What We Talk About When We Talk About LLM Planning: Evidence for Two Distinct Planning Abilities

Anonymous ACL submission

Abstract

When LLMs exhibit uneven performance across planning tasks, these gaps are often attributed to task difficulty. We argue that this explanation is incomplete, as task-level variation may reflect distinct latent planning competencies rather than differences along a single ability spectrum. We study this question on *ACPBench-Hard* by evaluating multiple LLM families under varying test-time reasoning budgets and applying a multidimensional item response theory model to uncover the latent competency structure underlying LLM planning. The analysis reveals two principal dimensions that shape planning performance: *operational reasoning*, the ability to evaluate local action applicability and immediate state transitions, and *structural enumeration*, the ability to reason about goal reachability and landmark structure. Operational reasoning improving under model scaling and longer reasoning traces, while structural enumeration remains comparatively insensitive. Our findings motivate competency-level evaluation of LLM planning, shifting the focus from whether models improve overall to which planning competencies improve, under what conditions, and why[1].

1 Introduction

As large language models (LLMs) transition from conversational interfaces to agentic systems equipped with tool use, persistent memory, and multi-step execution capabilities, planning has become a core skill underlying agentic behavior. (Chowa et al., 2026; Li et al., 2026). Symbolic planning offers a controlled testbed for this purpose: it requires finding a sequence of actions that transforms an initial state into a goal state under specified constraints, while allowing instances to be generated from formal specifications, solutions to be verified automatically, and difficulty to be varied systematically (Valmeekam et al., 2023b; Stein et al., 2025; Kokel et al., 2026).

Among recent benchmarks that use symbolic planning domains to probe LLM reasoning skills, *ACPBench* (Kokel et al., 2025) assesses reasoning about actions, state transitions, and plan construction and validation using multiple-choice and Boolean question types. Its recent extension, *ACPBench Hard* (Kokel et al., 2026), moves to generative, open-ended questions and partitions planning into eight distinct subtasks. Such multi-faceted probing is common in the literature: *PlanBench* (Valmeekam et al., 2023b) and *TRAC* (He et al., 2023) likewise evaluate planning through diverse question types. While these subtasks are useful for organizing evaluation and localizing failures, they should not be assumed to directly map onto the latent abilities governing model performance.

This distinction matters because recent progress in LLM reasoning has encouraged the expectation that LLM planning skill should improve with larger models, Chain-of-Thought (CoT), and stronger agent harness with tools or memory (Chen et al., 2025a; Xu et al., 2025; Li et al., 2026). Yet automated planning studies repeatedly find that LLMs remain brittle on planning benchmarks, even as models and test-time reasoning mechanisms improve (Kambhampati et al., 2024; Huang et al., 2025a). These results are often interpreted as evidence that planning is fundamentally difficult for LLMs, or that some subtasks are simply harder than others. An open question is whether uneven task-wise performance reflects merely difficulty variation along a single planning ability, or does it reveal multiple latent competencies that respond differently to scale and reasoning traces? Without addressing this question, we cannot reconcile these two seemingly conflicting observations.

To this end, we compare a family of Item Response Theory (IRT) and Multidimensional Item Response Theory (MIRT) models that vary in the

[1] code and data are provided via the Openreview platform.

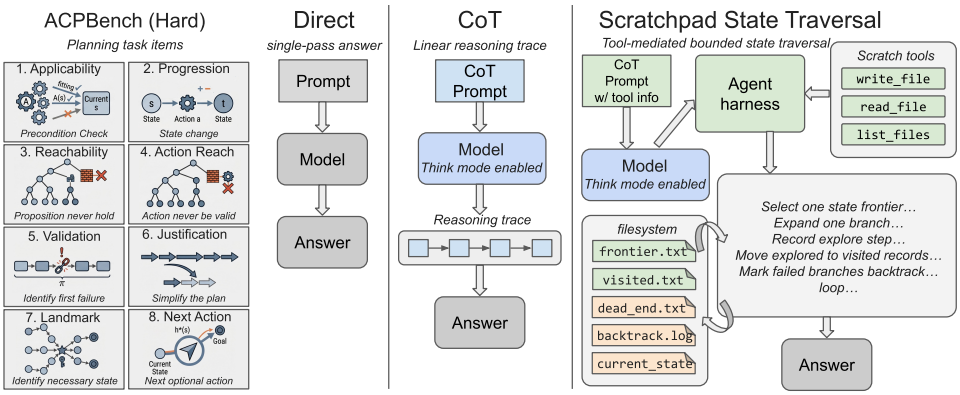

Figure 1: Evaluation design. Left: ACPBench (Hard)'s eight planning tasks (action, state, and plan levels). Right: the three inference conditions: direct answering, CoT with linear reasoning traces, and scratchpad harness, used to probe how latent planning competencies respond to varying prompting and test-time compute scaffolding.

number of latent dimensions and in how those dimensions combine to predict correctness. As illustrated in Figure 1, our evaluation spans three LLM families, *Qwen*, *Gemma*, and *Granite*, under three inference conditions: direct answering, Chain-of-Thought (CoT) prompting with the model's thinking mode enabled, and agent-harness scratchpad traversal. We evaluate these models on *ACPBench Hard*, using model selection to determine which latent structure best explains response patterns.

Our analysis supports the latter view. A non-compensatory two-dimensional MIRT model best explains the response data. This model reveals two interpretable dimensions: operational reasoning, reflected in applicability, progression, and validation tasks; and structural enumeration, reflected in reachability and landmark tasks. While operational reasoning improves with model size and chain-of-thought prompting, structural enumeration remains a persistent bottleneck. This asymmetric pattern holds across the model families we study.

Taken together, this work makes three contributions. First, it introduces a statistically grounded approach for interpreting LLM planning benchmark performance beyond aggregate accuracy, allowing model behavior to be analyzed through latent competency structure. Second, it provides evidence that LLM planning can decompose into distinct latent dimensions, challenging the view of planning as a single general ability. Third, it provides evidence that scaling model size and increasing test-time reasoning budget, both widely used to improve LLM reasoning, do not uniformly benefit planning, thereby identifying a systematic blind spot that aggregate benchmark scores do not expose.

2 Related Work and Background

2.1 Planning Benchmarks for LLMs

Recent LLM planning benchmarks, such as *ACPBench* (Kokel et al., 2025), *ACPBench-Hard* (Kokel et al., 2026), *TRAC* (He et al., 2023) and *ActionReasoningBench* (Handa et al., 2025), approach evaluation by decomposing end-to-end plan generation into fine-grained reasoning subtasks. These suites consistently report uneven task-wise performance. ACPBench Hard attributes such variation to differences in the reasoning difficulty of individual subtasks. However, such framing implicitly treats planning as a **single underlying ability**, with subtasks occupying different points along that single dimension. This perspective overlooks a key question: whether performance gaps arise not only from *how hard* a task is, but from *what kind* of planning competence it requires, and whether the predefined task categories in *ACPBench-Hard* correspond to distinct latent competencies. This work challenges the prevailing assumption of a single planning ability, instead identifying two competencies in LLM planning that respond asymmetrically to model scale and CoT mechanism.

2.2 Classical Planning Structure

Classical planning theory recognizes two modes of reasoning: transition-level reasoning (e.g., action applicability and progression) and state-space-level reasoning (e.g., reachability and landmarks) (Ghallab et al., 2004; Hoffmann et al., 2004). Modern planners do not treat these as two separate *competencies*, but fuse them into unified search procedures through heuristics (Bonet and Geffner, 2001), landmarks (Richter and Westphal, 2010), causal

graphs (Helmert, 2006), and novelty or width-based search (Lipovetzky and Geffner, 2012, 2017). Whether LLM planning behavior reflects separable, conjunctively required competencies along these lines remains an open question. Our work provides the first evidence for this two-dimensional competence structure in LLM planning behavior.

