WHAT TRANSFERS FROM SYNTHETIC PLANNING? A CONTROLLED STUDY OF LLM POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthetic planning offers verifiable supervision for language-model agents, yet strong task performance leaves open what these agents learn and transfer. We introduce PlanWorlds, a controlled testbed that connects executable symbolic worlds with natural-language interaction, enabling systematic study of supervision interfaces, training trajectories, and learning objectives. Across two model families and six post-training strategies, we find that planning post-training produces distinct execution and transfer profiles. Holding expert solutions fixed, full-plan and interactive supervision lead to sharply different execution behaviors. Moreover, policies that achieve similarly strong, often near-saturated performance within PlanWorlds can diverge substantially on downstream planning tasks, and further post-training from the same SFT checkpoint can reshape transfer without changing training-environment success. We further show that these transfer gaps have different interpretations. Some largely contract under target-interface controls, while others persist and are accompanied by differences in robustness to local trajectory deviations despite nearly identical clean performance. These findings motivate evaluating planning transfer through complementary views of task success, interface compatibility, and sequential robustness, and enable controlled investigation of what language-model agents actually acquire from synthetic planning post-training.

1 INTRODUCTION

Large language models are increasingly trained as interactive agents that learn through repeated actions and environment feedback (Chen et al., 2025; Kim et al., 2026). Planning is a key challenge for these agents, requiring them to select and adapt sequences of actions to achieve a goal. Synthetic, executable environments offer a scalable source of training experience, with automatically verified outcomes providing feedback for learning (He et al., 2026; Wang et al., 2026b). Their broader promise is to develop behaviors that remain useful beyond the environments in which they are acquired. This raises a central question: *how does the way models learn from verifiable planning experience shape their ability to act in new environments?*

The availability of verifiable planning tasks enables scalable training, but does not by itself reveal what behaviors models acquire. In interactive planning, each action shapes the states encountered next, making training inherently policy-dependent (Ross et al., 2011). Recent work further suggests that different post-training objectives and trajectory sources can lead to markedly different out-of-distribution behavior (Chu et al., 2025; Parmar et al., 2025). This motivates studying how the presentation and use of planning experience shape the behaviors that transfer.

Formal planning provides a useful setting for studying this question under controlled conditions. PDDL and STRIPS specify states, actions, transitions, and goals explicitly, making planning instances executable and their solutions mechanically verifiable. These formalisms have been widely used to evaluate LLM planning (Valmeekam et al., 2023; Silver et al., 2024) and improve planning through fine-tuning (Li et al., 2025; Verma et al., 2025). Their explicit structure also makes it possible to generate new tasks and construct supervision and feedback from a shared executable specification.

We introduce PlanWorlds, a controlled testbed for studying transfer from synthetic planning post-training. PlanWorlds combines procedural task generation, language rendering, and symbolic execu-

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.

tion. A shared task specification supports complete expert plans, stepwise trajectories, and verifiable rewards. This allows us to vary how planning experience is presented, selected, and optimized while preserving a common task pool and execution semantics. Separating symbolic dynamics from their linguistic realization further supports controlled evaluation under changes in language, planning difficulty, and domain dynamics.

Using PlanWorlds, we study two questions. First, how does the *supervision interface* shape the competence that is acquired and transferred? We compare full-plan and interactive supervision constructed from the same expert solutions. Second, how does *further post-training* reshape transfer after interactive competence has been acquired? Starting from a common SFT checkpoint, we compare planner-based correction, self-imitation, and reward optimization. Across two instruction-tuned model families, we evaluate internal generalization, external planning transfer, and retention of broader capabilities. Figure 1 summarizes the main empirical picture. Planning post-training produces distinct internal and downstream performance profiles. Large gains within PlanWorlds do not translate uniformly to external planning tasks.

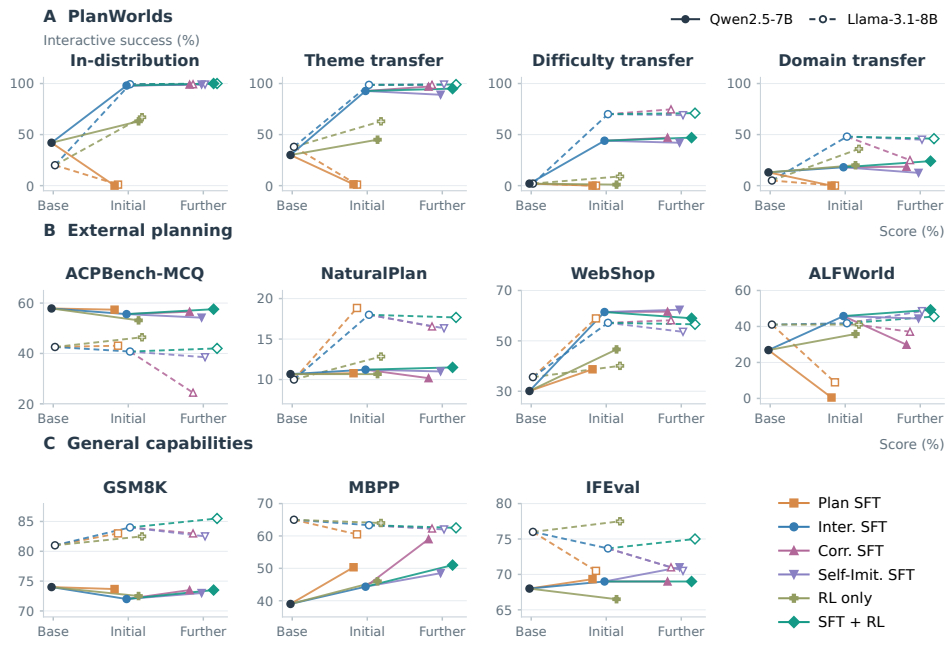

Figure 1: Performance along post-training paths on PlanWorlds, external planning tasks, and general-capability benchmarks. Solid lines and filled markers denote Qwen; dashed lines and open markers denote Llama.

The resulting transfer gaps also require care in interpretation. We probe whether they reflect execution quality within PlanWorlds, compatibility with the target evaluation interface, or robustness to local trajectory deviations. On Llama ACPBench, a large raw gap substantially contracts under explicit answer elicitation. On Qwen ALFWorld, in contrast, the gap between Corrective SFT and SFT+RL remains substantial when generation is restricted to admissible commands. A paired perturbation evaluation further reveals differences in robustness among policies with nearly identical success on clean continuations. These findings motivate evaluating planning transfer through complementary views of task success, target-interface compatibility, and robustness during sequential execution.

Our contributions are threefold. First, we introduce PlanWorlds, a verifiable planning testbed for controlled study of supervision interfaces, policy-generated experience, and learning objectives within shared executable worlds. Second, we show that planning post-training produces distinct execution and transfer profiles: strong competence within the training environment does not uniquely determine downstream behavior, and further training can reshape transfer despite near-saturated internal success. Third, we combine target-interface controls with matched trajectory perturbations to distinguish interface sensitivity from differences in sequential robustness.

2 PLANWORLDS: A CONTROLLED PLANNING TESTBED

PlanWorlds is designed to study how planning behavior changes with supervision and training experience. It separates executable task dynamics from their natural-language presentation and supports expert demonstrations and verifiable feedback within a shared simulator.

