PLANSPACE: SEPARATING REFERENCE REPRODUCTION, PLAN VALIDITY, AND INTERFACE COMPLIANCE IN SYMBOLIC PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A household task can have several correct plans: groceries may go into different allowed cabinets, and independent placements may occur in either order. Scoring only against one recorded sequence rejects such alternatives. It also leaves a low score hard to interpret: did the model format its answer incorrectly, choose an action it could not execute, or fail to reach the goal? We introduce PLANSPACE, an evaluation protocol that answers these questions separately. It checks whether a submitted plan can be interpreted, whether it matches a reference or the dependencies of a known plan family, and whether its actions achieve the goal under an explicit symbolic model. Across six open-weight instruction-tuned language models on 171 supported BEHAVIOR-1K tasks, exact matching misses successful plans for every model. For Qwen3-8B, the discrepancy is especially large when tasks permit alternative goal choices or require longer plans. Other low scores arise before planning can be assessed: a small formatting correction exposes successful plans in one model's existing outputs but leaves the other models unchanged. These findings show why reference agreement and task success should be reported separately, alongside the interface checks needed to interpret a plan.

1 INTRODUCTION

Consider a task that allows groceries to be stored in either of two cabinets. A demonstration places them in the first cabinet, one item at a time. Another plan uses the second cabinet or changes the order of independent placements. If both plans satisfy the task's access conditions and final goal, their different sequences do not make either one incorrect. Exact matching to the demonstration nevertheless rejects the alternative.

The same low score can also arise for different reasons. A plan may name an unknown action, try to place an object through a closed cabinet, or execute successfully while leaving a required item outside. These failures concern *interface compliance*, action applicability, and goal completion. None is identified by measuring only *reference reproduction*. To evaluate *behavioral planning*, the evaluator must first interpret the actions, then check their effects on the task state. Similarity to a recorded sequence cannot replace these checks.

Existing work provides the tools needed to distinguish these outcomes. Embodied benchmarks such as BEHAVIOR-1K evaluate task outcomes through simulation, while formal planning benchmarks such as PlanBench and PlanningBench use structured domains and automatic verification to test generated plans (Li et al., 2024; Valmeekam et al., 2023a; Zhao et al., 2026). LLM-Modulo similarly places model-based verifiers around language-model predictions (Kambhampati et al., 2024). In parallel, diverse planning and least-commitment planning study sets of solutions and partial orders. These lines establish how to verify a plan and how to represent more than one solution. PLANSPACE uses both to examine what a benchmark score measures. Replay checks whether a prediction reaches the goal; partial-order families show whether it follows the dependencies of a known solution even when its action order differs.

Starting from task states, goals, and an explicit action model, PLANSPACE constructs replay-validated representatives and bounded partial-order families. It compares each parsed output with these

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.

references and replays the output directly (Figure 1). References describe known solutions without defining the entire solution space.

On 171 supported BEHAVIOR-1K tasks, we evaluate 5,130 outputs from 6 open-weight instruction-tuned language models. Goal-achieving plans outnumber exact matches for 6 models. Controls reproduce this discrepancy without language-model generation, and search without the references discovers alternative solutions independently of the constructor. A different problem appears at the interface: removing one literal prefix from fixed outputs changes one model's goal-validity score by 38.6 points and leaves the other five unchanged. These comparisons show why a score change must be interpreted through the rule that produced it.

Our contributions are a joint measurement protocol, a supported task set with bounded structural annotations, and an empirical decomposition of reference undercoverage, interface mismatches, and planning failures.

2 Related Work

Table 1 compares the evaluation questions addressed by the closest benchmarks. PlanSpace adds a structural view of successful alternatives: it relates each submitted plan to both a selected sequence and known DAG families, while separately checking its interface and goal outcome. This joint view reveals whether a reference mismatch reflects a different successful plan or an actual failure.

Table 1: **Evaluation components across planning benchmarks.** ✓: included; ×: outside the evaluated protocol; –: not reported in the cited evaluation.

Work	Goal / constraint checks	Symbolic replay	Robot execution	DAG-family scoring	Joint four checks
BEHAVIOR-1K (2024)	✓	×	✓	–	–
WatchAct (2026)	✓	✓	✓	–	–
PlanBench (2023a)	✓	✓	×	–	–
PlanningBench (2026)	✓	×	×	–	–
LexiCon (2025)	✓	✓	×	–	–
PlanSpace (ours)	✓	✓	×	✓	✓

Embodied evaluation. Embodied benchmarks connect task specifications to observable outcomes. BEHAVIOR-1K specifies 1,000 everyday activities in a realistic simulator (Li et al., 2024). CALVIN and LIBERO evaluate long-horizon and lifelong robot manipulation from demonstrations (Mees et al., 2022; Liu et al., 2023), while EmbodiedBench and EmbodiedEval broaden evaluation of multimodal agents (Yang et al., 2025; Cheng et al., 2025). WatchAct separates video-to-plan reasoning, policy execution under oracle plans, and full-task success (Li et al., 2026). These outcome and stage-based evaluations motivate our measurement question. Using a supported subset of BEHAVIOR-1K's symbolic task definitions, we compare several acceptance rules on the same generated plans. Our primary evaluation concerns symbolic planning; the physical audit examines where that abstraction stops being sufficient.

