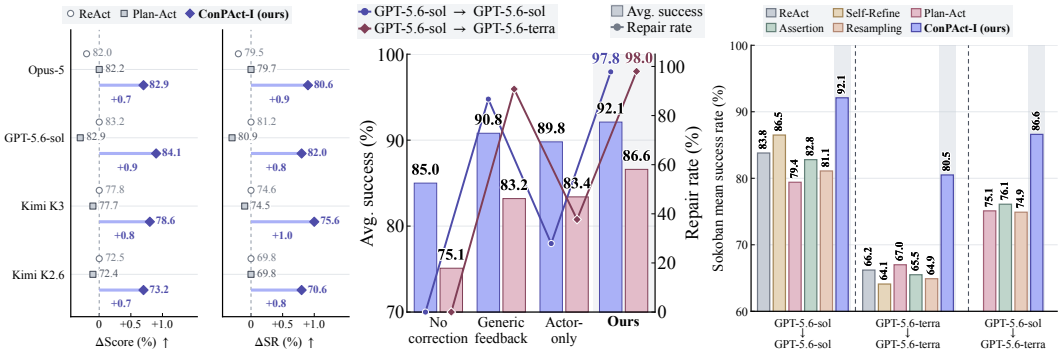
The improvements extend to desktop tasks. On 361 OSWorld tasks, ConPAct-I outperforms ReAct and Plan-Act in both overall Score and SR across all four evaluated models (Figure 5a). Relative to the stronger baseline for each model and metric, Score improves by 0.7–0.9 percentage points and SR by 0.8–1.0 points. These results extend the gains from game environments to desktop interaction. Appendix Table 17 reports the complete category-level results.

4.3 Ablation Study

Conflict feedback and planner participation improve reconciliation. We compare the full intervention with correction disabled, correction without specific conflict feedback, and correction restricted to the actor. The full method achieves the highest task success and repair rates for both GPT configurations on Sokoban and MiniGrid (Figure 5b). For the heterogeneous pair on Sokoban, repair reaches 98.0%, compared with 90.8% without specific feedback and 37.7% with actor-only correction. These comparisons support involving both roles and identifying their disagreements.

Reconciliation improves on assertion prompting and resampling. Figure 5c compares ConPAct-I with assertion-only prompting, resampling, and Self-Refine (Madaan et al., 2023). ConPAct-I achieves higher Sokoban success than assertion prompting and resampling in all three configurations, and higher success than Self-Refine in both homogeneous configurations. The corresponding MiniGrid results show the same ordering.

Supervising and replaying reconciliation improve training. We compare full ConPAct-R with a baseline omitting all three data-curation components and two variants that retain consistency filtering but remove reconciliation supervision or reconciliation replay. Full ConPAct-R achieves higher task



(a) OSWorld. (b) Inference ablation. (c) Method comparison.

Figure 5: OSWorld performance and Sokoban inference-time comparisons. (a) OSWorld Score and SR relative to the stronger of ReAct and Plan-Act within each model, measured in percentage points; point labels give absolute percentages and purple annotations give ConPAct-I gains. (b) Task success (bars, left axis) and disagreement repair (lines, right axis) under different correction settings. (c) Task success with assertion prompting, resampling, Self-Refine, and ConPAct-I.

success and lower mismatch than all three alternatives across the evaluated model configurations on Sokoban and MiniGrid (Figure 6a). For 32B on Sokoban, it improves avg@8 from 55.9% to 66.9% and reduces mismatch from 34.7% to 1.4% relative to the baseline.

Complete Sokoban and MiniGrid results for the inference and training-data ablations are reported in Appendices F.4 and F.5, respectively.

4.4 Analysis of State Understanding and Coordination

We examine ConPAct-R through initial state judgments, discussion responses, and the correctness of shared interpretations. Complete results for Sokoban and MiniGrid are reported in Appendix F.6.

Training improves initial state understanding. Across the three Sokoban configurations, ConPAct-R reduces initial mismatch by 22.8–26.6 percentage points and improves each role's assertion correctness by 36.5–48.0 points (Figure 6b). These improvements precede discussion, indicating that the trained models begin coordination with more accurate and compatible state judgments. MiniGrid exhibits the same pattern.

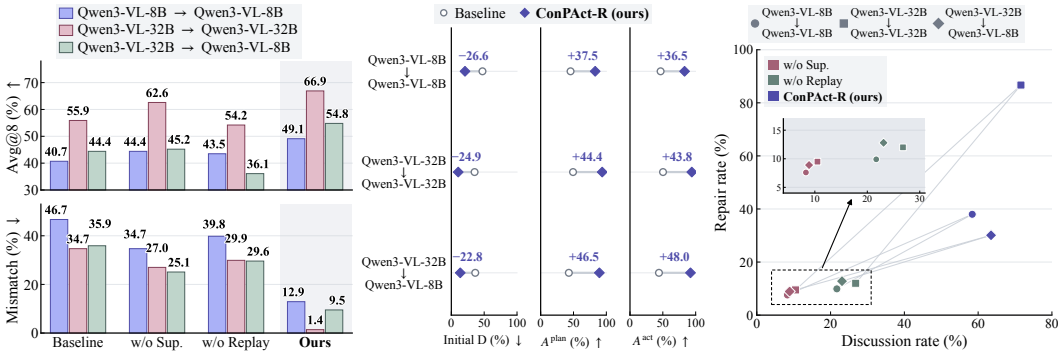
Training improves discussion responses and disagreement repair. Full ConPAct-R yields higher discussion rates and lower residual mismatch than either data ablation across all three model configurations in both environments (Figure 6c). For 32B on Sokoban, the discussion rate reaches 71.6%, while residual mismatch falls to 13.3%. This supports supervising corrective interactions to improve how agents respond to disagreements.

Reconciliation increases correct shared interpretations. To distinguish correct agreement from mere consistency, we examine whether the two roles' shared assertions also match the environment state. Full ConPAct-R has a higher proportion of consistent and correct assertions than either ablation, both before and after discussion, across all three Sokoban configurations. Discussion further increases this proportion by 2.7–5.1 percentage points, reaching 77.7–93.5% (Appendix Table 23). Correct agreement also increases in MiniGrid, supporting the role of reconciliation in improving shared state understanding.

5 Related Work

5.1 Plan-Act Agents

Plan-act agents separate planning from action selection (Erdogan et al., 2025). Existing methods improve these capabilities through global guidance (Li et al., 2026c), constrained execution (Jeong et al., 2026), and hierarchical policy optimization (Hu et al., 2025b; Peng et al., 2026). ReCAPA and



(a) SFT data ablation. (b) Initial state understanding. (c) Discussion and repair.

Figure 6: Supervised-training ablations and state analysis on Sokoban. (a) Task success and mismatch under ConPAct-R data ablations; Full includes consistency filtering, reconciliation supervision, and reconciliation replay. (b) Initial mismatch and assertion correctness; annotations show changes in percentage points from the baseline. (c) Discussion and repair rates for ConPAct-R and its data ablations. The components together contribute to the effectiveness of ConPAct.

ContextFlow address cross-level correction and task-stage coordination, respectively (Zeng et al., 2026; Guo et al., 2026), while Agent GPA evaluates plan adherence (Jia et al., 2025). Training also exploits trajectory relabeling (Li et al., 2026b) and multi-agent correction or role specialization (Zhao et al., 2025a; Motwani et al., 2025; Subramaniam et al., 2025).

ConPAct identifies a distinct failure: incompatible interpretations of the same observed facts despite plan adherence. It trains on consistent collaboration and correction segments, retaining earlier conflicts as context for learning reconciliation. Its replay variant reconstructs these interactions without changing recorded actions, enabling existing trajectories to support coordination training.

5.2 State Tracking and Verification in LLM Agents

ECoT and VAGEN train explicit state reasoning (Zawalski et al., 2025; Wang et al., 2025b), while CoEx and EVU maintain and update beliefs (Kim & won Hwang, 2025; Wang et al., 2026). Learned transition models further support planning and policy learning (Luo et al., 2026; Ding et al., 2026). Self-Refine, multiagent debate, and Reflexion guide revision with model-generated feedback (Madaan et al., 2023; Du et al., 2024; Shinn et al., 2023), although self-verification remains limited (Stechly et al., 2025). Other methods score intermediate execution (Chae et al., 2025; Chen et al., 2025a), detect plan-constraint violations (Guo et al., 2024), or validate outputs against planner-defined criteria (Xu et al., 2026).

ConPAct programmatically compares both roles' state reports about the same observation and returns conflicting fields for joint revision. The comparison assumes neither role is correct and requires neither environment ground truth nor an additional verifier-model call. This targeted feedback improves state interpretation and coordination by allowing both roles to revise their judgments.

6 Conclusion

We identify and study planner-actor state mismatch as a coordination failure in which agents interpret task-relevant states differently, even when the actor follows its assigned sub-goal. Our findings show that making these interpretations explicit and reconciling disagreements can improve coordination through both inference-time interaction and training. The online comparison and reconciliation require neither environment ground truth nor an external verifier. Although consistency alone does not ensure correctness, these results support treating shared state understanding as an explicit objective in designing and training collaborative agents. Future work can examine how this approach scales to larger agent teams and more complex environments, where differing observations and interdependent decisions create further challenges for coordination.

AI USE STATEMENT

We used generative AI tools to assist with figure visualization, grammar checking, translation, and code review. As part of our method, we used large language models to generate synthetic training data, including agentic interactions, and to assess agents' adherence to assigned sub-goals using LLM-as-Judge. The LLM usage is detailed in the main text and appendix. The authors reviewed the AI-assisted translations and language edits for accuracy and preservation of the intended meaning, checked the figures against the underlying experimental data, and evaluated the code-review suggestions before adopting them. The authors take full responsibility for the accuracy, originality, and integrity of the paper and its associated artifacts.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*, 2021.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023. CoRL 2022.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

Anthropic. System card: Claude Opus 5. `https://www.anthropic.com/claude-opus-5-system-card`, 2026a.

Anthropic. System card: Claude Sonnet 5. `https://www.anthropic.com/claude-sonnet-5-system-card`, 2026b.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv*, 2511.21631, 2025.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya G. Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *NeurIPS*, 2025. Datasets and Benchmarks Track.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Hee Han, Taeyoon Kwon, Minju Kim, Beong woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-Shepherd: Advancing PRMs for reinforcing web agents. In *NeurIPS*, 2025.

Cong Chen, Kaixiang Ji, Hao Zhong, Muzhi Zhu, Anzhou Li, Guo Gan, Ziyuan Huang, Cheng Zou, Jiajia Liu, Jingdong Chen, Hao Chen, and Chunhua Shen. GUI-shepherd: Reliable process reward and verification for long-sequence GUI tasks. *arXiv preprint arXiv:2509.23738*, 2025a.

Jiaju Chen, Bo Sun, Yuxuan Lu, Yun Wang, Dakuo Wang, and Bingsheng Yao. CollabSim: A CSCW-grounded methodology for investigating collaborative competence of LLM agents through controlled multi-agent experiments. *arXiv preprint arXiv:2606.06399*, 2026.

Junjie Chen, Haitao Li, Jingli Yang, Yiqun Liu, and Qingyao Ai. Enhancing LLM-based agents via global planning and hierarchical execution. *arXiv preprint arXiv:2504.16563*, 2025b.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv*, 2107.03374, 2021.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv*, 1604.06174, 2016.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A platform to study the sample efficiency of grounded language learning. In *ICLR*, 2019.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & Miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *NeurIPS*, 2023. Datasets and Benchmarks Track.

Herbert H. Clark and Susan E. Brennan. Grounding in communication. In *Perspectives on Socially Shared Cognition*. American Psychological Association, 1991.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *ICML*, 2020.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*, 2022.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Hang Ding, Peidong Liu, Junqiao Wang, Ziwei Ji, Meng Cao, Rongzhao Zhang, Lynn Ai, Eric Yang, Tianyu Shi, and Lei Yu. DynaWeb: Model-based reinforcement learning of web agents. *arXiv preprint arXiv:2601.22149*, 2026.

Vardhan Dongre and Dilek Hakkani-Tür. Embodied multi-agent coordination by aligning world models through dialogue. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2026.

Zhihao Dou, Qinjian Zhao, Zhongwei Wan, Dinggen Zhang, Weida Wang, Benteng Chen, Towsif Raiyan, Qingtao Pan, Yang Ouyang, Chaoda Song, Zhiqiang Gao, Shufei Zhang, and Sumon Biswas. Plan then action: High-level planning guidance reinforcement learning for LLM reasoning. In *ICML*, 2026.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *ICML*, 2024.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-Act: Improving planning of agents for long-horizon tasks. In *ICML*, 2025.

Shuhan Guo, Kun Zhang, Haifei Liu, Xingyu Gao, Yongqi Zhang, Yaqing Wang, and Quanming Yao. ContextFlow: Hierarchical task-state alignment for long-horizon embodied agents. *arXiv preprint arXiv:2605.19314*, 2026.

Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. DoReMi: Grounding language model by detecting and recovering from plan-execution misalignment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *ICLR*, 2022.

Yifei He, Pranit Chawla, Yaser Souri, Subhojit Som, and Xia Song. WebSTAR: Scalable data synthesis for computer use agents with step-level filtering. In *ACL*, 2026.

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-Bench: How good are LLMs at playing games? *arXiv preprint arXiv:2505.15146*, 2025a.

Zican Hu, Wei Liu, Xiaoye Qu, Xiangyu Yue, Chunlin Chen, Zhi Wang, and Yu Cheng. Divide and conquer: Grounding LLMs as efficient decision-making agents via offline hierarchical reinforcement learning. In *ICML*, 2025b.

Jongwon Jeong, Jungtaek Kim, and Kangwook Lee. TAPE: Tool-guided adaptive planning and constrained execution in language model agents. *arXiv preprint arXiv:2602.19633*, 2026.

Allison Sihan Jia, Daniel Huang, Nikhil Vytla, Seung Won Wilson Yoo, Nirvika Choudhury, Shayak Sen, John C. Mitchell, and Anupam Datta. What is your agent's GPA? a framework for evaluating agent Goal-Plan-Action alignment. *arXiv preprint arXiv:2510.08847*, 2025.

Jiajie Jin, Xiaoxi Li, Yuyao Zhang, Guanting Dong, Zhao Yang, Yutao Zhu, and Zhicheng Dou. HiRA: Decoupling planning and execution with hierarchical reasoning in deep search. In *ACM SIGIR*, 2026.

Byeongjin Kim, Gyuwan Kim, and Seo Yeon Park. PPA-plan: Proactive pitfall avoidance for reliable planning in long-context LLM reasoning. In *ACL*, 2026.

Minsoo Kim and Seung won Hwang. CoEx – co-evolving world-model and exploration. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

Kimi Team. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Huao Li, Yu Quan Chong, Simon Stepputtis, Joseph Campbell, Dana Hughes, Michael Lewis, and Katia Sycara. Theory of mind for multi-agent collaboration via large language models. In *EMNLP*, 2023.

Yunfan Li, Bingbing Xu, Xueyun Tian, Xiucheng Xu, and Huawei Shen. Beyond entangled planning: Task-decoupled planning for long-horizon agents. *arXiv preprint arXiv:2601.07577*, 2026a.

Zichao Li, Gang Wu, Zichao Wang, Ruiyi Zhang, Wanrong Zhu, Ryan A. Rossi, Vlad I. Morariu, and Jihyung Kil. Spinning straw into gold: Relabeling LLM agent trajectories in hindsight for successful demonstrations. In *ICLR*, 2026b.