Executable planning worlds. Each instance is a STRIPS-style planning problem $P = \langle F, A, s_0, g \rangle$, where F is the set of propositions, A the available actions, s_0 the initial state, and g the goal condition. Action preconditions and effects induce deterministic state transitions, while goal satisfaction provides an automatically verifiable success signal. We procedurally generate instances from 8 planning domains: *Blocksworld*, *Depot*, *Ferry*, *Gripper*, *Logistics*, *Miconic*, *Satellite*, and *Zenotravel*. Figure 2 illustrates an instance. Within each domain, the generator varies initial states, goals, and problem size subject to domain-specific constraints. A symbolic planner computes a reference plan, which is replayed in the simulator to verify executability and goal completion. Consequently, reference demonstrations and internal evaluations are all grounded in the same transition dynamics and executable notion of task success.

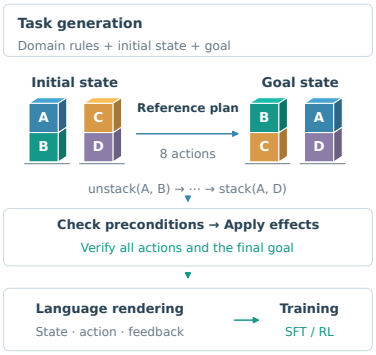

Figure 2: An executable Blocksworld instance in PlanWorlds.

Language rendering with preserved dynamics. Models access these worlds through template-generated descriptions and commands. The initial observation specifies the action interface, world layout, current state, and goal. At each step, the model's command is normalized and matched to a symbolic action, whose preconditions are checked before execution. The simulator returns execution feedback and an updated dynamic-state description, and explicitly reports goal completion. Each training domain has different language themes. Rendering the same instance under different themes preserves its underlying states, actions, and goal, enabling evaluation of linguistic variation without changing the symbolic problem.

Shared tasks, different training experiences. The same reference solution supports two supervision interfaces. Full-plan supervision pairs the task description with the complete action sequence. Interactive supervision replays that sequence to construct an action–observation trajectory, conditioning each action on the preceding interaction history. This comparison preserves the reference solution while changing its presentation and exposure to intermediate feedback. Beyond demonstrations, the simulator executes learner-generated trajectories and verifies their outcomes. Successful rollouts provide self-imitation targets, symbolic replanning supplies expert continuations from intermediate states, and action validity, goal progress, and task completion provide reward signals. These strategies share tasks and execution rules, while differing in trajectory construction, experience selection, and learning objectives.

Controlled generalization. PlanWorlds supports evaluation across themes, difficulty levels, and domains. Generation parameters vary problem size and configuration, with verified reference-plan length serving as a proxy for difficulty. Held-out themes preserve symbolic dynamics, whereas held-out domains introduce different action schemas and domain-specific language. Access to executable intermediate states also supports behavioral interventions. These capabilities connect aggregate task performance with targeted assessments of generalization and robustness.

3 EXPERIMENTAL DESIGN

3.1 MODELS AND TRAINING SETUP

We study *Qwen2.5-7B-Instruct* and *Llama-3.1-8B-Instruct*, using each unmodified instruction-tuned model as the Base condition. These open-weight 7–8B backbones allow us to compare multiple planning post-training strategies under a shared experimental setup.

Our training corpus $\mathcal{D}_{\text{train}}$ contains 9,200 symbolic planning problems from eight domains, each rendered under two language themes, yielding 18,400 natural-language instances. Plan SFT and

Interactive SFT use identical instances and reference action sequences. For continued post-training, downstream conditions are initialized from matched Interactive SFT checkpoints. We report 3 seeds for Plan SFT and Interactive SFT and 2 for each remaining post-training condition. We evaluate the final supervised checkpoints and step 328 for both RL conditions. Interactive SFT is trained for three epochs; Corrective SFT uses one collection–training round, and Self-Imitation SFT uses eight rounds. Because the methods differ in data collection and optimization procedure, we do not attempt to equate their training budgets exactly; instead, we treat each as a complete post-training strategy and report the corresponding optimization and data-collection budgets in Appendix B.

3.2 Post-Training Conditions

We organize six post-training conditions around three contrasts in how planning experience is presented, selected, and optimized, as shown in Table 1.

Condition	Init.	Experience	Interface	Objective
Plan SFT	Base	Expert plans	Full plan	$\mathcal{L}_{\mathrm{plan}}$
Interactive SFT	Base	Expert trajectories	Stepwise	$\mathcal{L}_{\mathrm{act}}$
Corrective SFT	Inter. SFT	Corrected policy traj.	Stepwise	$\mathcal{L}_{\mathrm{act}}$
Self-Imitation SFT	Inter. SFT	Successful policy traj.	Stepwise	$\mathcal{L}_{\mathrm{act}}$
SFT + RL	Inter. SFT	Policy rollouts	Stepwise	J_{RL}
RL Only	Base	Policy rollouts	Stepwise	J_{RL}

Table 1: Overview of the post-training conditions.

Supervision interface: Plan SFT vs. Interactive SFT. Both conditions use the same tasks and expert solutions, but differ in the supervision interface and exposure to intermediate state information. For a task x with reference plan $\mathbf{a}^{\mathrm{ref}} = (a_1^{\mathrm{ref}}, \ldots, a_H^{\mathrm{ref}})$, Plan SFT generates the complete plan in one response, with $\mathcal{L}_{\mathrm{plan}}(\theta) = -\log \pi_\theta(\mathbf{a}^{\mathrm{ref}} \mid x)$. Interactive SFT instead replays the same solution as a multi-turn action–observation trajectory. At step t, the model predicts a_t^{ref} from $h_t = (x, a_{<t}, o_{<t})$, denoting the task and preceding actions and observations. For a trajectory set $\mathcal{D}$, we define the action-level SFT objective as $\mathcal{L}_{\mathrm{act}}(\theta; \mathcal{D}) = -\sum_{\tau \in \mathcal{D}} \sum_{t \in \tau} \log \pi_\theta(a_t \mid h_t)$. Interactive SFT optimizes this objective on replayed expert trajectories. Both datasets are constructed once before training and remain fixed, with Interactive SFT additionally exposing intermediate states and execution feedback.

Use of learner experience: Corrective SFT, Self-Imitation SFT, and SFT+RL. Starting from Interactive SFT, we compare three uses of policy-generated experience. Corrective SFT collects greedy learner rollouts and intervenes when an action is invalid or fails to strictly reduce the remaining plan length returned by a budgeted planner. At the first intervention, the planner completes the trajectory from the state preceding that action. The resulting dataset $\mathcal{D}_{\mathrm{corr}}$ retains the accepted learner prefix and planner continuation, excluding the triggering action and its feedback. The model then optimizes $\mathcal{L}_{\mathrm{act}}(\theta; \mathcal{D}_{\mathrm{corr}})$. This intervention rule uses a planner-based progress proxy, not a certified optimal distance. Self-Imitation SFT iteratively collects successful policy-generated trajectories and optimizes $\mathcal{L}_{\mathrm{act}}(\theta; \mathcal{D}_{\mathrm{succ}})$, using the updated policy for subsequent collection. SFT+RL instead optimizes rewards using fresh policy rollouts at every update.

Supervised initialization for RL: RL Only vs. SFT + RL. Both conditions use the same interaction protocol, reward function, and RL objective, but differ in initialization: RL Only starts from Base, whereas SFT + RL starts from Interactive SFT. Let $G(s) = |g \cap s|$ denote the number of satisfied goal propositions and $\Delta G_t = G(s_{t+1}) - G(s_t)$. Both use $r_t = \alpha \Delta G_t - c - \lambda \mathbf{1}[\mathrm{invalid}_t] + \beta \mathbf{1}[\mathrm{success}_t]$ and maximize $J_{\mathrm{RL}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_t r_t]$, where the terms capture goal progress, step cost, invalid-action penalty, and terminal success bonus.