Language models and formal planning. Language models can produce high-level embodied plans when grounded by admissible skills or executable programs (Huang et al., 2022; Ahn et al., 2022; Singh et al., 2023). PlanBench probes reasoning about actions and change with formal domains (Valmeekam et al., 2023a); PlanningBench generates scalable, verifiable planning data (Zhao et al., 2026). LexiCon automatically generates constrained natural-language planning instances and verifies whether model plans are valid or optimal (Mantenoglou et al., 2025). LLM-Modulo uses an external model-based verifier to check model outputs (Kambhampati et al., 2024). We build on this use of explicit verification. Alongside goal validity, PlanSpace reports reference agreement and membership in known partial-order families, so a successful alternative can be distinguished from both imitation of a reference and failure to solve the task.

Diverse and partial-order planning. Least-commitment planning represents necessary precedence without committing to one total order (Weld, 1994). Prior work studies plan-set diversity, distance,

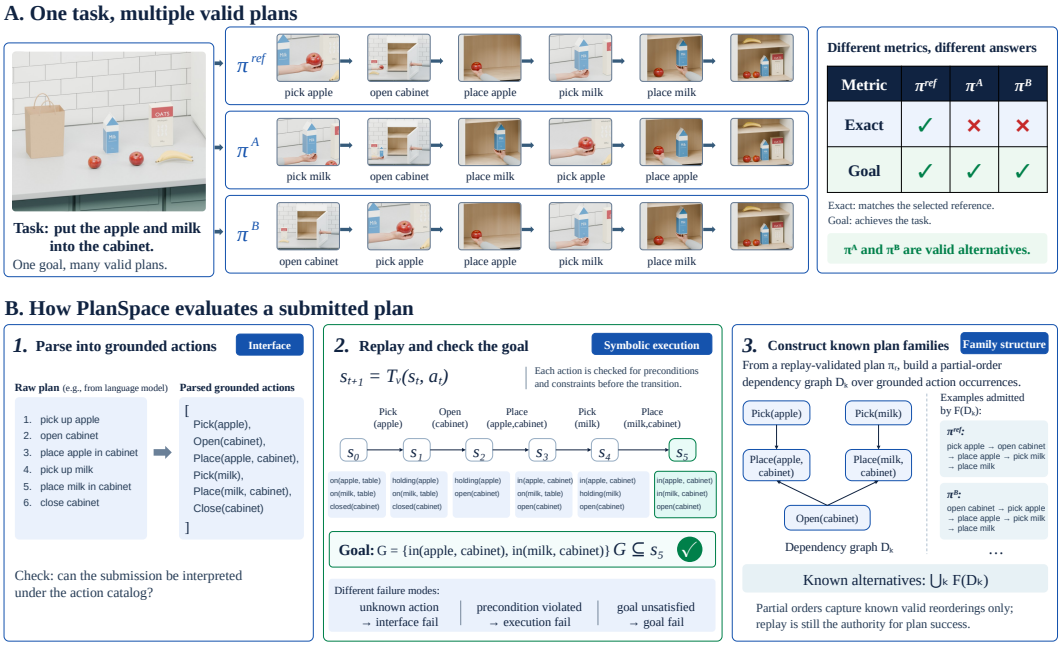


Figure 1: **One goal, multiple valid plans.** (A) Reordering the grocery-storage plan preserves goal achievement but fails exact matching to the selected reference. (B) PLANSPACE separates parsing, symbolic replay, and comparison with known partial-order families. Replay determines goal validity; the families describe known alternatives. The rendered scenes and pick/place actions are conceptual illustrations; the experimental interface and action model are defined in Section 3.

and quality (Roberts et al., 2014; Goldman & Kuter, 2015; Katz et al., 2022; Ghasemi et al., 2021; Abdelwahed et al., 2024), while partial-order reduction removes redundant interleavings (Xu et al., 2011). In PLANSPACE, partial orders annotate known solutions for evaluation. A DAG records the dependencies of a successful representative; its topological orders describe a known family against which model outputs can be compared.

