ALIENBODY: DIAGNOSING HOW LLM AGENTS ACQUIRE AND USE ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent benchmarks typically expose meaningful action names, conflating *name retrieval* with *action understanding*. We introduce *AlienBody*, a 900-instance benchmark that randomizes action mappings across six action families to separate exploration, interpretation, and planning. Controlled naming interventions reveal a model-specific prior: misleading labels lower GPT-4o's success below anonymous controls on two families while truthful labels raise it directionally, so named success need not reflect identification from interaction. Even when the mapping reaches the agent as effect names, the tested in-context agents achieve only 0–10% success on Relational environments, while symbolic breadth-first search that computes with the mapping achieves 100%. Direct probes on the two tested models show why: defining the operators behind the names lifts single-draw per-step predictions from at or below blind-guess accuracy to at most 52.6%, so a description of the actions does not by itself substitute for computing with them. A frozen interface study under one matched protocol isolates how that knowledge is used: on Relational grids, a programmable interface improves success over a conversational transition oracle by 49.75 points for DeepSeek V4 Flash and 35.75 for Qwen3.5-35B-A3B. Boolean workflows reverse that ranking for one model, and inspecting successful programs identifies fixed plans merely submitted through code. When the mapping is unknown, schema induction with verified search and active identification provides a constructive route from observations to control. These interventions make acquisition and use separately measurable, exposing the central shortfall: action information, whether a familiar label, a correct description, or a supplied mapping, can be available without being usable for control.

1 INTRODUCTION

An agent clicks a button labeled "Search" and succeeds. Is that because it *understands what searching does*, or because it *recognizes the word* from pretraining? These are different capabilities: *action understanding* builds a causal model of what an action does; *name retrieval* matches a label to pretrained vocabulary. Named success alone does not tell which one drove the behavior. Nor does knowing what a button does establish that the agent can compose its effects into a successful plan.

This distinction matters for evaluating large language model (LLM) agents. Web and desktop benchmarks expose familiar controls: click, type, and scroll (Zhou et al., 2024b; Xie et al., 2024), while game environments provide recognizable action vocabularies (Hafner, 2022; Paglieri et al., 2025). Such labels invite *name retrieval*, but task scores do not isolate it from *action understanding*. PlanBench tests planning from supplied domain models, with naming-sensitivity follow-ups (Valmeekam et al., 2023a;b); ARC-AGI-3 evaluates interaction through anonymous controls (ARC Prize Foundation, 2026). Until *acquisition* and *use* can be measured apart, success on familiar interfaces remains ambiguous: it does not establish that an agent identified what its actions do, nor that such knowledge can drive control.

We introduce *AlienBody*: an unfamiliar action "body" whose controls an agent must identify through interaction. The setting captures a practical problem shared by undocumented APIs and unfamiliar control panels: an agent must establish what the available operations mean before it can reliably compose them. Across 900 instances in six action families, button-to-effect mappings vary between episodes. A goal-hidden exploration phase precedes goal-directed execution, separating opportunities

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

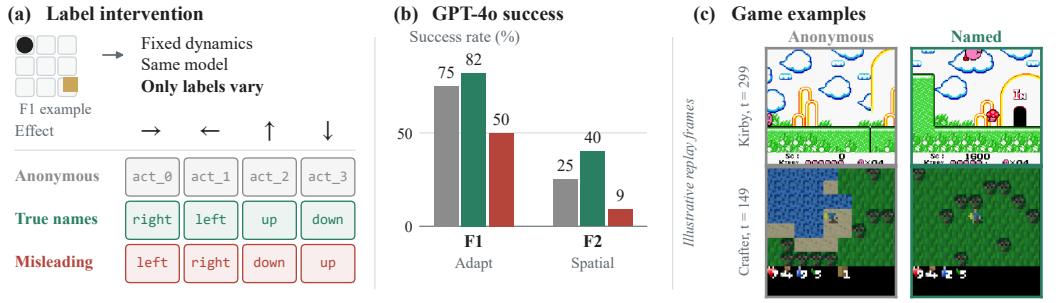


Figure 1: Action names change measured success. (a) An Adapt (F1) example: dynamics stay fixed while labels are anonymous, truthful, or misleading. (b) GPT-4o success on Adapt (F1) and Spatial (F2), pooled over two runs on the same 50 environments per family. (c) Anonymous and named game frames illustrate behavior; they are not paired trajectories. Quantitative game results: Table 2.

to learn the dynamics from the task of using them. Controlled interventions then vary the labels, supplied action information, and execution interface to examine acquisition and use separately.

Learning an unfamiliar action space has a long history in motor babbling and robotic self-modeling (Mahoor et al., 2018; Chen et al., 2021; Hoffmann, 2021). Language agents now build executable world models from feedback (Tang et al., 2024) and check plans with external verifiers (Kambhampati et al., 2024); the same generate–execute–keep pattern appears when agents revise their own code or exploration policies (Zhang et al., 2025b; Zheng et al., 2026). AlienBody asks how interaction evidence becomes a usable action model for control, without treating familiar labels or external scaffolding as a substitute for identification.

Acquiring action knowledge. Action labels change performance even when dynamics stay fixed (Figure 1). Misleading names reduce GPT-4o's success by 25 percentage points (pp) on Adapt and 16 on Spatial, whereas truthful names shift the point estimates up. Effects depend on the model and setting: game evaluations show the benefit of named controls and a reversal on *Crafter*. Naming is therefore part of the evaluation condition, not evidence of *action understanding*: scores under meaningful names are equally consistent with *name retrieval*.

Using action knowledge. With ground-truth mappings supplied as effect names, tested in-context agents achieve only 0–10% success on Relational tasks, where breadth-first search (BFS) using executable dynamics reaches 100%. Supplying an effect name is not the same as specifying its operator. In single-step probes, complete definitions substantially improve transition prediction for both tested models; in end-to-end control, however, DeepSeek V4 Flash improves from 5.0% to 12.5%, while no significant improvement is detected for Qwen3.5-35B-A3B (Section 4). A correct description can be available without being usable for control.

How the agent acts on that information depends on the interface. Across 400 paired instances per task, we compare a persistent draft, a conversational transition oracle, and a programmable code-plus-oracle interface. On Relational grids, switching from the conversational oracle to the programmable interface raises success from 5.25% to 55.00% for Flash and from 1.50% to 37.25% for Qwen. Boolean workflows reverse the conversational-versus-programmable ranking for Flash. Inspecting accepted programs further shows that many successful workflow submissions simply contain a fixed plan: success through code does not establish that execution computed the plan (Section 4.4).

When mappings are unknown, verified induction within a known operator vocabulary and active probes connect observations to executable models and planning (Section 5). Active probes target hypotheses that fit past observations but disagree on untried transitions. Together, these studies connect acquisition to use while separating successful control from *name retrieval*, stated knowledge, and the mere availability of code.

Our contributions are: (i) a **benchmark** for action-model identification: 900 instances, six action families, anonymous controls, and interventions that separate exploration, interpretation, and planning; (ii) controlled evidence on **acquisition** and **use**: naming, rule-definition, and interface interventions show that a familiar label, a correct description, or a supplied mapping can be available without being usable for control; and (iii) a **constructive** identification–planning study: verified induction and active hypothesis testing connect interaction evidence to executable action models. Code and environments: https://github.com/AnonymousPubCode/ICLR27-AlienBody.

2 Related Work

Named Interfaces dominate agent benchmarks, so measured success mixes name retrieval with the dynamics the agent identified. Web, desktop, and code agents expose documented controls Zhou et al. (2024b); Koh et al. (2024); Xie et al. (2024); Liu et al. (2024); Jimenez et al. (2024), and game suites likewise provide recognizable action vocabularies Wu et al. (2024); Zhang et al. (2025a); Paglieri et al. (2025); Hafner (2022). ARC-AGI-3 ARC Prize Foundation (2026) evaluates interaction through anonymous controls instead. AlienBody starts from anonymous controls, but adds naming, calibration, and diagnostics that separate acquisition from use (Appendix Table 4).

Interaction-Learned Models capture action effects from experience, yet stronger next-state prediction does not by itself establish usable control. Model-based agents learn dynamics from interaction Hafner et al. (2023); WorldLLM Levy et al. (2025) revises linguistic hypotheses via curiosity-driven inference, while WebEvolver Fang et al. (2025) and WorldEvolver Zhang et al. (2026) co-evolve or revise world models alongside the agent. Evaluations are shifting from single-step state match to behavior-level criteria, where one-step accuracy need not track control Huang et al. (2026); Lambert et al. (2020); Li et al. (2026). Motor babbling and body-schema learning offer a related calibration view (Mahoor et al., 2018; Thelen and Smith, 1994; Wolpert et al., 1995; Krakauer et al., 2019; Chen et al., 2021; Hoffmann, 2021); Appendix B elaborates. AlienBody puts action priors between identification and planning: controlled naming and assistance show when that knowledge supports control, and probes expose what passive observation cannot.

Supplied Action Models let language agents plan without first identifying what each control does. ReAct-style loops interleave reasoning, feedback, memory, and actions Yao et al. (2022); Shinn et al. (2023); Wang et al. (2023); Huang et al. (2022), and lookahead methods search over those loops Yao et al. (2023); Zhou et al. (2024a); Hao et al. (2023); Sun et al. (2023). PlanBench Valmeekam et al. (2023a) tests planning from supplied domain models, with naming-sensitivity follow-ups Valmeekam et al. (2023b; 2024); LLM+P Liu et al. (2023) pairs an LLM with a classical planner over the Planning Domain Definition Language (PDDL); PSALM-V Zhu et al. (2025) instead induces pre- and post-conditions from interaction. Tool choice shows the same name sensitivity: adversarially chosen names bias selection Sneh et al. (2025), and schema-misalignment errors shrink when schemas are renamed toward pretraining conventions Lee et al. (2025). AlienBody reverses the usual order: agents infer effects from interaction before an exact-mapping condition tests control from supplied names.

Executable Search keeps revisions only when execution can validate them. Code-as-Policies and Voyager Liang et al. (2022); Wang et al. (2024) emit reusable controllers; program synthesis and execution-guided search Nijkamp et al. (2023); Romera-Paredes et al. (2024) treat hypotheses as runnable, and HYSYNTH Barke et al. (2024) learns grammars for symbolic search. WorldCoder Tang et al. (2024) revises a code world model through interaction, and LLM-Modulo Kambhampati et al. (2024) argues for tight coupling with external verifiers. Recursive self-improvement applies the same generate–execute–keep loop to the agent or its exploration policy Zhang et al. (2025b); Zheng et al. (2026); ARC-AGI-3 agents synthesize executable world models online under execution and verification ablations Courtis et al. (2026); Rodionov (2026a;b). AlienBody does not evaluate self-improvement itself; under a controlled action prior and limited oracle access, it asks whether interaction evidence yields models usable for control, via search over a known transition model and schemas checked against observed transitions.

3 The AlienBody Benchmark

Let an environment have state space $\mathcal{S}$, anonymous actions $\mathcal{A} = \{a_0, \ldots, a_{n-1}\}$, and a transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ whose action mapping is fixed within an episode and randomized across instances. State includes position, orientation, color, and the action-history variables needed for temporal effects. The agent receives rendered observations o_t and action feedback, rather than the simulator's transition rules; evaluations use text or image-plus-text observations (Appendix J). Success means reaching a target position g within the action budget.

In *Phase 1 (Calibration)*, the target is hidden and no task reward is given. The agent explores through observation–action traces and can declare calibration complete; otherwise the phase ends at its exploration cap. In *Phase 2 (Execution)*, the target is revealed and control continues from the state reached during calibration, without a reset. Both phases consume the total action budget, so early

completion of calibration leaves more actions for execution. Agents use the resulting interaction history, or an explicitly constructed model $\hat{T}$, to guide execution.

3.1 Families and Difficulty Tiers

Six **F**amilies (F1–F6) vary which state features determine action effects under T. *Adapt* (F1) permutes the cardinal directions, testing identification of a fixed movement mapping. *Spatial* (F2) defines egocentric movement relative to facing, requiring orientation tracking. *Conditional* (F3) couples actions to the agent's color state: the color is observable, while its relationship to action effects must be learned. *Relational* (F4) makes effects depend on spatial relations among grid cells, such as moving toward the nearest differently colored cell. *Compositional* (F5) combines primitives within a button, requiring their combined effect to be predicted. *Temporal* (F6) makes effects depend on preceding actions or movement. Full definitions appear in Appendix G.

The generator supports four tiers that vary grid size, action count, obstacles, and exploration budget (Appendix D). The standard Tier-M setting uses four actions, a 16×16 grid, up to 20 calibration actions, and 50 total actions. The base release contains 900 BFS-validated instances across six families, split 300/120/300/180 into train/dev/test/secret sets. Targeted interventions use separately identified frozen evaluation sets; their sizes and splits are stated with the corresponding experiments. Relational Tier-L/XL test identification and planning with six/twelve actions (Appendix L).

3.2 Diagnostic Interventions and Metrics

The *Diagnostic Interventions* probe exploration, interpretation, and planning by supplying different **L**evels of assistance. L0 is unaided anonymous control. L1 supplies a familiarity cue (scrambled directions in the Adapt control), testing the effect of an action-type prior. L2 replaces free exploration with systematic action tests and supplies a structured observation summary for execution. L3 supplies the action–effect mapping—each action id with the *name* of its effect, as in "Action 0 = move_toward_brightest", but not an executable T or a definition of the operator behind that name—and starts execution without exploration; symbolic BFS, which plans with the true T, provides a reference. The comparisons include changes to exploration and the execution starting state. L0 provides the main evaluation; L1 and L3 are targeted diagnostics on the reported model–task subsets (Appendix H). L2 is specified but not evaluated here, so no L2 result is claimed: the archived runs cover two models only, and the summary artifact they injected was not preserved (Appendix H.3).

Exploration coverage (EC) measures the fraction of actions tried; P1 is Phase-1 length. We report three diagnostics: *calibration-accuracy (CA) proxy* (Table 1) applies fixed rules to each action's first observed transition; *stated-effect accuracy* scores the model's answers to effect questions, with prior/post-interaction comparisons; and *held-out prediction accuracy* tests archived forward models on unseen state–action contexts. Stated-effect and held-out probes cover the subsets in Appendix H. We report success rate (SR); action counts enter the efficiency indices in Appendix N.

3.3 The Workflow Suite

Section 4.4 adds a second instantiation of $(\mathcal{S}, \mathcal{A}, T)$: a non-spatial task with explicit tool schemas. Its state is $k{=}8$ boolean device flags rendered as a bit vector; an action ("tool") carries masked preconditions and a simultaneous set/clear assignment, fires only when its precondition flags hold, and consumes a step even when blocked. The agent sees the current flag vector, the goal's partial assignment, and the tool schemas, but a blocked call is indistinguishable from a call that fired and left the state unchanged. Instances ask for a plan reaching a target flag configuration within 24 actions over 16 tools; the 400-instance frozen set comprises 120/160/120 cases with shortest plans of 2–3/4–6/7–10 steps in two pre-registered halves, and BFS solves all instances.

4 Experiments

We ask diagnostic research questions: **(RQ1)** how do action names and observation modalities affect control vs. anonymous baselines? **(RQ2)** what planning gap remains when action effects are supplied, and how does it change across interfaces? **(RQ3)** when effects are unknown, can verified identification produce models that support control? We address the first two here, the third in Section 5.

Unless noted, we evaluate on the Tier-M test split: 300 environments with a 20-step calibration cap and a 50-step total budget (Appendix H.5), all BFS-validated (Section 3). We use the diagnostic interventions of Section 3 as operational diagnostics for Explore, Interpret, and Plan: L0 is the main anonymous setting, L1 and L3 are targeted probes, and L2 is specified but not evaluated (Appendix H.3). We compare heuristic baselines (Random, Systematic, Bayesian), Qwen3.5 across sizes and modalities, and five further models (Table 1). Oracle+BFS supplies a known-dynamics reference; Forward-Model Babbling (FMB) variants appear in Section 5. We report success rate (SR) together with calibration efficiency (CE/$\mathrm{CE}_{\mathrm{cond}}$), the trajectory-only CA proxy, exploration coverage (EC), and Phase 1 length (P1; definitions in Section 3 and Appendix H).

4.1 Main Results: Anonymous Control Across Models (RQ1)

Table 1: Main results on AlienBody Tier-M. Six families, 300 environments; bars: SR (0–100%), bold: best non-oracle CE per block. DeepSeek N=299 (Appendix C); CA is a trajectory-only proxy; image+text includes ASCII (Appendix J); API models use text (thinking off) except GPT-4o and Gemini 3.1 Pro (image+text); CE denominators in Appendix H.

		Success	**Efficiency**		**Exploration diagnostics**		
Evaluation	Method / model	SR (%)	CE	$\mathrm{CE}_{\mathrm{cond}}$	CA	EC	P1
Baselines Heuristics	Random	4.667	0.022	0.464	0.792	0.996	20.000
	Systematic	17.667	0.187	1.056	0.776	1.000	4.000
	Bayesian	22.667	**0.238**	1.050	0.782	1.000	4.103
Modality Image + text	Qwen3.5-4B	8.333	**0.098**	1.176	0.184	0.238	0.950
	Qwen3.5-9B	7.667	0.062	0.803	0.763	0.952	8.940
	Qwen3.5-35B	11.000	0.078	0.706	0.613	0.798	16.060
	Qwen3.5-397B	15.667	0.081	0.515	0.644	0.848	16.640
Modality Text only	Qwen3.5-4B	3.300	0.018	0.536	0.300	0.330	11.300
	Qwen3.5-9B	1.700	0.035	2.059	0.070	0.070	2.700
	Qwen3.5-35B	20.000	0.117	0.585	0.000	0.000	0.000
	Qwen3.5-397B	44.000	**0.291**	0.661	0.150	0.200	3.500
Cross-model Mixed input	GPT-4o	33.000	0.257	0.779	0.688	0.892	5.987
	Gemini 3.1 Pro	67.333	**0.591**	0.878	0.728	0.948	3.880
	DeepSeek V4 Pro	35.117	0.245	0.697	0.786	0.992	9.224
	GLM-5.2	66.333	0.490	0.739	0.794	0.998	12.560
	Kimi K3	50.667	0.365	0.720	0.789	0.998	9.267
Reference	*Oracle+BFS*	100.000	*1.187*	*1.187*	*0.776*	*1.000*	*4.000*

The model with the highest SR (Gemini 3.1 Pro: SR 67.3%, CE 0.591) still sits far below the oracle (SR 100%), and success collapses on the harder families. Exploration strategy is one contributor: a Bayesian baseline over permutation matrices (SR 22.7%) and a Systematic one-test-per-action heuristic (17.7%) beat every open-weight vision-language model (VLM) under image input (best 15.7%), while image VLMs waste Phase 1 re-testing actions (P1=16.6 at 397B vs. $\sim$4 for heuristic baselines); as the model-output-based replacement for the trajectory-only CA proxy, held-out next-state prediction reaches 0.71–0.89 against 0.05–0.08 for controls (Appendix H). Under image input (which also carries the ASCII rendering; Appendix J), scale barely moves SR (8.3% at 4B to 15.7% at 397B) and CE stays $\leq$0.10; the same Qwen stack with text-only ASCII grids (no image) flips the picture: 4B gets worse, but 397B jumps from 15.7% to 44.0% SR (2.8$\times$; Appendix J). Adding the image channel does not improve these larger models under the tested prompts. Pooled SR mixes easy/hard families. GPT-4o's family-clustered 95% confidence interval (CI) is $[11.0, 59.0]$ around 33.0% (per-family rates in Appendix C). Heuristic baselines solve mainly Adapt; success on Relational stays single-digit for all agents, even text-only API models (GLM-5.2 6%, Kimi K3 8%).

4.2 The Name Prior: Button Labels as a Causal Confound

We rerun Adapt and Spatial (50 envs each; GPT-4o and Gemini 3.1 Pro, image+text) under three labelings: anonymous (AlienBody default), named-true (labels match effects), and named-misleading (every label wrong). A familiarity-only control helps only modestly (Appendix L).