These conditions share the task pool and execution semantics, while differing in initialization, trajectory distribution, supervision source, objective, and training budget. They characterize complete post-training strategies rather than isolate the causal effect of the learning objective.

3.3 Evaluation Protocol

We evaluate the resulting policies along three complementary dimensions: generalization within PlanWorlds, transfer to external planning tasks, and retention of broader capabilities. Full evaluation protocols are provided in Appendix C.

Internal performance. We evaluate on a separately generated pool with no exact PDDL overlap with training. The evaluation spans four settings: **(1) in-distribution**, with unseen instances from the training domains and themes; **(2) theme transfer**, which holds out the natural-language rendering while preserving the underlying planning domains; **(3) difficulty transfer**, which extends evaluation to longer-horizon instances requiring substantially longer action sequences; and **(4) domain transfer**, which evaluates on the held-out Grid domain with new transition structure and domain-specific language. The first two settings vary instances or language while preserving the training domains, whereas the latter two introduce longer horizons or unseen dynamics.

All four settings are evaluated in both interactive and full-plan modes. We report task success under interactive execution and simulator-verified full-plan success, where every action must be executable and the final state must satisfy the goal. Sampling details and additional transfer probes are provided in Appendix C.

Planning transfer. We next evaluate transfer across a range of increasingly interactive and grounded planning settings. We select these benchmarks to span increasing degrees of interaction and grounding. **(1) ACPBench-MCQ** (Kokel et al., 2025) provides the most abstract setting, probing core reasoning about actions, state transitions, reachability, and plan properties without requiring environment interaction. **(2) NaturalPlan** (Zheng et al., 2024) moves from reasoning about plans to constructing them, requiring models to satisfy temporal and logistical constraints in calendar, meeting, and trip-planning scenarios. **(3) WebShop** (Yao et al., 2022) introduces closed-loop web interaction through sequential search, comparison, and purchase decisions. **(4) ALFWorld** (Shridhar et al., 2021) provides a long-horizon interactive setting, requiring agents to navigate and manipulate a household environment over extended action sequences while adapting to the evolving state.

General capabilities. We evaluate whether planning post-training preserves broader capabilities using **(1) GSM8K** (Cobbe et al., 2021) for mathematical reasoning; **(2) MBPP** (Austin et al., 2021) for code generation; and **(3) IFEval** (Zhou et al., 2023) for instruction following. Retention is assessed separately on each benchmark relative to the corresponding base model.

4 What Transfers from Planning Post-Training?

We first ask what aspects of behavior acquired in PlanWorlds persist beyond the training setting. Our results reveal a recurring disconnect between competence measured under the training interface and behavior observed after transfer: models with strong or even near-saturated performance in PlanWorlds can exhibit substantially different execution and transfer profiles. Complete absolute scores are reported in Appendix D.

Execution behavior is strongly interface-dependent. Plan SFT and Interactive SFT use the same tasks and expert solutions, but differ in the supervision interface and exposure to intermediate state information. As shown in Table 2, the resulting policies exhibit a pronounced cross-interface asymmetry. Plan SFT achieves 97–99% full-plan success in distribution, yet almost completely fails under stepwise interaction. Interactive SFT reaches 98–99% interactive success while remaining substantially weaker at full-plan generation. This asymmetry also appears under held-out language themes. Holding tasks and expert solutions fixed therefore does not produce comparable success across execution interfaces. These results show that execution behavior depends strongly on how planning experience is presented during training and how the resulting policy is evaluated; cross-interface failure alone does not establish a lack of underlying planning ability.

		Interactive success (%) ↑		**Full-plan success** (%) ↑	
Model	**Training**	**In-Dist.**	**Theme**	**In-Dist.**	**Theme**
Qwen2.5-7B	Plan SFT	0.0 ± 0.0	1.3 ± 1.2	**97.3** ± 2.3	**66.0** ± 7.2
	Inter. SFT	**98.0** ± 0.0	**92.7** ± 1.2	35.3 ± 7.0	24.7 ± 4.2
Llama-3.1-8B	Plan SFT	1.0 ± 1.4	1.0 ± 1.4	**99.0** ± 1.4	**89.0** ± 1.4
	Inter. SFT	**99.3** ± 1.2	**98.7** ± 1.2	22.0 ± 10.4	12.0 ± 2.0

Table 2: Supervision interface shapes execution behavior.

Strong in-environment competence does not predict transfer. We next ask whether improvements within PlanWorlds carry over to planning tasks with different interfaces, dynamics, and inter-



Figure 3: Performance change after planning post-training. Each cell reports the percentage-point change from the corresponding Base model.

action structures. Figure 3 reports changes relative to the corresponding Base model. The answer is strongly target dependent. For Qwen, Interactive SFT produces large gains on WebShop and ALFWorld, while producing little change on ACPBench-MCQ and NaturalPlan. Llama exhibits a different transfer profile: Interactive SFT improves WebShop and NaturalPlan, but yields little improvement on ALFWorld or ACPBench-MCQ. Plan SFT likewise transfers unevenly across targets. These results show that success under the training interface is not by itself a reliable indicator of downstream improvement. Even when the same post-training procedure produces strong competence within PlanWorlds, the extent to which that competence is useful depends on the target environment.

Further post-training changes transfer despite similarly high in-distribution success. A sharper separation appears when we continue training from the same Interactive SFT checkpoint. Corrective SFT, Self-Imitation SFT, and SFT+RL all retain near-saturated in-distribution interactive performance in PlanWorlds, yet their downstream behavior diverges substantially. Qwen ALFWorld provides the clearest example. Interactive SFT reaches 45.8% success. Continued Corrective SFT reduces this to 29.9%, Self-Imitation SFT largely preserves it at 44.4%, whereas SFT+RL increases it to 49.3%. Importantly, this pattern should not be interpreted as a universal ordering of the training strategies. The relative effects vary across downstream tasks and model families. On Llama, for example, Self-Imitation SFT improves ALFWorld relative to Interactive SFT, while Corrective SFT exhibits a pronounced drop on the raw ACPBench-MCQ evaluation.

The consistent observation is therefore not that one post-training strategy transfers best. Rather, policies that remain similarly successful in the training environment can acquire markedly different downstream transfer profiles. Continued post-training can preserve, weaken, or extend previously acquired transfer without producing a corresponding change in PlanWorlds task success.

General Capabilities Are Broadly Retained. The planning-specific changes above are not accompanied by a uniform degradation of broader capabilities. Across GSM8K, MBPP, and IFEval, most post-trained models remain close to their corresponding Base model, with method-specific gains and losses. Qwen shows several substantial improvements on MBPP, whereas changes for Llama are generally smaller across the three benchmarks.

5 What Explains Transfer Gaps?

Section 4 shows that policies with similar performance within PlanWorlds can diverge substantially on downstream planning tasks. We next ask what these gaps reflect. We examine compatibility with

the target evaluation interface, and robustness to local trajectory deviations. Execution diagnostics within the training environment are reported in Appendix E.

5.1 Target-Interface Controls Reveal Different Kinds of Transfer Gaps

A downstream score reflects both a policy's decisions and how they are expressed through the target evaluation interface. We revisit two prominent transfer gaps using task-specific interface controls and observe different responses.

ACPBench: the raw gap is strongly protocol-sensitive. For Llama-3.1-8B-Instruct, Corrective SFT's low raw ACPBench-MCQ accuracy coincides with frequent failures to produce an extractable option letter. We append a second turn explicitly requesting one option from $\{A, B, C, D\}$, retaining the original response in the interaction history and restricting generation to these four choices. We focus on conditions relevant to the shared Interactive-SFT continuation comparison. Because this intervention permits answer revision, we examine response transitions rather than attributing accuracy gains solely to improved extraction. Table 3 separates responses by their original status: unextractable, extractable but incorrect, or correct. For each category, we report the fraction that is correct after re-elicitation, together with the net accuracy change.