2.3 Scaling, CoT, and Reasoning Limits

LLM planning is a structured form of multi-step reasoning, and the prevailing assumption has long been that larger models and step-by-step reasoning via chain-of-thought (CoT) uniformly improve such capabilities (Kaplan et al., 2020; Wei et al., 2022). Yet this view has been challenged on two fronts. Chen et al. (2025b); Pu et al. (2025); Wu et al. (2026) show that LLMs often experience excessive test-time compute (i.e., *overthinking*) that degrades performance in reasoning tasks due to recursive self-doubt loops and error accumulation. In planning specifically, systematic evaluations tracking models from GPT-3 through GPT-4 to o1-style architectures show that even larger scales and better reasoning traces, such as those trained via Reinforcement Learning with Verifiable Rewards, cannot reliably plan (Valmeekam et al., 2022, 2025). These critiques leave the performance impact of scale and reasoning traces in planning genuinely uncertain. We show that their effects are not uniform: two latent dimensions of planning competence respond asymmetrically to scale and CoT. Recent work trains or scaffolds LLMs for planning via PDDL-aware instruction tuning (Verma et al., 2025), heuristic-integrated symbolic search (Tang et al., 2025), and LLM-as-formalizer (Huang et al., 2025b). We acknowledge this line of work but instead study off-the-shelf LLMs without domain-specific training in order to investigate intrinsic planning competences of LLMs.

3 Experimental Setup

ACPBench-Hard comprises eight planning-related subtasks: action applicability (`app`), action reachability (`areach`), action justification (`just`), landmark detection (`land`), next-action prediction (`nexta`), state progression (`prog`), state reachability (`reach`), and plan-goal validation (`val`) (the left panel of Figure 1 gives an intuitive visual summary). Each task contributes 130 items, yielding a total item bank of $J = 1{,}040$ planning instances. We construct a binary response matrix Y where $Y_{ij} = 1$ indicates respondent i solved item j.

3.1 Models and Inference Conditions

We evaluate three model families: *Qwen3.5* (0.8B, 2B, 4B, 9B, 27B), *Gemma 4* (2B, 4B), and *Granite 3.3* (2B, 8B). Qwen3.5 provides the primary scale sweep while the other two test cross-family generalization. We compare three core inference conditions that span a natural gradient of test-time reasoning budget.

Direct. The model receives a system prompt with few-shot examples that show only the final answer, with no intermediate reasoning trace demonstrated to encourage direct generation of the final answer.

CoT. Same system prompt with few-shot examples that include step-by-step reasoning traces before the final answer (Wei et al., 2022), with the model's thinking mode enabled.

Scratchpad. Planning often requires branching search and backtracking from dead ends. Tree-of-Thought (Yao et al., 2023) and Graph-of-Thought (Besta et al., 2024) orchestration provide these capabilities but mainstream LLMs lack native support for them. We bridge this gap with a lightweight *agent harness*. The model can access basic file read/write tools and instructed to externalize explicit state traversal for planning tasks. Specifically, it maintains four scratch files (`frontier.txt`, `visited.txt`, `dead_ends.txt`, and `backtrack_log.txt`) and follows a bounded search loop to simulate state-space traversal.

3.2 Item Response Formulation

Originally developed in educational testing and psychometrics, Item Response Theory (IRT) is widely used by assuming observed correctness is an indirect measurement of unobserved competence, making it a natural framework for our setting. We therefore cast LLM planning evaluation as an item-response problem. Each model-condition pair is a *respondent*, each *ACPBench-Hard* instance is an *item*, and each binary correctness score, graded by the benchmark's symbolic validator against PDDL ground truth, is a *response*.

3.2.1 Unidimensional IRT

In the simplest unidimensional IRT formulation, each respondent i has a scalar latent ability θ_i, and each item j has a scalar difficulty b_j. The probability that respondent i answers item j correctly is

modeled as

$$p_{ij} = \Pr(y_{ij} = 1 \mid \theta_i, b_j) = \sigma(\theta_i - b_j),$$

where $\sigma(x) = (1 + \exp(-x))^{-1}$. The observed response is then modeled as $y_{ij} \sim \text{Bernoulli}(p_{ij})$. Under this model, correct responses are likely when a respondent's latent ability exceeds item difficulty, and unlikely otherwise. In our context, it corresponds to the hypothesis that all planning tasks are governed by a single latent planning ability, with task variation captured only by item difficulty.

3.2.2 Multidimensional IRT

Multidimensional IRT (MIRT) generalizes scalar respondent ability and item difficulty into latent vectors $\boldsymbol{\theta} i = (\theta i1, \ldots, \theta_{iK})$ and $\mathbf{b} j = (bj1, \ldots, b_{jK})$ respectively. This formulation corresponds to the hypothesis that LLM planning may involve multiple latent competencies. Here, θ_{ik} denotes respondent i's ability on latent dimension k, and b_{jk} denotes item j's difficulty on that dimension. MIRT models differ in how these latent dimensions are combined into a response probability.

A common choice is a *compensatory* MIRT model, in which latent dimensions contribute additively to the logit of the response probability. For simplicity, we use a Rasch-style compensatory formulation,

$$p_{ij} = \sigma\left(\sum_{k=1}^{K} \theta_{ik} - d_j\right), \tag{1}$$

where d_j is an item difficulty parameter. In this formulation, dimensions can compensate for one another: high ability on one dimension may offset lower ability on another because all dimensions contribute to a single additive score before the sigmoid transformation.

An alternative is a *noncompensatory* MIRT model, in which each dimension contributes a separate success probability and the final response probability is their product:

$$p_{ij} = \prod_{k=1}^{K} \sigma(\theta_{ik} - b_{jk}). \tag{2}$$

Here, b_{jk} denotes the difficulty of item j on dimension k. In this formulation, a low dimension-specific success probability reduces the overall probability of a correct response, even when ability is high on another dimension. Thus, the noncompensatory model encodes the assumption that planning success may require multiple conditions to be satisfied jointly, whereas the compensatory model treats latent dimensions as partially substitutable.

3.2.3 Model Selection

To answer our research question about the competency structure of LLM planning, we treat the choice of dimensionality ($K \in \{1, 2, 3, 4\}$) and interaction type (compensatory vs. noncompensatory) as a model-selection problem. We compare candidate specifications using the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), and cross-validated negative log-likelihood (CV NLL). AIC and BIC are standard information criteria for comparing statistical models because they balance likelihood fit against model complexity, which is important because higher-dimensional models can improve in-sample fit simply by adding parameters. We complement these criteria with CV NLL, which directly evaluates predictive performance on held-out responses and provides a check against overfitting in more complex models. As an independent exploratory diagnostic, we also perform Principal Component Analysis (PCA) on the task-level accuracy matrix to assess the aggregate dimensionality of the response data.

Across all candidate models, we use a task-aware parameterization of item difficulty to make the learned difficulties interpretable with respect to the benchmark structure. In the unidimensional case, item difficulty is decomposed as

$$b_j = \alpha_{t(j)} + \epsilon_j,$$

where $t(j)$ denotes the benchmark subtask associated with item j, $\alpha_{t(j)}$ is a task-level difficulty baseline shared by all items from that subtask, and ϵ_j is an item-specific residual. In the multidimensional case, the same decomposition is applied separately for each latent dimension:

$$b_{jk} = \alpha_{t(j)}^{(k)} + \epsilon_j^{(k)}.$$

This formulation represents broad task-level differences explicitly while still allowing items within the same subtask to vary in difficulty.