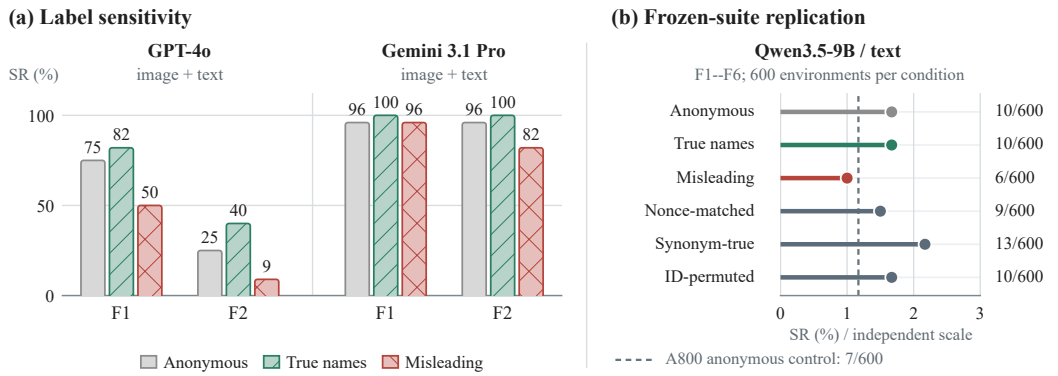


Figure 2: Effects of action naming across models and settings. (a) Same 50 AlienBody environments per family under three labelings: anonymous (default; the comparison baseline), named-true, and named-misleading; true−misleading differences: GPT-4o +32/+31pp, Gemini +4/+18pp (F1/F2). (b) Six-condition controlled replication at Qwen3.5-9B (text, 600 frozen environments per condition, own axis; each condition contrasted with anonymous on the same environments). Dots: SR; labels: solved/600; dashed line: an anonymous re-run on a second host, a control for the anonymous condition rather than a further condition.

The effect is large and model-specific, and it runs in both directions (Figure 2(a); paired bootstrap CIs; environment-level exact two-sided sign tests count each environment once, pooled-counting values in Appendix L.1). On Spatial, true labels raise GPT-4o's SR from 25% to 40% (+15pp, 95% CI $[+1, +29]$; p=0.071, a direction rather than an established effect) while misleading labels drop it to 9% (−16pp, $[-26, -6]$, p=0.011). On Adapt, where anonymous GPT-4o already reaches 75%, misleading labels still cut SR to 50% (−25pp, $[-38, -12]$, p=0.0014): misleading names affect control despite access to interaction feedback. Gemini 3.1 Pro is nearly immune to misleading labels (Adapt: 96%→96%) yet still exploits true ones.

The effect varies across the tested models and settings. On the frozen 600-environment suite, a six-condition controlled replication with Qwen3.5-9B (text, no mapping aid) adds *nonce-matched*, *synonym-true*, and *id-permuted* label controls to the three labelings: every condition lands at the benchmark floor for this model (pooled SR 1.0–2.2%) and every contrast is null, the largest being +0.50pp over anonymous (p=0.678; Figure 2(b); full condition list and a cross-host control in Appendix L.1). The design is floor-bound rather than an equivalence test, since at 1–2% SR a drop has little visible room (the floor truncates the downward direction) while the observed upward differences are also small. It does locate where the name prior stops being measurable: for a model with no room to move, label manipulations leave no trace.

The same confound appears outside the grid, but not always in the same direction. On *Kirby's Dream Land* (Game Boy via PyBoy; *qualitative*, three runs per condition, below the preregistered paired target), GPT-4o plays 300 visual steps under the three labelings (Table 2, left): named buttons (whose descriptions are supplied) reach 8.7 checkpoints (scroll 20.7), category hints (scrambled labels) yield partial progress (scroll 8.7) with more damage, and anonymous IDs leave every run at spawn (max scroll 1.0). On Crafter Hafner (2022), already flagged for leaking action names in Appendix Table 4, we removed the glosses and re-ran the contrast under a fully shared protocol (17 actions, 20 paired seeds, 150-step cap; Table 2, right). The v1 ordering inverts: true names yield 0.10 mean achievements against 1.85 (anonymous) and 2.45 (category rank labels); named−anonymous −1.75, bootstrap 95% CI $[-2.25, -1.30]$, exact Wilcoxon $p=9.7\times10^{-5}$; category vs. anonymous is within noise (+0.60, p=0.09). The post-episode probe reads named 0.79 > category 0.36 > anonymous 0.12, and named runs try only 4.3 of the 17 actions against 13.3 for anonymous (Table 2, right; per-action mix in Appendix L). The earlier v1 advantage, from a prompt that supplied effect descriptions and gave no condition history, is not reproduced here (Appendix L). Thus, higher stated-effect accuracy coexists with narrower action coverage and lower task progress. This is consistent with label-dependent exploration but not established here.

4.3 The Planning Wall (RQ2)

At L3, where the ground-truth mapping is supplied as effect names, success on Relational environments stays low (Figure 3): 0–10% SR across GPT-4o, Gemini 3.1 Pro, DeepSeek V4 Pro, and

Table 2: Name prior in two games (GPT-4o, visual). Kirby: 300 steps per run, full level 36 scroll units; †checkpoints at spawn. Crafter: 150-step cap, 20 paired seeds. Probe CA is post-episode stated-effect accuracy (0–1); actions tried are mean distinct actions of 17. Frames: Appendix Figure 6.

	Kirby (3 runs / condition)			**Crafter** (20 paired seeds)		
Condition	Max scroll	Checkpoints	Deaths/run	Achievements	Actions tried	Probe CA
Named	**20.7**	**8.7**	0.3	0.10	4.3	0.79
Category hints	8.7	6.0	5.7	**2.45**	7.0	0.36
Anonymous	1.0	4†	0.0	1.85	**13.3**	0.12

(a) L3 success by family (%)

Agent / method	F1	F2	F3	F4	F5	F6
GPT-4o	100	60	72	0	66	26
Gemini 3.1 Pro	100	100	100	10	100	38
DeepSeek V4 Pro	92	43	66	0	28	8
Qwen3-VL-32B	86	44	82	0	20	16
GPT-5.1*	—	—	—	4	—	—
FM_{heur}	100	92	80	88	0	4
Oracle BFS	100	100	100	**100**	100	100

* F4-only probes; — not evaluated. Italics: search over a model.

(b) F4 contrast

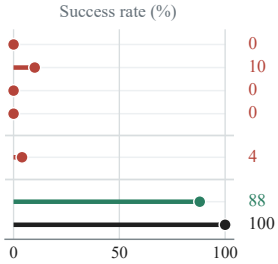


Figure 3: The planning wall at L3. (a) Success rates by family, with F4 highlighted. Rows align across panels: four in-context agents, one additional F4-only probe (*), and two methods that search over a model (italic). Dashes mark untested families in the F4-only probe. (b) F4 success on the same rows: 0–10% for the in-context agents and probes, compared with 88% for the forward-model heuristic (FM_{heur}) and 100% for Oracle BFS.

Qwen3-VL-32B, while BFS with the exact executable dynamics scores 100%. The wall survives Relational probes and reasoning models (Figure 3b): GPT-5.1 scores 4.0% (2/50) and Qwen3.5-4B (text and image) at 0/50; chain-of-thought and thinking-token configurations also fail (Appendix L.12).

In the L3 condition of the planning wall (Figure 3) the mapping reaches the agent as effect *names* inside the state, with no operator definitions, no oracle, and no episode history. Asked to predict a single transition on exactly that information, DeepSeek V4 Flash reaches 18.6% exact over 800 calls on 100 Relational environments (environment-clustered 95% CI $[16.0, 21.4]$), near the 20.9% of a blind guess among the four adjacent cells. Supplying complete prose definitions of the four operators and changing nothing else lifts the same predictions to **52.6%** ($[48.8, 56.4]$; paired $+34.0$ points, $[29.3, 38.8]$): a name is not an operator, and supplying the definition is what raises per-step accuracy. It does not carry the model the rest of the way, because even fully defined its predictions are wrong about half the time on a deterministic function (Appendix L.13). In a separate registered end-to-end study, supplying the full operator definitions raises DeepSeek V4 Flash's success from 5.0% to 12.5%, while for Qwen3.5-35B-A3B no significant change is detected (Appendix H.11); end-to-end success and per-step prediction are separate measurements.

The gap accompanies distinct behavioral patterns—high action entropy with 26% oracle-plan agreement for Gemini 3.1 Pro, degenerate repetition for a local Qwen model (Appendix L.11)—yet success collapses in both; to test whether it changes with the means of using action knowledge, we compare interfaces on fresh instances.

4.4 Using Action Knowledge: Interface Interventions

We evaluate DeepSeek V4 Flash and a locally served Qwen3.5-35B-A3B on two text-state suites: 400 fresh Relational instances and 400 Boolean workflows (Section 3.3). The three arms share the same public task information (Table 3). All arms share the task's real-action limit, and oracle arms share its query cap (1,024 on Relational grids; 4,096 on workflow). The programmable arm allows four program submissions, carrying previous code and errors across repairs; its implementation does not retain query results between submissions. Thus the comparison evaluates complete interfaces: the gap reflects programmatic search execution, higher realized simulation throughput, and error-tolerant submission formatting, not isolated planning ability. The conversational arms run under linear history scaffolding; tree-search agent architectures Zhou et al. (2024a); Hao et al. (2023) are not among the

Table 3: Using known action models through different interfaces. Success (%), 400 paired instances per task. Draft (J1): persistent draft; Oracle (J3L): draft + conversational oracle; Program (J4): code over the same oracle. Bold: best arm per model and task. Δ: paired Program−Oracle difference (pp), with 95% bootstrap CI; all four contrasts survive primary Holm correction. BFS (J5): 400/400 per task under the same oracle cap.

Task	Model	Draft	Oracle	Program	Δ [95% CI]
RELATIONAL (F4)	DeepSeek V4 Flash	6.00	5.25	**55.00**	$+49.75$ $[44.50, 55.00]$
	Qwen3.5-35B-A3B	2.75	1.50	**37.25**	$+35.75$ $[31.00, 40.75]$
BOOLEAN WORKFLOW	DeepSeek V4 Flash	**87.75**	76.25	55.00	-21.25 $[-28.00, -14.50]$
	Qwen3.5-35B-A3B	66.75	36.00	**86.75**	$+50.75$ $[45.25, 56.00]$

arms evaluated here. Full contracts, this protocol deviation, resource use, and complete arm results with A/B counts are in Appendix H.7.

The instances, prompts, parsers, budgets, and primary contrast were frozen before formal model evaluation. Each task has two fixed halves of 200 environments, with identical instances across arms and models. We report paired success differences with 10,000-draw bootstrap intervals and exact McNemar tests, applying Holm correction to the four primary programmable−oracle comparisons.

The Relational gap replicates across model families: the programmable interface improves over the conversational oracle by 49.75 points for Flash and 35.75 for Qwen (Table 3), with positive effects in both fixed halves. The plain draft arm also stays low (6.00% and 2.75%).

Workflow rankings depend on the model. Flash performs best with a draft and no oracle (87.75%); adding the conversational oracle yields 76.25%, and the code interface 55.00%. Qwen instead reaches 86.75% with the programmable interface, compared with 66.75% for the draft and 36.00% for the conversational oracle: the two models order the same three interfaces differently, so the programmable-interface advantage on Relational grids does not extend to a universal ranking. In the prespecified secondary comparisons, the oracle trails the draft on workflow for both models (-11.50 and -30.75 points), while on Relational grids the corresponding intervals include zero (Appendix H.7). Access to simulated transitions helps only where the interface turns it into control.

To see what success through code measures, we inspect the final accepted programs in every successful programmable-interface episode, using a post-hoc syntax audit of the stored responses. On Relational grids, all 220 successful Flash programs and 146 of 149 successful Qwen programs contain loops or comprehensions; all Flash successes and 145 Qwen successes use the external oracle somewhere in the episode. On workflow, however, 263 of Qwen's 347 successes end with only a constant `PLAN = [...]` assignment, and 346 use no external oracle. Those final programs submit an already specified plan. Other programs may compute with the supplied rules without calling the oracle, so zero oracle use alone does not imply an absence of search. Appendix Table 17 documents the audit: the interface's success rate and its computational role are distinct measurements.

The same gap persists with format retries. On a nested 100-environment subset of the registered set, both interfaces receive full operator definitions, a shared 256-query cap, and format retries at the interface seam. Success is 1% versus 91% for Flash and 1% versus 56% for Qwen (paired gaps of $+90$ and $+55$pp; Appendix H.8). The gap does not close with definitions supplied and rejections absorbed; execution-time repair and realized query throughput remain interface-dependent.

5 From Action Knowledge to Effective Control

The interface study supplies action knowledge and varies how agents use it; anonymous control instead requires acquiring it. We therefore evaluate an explicit identification–planning pipeline: induce a transition model from observations, test its predictions, and plan with it. Supplying a lookup primitive does not by itself close the gap: across one generation on the same relational split, interface-only arms stay at 0–4%, a conversational `next_state` tool at 10–26%, and a Python read-eval-print loop (REPL) at 90–100% (Appendix H.6; unmatched budgets, rematched in Section 4.4).

Absent L3, Phase 1 must induce an executable forward model $\hat{T}$. We call this plan-over-model pattern *Forward-Model Babbling* (FMB): babble to identify action schemas, then plan with BFS over a ForwardModelSimulator that applies typed effects—translate, rotate, relational, composite,

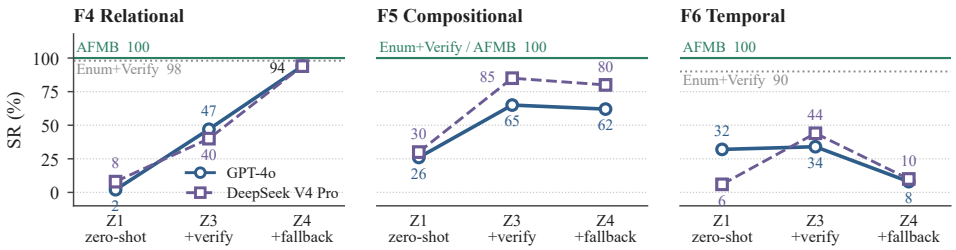


Figure 4: Verification-guided induction (FM_{ind}) on F4/F5/F6 (Tier-M, 50-env runs, SR%). No LLM loop reaches the search-only ceiling (paired CIs: Appendix L).

temporal—without calling the VLM (Appendix K). An executable $\hat{T}$ lets candidate transitions be checked repeatedly during search, so identification errors surface as failed plans rather than guessing.

As an existence check, FM_{heur} matches family-aware schemas from Phase-1 observations and plans with BFS. On Relational it enumerates the $4! = 24$ permutations of known relation types, scores each by replaying predicted effects against those observations, and never reads the ground-truth mapping (fairness audit: Appendix L.17). The two settings do not nest: FM_{heur} has operator semantics and a forward model but not the mapping, while L3 supplies the mapping without verified operator semantics. On Tier-M it reaches 88% SR on Relational, where no VLM exceeds 10% even with the mapping supplied as names, while failing on Compositional and Temporal (Appendix Table 23). Pooled SR (60.7%) remains below the best VLM-L3 (74.7%): FM_{heur} shows that Relational is solvable by plan-over-model given the relation vocabulary, not that the same heuristic is a general agent. Injecting the ground-truth schema into *the same* planner yields 100% on Relational and a content-permuted schema yields 2%, so search is not the bottleneck (Appendix L.14).

Within a public operator vocabulary, Enumerate+Verify (no LLM) scores candidates by exact trajectory replay: 98/100/90% on Relational, Compositional, and Temporal (Figure 4). At n=8 Relational it reaches 90%: remaining failures are perfect-score ties ($|H^\star|>1$), not verification misses. *Active* FMB (AFMB) selects disagreement probes that split $H^\star$, lifting the same split to **100%** and Temporal Tier-M from 90% to **100%**; ablations without information-gain selection fall back to 90–92% (Appendix L). At n=8, LLM proposers stay weak even with the same verifier (GPT-4o 2%; DeepSeek V4 Pro 6%/4%): verification helps when a strong search proposes candidates, but does not substitute for one. At n=12, an exact-assignment search still reaches 56% while FM_{ind} stays at 0% (Appendix L).

Training provides a complementary stress test. Across the valid archived low-rank adaptation (LoRA) and chain-of-thought (CoT) evaluations spanning 9B–32B, Relational success stays at the 2% no-model fallback: executed trajectories match that fallback action-for-action on 45–50 of 50 episodes, and usable schemas are rarely emitted (Appendix M). In these archived evaluations the bottleneck is emission, not planning—the same ground-truth injection yields 100% while a content-permuted schema returns 2%—and verification-filtered self-training does not internalize the search signal on Relational either (Appendix L).

6 Conclusion

Agent benchmarks that expose familiar controls leave a standing ambiguity: measured success may reflect name retrieval, stated knowledge of effects, or genuine control under an identified action model. AlienBody makes that decomposition operational. Holding dynamics fixed while varying labels shows that naming is part of the evaluation condition, not evidence of action understanding; supplying effect names—or even complete definitions—does not guarantee Relational control; and once a mapping is known, the interface that mediates planning, not merely access to transitions, determines whether knowledge becomes behavior. Programmability yields large Relational gains across model families, yet rankings reverse by model and some programs submit fixed plans.

When mappings are unknown, the same separation is constructive: verified induction and active disagreement probes turn interaction into executable models that plan. The claim is diagnostic for agent evaluation: until acquisition and use are measured apart, familiar interfaces will credit the wrong capability. Extending these interventions beyond discrete, bounded suites—richer observations, longer horizons, unknown-dynamics transfer—is the natural next stress test.

AI USE STATEMENT

The research idea, the study design, and the scientific claims in this paper are the authors' own; the authors directed the AI-assisted work. We used generative AI tools for the following tasks with required disclosure: feedback on experimental design, and assistance with code implementation and result analysis. We also used them for recommended-disclosure tasks: literature search, drafting and language editing, reference and figure preparation, and the release website. Benchmark instances were generated by seeded programs with no LLM invoked, and this paper contains no mathematical proofs. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

ARC Prize Foundation. Arc-agi-3: A new challenge for frontier agentic intelligence. *arXiv preprint arXiv:2603.24621*, 2026.

Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013. doi: 10.1016/j.robot.2012.05.008.

Shraddha Barke, Emmanuel Anaya Gonzalez, Saketh Ram Kasibatla, Taylor Berg-Kirkpatrick, and Nadia Polikarpova. HYSYNTH: Context-free LLM approximation for guiding program synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *NeurIPS*, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *ICLR*, 2019.

Boyuan Chen, Robert Kwiatkowski, Carl Vondrick, and Hod Lipson. Full-body visual self-modeling of robot morphologies. *arXiv preprint arXiv:2111.06389*, 2021.

David Courtis, Wenhao Li, and Scott Sanner. OPINE-World: Programmatic world modeling with ontology-error-prioritized interactive exploration for ARC-AGI-3. *arXiv preprint arXiv:2607.01531*, 2026. URL https://arxiv.org/abs/2607.01531.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. *Nature*, 590:580–586, 2021.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. WebEvolver: Enhancing web agent self-improvement with coevolving world model. *arXiv preprint arXiv:2504.21024*, 2025. URL https://arxiv.org/abs/2504.21024.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *EMNLP*, 2023.

Matej Hoffmann. Body models in humans, animals, and robots: mechanisms and plasticity. In *Body Schema and Body Image: New Directions*, pages 152–180. Oxford University Press, 2021. doi: 10.1093/oso/9780198851721.003.0010.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Youling Huang, Guanqiao Chen, Junchi Yao, Lu Wang, Fangkai Yang, Chao Du, ChenZhuo Zhao, Pu Zhao, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Beyond state consistency: Behavior consistency in text-based world models. *arXiv preprint arXiv:2604.13824*, 2026. URL https://arxiv.org/abs/2604.13824.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *ICLR*, 2024.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. LLMs can't plan, but can help planning in LLM-Modulo frameworks. *arXiv preprint arXiv:2402.01817*, 2024. URL https://arxiv.org/abs/2402.01817.