3 PLAN-SPACE EVALUATION

PLANSPACE asks two questions about a submitted action sequence: does it solve the task, and how does it relate to the recorded solutions? To answer the first, we need rules for executing actions and checking the final goal. To answer the second, we need references that distinguish required dependencies from arbitrary choices of action order. Sections 3.1 and 3.2 define these two evaluation resources: a symbolic task model and reference plan families represented by directed acyclic graphs (DAGs). Section 3.3 then applies both to a model's submission and reports separate interface, reference, and outcome checks. A schematic grocery task runs through the definitions and the scoring examples below. Reference families serve as structural diagnostics: they identify reorderings of known plans, while replay determines task success.

3.1 TASK SEMANTICS AND PLAN EXECUTION

Consider an apple and a carton of milk on a table, with a closed cabinet nearby. The task is to put both objects inside the cabinet. Let o open the cabinet, and let a and m transfer the apple and milk into it. Each transfer requires an open cabinet and its object to remain on the table; executing it replaces that object's table relation with an inside relation. These rules make (o, a, m) and (o, m, a) successful plans. In contrast, (a, o, m) attempts a transfer through a closed cabinet, and (o, a) leaves the milk outside. The evaluator must distinguish these outcomes even though all four sequences use recognized actions. Replay here means updating symbolic facts according to these rules. For instance, a transfer updates an object's location without simulating the robot's grasp or motion.

These rules are fixed before model outputs are evaluated. The benchmark supplies objects, an initial state, and a goal; our symbolic action model supplies the available operations and their conditions and effects (Section 4). Together they define a planning instance $M_v = (\mathcal{P}, s_0, \mathcal{A}_v, \mathcal{G}, c)$. The index v identifies the action-semantics version shared by reference construction and prediction replay. A *ground* fact or action refers to specific objects. The set $\mathcal{P}$ contains the task's ground facts, and a state $s \subseteq \mathcal{P}$ records which of them hold; s_0 is the initial state. Compiled Boolean facts include explicit negative forms, such as `not_open`, so a goal can require the cabinet to be closed. For example, opening the cabinet makes $\mathrm{open}(\mathrm{cabinet})$ hold. The action catalog $\mathcal{A}_v$ gives each available grounded operation a unique identifier, so transferring the apple and transferring the milk are distinct actions.

The set $\mathcal{G}$ specifies the allowed goal alternatives. In the running example, it contains one alternative g: the conjunction of $\mathrm{inside}(\mathrm{apple}, \mathrm{cabinet})$ and $\mathrm{inside}(\mathrm{milk}, \mathrm{cabinet})$. Both facts must hold. If the task permits several alternatives, satisfying any one complete conjunction suffices; facts from different alternatives cannot be combined to satisfy the goal. The cost function c assigns nonnegative action costs for the efficiency comparison in Section 3.3.

For any catalog action α, the evaluator checks its preconditions $\mathrm{Pre}(\alpha)$ and constraints $C_v(\alpha, s)$. Preconditions are required facts, such as the apple still being on the table. Constraints check relations in the current or resulting state, such as access through open containers and a unique location for each object. If both checks pass, the evaluator removes the delete effects and inserts the add effects:

$$T_v(s, \alpha) = \begin{cases} (s \setminus \mathrm{Del}(\alpha)) \cup \mathrm{Add}(\alpha), & \mathrm{Pre}(\alpha) \subseteq s \ \wedge \ C_v(\alpha, s), \\ \bot, & \text{otherwise.} \end{cases} \tag{1}$$

Here Del and Add are the facts removed and inserted, and $\bot$ denotes an inapplicable action. Starting from s_0, symbolic replay applies T_v to each action of $\pi = (a_1, \ldots, a_L)$ in order. The plan is *executable* if every transition succeeds. It is *goal valid* if it is executable and its final state s_L satisfies $g \subseteq s_L$ for some $g \in \mathcal{G}$. Thus, (a, o, m) fails during execution, whereas (o, a) executes but fails the goal check. Repeating the apple transfer also fails once the apple has left the table. These outcomes follow from action applicability and the goal definition; cost is evaluated separately. Replay can therefore determine success without a reference plan. The next subsection introduces references to describe which known solution structure a submitted plan follows.

3.2 From reference plans to DAG families

To construct a reference family, we start with a sequence that passes replay and replace its total order with a DAG of retained dependencies. This gives reference comparison a way to recognize reordered versions of a known solution. For the reference (o, a, m), opening must precede both transfers, but the apple and milk transfers can occur in either order because they change different objects' locations. Edges $o \to a$ and $o \to m$ express the required precedence while leaving the two transfers unordered (Figure 2a). Reference sequences are constructed from the task model before predictions are scored; their generation procedure is given in Section 4.