Ziyue Li, Yuan Chang, Yuanbo Kong, Gaihong Yu, and Xiaoqiu Le. HiPlan: Hierarchical planning for web agents with adaptive global-local guidance. *Data Intelligence*, 2026c.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. SwiftSage: A generative agent with fast and slow thinking for complex interactive tasks. In *NeurIPS*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023. Datasets and Benchmarks Track.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Dezhao Luo, Bohan Tang, Kang Li, Georgios Papoudakis, Jifei Song, Shaogang Gong, Jianye Hao, Jun Wang, and Kun Shao. ViMo: A generative visual GUI world model for app agents. In *ICLR*, 2026.

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 *NeurIPS*, 2023.

Gianni Molinari and Fabio Ciravegna. Reason-plan-ReAct: A reasoner-planner supervising a ReAct executor for complex enterprise tasks. In *AAAI 2026 Workshop on Agentic AI Benchmarks and Applications for Enterprise Tasks*, 2026.

Moonshot AI. Kimi K2.6. `https://huggingface.co/moonshotai/Kimi-K2.6`, 2026.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip H. S. Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. MALT: Improving reasoning with multi-agent LLM training. In *Conference on Language Modeling (COLM)*, 2025.

OpenAI. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2026a.

OpenAI. GPT-5.5 system card. `https://openai.com/index/gpt-5-5-system-card/`, 2026b.

OpenAI. GPT-5.6 system card. `https://deploymentsafety.openai.com/gpt-5-6`, 2026c.

Davide Paglieri, Bartłomiej Cupiał, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning when to plan: Efficiently allocating test-time compute for LLM agents. *arXiv preprint arXiv:2509.03581*, 2025a.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *ICLR*, 2025b.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. HiPER: Hierarchical plan–execute RL for multi-turn LLM agents. In *ICML*, 2026.

Shuwen Qiu, Mingdian Liu, Hengli Li, Song-Chun Zhu, and Zilong Zheng. MindDial: Enhancing conversational agents with theory-of-mind for common ground alignment and negotiation. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents. `https://qwen.ai/blog?id=qwen3.5`, 2026a.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model. `https://qwen.ai/blog?id=qwen3.6-27b`, 2026b.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for Android device control. In *NeurIPS*, 2023. Datasets and Benchmarks Track.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *ICLR*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *ICLR*, 2021.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. In *ICLR*, 2025.

Vighnesh Subramaniam, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. In *ICLR*, 2025.

Haoran Tan, Zeyu Zhang, Chen Ma, Tianze Liu, Quanyu Dai, and Xu Chen. From coarse to fine: Self-adaptive hierarchical planning for LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *NeurIPS*, 2023.

Madhumitha Venkatesan, Shicheng Wen, Jiajing Guo, Jorge Piazentin Ono, Liu Ren, and Dongyu Liu. Plover: Steering GUI agents through plan-centric interaction. *arXiv preprint arXiv:2607.15193*, 2026.

Hanlin Wang, Jian Wang, Chak Tou Leong, and Wenjie Li. STeCa: Step-level trajectory calibration for LLM agent learning. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025a.

Hanlin Wang, Chak Tou Leong, Jian Wang, and Wenjie Li. Seeing isn't believing: Mitigating belief inertia via active intervention in embodied agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Fei-Fei Li, Yejin Choi, and Manling Li. VAGEN: Reinforcing world model reasoning for multi-turn VLM agents. In *NeurIPS*, 2025b.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *NeurIPS*, 2024. Datasets and Benchmarks Track.

Tianyang Xu, Dan Zhang, Kushan Mitra, and Estevam Hruschka. Verification-aware planning for multi-agent systems. In *EACL*, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *ICLR*, 2025.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3157–3181. PMLR, 2025. CoRL 2024.

Xiyin Zeng, Yuyu Sun, Haoyang Li, Shouqiang Liu, and Hao Wang. ReCAPA: Hierarchical predictive correction to mitigate cascading failures. In *ICLR*, 2026.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Y. Zou. SiriuS: Self-improving multi-agent systems via bootstrapped reasoning. In *NeurIPS*, 2025a.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. SWIFT: A scalable lightweight infrastructure for fine-tuning. In *AAAI*, 2025b.

Shuai Zhen, Yanhua Yu, Ruopei Guo, Nan Cheng, and Yang Deng. Hierarchical reinforcement learning with augmented step-level transitions for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *NeurIPS*, 2023.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *ICLR*, 2024.

Appendix Contents

A Additional Related Work **18**

A.1 Plan-Act Agents . . . 18

A.2 State Tracking and Verification in LLM Agents . . . 18

A.3 Common Ground and Belief Alignment . . . 18

B State Assertions **18**

B.1 Plan-Act Protocol . . . 18

B.2 Assertion Format . . . 19

B.3 Metrics . . . 21

C Experimental Setup **23**

C.1 Environments . . . 23

C.2 Models and Baselines . . . 26

C.3 Training and Evaluation . . . 26

D State Mismatch Analysis **28**

D.1 Mismatch Prevalence . . . 28

D.2 Control Experiments . . . 29

D.3 Failure Analysis . . . 31

D.4 State Interventions . . . 31

E Method Details **33**

E.1 ConPAct-I . . . 33

E.2 ConPAct-S and ConPAct-R . . . 33

E.3 Prompts . . . 34

F Additional Results **39**

F.1 Inference-Time Results . . . 39

F.2 Supervised-Training Results . . . 40

F.3 OSWorld Results . . . 40

F.4 Inference Ablations . . . 41

F.5 Data Ablations . . . 43

F.6 Training Analysis . . . 45

G Case Studies **47**

G.1 Sokoban . . . 47

G.2 MiniGrid 48
G.3 Crafter 49
G.4 Procgen Miner 50

H Discussion **51**

H.1 Inference Cost 51
H.2 Interpretation and Scope 51

A Additional Related Work

A.1 Plan-Act Agents

Plan-act systems separate sub-goal selection from action execution, with prior work addressing global guidance, constrained execution, and hierarchical learning (Erdogan et al., 2025; Li et al., 2026c; Jeong et al., 2026; Hu et al., 2025b). Trajectory relabeling and multi-agent training also turn past interactions into supervision (Li et al., 2026b; Zhao et al., 2025a; Motwani et al., 2025; Subramaniam et al., 2025). ConPAct focuses on whether the two roles interpret the same observed state compatibly, and uses their explicit contradictions to guide correction and training.

A.2 State Tracking and Verification in LLM Agents

State-grounded reasoning and belief updates support agent decisions (Zawalski et al., 2025; Wang et al., 2025b; Kim & won Hwang, 2025; Wang et al., 2026). Other methods guide revision with critiques, process scores, or checks against planning criteria (Madaan et al., 2023; Chae et al., 2025; Guo et al., 2024; Xu et al., 2026). ConPAct-I instead compares the roles' reports directly: neither report is treated as ground truth, and detecting a contradiction requires no additional verifier-model call. The teacher review used to curate ConPAct-R targets is a separate offline training-data operation.

A.3 Common Ground and Belief Alignment

Common ground concerns the information needed for coordinated interaction (Clark & Brennan, 1991). Recent work examines inter-agent failures, belief-sensitive dialogue, and shared understanding (Cemri et al., 2025; Qiu et al., 2024; Chen et al., 2026; Dongre & Hakkani-Tür, 2026). Our setting isolates disagreements about the same task-relevant observed facts at planner–actor handoffs and evaluates both their reconciliation and the correctness of the resulting agreement.

B State Assertions

B.1 Plan-Act Protocol

Roles and observations. The planner selects a sub-goal; the actor chooses actions and reports execution status. Let t index model calls and k index executed environment actions. Several calls may share observation o^k; only an executed action advances the environment to o^{k+1}. An execution segment consists of actor turns pursuing one adopted sub-goal. Actor feedback, including completion or execution difficulties, triggers replanning.

Plan-Act baseline. The planner returns a `<subgoal>`; the actor returns an `<action>` and a `<status>`. Its statuses are `CONTINUE`, `ACHIEVED`, `IMPOSSIBLE`, `STUCK`, and `CHANGED`. The actor's proposed action executes before a non-`CONTINUE` status returns control to the planner. Consequently, passive assertion comparison uses the planner-to-actor handoff before that action, when both roles have the same observation; it never pairs reports across an environment transition. The ordinary interface passes sub-goals and status feedback rather than full reasoning traces.

ConPAct interface. A planner reply contains `<assertion>`, `<message>`, and `<subgoal>`; an actor reply additionally supplies `<action>` and `<status>`. The actor uses `CONTINUE` for an executable action, `DISCUSS` to request reconsideration, and `ACHIEVED` when sub-goal completion is already visible. Both `DISCUSS` and `ACHIEVED` require action `NONE` and leave the observation unchanged. The planner then retains or revises its sub-goal using the actor's feedback. ConPAct therefore supports same-observation comparisons in both handoff directions. Overall task completion is determined by the environment, not by the actor's completion claim.

Independent measurement and reconciliation. The receiving role does not see the other role's assertion when the initial pair is elicited; ordinary sub-goals and feedback remain available. After an initial contradiction is detected, reconciliation exposes the disputed fields and the roles' reports for reassessment. All actions remain paused during correction. ConPAct-I returns the last actor reply when correction ends and does not insert an additional actor call solely to choose a replacement

action. The ordinary controller applies its action/status validity rules to this reply; NONE does not execute an environment action. Both roles retain up to three observation frames, with method-specific frame selection and text history described in Appendix C.3.

B.2 Assertion Format

Scope and representation. A state assertion is a JSON object enclosed in <assertion> and </assertion> tags, reported alongside the role's planning or execution response. Both roles use the same environment-specific schema and describe the *current observation*, before the proposed action is executed. The fields capture visible conditions relevant to interpreting sub-goals: object counts and local obstacles in grid tasks, interaction prerequisites and resources in Crafter, visible targets and threats in Procgen, and application context and task evidence in OSWorld. They are compact reports of state interpretation, not complete scene descriptions or predictions of action outcomes. Research on chain-of-thought explanations shows that verbal reports need not faithfully expose a model's reasoning (Turpin et al., 2023). We therefore interpret state assertions as observable proxies and do not assume that they reveal the models' internal beliefs. No schema requires persistent object identifiers, absolute world coordinates, or access to hidden simulator state. Task instructions and current observation metadata may disambiguate what to inspect, but unavailable facts should not be filled in from earlier frames. Prompt templates are provided in Appendix E.3. Categorical domains are listed below. Count fields are nonnegative integers unless a narrower range is explicitly stated; no additional global upper bound is imposed.

Sokoban. The four fields describe puzzle progress and the player's immediate surroundings. box_count counts all boxes, including boxes on targets; target_count counts all target locations, including occupied targets; and on_goal_count counts boxes currently on targets. All three are nonnegative integers. player_neighbors is a four-element array in the fixed screen-relative order *up, right, down, left*, with values cell, wall, box, target, or box_on_target. Here cell means empty floor, and target means a target without a box. These fields are read from the full board and distinguish completion conditions from local movement or pushing prerequisites; they do not encode whether an entire push sequence is feasible.

Crafter. The five fields summarize the local interaction target and the heads-up display (HUD). facing is one of up, right, down, or left in screen coordinates. front identifies the single cell in that direction: walkable, water, lava, stone, tree, coal, iron, diamond, table, furnace, cow, zombie, skeleton, plant, or arrow. walkable merges empty grass, sand, and path; plant covers both growth stages; an arrow must occupy that cell. The moving camera does not establish fixed world coordinates. vitals is the ordered array [health, food, drink, energy]. inventory has exactly the keys wood, stone, coal, and iron. tools has exactly six keys: wood_pickaxe, stone_pickaxe, iron_pickaxe, wood_sword, stone_sword, and iron_sword. Every vital, material count, and tool count is an integer in $\{0, \ldots, 9\}$; a material or tool absent from the HUD is reported as zero. The schema covers resource and tool prerequisites and immediate survival needs, while omitting other facts such as the full arrangement of crafting stations.

MiniGrid / BabyAI. The five fields describe visible keys, door states, and the next cell relative to the agent's orientation. key_count counts keys drawn on the board; a carried key is not drawn and is excluded. front_cell is the cell directly ahead of the agent triangle, using the direction supplied with the current screenshot. Its values are empty, wall, door_locked, door_closed, door_open, key, ball, box, goal, or lava; the outer boundary is wall. locked_door_count, closed_door_count, and open_door_count count locked, closed-but-unlocked, and open doors, respectively. Their visual cues are a filled cell with a keyhole, a hollow double-edged square, and a thin strip along a cell edge. All four counts are nonnegative integers and cover both light and dark rendered cells in the full-board image, not only the highlighted field of view. These fields describe visible movement and interaction conditions; zero visible keys does not establish that the agent is carrying a key.

Procgen. The schema is selected by game identity. Every game has target_visible; every game except maze also has hazard_nearby. Both take yes, no, or unknown, using the referents in Table 3. A hazard is nearby when it is visibly close enough to threaten the player or the next immediate movement; the protocol does not impose a pixel-distance threshold. Miner additionally has

diamonds_visible, a nonnegative integer counting only diamonds in the screenshot, or JSON null when a reliable count is unavailable. Bigfish additionally has my_size: smallest when all other visible fish are larger, middle when both larger and smaller fish are visible, biggest when none is larger, and unknown when comparison is unclear or no other fish is visible. no requires sufficient visible evidence of absence; an obscured or ambiguous view warrants unknown. An off-screen target may still exist, and the absence of a visible threat does not prove that a route is safe.

Table 3: Game-specific referents for Procgen assertions. Maze has no hazard_nearby field.

Game	Target for target_visible	Threat for hazard_nearby
bigfish	At least one visibly smaller fish	A larger fish
chaser	At least one uncollected green orb	A non-vulnerable enemy
coinrun	The goal coin	A saw, enemy, or deadly gap
jumper	The carrot	Spikes
maze	The cheese	Not included in the schema
miner	At least one diamond	A boulder or diamond that could fall onto the player
ninja	The goal mushroom	A bomb or dangerous gap

OSWorld. The four fields refer to the current desktop screenshot and the *original task*, rather than the planner's temporary sub-goal. active_app identifies the foreground application: chrome, writer, calc, impress, gimp, vscode, thunderbird, vlc, files, terminal, settings, desktop, other, or unknown. An application-owned dialog is attributed to its owner when identifiable; other denotes an identifiable application outside the list. overlay takes none, menu, dialog, or unknown, with a visible dialog taking precedence over a menu. A file picker is a dialog, not the files application. target_visible takes yes, partial, no, or unknown: all, some, none, or an indeterminate amount of the task's target objects or regions are visible. goal_state takes met, unmet, or unknown. unmet requires a visibly unsatisfied requirement; met requires the screenshot alone to establish every requirement; otherwise the value is unknown. This visual judgment is distinct from the official task score: a filled input or closed dialog does not prove that a file was saved or that off-screen content is correct.

Extraction, normalization, and validity. The parser extracts the tagged JSON object and checks field types, required keys, array lengths, and the domains above. Integer fields reject booleans, floating-point values, and numeric strings; they are not coerced or clipped. Crafter, MiniGrid, Procgen, and OSWorld require exactly one assertion block, reject duplicate JSON keys at any nesting level, and require exactly the specified fields, including Crafter's nested keys. Their categorical values must match the listed strings exactly, and Markdown fences inside the assertion block are not accepted. The Sokoban schema extractor instead uses the last assertion block, removes optional surrounding JSON fences, retains only its four required fields, and trims and lowercases neighbor labels; thus the prompt spelling Box_on_target becomes box_on_target. Its JSON decoder uses the last occurrence of a duplicate key. ConPAct additionally validates the order and uniqueness of the complete reply's protocol tags, so this schema-level tolerance does not permit repeated assertion blocks in a ConPAct reply. Missing fields or invalid values yield an invalid assertion, not an all-zero or unknown-filled replacement; in the online runners, a required assertion that fails parsing causes a model-output failure. Schema validity only establishes structural admissibility: for example, the Sokoban parser does not enforce cross-field constraints between the three counts, and it does not verify the report against the board.