Kento Kawaharazuka, Kei Okada, and Masayuki Inaba. Gemuco: Generalized multisensory correlational model for body schema learning. *arXiv preprint arXiv:2409.06427*, 2024.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *ACL*, 2024.

John W. Krakauer, Alkis M. Hadjiosif, Jing Xu, Aaron L. Wong, and Adrian M. Haith. Motor learning. *Comprehensive Physiology*, 9(2):613–663, 2019.

Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022. doi: 10.1016/j.inffus.2022.03.003.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020. URL https://arxiv.org/abs/2002.04523.

Jonggeun Lee, Woojung Song, Jongwook Han, Haesung Pyun, and Yohan Jo. Don't adapt small language models for tools; adapt tool schemas to the models. *arXiv preprint arXiv:2510.07248*, 2025. URL https://arxiv.org/abs/2510.07248.

Guillaume Levy, Cedric Colas, Pierre-Yves Oudeyer, Thomas Carta, and Clement Romac. WorldLLM: Improving llms' world modeling using curiosity-driven theory-making. *arXiv preprint arXiv:2506.06725*, 2025.

Yixia Li, Hongru Wang, Peng Lai, Zhiwen Ruan, He Zhu, Youxin Zhu, Ganlong Zhao, Minda Hu, Yun Chen, Sibei Yang, Peng Li, Jeff Z. Pan, Jia Pan, Guanhua Chen, Yang Liu, and Guanbin Li. Bridging the agent-world gap: Text world models for LLM-based agents. *arXiv preprint arXiv:2606.09032*, 2026. URL https://arxiv.org/abs/2606.09032.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

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.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In *ICLR*, 2024.

Zahra Mahoor, Bruce MacLennan, and Allen McBride. Neurally plausible model of robot reaching inspired by infant motor babbling. *arXiv preprint arXiv:1801.00293*, 2018.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *ICLR*, 2023.

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

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Sergey Rodionov. Do coding agents need executable world models, simplification, and verification to solve ARC-AGI-3? *arXiv preprint arXiv:2607.15439*, 2026a. URL https://arxiv.org/abs/2607.15439.

Sergey Rodionov. Executable world models for ARC-AGI-3 in the era of coding agents. *arXiv preprint arXiv:2605.05138*, 2026b. URL https://arxiv.org/abs/2605.05138.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6.

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

Jonathan Sneh, Ruomei Yan, Jialin Yu, Philip Torr, Yarin Gal, Sunando Sengupta, Eric Sommerlade, Alasdair Paren, and Adel Bibi. ToolTweak: An attack on tool selection in LLM-based agents. *arXiv preprint arXiv:2510.02554*, 2025. URL https://arxiv.org/abs/2510.02554.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. In *NeurIPS*, 2023.

Hao Tang, Darren Key, and Kevin Ellis. WorldCoder, a model-based LLM agent: Building world models by writing code and interacting with the environment. *arXiv preprint arXiv:2402.12275*, 2024. URL https://arxiv.org/abs/2402.12275.

Esther Thelen and Linda B. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action*. MIT Press, 1994.

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 *NeurIPS (Datasets and Benchmarks Track)*, 2023a.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models – a critical investigation. In *NeurIPS*, 2023b.

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *arXiv preprint arXiv:2409.13373*, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *ACL*, 2023. doi: 10.18653/v1/2023.acl-long.147.

Daniel M. Wolpert, Zoubin Ghahramani, and Michael I. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

Yue Wu, Xuan Tang, Tom M. Mitchell, and Yuanzhi Li. Smartplay: A benchmark for llms as intelligent agents. In *ICLR*, 2024.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

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 *NeurIPS*, 2023.

Alex L. Zhang, Thomas L. Griffiths, Karthik R. Narasimhan, and Ofir Press. Videogamebench: Can vision-language models complete popular video games? *arXiv preprint arXiv:2505.18134*, 2025a.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025b. URL https://arxiv.org/abs/2505.22954.

Xuan Zhang, Wenxuan Zhang, See-Kiong Ng, and Yang Deng. Self-evolving world models for LLM agent planning. *arXiv preprint arXiv:2606.30639*, 2026. URL https://arxiv.org/abs/2606.30639.

Tong Zheng, Xidong Wu, Zheng Zhang, Zhankui He, Chaoyi Zhang, Benjamin Coleman, Ruoqiao Wei, Di Bai, Haolin Liu, Rui Liu, Xue Wang, Yue Zhuan, Wang-Cheng Kang, Renkai Xiang, Heng Huang, Xinwu Cheng, and Yunsong Guo. Dream-RSI: Recursive self-improvement through evolving worlds. *arXiv preprint arXiv:2609.14858*, 2026. URL https://arxiv.org/abs/2609.14858.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *ICML*, 2024a.

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*, 2024b.

Wang Bill Zhu, Miaosen Chai, Ishika Singh, Robin Jia, and Jesse Thomason. PSALM-V: Automating symbolic planning in interactive visual environments with large language models. *arXiv preprint arXiv:2506.20097*, 2025. URL https://arxiv.org/abs/2506.20097.

A Code and Environments

Anonymous code, frozen Tier-M train/dev/test environments, evaluation scripts, and an interactive web demo are released at https://github.com/AnonymousPubCode/ICLR27-AlienBody. The repository README links a static project page with trajectory clips. The secret split is withheld during review. A named camera-ready release will replace this mirror upon acceptance.

B Extended Related Work

B.1 Action Interfaces and Exploration

WebArena Zhou et al. (2024b), VisualWebArena Koh et al. (2024), OSWorld Xie et al. (2024), SWE-bench Jimenez et al. (2024), and AgentBench Liu et al. (2024) evaluate agents using familiar browser, desktop, code-editing, and interactive interfaces. SmartPlay Wu et al. (2024), VideoGameBench Zhang et al. (2025a), and BALROG Paglieri et al. (2025) extend evaluation to games. Such interfaces offer useful prior information through action names and documentation, even when task-specific consequences must still be learned. AlienBody varies this prior while holding the underlying environment fixed, then measures action-model identification and downstream control. Table 4 compares these suites with AlienBody along the design axes the benchmark varies.

Table 4: Benchmark design comparison. Domains below suite names. Calibration denotes a pre-goal interaction phase; the diagnostic interventions test exploration, interpretation, and planning separately.

Design axis	**WebArena** web	**OSWorld** desktop	**SWE-bench** code	**BALROG** games	**Crafter** survival	**ARC-AGI-3** puzzles	**AlienBody** grid (ours)
Anonymous controls	✗	✗	✗	✗	✗	✓	✓
Pre-goal calibration	✗	✗	✗	✗	✗	✗	✓
Diagnostic interventions	✗	✗	✗	✗	✗	✗	Targeted
Name ablation	✗	✗	✗	✗	✗	✗	✓

Reinforcement learning (RL) exploration methods use prediction error (Intrinsic Curiosity Module Pathak et al. (2017)), novelty (Random Network Distillation Burda et al. (2019)), visitation counts Bellemare et al. (2016), and return-and-explore strategies Ecoffet et al. (2021) to collect informative experience; see Ladosz et al. (2022) for a survey. A specified action space does not provide its transition dynamics: those effects can themselves be learned from exploration. AlienBody brings a complementary measurement focus to this setting, asking how action labels affect exploration and whether acquired action knowledge can be used to plan.

B.2 Motor Babbling and Body-Schema Learning

Motor babbling connects exploratory movements to the acquisition of sensorimotor mappings Mahoor et al. (2018); Thelen and Smith (1994) and forward models Wolpert et al. (1995). Krakauer et al. (2019) distinguish motor adaptation from de novo skill acquisition. In robotics, exploratory interaction supports self-model learning Chen et al. (2021), body schemas Hoffmann (2021); Kawaharazuka et al. (2024), and intrinsically motivated skill acquisition Baranes and Oudeyer (2013). AlienBody uses an unfamiliar action mapping to operationalize this calibration problem for LLM/VLM agents, with a subsequent goal-directed phase that tests the utility of what was learned.

B.3 From Language-Mediated Control to Planning

ReAct Yao et al. (2022), Reflexion Shinn et al. (2023), Plan-and-Solve Wang et al. (2023), and Inner Monologue Huang et al. (2022) place language generation within the control loop, with different uses of feedback, memory, and planning. Our exact-mapping condition tests a specific demand on such a loop: composing known action effects into a successful trajectory.

PlanBench Valmeekam et al. (2023a) evaluates plan generation given a PDDL domain; follow-up studies probe name obfuscation and newer reasoning models on the same suite Valmeekam et al. (2023b; 2024). AlienBody adds interactive identification before planning and uses diagnostic

conditions to distinguish the two demands. Its text and image-plus-text observations also permit comparisons across input representations. LLM+P Liu et al. (2023) separates task translation from search: the LLM generates a problem PDDL file, while an externally supplied domain PDDL file specifies action rules for the classical planner. Our constructive experiments study both search over a supplied transition model and search over a model inferred from interaction.

B.4 Program Synthesis and Interactive World Models

Code-as-Policies Liang et al. (2022) and Voyager Wang et al. (2024) use LLM-generated programs to control embodied agents and build reusable skills. Program synthesis from examples Nijkamp et al. (2023) and evaluation-guided program search Romera-Paredes et al. (2024) make generated hypotheses executable and testable. HYSYNTH Barke et al. (2024) uses LLM samples to estimate a probabilistic context-free grammar, which guides symbolic synthesis. AlienBody's schema induction similarly produces an executable hypothesis, checked by replaying observed transitions. Active identification extends the available evidence by selecting actions on which surviving hypotheses disagree.

DreamerV3 Hafner et al. (2023) learns a neural world model from experience and uses imagined trajectories to train an actor. WorldLLM Levy et al. (2025) represents hypotheses in natural language, revises them through Bayesian inference, and collects new evidence with curiosity-driven RL. Its hypotheses condition an LLM's transition predictions. World-model evaluation increasingly targets behavior-level criteria rather than single-step state match Huang et al. (2026); Lambert et al. (2020); Li et al. (2026). Our constructive instances instead represent action effects as executable schemas and evaluate their use in verified planning, alongside held-out prediction and naming interventions. The focus is the relationship between model identification and control within an episode; the structured search methods' operator vocabularies and access permissions are specified in Appendix L.

C Full Experiment Results

Scope of the current release. Unless stated otherwise, grid experiments reported in this version use the **Tier-M** configuration (16×16 grid, $n = 4$ actions): 900 instances across six families (300 train / 120 dev / 300 test / 180 secret). Tier-L (24×24, $n = 6$) and Tier-XL (32×32, $n = 12$) are additionally instantiated for the Relational family as scale stress tests (Appendix L); Tier-S (Table 8) is parameterizable through the same API but not instantiated in this release.

C.1 Full Per-Family SR and CE Tables

Table 5 reports per-family success rate (SR) and Table 6 per-family calibration efficiency (CE) for all measured agents (Tier-M test, 50 env/family). The Qwen3.5 rows are the image+text condition; the text-only condition of Appendix J has no per-episode log in this release and is reported only in aggregate there. The five API rows are recomputed from the per-episode logs of the most recent complete run of each model (run dates 2026-07-28 to 2026-09-07); GPT-4o also has an earlier complete run on the same 300 environments that scores 38.3% rather than 33.0%, so differences of a few points between runs are within run-to-run variation at temperature 1.0. Pooled rates also mix families of very different difficulty: a family-clustered bootstrap (resampling the six family means, then each family's environment outcomes; 10,000 draws, seed 42) gives GPT-4o a pooled 95% interval of $[11.0, 59.0]$ around 33.0%.

[‡]DeepSeek V4 Pro covers 299 environments: `family6_test_049` has no recorded episode. All of families 3–5 and thirteen family-6 environments carry a same-day second run; we keep the later record, matching the `-resume` write order. Selecting the earlier record instead gives SR 30.8% and CE 0.216.

C.2 Full Per-Family CE Table

Table 6 reports CE per family for all measured agents (Tier-M test).

Table 5: Per-family SR% (Tier-M test, 50 env/family).

Agent	F1	F2	F3	F4	F5	F6
Random	4	6	2	2	10	4
Systematic	100	0	2	2	0	2
Bayesian	100	0	10	2	0	24
Qwen3.5-4B	34	2	6	0	4	4
Qwen3.5-9B	18	8	6	0	12	2
Qwen3.5-35B	40	6	2	0	4	14
Qwen3.5-397B	44	10	14	4	8	14
GPT-4o	88	26	56	4	14	10
Gemini 3.1 Pro	100	94	94	4	68	44
DeepSeek V4 Pro	64	34	52	2	38	20‡
GLM-5.2	100	98	92	6	76	26
Kimi K3	98	82	68	8	32	16
Oracle	100	100	100	100	100	100

Table 6: CE per family (Tier-M, 50 env/family).

Agent	F1	F2	F3	F4	F5	F6
Random	0.015	0.028	0.014	0.006	0.042	0.025
Systematic	1.033	0.000	0.028	0.028	0.000	0.031
Bayesian	1.033	0.000	0.120	0.044	0.000	0.230
Qwen3.5-4B	0.329	0.037	0.094	0.000	0.098	0.030
Qwen3.5-9B	0.137	0.060	0.051	0.000	0.106	0.015
Qwen3.5-35B	0.326	0.025	0.008	0.000	0.019	0.088
Qwen3.5-397B	0.200	0.053	0.064	0.024	0.046	0.096
GPT-4o	0.723	0.160	0.395	0.069	0.130	0.066
Gemini 3.1 Pro	1.022	0.819	0.866	0.018	0.479	0.342
DeepSeek V4 Pro	0.565	0.186	0.297	0.017	0.264	0.137‡
GLM-5.2	0.852	0.750	0.572	0.030	0.590	0.149
Kimi K3	0.812	0.500	0.475	0.050	0.265	0.086
Oracle	1.033	1.048	1.097	1.025	1.626	1.294

C.3 Exploration Quality Metrics

Table 7 reports per-family Phase-1 redundancy for the scripted agents and oracle, computed from full trajectory logs.

Table 7: Per-family exploration quality (Tier-M): average Phase-1 repetition rate, i.e. the fraction of Phase 1 actions that repeat previously-tested actions (lower is better).

Agent	F1	F2	F3	F4	F5	F6
Random	0.80	0.79	0.81	0.80	0.78	0.82
Systematic	0.00	0.00	0.00	0.00	0.00	0.00
Bayesian	0.00	0.00	0.03	0.02	0.00	0.05
Oracle	0.00	0.00	0.00	0.00	0.00	0.00

Random agents waste ∼80% of exploration budget on redundant testing. Systematic and Bayesian achieve near-zero redundancy on Adapt through deterministic enumeration, but this strategy fails on harder families where action effects are context-dependent.

D Environment Generation Details

Table 8: Difficulty tier configurations. Tier-M (tinted) is instantiated and evaluated for the full suite; Tier-L and Tier-XL additionally for Relational (Appendix L); Tier-S is parameterizable through the same API but not instantiated in this release.

Parameter	Tier-S	Tier-M	Tier-L	Tier-XL
Grid size	8×8	16×16	24×24	32×32
# Actions (n)	4	4	6	12
Phase 1 budget	12	20	25	40
Max total steps	30	50	80	120
Obstacles	0	3–6	5–8	10–15

All environments are generated programmatically. Grid cells use the Abstraction and Reasoning Corpus for AGI (ARC-AGI) standard 10-color palette ARC Prize Foundation (2026) with minimum 3 distinct colors per grid. Agent/target positions are sampled uniformly with minimum Manhattan distance constraints. Action mappings are randomized per episode: a uniform permutation of the four primitives for Adapt and Spatial; for Conditional, the two movement and two state operators are shuffled within their halves; for Relational, relation types are assigned to indices uniformly; for Compositional/Temporal, composite/temporal effects are similarly randomized. Obstacles are sampled uniformly among interior cells other than the agent start and target (the Adapt family is instantiated without obstacles). Every environment passes BFS solvability validation. Validation is run *from the spawn state* under the exact action mapping, i.e. it assumes perfect calibration and certifies the instance as generated rather than every state an agent can subsequently reach; an environment is accepted only if its shortest plan is reachable within the Phase-2 budget from that spawn state, and the F4 suite is further stratified so that this plan length falls in a per-instance band of 5–22 actions. To check that the latter caveat does not drive our results, we re-ran the same BFS from the endpoint of every recorded episode in the main F4 runs: of the 3,600 episodes recorded, 3,405 executed at least one real action, and from those endpoints the target remained reachable in 3,399 cases (99.8%) and within 30 further actions in 3,394 (99.7%). All six unreachable endpoints come from a single environment containing a genuine dead-end region, entered independently by six different agents; the failures in our main results are therefore not an artifact of agents exploring into unsolvable states.

Tier scaling. The tier configurations are realized through the same generator API (grid size, action count, obstacle range, and minimum agent–target distance $d_{\min} = 5/8/16$ for Tier-M/L/XL). In this release the Relational family is additionally instantiated at Tier-L (n=6: Tier-M's first four relation types plus farthest-same-color teleport and anti-brightest step) and Tier-XL (n=12: the full

vocabulary; Appendix L); Tier-M is evaluated across the full suite. Tier-S is parameterizable through the same API but is not instantiated in this release.

E RL Baseline Details

RL baseline results. We trained Proximal Policy Optimization (PPO; convolutional neural network / CNN policy, 200K and 500K timesteps, entropy coefficient 0.01) and meta-RL / RL^2 (long short-term memory / LSTM policy, 2000 episodes) on Family 1 environments. PPO achieved SR $= 2.0\%$ (200K steps) and SR $= 0.0\%$ (500K steps), both worse than random (SR $= 4.7\%$), with zero exploration coverage (EC $= 0.00$), indicating total policy collapse to a deterministic action. The policy here is feedforward and conditions on a single observation, so a per-episode randomized mapping is not identifiable from any one step; memorylessness bounds what this configuration can adapt to, though we do not read the collapse as architecturally inevitable. Meta-RL, which uses an LSTM across episodes to learn exploration strategies, obtained zero positive reward across all 2000 training episodes. That run comprises on the order of 10^5 environment steps, so it characterizes behavior under this interaction budget rather than an asymptotic limit of recurrent meta-RL. These negative results hold for the configurations we trained (PPO, RL^2; Family 1 only): standard reward-driven learning in these settings did not discover the per-episode randomized mapping. These findings are consistent with our core argument that action calibration in this benchmark demands structured exploration strategies that these reward-driven baselines did not find.

F Prompt Templates

Agents in the main evaluation share one system prompt rendered from a single template (`build_system_prompt`, variant MINIMAL, `assist_level`$= 0$, `familiar`=False), so that what varies across our conditions is the action interface rather than the instructions; the interface-ladder arms (Appendix H.6) and the frozen interface study (Appendix H.7) use their own protocol-specific prompts, documented with them. For the Tier-M setting the system prompt reads as follows, with the released `prompts.py` the authoritative copy (the paper's typewriter font renders the template's em dash as a double hyphen):

```
You are in a grid world.  You can see your position (a white square with
a red arrow showing your facing direction) and the environment.
You have 4 available actions:  Action 0, Action 1, Action 2, Action 3.
You do NOT know what these actions do -- their effects are unknown and
randomized.
The episode has two phases:  - Phase 1 (Calibration):  Explore freely
to learn what each action does.  You can say "DONE" to end exploration
early.  - Phase 2 (Execution):  A target location (cyan border) will
appear.  Navigate to it efficiently.
Output ONLY a single number (0-3) to choose an action, or "DONE" to end
Phase 1.
```

Braced fields are template placeholders: `{n_actions}` renders the action count (4 here) and `{action_list}`/`{action_range}` the enumerated list and the integer range. The label-manipulation conditions reuse this template verbatim, inserting an interface-label block immediately above the "You do NOT know what these actions do" line. The chain-of-thought variant appends:

```
Before choosing your action, reason step-by-step:  1.  What have I
learned about each action so far?  2.  Which actions remain untested?
3.  What should I do next?
Then output your action on the LAST line as just a number.
```

G Action Type Details

Full specifications for each action type family:

Adapt (remapped): Each action maps to {up, down, left, right} via random permutation. The simplest family: one observation per action suffices.