	Correct after elicitation (%)			Δ Acc.
Condition	Unextractable	Wrong	Correct	(pp)
Base	45.2	5.2	96.8	+6.3
Inter. SFT	44.3	3.6	97.1	+4.5
Corr. SFT	44.0	**19.5**	**85.1**	**+24.3**
Self-Imit. SFT	51.7	2.7	97.4	+5.8
SFT + RL	48.4	3.3	98.5	+4.4

Table 3: ACPBench answer transitions, conditioned on original response status.

Corrective SFT exhibits a distinct response to re-elicitation. Among its previously unextractable responses, 44.0% become correct, a rate comparable to those of the other conditions. This recovery alone cannot distinguish improved answer expression from newly generated correct decisions. The clearer distinction concerns already extractable answers: 19.5% of its incorrect responses become correct, compared with 2.7–5.2% elsewhere, while only 85.1% of its correct responses remain correct, compared with 96.8–98.5%. Re-elicitation therefore changes answer correctness beyond making previously unextractable responses scorable. Overall, Corrective SFT gains 24.3 percentage points, compared with 4.4–6.3 points for the other conditions. Its raw ACPBench performance is thus strongly protocol-sensitive, with both recovery from unextractable responses and greater revision of already extractable answers. The improvement cannot be interpreted as a pure correction of formatting errors. Complete raw and elicited accuracies are provided in Appendix E.

ALFWorld: the gap persists under admissible-command decoding. For Qwen2.5-7B-Instruct, Corrective SFT and SFT + RL start from the same Interactive SFT checkpoint and use the same training task pool, yet diverge substantially on ALFWorld. Under the standard free-form evaluation, outputs that do not exactly match an available command are mapped to an admissible command before execution. We define the rescue rate as the fraction of actions requiring this mapping within each episode, averaged across episodes. To further control for this interface effect, we constrain decoding directly to the commands available in the current state.

	Task success (%)			Rescue rate (%)		
Condition	Free-form	Constr.	Δ	All	Success	Failure
Base	26.9	34.3	+7.5	34.9	14.4	42.4
Inter. SFT	45.8	47.5	+1.7	23.1	14.0	30.7
Corr. SFT	29.9	39.9	**+10.1**	**36.4**	**21.9**	**42.5**
Self-Imit. SFT	44.4	50.0	+5.6	22.6	13.5	30.1
SFT + RL	49.3	56.3	+7.1	14.7	9.2	20.2

Table 4: ALFWorld command-interface control. Δ denotes the gain from constrained decoding. Rescue rate is the fraction of generated actions mapped to an admissible command, averaged over episodes.

As shown in Table 4, constrained decoding improves all conditions and benefits Corrective SFT the most, raising its success rate by 10.1 points from 29.9% to 39.9%. The gap nevertheless remains substantial: under the same constraint, SFT + RL reaches 56.3%, leaving a 16.4-point difference. Corrective SFT also relies more heavily on command rescue, with a rescue rate of 36.4% compared with 14.7% for SFT + RL. Across all conditions, failed episodes exhibit higher rescue rates than successful ones. These results indicate that command-interface mismatch contributes to ALFWorld performance, but does not fully explain the transfer gap. Since constrained decoding can also alter action choices and subsequent trajectories, we interpret this control as evidence of interface sensitivity rather than a purely syntactic correction.

5.2 Local Deviations Reveal Differences in Robustness

The interface controls above establish whether downstream scores are sensitive to the target protocol, but they do not reveal whether policies with similar clean performance behave similarly once their trajectories depart from a reference path. We therefore probe robustness to a controlled local deviation.

We conduct a paired evaluation on 160 held-out in-distribution problems from the eight PlanWorlds domains, using Qwen2.5-7B-Instruct for each condition. For each problem, we replay a planner-generated prefix to an intermediate state s. The clean branch continues from s, whereas the perturbed branch first executes one legal, state-changing action and receives the resulting observation. The perturbed state s' remains solvable and satisfies $\delta = d^*(s') - d^*(s) \in \{0, 1\}$, where d^* denotes the verified shortest remaining plan length. All policies receive the same paired histories, and each branch is allowed ten additional actions beyond its own optimal remaining plan length.

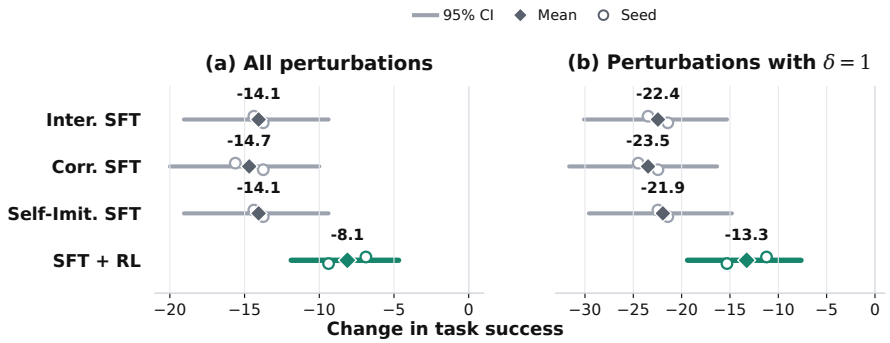

Figure 4: Success changes (pp) under matched local trajectory perturbations. Error bars show 95% CIs.

Figure 4 reveals differences that are hidden by clean task performance. All four conditions achieve 99.7–100% success on the clean continuations. After perturbation, Interactive SFT, Corrective SFT, and Self-Imitation SFT lose approximately 14–15 percentage points, whereas SFT + RL loses 8.1 points. The separation is larger when the perturbation increases the optimal remaining distance ($\delta = 1$): the three SFT conditions lose 21.9–23.5 points, compared with 13.3 points for SFT + RL. Thus, near-identical performance along clean trajectories does not imply identical robustness to locally suboptimal states. In this controlled probe, SFT + RL is less sensitive to the injected deviation than the evaluated SFT alternatives.

The similar perturbation sensitivity of Corrective SFT and the other SFT conditions is consistent with its trajectory construction. Planner intervention replaces the triggering action with a continuation from the preceding state, excluding that action and its feedback from training. Corrective SFT therefore does not directly supervise recovery from the resulting state, which may help explain its limited robustness gains in this probe. However, the similar perturbation losses across the SFT conditions do not explain their different ALFWorld scores, so this observation does not provide a complete account of external transfer.

Our results suggest that transfer from synthetic planning is not a single scalar property. Near-saturated in-environment performance can coexist with substantial downstream differences; downstream scores can also reflect compatibility with the target interaction protocol; and policies with similar clean execution can differ in robustness to local trajectory deviations. The ACPBench and

ALFWorld controls further show that these effects are target dependent. These comparisons should therefore be interpreted at the level of complete post-training strategies, which differ in trajectory distribution, supervision source, objective, and training budget, rather than as isolating SFT or RL alone. Likewise, the perturbation probe captures one aspect of sequential robustness but does not fully explain external transfer. Together, these results motivate evaluating planning transfer through complementary measures of in-environment success, interface compatibility, and robustness.