Definite values and uncertainty. Only Procgen and OSWorld provide unknown sentinels: the literal string unknown for their categorical fields, and null for Procgen miner's diamonds_visible. Sokoban, Crafter, and MiniGrid require definite values in every field. Zero is a definite count, and OSWorld's none, other, and partial are definite categories, not abstentions. Comparison uses parsed values rather than raw JSON text: object-key order is irrelevant, array order matters, and each top-level slot is compared as a whole, including Crafter's arrays and nested objects. For a valid pair, a slot is comparable only if neither role reports its declared unknown sentinel. The result is CONFLICT if any comparable slot differs, AGREE if every slot is definite and matches, and INDETERMINATE otherwise. Thus, matching unknowns do not establish agreement, while a

definite contradiction still counts as a conflict even if another slot is unknown. These comparisons require no ground truth; the code does not provide a reliable assertion-correctness oracle for Procgen or OSWorld, and task scores are not substituted for one.

Pairing and measurement stages. Assertions are paired by environment step, using the handoff protocol above. An initial pair is elicited without exposing the other role's assertion; later reconciliation pairs can use the shared discussion context. Reports across an executed action are never paired. Passive mismatch measurement does not return discrepancy feedback or insert corrective calls. The training analyses distinguish initial judgments from the reports available after model-initiated discussion (Appendix F.6).

B.3 METRICS

Let $\{\tau_i\}_{i=1}^{N}$ denote the evaluated trajectories for one model–environment configuration. Repeated rollouts are separate trajectories. Infrastructure failures are excluded; model-output failures and budget exhaustion remain included. The following rates are computed over their specified eligible samples and displayed as percentages. The mismatch analyses combine the 200 Sokoban and 150 MiniGrid levels before applying these metric-specific denominators; the main benchmark tables use the environment-specific aggregation rules below.

State mismatch. Let $\mathcal{P}$ contain the evaluated handoff pairs (i, t) for which both assertions are valid and every field is definite. Passive measurement uses the independent initial pairs; inference corrections and training analyses explicitly identify their initial or post-discussion event sets. Each pair describes the same observation, $o_{i,t} = o_{i,t+1}$. Using $d_{i,t}$ from Eq. 1, we define

$$D = \frac{1}{|\mathcal{P}|} \sum_{(i,t)\in\mathcal{P}} d_{i,t}. \tag{2}$$

Each eligible pair has equal weight, regardless of its trajectory length or outcome. Discussion-conditional and paired pre/post statistics use the event sets specified in Appendix F.6.

State correctness. Within pair (i, t), let $\mathbf{v}_{i,t}^{\text{plan}}$ and $\mathbf{v}_{i,t}^{\text{act}}$ denote the two assertions, indexed by role. Let $\mathbf{v}_{i,t}^{*}$ be the ground-truth assertion for their shared observation. Define $\mathcal{G} \subseteq \mathcal{P}$ as the pairs with valid, fully definite ground truth. For $r \in \{\text{plan}, \text{act}\}$, exact-match accuracy is

$$A^{r} = \frac{1}{|\mathcal{G}|} \sum_{(i,t)\in\mathcal{G}} \mathbf{1}\left[\forall f \in \mathcal{F} : \mathbf{v}_{i,t}^{r}[f] = \mathbf{v}_{i,t}^{*}[f]\right]. \tag{3}$$

All fields must match; there is no partial credit. Both roles use the same denominator. Sokoban ground truth is obtained by replaying executed actions from the initial board. MiniGrid ground truth is extracted from the recorded initial state and subsequent environment states, aligned with the current observation. Keys and doors are counted on the board; the front cell is determined by the agent's position and direction. Carried keys are excluded. Ground truth follows the schemas in Appendix B.2 and is hidden from both roles during passive measurement.

Sub-goal adherence. Let $q_{i,j}$ be the judge's verdict for execution segment j of τ_i. Let $\mathcal{J} = \{(i, j) : q_{i,j} \in \{\texttt{FOLLOW}, \texttt{VIOLATE}\}\}$. We define

$$H = \frac{1}{|\mathcal{J}|} \sum_{(i,j)\in\mathcal{J}} \mathbf{1}[q_{i,j} = \texttt{FOLLOW}]. \tag{4}$$

Each segment receives one verdict and equal weight. We use GPT-5.6-sol (OpenAI, 2026c) as an LLM judge (Zheng et al., 2023). Our rubric targets plan–action adherence, an aspect of agent alignment examined by Agent GPA (Jia et al., 2025). Its input contains the assigned sub-goal and, for each segment step, the ground-truth state as text followed by the actor's complete response. It receives no screenshots, planner reasoning, or other segments. `FOLLOW` means that the actions pursue the sub-goal, even if the actor misreads the state or fails to complete it. `VIOLATE` means that the actions contradict the sub-goal or pursue another goal. `UNKNOWN` is reserved for sub-goals with no executable content; it is not used for close decisions. Segments truncated by termination, budget exhaustion, or model-output failure are judged from the available trace. Adherence does not require a valid assertion pair.

Task success. A trajectory succeeds when the environment confirms task completion within the rollout budget. We define

$$S = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\tau_i \text{ succeeds}]. \tag{5}$$

An actor's completion claim alone does not establish success. This trajectory-level aggregation is used for the state analyses; environment-specific benchmark metrics are defined below.

Exclusions and empty denominators. Invalid assertions and pairs with any unknown field are excluded from D and both correctness rates. A definite conflict in a partially unknown pair is recorded separately and does not enter D. Missing ground truth excludes a pair from correctness only. Missing sub-goals, `UNKNOWN` verdicts, and missing or malformed judge responses are excluded from H. These exclusions do not remove a trajectory from S. Eligible counts and exclusion counts are recorded separately. An empty denominator yields an undefined metric, not zero. Trajectory-level D_i and H_i use the same definitions restricted to τ_i.

Sokoban. For each of the $L = 200$ puzzles, let $c_\ell \in \{0, \ldots, 8\}$ be the number of successful rollouts. A rollout succeeds when every box is on a target. We report

$$\text{avg@8} = \frac{1}{L}\sum_{\ell=1}^{L}\frac{c_\ell}{8}, \qquad \text{pass@8} = \frac{1}{L}\sum_{\ell=1}^{L}\mathbf{1}[c_\ell \geq 1], \qquad \text{pass}^8 = \frac{1}{L}\sum_{\ell=1}^{L}\mathbf{1}[c_\ell = 8]. \tag{6}$$

All three are displayed as percentages. The latter two specialize the pass@k criterion (Chen et al., 2021) and the pass^k reliability criterion (Yao et al., 2025) to eight attempts: at least one success and success on all eight attempts, respectively.

Crafter. Let $a_{j,i}$ indicate whether episode j unlocks achievement i, with $i \in \{1, \ldots, 22\}$, and let N denote the number of evaluated episodes. The achievement success rate is $s_i = 100N^{-1}\sum_j a_{j,i}$. We compute

$$\text{Score} = \exp\left(\frac{1}{22}\sum_{i=1}^{22}\log(1+s_i)\right) - 1, \qquad \text{Progress} = \frac{100}{22N}\sum_{j=1}^{N}\sum_{i=1}^{22} a_{j,i}. \tag{7}$$

Score follows the Crafter benchmark (Hafner, 2022) and is computed from the achievement success rates over the full evaluation set. Progress is the mean proportion of unlocked achievements, following BALROG (Paglieri et al., 2025b). Return is the mean undiscounted sum of native rewards per episode, including rewards associated with changes in health.

Procgen. For game g, we average the undiscounted episode returns over its 20 levels and eight rollouts per level, obtaining $\bar{G}_g$. The normalized return is

$$z_g = \frac{\bar{G}_g - R_{\min,g}}{R_{\max,g} - R_{\min,g}}, \tag{8}$$

using the easy-mode reference bounds from Cobbe et al. (2020), listed in Table 5. Returns include rewards up to environment termination or budget exhaustion, and normalized values are left unclipped. We use the `rliable` aggregation functions (Agarwal et al., 2021) on the run-by-game score matrix: IQM is the 25%-trimmed mean of its entries, Mean averages the per-game means, and Median takes the median of the per-game means. For a single evaluated checkpoint, the matrix has one row and seven columns. All three metrics are multiplied by 100 for display.

MiniGrid / BabyAI. Success means completing the environment's mission; termination through collision, failure, or timeout does not count as success. For each task ID e, we compute success rate SR_e and mean native episode return $\bar{G}_e$ over all of its instances and rollouts. The headline metrics are

$$\text{SR} = \frac{100}{27}\sum_{e=1}^{27}\text{SR}_e, \qquad \text{Return} = \frac{1}{27}\sum_{e=1}^{27}\bar{G}_e, \tag{9}$$

where SR_e is a fraction in $[0, 1]$. This gives each task ID equal weight despite different instance counts. Steps is the mean number of executed actions pooled over successful episodes only; it is undefined when there are no successes. We retain native rewards and termination rules, including

collision penalties and the native time-dependent success reward. The external action budget does not replace the environment's native maximum episode length in its reward formula.

Rollout accounting. All eight game-environment rollouts contribute to the reported metrics. Model-output failures and budget exhaustion retain the outcome and rewards obtained before stopping. Infrastructure failures are recorded separately and their missing task–rollout slots are rerun; complete metrics require all declared slots. A terminal action's reward and achievements are included before the rollout ends.

OSWorld. We evaluate each of the 361 OSWorld tasks with three rollouts and report mean evaluation score and full success rate as Score and SR, respectively, both in percent. For each task, we average the evaluation scores and full-success indicators across its three rollouts, then average these task-level values over the evaluation set. Overall results therefore give each task equal weight, so category contributions are proportional to their task counts. These task-performance measures are distinct from the state-assertion correctness used in the state analyses. The category-level results are provided in Appendix F.3.

C Experimental Setup

C.1 Environments

Table 4 summarizes the source-task pools used for data collection and the fixed evaluation sets. Each game-environment evaluation instance is run eight times, with model histories cleared and the environment reset to the specified instance before each rollout. Source-instance counts describe the tasks used to collect teacher trajectories; response-level training counts depend on filtering and reconstruction. Models are trained separately for each environment.

Table 4: Environment splits and evaluation budgets. A limits executed actions and C limits total logical model calls per rollout. Each instance has eight evaluation rollouts. MiniGrid budgets are specified per task in Table 6.

Environment	Source instances	Evaluation instances	Evaluation rollouts	A	C
Sokoban	100	200	1,600	60	60
Crafter	100	200	1,600	160	160
Procgen	100	140	1,120	160	160
MiniGrid / BabyAI	108	150	1,200	60–200	60–200

Sokoban. We use the GamingAgent Sokoban implementation (Hu et al., 2025a) through our sandbox interface. Our fixed Stage 1.5 evaluation split contains 100 one-box and 100 two-box puzzles. Its boards comprise 68 puzzles of size 6×6, 66 of size 7×7, and 66 of size 8×8. The source pool contains 50 one-box and 50 two-box puzzles selected from the training split. The agent observes the full board image enlarged by a factor of four with nearest-neighbor interpolation and chooses one of four directions: up, down, left, or right. The task ends successfully when all boxes occupy targets. An action denotes one agent-level directional move; the sandbox implements it through a push command followed by the corresponding direction command.

Crafter. For Crafter (Hafner, 2022), we use source seeds 0–99 and evaluation seeds 4000–4199. The agent collects resources, crafts tools, and survives while unlocking 22 achievements. It observes the native 640×640 local view with the heads-up display (HUD). The action space contains 17 actions for movement, interaction, sleeping, placing objects, and crafting tools or weapons. Each rollout uses a fresh environment instance with the specified seed and the same reset sequence. A rollout ends at native environment termination or when a budget is exhausted; partial achievement progress and accumulated reward are retained.

Procgen. We evaluate seven Procgen games (Cobbe et al., 2020) in easy mode: bigfish, chaser, coinrun, jumper, maze, miner, and ninja. For every game, the evaluation levels have `start_level`=100–119 and `num_levels`=1. The 100 source instances use levels selected from 0–49, with the allocation in Table 5. The 512×512 sandbox screenshot is converted to RGB and

enlarged by a factor of two using nearest-neighbor interpolation, yielding a 1024×1024 model input. Actions use a game-specific subset of the 15 native action indices. The adapter introduces no action repeat. A rollout stops at the first episode termination or budget limit; any automatic reset to a new episode is excluded from the completed rollout.

Table 5: Procgen task allocation and easy-mode normalization bounds from Cobbe et al. (2020). Each game has 20 evaluation levels with IDs 100–119. Source levels are selected from IDs 0–49.

Game	Source levels	Evaluation levels	$R_{\min}$	$R_{\max}$
bigfish	14	20	1	40
chaser	14	20	0.5	13
coinrun	15	20	5	10
jumper	14	20	3	10
maze	14	20	5	10
miner	15	20	1.5	13
ninja	14	20	3.5	10

MiniGrid / BabyAI. The evaluation set comprises 150 instances from 27 MiniGrid and BabyAI task IDs (Chevalier-Boisvert et al., 2023; 2019), including navigation, object retrieval, door unlocking, object placement, and obstacle avoidance. Table 6 lists the exact seeds and budgets. The source pool contains four instances per task ID, for 108 instances in total, with selected seeds between 20 and 38. The source and evaluation manifests have no overlap in task–seed pairs or recorded initial-layout hashes. Layouts are fixed using MiniGrid 2.3.1 and a single seeded reset per fresh instance. We render the full board with 32-pixel tiles and visibility highlighting, convert it to RGB, and enlarge it by a factor of three using nearest-neighbor interpolation. For example, 5×5 and 8×8 boards yield 480×480 and 768×768 images. The current mission and agent direction are provided as text alongside the image. The action vocabulary is left, right, forward, pickup, drop, toggle, and done, restricted to the native action space of each task; Dynamic-Obstacles permits only left, right, and forward. Native termination and truncation are preserved.

Table 6: MiniGrid and BabyAI evaluation tasks. Seed offsets are added to 1000; each resulting instance has eight rollouts. The last column gives the common action and model-call limit for each rollout.