Building nodes and dependencies. For a reference sequence $(a_1, \ldots, a_L)$, we create a node i for each *occurrence* of an action and label it with that action's identifier. Two occurrences of the same action therefore remain distinct nodes. For each pair $i < j$, we retain $i \to j$ if the actions have a conservative dependency: one changes a fact tested by the other, their add and delete effects conflict, or they interact through an object's access chain or a container's open state. Edges follow the reference order, so the graph is acyclic. Transitive reduction then removes edges whose precedence is already implied by a path, preserving the admitted orders. For example, if the graph contains $i \to j \to k$, a direct edge $i \to k$ is redundant. In the grocery example, opening supplies the access required by both transfers; neither transfer introduces a dependency on the other.

From one sequence to a family. A topological order visits every node once while placing each predecessor before its successors. We write $\mathcal{F}(D)$ for the action sequences obtained by reading the node labels in all topological orders of D. For the three-node graph, $\mathcal{F}(D) = \{(o, a, m), (o, m, a)\}$: one reference represents two schedules. The graph defines membership through its labeled nodes and precedence edges.

To check the orders that the graph *does* admit, we also replay them, checking all orders when they fit within the validation budget and a bounded subset otherwise. We record this validation scope with the reference family. Section 3.3 independently replays each submitted plan to establish its goal validity.

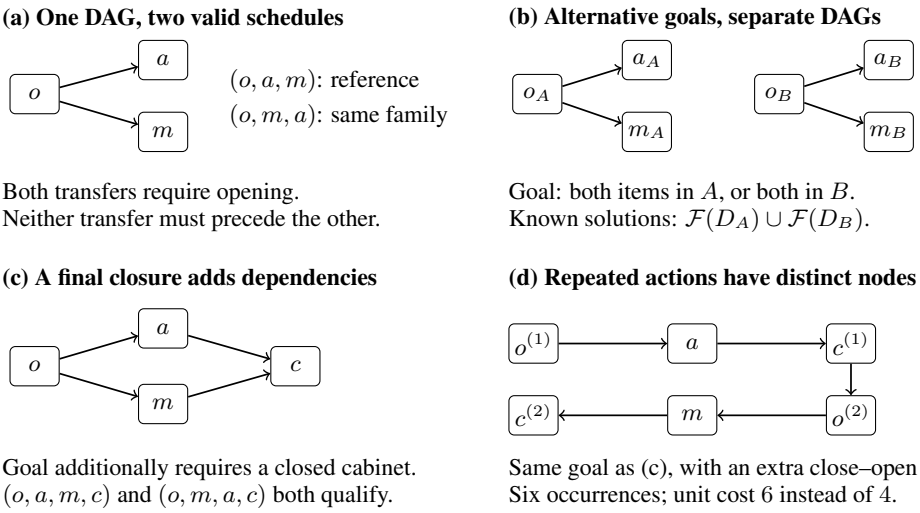


Figure 2: **How DAGs represent known plan families.** Nodes are grounded action occurrences; arrows are precedence constraints, with transitive edges omitted. o opens a cabinet, a and m transfer apple and milk from the table, and c closes it. Subscripts select a cabinet; superscripts distinguish repeated occurrences of the same action. (a) Independent transfers admit two schedules. (b) Alternative destinations produce separate families. (c) Requiring final closure creates a join after both transfers. (d) A longer successful plan contains additional action occurrences and has its own dependency graph. These schematic instances illustrate the protocol; $\mathcal{F}(D)$ denotes the action sequences read from topological orders of DAG D.

Alternative goals and action compositions. Reordering a graph's nodes preserves its *action multiset*: which grounded actions occur and how many times each occurs. Some tasks admit choices that change this multiset. Suppose both items may go into cabinet A or both into cabinet B. These goal alternatives produce separate reference graphs D_A and D_B, with different grounded action labels (Figure 2b). Their families are combined by union, $\mathcal{F}(D_A) \cup \mathcal{F}(D_B)$. A plan belongs to this union only if one complete graph admits it; combining the apple transfer from D_A with the milk transfer from D_B does not qualify and also misses this task's goal. With K reference graphs $D_1, \ldots, D_K$, the same construction gives the known plan set $\bigcup_{k=1}^{K} \mathcal{F}(D_k)$. Each graph describes reorderings of one representative; the union also accommodates differences between representatives.

Figure 2c–d separates a change in the goal from a change in the chosen solution. Requiring final closure adds dependencies after both transfers; inserting another close–open cycle adds action occurrences and requires a different graph. Replay can recognize either successful plan even if its graph is absent from the known set. Appendix A works through these variants, and Appendix M gives graphs from all evaluated tasks.

3.3 Evaluating a submitted plan

The evaluator applies three operations to a submitted sequence: resolve its actions, compare it with the references, and replay it in the task model. Reference comparison returns two indicators, Exact and Family; interface checking and replay supply the other two, Interface and Goal. The model submits a sequence.