4 Results

4.1 Model Selection Favors a 2D Noncompensatory Structure

Table 1 details parameter counts, cross-validation metrics, and information criteria across all specifi-

Table 1: Fused held-out and information-criterion summary across IRT specifications. Train NLL is omitted for compactness; best values per column are bolded.

Dims	Type	Params	CV NLL	CV Acc	AIC	BIC
1	-	1,089	0.3571	0.847	21244.7	21459.5
2	Comp	2,178	0.3528	0.847	23032.8	23462.3
2	**Noncomp**	2,178	0.3249	0.861	**20978.2**	**21407.8**
3	Comp	3,267	0.3518	0.846	25070.6	25714.9
3	Noncomp	3,267	0.3208	0.862	22823.8	23468.1
4	Comp	4,356	0.3522	0.846	27170.9	28030.0
4	Noncomp	4,356	**0.3194**	**0.863**	24773.3	25632.4

cations. The 2D noncompensatory model achieves the lowest overall AIC and BIC. Most of the improvement occurs when moving from the 1D to the 2D model. This transition accounts for 85.4% of the total NLL reduction obtained by expanding the model from 1D to 4D, while additional dimensions provide only marginal gains and are penalized for their added complexity. Importantly, the compensatory 2D model offers little improvement over the 1D baseline. This suggests that the two dimensions are not simply interchangeable sources of ability. Instead, weakness in either dimension can limit performance in a way that strength in the other dimension cannot fully offset.

Figure 2 provides an auxiliary check using PCA on the raw response matrix. The first two components explain 84% of the variance, with a clear elbow after the second component, while all remaining components each explain less than 9%. This pattern agrees with AIC and BIC, which favor a two-dimensional noncompensatory model as the simplest model that explains the main structure of the response data. Although CV NLL and CV Acc is lowest for the 4D noncompensatory model, the improvement over the 2D model is marginal. As a result, for the remainder of the analysis, we focus on the 2D noncompensatory model because it offers the best balance between simplicity, interpretability, and predictive performance.

4.2 Interpreting the Two Latent Dimensions

Having selected the two-dimensional noncompensatory model, we next examine what each latent dimension represents. Our interpretation is grounded in the learned parameters of the model: the task-level difficulty profiles $\alpha_{t(j)}^{(k)}$, which show how benchmark subtasks differ across dimensions; and the respondent abilities θ_{ik}, which reveal how each dimension varies across model size and inference condition.

For readability, we denote the task-level difficulty coordinates $(\alpha_t^{(1)}, \alpha_t^{(2)})$ as (b_1, b_2) when discussing task-level plots and tables. Detailed task-level difficulty coordinates are presented in Table 10 in the appendix. Figure 3 visualizes the tasks in the two-dimensional difficulty plane, where color denotes dimensional dominance. Red vectors correspond to tasks with stronger difficulty along Dimension 1, while blue vectors correspond to tasks with stronger difficulty along Dimension 2.

The tasks form two clear groups in this plane. `val`, `prog`, `nexta`, and `app` have larger positive values on b_1 and near-zero or negative values on b_2, indicating that they are primarily difficult along the first dimension. In contrast, `reach`, `land`, and `areach` have larger positive values on b_2, with `areach` emerging as a clear outlier ($b_2 = +4.86$). This pattern indicates that these tasks impose stronger difficulty along the second dimension.

This grouping has a natural theoretical interpretation. Tasks associated with Dimension 1 primarily involve *local transition reasoning*, such as judging whether an action is applicable, predicting the resulting state transition, or validating a proposed

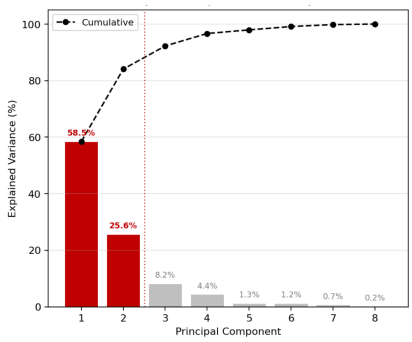

Figure 2: PCA scree plot of the response matrix. PC1 and PC2 explain 84% of variance with an elbow at component 2, independently corroborating the IRT dimensionality selection.

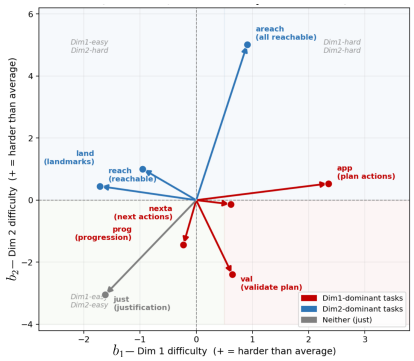

Figure 3: Biplot of task difficulty coordinates in the 2D noncompensatory model.

plan step. Success on these tasks depends on understanding individual actions and their immediate consequences. We therefore interpret Dimension 1 as **operational reasoning**.

By contrast, tasks associated with Dimension 2 involve *global state-space structure*. These tasks require reasoning about reachability, necessary landmarks, and constraints that hold across the planning space, rather than about isolated local transitions. We therefore interpret Dimension 2 as **structural enumeration**.

The remaining tasks occupy intermediate or low-demand regions of the difficulty plane. `nexta`, which requires both local action selection and goal-directed reasoning, lies near the origin, consistent with a task that draws on both competencies without strongly isolating either one. `just` loads weakly on both dimensions, suggesting that it places relatively low demands on either competency for the models considered here.

4.3 External Validation Supports the Semantic Interpretation

The operational–structural interpretation should not rest only on a post-hoc reading of the learned coordinates. We therefore compare the IRT difficulty structure with independent sources of evidence. Specifically, we use three convergent checks: a pre-specified semantic codebook, PDDL-derived problem-structure features, and a downstream failure-mode analysis reported in § 5.3.

Semantic codebook and correlations. Before fitting any IRT model, we constructed a semantic codebook for the eight ACPBench-Hard tasks. The codebook assigns binary tags to each task, describing its reasoning type, output demand, and structural demand independently of model performance. The full task-to-tag mapping is reported in Appendix G. We use these tags as an external validation of the learned IRT dimensions. Specifically, each item inherits the semantic tags of its parent task, and we test whether these tags correlate with the IRT-estimated difficulty coordinates. This lets us ask whether the learned dimensions align with independently defined semantic properties of the benchmark tasks.

The resulting alignments (see Table 2) strongly reinforce the operational-structural interpretation. The structural axis (b_2) shows symmetric opposite associations with `verification_output` ($r = -0.585$) and `requires_search` ($r = +0.585$).

Table 2: Selected direct correlations between task-default semantic tags and IRT difficulty dimensions.

Tag	Dim.	Spearman r	p-value
`requires_counterfactual_consistency`	b_2	-0.645	1.41e-123
`verification_output`	b_2	-0.585	1.03e-96
`requires_search`	b_2	$+0.585$	1.03e-96
`requires_goal_structure_tracking`	b_2	$+0.579$	3.05e-94
`requires_composition_over_steps`	b_2	-0.552	6.23e-84
`requires_goal_structure_tracking`	b_1	-0.531	7.28e-77

On the operational axis (b_1), `requires_goal_structure_tracking` correlates negatively ($r = -0.531$).

PDDL-derived problem-structure features. As a second convergent check, we test whether the structural difficulty dimension aligns with independently measured properties of the planning problems. For each grounded PDDL domain–problem pair, we compute parser-derived features including branching factor, grounded action count, precondition size, goal and initial-state size, object count, and shortest-plan length. The strongest PDDL-derived associations are with the structural dimension (b_2): maximum precondition size ($r = 0.158$), initial branching factor ($r = 0.128$), and mean precondition size ($r = 0.111$). These correlations are modest, but they are directionally consistent with the semantic interpretation: b_2 is more closely related to structural properties of the underlying planning problem, whereas b_1 is not simply tracking parser-visible problem size or superficial naming cues. The full correlation matrix is reported in Appendix H.