6 Related Work

Planning with Language Models. Language-model planning has been approached through reasoning–action interleaving, explicit search, and trajectory-based adaptation. ReAct interleaves reasoning with environment actions (Yao et al., 2023b), while Tree of Thoughts and RAP introduce explicit search over intermediate reasoning states (Yao et al., 2023a; Hao et al., 2023). Trajectory-based approaches further adapt language models for interactive decision making, for example through agent-oriented fine-tuning (Zeng et al., 2024). A complementary line combines language models with symbolic planning. LLM+P translates natural-language tasks into PDDL and delegates plan synthesis to a classical planner (Liu et al., 2023), while generalized-planning work uses language models to synthesize reusable planning programs over PDDL domains (Silver et al., 2024). PDDL-Instruct instead improves planning behavior through logical instruction tuning (Verma et al., 2025), and related work has emphasized the value of external symbolic verifiers when using LLMs for planning (Kambhampati et al., 2024).

Post-Training Objectives and Generalization. A growing body of work studies how supervised and reward-based post-training shape reasoning and generalization. Self-training methods such as STaR bootstrap reasoning from model-generated solutions (Zelikman et al., 2022), while verifiable-reward approaches such as DeepSeekMath optimize reasoning from automatically checkable outcomes (Shao et al., 2024). PLAN-TUNING further compares supervised and reinforcement-learning objectives on synthetic planning trajectories and reports gains in out-of-domain reasoning (Parmar et al., 2025). Comparative studies suggest that SFT and RL can induce different generalization behavior: Chu et al. (2025) report stronger out-of-distribution generalization from outcome-based RL than SFT in controlled settings, while recent analyses examine how these objectives reshape learned representations (Shi et al., 2026). Relatedly, Huan et al. (2025) show that broader transfer from reasoning post-training depends strongly on the training method. We extend these questions to executable planning, comparing supervision format, learner-generated trajectories, and reward optimization under shared transition dynamics.

Planning Testbeds and Transfer Evaluation. Several benchmarks evaluate language-model planning at different levels of abstraction. PlanBench targets plan generation and reasoning in formally specified domains (Valmeekam et al., 2023), while ACPBench evaluates reasoning about actions, state changes, reachability, and plan properties (Kokel et al., 2025). NaturalPlan focuses on planning under natural-language temporal and logistical constraints (Zheng et al., 2024). Interactive benchmarks such as ALFWorld and WebShop instead require multi-step action under evolving observations (Shridhar et al., 2021; Yao et al., 2022). Recent work also uses procedurally generated executable environments for scalable agent training: ReSyn constructs verifiable synthetic environments for reasoning post-training (He et al., 2026), while Agent World Model generates large collections of executable environments for agentic reinforcement learning (Wang et al., 2026a).

7 Conclusion

We introduced PlanWorlds to study planning post-training and transfer through shared symbolic tasks and verifiable execution. Across two model families, supervision format shapes execution behavior, while transfer gains depend on the training strategy and target environment. Interface controls show that some apparent performance losses substantially diminish under explicit answer elicitation, whereas others persist under admissible-command decoding. A paired perturbation evaluation further reveals greater robustness to local trajectory deviations for SFT-initialized RL than for the evaluated SFT alternatives. These findings highlight the value of combining downstream outcomes, interface controls, and behavioral probes to assess what transfers from synthetic planning.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. doi: 10.48550/arXiv.2108.07732. URL https://arxiv.org/abs/2108.07732.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents, 2025.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/chu25c.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/.

Andre He, Nathaniel Weir, Kaj Bostrom, Allen Nie, Darion Cassel, Sam Bayless, and Huzefa Rangwala. Resyn: Autonomously scaling synthetic environments for reasoning models, 2026.

Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general LLM capabilities? understanding transferability of LLM reasoning. *arXiv preprint arXiv:2507.00432*, 2025. URL https://arxiv.org/abs/2507.00432.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B. Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 22895–22907. PMLR, 2024. URL https://proceedings.mlr.press/v235/kambhampati24a.html.

Sunghwan Kim, Junhee Cho, Beong-woo Kwak, Taeyoon Kwon, Liang Wang, Nan Yang, Xingxing Zhang, Furu Wei, and Jinyoung Yeo. On training large language models for long-horizon tasks: An empirical study of horizon length. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. ACPBench: Reasoning about action, change, and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. URL https://arxiv.org/abs/2410.05669.

Wenjun Li, Changyu Chen, and Pradeep Varakantham. Unlocking the planning capabilities of large language models with maximum diversity fine-tuning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3318–3340. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.183.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023. doi: 10.48550/arXiv.2304.11477. URL https://arxiv.org/abs/2304.11477.

Mihir Parmar, Palash Goyal, Xin Liu, Yiwen Song, Mingyang Ling, Chitta Baral, Hamid Palangi, and Tomas Pfister. PLAN-TUNING: Post-training language models to learn step-by-step planning for complex problem solving. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21419–21433. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1087. URL https://aclanthology.org/2025.emnlp-main.1087/.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Dan Shi, Zhuowen Han, Simon Ostermann, Renren Jin, Josef van Genabith, and Deyi Xiong. Why does reinforcement learning generalize? a feature-level mechanistic study of post-training in large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B. Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. Generalized planning in PDDL domains with pretrained large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2305.11014.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2206.10498.

Pulkit Verma, Ngoc La, Anthony Favier, Swaroop Mishra, and Julie A. Shah. Teaching llms to plan: Logical chain-of-thought instruction tuning for symbolic planning, 2025.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://arxiv.org/abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b. URL https://arxiv.org/abs/2210.03629.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2203.14465.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. NATURAL PLAN: Benchmarking LLMs on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024. doi: 10.48550/arXiv.2406.04520. URL https://arxiv.org/abs/2406.04520.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. doi: 10.48550/arXiv.2311.07911. URL https://arxiv.org/abs/2311.07911.

A Data

Training Data and Evaluation Settings. PlanWorlds represents planning tasks as executable symbolic problems rendered in natural language. Our post-training corpus contains 9,200 symbolic problems across eight domains: *Blocksworld*, *Depot*, *Ferry*, *Gripper*, *Logistics*, *Miconic*, *Satellite*, and *Zenotravel*. Each problem is rendered under two surface themes, yielding 18,400 natural-language training instances.

Evaluation problems are generated independently of the training corpus and organized into four settings. **In-distribution** evaluation uses new problems from the source domains within the training difficulty range and under the training themes. **Theme transfer** renders the same underlying evaluation problems under a third theme excluded from post-training, varying surface form while holding the symbolic task fixed. **Difficulty transfer** uses more challenging problems from the source domains while retaining the training themes. **Domain transfer** uses the separately constructed *Grid* domain, which is absent from post-training. Its traversal rules depend on key–shape compatibility, requiring agents to acquire suitable keys before accessing some locations. Evaluated sample sizes and interaction protocols are provided in the evaluation details.

Theme Rendering. Theme transfer changes only the linguistic realization of a problem while preserving its symbolic states, actions, transitions, and goals. For example, the same Gripper state may be rendered as "you are at the library," "your rover hovers at the wreck bay," or "you are in the lab" under different themes, while all three correspond to the same symbolic fact `(at-robby room3)`. Actions are transformed analogously, with theme-specific expressions such as `walk`, `steer`, and `go` mapping to the same underlying move operator. Theme transfer therefore isolates generalization to unseen linguistic realizations without changing the underlying planning problem.

Reference Plan Lengths. Table 5 quantifies the planning-horizon shift across PlanWorlds splits. Difficulty-transfer reference plans average 24.3 actions, compared with 7.9 in training and 6.5 in-distribution. Their length range is disjoint from training by construction: training references contain at most 12 actions, whereas difficulty-transfer references contain at least 13. Held-out Grid references remain within the training length range, providing a complementary evaluation of transfer to new dynamics.