Resolving actions through the interface. The evaluator first parses the submission into a sequence of grounded action identifiers $\hat{\pi}$ and resolves each identifier against $\mathcal{A}_v$. Malformed output or an unknown identifier produces an interface failure; truncation at the output-token limit is recorded separately. For example, if the catalog contains `open::cabinet` but no `unlock::cabinet`,

submitting the latter fails identifier lookup. In contrast, the apple transfer in (a, o, m) is a recognized action whose preconditions fail at the initial state. It passes the interface check and fails during replay. Interface checking therefore establishes whether the submission has an interpretation under the action model; replay determines what that interpreted sequence can do.

Comparing with a sequence and with a family. For exact agreement, one of the validated reference sequences is designated π_r. The submitted sequence must contain the same grounded actions in the same order. Family membership compares against all K reference graphs, accepting any order admitted by at least one of them:

$$\text{Exact}(\hat{\pi}) = \mathbf{1}[\hat{\pi} = \pi_r], \tag{2}$$

$$\text{Family}(\hat{\pi}) = \mathbf{1}\left[\hat{\pi} \in \bigcup_{k=1}^{K} \mathcal{F}(D_k)\right]. \tag{3}$$

Here $\mathbf{1}[\cdot]$ equals one when its condition holds and zero otherwise. Family membership requires the same action occurrences as one reference DAG, ordered consistently with its edges. The evaluator matches the sequence directly to the graph, independently of the ordering-validation cap; Appendix L gives the occurrence-level procedure.

Testing the task outcome. Every submission that passes the interface check is also replayed from s_0 using Section 3.1, regardless of its reference scores. The goal indicator is one only if every action executes and the final state satisfies an allowed goal alternative. If an action fails, the report identifies its position and the unmet precondition or constraint. If all actions execute but the goal is unsatisfied, the report records a goal miss. Goal validity is the primary success criterion, while reference scores describe how a plan relates to known solutions.

For example, (o, m, a) scores $(1, 0, 1, 1)$ in the order (Interface, Exact, Family, Goal): it changes the schedule while preserving the dependencies and goal. Replay further distinguishes the failed action in (a, o, m) from the unfinished goal in (o, a). Appendix A gives the complete score assignments; Appendix L connects the distinctions to actual model outputs.

Aggregating outcomes and comparing costs. The per-output indicators answer different aggregate questions. Single-reference false rejection asks what fraction of a model's successful outputs exact matching rejects:

$$\text{FRR}_{\text{1-ref}} = \frac{\sum_i \mathbf{1}[\text{Goal}(\hat{\pi}_i) = 1 \wedge \hat{\pi}_i \neq \pi_{r,i}]}{\sum_i \mathbf{1}[\text{Goal}(\hat{\pi}_i) = 1]}. \tag{4}$$

Here i indexes evaluated submissions and $\pi_{r,i}$ is the selected reference for submission i's task. The denominator counts successful outputs, and the numerator counts those successful outputs that differ from their reference. To measure diversity over repeated samples, we also count the fraction of known reference families matched by at least one sample.

We assess efficiency separately from validity, comparing a successful plan's additive action cost with the cheapest constructed representative. Appendix F defines reference-relative regret and the cost-bounded analysis.

4 Benchmark Construction

This section explains how we build the three resources used in Section 3: task instances, an executable action model, and reference families. BEHAVIOR supplies objects, initial states, and goals; PlanSpace supplies the high-level skills and constructs references under the same action model used to evaluate predictions.

From source definitions to comparable task instances We translate supported BEHAVIOR-1K v3.9.2 definitions into initial facts and complete goal alternatives. Of 173 supported candidates, a shared-context filter retains 171 using OLMo-2 tokenizer lengths. Both semantic selection and context filtering are independent of model outputs. This yields a fixed comparison set within the supported action abstraction. Appendix I.2 specifies the translation and exclusion rules; Appendix E.2 gives the context filter and its two boundary cases.

Making the action semantics explicit The action model instantiates Section 3.1 with five skills: OPEN, CLOSE, TRANSFER, TOGGLEON, and TOGGLEOFF. A transfer combines pickup, transport, and placement; it requires the object at its specified source and open access through any enclosing gated containers. Models therefore choose and order high-level skills. All primary evaluations use the spatially constrained action model (domain-v2). Appendix I details its effects, state constraints, and completion rules.

Constructing references and organizing the evaluation Before scoring model outputs, a deterministic causal constructor builds representatives for grounded goal alternatives. We replay each candidate, retain successful traces, merge duplicates, and form the DAGs in Section 3.2. The 171-task suite contains 1,403 unique goal-plan representatives; 145 tasks admit multiple representatives or orderings. These references describe known solutions, while direct replay can accept solutions outside them. Appendix N gives the algorithm and its coverage properties, and Appendix F reports ordering-validation budgets.