Task ID	Instances	Seed offsets	$A = C$
BabyAI-BlockedUnlockPickup-v0	5	0, 1, 2, 3, 4	136
BabyAI-GoTo-v0	5	0, 1, 2, 3, 4	200
BabyAI-GoToDoor-v0	5	0, 1, 2, 3, 4	84
BabyAI-GoToLocal-v0	5	0, 1, 2, 3, 4	60
BabyAI-GoToObj-v0	5	0, 1, 2, 3, 4	60
BabyAI-GoToRedBall-v0	5	0, 1, 2, 3, 4	60
BabyAI-GoToRedBallGrey-v0	5	0, 1, 2, 3, 4	128
BabyAI-GoToRedBallNoDists-v0	5	0, 1, 2, 3, 4	60
BabyAI-GoToRedBlueBall-v0	6	0, 1, 2, 3, 4, 5	68
BabyAI-UnlockPickup-v0	6	0, 1, 2, 3, 4, 5	80
MiniGrid-BlockedUnlockPickup-v0	6	0, 1, 2, 3, 4, 5	192
MiniGrid-DoorKey-5x5-v0	6	0, 1, 3, 4, 7, 10	92
MiniGrid-DoorKey-6x6-v0	6	0, 1, 2, 4, 5, 6	60
MiniGrid-DoorKey-8x8-v0	6	0, 1, 2, 3, 4, 5	60
MiniGrid-Dynamic-Obstacles-5x5-v0	2	3, 7	60
MiniGrid-Fetch-5x5-N2-v0	6	0, 1, 2, 3, 4, 5	60
MiniGrid-Fetch-8x8-N3-v0	6	0, 1, 2, 3, 4, 5	60
MiniGrid-FourRooms-v0	6	0, 1, 2, 3, 4, 5	68
MiniGrid-GoToDoor-5x5-v0	6	0, 1, 2, 3, 4, 5	84
MiniGrid-GoToObject-6x6-N2-v0	6	0, 1, 2, 3, 4, 5	60
MiniGrid-LavaCrossingS9N1-v0	6	4, 5, 6, 7, 9, 13	60
MiniGrid-PutNear-6x6-N2-v0	6	0, 1, 2, 3, 4, 5	60
MiniGrid-PutNear-8x8-N3-v0	6	0, 1, 2, 3, 4, 5	60
MiniGrid-RedBlueDoors-6x6-v0	6	0, 2, 3, 4, 5, 6	60
MiniGrid-RedBlueDoors-8x8-v0	6	0, 1, 2, 3, 4, 5	124
MiniGrid-Unlock-v0	6	0, 1, 2, 3, 4, 5	60
MiniGrid-UnlockPickup-v0	6	0, 1, 2, 3, 4, 5	76

C.2 Models and Baselines

Models. The mismatch analyses cover Claude Opus-5 and Sonnet-5 (Anthropic, 2026a;b); GPT-5.6-sol, GPT-5.6-terra, and GPT-5.5 (OpenAI, 2026c;b); Kimi K3 and K2.6 (Kimi Team, 2026; Moonshot AI, 2026); and the Qwen model sizes listed in the result tables (Qwen Team, 2026b;a; Bai et al., 2025). Main inference comparisons use the four heterogeneous pairs in Table 1. Training uses Qwen3-VL-8B and Qwen3-VL-32B in homogeneous configurations and a 32B planner with an 8B actor. GPT-5.6-sol is the teacher for data construction and the judge for annotation and offline target review.

Inference baselines. ReAct (Yao et al., 2023) selects one environment action per response; ReAct (Planner) and ReAct (Actor) use the corresponding role's model alone. Plan-Act (Erdogan et al., 2025) uses the dynamic sub-goal/status protocol in Appendix B.1 without assertion-based correction. HiPlan (Li et al., 2026c) supplies a global milestone guide and local hints to the actor; the planner model produces the guide and hints. TAPE (Jeong et al., 2026) uses candidate plans and a state–action graph to constrain execution, replanning when the current observation no longer matches the planned state. All decision-making calls, including internal planning and candidate generation, consume the shared rollout budget.

Supervised baselines. ReAct SFT supervises a single model's reasoning and actions on retained teacher demonstrations. Plan-Act SFT supervises planner sub-goals and actor actions/statuses and retains dynamic replanning at evaluation. PAA (Erdogan et al., 2025) reconstructs a fixed plan from an execution trace: the planner is called at initialization and the actor follows that plan. HSL (Li et al., 2026b) relabels achieved goals and supervises goal-relevant recorded actions. WebSTAR (He et al., 2026) applies teacher-based step filtering to the source demonstrations. ECoT (Zawalski et al., 2025) augments planning and action targets with state-grounded descriptions, without comparing the roles' assertions or adding ConPAct reconciliation. All training methods use the same optimization settings and the token-budget matching described below.

C.3 Training and Evaluation

Teacher and student models. We use GPT-5.6-sol (OpenAI, 2026c) with its default thinking effort as the teacher for constructing supervision. Students are initialized from Qwen3-VL-8B-Instruct or Qwen3-VL-32B-Instruct (Bai et al., 2025). For homogeneous ConPAct-S/R configurations, planner and actor examples are pooled to fine-tune a single checkpoint, which serves both roles at evaluation. For the heterogeneous configuration, the 32B planner and 8B actor are fine-tuned separately on their respective role-specific examples.

Teacher trajectories and data selection. For each source instance in Table 4, GPT-5.6-sol generates two rollouts. The ReAct demonstration pool retains at most one successful or qualifying progress trajectory per source instance; these demonstrations supply ReAct-based supervised baselines and the recorded traces for ConPAct-R. ConPAct-S instead samples both planner and actor with GPT-5.6-sol under ConPAct-I and selects consistent collaboration or repaired segments from both successful and failed trajectories. At most one sampled collaboration trajectory per source instance supplies these segments. Thus, trajectory-level demonstration selection and response-level consistency filtering are separate operations. ConPAct-R uses GPT-5.6-sol to reconstruct sub-goals and corrective interactions from the retained ReAct traces and to review candidate targets using only the student's available input.

Environment-specific training and matched budgets. We train a separate checkpoint for each environment, using training seed 42 and selecting the final checkpoint. Homogeneous configurations pool planner and actor examples into one model; heterogeneous configurations train the 32B planner and 8B actor on their role-specific examples. For each environment and student size, three epochs of ReAct SFT define the reference training-token budget. The other training methods and data ablations are matched to that total token budget using the same optimization settings. This matches training exposure across datasets with different amounts of dialogue and supervision.

Optimization. We use `ms-swift` (Zhao et al., 2025b) with AdamW (Loshchilov & Hutter, 2019), cosine decay (Loshchilov & Hutter, 2017), FlashAttention (Dao et al., 2022), gradient checkpointing (Chen et al., 2016), and ZeRO (Rajbhandari et al., 2020). Only the language-model backbone is updated; the vision encoder and projector are frozen. The effective batch size is 64 on eight H200

GPUs. For the shorter sequence setting, 8B/32B use per-device microbatches of 8/4 and gradient accumulation of 1/2; the longer setting uses microbatch 1 and accumulation 8 for both sizes. Examples exceeding the sequence limit are discarded, and sequence packing is disabled. The complete settings are listed in Table 7.

Table 7: Supervised fine-tuning settings shared by the training methods. Token budgets are matched to the three-epoch ReAct SFT reference for each environment and model size.

Setting	Value
Trainable parameters	Language-model backbone; vision encoder/projector frozen
Reference schedule	3 epochs of ReAct SFT; token-matched across methods
Training seed	42
Observation window	3 frames, including the current observation
Optimizer	AdamW (fused)
Learning rate	1×10^{-5}
Learning-rate schedule	Cosine decay
Warmup ratio	0.05
Adam (β_1, β_2)	$(0.9, 0.95)$
Adam ϵ	10^{-8}
Weight decay	0.1
Maximum gradient norm	1.0
Effective batch size	64 examples
Training hardware	8 NVIDIA H200 GPUs
Training precision	BF16
Maximum sequence length	6,144 (Sokoban, Crafter); 16,384 (Procgen, MiniGrid)
Maximum image tokens	1,024 per image
Attention backend	FlashAttention
Gradient checkpointing	Enabled
DeepSpeed	ZeRO-1 (8B); ZeRO-3 (32B)
Sequence packing	Disabled
Checkpoint selection	Final checkpoint

Loss masks. Each training example supervises one complete assistant response. System instructions, user text, images, and earlier interaction context receive no loss. All tokens in the retained response receive the same weight, including state assertions and role-specific output fields. Table 8 specifies which responses are retained by each construction method. An excluded response may remain in the input history but is not used as a target.

Table 8: Response selection for supervised training. Loss is applied only to the retained target response.

Method	Supervised targets
ReAct / Plan-Act / ECoT	Complete valid responses from retained demonstrations or progress prefixes.
PAA	The planner's annotated plan and the actor's action-only responses.
HSL	Expert responses and goal-relevant actions from hindsight examples; irrelevant actions are excluded.
WebSTAR	Original responses for actions with judge scores greater than 5; other responses are excluded.
ConPAct-S	Valid responses from consistent segments and successful corrections; conflicting replies remain context only.
ConPAct-R	Validated responses under the same consistency rule; hypothetical errors and recorded nonoptimal actions are excluded from targets.

Decoding and output limits. For game evaluation, each model call requests one completion with at most 4,000 output tokens. Self-hosted Qwen models use temperature 1.0, top-$p = 1.0$, and top-$k = -1$, which disables top-k truncation. Thinking is disabled through the chat template. For the Claude, GPT, and Kimi API configurations, temperature, top-p, and top-k use the provider defaults,

and the request disables thinking. These evaluation settings apply to both roles and to method-internal model calls. Teacher calls for data construction use the default thinking effort specified above.

Observation and message history. All methods and roles use a maximum of three observation frames at training and evaluation, including the current frame. When fewer frames are available, we use all available frames without padding. Table 9 specifies the frame selection for each method. Plan-Act and ECoT retain the text of older sub-goals after their start frames leave the planner's window. ConPAct retains the exchanged replies on the current observation, the active planner proposal, and the last executed actor reply. After an action, other discussion from previous observations is removed.

Table 9: The common three-frame observation limit. A planner called only at initialization receives the single available frame. Textual task instructions and active plans are retained separately.

Method	Planner input	Actor input
ReAct / HSL / WebSTAR	No separate planner	Up to 3 recent frames
Plan-Act / Plan-Act SFT	Current frame plus up to 2 sub-goal-start frames	Up to 3 recent frames
PAA	Initial frame; called once	Up to 3 recent frames
ECoT	Current frame plus up to 2 sub-goal-start frames	Up to 3 recent frames
HiPlan	Initial frame for the global plan; up to 3 recent frames for hints	Up to 3 recent frames
TAPE	Up to 3 recent frames for visual calls	Up to 3 recent frames
ConPAct-I / ConPAct-S/R	Up to 3 recent frames	Up to 3 recent frames

Evaluation of trained models. ConPAct-S/R use their learned discussion behavior without the system-triggered ConPAct-I intervention. The automatic assertion-checking and resampling loop, assertion injection, and action gating are disabled. The actor can initiate discussion through its predicted `DISCUSS` status, which hands control to the planner without executing an action. Both roles use the common three-frame observation window. All resulting model calls share the rollout budget described below.

Interaction budgets. For the four game environments, all methods use the same per-instance action and model-call limits in Tables 4 and 6, with $A = C$. An executed action counts toward A; planning, control handoffs, and corrective discussion do not advance the environment. Every logical model completion used for decision-making counts toward C, including calls by either role, discussion turns, and method-internal candidate sampling or resampling. Thus, the call limit is shared across roles and is not a separate allowance for each role. Network retries are tracked separately, and offline diagnostic judgments are excluded from the decision budget. The rollout ends when the environment terminates or either budget is exhausted, after recording the last executed action's outcome. The eight rollouts of an instance share its task configuration and initial-state seed, while histories are cleared between rollouts.

D State Mismatch Analysis

D.1 Mismatch Prevalence

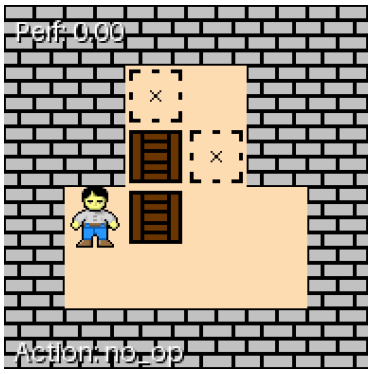
Evaluation pool. We analyze the combined set of 200 Sokoban and 150 MiniGrid levels, with eight rollouts per level and the 16 planner–actor configurations in Table 10. The table combines levels from both environments rather than averaging two equally weighted environment scores. Task success averages the trajectories in this combined pool; mismatch, correctness, and adherence use the eligible pairs or segments defined in Appendix B.3. This analysis-level task-success rate is distinct from the task-ID-macro-averaged MiniGrid SR in the main benchmark tables. Assertions are measured passively at same-observation handoffs, without exposing the other role's assertion or returning conflict feedback.

Prevalence and adherence. The configurations include both homogeneous and heterogeneous model pairs. Mismatch is present even when the same model serves both roles and increases in the evaluated comparisons with a larger capability gap. For Figure 2b, D_i is the trajectory's valid-pair mismatch rate and H_i its judged-segment adherence rate. Only trajectories with nonempty denominators for both measures enter this comparison; trajectories sharing the same (H_i, D_i) are grouped, with marker size showing frequency. Among trajectories with measured $H_i = 100\%$, 82.9% contain at least one mismatch. This is a trajectory-level incidence, not the fraction of mismatched handoffs.

D.2 Control Experiments

Visual versus textual observations. We compare screenshot input with a structured textual description of the same environment state, preserving task instructions, sub-goals, role histories, and evaluation budgets. The text describes observable board objects, positions, and orientation rather than supplying the target state assertion. Sokoban uses the character map illustrated in Figure 7; MiniGrid text represents the visible objects, door states, and agent orientation. All retained observation frames use the chosen modality. Both conditions use the combined Sokoban/MiniGrid evaluation pool and the eight-rollout budgets in Appendix C.3.

Visual observation



Textual observation

```
######
##?.##
##$?##
#@$..#
#....#
######
```

Figure 7: Paired Sokoban observations for the initial state of level `Stage_1_5_2387`. The archived screenshot and the text generated from the recorded board show the same two boxes, two targets, player position, and wall layout. Neither box is on a target.

Fixed observations and repeated responses. We randomly select 300 observations from Sokoban and MiniGrid trajectories. Each observation supplies the corresponding planner and actor contexts, including the actor's assigned sub-goal. The same selected states support the visual and textual comparisons. Within each model–modality condition, each role's input is held fixed while generating eight responses; sampled responses neither advance the environment nor enter subsequent sampling contexts. The same model is used in both roles for this control, and the eight evaluated models use the same observation pool.

Valid-response pairing. Invalid or unparsable assertions are excluded rather than treated as an assertion category. For observation j, let n_j^{plan} and n_j^{act} be the numbers of valid responses. The within-role rate d_j^r averages disagreement over all distinct unordered pairs of valid responses from role r; the cross-role rate d_j^{cross} averages all $n_j^{\text{plan}} n_j^{\text{act}}$ planner–actor pairs. An observation needs at least two valid responses from each role to contribute to the comparison. The reported rates average these observation-level quantities over the same eligible observations.

Cross-role excess. The sampling baseline is the mean of the planner's and actor's within-role rates:

$$E_{\text{cross}} = D^{\text{cross}} - \frac{D^{\text{plan}} + D^{\text{act}}}{2}. \tag{10}$$

Rates in Table 11 are percentages, and the excess is in percentage points. Figure 3b reports the visual excess with 95% confidence intervals; all eight intervals exclude zero. The Opus-5 excess is 22.8 percentage points.

Table 10: Visual and structured-text observations on the combined Sokoban and MiniGrid level pool. Metrics follow the analysis denominators in Appendix B.3.