Spatial (rotation): Actions are `rotate_cw`, `rotate_ccw`, `forward`, `backward`. Forward/backward move relative to facing direction. Requires orientation tracking and composition of rotation with translation.

Conditional (state-modifying): each episode combines two movement operators (`rotate_cw`, `forward`) with two state operators (`color_inc`, `color_move`); `color_inc` changes the internal color state (no movement); `color_move` moves in direction $f(c)$ where $f : \{0, 1, 2, 3\} \to \{N, E, S, W\}$; agent color state $c \in \{0, 1, 2, 3\}$. The operator library also defines `color_dec` and `color_interact`; neither is sampled in this release, so no episode contains them, none is required for any episode to complete, and their semantics enter no reported number.

Relational: `move_to_nearest_diff_color`, `move_to_nearest_same_color`, `move_toward_brightest`, `flee_same_color`. Effects depend on spatial relationships between agent and all grid objects. The two `move_to_nearest_*` operators teleport to the nearest cell satisfying the relation (Manhattan distance, row-major tie-break, the agent's own cell excluded, obstacles *not* excluded); `move_toward_brightest` and `flee_same_color` take a single step instead. Operational rules: Algorithm 1 and Appendix L.13.

Compositional: `move_and_rotate_cw` (forward + rotate_cw), `rotate_and_strafe` (rotate_ccw + forward), `double_forward`, `retreat_and_spin` (backward + 180° rotation). Decomposition required for understanding.

Temporal: Action 0 after self = north, else east. Action 1 after self = south, else west. Action 2 = repeat previous movement. Action 3 = inverse of previous movement. Requires tracking action history; when no prior movement exists, Actions 2 and 3 are no-ops.

H Evaluation Protocol Details

H.1 CA-proxy Measurement

Calibration accuracy (CA) is a *trajectory-heuristic proxy*: for each action, fixed per-family rules are applied to the agent's *first* Phase-1 observation of that action (*e.g.*, displacement distance → relational effect name), and the reported score is the fraction of inferred effects matching the ground-truth mapping. No model utterance enters the score—it is a function of the trajectory alone, for every agent. Our model-output-based measure of the same quantity is held-out next-state prediction: on probes covering unseen positions, directions, colors, and histories, archived forward models score 0.71–0.73 (F4) and 0.80–0.89 (F5), against 1.00 for the oracle positive control and 0.05–0.08 for permuted/no-op negative controls.

A complementary, utterance-level probe asks the model to state the mapping outright. We replay each stored Phase-1 conversation exactly (same renderer, prompt template, history trimming, and parser, with the stored responses fed back in order) and then append one multiple-choice quiz (one line per action, one letter per effect, options shuffled and rotated), scoring the model's *stated* mapping against ground truth; a second arm asks the same quiz with no interaction (prior). Table 9 reports stated versus proxy CA on the probe's 50-environment subsets (chance $= 1/4$; a per-family breakdown appears in the appendix of the released results). Three readings. First, the proxy overstates what four of five models can state, by 0.18–0.33 CA: GPT-4o $0.74 \to 0.49$, GLM-5.2 $0.81 \to 0.63$, Kimi K3 $0.81 \to 0.54$, DeepSeek V4 $0.80 \to 0.47$. Without interaction every model sits near chance (prior 0.22–0.34), so the proxy credits far more than the model's label-free prior. Gemini 3.1 Pro is the one row whose stated CA (0.74) approaches its proxy (0.78); since its answer budget is 64× that of the other four (*Budget note* below), we do not read that row as evidence that this model internalised more, and we draw no cross-model conclusion from the one row whose answering conditions differ. Second, the shortfall is family-specific rather than uniform: on Temporal (F6) all five models have mean stated accuracy below the 0.25 chance reference after interaction (0.09–0.19 vs. 0.25), and on Relational (F4) stated CA stays at 0.19–0.41 (the two families where success collapses; Section 4) while on F2/F5 the strongest model states the mapping near-perfectly (1.00). Third, interaction is the main source of stated knowledge where it exists: F1–F3 priors sit at 0.16–0.63 and rise to 0.42–1.00 post. *Budget note.* Gemini 3.1 Pro is a long-thinking model whose quiz calls think 20–27k tokens; its answer budget was raised to 16,384 tokens with a 420 s HTTP read timeout. At smaller budgets (256; 4,096) its rows truncate mid-answer, and because the truncated cells concentrate on the hard

Table 9: Stated vs. trajectory-proxy calibration accuracy (CA), 50 environments per model; both columns scored on the same subsets. "Stated" replays the model's own Phase-1 conversation and asks it to name each action's effect (post), or asks with no interaction (prior). Chance is $1/4$ (four shuffled effect options). Δ is proxy $-$ stated (post).

Model	Proxy CA	Stated post	Stated prior	Δ	Chance
GPT-4o	0.735	0.490	0.270	+0.245	0.25
Gemini 3.1 Pro	0.780	**0.740**	0.245	+0.040	0.25
DeepSeek V4 Pro	0.800	0.470	0.340	+0.330	0.25
GLM-5.2	0.805	0.625	0.325	+0.180	0.25
Kimi K3	0.805	0.535	0.220	+0.270	0.25

families (F4/F6) such rows would *inflate* the proxy–stated gap spuriously; they are discarded. The reported Gemini row is the final protocol (0 errors; 1/50 incomplete). The other four models use the fixed-format answer at 256 tokens (0–2 parse failures).

H.2 CE Human Denominator

CE and $\mathrm{CE}_{\mathrm{cond}}$ use the same family-specific reference constant b_f in the numerator and each episode's total action count in the denominator. The constants are informal human-step estimates from an $n{=}1$ pilot, not a formal human baseline: Adapt 13.5, Spatial 15.0, Conditional 16.5, Relational 15.5, Compositional 17.0, and Temporal 18.5. We retain them as fixed normalization constants; SR and action counts provide the direct performance measures. Within a family, changing b_f uniformly rescales CE. Across families, these constants act as weights and can affect pooled model rankings.

CE assigns zero to failures before averaging; $\mathrm{CE}_{\mathrm{cond}}$ averages only over successful episodes, so on the same episode set $\mathrm{CE} = \mathrm{SR} \cdot \mathrm{CE}_{\mathrm{cond}}$ holds by construction (SR as a fraction). $\mathrm{CE}_{\mathrm{cond}}$ is therefore undefined only for a row with no successful episode; a row that takes no Phase-1 steps still has one, because the denominator is each successful episode's total action count. The Qwen3.5-35B text row is such a row (P1=0, EC=0, no per-episode log in this release); its $\mathrm{CE}_{\mathrm{cond}}$ (0.585) is the value the identity fixes from that row's printed CE and SR. Low CE with high conditional efficiency indicates infrequent success, whose cause must be assessed using exploration and knowledge measurements.

H.3 L2 Is Specified but Not Evaluated

The L2 protocol (systematic action tests in Phase 1, followed by a structured observation summary for execution) is part of the design but is *not* evaluated in this paper, and no L2 number is claimed anywhere in the main text. Three reasons, none of them a result.

First, the protocol as implemented and as specified diverged: the archived runs use a Phase-1 round count that the specification does not license, so what ran is not the treatment L2 names. Second, the summary text actually injected in those runs has no preserved artifact—the code path that builds it changed afterwards, so byte-identity with the current `build_phase1_summary` cannot be established; the protocol may be described as specified, not as executed. Third, the runs that exist cover GPT-4o and Gemini 3.1 Pro only, and no Qwen model has an L2 run at all, so the roster is incomplete for every claim L2 was meant to support.

A further structural limit applies even where runs exist: L2's Phase-1 trajectories are constructed by systematic testing and L0's by free exploration, and the two also differ in execution starting state and remaining budget. L0→L2 therefore does not isolate the summary representation from the collection policy that produced it, and we draw no causal conclusion about either. Historical L2 rows are retained in the evidence ledger as provenance (*e.g.*, on F6 the Gemini 3.1 Pro row moves $36\% \rightarrow 18\%$ with $n{=}50$; we read this as a single model $\times$ family observation, not as a general "more assistance is worse" direction). The proposed three-arm design that would separate collection strategy from summary representation (same exploration trajectory, with a raw-trace arm replacing the summary) is left to future work.

H.4 Scope of L1 and of the Workflow Suite

L1 is a single-family diagnostic: it is instantiated only on Adapt (F1), where the familiarity cue is the statement that the controls are scrambled cardinal directions. That cue has no counterpart on F2–F6 that is not itself a new experimental condition, so L1 is not a benchmark-wide level of assistance; its results are confined to Adapt (the Familiar/Normal comparison of Appendix L) and are presented as such.

The Workflow Suite (Section 3.3) is an auxiliary planning and interface instantiation: it supplies its tool schemas, it includes no anonymous calibration or Phase 1 exploration, it is exercised only within the matched interface study (Section 4.4, Appendix H.7), and its results bear only on the interface comparison (RQ2), not on acquisition from interaction.

H.5 Step Budgets of the Archived Ladder Arms

The environment enforces a single total-step counter: the calibration cap is an auto-switch deadline, not a deduction, so calibration steps an agent does not spend remain available for execution (`env/core.py`, `AlienBodyEnv.step`). The archived arms therefore do not share one execution budget, and the differences run in favor of the mapping-supplied arms. Read off the `phase1_steps`/`phase2_steps` fields of the episode records: the L3 arms, which skip exploration, ran with the full 50 execution steps; the L2 runs used 8 calibration steps and 42 execution steps; free-exploration arms used as many as 46, since a four-step calibration phase leaves the rest of the 50. Nothing was re-run or backfilled. Phase 2 always resumes from the coordinate where calibration ended, so arms with different Phase-1 policies also begin execution at different distances from the goal. We checked this against the recorded episodes rather than assuming it away: restarting BFS from the endpoint of every recorded F4 episode leaves the target reachable in 3,399 of the 3,405 episodes that executed at least one action (99.8%), and within 30 further actions in 3,394 (99.7%) (Appendix D). The consequence for interpretation is one-directional: the L3 wall (0–10% with the ground-truth mapping supplied) is measured at the largest execution budget in the suite, so equalizing execution budgets to 30 steps would not have removed it.

H.6 Interface Affordances by Arm

The interface arms of this appendix differ in what they expose, not only in output format; Table 10 states the exact budgets. This ladder (protocol v3/v4) predates the matched ladder of Section 4.4: it runs on the published 50-environment F4 split, with a shared 1,000-query cap for its oracle arms; the earlier high-budget generation is listed separately below. Every completed arm has a cap of 30 Phase-2 actions on that split with the canonical `STATE_V2` serialization and strict parsers that record malformed outputs as `action=None` rather than silently mapping them to action 0. Interface-only arms (I0 stepwise, I1 scratchpad, I2 plan-first) receive the action→effect mapping description in the prompt and act directly. The conversational arm (I3) adds a verified `next_state` transition oracle; I3 and the code arm (I4) share the same 1,000-call oracle budget. I3L is a context-tolerant control for I3: it keeps the same per-call budget accounting but re-issues each request in a fresh context and renders a ledger of *distinct* calls, so the prompt grows with distinct queries rather than with the call count. The code arm runs a restricted Python sandbox that *exposes the shared state as a parsed dictionary* (`STATE`, identical in content to the prompt's `STATE_V2`; no extra grid or mapping globals) and a `next_state` oracle under the same 1,000-call budget, accepting either a full `PLAN` or a single `ACTION`. The completed arms (Table 11) separate by interface rather than by knowledge: I0/I1/I2 stay at $\leq$2% for DeepSeek V4 Pro and Qwen3.5-9B; I3 scores 0/50 for Qwen3.5-9B, and I3L removes every context-wall failure (40→0) yet still scores 1/50, so removing the observed context errors alone is insufficient for this model; I4 reaches **62%** (GPT-4o), **50%** (DeepSeek V4 Pro) and 4% (Qwen3.5-9B). The code result should therefore be read as *verified search assembled by the model over programmable state*, a joint code+oracle effect, not as an isolated effect of code syntax: execution, memory, and query scheduling also differ; for the strong models the matched conversational control under this ladder has not been run, so we do not claim that dialogue cannot assemble search. The same question is re-run for DeepSeek Flash and an independent Qwen model family under one matched protocol in Section 4.4 (full tables in Appendix H.7).

Table 10: Per-arm interface affordances and budgets under the earlier ladder (protocol v3/v4), single-token row labels: **I0** stepwise, **I1** scratchpad JSON, **I2** plan-first JSON, **I3** conversational `next_state`, **I3L** its ledger control, **I4** joint code+oracle REPL, **I5** symbolic BFS reference.

Arm	State access	Transition oracle	Budget
I0 stepwise	shared state JSON	none	30 actions
I1 scratchpad	shared state JSON	none	30 actions
I2 plan-first	shared state JSON	none	30 actions
I3 conversational	shared state JSON	textual, 1,000 calls	30 actions
I3L ledger control	shared state JSON	textual, 1,000 calls, distinct ledger	30 actions
I4 code REPL	same state, parsed as `STATE`	`next_state`, same 1,000 calls	up to 4 rounds
I5 BFS reference	shared state	same oracle, 1,000 calls	30 actions

Historical configuration, not matched. An earlier ladder generation ran on the same 50 F4 environments over GPT-4o / DeepSeek V4 Pro / GPT-5.1: interface-only L3 (0/0/4%), a conversational `next_state` tool capped at 120 calls (26/16/10%), and a Python REPL over preloaded grid state with a 50,000-call budget (98/100/90%). Its oracle budgets are unmatched across arms and generations (120 vs. 50,000 then; 1,000 now) and its model set differs (GPT-5.1 rather than Qwen3.5-9B), so the two generations are never plotted or tested as one series.

Table 11: Completed F4 interface results for the earlier ladder (successes / 50 environments). All model runs below use its 30-action contract; I3, I3L and I4 allow 1,000 oracle calls; I3L uses the separately documented ledger protocol. Dashes denote untested full-size arms; preflights are excluded. The deterministic I5 reference solves 50/50 with the same oracle cap.

Model	I0	I1	I2	I3	I3L	I4
GPT-4o	—	—	—	—	—	31/50
DeepSeek V4 Pro	1/50	0/50	1/50	—	—	25/50
Qwen3.5-9B	1/50	1/50	0/50	0/50	1/50	2/50

H.7 Frozen Interface Study: Protocol and Full Results

Tasks and models. We evaluate DeepSeek V4 Flash through the recorded gateway endpoint and the locally served `Qwen/Qwen3.5-35B-A3B-FP8` checkpoint (thinking disabled, vLLM 0.17.1, one H200). Both use text-state inputs. The fresh F4 set contains 400 instances disjoint from the earlier published split; the Boolean workflow set contains 400 instances. Both sets have fixed A/B halves of 200 and easy/medium/hard quotas of 120/160/120. Workflow schemas are excluded when isomorphic to a development instance. The 900-instance base benchmark and these two targeted suites are separate evaluation sets.

Interfaces and budgets. J0 receives an event table and emits one action per round. J1 adds a persistent draft capped at 2,000 characters. J2 emits one complete plan. J3L augments J1 with a transition oracle and a ledger of distinct queries with repeat counts; each request is reconstructed from the current state, events, ledger, and draft. J4 receives the public state as a Python dictionary, the same transition oracle, and a restricted sandbox; it returns a plan or an action in at most four submissions. J5 is deterministic BFS and makes no model calls. Within each task, real-action limits are shared (30 for F4, 24 for workflow), as are the oracle caps (1,024 and 4,096). These cover all reference BFS searches, whose maxima are 986 and 2,178 queries. Model-call counts, generated tokens, and execution organization differ between interfaces; equal oracle caps do not equate their total computation.

Recorded protocol deviation. The planned J4 query ledger was not populated by the frozen sandbox implementation. Code and errors persist across repair rounds, but program variables and individual query results do not. All results retain this as-run behavior. Aggregate oracle calls are counted; J4 distinct-query and repetition statistics are unavailable. The comparison therefore does not isolate execution under identical cross-round query memory. The input snapshot and frozen implementation are preserved alongside this deviation record.

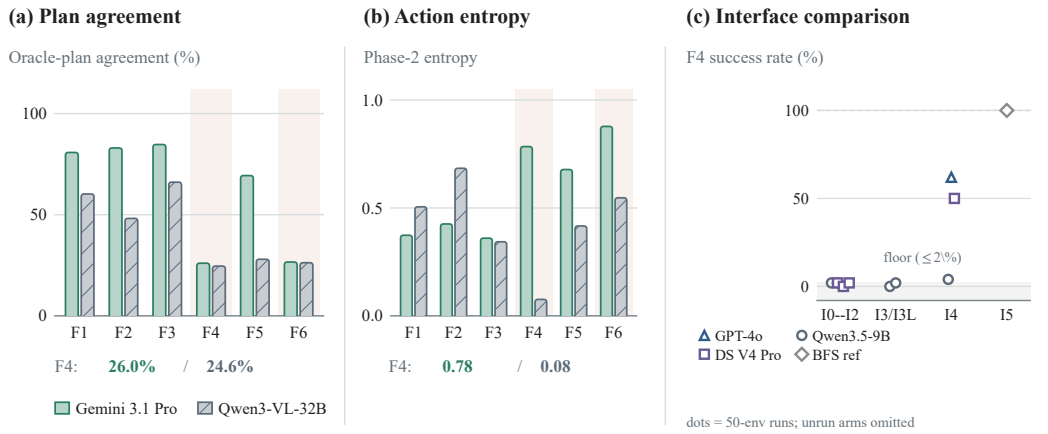


Figure 5: Historical diagnostics and the v3/v4 interface comparison. (a) Oracle-plan agreement at L3; shaded columns highlight F4/F6. (b) Phase-2 action entropy. F4 values appear below both panels, in the same model order as the legend. (c) F4 success by interface arm under the earlier ladder protocol: I0 stepwise, I1 scratchpad, I2 plan-first, I3 conversational `next_state` (I3L, its ledger control, for Qwen3.5-9B), I4 the joint code+oracle REPL, I5 a symbolic BFS reference; each dot is a stored run of 50 environments, an arm a model did not run is omitted, and shading marks the $\leq 2\%$ floor. I4 parses the shared state into a Python dictionary (Appendix H.6); the same comparison rebuilt under one matched protocol, on a different instance set and with stepwise controls for every model, is Table 3.

Table 12: Per-cell success counts for the frozen interface study. Each half is 200 environments; % is pooled over the 400. J0 and J2 were run for DeepSeek V4 Flash only, and Qwen was run on J1/J3L/J4 (the arms of the registered contrast). J5 is a deterministic reference.

	DeepSeek V4 Flash			**Qwen3.5-35B-A3B**		
Relational F4	A	B	%	A	B	%
J0 (table only)	6	3	2.25	—	—	—
J1 (+draft)	12	12	6.00	6	5	2.75
J2 (plan-first)	1	4	1.25	—	—	—
J3L (+oracle, ledger)	9	12	5.25	3	3	1.50
J4 (code+oracle)	110	110	55.00	72	77	37.25
J5 (BFS reference)	200	200	100.00	200	200	100.00
Workflow	A	B	%	A	B	%
J0 (table only)	5	12	4.25	—	—	—
J1 (+draft)	181	170	87.75	140	127	66.75
J2 (plan-first)	3	10	3.25	—	—	—
J3L (+oracle, ledger)	153	152	76.25	74	70	36.00
J4 (code+oracle)	106	114	55.00	179	168	86.75
J5 (BFS reference)	200	200	100.00	200	200	100.00

Design and analysis. Prompts, parsers, budgets, task sets, and the primary endpoint were fixed before formal evaluation. Each arm of a task–model pair uses identical environments. The primary endpoint is J4−J3L, paired by environment, with 10,000 bootstrap resamples and exact two-sided McNemar tests; Holm correction covers the four primary task–model comparisons. The A/B halves retain their original assignments. J4−J1 and J3L−J1 are prespecified secondary contrasts, reported with pointwise intervals. A planned Gemini validation subset was not run and contributes no observations. All 7,200 recorded episodes (6,400 model episodes and 800 shared BFS references) remain in the analysis, including four Flash F4/J1 transport failures. Input files and manifests are hash-verified before analysis. Per-cell counts for every arm and half are in Table 12.