Train–Evaluation Separation. Training and evaluation problems are generated independently. An exact comparison using MD5 hashes of the PDDL problem files finds no byte-identical overlap between the post-training and evaluation pools. All post-training conditions use only the post-training pool, including the collection of learner-generated trajectories for Corrective SFT, Self-Imitation SFT, and RL. Evaluation problems, the held-out surface theme, and the unseen domain are excluded from supervised training and RL optimization.

Split	Mean ± SD	Median	IQR	Range
Training pool	7.9 ± 2.8	8	[6, 10]	2–12
In-distribution	6.5 ± 3.0	5.5	[4, 9]	2–12
Theme transfer	6.5 ± 3.0	5.5	[4, 9]	2–12
Difficulty transfer	$\mathbf{24.3 \pm 9.4}$	**22**	**[17, 31]**	**13–60**
Held-out Grid	9.1 ± 1.8	9	[8, 10]	6–12

Table 5: Reference-plan lengths across the constructed data pools.

Plan Validity and Optimality. We validate reference plans by replaying their action sequences in the symbolic simulator, checking action applicability at every step and goal satisfaction in the terminal state. All symbolic problems in the post-training pool have certified-optimal reference plans. Task success is determined by goal satisfaction and does not depend on reference-plan optimality. For evaluation problems with uncertified references, a successful model trajectory may be shorter than the stored plan. Reference-relative step differences should therefore be interpreted relative to the stored solution rather than as optimality gaps.

B Training

B.1 Training Setup

We train *Qwen2.5-7B-Instruct* and *Llama-3.1-8B-Instruct*. Plan SFT, Interactive SFT, and RL Only start from the corresponding instruction-tuned base model; Corrective SFT, Self-Imitation SFT, and SFT + RL continue from Interactive SFT. Plan SFT and Interactive SFT use the same 18,400 instances and reference actions, differing in their supervision format.

All supervised conditions use full-parameter fine-tuning with assistant-only loss, native tokenizers and chat templates, and within-batch padding without sequence packing. We report final supervised checkpoints and step 328 for RL-based conditions. Optimization settings are summarized in Table 6.

Supervised fine-tuning		**Reinforcement learning**	
Learning rate	1×10^{-5}	Learning rate	1×10^{-6}
Optimizer	AdamW	Algorithm	GRPO
Schedule	Cosine	Prompts per update	32
Warmup ratio	0.03	Rollouts per prompt	8
Weight decay	0	Policy clipping	0.2
Gradient clipping	1.0	KL coefficient	1×10^{-3}
Per-device batch size	1	Entropy coefficient	0
Gradient accumulation	16	Prompt token limit	2,048
GPUs / global batch	2 / 32	Response token limit	8,192
Sequence token limit	8,192	Updates per step	1
Precision	bfloat16	Reported checkpoint	Step 328

Table 6: Optimization settings for supervised and RL post-training.

B.2 Trajectory Construction and Optimization

Plan SFT. The model predicts a complete reference action sequence from the initial problem description, without intermediate environment feedback.

Interactive SFT. The same reference actions are presented as an interaction trajectory, with the resulting state provided after each action. The 18,400 expert trajectories contain 145,966 next-action targets, averaging 7.9 actions per instance.

Corrective SFT. We collect one round of greedy rollouts from Interactive SFT and intervene at the first invalid action or action that fails to reduce the planner-returned remaining plan length. Specifically, a valid action from s_t to s_{t+1} is accepted when $\widehat{L}(s_{t+1}) < \widehat{L}(s_t)$, where $\widehat{L}(s)$ is the length of the plan returned from state s. Upon intervention, the planner completes the task from the state preceding the triggering action. The supervised trajectory consists of the accepted learner

prefix and planner continuation; the triggering action and its feedback are excluded. Replanning uses A* with LM-cut and a four-second budget, falling back to greedy best-first search with hFF and an eight-second budget when necessary. The acceptance rule uses returned plan lengths without requiring optimality certification or matching a particular reference action. Thus, $\widehat{L}$ provides a planner-based progress signal, not a guaranteed shortest-path distance.

Self-Imitation SFT. We alternate rollout collection and supervised updating for eight rounds. Each round samples eight trajectories per prompt for 1,312 prompts at temperature 1.0. Repeated symbolic states within a retained trajectory contribute at most one target, taken from the learner's original response. We train for one epoch on each round's retained targets before collecting the next round. This procedure samples 83,968 trajectories before filtering and state-level deduplication.

Reinforcement Learning. RL Only and SFT + RL share the same GRPO configuration and reward function, differing only in initialization. Let $G(s) = |g \cap s|$ count the satisfied goal propositions. The per-step reward is $r_t = G(s_{t+1}) - G(s_t) - 0.05 - 0.2\,\mathbf{1}[\text{invalid}_t] + 2.0\,\mathbf{1}[\text{success}_t]$, combining goal progress, a step cost, an invalid-action penalty, and a terminal success bonus. Episode returns are normalized within each prompt group to obtain advantages. Each step collects eight trajectories for each of 32 prompts and performs one gradient update, without an inner multi-epoch PPO loop. The reported step-328 checkpoint therefore corresponds to 10,496 prompt draws and 83,968 sampled trajectories per run; prompts may recur across updates.

These conditions compare complete post-training strategies, which differ in trajectory distribution, supervision, filtering, initialization, and optimization. Although Self-Imitation SFT and RL have equal nominal rollout counts, their token exposure and compute are not necessarily matched. Observed differences therefore cannot be attributed to the learning objective alone.

C Evaluation

Within each benchmark, all post-training conditions are evaluated on the same instances using the same task prompts, interaction protocols, and scoring procedures, without condition-specific prompt tuning. External benchmarks retain their task-specific interfaces.

PlanWorlds Evaluation. We evaluate interactive and single-shot planning, using 100 instances for each of In-Dist., Theme, Difficulty, and the Grid domain-transfer setting. Task success is determined by goal satisfaction in the symbolic simulator. Interactive evaluation measures closed-loop completion, whereas single-shot evaluation executes the predicted action sequence and checks its terminal state.

External Planning Transfer. Table 7 summarizes the benchmarks, sample sizes, and reported metrics. ALFWorld uses the full `eval_out_of_distribution` split with a 50-step budget. The model receives the current observation and admissible commands; under the standard free-form protocol, non-matching outputs are mapped to admissible commands before execution. We report environment-defined task success and separately track the fraction of actions requiring this mapping. WebShop (Yao et al., 2022) uses sessions 0–199 with a 20-step budget and is scored by the mean native reward for the purchased product, reflecting its match to the requested attributes, options, and price. NaturalPlan covers calendar, meeting, and trip planning with the benchmark's task-specific few-shot prompts and scoring procedures. ACPBench-MCQ uses seven multiple-choice subtasks with their standard task configurations and answer-extraction rules. Each subtask contains 100 evaluated instances, and both benchmarks are reported as the unweighted mean across subtasks.

General-Capability Evaluation. We evaluate 100 instances each from GSM8K, MBPP, and IFEval with five-shot, three-shot, and zero-shot prompting, respectively, using greedy decoding with sampling disabled. GSM8K uses the test split of the `main` configuration and reports exact-match accuracy with `flexible-extract` answer extraction. MBPP uses the test split of the `full` configuration and reports pass@1 from one generated completion per problem. IFEval uses a generation limit of 1,280 tokens. We report prompt-level strict accuracy, requiring every instruction in a prompt to pass its strict verification rule.

Paired Perturbation Evaluation. We assess robustness to controlled trajectory deviations on 160 in-distribution problems, with 20 per source domain. For each domain, we scan a deterministically shuffled evaluation pool and retain the first 20 problems admitting a qualifying perturbation.