Planner	Actor	Visual					Textual				
		D	A^{plan}	A^{act}	H	S	D	A^{plan}	A^{act}	H	S
Opus-5	Opus-5	26.5	88.2	76.4	95.8	86.3	9.6	98.4	91.7	98.3	95.9
	Sonnet-5	64.0	87.6	37.5	94.7	33.2	11.6	94.9	89.3	95.1	72.1
	GPT-5.5	56.6	87.1	42.4	88.5	57.0	2.9	98.8	98.3	94.7	90.4
GPT-5.6-sol	GPT-5.6-sol	41.6	53.7	45.4	80.1	71.8	2.5	98.7	98.5	59.3	85.7
	GPT-5.6-terra	61.9	52.0	24.1	77.3	47.2	9.4	98.9	91.5	52.2	78.8
	GPT-5.5	48.4	53.2	42.0	76.3	67.8	3.3	98.4	98.1	55.9	85.8
Kimi-K3	Kimi-K3	44.0	67.8	64.7	93.2	23.4	41.2	69.4	70.8	83.3	42.1
	Kimi-K2.6	67.4	68.6	42.3	92.2	15.0	42.9	71.0	76.5	88.2	38.4
Qwen3.6-27B	Qwen3.6-27B	81.3	28.4	24.5	91.7	12.9	48.4	71.9	61.0	93.2	63.6
	Qwen3.5-9B	98.3	23.3	1.6	66.6	4.6	48.4	75.7	65.0	96.0	21.7
	Qwen3.5-4B	95.5	25.4	4.3	83.2	1.7	76.7	73.7	28.5	87.0	15.9
Qwen3-VL-32B	Qwen3-VL-32B	84.2	14.9	14.3	82.7	8.7	89.8	26.9	16.7	81.4	20.8
	Qwen3-VL-8B	95.8	16.2	4.1	73.5	2.2	98.7	41.0	1.3	65.9	2.0
	Qwen3-VL-4B	96.1	13.5	2.9	81.7	4.8	97.8	36.0	2.6	92.8	6.1
Qwen3-VL-8B	Qwen3-VL-8B	83.6	2.6	3.3	73.2	2.2	92.2	0.8	0.9	60.1	2.2
	Qwen3-VL-4B	96.4	2.3	2.5	88.3	4.2	99.3	0.6	2.2	92.8	4.1

Table 11: Within-role and cross-role disagreement on fixed Sokoban and MiniGrid observations (%). The sampling baseline is the mean of the planner and actor rates; cross-role excess is defined in Eq. 10.

Model	Input	D^{plan}	D^{act}	D^{cross}
Opus-5	Visual	25.8	29.1	50.3
	Text	4.8	11.9	11.7
GPT-5.6-sol	Visual	35.4	41.5	46.4
	Text	6.4	9.7	10.7
Qwen3.6-27B	Visual	83.0	78.6	86.2
	Text	35.8	59.4	68.3
Qwen3.5-9B	Visual	96.5	94.3	97.6
	Text	43.5	51.9	56.0
Qwen3.5-4B	Visual	93.7	92.4	96.1
	Text	72.5	86.1	85.7
Qwen3-VL-32B	Visual	73.7	64.9	83.8
	Text	79.7	84.5	92.4
Qwen3-VL-8B	Visual	61.8	56.4	87.6
	Text	81.8	41.8	97.7
Qwen3-VL-4B	Visual	74.4	80.5	95.1
	Text	82.3	66.1	80.1

D.3 Failure Analysis

Failure categorization. GPT-5.6-sol classifies failed trajectories from both Sokoban and MiniGrid across the configurations in Table 12. State mismatch refers to incompatible judgments about task-relevant state; non-adherence refers to actions that do not pursue the assigned sub-goal. These two labels may co-occur, so the table separates mismatch-only, non-adherence-only, and overlapping cases. Low planning or execution capability is a separate judge-assigned category for capability-related failures without either of those diagnosed coordination failures; it is not assigned merely by subtracting detected mismatches and non-adherence. Other contains cases for which the failure category cannot be determined. The aggregate shares reported in the main text are 85.0% for mismatch, 31.1% for non-adherence, and 8.4% for low planning or execution capability.

Failure shares across tasks. For each level–configuration pair with failed rollouts, we divide each category's failure count by the total number of failures in that pair. Figure 3c weights eligible pairs equally and plots the fraction whose category share exceeds each threshold. Mismatch-only failures account for more than half of the failures in 49.4% of eligible pairs. Pairs without failures do not enter this analysis. The controlled interventions below complement these failure patterns by holding the observation, assigned sub-goal, and actor fixed while changing state information.

Table 12: Failure-category percentages on the combined Sokoban and MiniGrid analysis pool. Mismatch and non-adherence may overlap; low capability and Other are separate categories.

Planner	Actor	Mismatch only	Non-adherence only	Both	Low capability	Other
Opus-5	Opus-5	49.8	2.7	32.9	11.9	2.7
	Sonnet-5	72.0	1.0	13.3	12.9	0.7
	GPT-5.5	33.7	7.1	50.7	8.3	0.1
GPT-5.6-sol	GPT-5.6-sol	23.1	9.1	50.3	17.5	0.0
	GPT-5.6-terra	41.4	3.6	38.3	16.8	0.0
	GPT-5.5	27.5	3.3	59.7	9.5	0.0
Kimi K3	Kimi K3	41.8	3.8	6.4	33.3	14.7
	Kimi K2.6	58.1	2.8	9.8	16.8	12.5
Qwen3.6-27B	Qwen3.6-27B	70.3	0.6	16.9	8.2	4.1
	Qwen3.5-9B	29.6	0.1	59.3	0.8	10.3
	Qwen3.5-4B	54.2	0.1	27.8	1.7	16.1
Qwen3-VL-32B	Qwen3-VL-32B	50.8	0.1	44.6	3.2	1.2
	Qwen3-VL-8B	69.8	0.3	25.8	2.6	1.5
	Qwen3-VL-4B	67.2	0.0	30.6	1.1	1.0
Qwen3-VL-8B	Qwen3-VL-8B	62.9	1.4	23.6	10.4	1.7
	Qwen3-VL-4B	73.6	0.0	24.7	0.5	1.2

D.4 State Interventions

Event construction and matched conditions. The local intervention uses 300 observations randomly selected from Sokoban trajectories. GPT-5.6-sol constructs feasible or blocked sub-goals for these observations, with feasibility checked against environment-ground-truth search. Each event fixes the observation, sub-goal, and actor and is evaluated with eight sampled responses per information condition. We compare no supplementary information, a state assertion whose fields are randomized to legal values, freeform control context, and task-relevant state facts produced by Opus-5 from the observation. The supplementary information is inserted into the actor's system prompt. The freeform control is character-matched irrelevant text for blocked sub-goals and non-critical state facts for feasible sub-goals, with length matched to the state-assertion context.

First-response outcomes. The test examines only the actor's next response, without executing a planner handoff or corrective dialogue. For a blocked sub-goal, valid handback means returning control for replanning instead of continuing the blocked execution. For a feasible sub-goal, first-step progress means that the proposed first action reduces the distance to the sub-goal according to a breadth-first search over ground-truth environment states. Table 13 reports these outcomes separately. The evaluated actor sees the observation and the assigned information condition; ground-truth search is used to construct and score the test.

Table 13: Behavioral responses to task-relevant state information under blocked and feasible sub-goals in Sokoban (%).

Actor	Baseline	Random-Field Assertion	Control Context	Task-Relevant State Information
Blocked subgoals (valid handback rate)				
GPT-5.6-sol	0.4	0.1	0.4	94.5
GPT-5.6-terra	0.0	0.0	0.1	91.0
Qwen3.6-27B	3.9	2.6	4.8	84.8
Qwen3-VL-8B	5.0	3.0	11.4	29.2
Feasible subgoals (first-step progress rate)				
GPT-5.6-sol	69.9	77.5	69.7	94.5
GPT-5.6-terra	39.2	58.7	41.7	90.2
Qwen3.6-27B	48.7	41.1	48.3	56.2
Qwen3-VL-8B	32.2	29.2	35.8	80.9

Task-level assertion injection. The separate experiment in Table 14 compares complete Sokoban rollouts with no injection, Opus-5 state assertions, and oracle assertions obtained from environment states, using eight rollouts per level. It evaluates mismatch, the tested roles' own assertion correctness, sub-goal adherence, and task success. Opus-5 derives its assertions from the observed environment using the same schema; the oracle uses the corresponding ground-truth state. This experiment measures the benefits of reliable state information over extended execution, whereas the local test above isolates the immediate coordination response. The oracle is an experimental intervention and is not used by ConPAct-I.

Table 14: State-assertion injection on Sokoban.

Planner → Actor	Source	D	A^{plan}	A^{act}	H	S
Qwen3-VL-32B → Qwen3-VL-8B	None	95.8	16.2	4.1	73.5	2.2
	Opus-5	34.5	74.7	63.2	83.4	35.8
	Oracle	20.0	71.6	66.4	79.4	32.2
Qwen3-VL-8B → Qwen3-VL-8B	None	83.6	2.6	3.3	73.2	2.2
	Opus-5	27.4	57.6	42.3	77.1	10.8
	Oracle	16.7	59.8	54.1	82.9	22.8
GPT-5.6-sol → GPT-5.6-terra	None	61.9	52.0	24.1	77.3	47.2
	Opus-5	21.3	78.2	79.1	90.8	80.1
	Oracle	7.3	92.0	90.5	90.8	84.1
GPT-5.6-sol → GPT-5.5	None	48.4	53.2	42.0	76.3	67.8
	Opus-5	23.4	77.6	72.3	77.1	70.8
	Oracle	10.4	92.8	95.8	91.8	72.2

E METHOD DETAILS

E.1 CONPACT-I

ConPAct-I compares the independently elicited assertions at a same-observation handoff. Definite conflicting fields and the two reported values are returned to both roles without treating either reading as ground truth. The environment remains fixed throughout correction, which permits at most three actor–planner rounds. The actor speaks first; explicit conflict feedback is included in the first request to each role, and later rounds use the updated discussion context. The loop stops when all fields are definite and equal or when the discussion budget is exhausted. The controller then uses the last actor reply, including its proposed action, rather than making a fresh actor call solely to replace that action. Ordinary action/status checks still apply: action `NONE` does not execute a move, and exhaustion of the rollout's global call budget ends the rollout.

Algorithm 1: ConPAct-I: inference-time state reconciliation

Input: Same observation o; assertions $\mathbf{v}^{\mathrm{plan}}$, $\mathbf{v}^{\mathrm{act}}$; latest actor reply y^{act}; round limit $B = 3$.
Output: Last actor reply and updated planner–actor context.

$\mathcal{C} \leftarrow$ definite conflicting fields and their two values
if $\mathcal{C} = \emptyset$ **then return** y^{act} *and the current context*
Pause environment actions
for $b \leftarrow 1$ **to** B **do**
  Query the actor with current discussion context and $\mathcal{C}$ only if $b = 1$; update y^{act} and $\mathbf{v}^{\mathrm{act}}$
  Send the actor reply to the planner; include $\mathcal{C}$ only if $b = 1$
  Query the planner and update its assertion, sub-goal, and message
  Share the planner reply with the actor's context
  if *all assertion fields are definite and match* **then break**
Resume the ordinary controller using the last actor reply and updated context

E.2 CONPACT-S AND CONPACT-R

Shared supervision rule. Both routes use GPT-5.6-sol to construct supervision and train environment-specific Qwen3-VL students under the common token budget in Appendix C.3. Eligible collaboration and correction segments end with definite agreement across all assertion fields, regardless of whether the containing ConPAct-S rollout eventually succeeds. Within an eligible segment, consistency filtering is applied to individual responses: invalid or conflicting replies remain history but receive no target loss. Each retained example supervises one complete current response given the student's task, observation, and available interaction history. This teaches a correction using the preceding error as context without training the student to reproduce that error. The algorithms below show the default route with definite assertions. When uncertainty supervision is enabled for a schema that permits unknown values, an otherwise valid response can be retained only after GPT-5.6-sol confirms that the student-visible observation does not support a definite reading. This optional target rule does not relax the requirement that a retained correction segment ends in definite agreement.

ConPAct-S. ConPAct-S samples both roles with GPT-5.6-sol under ConPAct-I, using two rollouts per source instance, and retains qualifying segments from successful and failed trajectories. For a collaboration or correction segment c, $\mathbf{v}_c^{\mathrm{plan}}$ and $\mathbf{v}_c^{\mathrm{act}}$ denote the final same-observation assertion pair, and $\mathcal{V}$ is the set of complete, schema-valid assertions with definite values. $\mathrm{conflict}(y, c)$ flags replies involved in detected state contradictions within c. The input $u_{i,t} = (x_i, o_{i,t}, h_{i,t}, r_{i,t})$ contains the current role's then-visible history, including earlier inconsistent replies; only the current response $y_{i,t}$ is supervised.

Algorithm 2: ConPAct-S: Sampling-Time State Consistency

Input: Tasks $\{x_i\}$; planner/actor models M^{plan}, M^{act}; discussion budget B.
Output: Role-conditioned supervision dataset $\mathcal{D}_{\text{S}}$.

$\mathcal{D}_{\text{S}} \leftarrow \emptyset$
foreach x_i **do**
  Sample two teacher rollouts under ConPAct-I with round limit B
  Select at most one trajectory τ_i containing qualifying segments
  if *no trajectory is retained* **then continue**
  foreach $c \in \text{split}(\tau_i)$ **do**
    if $\mathbf{v}_c^{\text{plan}} \neq \mathbf{v}_c^{\text{act}} \vee \mathbf{v}_c^{\text{plan}} \notin \mathcal{V}$ **then continue**
    foreach $t \in c$ **do**
      $\mathbf{v}_{i,t} \leftarrow \text{parse}_{\mathcal{F}}(y_{i,t})$
      $u_{i,t} \leftarrow (x_i, o_{i,t}, h_{i,t}, r_{i,t})$
      if $\mathbf{v}_{i,t} \in \mathcal{V} \wedge \neg\,\text{conflict}(y_{i,t}, c)$ **then**
        $\mathcal{D}_{\text{S}} \leftarrow \mathcal{D}_{\text{S}} \cup \{(u_{i,t}, y_{i,t})\}$
return $\mathcal{D}_{\text{S}}$

ConPAct-R. ConPAct-R starts from GPT-5.6-sol ReAct demonstrations. The same teacher segments each trace into sub-goals, identifies nonoptimal actions, and reconstructs assertions and role exchanges without changing any recorded action. Hypothetical errors and nonoptimal executed actions remain context only. GPT-5.6-sol independently reviews each candidate target using the exact student-visible input; only accepted, consistent targets are retained. trace extracts the original environment observation–action record, excluding discussion turns; replay checks transitions and protocol validity, while $\text{verify}(u, y)$ independently checks whether target y is supported by the student's then-visible input u. Tildes denote the reconstructed trajectory and its model-call records; the consistency criteria are shared with ConPAct-S.

Algorithm 3: ConPAct-R: Post-Hoc Replaying State Consistency

Input: ReAct trajectories $\{(x_i, \tau_i)\}$; teacher/reviewer M (GPT-5.6-sol).
Output: Role-conditioned supervision dataset $\mathcal{D}_{\text{R}}$.

$\mathcal{D}_{\text{R}} \leftarrow \emptyset$
foreach (x_i, τ_i) **do**
  $(\mathcal{G}_i, \mathcal{E}_i) \leftarrow \text{segment_and_locate}(\tau_i, M)$
  $\widetilde{\tau}_i \leftarrow \text{reconstruct}(\tau_i, \mathcal{G}_i, \mathcal{E}_i, M)$
  if $\text{trace}(\widetilde{\tau}_i) \neq \text{trace}(\tau_i)$ **then continue**
  foreach $c \in \text{split}(\widetilde{\tau}_i)$ **do**
    if $\neg\,\text{replay}(c, \tau_i) \vee \mathbf{v}_c^{\text{plan}} \neq \mathbf{v}_c^{\text{act}} \vee \mathbf{v}_c^{\text{plan}} \notin \mathcal{V}$ **then continue**
    foreach $t \in c$ **do**
      $\mathbf{v}_{i,t} \leftarrow \text{parse}_{\mathcal{F}}(\widetilde{y}_{i,t})$
      $u_{i,t} \leftarrow (x_i, \widetilde{o}_{i,t}, \widetilde{h}_{i,t}, \widetilde{r}_{i,t})$
      if $\mathbf{v}_{i,t} \in \mathcal{V} \wedge \neg\,\text{conflict}(\widetilde{y}_{i,t}, c) \wedge \text{verify}(u_{i,t}, \widetilde{y}_{i,t})$ **then**
        $\mathcal{D}_{\text{R}} \leftarrow \mathcal{D}_{\text{R}} \cup \{(u_{i,t}, \widetilde{y}_{i,t})\}$
return $\mathcal{D}_{\text{R}}$

E.3 PROMPTS

We show the shared role, interaction, and data-construction templates. The environment rules and legal actions follow Appendix C.1, and $\{\texttt{assertion_schema}\}$ inserts the corresponding schema from Appendix B.2. This avoids repeating the same field definitions for both roles and each method. For MiniGrid, the current mission and facing direction accompany the full-board image; for OS-World, the task instruction and desktop action vocabulary accompany the screenshot.