Secondary comparisons. On F4, J4−J1 is $+49.00$ points for Flash (95% CI $[43.75, 54.25]$) and $+34.50$ for Qwen ($[29.50, 39.50]$). On workflow the differences are -32.75 ($[-38.50, -27.00]$) and

Table 13: Study C, primary contrast: $J4 - J3L$ at a 256-query cap under format-retry, 100 environments per model, paired by environment. The interval is the block-stratified environment-resampled bootstrap; p is the exact two-sided McNemar test. Holm over the pre-registered two-model family gives thresholds 0.0250 and 0.0500, and both rows reject.

Model	SR $J3L \to J4$	Δ	95% CI	only $J4$/$J3L$	McNemar p
DeepSeek V4 Flash	$1.0\% \to 91.0\%$	$+90.0$ pp	$[+84.0, +95.0]$	$90/0$	1.62×10^{-27}
Qwen3.5-35B-A3B	$1.0\% \to 56.0\%$	$+55.0$ pp	$[+45.0, +65.0]$	$56/1$	8.05×10^{-16}

$+20.00$ ($[14.75, 25.25]$). J3L$-$J1 is -0.75 ($[-3.50, 2.00]$) and -1.25 ($[-3.00, 0.25]$) on F4, and -11.50 ($[-16.00, -7.00]$) and -30.75 ($[-36.25, -25.00]$) on workflow. Intervals covering zero do not establish equivalence.

Error accounting. J3L records 389/2,385 rejected responses on F4 and 515/3,195 on workflow for Flash/Qwen. The corresponding J4 code-error counts are 989/963 and 958/373; these are submission counts, not failed-episode counts. J4 does not use the stepwise parser, so its zero in that parser's failure field is not a comparable error rate. Offline replay reproduces the recorded code-error totals in all 800 Flash J4 episodes; forbidden imports dominate those rejections. Successful episodes often recover within the repair budget. These observations expose protocol friction but do not measure the counterfactual success rate under a more permissive parser or sandbox.

Query repetition. J3L repetition is computed within each episode from the ledger's counts, then aggregated; identical state–action keys from different environments remain distinct. For Flash, repeated attempts account for 52.82%/53.26% of F4 queries in A/B, and 57.61%/53.35% on workflow. J4 has no recorded per-query ledger, so the corresponding comparison is unavailable.

H.8 Interface Throughput and Format-Retry Sensitivity

A third registered study (C) uses the same two models as Appendix H.11 and a *nested subset* of the same registered environments: the first fifty of block A and the first fifty of block B of that study's frozen identifier file, resolved through the runner's own path (`dataset_sha256` prefix `ad165359aaf2cec9`), one hundred in all, with every cell below a complete pass over all one hundred. It is nonetheless a different intervention on a different protocol: study B varies the information supplied *inside* an episode, whereas study C varies the interface, the query capacity, and the response policy, with both of its interfaces receiving the full operator definitions. The two studies are therefore neither pooled nor read against each other. The registered primary contrast is $J4 - J3L$ at a 256-query cap under the *format-retry* response policy, paired by environment, with a block-stratified paired bootstrap (10,000 draws, seed 42) and an exact two-sided McNemar test, Holm-corrected over the pre-registered two-model family. Every other quantity in this appendix is a *sensitivity* contrast: a paired difference with a bootstrap interval, given no corrected significance claim and described as significant nowhere.

Table 13 reports the primary contrast. The gap between the conversational and the programmable interface does not close when outer format friction is absorbed: success rises from 1.0% to 91.0% for DeepSeek V4 Flash (paired Δ $+90.0$ pp, stratified 95% CI $[+84.0, +95.0]$; 90 environments where only $J4$ succeeds against none where only $J3L$ does; exact McNemar $p = 1.62\times10^{-27}$, below the Holm threshold 0.0250) and from 1.0% to 56.0% for Qwen3.5-35B-A3B ($+55.0$ pp, $[+45.0, +65.0]$, $56/1$, $p = 8.05\times10^{-16}$, threshold 0.0500). The same-cap as-run contrast is of the same order ($+78.0$ and $+36.0$ pp), so a large gap is present with and without the retry policy. The two policies are separate conditions, and their difference is not a decomposition of the primary gap: Flash's $J4$ cell at cap 256 absorbed no outer rejection at all (Table 15), yet its two runs differ by ten points ($[+0.0, +20.0]$), an interval that includes zero, a difference no repair mechanism can account for.

Scope of the repair condition. Format-retry re-asks at the interface seam when a response cannot become an action or a program, with a budget of four re-asks per episode and a fallback to as-run semantics when the budget is exhausted; the feedback it sends is a constant instruction plus the rejection kind and never echoes the rejected response. It therefore measures how much of the interface gap survives *outer* format friction. It does not measure in-loop repair: errors raised at execution time stay inside the frozen loop's own rounds in both arms, and this contrast is not evidence about runtime repair in either direction.

Table 14: Study C, sensitivity contrasts (paired by environment, 100 environments per row; intervals are the block-stratified bootstrap). Neither block carries a corrected significance claim.

Model	Interface	Contrast	Δ	95% CI
Response policy: format-retry − as-run, at cap 256				
DeepSeek V4 Flash	$J3L$		-2.0 pp	$[-5.0, +0.0]$
DeepSeek V4 Flash	$J4$		$+10.0$ pp	$[+0.0, +20.0]$
Qwen3.5-35B-A3B	$J3L$		-1.0 pp	$[-4.0, +2.0]$
Qwen3.5-35B-A3B	$J4$		$+18.0$ pp	$[+6.0, +30.0]$
Query capacity: cap 256 − cap 32				
DeepSeek V4 Flash	$J3L$	as-run	$+0.0$ pp	$[-4.0, +4.0]$
DeepSeek V4 Flash	$J3L$	format-retry	-2.0 pp	$[-6.0, +2.0]$
DeepSeek V4 Flash	$J4$	as-run	$+2.0$ pp	$[-9.0, +13.0]$
DeepSeek V4 Flash	$J4$	format-retry	$+5.0$ pp	$[-4.0, +14.0]$
Qwen3.5-35B-A3B	$J3L$	as-run	$+0.0$ pp	$[-3.0, +3.0]$
Qwen3.5-35B-A3B	$J3L$	format-retry	$+0.0$ pp	$[-3.0, +3.0]$
Qwen3.5-35B-A3B	$J4$	as-run	-11.0 pp	$[-23.0, +1.0]$
Qwen3.5-35B-A3B	$J4$	format-retry	$+16.0$ pp	$[+3.0, +29.0]$

Response policy and capacity. Table 14 reports the two sensitivity axes. At a fixed 256-query cap, switching from as-run to format-retry leaves the conversational arm essentially unchanged (Flash -2.0 pp $[-5.0, +0.0]$; Qwen -1.0 pp $[-4.0, +2.0]$) and raises the programmable arm (Flash $+10.0$ pp $[+0.0, +20.0]$; Qwen $+18.0$ pp $[+6.0, +30.0]$). On the capacity axis, raising the cap from 32 to 256 changes the conversational arm by at most two points in either direction, and the programmable arm by $+2.0$ pp $[-9.0, +13.0]$ (Flash, as-run) and $+5.0$ pp $[-4.0, +14.0]$ (Flash, format-retry); for Qwen's programmable arm the capacity contrast changes sign with the policy in force (-11.0 pp $[-23.0, +1.0]$ under as-run, $+16.0$ pp $[+3.0, +29.0]$ under format-retry), so its capacity effect is defined only together with the policy it is measured under; the disagreement between the two is consistent with a capacity–policy interaction, an exploratory reading that the registered design does not size or test and from which no claim is taken. These are directions of movement under protocol variation, not established effects.

What the re-asks absorb. Table 15 lists the interface-seam rejections the retry arm intercepted. In the conversational arm those rejections are parse-shape failures (responses that are not JSON, JSON that never terminates, drafts over the cap, or fenced JSON where plain JSON is required), and they are frequent: 174–179 extra calls per 100 Flash episodes and 387–388 per 100 Qwen episodes. In the programmable arm the only rejection kind is a missing fenced program block, and it is rare: two intercepts for Flash at cap 32 and none at cap 256, 23 and 16 for Qwen. The arm whose responses repeatedly fail to parse is also the arm that gains least from re-asking, which is consistent with the small policy effect on that arm's success rate in Table 14.

Resource accounting. Table 16 gives paired per-episode resource means for the primary cells, counting the calls the frozen loop itself saw; the extra calls a retry adds are additional to those and appear separately in Table 15 (174 for Flash and 388 for Qwen over the hundred $J3L$ episodes of the primary cell, against 0 and 16 for $J4$). Adding them, the programmable arm still issues under a tenth of the model calls, roughly an order of magnitude less input text and about half the output text, and the loop's wall-clock time is lower. These are provider-reported usage counters for the model calls made, not a measure of the total computation a program performs: a $J4$ episode executes code in the sandbox, which the counters do not price. For the 334 retry episodes launched before the ledger amendment the repair-attributable tokens were never recorded, so complete total-token comparisons are unavailable for those cohorts and we report recorded usage together with its missing coverage; where the missing quantities fall on both arms (Qwen's primary cell), the recorded reduction is not a bound on the true reduction, while Flash's $J4$ cell at the primary budget absorbed no rejection at all, so its recorded reduction remains a lower bound. Two counters rise because the programmable arm executes programs: code errors appear where the conversational arm has none, and for Flash more steps are served by the oracle during execution (12.6 → 42.4). The oracle's conversational channel is used less: Flash's $J3L$ arm issues 6.0 distinct queries per episode and Qwen's 1.9, while the $J4$ cells keep no per-query ledger at all: their stored distinct-query counter is zero, but the served-query counter on the same episodes is not (42.4 and 2.6), so that zero is a recording gap rather than evidence that the oracle went unused, and no statistic here is computed from it. Where a per-success statistic is

Table 15: Study C, rejections absorbed by the format-retry arm, from the runner's repair sidecar, over 100 episodes per row. Rejection kinds and extra calls are recorded for both cohorts; the token column counts only calls the frozen loop never saw, and for episodes launched before the ledger amendment those token quantities were never recorded, so a mixed cell reports a lower bound (*unknown*, never zero).

Cell	Rejections absorbed (kind: count)	Extra calls	Intercepted output tok.
DeepSeek V4 Flash, $J3L$, cap 32	`not_json` 147, `unterminated_json` 14, `draft_over_cap` 14, `fenced_json` 4	179	199,897
DeepSeek V4 Flash, $J3L$, cap 256	`not_json` 146, `draft_over_cap` 14, `unterminated_json` 12, `fenced_json` 2	174	0 + unknown (100 ep.)
DeepSeek V4 Flash, $J4$, cap 32	`no_fenced_block` 2	2	32,000
DeepSeek V4 Flash, $J4$, cap 256	none	0	0
Qwen3.5-35B-A3B, $J3L$, cap 32	`not_json` 171, `fenced_json` 164, `unterminated_json` 40, `draft_over_cap` 12	387	2,078,283
Qwen3.5-35B-A3B, $J3L$, cap 256	`fenced_json` 184, `not_json` 140, `unterminated_json` 48, `draft_over_cap` 15, `action_out_of_range` 1	388	1,189,029 + unknown (34 ep.)
Qwen3.5-35B-A3B, $J4$, cap 32	`no_fenced_block` 23	23	368,000
Qwen3.5-35B-A3B, $J4$, cap 256	`no_fenced_block` 16	16	0 + unknown (100 ep.)

Table 16: Study C, paired per-episode means in the primary cells ($J3L \rightarrow J4$ at 256 cap under format-retry; 100 environments, paired). Means and differences are rounded independently, so a displayed difference may differ from the displayed means in the last digit. Counters are those written by the frozen loop; repair-arm extra calls are excluded and appear in Table 15. *n/a* marks a quantity the arm does not record rather than a measured zero: the $J4$ path serves oracle queries without writing a per-query ledger, so its distinct-query count is unavailable and its stored zero is a recording gap (the served-query row is non-zero on the same episodes).

Quantity	DeepSeek V4 Flash	Qwen3.5-35B-A3B
model calls	$41.9 \rightarrow 3.2$ (-38.7)	$37.5 \rightarrow 2.4$ (-35.0)
input tokens	$99{,}285.3 \rightarrow 10{,}267.9$ ($-89{,}017.4$)	$85{,}084.0 \rightarrow 8{,}094.2$ ($-76{,}989.8$)
output tokens	$8{,}032.7 \rightarrow 4{,}314.3$ ($-3{,}718.4$)	$42{,}005.6 \rightarrow 21{,}086.9$ ($-20{,}918.6$)
parse failures	$0.1 \rightarrow 0.0$ (-0.1)	$5.1 \rightarrow 0.0$ (-5.1)
code errors	$0.0 \rightarrow 2.2$ ($+2.2$)	$0.0 \rightarrow 1.6$ ($+1.6$)
valid real actions	$29.1 \rightarrow 11.4$ (-17.7)	$24.9 \rightarrow 12.8$ (-12.1)
oracle served	$12.6 \rightarrow 42.4$ ($+29.8$)	$7.5 \rightarrow 2.6$ (-4.9)
distinct queries	$6.0 \rightarrow$ n/a	$1.9 \rightarrow$ n/a
wall seconds	$93.2 \rightarrow 16.1$ (-77.0)	$456.7 \rightarrow 187.9$ (-268.8)

measurable at all it agrees: in the programmable arm at the primary cell, a successful Flash episode costs 3.1 calls, 9,953 input and 4,246 output tokens (Qwen: 2.8, 9,579, 21,848). The conversational arm succeeds in at most three episodes in any cell of this study, too few to support a per-success statement.

Accounting and conformance. The retry arm's extra calls are invisible to the frozen episode loop: they consume no environment step and leave no trace in the episode record, so they are reported from

Table 17: Actual use of the programmable interface. Counts among successful J4 episodes. Literal plan, iteration, and other classify the final accepted code and sum to successes. Zero oracle counts episodes with no external oracle calls across any submission; it overlaps the syntax categories.

Task	Model	Successes	Literal plan	Iteration	Other	Zero oracle
F4	Flash	220	0	220	0	0
	Qwen	149	3	146	0	4
Workflow	Flash	220	1	218	1	127
	Qwen	347	263	71	13	346

the runner's own sidecar ledger rather than inferred from the episodes. The amendment that named the policy added this ledger only; it did not change the arm's accept/reject semantics. Of the 800 retry episodes, 466 carry the full per-call ledger and 334 were launched before the amendment and carry none; for those, repair-attributable token quantities are recorded as *unknown* and never as zero, so a cell mixing the two cohorts reports a lower bound. This is why the primary cells' repair arms show no intercepted-token total in Table 15: their success and termination fields were recorded identically in both cohorts, and only the accounting is missing. Protocol conformance is reported per shard in the analysis artifact, which also records the sha256 of every input file it read (amendment document sha256 `4293f920d2112e7b...`): all 48 shards are consistent with the rules that applied at their launch, with the ledger gap confined to the pre-amendment cohort and with as-run arms outside the ledger's domain by construction, since they make no repair calls.

Naming. The condition was launched under the label `unified-repair` and named *format-retry* by the amendment that defined it; the analysis normalises both labels into one cell and marks the pre-amendment shards wherever they appear, so the rename cannot split one condition into two.

H.9 What Successful Programs Contain

We conduct a post-hoc descriptive audit of the stored final responses in all successful J4 episodes. The runner stops requesting repairs after accepting a program, so the last response contains the program whose returned plan is executed. We extract code using the runner's own function and parse its Python abstract syntax tree. A *literal plan* is exactly one assignment of an integer list or tuple to `PLAN`; *iteration* denotes a loop or comprehension; all remaining programs are classified as *other*. The three categories are mutually exclusive. All final successful programs parse; two examples per nonempty task–model–category group (or all if fewer) are selected with fixed seed 20260917 for manual inspection. This classifies syntax, not the model's internal reasoning or the correctness of a generated simulator.

Literal-plan programs demonstrate plan submission without planning computation in the final code block; they do not reveal how the model derived the list or exclude oracle use in an earlier submission. Conversely, a loop is not necessarily a search algorithm, and zero external queries do not exclude search over a simulator constructed from the supplied rules. The observed contrast supports inspecting computational behavior separately from the name of the interface.

H.10 Interface Resource Use

Table 18 reports model calls, tokens, and external oracle calls separately for the core arms. The code interface uses fewer model round trips in these comparisons, but it does not always use fewer output tokens: on F4, Qwen J4 generates more output tokens than either stepwise arm. Token counts are provider-reported usage rather than a common hardware-cost measure.

H.11 End-to-End Control Under Full Operator Definitions

The single-step probe measures what the model can *predict* about one transition; a separate registered study measures what changes *end-to-end* when the same information is delivered inside the episode. Both conditions share the interface, the action budget, the decoding contract and the parser; the only difference is whether the episode supplies the full operator definitions (B-rules) or the operator names alone (B-name). The registered design is two models $\times$ 2 $\times$ 100 environments (100+100 halves) $\times$ 2

Table 18: Resource use per episode, averaged over all 400 episodes per row, including failures. Input/output tokens are in thousands; model calls and oracle queries are counts.

Task	Model	Arm	Model calls	Input (k)	Output (k)	Oracle
F4	Flash	J1	29.13	46.67	8.04	0.00
F4	Flash	J3L	61.56	113.75	9.86	32.20
F4	Flash	J4	3.31	8.35	3.60	213.27
F4	Qwen	J1	29.55	50.19	17.25	0.00
F4	Qwen	J3L	44.72	76.94	17.71	14.99
F4	Qwen	J4	3.10	9.12	21.60	205.68
Workflow	Flash	J1	8.47	13.77	4.75	0.00
Workflow	Flash	J3L	28.98	50.83	9.40	15.88
Workflow	Flash	J4	3.24	7.34	2.72	312.61
Workflow	Qwen	J1	15.13	27.31	20.39	0.00
Workflow	Qwen	J3L	20.61	36.90	25.14	1.00
Workflow	Qwen	J4	1.85	4.68	11.23	3.07

Table 19: End-to-end control under two information conditions, 200 environments per model (100+100 registered halves). The two conditions differ only in whether the episodes supply the full operator definitions. Δ is paired by environment; the interval is the stratified environment-resampled bootstrap; p is the exact two-sided McNemar test; Holm is over the pre-registered two-model family.

Model	B-name	B-rules	Δ	95% CI	McNemar p
DeepSeek V4 Flash	5.0%	12.5%	$+7.5$ pp	$[+3.0, +12.5]$	0.0041
Qwen3.5-35B-A3B	3.0%	0.5%	-2.5 pp	$[-5.0, +0.0]$	0.125

conditions = 800 episodes; the primary quantity is the paired difference in success, by environment, with an environment-resampled bootstrap (10,000 draws, seed 42; pooled intervals resampled *within* each half), an exact two-sided McNemar test, and a Holm correction over the two-model family. Table 19 reports the result.

The intervention improves end-to-end success significantly for DeepSeek V4 Flash, but no significant change is detected for Qwen3.5-35B-A3B. Supplying the definitions raises DeepSeek V4 Flash's end-to-end success from 5.0% to 12.5% (paired Δ $+7.5$ pp, stratified 95% CI $[+3.0, +12.5]$, exact McNemar $p = 0.0041$ below the Holm threshold 0.025), with the same sign in both halves ($+7.0$ and $+8.0$). For Qwen3.5-35B-A3B the same information produces no significant change (3.0% $\rightarrow$ 0.5%, Δ -2.5 pp, $[-5.0, +0.0]$, $p = 0.125$; halves -2.0 and -3.0). We therefore report the two models separately, do not quote a pooled effect, and do not read the smaller model's point estimate as a demonstrated degradation.