Benchmark	N	Metric
Planning transfer		
ALFWorld	134	Task success
WebShop	200	Mean task reward
NaturalPlan	3×100	Mean subtask score
ACPBench-MCQ	7×100	Mean accuracy
General capabilities		
GSM8K	100	Exact match
MBPP	100	Pass@1
IFEval	100	Prompt-level strict accuracy

Table 7: External evaluation suite. N denotes evaluated instances; multipliers indicate subtasks.

Each problem yields a clean–perturbed pair constructed from a shared reference trajectory prefix. The problems, prefixes, and perturbations are fixed before model evaluation and shared across all conditions and training seeds. The intervention state s is selected from the middle 30–70% of the reference trajectory, subject to $d^*(s) \geq 2$, where d^* denotes the shortest remaining plan length. We sample a legal, state-changing action leading to a solvable state s' with $\delta = d^*(s') - d^*(s) \in \{0, 1\}$. Both distances must be certified optimal by A* with LM-cut. The resulting set contains 62 pairs with $\delta = 0$ and 98 with $\delta = 1$. The clean branch continues directly from s; the perturbed branch appends the selected action and its actual environment feedback to the shared history, then continues from s'. Neither branch receives an explicit recovery instruction.

We evaluate Interactive SFT, Corrective SFT, Self-Imitation SFT, and SFT + RL on Qwen2.5-7B-Instruct, using two training seeds per condition. Continuation uses greedy decoding with at most 2,048 tokens per response, without action mapping, retries, or constrained decoding. Each branch receives a continuation budget of $d^*(s_{\text{start}}) + 10$ steps, providing equal slack relative to its shortest remaining plan. The shared prefix and inserted action do not consume this budget; invalid or unparsable model responses do. We report clean success, perturbed success, and their paired difference, averaging first over problems within each training seed and then across seeds. All 160 pairs are included in every run, regardless of clean-branch success. We compute 95% percentile confidence intervals using 10,000 bootstrap resamples of problems, retaining both branches and both training seeds together whenever a problem is sampled.

D Complete Main Results

Tables 8 and 9 provide the complete main results for both model families. The internal evaluation covers four transfer settings under interactive and single-shot interfaces; the external evaluation covers four planning benchmarks and three general-capability benchmarks. We report means and standard deviations across training seeds.

	Interactive success (%)				Full-plan success (%)			
Condition	**In-Dist.**	**Theme**	**Difficulty**	**Domain**	**In-Dist.**	**Theme**	**Difficulty**	**Domain**
Qwen2.5-7B-Instruct								
Base	42.0	30.0	2.0	13.0	4.0	6.0	0.0	0.0
Plan SFT	0.0 ± 0.0	1.3 ± 1.2	0.0 ± 0.0	0.0 ± 0.0	**97.3** ± 2.3	**66.0** ± 7.2	**8.0** ± 2.0	0.0 ± 0.0
Interactive SFT	98.0 ± 0.0	92.7 ± 1.2	44.0 ± 3.5	18.0 ± 3.0	35.3 ± 7.0	24.7 ± 4.2	1.3 ± 1.2	0.0 ± 0.0
Corrective SFT	99.0 ± 1.4	**97.0** ± 1.4	**47.0** ± 1.4	18.5 ± 4.9	18.0 ± 2.8	10.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Self-Imit. SFT	99.0 ± 1.4	89.0 ± 1.4	42.0 ± 2.8	12.5 ± 4.9	21.0 ± 7.1	14.0 ± 2.8	0.0 ± 0.0	0.0 ± 0.0
RL Only	63.0 ± 9.9	45.0 ± 1.4	1.0 ± 1.4	20.0 ± 11.3	5.0 ± 4.2	12.0 ± 2.8	0.0 ± 0.0	0.0 ± 0.0
SFT + RL	**100.0** ± 0.0	95.0 ± 1.4	**47.0** ± 9.9	**24.0** ± 1.4	16.0 ± 8.5	12.0 ± 11.3	0.0 ± 0.0	0.0 ± 0.0
Llama-3.1-8B-Instruct								
Base	20.0	38.0	2.0	5.0	6.0	0.0	0.0	0.0
Plan SFT	1.0 ± 1.4	1.0 ± 1.4	0.0 ± 0.0	0.0 ± 0.0	**99.0** ± 1.4	**89.0** ± 1.4	**28.0** ± 0.0	0.0 ± 0.0
Interactive SFT	99.3 ± 1.2	98.7 ± 1.2	70.0 ± 2.0	**48.0** ± 0.0	22.0 ± 10.4	12.0 ± 2.0	0.7 ± 1.2	0.0 ± 0.0
Corrective SFT	99.0 ± 1.4	**100** ± 0.0	**72** ± 0.0	25.0 ± 0.0	5.0 ± 7.1	3.0 ± 4.2	2.0 ± 2.8	0.0 ± 0.0
Self-Imit. SFT	99.0 ± 1.4	99.0 ± 1.4	69.0 ± 1.4	45.0 ± 1.4	14.0 ± 0.0	8.0 ± 2.8	0.0 ± 0.0	0.0 ± 0.0
RL Only	67.0 ± 4.2	63.0 ± 7.1	9.0 ± 4.2	36.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
SFT + RL	**100.0** ± 0.0	99.0 ± 1.4	71.0 ± 4.2	46.0 ± 4.2	28.0 ± 0.0	11.0 ± 1.4	0.0 ± 0.0	0.0 ± 0.0

Table 8: Internal performance on PlanWorlds (%). Interactive success measures closed-loop goal completion; full-plan success measures goal completion after executing a single-shot action sequence in the simulator.

	Planning transfer				General capabilities		
Condition	**ACPBench MCQ**	**NaturalPlan**	**WebShop**	**ALFWorld**	**GSM8K**	**MBPP**	**IFEval**
Qwen2.5-7B-Instruct							
Base	**57.86**	10.67	30.09	26.87	**74.00**	39.00	68.00
Plan SFT	57.38 ± 2.26	10.78 ± 0.51	38.71 ± 2.32	0.50 ± 0.43	73.67 ± 1.15	50.33 ± 5.86	69.33 ± 1.15
Interactive SFT	55.62 ± 0.30	11.22 ± 0.51	61.45 ± 1.13	45.77 ± 2.83	72.00 ± 2.65	44.33 ± 3.21	69.00 ± 0.00
Corrective SFT	56.57 ± 1.41	10.17 ± 0.24	61.59 ± 3.89	29.85 ± 1.06	73.50 ± 3.54	**59.00** ± 1.41	69.00 ± 2.83
Self-Imit. SFT	54.21 ± 0.30	11.00 ± 0.94	**62.35** ± 0.29	44.40 ± 3.69	73.00 ± 1.41	48.50 ± 6.36	**71.00** ± 0.00
RL Only	53.14 ± 0.20	10.67 ± 0.94	46.61 ± 2.37	35.82 ± 3.17	72.50 ± 2.12	46.00 ± 1.41	66.50 ± 2.12
SFT + RL	57.57 ± 0.81	**11.50** ± 1.18	58.97 ± 3.33	**49.25** ± 2.11	73.50 ± 0.71	51.00 ± 1.41	69.00 ± 1.41
Llama-3.1-8B-Instruct							
Base	42.57	10.00	35.61	41.04	81.00	**65.00**	76.00
Plan SFT	43.07 ± 1.32	**18.84** ± 0.23	**58.91** ± 0.90	8.96 ± 2.11	83.00 ± 0.00	60.50 ± 0.71	70.50 ± 0.71
Interactive SFT	40.76 ± 0.36	18.00 ± 0.67	57.16 ± 1.90	41.79 ± 3.95	84.00 ± 1.73	63.33 ± 1.53	73.67 ± 2.08
Corrective SFT	24.43 ± 6.64	16.56 ± 0.51	58.18 ± 0.67	37.07 ± 6.69	83.00 ± 1.00	62.30 ± 1.20	71.00 ± 2.60
Self-Imit. SFT	38.50 ± 2.32	16.33 ± 0.47	53.59 ± 2.65	**48.51** ± 3.17	82.50 ± 0.71	62.00 ± 0.00	70.50 ± 3.54
RL Only	**46.43** ± 2.02	12.84 ± 0.23	40.08 ± 2.45	41.04 ± 0.00	82.50 ± 0.70	64.00 ± 1.40	**77.50** ± 2.10
SFT + RL	42.00 ± 0.40	17.67 ± 0.00	56.57 ± 0.07	45.52 ± 2.11	**85.50** ± 0.71	62.50 ± 0.71	75.00 ± 1.41

Table 9: External planning transfer and general-capability results (%).