5 EXPERIMENTS

5.1 MODELS AND PLANNING PROTOCOL

Evaluation questions and models. The experiments ask how often exact matching misses successful plans, what task and plan structure reveal about these misses, and how interface errors affect measured planning performance. We evaluate Qwen3-4B, Qwen3-8B, and Qwen3-14B (Qwen Team, 2025), Phi-4-mini-instruct (Abouelenin et al., 2025), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and OLMo-2-7B-Instruct (OLMo Team et al., 2024). Appendix D.1 lists the model revisions used. The three Qwen sizes provide a within-family comparison; the other models test whether the observations extend across model families.

Planning setting. Each model chooses and orders the high-level skills constructed in Section 4. For example, it must decide to open a cabinet before transferring an item into it. Models receive the task's starting facts, goal choices, and available action names; the evaluator retains the formal preconditions and effects. Thus the experiment tests planning from named skills. We collect five plans per model on each of the 171 tasks, yielding 855 outputs per model and 5,130 overall. Appendix H gives the exact prompt construction, decoding settings, and output parser; These scores concern symbolic plans. The separate physical audit and adapted robot demonstrations are reported in Appendix J.

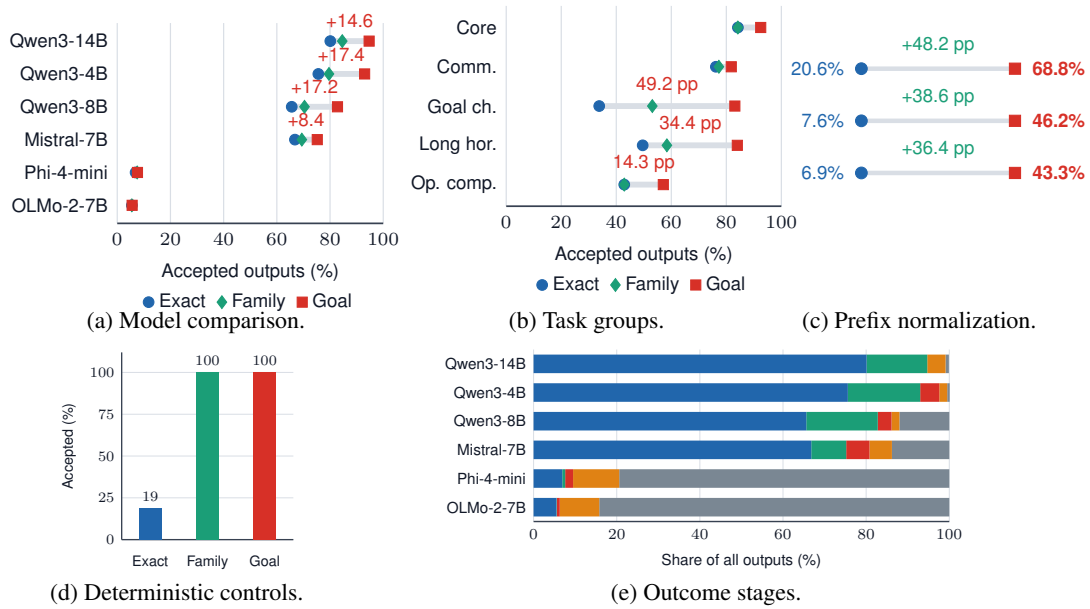
5.2 CONTROLLED COMPARISONS AND UNCERTAINTY

Comparisons and uncertainty. The main comparison changes the acceptance rule while holding each submitted plan fixed. Constructed positive and negative controls test which changes to a plan each rule accepts. Reference-blind search and an action-cost bound check whether successful alternatives depend on our reference constructor or on using extra actions. Structural groups then locate the score differences, while a parser-only intervention tests one source of interface failure without regenerating plans. These comparisons have distinct sample sets, summarized in Appendix E.1.

Primary acceptance rates use all 855 outputs per model; conditional false-rejection rates use only successful outputs. We report 95% task-bootstrap intervals and paired model comparisons, with resampling and multiple-comparison procedures in Appendix D.2.

6 RESULTS

We first measure successful plans rejected by exact matching, then use task structure and DAG families to explain reference coverage, and finally separate interface failures from failures during replay. Figures 3–3 address these three questions. Complete estimates, intervals, and secondary breakdowns appear in Appendix Tables 15–20.



(a) Model comparison. (b) Task groups. (c) Prefix normalization.

(d) Deterministic controls. (e) Outcome stages.