The same episodes show a concurrent shift in that model's output contract: its parse-failure rate rises from 13.6% to 35.8% of responses, every one of its 200 rules episodes contains at least one parse failure, its output tokens per environment grow $4.1\times$ (18.3k $\rightarrow$ 74.9k), and its counted real actions per episode fall from 25.5 to 19.2 (DeepSeek V4 Flash: $0.7\% \rightarrow 10.9\%$; 8.2k $\rightarrow$ 15.2k; $29.2 \rightarrow 25.3$). These quantities move together inside the same episodes, so we treat output-contract friction as a *candidate* explanation rather than an established mechanism: this design does not isolate the channel. What the result does show is that the definitions were present in every episode and did not convert into control for this model under this output contract.

I Cost Summary

Reported Qwen results (hosted with the vLLM inference engine) cost ~$0.02–0.08 per episode depending on model size; total ~$60 for all measured baselines. Frontier runs (GPT-4o, Gemini 3.1 Pro, DeepSeek V4 Pro, GLM-5.2, Kimi K3) completed via application programming interface (API); per-episode cost ~$0.10–0.30 depending on model and prompt complexity, roughly $30–90 across the 300 test episodes per model.

J TEXT MODALITY CONTROL STUDY: FULL SETUP

J.1 MOTIVATION AND DESIGN

To isolate whether VLM failure on AlienBody stems from visual perception limitations or from exploration strategy deficits, we conduct a controlled text-modality experiment. The same Qwen model families (4B, 9B, 35B, 397B) are evaluated on the identical 300 test environments with all parameters held constant except the input representation.

J.2 MODELS AND INFRASTRUCTURE

We use Qwen3.5 text-only models served via vLLM (v0.17.1) on NVIDIA A800 80GB GPUs:

- **Qwen3.5-4B**: 1×A800, port 8000, `max-model-len` 32768
- **Qwen3.5-9B**: 1×A800, port 8001
- **Qwen3.5-35B-A3B** (mixture of experts, 35B total/3B active): 2×A800, tensor-parallel-size 2
- **Qwen3.5-397B-A17B-GPTQ-Int4** (mixture of experts, 397B total/17B active, 4-bit quantized): 4×A800, tensor-parallel-size 4

All models use greedy decoding (temperature 0), `max_tokens`=128, and history window of 10 turns. The 4B model uses the `-nothink` variant to avoid inflated prompt tokens from internal reasoning traces. Total inference time: ∼15 GPU-hours across all four models. Inference cost: ∼$0 (on-premise A800). These runs were executed on a separate on-premise host and their per-episode logs are not part of this release, so this condition is reported in aggregate only: its four rows appear in Table 20 but not in the per-family tables of Appendix C.

J.3 TEXT RENDERING FORMAT

The 16×16 grid is rendered as an ASCII table with row/column indices and agent position explicitly labeled:

```
Grid:  16x16  Agent:  (14,5) facing right  Actions:  0,1,2,3
0 1 2 3 4 5 6 7 8 9101112131415
--------------------------------
0 | . . . . . . . . . . . . . . .
1 | . . . . . . . . . . . . . . .
...
14 | .  .  .  .  .  > .  .  .  .  .  .  .  .  .
15 | .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
```

The agent marker (>) indicates facing direction. Prompt tokens: ∼613 initial, ∼2000–4000 after 10 turns. History window of 10 turns keeps context within the 32K limit.

J.4 PROTOCOL

The evaluation protocol is identical to the image-modality setting (Section 4): a ≤20-step calibration phase with early termination allowed, a 50-step total budget (Appendix H.5), same minimal prompt template, same per-family human baselines for CE computation. The only difference is that `observation["text"]` replaces `observation["image"]` in the agent input. Because the prompt template includes the text block whenever the observation carries it, the image condition also contains this rendering: the contrast studied here is image+text versus text, not vision versus language.

J.5 KEY RESULT

The 397B text model (CE $= 0.291$, SR $= 44.0\%$) outperforms its image counterpart (CE $= 0.081$, SR $= 15.7\%$) by a factor of $2.8\times$, and surpasses the Bayesian baseline (CE $= 0.238$). Note that the image arm is not vision alone: the prompt builder appends the text grid whenever the observation

Table 20: Image vs. text modality (Qwen3.5). Δ columns are text − image (▲ text better, ▼ image better): the switch hurts the small models and helps the large ones.

	Image			Text			Δ (text − image)	
Qwen3.5	CE	SR%	P1	CE	SR%	P1	CE	SR
4B	0.098	8.3	0.95	0.018	3.3	11.3	▼0.080	▼5.0pp
9B	0.062	7.7	8.94	0.035	1.7	2.7	▼0.027	▼6.0pp
35B	0.078	11.0	16.06	0.117	20.0	0.0	▲0.039	▲9.0pp
397B	0.081	15.7	16.64	**0.291**	**44.0**	3.5	▲0.210	▲28.3pp

carries it, so the image condition is image+text and the text condition is the same ASCII rendering without the image. The comparison therefore shows that adding the image does not help capable models here (and can hurt), not that visual capability alone is deficient. One asymmetry is not about the representation: the arms differ in interaction length, and the Phase-1 column of Table 20 shows the image arm exploring for a mean 16.64 steps at 397B (16.06 at 35B) against 3.5 (0.0) for text, while every run carries a 10-turn sliding history window. Eviction is therefore concentrated in the image arm, and part of the gap at the larger sizes may reflect interaction dynamics and window retention rather than the sensory representation alone; restricting to short-calibration episodes would be descriptive only, since Phase-1 length is itself an outcome of the policy being measured.

K FMB Training Details

This appendix details training for Forward-Model Babbling (FMB): inducing an executable $\hat{T}$, then planning with BFS over the symbolic simulator below.

K.1 Stage 1: Self-Supervised Pre-training

Data. We collect 1,200 training trajectories from the AlienBody Tier-M train split (6 families × 50 environments × 4 actions, systematic testing). Each record contains: grid image (PNG, 512×512), action index, pre-action state (position, facing direction, color), observed effect (position delta, direction change, color change), and ground-truth ActionSchema. We reserve 480 additional records from the dev split for validation.

Model and training. We fine-tune Qwen3-VL-4B-Instruct and Qwen3-VL-32B-Instruct using Low-Rank Adaptation (LoRA; rank 16, alpha 32, targeting all linear layers). Training uses the HuggingFace Trainer with DeepSpeed ZeRO-3 across 8×NVIDIA A800 (80GB), batch size 1 per GPU, learning rate 1×10^{-4} with cosine schedule, 10 epochs. The model receives the grid image and a text prompt ("Action a was executed. Initial state: … Observed effect: … What is the ActionSchema?"), and is trained to output the JSON ActionSchema via standard next-token prediction with loss computed only on assistant tokens.

Results. On F1 (Directional), both 4B and 32B achieve schema accuracy $= 1.00$ after 10 epochs (train loss 0.023–0.165). On F4 (Relational), both models achieve schema accuracy ≤ 0.29, consistent with image-based relational induction being a bottleneck at both scales, though two models cannot separate scale from architecture or from the training-data differences between them. (These Stage-1 numbers come from the training-log metric; the raw schema outputs were not archived, and the behavioral evidence in Appendix L.14 is consistent with them; no trained variant produces plan-usable F4 schemas beyond a single episode.)

K.2 Stage 2: GRPO Fine-tuning

Setup. We use Group Relative Policy Optimization (GRPO) with group size 4, KL penalty $\beta = 0.01$, learning rate 5×10^{-5}. The reward function is $R = \mathrm{Acc}(\hat{T}, T^*) + 0.3 \cdot (1 - \text{steps/budget}) + 0.2 \cdot \mathbb{I}[\text{success}]$, where $\mathrm{Acc}(\hat{T}, T^*)$ is the schema accuracy of the induced forward model $\hat{T}$ against the ground-truth T^*. Each episode runs a full AlienBody episode (Phase 1 exploration + Phase 2 BFS

Algorithm 1 ForwardModelSimulator: predict next state from induced $\hat{T}$

Require: State $s = (x, y, d, c, p, \Delta)$—position, facing, color, previous action, last movement delta—action index a, forward model $\hat{T}$, environment config (cell colors, obstacles)
Ensure: Next state s'

schema $\leftarrow \hat{T}$.actions$[a]$
if schema $=$ null **then**
  return s {unidentified action: no-op}
end if
$t \leftarrow$ schema.effect_type
if $t =$ TRANSLATE **then**
  $(dx, dy) \leftarrow$ schema.params.direction
  if schema.params.ego_centric **then**
    $(dx, dy) \leftarrow$ rotate_by_dir(dx, dy, d)
  end if
  $s' \leftarrow s$ with $(x, y) \leftarrow (x{+}dx, y{+}dy)$ if the destination is in-bounds and not an obstacle, else $s' \leftarrow s$ {blocked step: no-op}
else if $t =$ ROTATE **then**
  $s' \leftarrow (x, y, (d{+}$schema.params.rotation$) \bmod 4, c)$ {signed parameter: $+1$ for `rotate_cw`, -1 for `rotate_ccw`}
else if $t =$ STATE_CHANGE **then**
  $s' \leftarrow$ apply_color_change$(s,$ schema.params$)$
else if $t =$ RELATIONAL **then**
  ref $\leftarrow$ color(x, y) {dispatch uses the color of the cell under the agent}
  if schema.params.shape $=$ TELEPORT **then**
    $C \leftarrow \{(r, c) \neq (x, y) :$ color(r, c) satisfies the relation to ref$\}$ {nearest/farthest same- or diff-color cell; nearest brighter/darker cell}
    $(\hat{r}, \hat{c}) \leftarrow$ the element of C minimising (nearest) or maximising (farthest) $|r{-}x| + |c{-}y|$, ties by row-major order
    $s' \leftarrow (\hat{r}, \hat{c}, d, c)$ if $C \neq \emptyset$, else s {obstacles are *not* excluded from C}
  else
    $N \leftarrow$ the in-bounds, non-obstacle 4-neighbor with the extremal color (brightest, provided its color >0; or darkest) for toward/away, or the nearest same/diff-color cell for flee {single step: toward/away from the extremal neighbor; flee the nearest color match}
    $s' \leftarrow$ one unit from (x, y) toward N, or away from it (flee: along the dominant axis of the away vector); no-op if N does not exist or the destination is out of bounds / blocked
  end if
else if $t =$ COMPOSITE **then**
  $s' \leftarrow s$
  for prim **in** schema.params.primitives **do**
    apply the effect handler for prim.effect_type to s' with prim.params {stored primitive schemas applied in order, not action-table lookups}
  end for
else if $t =$ TEMPORAL **then**
  $s' \leftarrow$ apply_temporal(s, a) {effect keyed on whether $a = p$; repeat/inverse replay $\pm\Delta$; updates p, Δ}
else
  $s' \leftarrow s$ {NOOP}
end if
return s'

planning), and the VLM agent uses the Stage 1 LoRA adapter for induction. We train for 200 episodes per family on 4×A800.

Results. On F1, GRPO improves Phase 1 efficiency while maintaining schema accuracy $= 1.00$. On F4, GRPO produces negligible schema-accuracy improvement (0.03), consistent with the floor-identity result of Appendix L.14: policy-gradient feedback does not fix an emission failure.

K.3 Standard RL Baselines

We evaluate standard deep RL algorithms (PPO, DQN, ICM, and meta-RL) on AlienBody environments. All algorithms operate in the default setting (anonymous actions, no oracle information). For the standard RL baselines we tested (PPO, DQN, ICM, meta-RL) on Family 1, near-random performance obtains (SR $< 5\%$ for PPO and DQN); ICM Pathak et al. (2017) reaches 8% SR by rewarding novel states. We report these as negative results for these specific configurations, not as a general claim about model-based or system-identification RL.

L Additional Experiments and Analyses

Figure 2 (main text) reports the AlienBody name-prior swing; the subsections below give external-game setup, gameplay frames, and further ablations.

L.1 Name-Prior Statistics and Replication Details

The main-text tests count each environment once: we average the two batches within each environment and apply an exact two-sided sign test to the nonzero paired differences; this leaves the misleading-label effects significant on both families (F1 p=0.0014; F2 p=0.011) and weakens only the F2 true-label contrast to p=0.071 (21 of 50 environments favoring true labels), which the main text reports as a direction rather than an established effect. Treating the two batches as 2×50 independent pairs instead (the more optimistic counting, since each environment then enters twice) gives F2 true-label p=0.040, F2 misleading p=0.0052, and F1 misleading p=1.7×10^{-4}. The paired bootstrap CIs resample environments and do not depend on this counting distinction.

The frozen-suite replication runs Qwen3.5-9B (text, no mapping aid) on the 600-environment suite (F1–F6, 100 each) in six conditions: anonymous, true, misleading, *nonce-matched* (non-word labels of matched form), *synonym-true* (correct but non-canonical words), and *id-permuted* (labels permuted across actions). Every condition lands at the benchmark floor for this model (pooled SR 1.0–2.2%, 6–13 successes per 600) and every contrast against anonymous is null: the largest is synonym-true at $+0.50$pp (exact two-sided McNemar p=0.678), and no condition differs by more than 0.67pp (all $p \geq 0.42$; Figure 2(b)). A cross-host anonymous control on A800 (7/600 vs. 10/600) is within three environments.

L.2 Crafter Experiment Setup and Calibration Probe

Crafter Hafner (2022) exposes 17 actions (`noop`, `move_*`, `do`, `sleep`, `place_*`, `make_*`) and 22 built-in achievement counters. We re-ran the name-prior contrast under a controlled protocol: one system-prompt template, shared by all conditions, differing *only* in the label string attached to each action index; the 64×64 render is upscaled to 512×512, and all conditions share the same text status line and goal list. **named** uses the 17 true action names; **category** replaces each multi-member group with rank labels in index order (`move_1..4`, `place_1..4`, `make_1..6`) while the singleton groups keep their own name (`noop`, `do`, `sleep`); **anonymous** labels actions `action_0..action_16`. *No condition receives effect descriptions.* Every step call carries the same history block: the last 20 lines (label, position delta, reward, inventory/status/achievement deltas) plus a cumulative action tally over the whole episode, so the only manipulated variable is the label string. Each run is up to 150 VLM calls (one action per call); a response parses iff it contains a full label token or is entirely a bare index 0–16, and any failure is a recorded no-op (parse failures: 0.7% / 2.5% / 0% of steps for named / category / anonymous). Death ends the episode early. Runs use 20 paired seeds across conditions, and all reported comparisons are paired bootstrap CIs over the shared seeds. The calibration probe is asked after the episode in two arms: *post* includes the episode's own transcript (window 60) and *prior* provides no experience; both ask what each action does in the condition's own vocabulary and score each answer FULL/PARTIAL/NONE against the ground-truth effect via an LLM judge, with $\mathrm{CA} = (\mathrm{FULL} + 0.5\,\mathrm{PARTIAL})/17$.

An earlier protocol (v1) differed in two ways: its named prompt spelled out a one-line effect description for each action, and no condition received any action history. Its ordering (3.00 named / 1.60 category / 0.20–0.25 anonymous mean achievements) is the historical interface result and is not comparable to the controlled contrast: it confounds the names with prompt-supplied effect

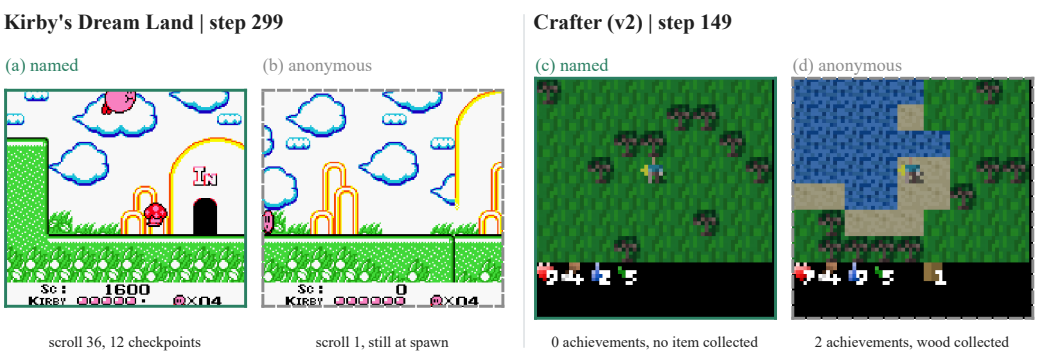


Figure 6: The name prior on screen. Four selected replay frames: Kirby at step 299 (a,b), Crafter at step 149 (c,d), seed 0 under each Crafter v2 condition. Named conditions use solid green borders; anonymous conditions use dashed gray borders. These are illustrative frames, not paired evaluation trajectories.

knowledge, and its 300-step budget differs from the 150-step cap used here. Under the controlled protocol the ordering inverts (0.10 / 2.45 / 1.85), with the Crafter score (geometric mean over the 22 achievements) following the same order: 0.004 / 0.092 / 0.074. The inverted ordering is visible in the action mix: named runs spend half their steps on `move_up` and press `do` 3 times across the 20 episodes, while anonymous runs press `do` 669 times. Label semantics account for this asymmetry: the directional `move_*` names carry a strong prior that absorbs the named agent's steps, whereas `do` is a single context-dependent primitive that harvests, attacks, or crafts depending on what lies in front of the agent, so its bare name supplies no comparable prior toward gathering; only the anonymous agent, reading its own feedback, discovers its value.

L.3 Gameplay Evidence

Figure 6 shows frames from the actual recorded trajectories (deterministically replayed from the run logs; for Crafter, the median-achievement run of each condition replayed under its own seed). In *Kirby's Dream Land*, the named agent advances two screens deep (scroll 36, 12 checkpoints; best of the three named runs) while the anonymous agent's screen at step 299 is indistinguishable from spawn (scroll 1). In *Crafter* (v2 protocol), the named agent's step-149 frame shows it in the forest, 21 tiles from spawn, with zero achievements and nothing collected, while the anonymous agent has reached the lake and collected wood (two achievements, one wood item held). The label manipulation is visible on screen, not just in metrics.

L.4 Exploration Pathology

Rather than a manual failure taxonomy, we quantify exploration quality directly from Phase-1 trajectories (Figure 7). **Premature commitment (image):** the 4B VLM acts after only $\sim$1 Phase-1 step (P1 $=$ 0.9) and tries just 24% of actions (EC $=$ 0.24); yet when it succeeds it is efficient (CE_{cond}=1.18). **Redundant probing (image):** the 35B and 397B VLMs consume 16–17 Phase-1 steps, and the trajectory-only CA proxy credits them with only 61–64% of action effects. **Text-modality divergence:** text input inverts these patterns: the 4B text model explores *more* (P1 $=$ 11.3 vs. 0.9) but succeeds less, while the 397B text model explores *less* (P1 $=$ 3.5 vs. 16.6) yet succeeds far more; the 35B text model terminates Phase 1 immediately (P1 $=$ 0.0) in most episodes yet still achieves SR $=$ 20.0% through Phase 2 trial-and-error, behaving more like a reactive policy than an exploration agent. Scripted agents avoid both pathologies, testing each action exactly once (P1 $=$ 4) to reach CA-proxy $=$ 0.78.