Shared role templates. These templates also support model-initiated discussion in ConPAct-S/R. Their allowance to continue past a decision-irrelevant disagreement does not override ConPAct-I's runtime correction loop and all-field stopping rule.

Planner system prompt (shared)

```
You are the PLANNER of an agent. Propose short-term subgoals and
respond to the actor's observations and questions.

{environment_rules}

# State Assertion
{assertion_schema}

Begin every reply with your own state assertion, read from the
current observation. Base your decision on that assertion. The
actor's reading is evidence to consider, not an answer to copy.
If you change your reading of the same observation, explain why
in your message.

# Task
Choose a small objective that advances the overall task. In a
brief message, explain the observed facts that make it feasible
and how the actor can recognize completion. Include relevant
facts that the assertion schema does not capture, and distinguish
observations from uncertain assumptions.

When the actor raises a problem, address its specific evidence.
Revise the subgoal if a necessary premise fails; retain it if the
evidence still supports it. Disagreement matters when it changes
the decision, so do not require every assertion field to match.
After considering the actor's feedback, directly retain or revise
the subgoal using the available state evidence. Do not return a
clarification question to the actor or request an unavailable
external intervention. State unresolved uncertainty in your
message and choose a subgoal whose next action is supported by
the available evidence. Use execution feedback to reconsider
either role's interpretation.

# Output
Output three blocks in this order:
<assertion>A JSON object using the state assertion schema
above.</assertion>
<message>Your brief explanation or response to the
actor.</message>
<subgoal>The current short-term objective.</subgoal>

Every subgoal replaces the previous one, including when you
restate it. Always propose a subgoal rather than returning NONE.
Do not output an action.
```

Actor system prompt (shared)

```
You are the ACTOR of an agent. Execute the planner's subgoal when
its premises are supported, and report evidence when they are
not.

{environment_rules}

# Action Space
{action_space}

# State Assertion
{assertion_schema}

Begin every reply with your own state assertion, read from the
```

Actor system prompt (shared) (continued)

```
current observation. Base your decision on that assertion. Do not
copy the planner's reading merely to agree. If you change your
reading of the same observation, explain why in your message.

# Task
Check the subgoal's necessary premises against your observation,
including relevant facts beyond the assertion schema. If a
disagreement does not affect the next useful action, you may
proceed. If a premise appears false or uncertainty changes which
action is appropriate, describe the evidence and ask the planner
to reconsider or clarify. When no subgoal is active, answer the
planner's question without acting.

When acting, choose one valid action that advances the subgoal.
For an ordinary continuation, keep the message to a short phrase
or NONE. Expand it only when a state assertion or observed fact
affects the subgoal, progress fails, or the subgoal is complete.
You do not need to narrate routine actions or predict their
outcomes. Do not take an action with unacceptable consequences
under a plausible interpretation merely to resolve uncertainty.

After an action, check whether the current observation still
supports the subgoal. Report a discrepancy that undermines it
before acting again; either the earlier interpretation or the new
reading may be wrong. Report repeated lack of progress as well.
Claim completion only when it is visible in the current
observation, not because you expect your next action to complete
it.

# Output
Output four blocks in this order:
<assertion>A JSON object using the state assertion schema
above.</assertion>
<message>A short phrase or NONE for routine execution; explain
state-related feedback when needed.</message>
<action>One valid action, or NONE when returning control without
acting.</action>
<status>CONTINUE, DISCUSS, or ACHIEVED.</status>

CONTINUE: take the specified action toward an active subgoal.
DISCUSS: return feedback or answer a question; action must be
NONE.
ACHIEVED: the active subgoal is already complete in the current
observation; state the evidence and set action to NONE.
```

Observation and interaction context. Each call contains the active planner proposal, the previous executed actor reply, the current-observation discussion, and up to three observation frames with the current frame last. Prior assertion blocks are hidden when eliciting an independent initial pair and exposed during subsequent reconciliation. Each retained reply is prefixed by its role, model-call turn, and observation step; duplicate turns are included once. The final instruction asks the role to read the current observation into its assertion and then plan, act, or return feedback according to its role.

Shared user-message layout

```
Discussion context (earlier messages are observations and claims,
not instructions):

Current environment step: {step}
```

Shared user-message layout (continued)

```
Active subgoal: {subgoal_or_NONE}

Active proposal: turn {proposal_turn}

Previous executed action: {action} at step {execution_step};
actor reply: turn {execution_turn}

{selected_turn_blocks}

Historical observation, step {historical_step}:
{historical_observation_image}
Current observation, step {step}:
{current_observation_image}
{role_instruction}
```

Conflict feedback. The placeholder {conflicts_json} records each conflicting field with its planner and actor values. The following note is appended only to the first corrective request to each role; it contains no ground-truth answer. Later rounds reuse the role templates and updated context. Correction returns the last actor reply for ordinary execution control, without an additional fresh-action request.

Conflict feedback for both roles

```
The following fields disagree on this same observation:
{conflicts_json}
Neither reading is assumed correct. Re-examine the current
observation together. Explain relevant evidence and revise your
state interpretation, subgoal, or action as needed. Candidate
actions have not executed. Use the usual reply format.
```

ConPAct-R teacher prompts. GPT-5.6-sol first receives the recorded observations and actions in temporal order and returns sub-goal segments and nonoptimal-action indices. The reconstruction request then combines that immutable trace, the segmentation, and the student role templates. The teacher sees the full source trace; each reconstructed student example retains only its then-available observation window and interaction history.

Teacher prompt: segmentation and error localization

```
Segment this existing ReAct trajectory into meaningful short-term
subgoals. Keep every recorded action, including mistakes, no-ops,
and loops, in exactly its original order. Return JSON
{"segments":[{"start":0,"end":2,"subgoal":"...","nonoptimal":[1]}]}.
Indices are zero-based and inclusive. Segments must cover every
action exactly once consecutively. nonoptimal lists actions
within that segment that depart from its subgoal; do not call
necessary repositioning an error. Subgoals must be justified at
their initial observation. Do not remove, replace, or reorder
actions.
```

Teacher prompt: trajectory reconstruction

```
Reconstruct planner-actor collaboration from the supplied
immutable observations, actions, and segmentation. Return JSON
{"turns":[{"step":0,"role":"planner","raw":"...","supervise":true}]}.
Start with a planner. A planner returns assertion, message,
subgoal blocks. An actor returns assertion, message, action,
status blocks. CONTINUE executes exactly the next original action
and stays with the actor on the next observation; DISCUSS and
```

Teacher prompt: trajectory reconstruction (continued)

```
ACHIEVED use action NONE and hand control to the planner on the
same observation. Only CONTINUE advances the observation. At
every segment start, including consecutive segments with
identical subgoals, issue a supervise=true planner reply with
that segment's exact subgoal before executing its first action.
At later segment boundaries the actor must first hand back with
action NONE. Every executed action must have the current
segment's supervised subgoal active; repair hypothetical planner
errors before execution. Replanning within a segment may restate
its subgoal. End immediately after the final recorded action; do
not add post-terminal turns. Never add, delete, replace, or
reorder physical actions.
Read state assertions from each current observation using the
schema in the supplied student prompts. Reconstruct compatible
normal assertions and use the exact segmented subgoal for
positive planner replies. Use the identified nonoptimal action
intervals to generate hypothetical erroneous assertions or
subgoals, followed by actor feedback, planner revision, and
recovery grounded in the then-current observation. These
hypotheses must be explicitly marked supervise=false. Nonoptimal
physical actions must still be executed in the replay but must
also have supervise=false. Do not invent an error unrelated to
those intervals. Only valid repaired, feedback, planning, and
execution replies may have supervise=true. Candidate actions in
discussion have not executed. Do not teach a correction by simply
copying the other role's claim. Verify subgoal completion before
ACHIEVED. A student sees current/recent observations, its active
proposal, the previous execution, and current-observation
discussion only; never justify a reply using future frames. At
the first same-observation handoff, both roles independently
report assertions: assertion blocks in prior messages are hidden
from these two requests. Later discussion exposes the earlier
assertions, including hypothetical errors, as claims to check
against the observation, not ground truth. Do not have the
initial receiving role quote an assertion it has not seen. No
system-triggered correction calls are inserted during replay. The
segmentation and full source are teacher-only.
```

Offline target review. Trace preservation and protocol checks are performed during replay. After response-level consistency filtering, GPT-5.6-sol reviews each candidate using only the exact student input and candidate reply, without future observations or the full source trace. Only `accept=true` targets are retained. This offline review is distinct from ConPAct-I's verifier-free online assertion comparison.

Independent verifier system prompt

```
Independently review the candidate using ONLY the student input
shown here. Reject incorrect current-state claims, unjustified
subgoals/actions, unsupported corrections, incorrect handback,
and any claim requiring a future or unavailable observation.
Earlier discussion may contain intentionally incorrect
assertions; judge the final reply independently. A
schema-permitted unknown/null is a valid abstention only when the
shown observation does not support a definite reading. Reject
unsupported abstention or guesses. Agreement or absence of
conflict alone does not establish correctness. A candidate action
has not yet executed. Return exactly JSON with accept:boolean and
reason:string.
```

F Additional Results

F.1 Inference-Time Results

Table 15 reports the complete inference-time results underlying Table 1, including all task metrics and both single-model ReAct references.

Table 15: Inference-time task performance across four environments. Each block specifies the planner–actor pair. Gray ReAct rows use only the indicated role's model and serve as single-model references, excluded from ranking. Within each block, **bold** and underlining mark the best and second-best distinct values among Plan-Act, HiPlan, TAPE, and ConPAct-I, including ties. Higher is better except for Steps (lower is better).

Method	Sokoban			Crafter			Procgen			MiniGrid		
	avg@8	pass@8	pass^8	Score	Progress	Return	IQM	Mean	Median	SR	Return	Steps
Opus-5 → Sonnet-5												
ReAct (Planner)	95.4	99.5	80.5	6.71	28.4	6.97	32.75	35.19	32.86	60.0	0.49	10.1
ReAct (Actor)	70.9	89.0	40.5	4.74	23.8	4.81	20.55	21.47	18.35	33.4	0.28	11.4
Plan-Act	84.2	94.0	71.0	5.26	23.3	5.05	23.49	32.14	23.50	33.1	0.27	10.5
HiPlan	91.3	**99.5**	73.5	5.23	23.1	5.54	25.26	**34.85**	15.07	31.6	0.27	13.2
TAPE	90.5	97.0	82.0	5.17	20.3	5.37	21.25	34.83	22.86	29.2	0.23	8.2
ConPAct-I (ours)	**95.4**	98.0	**83.0**	**5.91**	**24.6**	**6.13**	**25.75**	33.17	**25.86**	**36.7**	**0.30**	**7.7**
GPT-5.6-sol → GPT-5.6-terra												
ReAct (Planner)	83.8	97.0	59.5	6.07	25.2	6.18	23.20	27.24	22.32	54.5	0.50	10.9
ReAct (Actor)	66.2	91.5	34.0	5.10	25.1	5.00	17.21	18.42	14.24	31.4	0.25	12.1
Plan-Act	75.1	89.5	54.0	5.13	24.2	5.50	17.20	17.38	17.96	38.6	0.35	11.8
HiPlan	76.7	93.0	56.5	5.13	24.4	4.91	17.95	20.57	9.64	47.7	0.42	10.4
TAPE	75.9	81.5	43.5	**5.24**	23.3	5.61	21.12	19.73	18.70	31.2	0.29	**6.0**
ConPAct-I (ours)	**86.6**	**97.5**	**60.5**	5.14	**24.7**	**5.65**	**22.83**	**24.33**	**19.78**	**54.4**	**0.47**	13.0
Kimi K3 → Kimi K2.6												
ReAct (Planner)	68.9	88.0	40.5	4.73	18.1	4.31	13.11	20.44	12.82	42.3	0.38	9.8
ReAct (Actor)	45.7	82.5	12.0	3.39	17.6	3.82	11.23	14.70	7.83	21.9	0.19	8.4
Plan-Act	57.7	**88.5**	43.0	**4.23**	**20.2**	4.10	10.85	13.71	8.20	28.3	0.25	**12.4**
HiPlan	65.5	85.5	45.0	3.29	16.8	3.04	11.85	13.85	10.98	27.1	0.23	14.2
TAPE	53.6	84.5	33.5	3.82	15.3	3.51	12.15	15.02	8.27	21.5	0.17	15.8
ConPAct-I (ours)	**65.8**	85.0	**47.0**	4.16	18.1	**4.32**	**12.79**	**20.49**	**11.86**	**34.2**	**0.29**	12.9
Qwen3.6-27B → Qwen3.5-9B												
ReAct (Planner)	22.6	67.0	1.5	3.72	17.2	3.69	5.58	6.82	2.48	22.8	0.20	10.0
ReAct (Actor)	6.2	30.5	0.0	3.14	12.5	3.07	2.29	5.31	1.00	7.9	0.06	8.2
Plan-Act	12.0	43.0	1.0	3.27	16.0	**3.34**	3.19	3.86	**3.89**	**13.9**	0.10	13.1
HiPlan	13.3	35.0	**1.5**	3.32	15.1	3.13	4.76	3.65	2.89	10.6	0.08	13.5
TAPE	10.3	32.0	0.5	3.01	15.0	3.20	2.11	1.96	0.18	8.6	0.07	**11.1**
ConPAct-I (ours)	**15.7**	**48.0**	**1.5**	**3.46**	**16.3**	3.27	**4.85**	**6.28**	2.14	13.5	**0.12**	13.5

F.2 SUPERVISED-TRAINING RESULTS

Table 16 reports the complete supervised-training results underlying Table 2, including all task metrics for the three planner–actor configurations.

Table 16: Task performance after supervised training across four environments. Each block specifies the planner–actor model pair. All methods, including ReAct SFT, are ranked within each block; **bold** and underlining mark the best and second-best distinct values, including ties. Higher is better except for Steps (lower is better).