Failure-mode analysis. A third line of evidence asks whether the latent dimensions predict distinct generation failures. If b_2 captures structural enumeration difficulty, then high-b_2 items should elicit not only lower accuracy but also characteristic structural failure modes. In § 5.3, we show that high-b_2 items are dominated by externalization failures: the correct label is often linearly decodable from hidden states, yet the generated answer is wrong. This alignment between IRT difficulty and failure topology provides behavioral support for the operational–structural interpretation.

5 Effects of Scale and CoT on Planning

If planning competence reflected a single latent ability, model scaling and test-time reasoning mechanisms should yield uniform improvements. Under the two-factor model, however, these interventions

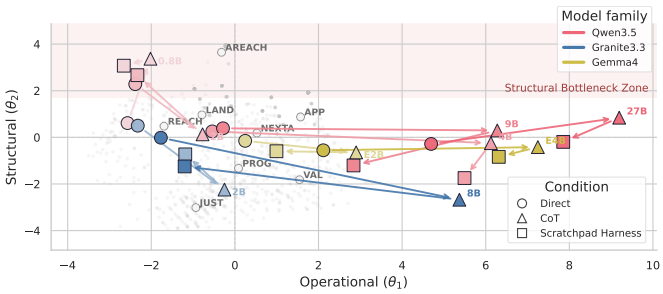

Figure 4: Asymmetric scaling of planning competence under joint calibration. Operational ability (θ_1) scales with model size and prompting, while structural enumeration (θ_2) flattens out.

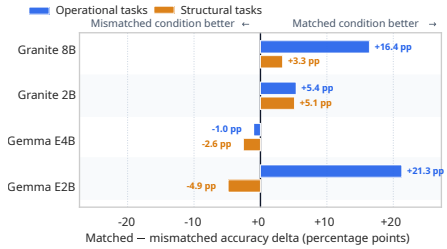

Figure 5: Matched-minus-mismatched accuracy advantage for operational and structural tasks; Negative values indicate the mismatched condition outperformed the matched condition.

may affect the two competencies differently. This section tests that prediction.

5.1 Scale and CoT Improves Operational Reasoning, Not Structural Enumeration

We evaluate three model families under three inference conditions and place all model–condition pairs in the same (θ_1, θ_2) space[2].

Figure 4 plots the resulting trajectories, with arrows connecting Direct → CoT → Scratchpad for each model. The dominant movement is horizontal: as models grow larger and are given increasingly explicit reasoning scaffolds, operational ability (θ_1) increases while structural ability (θ_2) stays flat. The shaded band marks the top quartile of structural difficulty ($b_2 \geq 1.37$), where even the strongest models remain bottlenecked; a regression analysis in Appendix I also supports the visual pattern.

Despite explicit state-traversal scaffolding, scratchpad shows no improvement over CoT. This aligns with evidence that fragmenting reasoning across discrete interaction turns degrades reliability compared to generating a complete reasoning trace in a single pass (Laban et al., 2026). The failure pattern is heterogeneous: Qwen 3.5 adopts tools moderately (use rate 0.619) but produces shallow traversals, whereas Gemma 4 barely invokes them (0.205). Both patterns indicate that reliable scratchpad-based search is not an emergent zero-shot capability.

5.2 Targeted Prompting Partially Helps

The previous section suggests that generic CoT mainly aids operational reasoning. A natural follow-up is to ask whether a prompt designed for structural enumeration can better target the structural dimension. We add a `structural_cot` system prompt that asks the model to identify landmarks, reason about reachability, and assess global path structure before answering. We compare it with generic CoT on a six-task slice: operational tasks (`app`, `prog`, `val`) and structural tasks (`land`, `reach`, `areach`).

Figure 5 provides partial support for dimension-matched prompting. On operational tasks, generic CoT consistently outperforms `structural_cot`, with large matched advantages for Gemma 4-E2B (+21.3 pp) and Granite 3.3-8B (+16.4 pp). On structural tasks, `structural_cot` helps Granite 3.3 (+5.1 pp at 2B and +3.3 pp at 8B), but the effect does not replicate for Gemma 4.

Critically, however, even within Granite 3.3, the structural gain remains substantially smaller. This within-family asymmetry aligns with the scaling and CoT results in § 5.1. The prompt was not the bottleneck—structural difficulty resists improvement from inference-time prompting interventions regardless of how carefully they are targeted.

5.3 Externalization Failures

We next ask whether the two latent dimensions correspond to different generation failures. We construct a probe-output taxonomy across 15,600 Qwen3.5 responses across five sizes, three conditions, and eight tasks. For each response, we extract hidden states across layers and train a low-capacity linear probe to predict ground-truth labels using five-fold cross-validation. Each response is then classified as: *1) latent absent*, when both probe and output are wrong; *2) externalization failure*, when the correct target is linearly decodable but the output is wrong; *3) successful*, when both are correct; or *4) spurious*, when probe and output agree on an incorrect label. Figure 6 shows that

[2]Respondent parameters are estimated via joint calibration with shared item parameters, enabling cross comparison.

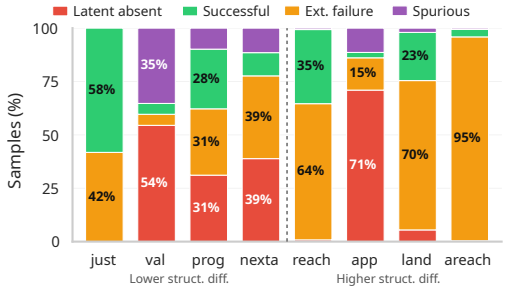

Figure 6: Probe-output failure taxonomy by task, averaged across *Qwen3.5* model sizes and inference conditions, with tasks ordered by absolute structural difficulty.

Table 3: Generative accuracy (Gen Acc), MCQ−Gen gap, and Bool→Gen gap (Bool−Gen), sorted by structural difficulty b_2. Positive gap indicates inflates over Gen; negative indicates Gen exceeds.

Task	b_2	Gen Acc	MCQ−Gen	Bool−Gen
`just`	−2.69	0.931	−0.177	−0.162
`val`	−2.10	0.708	−0.315	−0.038
`prog`	−1.50	0.723	+0.138	+0.191
`reach`	+0.31	0.423	+0.315	+0.477
`land`	+0.79	0.223	+0.531	+0.608
`areach`	+4.86	0.077	+0.623	+0.681
Spearman $\rho(b_2$, *Bool−Gen gap)* $= 0.96$				

externalization failure is the dominant error mode on the structural dimension. It accounts for 45.1% of responses overall, but is much higher on structurally demanding tasks: `areach` reaches 95.5%, `land` 69.9%, and `reach` 63.8%. In contrast, operational tasks show more mixed profiles. For `app`, its extreme operational difficulty ($b_1 = +2.74$) creates a bottleneck before externalization becomes the limiting factor. Thus, even though it carries above-average structural difficulty, its error profile is dominated by latent-absent failures rather than externalization failures.

These results reveal a systematic signature of structural enumeration failure: the bottleneck lies in the mapping from internal representations to explicitly structured generation, not in representational capacity per se, as evidenced by the dissociation between linearly decodable correct labels in hidden states and erroneous generated outputs. Future work on LLM planning should therefore shift focus from simply scaling model size or CoT toward explicitly improving the externalization mechanisms that translate latent competence into correct answers.

5.4 Format Impact on Two Dimensions

The results of § 5.3 suggest that the bottleneck of LLM θ_2 competence lies in translating latent knowledge into structured output. We therefore test whether supplying candidate answers via Boolean or MCQ formats disproportionately rescues structural tasks.