Figure 3: **Reference agreement, goal achievement, and interface compliance reveal different sources of low scores.** (a) Six-model comparison; annotations show goal–exact gaps exceeding three percentage points. (b) Qwen3-8B across Core, commutation (Comm.), goal choice (Goal ch.), long horizon (Long hor.), and operator composition (Op. comp.). (c) Phi-4-mini Parse, Goal, and Exact rates, from top to bottom: strict parsing (blue circles) and prefix normalization (red squares). The other five models are unchanged. (d) On 100 control tasks, family membership and goal replay accept every valid alternative, versus 19% for exact match; replay rejects all required-action deletions as goal misses. (e) Strict-parser outcomes across 5,130 outputs: ■ valid exact, ■ valid non-reference, ■ executable goal miss, ■ failed action, and ■ parse failure. Full intervals and group rates appear in Tables 17 and 6.

6.1 Reference Undercoverage: Success Beyond a Selected Plan

Exact matching misses successful alternatives. Figure 3a shows that goal validity exceeds exact match for all six models. Two quantities summarize this shortfall, and they answer different questions. The *score gap* measures how much success exact matching fails to credit across all submissions. It is the difference, in percentage points, between the goal-validity and exact-match rates over all 855 outputs of a model, and reaches 17.4 points for Qwen3-4B. The *false-rejection rate* measures how often exact matching rejects a successful plan. Its denominator contains only goal-valid outputs. It reaches 20.8% for Qwen3-8B, meaning that about one in five of this model's successful plans is marked wrong by exact matching. Both measures show that a plan can depart from the recorded sequence and still complete the task, as illustrated by alternative goal bindings and action orderings in Appendices A and L.

Controls isolate the effect of the acceptance rule. Across one positive control per task in the historical frozen 100-task subset, replay and partial-order membership accept 100%, while exact match accepts only 19.0% (Figure 3d). Conversely, all 100 required-action deletions are rejected (100.0%), each as an executable goal miss. Because these plans are constructed without model generation, the contrast isolates the acceptance rule. Replay credits alternative goal bindings and commuting orders while retaining a check on required actions.

The gap extends beyond constructed controls and longer plans. Uniform-cost search without reference access finds alternative solutions on 145 of 171 tasks under the shared action model. Its 591 retained examples all pass corrected-source goal replay; they are search-generated probes, separate from the 100 positive controls above. Successful alternatives also remain when plans may use no more actions than the cheapest constructed reference: Qwen3-4B still scores 86.2% under this bound, compared with 75.6% by exact matching. Appendix F gives the search and cost analyses.

Independent-executor checks and counterfactual exact scoring on external planning archives are reported in Appendix B.

6.2 Structural Diagnosis: Task Groups and Reference Families

Task structure changes the diagnosis of a low score. Figure 3b groups Qwen3-8B's tasks by their goal choices, dependencies, plan length, and required skills (Appendix E.5). Goal-choice and long-horizon tasks have large goal–exact gaps (49.2 and 34.4 points), indicating many successful plans outside the recorded sequence. The original/new-task breakdown in Appendix E.3 also shows reference undercoverage on the 71 added tasks.

Family membership describes reference coverage. A successful plan can fall outside the known DAGs either by changing the action multiset or by violating a conservative dependency that replay does not need. Appendix C separates these cases and tests dependency relaxation. Thus a family miss alone does not establish a new action composition.

6.3 Failure Attribution: Interface Errors and Planning Outcomes

Failure stages distinguish interface errors from planning errors. Figure 3e partitions every output into valid exact, valid non-reference, executable goal miss, failed action, or parse failure. Whereas Figure 3 measures uncredited success, this breakdown locates unsuccessful submissions. Phi-4-mini and OLMo-2 lose 79.4% and 84.2% of outputs at parsing, respectively. Mistral-7B has 13.8% parse failures and 5.4% failed actions. The same low aggregate score can therefore indicate an interface bottleneck, an action-applicability error, or a goal-completion failure. The stage counts identify which part of the submitted output the evaluator was able to test.

A small interface change reveals different failure causes. Figure 3c then holds raw outputs fixed and removes only one literal `action_id=` prefix. Phi-4-mini's parse rate rises from 20.6% to 68.8%, and its goal-validity score rises from 7.6% to 46.2%; the other five models are unchanged. The change shows how much of Phi-4-mini's measured failure depends on this particular serialization rule. In contrast, OLMo-2 retains all 706 unknown-action failures after the same normalization. Similar parse rates can therefore conceal different interface problems.

This distinction changes the next step: correcting a known formatting mismatch can recover an already successful plan, whereas a replayed goal miss requires changing the plan. Repeated sampling is a separate intervention; its one-to-five-draw success and family-coverage curves are reported in Appendix E.6.