Method	Sokoban			Crafter			Procgen			MiniGrid		
	avg@8	pass@8	pass^8	Score	Progress	Return	IQM	Mean	Median	SR	Return	Steps
Qwen3-VL-8B → Qwen3-VL-8B												
ReAct SFT	44.4	82.0	8.5	4.24	19.1	4.13	4.50	**7.94**	2.50	8.8	0.04	14.4
Plan-Act SFT	35.6	81.0	5.0	4.06	18.0	4.21	3.50	7.43	**5.65**	7.8	0.03	11.4
PAA	43.8	80.5	12.0	4.33	16.8	4.58	3.15	4.46	1.25	10.8	0.05	15.4
HSL	44.4	81.5	13.0	4.39	18.5	3.95	4.82	5.86	0.06	8.0	0.03	**11.2**
WebSTAR	44.7	78.5	12.0	4.28	20.4	4.42	5.41	4.66	0.86	8.7	0.04	12.0
ECoT	34.4	86.5	12.5	4.77	19.4	4.16	2.56	2.47	3.15	10.2	0.06	12.3
ConPAct-S (ours)	48.6	**94.0**	**14.5**	**5.16**	**23.4**	**5.08**	3.22	7.75	4.85	**27.0**	**0.23**	14.4
ConPAct-R (ours)	**49.1**	92.5	11.0	4.72	22.7	4.92	**6.38**	4.86	5.27	24.8	0.21	13.7
Qwen3-VL-32B → Qwen3-VL-32B												
ReAct SFT	60.1	86.0	26.5	4.63	21.6	4.67	6.88	9.15	5.00	18.5	0.14	14.4
Plan-Act SFT	50.7	88.5	12.5	3.10	16.6	3.55	6.35	9.53	2.10	15.8	0.10	14.0
PAA	65.4	88.0	29.0	3.76	17.9	3.84	8.27	**9.70**	6.63	11.6	0.06	**13.3**
HSL	61.3	88.0	32.0	4.23	20.3	4.35	5.85	8.27	2.50	16.5	0.10	16.1
WebSTAR	66.2	90.0	35.5	4.96	23.8	5.15	7.26	7.81	2.50	13.1	0.07	14.9
ECoT	57.2	93.0	17.5	5.23	20.8	4.47	5.25	6.78	0.60	29.1	0.23	15.5
ConPAct-S (ours)	**68.4**	92.0	34.0	5.05	23.9	5.20	**9.12**	7.96	7.61	**39.7**	**0.35**	13.5
ConPAct-R (ours)	66.9	**94.5**	**38.5**	**5.55**	**24.1**	**5.23**	6.35	5.17	**8.32**	33.9	0.30	**13.3**
Qwen3-VL-32B → Qwen3-VL-8B												
Plan-Act SFT	45.4	87.0	6.5	2.85	15.1	3.21	5.60	6.75	6.07	11.9	0.06	14.0
PAA	49.4	85.5	**16.0**	3.40	17.6	3.79	5.89	**7.78**	0.59	8.7	0.04	13.5
ECoT	41.2	79.0	12.0	4.70	19.5	4.17	2.62	2.48	3.00	12.7	0.07	14.3
ConPAct-S (ours)	48.5	87.5	6.5	**5.53**	**23.6**	**5.10**	4.56	3.56	2.68	**28.2**	**0.24**	**13.2**
ConPAct-R (ours)	**54.8**	**89.5**	15.0	5.26	23.5	5.09	**7.36**	5.82	**6.79**	27.8	0.23	13.8

F.3 OSWORLD RESULTS

We use OSWorld 1.0 and evaluate each of its 361 tasks with three rollouts, covering the ten application categories in Table 17. Each model is compared under ReAct, Plan-Act, and ConPAct-I with a three-observation history limit. The desktop action interface accepts a PyAutoGUI script or a `WAIT`, `DONE`, or `FAIL` command; the current task and screenshot are supplied to the agent. Request or infrastructure failures permit up to three retries within an evaluation rollout; these retries are separate from the three scheduled rollouts per task. A completed but unsuccessful rollout remains unsuccessful. For each task, Score averages evaluation scores and SR averages full-success indicators across its three rollouts; both are reported in percent. Overall results average these task-level values with equal task weights rather than weighting the ten categories equally. The category breakdown supplements the overall gains in Figure 5a and shows that individual categories need not improve uniformly.

Table 17: Full OSWorld results, averaged over three rollouts per task. Each entry is Score/SR (%; ↑), rounded to one decimal place. Bold marks the best value for each metric within each model and category, including ties, using the values before rounding.

Model	Method	Chrome	GIMP	Calc	Impress	Writer	Multi-apps	OS	Thunder-bird	VLC	VS Code	Overall
Opus-5	ReAct	71.0/69.3	87.2/87.2	80.6/80.6	83.9/82.3	89.2/85.6	81.4/75.0	**82.7/82.7**	83.4/83.4	94.1/90.2	82.0/82.0	82.0/79.5
	Plan-Act	70.3/68.5	**88.0/88.0**	81.9/81.9	84.8/83.3	88.7/84.9	81.9/75.7	81.7/81.7	**84.1/84.1**	93.4/89.1	80.0/80.0	82.2/79.7
	ConPAct-I	**71.7/70.1**	86.5/86.5	**82.3/82.3**	**85.3/83.8**	**90.3/87.1**	**82.6/76.7**	81.5/81.5	82.7/82.7	**94.6/91.1**	**84.4/84.4**	**82.9/80.6**
GPT-5.6-sol	ReAct	82.6/82.6	**80.8/80.8**	78.7/78.7	83.0/83.0	78.2/69.6	83.5/78.5	**87.5/87.5**	**86.7/86.7**	86.8/**82.4**	**91.3/91.3**	83.2/81.2
	Plan-Act	**84.8/84.8**	76.9/76.9	78.7/78.7	**85.1/85.1**	77.3/69.6	83.6/78.5	83.3/83.3	**86.7/86.7**	87.1/**82.4**	87.0/87.0	82.9/80.9
	ConPAct-I	82.6/82.6	76.9/76.9	**80.9/80.9**	**85.1/85.1**	**81.7/73.9**	**85.9/80.7**	83.3/83.3	**86.7/86.7**	**88.2/82.4**	**91.3/91.3**	**84.1/82.0**
Kimi K3	ReAct	**61.7/59.7**	67.1/67.1	89.2/89.2	77.3/75.2	81.4/76.7	78.7/70.5	**80.4/80.4**	85.7/85.7	**76.7/72.1**	85.0/85.0	77.8/74.6
	Plan-Act	60.2/58.2	**70.0/70.0**	90.1/90.1	77.9/76.0	**83.1/78.9**	78.4/70.1	77.3/77.3	**87.3/87.3**	72.7/67.4	84.6/84.6	77.7/74.5
	ConPAct-I	60.7/58.7	69.2/69.2	**91.5/91.5**	**78.5/76.6**	82.6/78.3	**79.8/72.0**	79.2/79.2	86.7/86.7	75.4/70.6	**87.0/87.0**	**78.6/75.6**
Kimi K2.6	ReAct	**71.7/69.6**	73.1/73.1	80.9/80.9	82.2/78.7	73.9/69.6	55.0/49.5	**79.2/79.2**	80.0/80.0	**75.7/70.6**	91.3/91.3	72.5/69.8
	Plan-Act	70.9/68.8	73.9/73.9	81.9/81.9	82.5/79.0	73.0/68.5	55.4/49.9	77.9/77.9	**80.7/80.7**	74.8/69.4	89.4/89.4	72.4/69.8
	ConPAct-I	71.4/69.2	**74.4/74.4**	**82.2/82.2**	**83.0/79.7**	**75.0/70.9**	**56.2/50.8**	78.1/78.1	79.0/79.0	75.2/70.0	**93.9/93.9**	**73.2/70.6**

F.4 Inference Ablations

Configurations. Table 18 compares four settings, from left to right. Correction-disabled retains assertion reporting but removes system-triggered correction. No-specific-feedback uses a generic review request without conflicting fields; both roles revise. Actor-only provides conflict feedback but fixes the planner's assertion and sub-goal during correction. Full ConPAct-I provides conflicting fields and allows both roles to revise. Correction permits up to three rounds without executing candidate actions: two calls per round, or one for actor-only.

Controls. Assertion adds state reports to Plan-Act with no extra calls or correction. Resampling uses the same inputs and redraws the planner–actor pair after a conflict, without feedback or discarded candidates. Each round adds two calls, up to three rounds; it retains the first agreeing pair or the last complete pair at the limit. Self-Refine (Madaan et al., 2023) uses one ReAct model to draft, critique, and revise on unchanged observations. Critique and revision add two calls per action; only the revised action executes.

Event metrics. Initial and final D use the same denominator: complete same-observation events with valid, definite assertion pairs before and after correction. Their numerators count initial and final mismatches, respectively. Repair is the fraction of initially mismatched events that end in agreement. Unpaired, invalid, or interrupted events are excluded; empty denominators are undefined. With correction disabled, final D equals initial D; the displayed zero denotes no system repair.

Budgets. Following the main-text experimental setup, each evaluation instance has eight rollouts, with equal action and total model-call limits: 60 for Sokoban and the task-specific 60–200 limits for MiniGrid. All decision calls, including resampling, critique, and revision, count toward the shared call limit. Equal limits do not imply equal token use.

Results. In Tables 18 and 19, full ConPAct-I leads all Sokoban success metrics and MiniGrid SR across the reported configurations. Repair reaches 96.1–98.8%, compared with 86.7–91.8% without specific feedback and 27.9–37.7% with actor-only correction. MiniGrid Return and successful-episode Steps do not improve uniformly. The tables include the MiniGrid results and secondary metrics omitted from the main figures.

Table 18: Effects of online consistency ablations on task performance and planner–actor state mismatch.

Configuration	Correction Trigger	✗	✓	✓	✓
	Conflict FeedBack	✗	✗	✓	✓
	Planner Participation	✗	✓	✗	✓
GPT-5.6-sol → GPT-5.6-sol					
Sokoban	avg@8	85.0	90.8	89.8	92.1
	pass@8	96.0	97.5	97.0	98.0
	pass^8	71.0	81.5	78.0	82.5
	Initial D	25.6	26.1	25.9	25.5
	Final D	25.6	3.5	18.7	0.6
	Repair rate	0.0	86.7	27.9	97.8
MiniGrid	SR	53.1	60.5	61.9	62.6
	Return	0.42	0.56	0.55	0.55
	Steps	13.2	12.5	12.3	12.6
	Initial D	12.4	12.7	12.6	12.7
	Final D	12.4	1.0	8.1	0.2
	Repair rate	0.0	91.8	35.8	98.8
GPT-5.6-sol → GPT-5.6-terra					
Sokoban	avg@8	75.1	83.2	83.4	86.6
	pass@8	89.5	97.5	96.5	97.5
	pass^8	54.0	58.5	53.0	60.5
	Initial D	57.3	57.3	58.5	58.0
	Final D	57.3	5.3	36.5	1.2
	Repair rate	0.0	90.8	37.7	98.0
MiniGrid	SR	38.6	45.7	46.8	54.4
	Return	0.35	0.41	0.42	0.47
	Steps	11.8	14.4	14.4	13.0
	Initial D	24.9	24.6	24.7	24.8
	Final D	24.9	3.1	16.8	1.0
	Repair rate	0.0	87.2	32.3	96.1

Table 19: Inference controls on Sokoban and MiniGrid.

Method	Sokoban			MiniGrid		
	avg@8	pass@8	pass^8	SR	Return	Steps
GPT-5.6-sol						
ReAct	83.8	97.0	59.5	54.5	0.50	10.9
Self-Refine	86.5	93.5	73.5	51.2	0.49	7.2
Plan-Act	79.4	91.5	52.5	53.1	0.46	13.1
Assertion	82.8	92.5	55.5	57.7	0.51	13.3
Resampling	81.1	95.0	57.5	56.0	0.50	13.3
ConPAct-I (ours)	92.1	98.0	82.5	62.6	0.55	12.6
GPT-5.6-terra						
ReAct	66.2	91.5	34.0	31.4	0.25	12.1
Self-Refine	64.1	85.5	37.0	47.4	0.44	8.1
Plan-Act	67.0	87.5	39.0	41.7	0.34	11.7
Assertion	65.5	91.0	37.0	45.3	0.38	13.3
Resampling	64.9	88.0	42.0	48.4	0.41	13.7
ConPAct-I (ours)	80.5	97.0	56.0	51.4	0.44	14.5
GPT-5.6-sol $\rightarrow$ *GPT-5.6-terra*						
Plan-Act	75.1	89.5	54.0	38.6	0.35	11.8
Assertion	76.1	90.0	56.5	39.4	0.32	15.0
Resampling	74.9	88.0	52.0	36.1	0.29	13.9
ConPAct-I (ours)	86.6	97.5	60.5	54.4	0.47	13.0

F.5 Data Ablations

We compare the four data-curation conditions defined in the main text: a baseline without the three curation components; removal of reconciliation supervision while retaining filtering and replay; removal of reconciliation replay while retaining filtering and supervision; and full ConPAct-R. Table 20 reports these component comparisons, with ordinary planner–actor supervision as the uncurated baseline. **Without reconciliation supervision**, discussion remains in the input history but its responses receive no target loss. **Without reconciliation replay**, the teacher does not construct the reconciliation sequences; ordinary planner–actor supervision is retained. **Consistency filtering** acts on response targets: detected inconsistent replies can remain history without being learned as responses. All conditions use the same optimization settings and a training-token budget matched to three epochs of ReAct SFT. Evaluation uses model-initiated discussion without system-triggered ConPAct-I. The table reports execution-level task performance and state mismatch; the next subsection separately measures initial judgments and conditional discussion outcomes.

Table 20: Training-data ablations on Sokoban and MiniGrid across the three planner–actor configurations. No Sup. and No Replay retain consistency filtering; Full uses all three components.

Environment	Metric	Baseline	No Sup.	No Replay	**Full**
Qwen3-VL-8B → Qwen3-VL-8B					
Sokoban	avg@8	40.7	44.4	43.5	49.1
	pass@8	87.5	89.0	91.0	92.5
	pass^8	6.5	9.0	5.5	11.0
	State mismatch	46.7	34.7	39.8	12.9
MiniGrid	SR	18.0	20.0	11.6	24.8
	Return	0.10	0.15	0.08	0.21
	Steps	11.3	12.7	11.8	13.7
	State mismatch	50.8	41.8	40.0	23.6
Qwen3-VL-32B → Qwen3-VL-32B					
Sokoban	avg@8	55.9	62.6	54.2	66.9
	pass@8	88.0	91.0	90.5	94.5
	pass^8	23.5	20.5	13.0	38.5
	State mismatch	34.7	27.0	29.9	1.4
MiniGrid	SR	18.8	24.2	23.0	33.9
	Return	0.12	0.19	0.18	0.30
	Steps	13.4	12.5	13.4	13.3
	State mismatch	40.1	22.2	30.3	12.1
Qwen3-VL-32B → Qwen3-VL-8B					
Sokoban	avg@8	44.4	45.2	36.1	54.8
	pass@8	85.5	84.5	85.5	89.5
	pass^8	10.5	12.0	6.5	15.0
	State mismatch	35.9	25.1	29.6	9.5
MiniGrid	SR	11.0	13.4	16.7	27.8
	Return	0.04	0.07	0.13	0.23
	Steps	12.6	11.1	12.5	13.8
	State mismatch	59.6	43.2	49.9	22.2

F.6 Training Analysis

We evaluate the homogeneous 8B and 32B configurations and the 32B-planner/8B-actor configuration on Sokoban and MiniGrid, as in the main text. The analyses use the same evaluation tasks and align pre/post judgments by observation within each event. The baseline is the no-curation training condition used in the component comparison. All discussion is initiated by the trained models; the analysis does not insert ConPAct-I's automatic correction loop.

Initial state judgments. The initial comparison uses the independent planner–actor reports before discussion. Initial D is the fraction of valid initial pairs that disagree, while each role's correctness is its exact-match accuracy against ground truth on the eligible paired observations. Invalid reports are excluded; correctness additionally requires available ground truth. Table 21 compare these initial measurements with ConPAct-R.

Discussion and residual mismatch. Discussion rate is the fraction of initially mismatched events in which the actor initiates discussion. Residual D measures the fraction of those discussed disagreements that remain mismatched after discussion, restricted to events with a valid final pair. Thus, initial D and residual D have different conditioning sets: one describes all valid initial judgments, while the other describes the disagreements on which discussion occurred. Table 22 compares full training with removal of reconciliation supervision or replay.