E Additional Diagnostic Results

E.1 PlanWorlds Execution Diagnostics

We complement task success with two measures of interactive execution: invalid-action rate and reference-relative step difference. Invalid-action rate captures failures in command parsing or action applicability across all episodes. Reference-relative step difference is the mean of $T - T_{\mathrm{ref}}$ over successful episodes, where T is the number of interaction steps and T_{ref} is the stored reference-plan length. Because references need not be optimal, this difference can be negative. It should be interpreted alongside task success, since the set of successful problems varies across conditions.

Table 10 shows that Plan SFT produces invalid actions on more than 93% of interaction steps across all four settings and both model families, consistent with its limited transfer to the interactive interface. Interactive SFT and its continuations largely avoid invalid actions in-distribution, but invalid-action rates increase on more challenging problems and the unseen domain. Thus, reliable execution within the training distribution does not ensure valid action selection under planning shifts.

	Invalid-action rate (%) ↓				Reference-relative steps ↓			
Condition	**In-Dist.**	**Theme**	**Difficulty**	**Domain**	**In-Dist.**	**Theme**	**Difficulty**	**Domain**
Qwen2.5-7B-Instruct								
Base	48.6	62.1	71.9	82.3	6.81	3.38	2.00	8.15
Plan SFT	98.3 ± 0.6	93.4 ± 1.8	95.3 ± 3.0	96.8 ± 0.4	—	8.50 ± 0.71	—	—
Interactive SFT	1.4 ± 0.3	8.5 ± 0.4	23.7 ± 3.6	59.6 ± 3.1	0.07 ± 0.08	0.64 ± 0.07	3.82 ± 0.46	5.04 ± 0.59
Corrective SFT	0.7 ± 1.0	7.6 ± 1.7	22.1 ± 0.7	58.0 ± 0.1	0.09 ± 0.13	0.73 ± 0.14	2.73 ± 0.85	4.91 ± 0.51
Self-Imit. SFT	0.8 ± 0.8	10.8 ± 0.8	22.3 ± 1.0	59.7 ± 5.2	0.14 ± 0.08	0.65 ± 0.15	3.04 ± 0.91	7.55 ± 0.95
RL Only	22.9 ± 5.4	28.7 ± 3.0	47.8 ± 7.0	52.2 ± 6.0	3.98 ± 0.18	3.63 ± 0.71	15.00†	6.54 ± 2.88
SFT + RL	1.6 ± 1.5	4.4 ± 0.0	18.0 ± 0.3	47.3 ± 1.2	1.23 ± 0.44	0.77 ± 0.39	4.56 ± 1.99	5.89 ± 1.57
Llama-3.1-8B-Instruct								
Base	58.6	45.3	60.3	82.3	8.90	4.95	3.00	10.80
Plan SFT	97.7 ± 1.1	97.4 ± 1.0	93.1 ± 4.2	98.8 ± 0.0	20.0	4.0	—	—
Interactive SFT	0.3 ± 0.6	2.1 ± 0.4	14.1 ± 1.2	43.9 ± 1.6	0.46 ± 0.03	0.26 ± 0.06	1.00 ± 0.93	7.18 ± 0.44
Corrective SFT	0.7 ± 0.9	0.7 ± 1.0	10.0 ± 2.3	51.1 ± 0.2	0.22 ± 0.31	0.47 ± 0.49	0.53 ± 0.12	5.08
Self-Imit. SFT	0.1 ± 0.1	2.0 ± 2.0	11.0 ± 0.7	43.5 ± 0.2	0.55 ± 0.02	0.58 ± 0.25	2.18 ± 0.58	7.42 ± 1.36
RL Only	17.5 ± 0.3	17.8 ± 2.1	26.7 ± 2.1	35.8 ± 2.6	6.00 ± 0.04	3.51 ± 0.06	6.58 ± 1.77	12.16 ± 0.22
SFT + RL	0.0 ± 0.1	1.9 ± 0.7	11.5 ± 2.2	41.3 ± 0.6	0.49 ± 0.01	0.17 ± 0.01	1.62 ± 0.54	6.85 ± 0.84

Table 10: Interactive execution diagnostics on PlanWorlds. Invalid-action rates include all episodes; reference-relative step differences include successful episodes only and are measured in steps. Standard deviations are omitted when only one seed-level estimate is available. Dashes indicate unavailable estimates.

E.2 ACPBench: Unconstrained Answer Elicitation

We extend the main-text interface analysis with an unconstrained answer-elicitation control on Llama-3.1-8B-Instruct. Both *Reask* and *Forced* retain the original response and append the same request for an explicit answer; only Forced constrains decoding to option letters. The two protocols

are evaluated on the same instances and original responses for each training seed. As shown in Table 11, their accuracies differ by at most 0.3 percentage points across conditions. Explicit answer elicitation thus achieves similar performance with or without the decoding constraint. However, the additional turn allows answer revision, so this comparison does not isolate answer formatting from reasoning.

Condition	Raw (%)	Reask (%)	Forced (%)
Base	42.6	48.6	48.9
Interactive SFT	40.8	45.4	45.3
Corrective SFT	24.4	48.4	48.7
Self-Imit. SFT	38.5	44.4	44.3
RL Only	46.4	51.6	51.4
SFT + RL	42.0	46.4	46.4

Table 11: ACPBench-MCQ accuracy under the standard and explicit-answer elicitation protocols.

E.3 ALFWorld: Performance Across Task Types

Table 12 breaks down free-form ALFWorld success for Qwen2.5-7B-Instruct across six task types. SFT + RL achieves higher success than Corrective SFT in every category, indicating that their aggregate gap spans multiple task structures.

Condition	Seeds	Pick & place $n = 24$	Pick two & place $n = 17$	Examine in light $n = 18$	Heat & place $n = 23$	Cool & place $n = 21$	Clean & place $n = 31$
Base	—	62.5	17.6	11.1	26.1	14.3	22.6
Interactive SFT	3	70.8	17.6	27.8	53.6	46.0	46.2
Corrective SFT	2	52.1	2.9	25.0	39.1	19.0	30.6
Self-Imit. SFT	2	75.0	20.6	41.7	39.1	40.5	41.9
SFT + RL	2	75.0	29.4	44.4	56.5	38.1	45.2

Table 12: ALFWorld success (%) by task type for Qwen2.5-7B-Instruct under the free-form protocol. Scores are means over the indicated numbers of training seeds; n denotes tasks per category. The base model is evaluated once.

F AI Use Statement

Generative AI tools were used for language editing and polishing of the manuscript, as well as for code editing and debugging. All research ideas, methodology, experimental design, analyses, and conclusions were developed and verified by the authors, who take full responsibility for the final content of the paper.