7 Conclusion

PlanSpace separates interface compliance, reference reproduction, known-family structure, and goal validity. Exact matching misses successful plans for all six models, with a largest gap of 17.4 points. Controls, search, and verifier comparisons distinguish this undercoverage from interface and planning errors. Reporting these checks on the same outputs makes the acceptance rule and the source of each failure explicit. The resulting diagnosis points to different remedies: goal-based credit for successful non-reference plans, interface inspection for unrecognized actions, and replanning for executable goal misses.

Reproducibility statement

The research artifacts retain the versioned evaluator, frozen task queue, model revisions, decoding settings, and raw outputs. Appendix D describes how the recorded provenance connects outputs to their evaluation conditions.

AI Use Statement

The authors used AI-based writing assistance for language editing and manuscript formatting. All research content—the audit design, detector and review protocol, contract construction, experiments,

statistical analyses, and all reported numbers—was produced and verified by the authors, who take full responsibility for the content of this submission.

Ethics statement

This study uses simulated household task specifications and no personal data.

References

Mustafa F. Abdelwahed, Joan Espasa, Alice Toniolo, and Ian P. Gent. Behaviour planning: A toolkit for diverse planning. *arXiv preprint arXiv:2405.04300*, 2024. URL https://arxiv.org/abs/2405.04300.

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025. URL https://arxiv.org/abs/2503.01743.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. URL https://arxiv.org/abs/2204.01691.

Zhili Cheng, Yuge Tu, Ran Li, Shiqi Dai, Jinyi Hu, Shengding Hu, Jiahao Li, Yang Shi, Tianyu Yu, Weize Chen, Lei Shi, and Maosong Sun. Embodiedeval: Evaluate multimodal llms as embodied agents. *arXiv preprint arXiv:2501.11858*, 2025. URL https://arxiv.org/abs/2501.11858.

Mahsa Ghasemi, Evan Scope Crafts, Bo Zhao, and Ufuk Topcu. Multiple plans are better than one: Diverse stochastic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pp. 140–148, 2021. doi: 10.1609/icaps.v31i1.15956.

Robert P. Goldman and Ugur Kuter. Measuring plan diversity: Pathologies in existing approaches and a new plan distance metric. In *AAAI Conference on Artificial Intelligence*, 2015. doi: 10.1609/aaai.v29i1.9669.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022. URL https://arxiv.org/abs/2201.07207.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

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.

Michael Katz, Shirin Sohrabi, and Octavian Udrea. Bounding quality in diverse planning. In *AAAI Conference on Artificial Intelligence*, 2022.

Baiqi Li, Ce Zhang, Yu Fang, Yue Yang, Shangzhe Li, Mingyu Ding, and Gedas Bertasius. Watchact: A benchmark for behavior-grounded robot manipulation. *arXiv preprint arXiv:2606.26443*, 2026. URL https://arxiv.org/abs/2606.26443.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R. Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio

Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1K: A Human-Centered, Embodied AI Benchmark with 1,000 Everyday Activities and Realistic Simulation. *arXiv preprint arXiv:2403.09227*, 2024. URL https://arxiv.org/abs/2403.09227.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.03310.

Periklis Mantenoglou, Rishi Hazra, Pedro Zuidberg Dos Martires, and Luc De Raedt. LexiCon: a benchmark for planning under temporal constraints in natural language. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/8ef63909f4bc1cedffb993c0e285a4e7-Abstract-Datasets_and_Benchmarks_Track.html. doi: 10.52202/085713-3318.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. In *IEEE Robotics and Automation Letters*, 2022. URL https://arxiv.org/abs/2112.03227.

OLMo Team, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, et al. 2 OLMo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024. URL https://arxiv.org/abs/2501.00656.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Mark Roberts, Adele E. Howe, and Indrajit Ray. Evaluating diversity in classical planning. In *International Conference on Automated Planning and Scheduling*, 2014. doi: 10.1609/icaps.v24i1.13649.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation*, 2023. URL https://arxiv.org/abs/2209.11302.

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

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models: A critical investigation. In *Advances in Neural Information Processing Systems*, 2023b. URL https://arxiv.org/abs/2305.15771.

Daniel S. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61, 1994. doi: 10.1609/aimag.v15i4.1109.

You Xu, Yixin Chen, Qiang Lu, and Ruoyun Huang. Theory and algorithms for partial order based reduction in planning. *arXiv preprint arXiv:1106.5427*, 2011. URL https://arxiv.org/abs/1106.5427.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025. URL https://arxiv.org/abs/2502.09560.

Ziliang Zhao, Zenan Xu, Shuting Wang, Hongjin Qian, Yan Lei, Minda Hu, Zhao Wang, Shihan Dou, Zhicheng Dou, and Pluto Zhou. Planningbench: Generating scalable and verifiable planning data for evaluating and training large language models. *arXiv preprint arXiv:2605.20873*, 2026. URL https://arxiv.org/abs/2605.20873.