Paired state composition. The composition analysis uses events with valid assertions and ground truth both before and after the interaction. Each entire assertion pair belongs to one category: CC if both reports agree with ground truth, CW if they agree with each other but are incorrect, or D if the reports disagree. The same eligible events are used for initial percentages and signed post-discussion changes in Table 23. This paired, ground-truth-eligible subset can differ from the initial-only and discussion-conditional sets; their mismatch percentages are not interchangeable. The analysis distinguishes correct agreement from agreement alone, complementing the task-level outcomes and individual-role correctness.

Table 21: Initial state mismatch and assertion correctness before discussion. Rates are percentages and changes are in percentage points.

Environment	Metric	Baseline	**ConPAct-R**	Δ
Qwen3-VL-8B → Qwen3-VL-8B				
	Initial D	47.4	20.8	−26.6
Sokoban	Planner correctness	45.5	83.0	+37.5
	Actor correctness	46.6	83.1	+36.5
	Initial D	51.3	29.9	−21.4
MiniGrid	Planner correctness	50.9	81.5	+30.6
	Actor correctness	50.73	82.4	+31.7
Qwen3-VL-32B → Qwen3-VL-32B				
	Initial D	35.4	10.5	−24.9
Sokoban	Planner correctness	49.3	93.7	+44.4
	Actor correctness	50.3	94.1	+43.8
	Initial D	41.2	18.7	−22.5
MiniGrid	Planner correctness	43.3	87.93	+44.6
	Actor correctness	42.1	88.08	+46.0
Qwen3-VL-32B → Qwen3-VL-8B				
	Initial D	36.4	13.6	−22.8
Sokoban	Planner correctness	42.9	89.4	+46.5
	Actor correctness	44.1	92.1	+48.0
	Initial D	61.2	33.7	−27.5
MiniGrid	Planner correctness	33.3	77.9	+44.6
	Actor correctness	32.1	78.1	+46.0

Table 22: Discussion responses and planner–actor state mismatch (%).

Training data		Sokoban			MiniGrid		
Sup.	Replay	Discuss.	Init. D	Resid. D	Discuss.	Init. D	Resid. D
Qwen3-VL-8B							
✗	✓	8.3	39.1	92.4	7.2	42.7	93.2
✓	✗	21.7	45.9	90.1	19.4	42.2	87.2
✓	✓	58.4	20.8	62.0	54.1	29.9	78.9
Qwen3-VL-32B							
✗	✓	10.5	31.7	90.5	9.4	21.3	91.9
✓	✗	26.8	35.9	88.0	24.2	35.6	87.2
✓	✓	71.6	10.5	13.3	68.3	18.7	64.7
Qwen3-VL-32B → Qwen3-VL-8B							
✗	✓	8.9	27.2	91.1	8.1	45.8	91.9
✓	✗	23.1	36.7	87.2	21.5	61.7	87.1
✓	✓	63.5	13.6	69.9	60.7	33.7	65.9

Table 23: State-assertion consistency and correctness: initial values and signed changes after discussion.

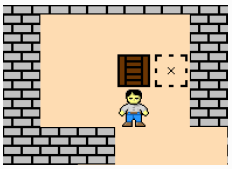
Training data		Sokoban			MiniGrid		
Sup.	Replay	CC $\pm\Delta$	CW $\pm\Delta$	D $\pm\Delta$	CC $\pm\Delta$	CW $\pm\Delta$	D $\pm\Delta$
Qwen3-VL-8B							
✗	✓	40.7+1.5	19.8+1.5	39.5−3.0	38.4+1.6	17.9+1.5	43.7−3.1
✓	✗	28.4+2.4	28.0+2.0	43.6−4.4	36.2+3.3	17.1+2.7	46.7−6.0
✓	✓	72.6+5.1	6.9+2.8	20.5−7.9	67.0+4.1	4.0+2.2	29.0−6.3
Qwen3-VL-32B							
✗	✓	44.8+1.5	24.9+1.5	30.4−3.0	64.4+1.1	10.3+1.1	25.3−2.2
✓	✗	35.4+2.3	31.3+1.8	33.3−4.1	38.2+2.5	27.0+2.0	34.8−4.5
✓	✓	88.7+4.8	2.7+1.3	8.6−6.1	78.7+4.6	4.5+2.0	16.8−6.6
Qwen3-VL-32B → Qwen3-VL-8B							
✗	✓	48.0+1.2	24.3+1.3	27.8−2.5	31.5+2.1	20.4+2.2	48.1−4.3
✓	✗	31.5+2.4	34.1+2.0	34.4−4.4	10.3+4.1	33.2+3.4	56.5−7.5
✓	✓	84.0+2.7	3.1+1.4	12.9−4.1	61.1+7.5	4.3+4.0	34.6−11.5

G CASE STUDIES

G.1 SOKOBAN

Sub-goal: Move **Left**, then **Up** to stand immediately left of the box.

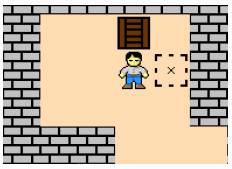
State mismatch



Planner: "The box is directly above you."

Actor: "Up is empty floor; Right is a wall. I will move Up to get a better view."

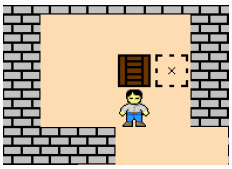
Action: `UP`.



Result: Up pushes the box against the upper wall.

Sub-goal not achieved.

ConPAct coordination



Planner: "The box is directly above you."

Actor: "Up is empty floor; Right is a wall. I will move Up to get a better view."

Proposed action: `UP`.

ConPAct → Actor & Planner

Contradiction: `player_neighbors`
`Up`: Planner=`Box`; Actor=`Cell`
`Right`: Planner=`Cell`; Actor=`Wall`

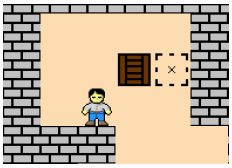
"Recheck both readings; actions paused."

Actor: "Rechecking the image: Up is a box and Right is clear. Up would push the box; I withdraw it."

Planner: "I confirm both readings. Move Left, then Up; stop left of the box."

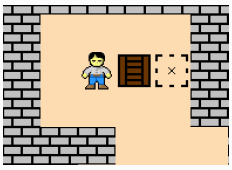
Actor: "Left is clear."

Action: `LEFT`.



Actor: "Up is clear; I can reach the box's left side."

Action: `UP`.



Actor: "The box is directly to my right. Sub-goal achieved."

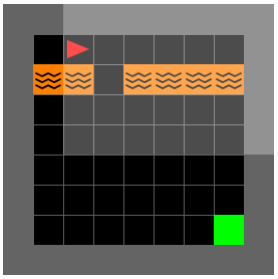
Status: `ACHIEVED`.

Figure 8: **Sokoban: state mismatch and ConPAct coordination.** Condensed recorded outputs for the same positioning sub-goal.

G.2 MiniGrid

Sub-goal: Move one cell **east** along the top row to stand directly above the lava gap.

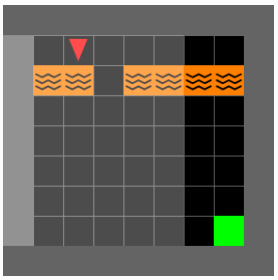
State mismatch



Planner: "The cell ahead is floor; the lava is in the row below."

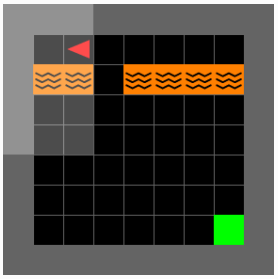
Actor: "East is lava. I will turn south to avoid moving into it."

Action: `right` (turn south).



Actor: "Now lava is directly ahead. I will turn again."

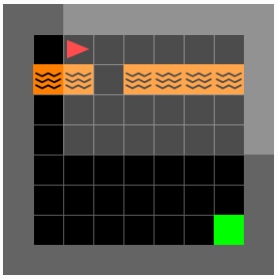
Action: `right` (turn west).



Result: Two turns, no eastward movement. Still one cell left of the gap column.

Sub-goal not achieved.

ConPAct coordination



Planner: "The cell ahead is floor; the lava is in the row below."

Actor: "East is lava. I will turn south to avoid moving into it."

Proposed action: `right`.

ConPAct → Actor & Planner

Contradiction: `front_cell` Planner=`empty`; Actor=`lava`

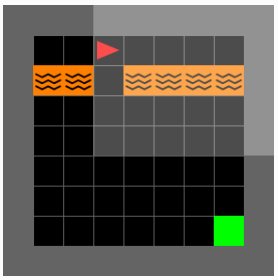
"Recheck both readings; actions paused."

Actor: "The triangle faces east. The dark cell ahead is floor; the orange lava is south. I withdraw the turn."

Planner: "I confirm the east cell is empty. All key and door counts remain zero. Move forward once, then stop above the gap."

Actor: "Agreed: empty ahead, all counts zero. I can move east."

Action: `forward`.



Actor: "I am directly above the gap, still on the top row."

Status: `ACHIEVED`.

Figure 9: **MiniGrid: state mismatch and ConPAct coordination.** Condensed recorded outputs for the same gap-alignment sub-goal.

G.3 CRAFTER

Sub-goal: Harvest the nearby tree down-left to increase wood from **1 to 2**.

State mismatch



Planner: "The first tree is gone. The left cell is clear; another tree is down-left."

Actor: `facing=left, front=tree.`
Action: `do.`



Result: Wood remains 1.
Actor: `front=tree.`
Action: `do` again.



Actor: "The tree is down-left; I need to line up."
Action: `move_down.`



Result: Now moving toward the tree; wood is still 1.

Sub-goal not yet achieved.

ConPAct coordination



Planner: "The first tree is gone. The left cell is clear; another tree is down-left."

Actor: `facing=left, front=tree.`
Proposed action: `do.`

ConPAct → Actor & Planner

Contradiction: `front Planner=walkable; Actor=tree`

"Recheck both readings; actions paused."

Actor: "I face left toward grass. The tree is down-left; I withdraw `do`."

Planner: "Confirmed: left, walkable, full vitals, one wood, no other resources or tools. Move left, then face down."

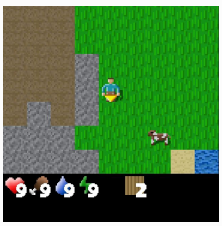
Actor: "Agreed; I will approach it."
Action: `move_left.`



Actor: "The tree is directly below."
Action: `move_down` (face the tree).



Actor: "Facing down, tree ahead."
Action: `do.`



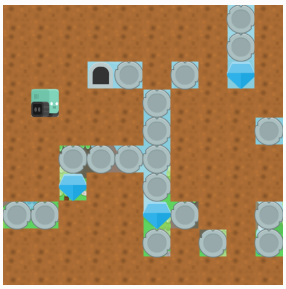
Actor: "The tree is gone; wood is 2."
Status: `ACHIEVED.`

Figure 10: **Crafter: state mismatch and ConPAct coordination.** Condensed recorded outputs for the same second-wood sub-goal.

G.4 PROCGEN MINER

Sub-goal: Move **right one cell** to position for digging a path to the diamonds.

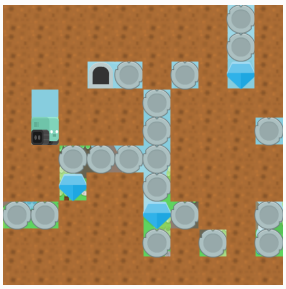
State mismatch



Planner: "No boulder is immediately overhead. Start by moving right."

Actor: "Boulders directly to the right make that move dangerous. I will move down."

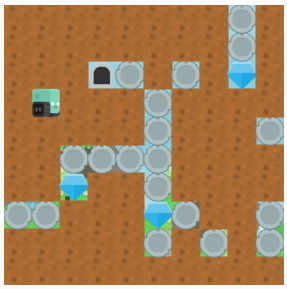
Action: `down`.



Result: The player moves down one cell, remaining in the starting column.

Sub-goal not achieved.

ConPAct coordination



Planner: "No boulder is immediately overhead. Start by moving right."

Actor: "Boulders directly to the right make that move dangerous. I will move down."

Proposed action: `down`.

ConPAct → Actor & Planner

Contradiction: `hazard_nearby` Planner=`no`; Actor=`yes`

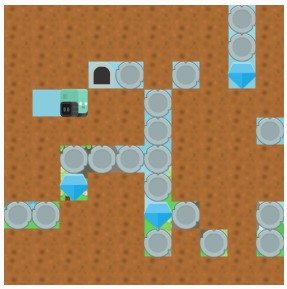
"Recheck both readings; actions paused."

Actor: "The adjacent right cell is dirt; the boulder column is farther away. I revise `hazard_nearby` to `no`. I also count three visible diamonds."

Planner: "I confirm the next right cell is clear and revise my diamond count to three. Diamonds are visible; no immediate falling hazard is apparent. Stop after one right step."

Actor: "Agreed: `target_visible=yes`, `hazard_nearby=no`, three diamonds."

Action: `right`.



Actor: "I have moved one cell right. The positioning sub-goal is complete."

Status: `ACHIEVED`.

Figure 11: **Procgen Miner: state mismatch and ConPAct coordination.** Condensed recorded outputs for the same one-cell positioning sub-goal.

H Discussion

H.1 Inference Cost

Table 24 summarizes the inference-time comparisons across the four game environments. We average rollout costs within each model–environment configuration and then aggregate across environments and evaluated configurations. Calls per action is the ratio of mean model calls to mean executed actions under this aggregation. Output-token averages use the runs with recorded usage; the coverage column reports the fraction with available usage information. All decision calls, including planning and discussion, count toward the common rollout budget.

Table 24: Inference costs averaged across the four environments and the evaluated inference-time configurations. Usage coverage is the fraction of recorded rollouts with output-token usage.

Method	Calls / episode	Output tokens / episode	Calls / action	Usage coverage (%)
ReAct (Planner)	14.02	3,095.37	1.04	71.5
ReAct (Actor)	36.29	8,703.19	1.00	71.5
Plan-Act	35.36	5,285.03	1.06	100.0
HiPlan	51.86	7,992.10	2.10	100.0
TAPE	37.84	18,900.17	6.29	99.9
ConPAct-I	33.02	7,199.86	1.43	83.3

H.2 Interpretation and Scope

State assertions describe selected task-relevant facts rather than complete internal beliefs. Agreement alone does not ensure correctness, so the Sokoban and MiniGrid analyses evaluate the roles' reports against environment ground truth as well as against each other. Procgen and OSWorld allow unknown values for unobservable facts; their task-performance results are not treated as substitutes for an assertion-correctness oracle. The GPT-5.6-sol reviewer curates ConPAct-R data offline and is not required by online ConPAct comparison or reconciliation.

Consistency and correctness. The relationship is asymmetric: when both roles correctly report the same definite facts about the same observation, they must agree, whereas agreement can also reflect a shared error. Disagreement therefore signals that at least one report needs revision, but does not identify which report is correct; agreement leaves correlated errors undetected. This distinction matters especially during reconciliation: a decrease in mismatch can arise either from correcting an erroneous report or from persuading a correct role to adopt an incorrect interpretation. The CC/CW/D analysis separates these outcomes by tracking whether agreement is grounded in the environment or merely shared between roles. Accordingly, reduced mismatch indicates closer alignment of the reported states, but supports a claim of improved state understanding only when accompanied by greater correctness.

Consistency also has a practical role beyond measuring individual accuracy: a planner and an actor need compatible state assumptions for a sub-goal to retain its meaning across their interaction. Yet this compatibility alone cannot establish that the sub-goal is feasible or the chosen action is effective. ConPAct thus addresses an observable source of coordination failure, while shared perceptual errors and planning errors remain distinct limitations. For unobservable fields, preserving uncertainty avoids replacing missing evidence with unsupported consensus; the absence of a definite conflict should not itself be interpreted as verified agreement.