As Table 3 shows, the Bool→Gen gap scales almost linearly with structural difficulty b_2 (Spearman $\rho = 0.96$). Dim2-dominant tasks suffer severe generative collapse: `areach` plunges from 0.808 (Bool, CoT) to 0.069 (Gen), and `land` and `reach` exhibit comparable drops. Operational tasks, by contrast, are format-robust: `just` actually scores higher under Gen (0.985), and `val` shows near-zero gap. MCQ is even more deceptive: `areach` reaches 0.892 under MCQ while remaining near-floor under Gen, a gap of −0.82. A benchmark providing candidate answers would thus severely overestimate structural competence.

Format sensitivity therefore provides an independent signature of the two-dimensional structure: the tasks that load most heavily on the structural dimension are precisely those that benefit most from answer scaffolding.

6 Conclusion

This paper examined whether LLM planning benchmark performance reflects a single general ability or a structured set of underlying competencies. Using IRT-based model comparison across three model families and multiple inference conditions, we find that a two-dimensional noncompensatory model provides the clearest account of the response patterns. The recovered dimensions separate local action-centered reasoning from global structure-centered reasoning, corresponding to *operational reasoning* and *structural enumeration*.

This distinction changes how we interpret progress in LLM planning. Because structural enumeration remains flat across model sizes and inference budgets, closing this gap likely requires interventions beyond scaling or longer reasoning traces.

The broader lesson is that LLM planning should be evaluated not only by how much models improve, but by the structure of that improvement. Moving from aggregate accuracy to diagnostic competence profiles makes this structure visible, and our IRT framing provides a concrete tool for doing so. This perspective can guide future work from asking whether models improve overall toward identifying which planning competencies improve, under what conditions, and why.

7 Limitations

Our findings are subject to two scope limitations that also point to future work.

Model coverage. We jointly calibrate Qwen3.5, Gemma 4, and Granite 3.3, all of which are open-weight models with publicly available checkpoints. We do not evaluate the latest proprietary systems, such as GPT-5.5 or Opus 4.7, due to our commitment to open-source reproducibility and budget constraints. This choice bounds the upper range of model scale we can observe, and it is possible that qualitatively different structural reasoning capabilities emerge at larger scales or under different training regimes. However, the consistency of the two-factor structure and the asymmetric scaling pattern across three independent model families suggests that our main findings are unlikely to be artifacts of a particular model series. Extending the analysis to larger proprietary models, where feasible, remains a natural next step.

Inference scaffolds. Recent progress in agentic frameworks shows that sophisticated inference harnesses, (i.e., integrating tool use, code execution, persistent memory, and multi-step orchestration), can substantially improve LLM performance on complex reasoning tasks, including those that require planning-like behavior (Li et al., 2026). Our scratchpad condition represents a lightweight exploration in this direction. However, it does not approach the sophistication of state-of-the-art agent architectures. One concrete capability our scratchpad lacks is *context compaction*: advanced harnesses can summarize and compress earlier reasoning states when the interaction history approaches the model's maximum context length, allowing structural search to continue beyond what a flat reasoning trace can maintain. This may be particularly relevant for structural enumeration tasks such as reachability and landmark reasoning, where exhaustive state-space exploration can generate reasoning traces that exceed naive context budgets. Consequently, our finding that structural enumeration remains a bottleneck should be understood as characterizing the limits of *generic* reasoning traces and simple scratchpad scaffolds, rather than as a claim that no scaffold can ever address structural planning failures. Note the externalization failure analysis points to the same conclusion: the bottleneck lies in the mapping from latent competence to output, not in the reasoning process itself.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press.

Blai Bonet and Hector Geffner. 2001. Planning as heuristic search. *Artif. Intell.*, 129(1-2):5–33.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025a. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net.

Sadia Sultana Chowa, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raiaan, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. 2026. From language to action: a review of large language models as autonomous agents and tool users. *Artificial Intelligence Review*.

Malik Ghallab, Dana Nau, and Paolo Traverso. 2004. *Automated Planning: theory and practice*. Elsevier.

Divij Handa, Pavel Dolin, Shrinidhi Kumbhar, Tran Cao Son, and Chitta Baral. 2025. Actionreasoningbench: Reasoning about actions with and without ramification constraints. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Weinan He, Canming Huang, Zhanhao Xiao, and Yongmei Liu. 2023. Exploring the capacity of pretrained language models for reasoning about actions and change. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4629–4643.

Malte Helmert. 2006. The fast downward planning system. *J. Artif. Intell. Res.*, 26:191–246.

Jörg Hoffmann, Julie Porteous, and Laura Sebastia. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278.

Sukai Huang, Trevor Cohn, and Nir Lipovetzky. 2025a. Chasing progress, not perfection: Revisiting strategies for end-to-end llm plan generation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 35, pages 204–212.

Sukai Huang, Nir Lipovetzky, and Trevor Cohn. 2025b. Planning in the dark: Llm-symbolic planning pipeline without experts. In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 26542–26550. AAAI Press.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. 2024. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. 2025. Acpbench: Reasoning about action, change, and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26559–26568.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. 2026. ACPBench hard: Unrestrained reasoning about action, change, and planning. In *The Fourteenth International Conference on Learning Representations*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2026. LLMs get lost in multi-turn conversation. In *The Fourteenth International Conference on Learning Representations*.

Junjie Li, Xi Xiao, Yunbei Zhang, Chen Liu, Lin Zhao, Xiaoying Liao, Yingrui Ji, Janet Wang, Jianyang Gu, Yingqiang Ge, Weijie Xu, Xi Fang, Xiang Xu, Tianchen Zhao, Youngeun Kim, Tianyang Wang, Jihun Hamm, Smita Krishnaswamy, Jun Huan, and Chandan Reddy. 2026. Agent harness engineering: A survey.

Nir Lipovetzky and Hector Geffner. 2012. Width and serialization of classical planning problems. In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, Frontiers in Artificial Intelligence and Applications, pages 540–545. IOS Press.

Nir Lipovetzky and Hector Geffner. 2017. Best-first width search: Exploration and exploitation in classical planning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3590–3596. AAAI Press.

Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. 2025. Thoughtterminator: Benchmarking, calibrating, and mitigating overthinking in reasoning models. In *Second Conference on Language Modeling*.

Silvia Richter and Matthias Westphal. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res.*, 39:127–177.

Jendrik Seipp, Álvaro Torralba, and Jörg Hoffmann. 2022. PDDL generators. https://doi.org/10.5281/zenodo.6382173.

Katharina Stein, Daniel Fišer, Jörg Hoffmann, and Alexander Koller. 2025. Automating the generation of prompts for llm-based action choice in pddl planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 35, pages 250–259.

Junfeng Tang, Yuping Yan, Zihan Ye, Zhenshou, Song, Zeqi Zheng, and Yaochu Jin. 2025. Think small, plan smart: Minimalist symbolic abstraction and heuristic subspace search for llm-guided task planning. *Preprint*, arXiv:2501.15214.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo Hernandez, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. On the planning abilities of large language models-a critical investigation. *Advances in neural information processing systems*, 36:75993–76005.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Karthik Valmeekam, Kaya Stechly, Atharva Gundawar, and Subbarao Kambhampati. 2025. A systematic

evaluation of the planning and scheduling abilities of the reasoning model o1. *Trans. Mach. Learn. Res.*, 2025.

Pulkit Verma, Ngoc La, Anthony Favier, Swaroop Mishra, and Julie A Shah. 2025. Teaching llms to plan: Logical chain-of-thought instruction tuning for symbolic planning. *arXiv preprint arXiv:2509.13351*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2026. When more is less: Understanding chain-of-thought length in LLMs. In *The Fourteenth International Conference on Learning Representations*.