L.5 Test-Time Scaling

Best-of-8 sampling lifts SR from 8.3% to 14.9% (Table 21), a real inference-time gain that still trails Bayesian (22.7%). Neither prompt-level intervention recovers performance. Constrained action selection is an exact wash on the complete runs (25/300 successes either way), which is the prediction

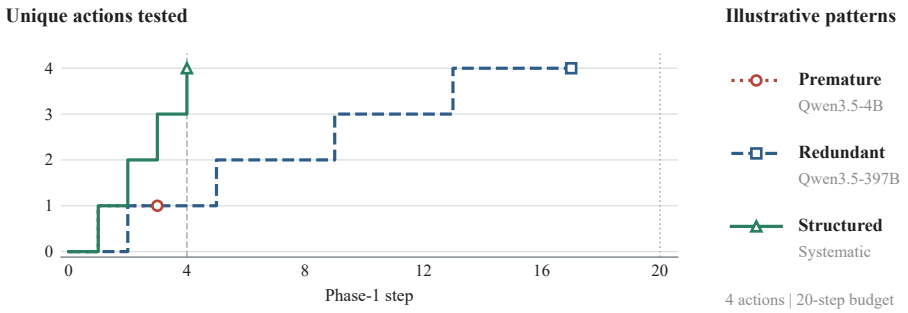


Figure 7: Three exploration patterns on AlienBody (Family 1, Tier-M). Schematic reconstructions from aggregate exploration metrics, not measured episode traces; curves illustrate unique actions tested over Phase 1 steps. *Premature* (Qwen3.5-4B): commits after $\sim$1 step, discovers 1 action (EC $=$ 0.24). *Redundant* (Qwen3.5-397B): tests 85% of actions but re-tests each action $\sim 4\times$, consuming most of the budget. *Structured* (Systematic): discovers all 4 actions in exactly 4 steps (optimal exploration).

Table 21: Test-time methods (Qwen3.5-4B, Tier-M). ΔSR is relative to the plain model, restricted to runs with complete 300-episode coverage: Best-of-N helps; demonstrations and constraints do not.

Method	SR%	CE	ΔSR
Qwen3.5-4B (plain)	8.3	0.098	—
+ Best-of-8 (BoN)†	**14.9**	**0.099**	†
+ Expert demo	4.3	0.042	▼ 4.0pp
+ Constrained	8.3	0.065	0.0pp

†The Best-of-8 run was interrupted at 228/300 episodes with an unbalanced family mix (Family 1 over-covered at 66 records; Families 4 and 5 at 4 and 8), so its rate is reported but not differenced against the complete rows. On the common valid set (the three families in which all four configurations have complete 50-episode coverage: F2/F3/F6, n=150 each), SR is plain 4.0%, BoN 12.0%, expert demo 2.0%, constrained 6.0%; the ordering is unchanged.

if the size of the action space is not what limits these agents. The demonstration arm varies both the informational content and the response format at once and is smaller than the effect it would need to detect, so we read the two together as consistent with exploration strategy as the bottleneck, not as a test that isolates it.

The demonstration arm's prompt is a fixed four-step worked example drawn from a *different* environment, showing one-test-per-action calibration followed by an early stop; it states in the prompt that the actual action mapping differs and must be discovered. Action mappings are shuffled per environment at generation time, so the example's own assignment (Action 0 moves right, 1 down, 2 left, 3 up, in row–column coordinates) is not the mapping of an evaluated environment: the demonstration conveys the procedural pattern, not the mapping itself. No per-episode relabelling is applied, so the arm measures whether a generic exploration scaffold transfers, not whether the correct mapping, if demonstrated, would be used.

L.6 Familiarity Effect

Does knowing *what kind* of actions exist help? We compare two conditions on Family 1 (Adapt): *Familiar*, where the system prompt reveals that actions are scrambled cardinal directions, and *Normal*, which provides no such hint (Qwen3-VL-4B-Instruct, image-modality, Tier-M test, 50 env/condition; Table 22).

The familiarity prior yields a modest CE gain of $+0.010$ ($+2\%$ SR). Both conditions achieve perfect coverage (EC $=$ 1.00) and a saturated CA-proxy (CA-proxy $=$ 1.00: identical systematic sweeps yield identical proxy scores) on the simplest family, yet consume the full Phase 1 budget (P1 $=$ 20.0), four times the optimal 4 steps, confirming that the model performs redundant probing even when

Table 22: Familiarity effect on Family 1 (Adapt). Category hints help only modestly; the trajectory-only CA-proxy saturates for both conditions.

Condition	CE	SR%	EC	CA-proxy	P1
Familiar (directions)	0.058	14.0	1.00	1.00	20.0
Normal (no prior)	0.048	12.0	1.00	1.00	20.0
Δ (Familiar − Normal)	▲ 0.010	▲ 2.0pp	0	0	0

the action categories are supplied through familiar labels, whose exact mappings still have to be identified from interaction. We also tested Qwen3.5-9B in text modality under the same conditions: both achieved SR $= 0.0\%$ with identical metrics (EC $= 0.34$, CA $= 0.34$), indicating that for models with near-zero baseline capability, providing action-space priors has no measurable effect.

L.7 Tier-L: The Wall at Scale

We instantiate Tier-L (Table 8) for the Relational family: 24×24 grids, n=6 relational actions (two additional relation types: farthest-same-color teleport and anti-brightest step), 5–8 obstacles, 25-step Phase-1 and 80-step total budgets, 50 test environments, all BFS-validated. The wall persists at scale: GPT-4o with the exact action mapping (L3, text) scores **2.0%** (1/50) on Tier-L F4, while Oracle BFS solves all 50 (100%). Tier-L is strictly harder for every method: FM_{heur}'s induction drops from 88% (Tier-M) to **30.0%** (CA-proxy 0.25); with 6 actions and a 25-step calibration budget, the 6!=720 permutation enumeration is under-determined, yet still far above the trained variants' Tier-M 2%. Tier-L thus separates the scale at which induction degrades (heuristic: 88%→30%) from the planning deficit itself, which neither grid size nor action count moves.

L.8 Tier-XL: Scale Degrades Identification, Not Enumeration

We instantiate Tier-XL (Table 8) for the Relational family: 32×32 grids, n=12 relational actions (six additional relation types: farthest-diff-color teleport, flee-nearest-diff, toward/away-darkest, nearest-brighter/darker), 10–15 obstacles, 40-step Phase-1 and 120-step total budgets, 50 test environments, all BFS-validated. The $12! \approx 4.8\times10^8$ candidate mappings are not a barrier here: for the Relational family the replay score is a sum of per-transition potentials that depend only on the assigned pair, $\text{score}(\pi) = \sum_a W[a, \pi(a)]$, so the optimal mapping is a linear assignment problem solved exactly in $O(n^3)$ (Hungarian) with no enumeration. `check_matching_vs_enum.py` verifies that the two solvers return identical optima *and* identical tie-set cardinalities at n=4/6/8 (decomposition deviation $\leq 2\times10^{-15}$), and that at n=12 the assignment optimum is not beaten by 800 sampled permutations. Results: Oracle BFS solves all 50 (100%); the enumerator-permission arm with the exact-assignment solver (Enum+Match) reaches **56.0%** (28/50). The previously reported 0% was an artifact of the $n\leq8$ enumeration guard, not of the regime: the shipped FM_{heur}, which still falls back to positional-delta heuristics above n=8, remains at **0.0%**. The LLM proposer is unchanged: FM_{ind} (GPT-4o, k=4 rounds, fallback off) scores **0.0%** in a 10-environment pilot with mean CA-proxy 0.075.

Exact optimization is not identification. We also run an observation-only matching baseline under the same permission contract as the Bayesian and Systematic baselines: it builds W only from transitions it has itself observed, then solves the same assignment problem. The two arms differ in what may enter W: both score each candidate pair by the transitions it explains, but Enum+Match may replay a candidate mapping through the environment's effect primitives at every position visited in Phase 1 (config-level replay scoring), whereas the observation-only arm may reuse only the (s, a, s') triples it actually executed; an action row never exercised contributes nothing to its W and stays freely permutable. It reaches **22.0%** (11/50) at n=12, and **36.0%** (18/50) with a structured round-robin Phase-1. Its induction log is diagnostic: the observation-only objective admits a mean of 52,480.6 optimal assignments per episode (max 1.13×10^6), and the true mapping is not even optimal under that objective in 23/50 episodes. We audited both statistics against the true dynamics. Replayed with the exact transition function, the truth is optimal in all 50 episodes: the rejections trace to the surrogate's prediction rule, which falls back to predicting a teleport to the nearest cell of unknown color (in Tier-XL, always an obstacle) when no color-qualified cell exists, where the true action is a no-op; removing that single fallback, an information-free change, reduces rejections

to $3/50$ and raises the mean achieved score by 16.0 points, the residual three being near-ties broken differently. What the observations genuinely fail to do is single the truth out: replayed exactly, it is the *unique* optimum in only $11/50$ episodes at $n{=}12$. The large optimal-assignment counts are likewise coverage-bound (unexercised actions are freely permutable, contributing a factorial term), and the round-robin arm that exercises every action collapses the mean count to 3.9. At scale the binding constraint is therefore *identification*—what the agent's own observations can pin down, not the cost of searching; the enumerator sidesteps it only through config-level replay scoring that the observation-only baselines are not permitted. The same agent's SR falls $30/50 \to 26/50 \to 11/50$ as n grows $4 \to 8 \to 12$, so the degradation tracks n, not grid size.

L.9 The n=8 Intermediate Regime: Ties, Not Collapse

To locate *where* passive enumeration is incomplete, we instantiate an intermediate regime: $32{\times}32$ grids, $n{=}8$ relational actions (the first eight of the Tier-XL vocabulary), 40-step Phase-1 and 120-step budgets, 50 BFS-validated environments (sparse color layout). At $n{=}8$ the permutation space ($8!{=}40{,}320$) is still cheaply enumerable. Oracle BFS solves all 50 (100%). Corrected Enumerate+Verify (which actually searches the type-C hypothesis space) reaches **90.0%** (45/50): the five failures all achieve a perfect trajectory replay score yet leave a non-singleton tie set $H^\star$ (sizes $2, 2, 4, 10, 12$ across the five), so the inducer must break ties by chance. Across the split, $18/50$ episodes end with $|H^\star|{>}1$; passive enumeration happens to pick the true mapping on 13 of them. Thus $n{=}8$ is not "enumeration dies" but "enumeration is mostly solved, with a residual under-identification mass."

Active Forward-Model Babbling (AFMB) closes that mass: after a short systematic warm-start it maintains $H^\star$, selects actions that maximize predicted-outcome diversity under the tie set (or preparatory moves toward rarely visited cells), and early-stops when $|H^\star|{=}1$. On the same 50 environments AFMB reaches **100%** SR with mean Phase-1 length 17.6 (vs. 40 for passive enum) and higher calibration efficiency; the success gain over Enumerate+Verify is $+10$pp (paired env bootstrap 95% CI $[+2, +20]$, $10{,}000$ resamples, seed 42; exact two-sided McNemar $p{=}0.0625$ over five discordant pairs, all in one direction, which is the minimum attainable at this discordance count: against a 90% baseline at $n{=}50$ at most five pairs can ever be discordant, so the exact two-sided test cannot reach $\alpha{=}0.05$ here even at a 100% rescue rate. We therefore read the gain as a directional trend and take the paired bootstrap interval as the substantive evidence), and all five rescued environments are perfect-replay ties. Ablations isolate the active ingredient: warm-start-only AFMB scores 90% (P1=16), systematic passive AFMB (full budget, no expected information gain / EIG) scores 92%, and EIG-disabled AFMB (early-stop on unique $H^\star$ only) also 92% (matching the passive-enum ceiling), while full AFMB reaches 100% at P1=17.6. This is consistent with the extra points coming from discriminating experiments rather than from spending more Phase-1 steps.

The LLM proposer remains weak in this regime: $\mathrm{FM}_{\mathrm{ind}}$ (GPT-4o, 50 environments) scores **2.0%** at zero-shot and after $k{=}4$ verification rounds; DeepSeek V4 Pro (text, same protocol, full 50) scores **6.0%** zero-shot and **4.0%** with $k{=}4$ rounds; verification does not help, matching GPT-4o's flat pattern. Combined with Tier-XL ($n{=}12$: exact-assignment search still reaches 56% while every LLM proposer tested stays at 0–2%), the value curve is: at $n{=}4$ brute-force enumeration dominates LLMs; at $n{=}8$ passive enum is strong but leaves residual ties that *active* experiment design removes; at $n{=}12$ search still leads by a wide margin and the open frontier is the *proposer*, not the search.

On standard Tier-M ($n{=}4$), the same AFMB policy does not hurt where Enumerate is already saturated (F4: 100% vs. 98%; F5: both 100%, with AFMB cutting mean Phase-1 length from 18.3 to 9.8) and still improves where passive enum under-identifies (F6: **100%** vs. 90%). Thus the active ingredient generalizes beyond the $n{=}8$ stress test: wherever $|H^\star|{>}1$ after passive observation, discriminating experiments buy success rate or calibration efficiency.

Figure 4: point provenance and paired intervals. The plotted proposer points are F4 (GPT-4o $2/47/94$, DeepSeek V4 Pro $8/40/94$), F5 ($26/65/62$ and $30/85/80$), and F6 ($32/34/8$ and $6/44/10$) at zero-shot, +verification, and +fallback; the reference rules are Enumerate+Verify ($98/100/90$) and AFMB ($100/100/100$). Every archived run covers the same 50-environment split, and the three +verification points of F4/GPT-4o and of both F5 proposers pool two archived runs of that split (F4/GPT-4o $23{+}24$, F5 $32{+}33$ and $43{+}42$), so those three points rest on 100 episodes; the F4/DeepSeek V4 Pro +verification point is a single run, as no second one exists. The six F6 points

Table 23: Plan-over-model on Tier-M. Family-wise and pooled results.

Method	Metric	F1	F2	F3	**F4**	F5	F6	**Pooled**
Best VLM-L3	SR (%)	100	100	100	10	100	38	**74.7**
FM_{heur}	SR (%)	100	92	80	**88**	0	4	**60.7**
	CA-proxy	1.00	1.00	1.00	0.27	0.98	0.50	0.79
Oracle BFS	SR (%)	100	100	100	100	100	100	**100**

FM_{heur}: heuristic schema induction + BFS; CA-proxy uses a 0–1 scale.
Best VLM-L3: best of GPT-4o and Gemini 3.1 Pro at L3 (F4: 0% / 10%).

are not read off the archived F6 runs: they are fixed-simulator replays of the re-run set whose buggy-simulator control reproduces the stored runs episode for episode (Appendix L.14). Environment-clustered paired bootstrap (10,000 resamples, seed 42; the environment is the resampling unit) for zero-shot→verification gives F4/GPT-4o $+45.0$ pp $[+33, +57]$, F4/DeepSeek V4 Pro $+32.0$ $[+20, +46]$, F5/GPT-4o $+39.0$ $[+26, +53]$, F5/DeepSeek V4 Pro $+55.0$ $[+41, +68]$, and F6/DeepSeek V4 Pro $+38.0$ $[+24, +52]$, while F6/GPT-4o is flat ($+2.0$ pp $[-12, +16]$; seven environments improve and six worsen). The +fallback stage keeps an unverified proposal only when it explains at least 80% of the observed transitions, and otherwise hands planning to the heuristic induction (permutation enumeration on Relational, pattern matching elsewhere); the retention path fired in only $3/50$ and $2/50$ Temporal episodes (GPT-4o and DeepSeek V4 Pro), so this stage sits close to "LLM when fully verified, otherwise heuristic." The handover fired in $27/50$ and $30/50$ Relational, $19/50$ and $10/50$ Compositional, and $39/50$ and $43/50$ Temporal episodes, which is why this stage recovers Relational to 94% under both proposers (the family where the heuristic is strong) yet costs Temporal 24.0 pp relative to zero-shot under GPT-4o ($[-40, -8]$), where the heuristic is weak.

L.10 Tier-M Plan-over-Model: Full Table

Table 23 reports the complete Tier-M family breakdown behind the plan-over-model comparison of Section 5 (FM_{heur}'s per-family SR and CA-proxy, the best VLM-L3 row, and the Oracle BFS upper bound).

L.11 The Wall Is Not Degenerate Behavior (Artifact Defense)

A reviewer may suspect that the 0–10% F4 results reflect generic model collapse (e.g., an agent stuck repeating one action) rather than a relational-specific planning failure. We test this directly on the strongest L3 model (Gemini 3.1 Pro, image+text) by measuring, per family, the diversity of Phase-2 actions and the agreement between the model's chosen actions and the oracle BFS plan, the plan the model *could* compute, since it holds the exact action mapping (Figure 5). On F4 the model's action diversity is the *highest* of any family (entropy 0.78 vs. 0.36–0.43 on F1–F3): it actively cycles through relational actions whose outcomes shift with the agent's position, but reaches the target on only 5 of 50 environments; oracle-plan agreement collapses from 81–85% on F1–F3 to 26% on F4 (and 27% on F6, the other hard family). A local Qwen3-VL-32B shows the opposite F4 signature (entropy 0.08, degenerate repetition of a single action) while still solving F1 at 86% and F3 at 82%. The failure thus has no single behavioral signature: low Relational success occurs under different action-selection patterns, so the wall is not an artifact of a single collapse mode. Nor is the deficit closed by supervised induction: the trained variants of Table 24 land on the no-model floor, while ground-truth schema injection into the same planner reaches 100% (two of those rows carry evaluation-side defects and contribute no evidence either way; Appendix L.14).

L.12 Additional L3 Anchors (Local Open Models)

As a local stack under the same L3 protocol we evaluate Qwen3.5-4B, which scores **0.0%** in both text and image modalities. This anchor sits inside the same 0–10% band as the main-text wall; we do not yet claim a complete open-model scale sweep (mid/large Qwen3.5, Best-of-N, thinking), and leave that matrix for an extended version once every cell is run under one harness.

Reasoning-token and chain-of-thought probes. Three further configurations stay inside the same band on F4 under the same L3 contract: effect names inside the state, no operator definitions, no

oracle, and no episode history: a chain-of-thought-prompted GPT-4o run (1/50), a thinking-token variant of Qwen3-VL-32B on image input (0/20), and a DeepSeek V4 Pro reasoning endpoint (0/50). Each is a single small probe rather than a rate estimate; GPT-5.1's starred F4-only row of Figure 3 (2/50) was run under the same L3 contract.

Planning-wall provenance. Gemini 3.1 Pro's F4 and F6 cells in Figure 3 are per-environment first-attempt rates recomputed from the complete run log, because that run was restarted mid-way and its own summary covers only the resumed segment: F4 $5/50 = 10\%$ and F6 $19/50 = 38\%$. Weighting every recorded episode instead (each environment in proportion to its number of attempts) gives 4.9% (8/163) and 27.4% (45/164) on the same log; we report the per-environment rates because they are the only reading strictly comparable with the single-rollout rows of the other models.

L.13 Single-Step Transition Prediction (R-Probe)

The L3 agents of Figure 3 receive the action mapping as names inside the state, with no effect glosses, no transition oracle, and no episode history. A planning deficit would be uninformative if the model could not compute even one transition from that same information, so we measure the atomic operation directly, separately from planning.

Protocol, fixed before any call. Instances are the first 100 environments of `nextstep_f4_formal` in `env_id` order. For each we take two states (the episode's initial state and the state after exactly one application of action 0) and ask, in four independent calls per state (one per action), for the agent's position after that action is applied once. Each call carries no probe history and no answer feedback, so no call can be informed by another; the system prompt states that the mapping and target in STATE are exact, matching the L3 condition. Ground truth is computed with `apply_action` on the real state, never through the oracle, so no budget is consumed and no frozen artifact is modified. Budget: 800 calls per arm per model at temperature 1.0 (3,200 in total), matching the frozen next-step decoding block. Both models run the identical instances, prompts, and caps: DeepSeek V4 Flash over the API and Qwen3.5-35B-A3B served locally, the two families of the matched study. All intervals below are environment-clustered bootstraps over the 100 environments (10,000 draws), as in the registered analysis. An earlier single-model draw of the same protocol (identical instrument bytes, prompts, and decoding settings) gave 17.9% and 52.4% for DeepSeek V4 Flash; the two-model run reported here supersedes it.