Fengli Xu, Qianyue Hao, Chenyang Shao, Zefang Zong, Yu Li, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, and 1 others. 2025. Toward large reasoning models: A survey of reinforced reasoning with large language models. *Patterns*, 6(10).

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Benchmark Scope and Generality

Our conclusions are derived from ACPBench-Hard, but ACPBench-Hard should not be viewed as an arbitrary natural-language benchmark. Like other prominent LLM planning benchmarks such as PlanBench (Valmeekam et al., 2023a) and AutoPlanBench (Stein et al., 2025), it is built on a PDDL/IPC-style symbolic planning substrate (Seipp et al., 2022); these benchmarks differ primarily in the scaffolding layer through which formal planning instances are exposed to LLMs, rather than in the underlying notion of planning itself. ACPBench-Hard is the most recent, and is especially suitable for our analysis because it provides both domain diversity and task diversity: it covers 13 PDDL domains and exposes them through eight open-ended reasoning tasks spanning action-level, state-level, and plan-level queries, rather than only through end-to-end plan generation. We therefore interpret our results as evidence about latent competencies in PDDL-based symbolic planning evaluation. Additional benchmarks would be valuable as scaffold-robustness checks, but they would primarily test robustness to alternative LLM-facing interfaces rather than replace the underlying formal planning substrate on which these evaluations are based.

B On the Label *Structural Enumeration*

In classical AI, planning is synonymous with search over a state space (Bonet and Geffner, 2001). If we were to label the second dimension *search complexity*, it would misleadingly suggest that it captures all of planning-as-search, and that operational reasoning lies outside the scope of search. We therefore adopt the term structural enumeration to denote specifically the ability to engage with the combinatorial exploration of the state space, and clarifies that it is one facet of planning-as-search, not a proxy for the entire activity.

C Cross-Family Competence Trajectories

§ 5.1 of the main text reports that the asymmetric scaling pattern, i.e., operational ability improving with scale and test-time reasoning mechanisms while structural enumeration remains comparatively flat, is consistent across model families under joint calibration. Here we provide the full per-family trajectory plots as well.

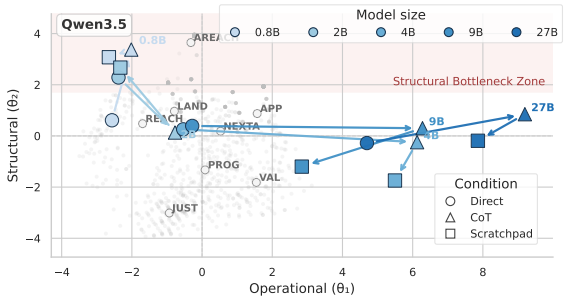

Figure 7: Asymmetric scaling of planning competence. Pale points: item difficulties on the two latent dims. Colored trajectories: ability estimates under Direct → CoT → Scratchpad. Operational ability (Dim 1) increases with scale ($\beta_{\log \text{size}} = 2.72$) and prompting (CoT 4.02, Scratchpad 2.37); structural enumeration (Dim 2) does not (-0.63, -0.49, -0.50). The shaded zone marks the top quartile of structural difficulty, where scale provides little gain.

All three figures share the same jointly calibrated coordinate system: item difficulties (b_1, b_2) are estimated once from the stacked response matrix span-

ning all families, and respondent abilities (θ_1, θ_2) are placed in that common space. This ensures that horizontal movement corresponds to gains in operational reasoning and vertical movement to gains in structural enumeration across all families.

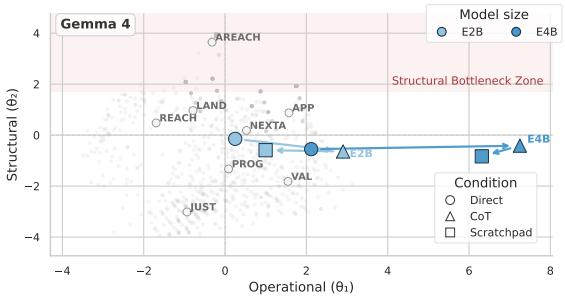

Figure 8: Gemma 4 competence trajectories under joint calibration. Pale points show shared item difficulty coordinates; colored points show Gemma 4 ability estimates for Direct, CoT, and Scratchpad conditions. The dominant movement is rightward along the operational axis; structural ability remains comparatively flat. The shaded band marks the top quartile of structural difficulty ($b_2 \geq 1.71$), where all conditions remain bottlenecked.

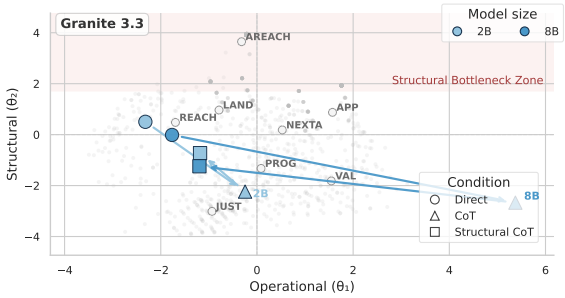

Figure 9: Granite 3.3 competence trajectories under joint calibration. Pale points show shared item difficulty coordinates; colored points show Granite 3.3 ability estimates for Direct, CoT, and Structural CoT conditions. As with Qwen and Gemma, the primary movement is rightward along the operational axis, with limited structural gains.

C.1 Summary of Cross-Family Evidence

Across Qwen3.5 (dense 0.8B–27B sweep), Gemma 4 (E2B, E4B), and Granite 3.3 (2B, 8B), the same qualitative pattern holds under joint calibration: model scale and reasoning-style prompting primarily improve operational ability, while structural enumeration remains a persistent bottleneck. The absolute ability levels differ across families, but the direction of improvement is stable. This cross-family consistency suggests that the two-dimensional noncompensatory structure and its asymmetric response to scale and prompting are not artifacts of a single model series, but a more general property of the LLM planning behavior captured by ACPBench Hard.

D Inference Condition System Prompts

This appendix documents the exact system prompts and generation configurations for the three core inference conditions used in the main text: `no_think` (Direct), `think` (Chain-of-Thought), and `think_tools_state_traversal` (Scratchpad with state traversal). All prompts and configurations are defined in `src/acpbench_evaluation/config.py`.

D.1 Direct (`no_think`)

Generation configuration.

```
GenerationConfig(temperature=0.0,
max_tokens=8192,
initial_max_tokens=2048,
enable_thinking=False,  use_tools=False,
repetition_penalty=1.4,      top_p=0.95,
topk=20, stop=[])
```

System prompt.

```
You are an expert in classical planning and
    reasoning about actions,
state transitions, and goals. Answer the
    question precisely following
the specified output format. In think mode, keep
     reasoning finite and
search-oriented: enumerate candidates, check
    them, and stop once the
answer is determined. Wrap the reasoning trace
    in <think>...</think>.
Be concise and avoid repetition.
```

The system prompt is followed by zero to two few-shot examples showing question→answer pairs (without reasoning traces) and the task-specific format instruction (see Appendix E). The test question is appended as a user message.

D.2 Chain-of-Thought (`think`)

Generation configuration.

```
GenerationConfig(temperature=0.0,
max_tokens=32768,  enable_thinking=True,
use_tools=False,  thinking_budget=16384,
repetition_penalty=1.05,      top_p=1.0,
topk=0, stop=[])
```

System prompt. Identical to the Direct condition system prompt shown above.