Result (names arm). Exact-position accuracy is **18.6%** (149/800), environment-clustered 95% bootstrap CI $[16.0, 21.4]$, for DeepSeek V4 Flash and **9.0%** (72/800), $[7.1, 11.0]$, for the local Qwen model. Parsing succeeded on every call of this arm for both models, with no truncations and no transport errors, so the failures are wrong answers rather than format failures. Both rates are at or below the level of blind guessing: 91.1% of true successors lie within Manhattan distance 1 of the current cell (83.5% at distance exactly 1), so a strategy that picks uniformly among the four neighboring cells scores **20.9%**, above both point estimates. Predicted directions barely track the action: for DeepSeek V4 Flash the modal predicted displacement is $(0, -1)$ for three of the four actions and $(0, +1)$ for the fourth, while the true successor moves $(-1, 0)$ in the modal case for every action (31–43% of the 739 calls in which the agent does not stay put), and the model places 19.8% of its mass on "no move" against a true no-move rate of 7.6%; the local model places 13.5%. Per environment, 20 of 100 environments have none of their eight predictions correct for DeepSeek V4 Flash (46 of 100 for the local model), none has more than five (three), and no environment of either model is correct on all eight.

Paired arm: supplying the operators. The mapping the L3 agents receive names each effect but never defines it: `move_to_nearest_diff_color` does not say which distance, which tie-break, or whether obstacles count. We therefore re-ran the identical protocol with one change: the system prompt additionally carries prose definitions of the four operators used by this suite (nearest by Manhattan distance with row-major ties, own cell excluded, obstacles not excluded from teleport targets; `flee_same_color` stepping one cell along the dominant axis away from the nearest same-color cell, blocked by obstacles and by the grid edge; `move_toward_brightest` taking the first strictly brightest of the four in-bounds, non-obstacle neighbors in the order up, right, down, left, and moving only if that value exceeds zero). A from-scratch reimplementation of those definitions, importing nothing from the environment, reproduces the ground truth on all 800 paired calls, so the text describes the environment as implemented. States, questions, action ids, model, temperature,

and caps are byte-identical across the two arms; only the system prompt differs. Accuracy rises from 18.6% to **52.6%** (421/800; $[48.8, 56.4]$) for DeepSeek V4 Flash, a paired difference of $+34.0$ points ($[29.3, 38.8]$), with 83 of 100 environments improving, 3 worsening, and 14 unchanged (one-sided sign test $p = 1.4 \times 10^{-21}$), and from 9.0% to **46.8%** (374/800; $[43.4, 50.3]$) for the local model, $+37.8$ points ($[33.5, 41.8]$), with 95 improving, 3 worsening, and 2 unchanged ($p = 5.0 \times 10^{-25}$); the local model's parse rate on this arm is 91.8%, and unparsed calls count as misses. The two predictions are therefore not the same quantity: a per-step deficit under a names-only mapping is substantially a *semantic* deficit, and this is the information the names-only arms of this paper receive (the registered B-rules arm and both C interfaces instead supply the full operator definitions; Appendices H.11 and H.8). What does not change is the ceiling: the no-move rate becomes well calibrated for DeepSeek V4 Flash (8.8% against a true 7.6%; the local model stays at 15.5%), but 40.9% of DeepSeek V4 Flash's errors are off by two or more cells (36.5% for the local model), and the per-operator rates remain uneven: `move_toward_brightest`, which needs only the four adjacent cells, reaches 71.5% for DeepSeek V4 Flash and 55.5% for the local model, while the two nearest-color teleports, which require a global row-major argmin over a 16×16 grid, reach 53.0% and 44.0% (49.5% and 37.0%), with `flee_same_color` at 42.0% (45.0%). A model that reasons correctly about a fully specified operator about half the time still cannot serve as the transition function its own planner iterates.

Greedy-decoding control. Those rates could in principle be an artifact of sampling, so we re-ran the identical protocol (same instances, prompts, parser, caps, and instrument bytes) with one change, a request for temperature 0.0 (1,600 further calls, DeepSeek V4 Flash only; one per fresh context, so no control loop is involved and the setting cannot make a loop degenerate). The fully specified arm then reproduces the registered aggregate to the call: **52.6%** either way (421/800; $[48.8, 56.5]$ against $[48.8, 56.4]$ at temperature 1.0; paired $\Delta = +0.0$ points, $[-4.0, +4.0]$), and the names arm does not improve (18.6% $\rightarrow$ **15.4%**; paired $[-6.1, -0.4]$). The aggregate is therefore not sampling noise. But the individual answers are not stable either: only 264 of the 800 keys are answered correctly at both settings, 157 flipping to correct and 157 to incorrect. One caveat is recorded with the run: the serving provider does not execute strict argmax decoding (eight byte-identical requests at temperature 0 returned four distinct replies, three of them distinct answers), so this arm is the closest setting the instrument accepts rather than a guaranteed deterministic read-out, and what it establishes is the paired contrast against the registered run, not the nominal temperature by itself.

Decomposing the residual error. We then drew five independent answers per key at the registered temperature (8,000 calls, 4,000 per arm, DeepSeek V4 Flash; identical prompts, fresh context and no feedback each). In the fully specified arm, majority vote over the five draws reaches **65.5%** ($[61.8, 69.4]$; single draw in this run: 51.0%), a paired $+12.9$ points over the registered single draw ($[+8.5, +17.4]$), and 92.9% of keys have at least one correct draw; but only 10.1% of keys are answered identically in all five draws. The errors separate into two kinds: of the 276 keys whose majority is wrong, only 10 are unanimous on a single wrong answer, so the dominant residual is disagreement between draws rather than a confidently repeated miscalculation. An exploratory response-length split accompanies this error pattern (diagnostic cut at 200 output tokens, chosen post hoc): calls answered with a short direct reply are correct 30.2% (temperature 0) and 22.3% (temperature 1.0) of the time, while calls that emit an in-content reasoning trace are correct 59.2% and 66.9%, and the names arm rarely enters that mode (at temperature 0, 769 of its 800 calls stay under 200 output tokens, against 182 of 800 in the fully specified arm). The same aggregation does not repair the names arm: its own five-draw majority reaches 18.4% ($[15.6, 21.2]$; paired -0.3 points against the registered draw, $[-3.0, +2.2]$), and its draws are no less consistent than the fully specified ones (11.5% of keys identical in all five draws, against 10.1% here), only more consistently wrong (11.0% of its majority-wrong keys unanimous, against 3.6%). Repeated sampling does not substitute for the operator definitions. Aggregation thus recovers about a quarter of the single-draw error once the operators are fully specified, but a planner that consumes each prediction once, unaggregated, cannot collect it; and the wall of Figure 3 sits far below even the aggregated rate.

Reading. With exactly the information the L3 agents receive, this model has no usable single-step transition function: it performs at or below chance on the atomic operation that any planner must iterate. Two things follow. First, the per-step deficit has both a semantic and a computational component: the model does not know the operators from their names, and even when told, it evaluates them correctly only about half the time. Neither "the agent was never told what its actions do" nor "the agent cannot compute a transition it understands" accounts for the wall on its own.

Second, this does not reduce the wall to a naming problem: the conversational arm J3L obtains transitions from a verified oracle instead of evaluating them, so operator semantics are moot there, and it still scores 5.25% and 1.50%. What separates J3L from the programmable arm is control over how lookups are issued and composed, which is what bounds realized lookup throughput: the conversational arm obtains transitions one at a time, each through a fresh conversational turn, whereas the programmable arm schedules its own queries inside submitted programs, where transitions can be looped over, composed, and repaired at the interface seam (Section 4.4; Appendix H.8). We scope these rates to the two matched-study families probed here (DeepSeek V4 Flash and a locally served Qwen3.5-35B-A3B) and do not claim the same rate for the frontier image models of Figure 3.

L.14 Induction Frontier: Reproduction and Floor Separation

A natural objection to Table 24 is that the 2.0% floor reflects a broken trainer rather than an unlearnable relational schema. We cannot exclude a training defect from these archived runs; what we do here is diagnose their outputs (with a bit-reproduction, a floor-identity test, and a ground-truth injection ceiling) and localize where the failure occurs.

Bit-reproduction under fixed seed. We freeze a one-shot reproduction script (`reproduce_induction_frontier.sh`) with global seed 42 and retrain+reevaluate the LoRA (200/500/1000/2000), CoT LoRA, Full FT 4B, and multi-obs LoRA variants. All fourteen test/secret points match the reported numbers exactly: test SR 2.0% / CA-proxy 0.28 and secret SR 3.3%, with the sole successful environment again `family4_test_006` (the degenerate column-aligned layout). The reproduction is also behaviorally exact: all 50 test episodes are action-for-action identical to a zero-information agent, and the sole success is the environment the no-model fallback also solves. The floor is therefore not a one-off training draw, nor is it evidence of a partial capability.

Floor identity and ground-truth injection. Across the six F4 trained rows, executed behavior is identical to a model-free greedy fallback on 45–50 of 50 episodes (the failure is at schema *emission*: those episodes never run a plan), and each row's success set equals the fallback's. Injecting the ground-truth schema into the same downstream planner yields **100%** (50/50 test, 30/30 secret), with the injected model reproducing the environment dynamics at 1.000; a structurally valid but content-permuted schema drops to 2.0% (the floor). The downstream stack is therefore healthy and content-sensitive, and the 2.0% of the trained rows measures schema emission, not planning.

F1 positive control. Using the same LoRA recipe on Qwen3.5-9B (seed 42) but training on 400 Family 1 trajectories yields SR **12.0%** on the F1 test split; the checkpoint emits a plan-usable schema in only 4/50 episodes, and in all four the plan succeeds (4/4), so the failure is at emission frequency rather than schema content. Evaluating that same checkpoint on F4 returns SR **2.0%**, matching the F4 floor. This control does *not* by itself validate the training procedure: 46/50 of its episodes are action-for-action identical to the model-free agent, and 12.0% is below the **100%** of a target-directed greedy policy that uses no learned model, and the ground-truth (GT) injection ceiling above validates the downstream simulator and planner under a supplied correct schema; neither control establishes that the training procedure itself is sound.

Deterministic baseline regression. As a release smoke test we also re-run the non-learned search agents on the frozen splits: FM_{heur} F4 88%, AFMB F4 100%, Enumerate+Verify / AFMB at $n{=}8$ 90% / 100%, and Enumerate+Verify / AFMB on Tier-M F6 90% / 100%, all bit-matching the numbers cited in Section 5.

L.15 Temporal Simulator Correction

The temporal (F6) handler of the ForwardModelSimulator was corrected twice. First, it originally compared the previous action only against a non-negative sentinel (treating any history as "same action") and never propagated `last_move_delta`, so `repeat`/`inverse` effects were unreachable in planning; the BFS visited-state keys also omitted `last_move_delta`, silently collapsing temporal states. We fixed the handler to compare against the current action and to propagate the delta, and extended both BFS implementations (oracle agent and FM_{heur}/FM_{ind} planner) with the full temporal

state. Second, a later fix (`0cf26b2`, *temporal-history fallback*) found that a `repeat`/`inverse` schema with no usable history still fell through to a default cardinal move, whereas the environment applies a no-op; the simulator now returns that no-op. All F6 numbers reported in this paper—FM_{heur} (4%, unchanged), the figure's LLM-proposer points (zero-shot, verification, and fallback variants; Z1/Z3/Z4 in the legend), Enumerate+Verify (90%), and AFMB (100%)—are recomputed from the twice-corrected code; the Oracle agent's F6 rises from the previously reported 86% to 100%, matching the BFS upper bound. F4/F5 results are unaffected (their handlers were untouched; the extra visited-state field is constant for non-temporal families).

L.16 Verification-Filtered Expert Iteration (ReST)

A natural question is whether the verification signal of FM_{ind} can be *internalized* by training. We run verification-filtered expert iteration (Reinforced Self-Training / ReST-style Gulcehre et al. (2023)): a Qwen3.5-4B proposer (LoRA) samples mappings for 120 environments per round, we keep proposals with verification score $\geq \tau$ (curriculum τ=0.3→0.8), supervised fine-tuning (SFT) on the accepted pairs, and repeat, after a one-time bootstrap SFT on ground-truth mappings. The untrained 4B proposer scores 0.14 mean verification agreement on the test split (its zero-shot proposals are *worse than random* assignment, reflecting the same inductive blindness as the trained variants of Table 24). One round of verification-filtered self-training lifts the test score to 0.196 ($+41\%$) and produces its first fully-verified mappings (3%); subsequent rounds degrade as rising acceptance thresholds shrink the training set (2–8 accepted pairs per round) and the policy collapses. The pattern is consistent with the loop *multiplying* the proposer's base capability rather than substituting for it: frontier models gain 20–40pp of success in the same pipeline, while the 4B proposer gains a fraction of a point. We read verification-guided induction as complementing, rather than replacing, model capacity. The caveat is that this contrast pairs a single 4B self-training run against previously reported frontier numbers rather than a controlled sweep across scales, so we state it as an interpretation, not a measured scaling law. We also evaluate GRPO fine-tuning (Stage 2) on F1, where the trained model learns to achieve the same induction accuracy with fewer exploration steps, improving Phase 1 efficiency; full details in Appendix K.

L.17 FMB Fairness Audit

FM_{heur}'s F4 induction enumerates the $4! = 24$ permutations of the four known relation types, and scores each candidate mapping by simulating the relation effects with the environment's effect primitives at every observed position, comparing predicted outcomes against the agent's own Phase-1 observations. It never receives the ground-truth action mapping; its inputs are the same observations available to every VLM we evaluate, plus two structural priors: (i) the vocabulary of relation types, and (ii) access to a forward-model simulator for verifying candidate hypotheses (and, unlike image agents, direct access to the grid's cell-color matrix). By contrast, the L3 setting hands the VLM the mapping outright: the action names and which id carries which name, but not the definitions behind them, and the model does not exploit it. The two settings do not nest: FM_{heur} has the operator semantics and a forward model but not the mapping, while L3 has the mapping but no verified operator semantics. A one-line reading of the contrast as "FM_{heur} had strictly more information" is therefore not available, and the comparison does not single out one missing ingredient. These results motivate external search, while the structural priors and representation differences limit causal attribution: FM_{heur} succeeds because it *searches* over a small structured hypothesis space and verifies candidates against a forward model, a computation not reliably realized by the tested in-context agents. We also stress that FM_{heur} is not a general solution: it fails on composite (F5) and temporal (F6) families, and its pooled SR (60.7%) is *below* the best VLM-L3 (74.7%), which dominates on those families. FM_{heur}'s role is an existence proof: F4 is solvable by plan-over-model given the relation vocabulary and a search procedure that iterates it.

Enumerate+Verify and AFMB inherit the same priors and the same non-use of the ground-truth `action_mapping`: candidate mappings are written into temporary configs for replay scoring; the true mapping is never read during induction. AFMB adds only an active Phase-1 policy over that verified search.

M SCHEMA INDUCTION UNDER TRAINING

Can a VLM *learn* to induce relational schemas? Across the trained protocols of Table 24: Low-Rank Adaptation / LoRA on 4B–27B text, CoT, and a 32B image variant, 200–2,700 examples, excluding two runs whose evaluation-side defect is marked ♯ and which contribute no evidence either way: no variant beats the *no-model floor* on F4: their executed behavior is action-for-action identical to a model-free greedy fallback on 45–50 of 50 episodes (50/50 for the seed-42 reproduction, action-for-action), and their success set equals the fallback's, the single degenerately column-aligned environment `family4_test_006`. The reported SR of 2.0% equals Random because it *is* the floor, not a partial frontier. The bottleneck is schema emission, not the downstream stack: injecting the ground-truth schema into the same planner yields **100%** on F4 (50/50 test, 30/30 secret; the GT model reproduces the environment dynamics at 1.000), while a structurally valid but content-permuted schema returns to the 2.0% floor, so the 2% is not "the planner cannot use a model." A same-recipe F1 positive control emits a plan-usable schema in only 4/50 episodes, but in all four the plan succeeds (4/4), so the failure there is emission frequency rather than schema content, and its 12.0% SR sits below the **100%** of a target-directed greedy policy that uses no learned model; F1 thus likewise does not validate the training procedure, and the GT-injection ceiling above validates the downstream simulator and planner under a supplied correct schema rather than the training procedure. We bit-reproduce every F4 trained entry under fixed seed 42 (Appendix L.14). ReST-style expert iteration and Group Relative Policy Optimization (GRPO) do not internalize the verification signal on F4 (Appendix L). Within our training envelope, plan-usable schemas stay at the floor across every F4 variant that was validly evaluated: those variants never execute a plan from a correct schema (0/50 episodes), and the one 32B image variant with high structural emission (54% of episodes) gets their content wrong, so its single success is the degenerate environment noted above. The one row that executes a plan at all (5/50, none usable) is a defective evaluation and is retained as a record only. Together, these experiments locate a concrete opportunity for improvement: learning schemas whose predictions survive verification and support planning. Verified search provides a working route in the identifiable regimes, and the joint code+oracle interface makes that search programmable at L3.

Text-path and provenance caveats. The text Stage-1 variants use a separate trainer whose tokenizer path differs from the image recipe of Appendix K: a 512-token cap, loss over all tokens rather than assistant tokens only (the image trainer masks pre-assistant tokens), a hand-written chat template, and a Stage-1 system prompt that differs from the evaluation prompt by one line. These differ from the evaluation-time inputs and are reported here rather than corrected in place: no run is retrained with a corrected text path in this work. Training runs also carry no recorded code hash or command line: the version of the evaluation code that reproduces the archived F4 behavior is the one at the current revision (Appendix L.14), but the local and remote trees are not provably byte-identical, so the training-envelope claims are scoped to the configurations whose loader and prompt paths are verified. None of these caveats touches the ground-truth-injection ceiling or the fallback-identity measurement, which are computed from episode records without a checkpoint or prompt path.

N AUXILIARY EFFICIENCY INDICES

As an auxiliary efficiency index, we report

$$\mathrm{CE} = \mathbb{E}\left[\mathbf{1}_{\mathrm{success}} \frac{b_f}{N}\right], \qquad \mathrm{CE}_{\mathrm{cond}} = \mathbb{E}\left[\frac{b_f}{N} \,\middle|\, \mathrm{success}\right], \tag{1}$$

where N counts actions across both phases and b_f is the fixed reference constant for family f. These constants originate from informal human-step estimates, documented in Appendix H. For the same episode set with at least one success, $\mathrm{CE} = \mathrm{SR} \cdot \mathrm{CE}_{\mathrm{cond}}$ (SR as a fraction): the two scores distinguish how often an agent succeeds from its normalized action cost on successes. Family-specific constants also weight pooled comparisons.

Table 24: Induction frontier on F4 (50 test envs). Trained protocols match the no-model fallback's success set; trajectories match action-for-action in 45–50/50 episodes (50/50 at seed 42; Appendix L.14). CA is trajectory-only CA-proxy. The F1 control scores 12%, below a model-free greedy policy's 100%; search references require no training.

(a) Trained protocols (F4)

Protocol	Model	Examples
LoRA (5 configs)	9B text	200–2,700
Full FT[‡,♯]	4B text	2,000
CoT LoRA	9B text	2,000
LoRA (DS-Z3)	27B text	200 (mixed)
LoRA	32B image	200 (mixed)
LoRA (multi-obs)[♯]	9B text	212

(b) Outcomes and controls

Method / evaluation	SR (%)	CA-proxy
All protocols in (a)	**2.0**	**0.28**
Random (uniform)	2.0	—
No-model fallback	2.0	—
+ GT schema	**100.0**	—
FM_{heur} (enum+BFS)	88.0	0.27
Oracle BFS (GT map)	100.0	—
LoRA / F1 control[†]	12.0	1.00[§]

[†]Same 9B LoRA recipe, 400 F1 trajectories (episodes): usable schemas in 4/50 episodes (4/4 succeed); F4 SR: 2%.
[‡]Full FT output-format schema accuracy: 0.10.
[♯]Evaluation-side implementation defect: the multi-observation branch raised an unbound local before generating, and the Full FT row was loaded as base weights with the ground-truth schema line present in its training input. Both rows are retained as records and contribute no evidence about the trained model; the "All protocols" aggregate includes them, and dropping them leaves it unchanged, since every remaining row's success set equals the no-model fallback's (2.0%). See the text-path caveats below.
[§]Training-log output-format schema accuracy, not the trajectory-only proxy.