Few-shot example structure. The few-shot examples for think include explicit <think>...</think> reasoning traces:

```
Example 1:
<context>...</context>
<question>...</question>
<format_instruction>

<think>
[step-by-step reasoning: enumerate candidates, check preconditions,
verify effects, determine answer]
</think>

[final answer]
```

D.3 Scratchpad with State Traversal (think_tools_state_traversal)

Generation configuration. GenerationConfig(temperature=0.0, max_tokens=32768, enable_thinking=True, use_tools=True, max_tool_iterations=40, thinking_budget=16384, repetition_penalty=1.25, top_p=1.0, topk=0, stop=[])

System prompt.

```
You are an expert in classical planning and reasoning about actions,
state transitions, and goals.

You have access to scratch tools:
- write_file(filename, content): overwrite or create a scratch file
- read_file(filename): read a scratch file
- list_files(): list scratch files

Use the scratch space to externalize explicit state traversal for
structural planning questions. Maintain these files as you work:
frontier.txt, visited.txt, dead_ends.txt, and backtrack_log.txt.

Follow a bounded search loop: initialize the current STATE summary,
enumerate one BRANCH or candidate transition at a time, mark each EXPLORE
step in scratch, record impossible or exhausted branches as DEAD_END, and
append BACKTRACK notes when you return to an earlier frontier. Prefer
short, auditable scratch updates over long free-form reasoning.

Stop once the answer is determined. Keep the final answer outside <think>
and follow the required task output format exactly.
```

Tool-call format. Each tool step consists of brief reasoning inside <think>...</think> followed by exactly one XML tool call:

```
<think>brief reasoning about what to write or read next</think>
<tool_call><function=write_file><parameter=frontier.txt>...</parameter></function></tool_call>
```

E Task Format Instructions

Every task-specific user prompt appends a short format instruction after the question text. The full set of instructions is defined in `src/acpbench_evaluation/config.py` (`TASK_FORMAT_INSTRUCTIONS`).

Generative-format instructions.

Boolean-format instructions. All eight tasks use the same instruction: "Answer only yes or no."

MCQ-format instructions. All eight tasks use the same instruction: "Answer only the single correct option label (A, B, C, or D)."

F Inference Infrastructure and Model Serving

All models were served with VLLM (Kwon et al., 2023) using an OpenAI-compatible chat-completion endpoint. The server ran on a single node with four NVIDIA RTX 5090 GPUs (32 GB each), and we enabled prefix caching to accelerate shared-prompt evaluations. Generation parameters were controlled via the chat-completion API; the values used in each behavioural condition are listed in Table 5.

Key design choices. Temperature. $t = 0$ (greedy decoding) was used throughout because stochasticity in tool-use and reasoning traces produced divergent multi-turn trajectories.

Thinking budget. The thinking_budget of 16,384 tokens caps the <think>...</think> phase independently of max_tokens, preventing unproductive reasoning loops while still accommodating the longest ACPBench-Hard instances.

Repetition penalties and sampling. no_think uses a higher repetition penalty (1.40) with nucleus sampling ($p = 0.95$, $k = 20$) to counteract degenerate token loops common in small models forced

Table 4: Task format instructions for generative (free-text) questions.

Task	Format instruction
app	Each action starts with an opening parenthesis and ends with closing parenthesis. Provide only the actions.
areach	Each action starts with an opening parenthesis and ends with closing parenthesis. Provide one action or None.
just	(none)
land	Provide only the ground proposition or None.
nexta	Each action starts with an opening parenthesis and ends with closing parenthesis. Provide only the action.
prog	Provide only the two lists with the ground propositions.
reach	Provide one proposition or None.
val	Provide only the index of the action.

Table 5: Per-condition generation configuration. All conditions use temperature $t = 0$ (deterministic greedy decoding). The `thinking_budget` caps the `<think>` phase independently of `max_tokens`.

Condition	max_tokens	think_budget	Tools	Rep. penalty
Direct	8,192	—	off	1.40
Cot	32,768	16,384	off	1.05
Scratchpad	32,768	16,384	on	1.25

to answer planning questions without intermediate reasoning. CoT and scratchpad conditions use pure greedy decoding ($p = 1.0$, $k = 0$) and lower repetition penalties (1.05–1.25), as the reasoning trace itself provides a natural guard against loops. Scratchpad conditions use a slightly higher penalty (1.25) as an additional safeguard against tool-call repetition cycles.

Tool use. Tool-enabled runs used `tool_choice="auto"` and a multi-turn loop (up to 40 tool-call iterations), terminating when the model produced a final answer without a tool call.

All model-specific vLLM settings (tensor-parallel size, GPU memory fraction, family-specific parsers) and the orchestration scripts (`run_ablation.sh`, etc.) are available in the accompanying code repository.

G Semantic Codebook

This appendix documents the full semantic tagging codebook used to assign interpretable labels to ACPBench tasks. Each task receives a set of task-default binary tags drawn from three complementary categories: *Reasoning-type*, which captures the cognitive operation the model must perform; *Output-demand*, which captures how the answer must be expressed; and *Structural-demand*, which captures the computational structure that makes the problem difficult. These tags were specified before any IRT model fit and are intended as transparent task-default semantic annotations rather than manually reviewed item-by-item labels.

The main text does not correlate every tag in this appendix directly with the IRT dimensions. Instead, the main-text validation analysis uses a reduced set of four focal semantic proxies derived from this broader codebook: verification, construction, requires search, and composition over steps. That proxy layer is intentionally coarser than the full codebook presented here. The purpose of this appendix is therefore transparency and auditability: it records the full semantic inventory from which the main-text proxy analysis is derived.

G.1 Tag Glossary

The 17 tags enumerated in Table 6 span three complementary semantic families. Reasoning-type tags capture *what* cognitive operation the model must perform, for example checking local state facts versus reasoning about global reachability. Output-demand tags capture *how* the answer must be expressed, for example verification versus construction, single-witness reporting versus set enumeration, and negative unreachability certification. Structural-demand tags capture the computational properties that make a problem difficult, for example search, multi-step composition, counterfactual consistency, and goal-structure tracking.

G.2 Task-to-Tag Mapping

Each ACPBench task is assigned a fixed set of semantic tags based on its canonical definition. The mapping is shown in Table 7.

Tags are assigned at the task-default level, which means each item inherits the semantic tag pattern of its parent task. This codebook was specified before any IRT model fit was conducted, ensuring that the semantic dimensions used for interpretation are not confounded by post-hoc knowledge of model behavior.

H PDDL-Derived Structural Properties

This appendix reports the complete correlation matrix between 18 PDDL-derived properties and the two IRT difficulty dimensions (b_1, b_2) from the

Table 6: Semantic tag definitions with their categories.

Tag	Definition	Category
local_state_check	Reasoning about the truth of facts in a single state	Reasoning-type
state_transition	Computing the effects of applying an action	Reasoning-type
plan_validation	Auditing a proposed plan for correctness	Reasoning-type
plan_justification	Determining whether plan steps are necessary	Reasoning-type
global_reachability	Reasoning about whether facts can hold in any reachable state	Reasoning-type
action_reachability	Reasoning about whether an action can ever become applicable	Reasoning-type
landmark_reasoning	Identifying facts that must hold in all valid plans	Reasoning-type
next_action_construction	Constructing a goal-directed next action	Reasoning-type
verification_output	Answer requires verifying a given claim	Output-demand
construction_output	Answer requires constructing a novel answer	Output-demand
single_witness_output	Answer is a single fact, action, or index	Output-demand
set_enumeration_output	Answer is a set or list	Output-demand
negative_unreachability_case	Answer requires certifying that no valid witness exists for a reachability-style query	Output-demand
requires_search	Requires exploring the state space	Structural-demand
requires_composition_over_steps	Requires composing across multiple plan steps	Structural-demand
requires_counterfactual_consistency	Requires consistent reasoning about counterfactuals	Structural-demand
requires_goal_structure_tracking	Requires tracking the relationship of facts to goals	Structural-demand

Table 7: Task-to-tag mapping for all eight ACPBench tasks. Tags are assigned at the task-default level.

Task	Description	Active Tags
app	Applicable-action enumeration	local_state_check, state_transition, verification_output, set_enumeration_output
prog	Apply local effects and report delta	local_state_check, state_transition, verification_output
val	Audit plan, find first invalid step	plan_validation, verification_output, requires_composition_over_steps, requires_counterfactual_consistency
just	Validate necessary steps, simplify a plan	plan_justification, verification_output, requires_composition_over_steps, requires_counterfactual_consistency
reach	Which facts can hold in reachable states	global_reachability, single_witness_output, negative_unreachability_case, requires_search, requires_goal_structure_tracking
areach	Can an action ever become applicable	action_reachability, single_witness_output, negative_unreachability_case, requires_search, requires_goal_structure_tracking
land	Construct a non-trivial landmark	landmark_reasoning, construction_output, single_witness_output, requires_search, requires_goal_structure_tracking
nexta	Construct a goal-directed next action	next_action_construction, construction_output, single_witness_output, requires_search, requires_composition_over_steps

Track B natural experiment (66 PDDL domain–problem pairs, 1040 items). All properties are computed from grounded PDDL representations independently of any LLM responses. These are observational correlations, not causal evidence.

The strongest correlations are the IRT-internal task offsets ($\alpha_{\text{task},1}$ with b_1 at $r = 0.873$, $\alpha_{\text{task},2}$ with b_2 at $r = 0.827$), confirming that the task-level difficulty structure recovered by IRT aligns with domain-level planning features. Among item-level residuals, ε_2 (structural discrimination) correlates with its domain counterpart at $r = 0.588$. Domain structural properties (branching factor, precondition size) show modest alignment with b_2 in the expected direction.

Table 8: Spearman correlations between PDDL properties and IRT difficulty dimensions (b_1, b_2). All correlations computed over $n = 1040$ items except where noted. Properties grouped by type.

Property	b_1 (r)	b_2 (r)
Task-level IRT offsets (α_{task})		
$\alpha_{\text{task},1}$	0.873	−0.079
$\alpha_{\text{task},2}$	0.035	0.827
Item-level IRT residuals (ε)		
ε_1	0.291	0.158
ε_2	0.131	0.588
Domain structural properties		
initial_branching_factor	−0.038	0.128
max_precondition_size	0.073	0.158
mean_precondition_size	0.068	0.111
grounded_action_count	−0.051	0.073
shortest_plan_length	0.078	−0.052
Problem size properties		
goal_fact_count	−0.045	−0.032
init_fact_count	0.033	0.094
object_count	−0.021	0.069
sample_count	0.001	−0.063
task_count	−0.052	−0.062
Problem-name numeric features		
problem_name_numeric_max	−0.058	−0.006
problem_name_numeric_sum	−0.052	0.013
problem_name_numeric_token_count	0.014	0.011

H.1 Full Correlation Table

These properties are parser-backed from PDDL specifications. The IRT-internal task offsets ($\alpha_{\text{task},1}$ and $\alpha_{\text{task},2}$) serve as a validity check: their high correlation with the corresponding latent dimensions confirms that the IRT model has successfully separated task-level from item-level difficulty. The domain structural properties—particularly *initial_branching_factor* and *max_precondition_size*—show modest positive correlations with b_2, consistent with the interpretation that b_2 captures structural complexity. The problem-name numeric features show negligible correlations across both dimensions, confirming that surface-level lexical cues in domain names do not drive the latent structure.

I Scaling Regression

To quantify the trajectory pattern in Figure 4, we fit a simple linear regression to the jointly calibrated ability estimates. Each observation is a model–condition pair. The dependent variable is the estimated ability on one latent dimension, θ_{id}, and the predictors are centered log model size and inference-condition indicators:

$$\begin{aligned}\theta_{id} = \alpha_d &+ \beta_{d,\text{size}} \log N_i \\ &+ \beta_{d,\text{CoT}}\, \mathbb{I}[\text{CoT}] \\ &+ \beta_{d,\text{Scratchpad}}\, \mathbb{I}[\text{Scratchpad}] + \epsilon_{id}\end{aligned} \tag{3}$$

where $d \in \{1, 2\}$, N_i is the parameter count of model i, and Direct inference is the reference condition. We fit the regression separately for the operational dimension (θ_1) and the structural dimension (θ_2).

Table 9: Regression of jointly calibrated ability estimates on model size and inference condition. Operational ability increases with model size and reasoning-style prompting, while structural enumeration shows no comparable positive effect. Direct inference is the reference condition.

Predictor	Operational θ_1	Structural θ_2
Centered log model size	$+2.72$	-0.63
CoT	$+4.02$	-0.49
Scratchpad	$+2.37$	-0.50

The coefficients mirror the qualitative pattern in Figure 4. Model size, CoT, and scratchpad inference all have positive effects on operational ability, whereas their effects on structural enumeration are near zero or negative. This supports the interpretation that scaling and reasoning scaffolds primarily move models along the operational axis rather than the structural axis.

J IRT Model Diagnostics

J.1 Task (Item) Difficulty Structure

Note: `areach` is extremely hard on Dim2 only; `just` is easiest overall (both dimensions easy). Dim1 correlates with planning/local transitions; Dim2 with global state-space reasoning.

Key observations:

- **app** is a pure Dim1 (general planning) challenge. It anchors the rotation: the model is constrained so that ‘app‘ has zero Dim2 load, which fixes the rotational ambiguity of the 2D solution.

Table 10: Task-level mean difficulties in the 2D noncompensatory model. Positive values indicate harder-than-average on that dimension. Tasks separate into two groups: those requiring local transition reasoning (Dim1) and those requiring global state-space reasoning (Dim2).

Task	b_1 (Dim1)	b_2 (Dim2)
`app`	+2.74	+0.43
`nexta`	+0.98	-0.09
`val`	+0.47	-2.10
`prog`	+0.19	-1.50
`areach`	+0.05	+4.86
`land`	-1.24	+0.79
`just`	-1.28	-2.69
`reach`	-1.90	+0.31

- **areach** has an extreme Dim2 difficulty (θ_2 = 4.86), placing it at the floor for virtually all model-condition pairs. It provides little discriminative power for Dim2 ability estimation.

- **just** and **reach/land** form opposite poles: just is near-trivial, while reach and land require structural traversal reasoning.

- Tasks splitting across dimensions (land, areach) validate the noncompensatory structure: you cannot substitute excess Dim1 ability for Dim2 deficits.

J.2 IRT Training Configuration

All IRT models are trained via maximum a posteriori (MAP) estimation with the following priors and constraints:

- **Gaussian priors:** $\sigma_u = 2.0$ for respondent ability residuals, $\sigma_\alpha = 1.0$ for task effect parameters, $\sigma_\varepsilon = 1.0$ for item residual parameters. Regression coefficients receive a weak prior with $\sigma_B = 10.0$.

- **Anchor constraint:** The `app` task (index 0) is constrained to have zero difficulty on the final latent dimension, enforced with a penalty weight of 100.0.

- **Optimizer:** Adam with learning rate 0.03, combined with a ReduceLROnPlateau scheduler (patience= 500 epochs, decay factor= 0.5).

- **Training budget:** Maximum 8,000 epochs with early stopping triggered after 2,000 epochs of no validation improvement.

- **Cross-validation:** 5-fold cross-validation with random respondent-by-item cell holdout, stratified by item to ensure each item appears in every fold.

- **Latent regression specification:**

$$\theta_{id} = \mathbf{B}_d^\top \mathbf{x}_i + u_{id}$$

  where $\mathbf{x}_i$ includes centered log model size, model family indicators (Gemma 4, Granite 3.3; Qwen3.5 as reference), and inference condition indicators (CoT, Scratchpad; Direct as baseline), and u_{id} is a respondent-specific residual.