Can a Language Model Plan, or Is It Copying Plans It Has Seen?

An Enhanced Review Organised Around the Output-Only Test

Hubert Yakpe Trolindo and CAP 6671 group members
University of Central Florida
Orlando, FL, USA
hu881229@ucf.edu

Abstract

We ask whether today's language and reasoning models can be called planners in the classical sense, judging only from what they output, and what output-only test would separate a valid plan from plan-shaped text. We review the three core readings and eight further papers, and group them by the claim they make: that LLMs retrieve plan-shaped text, that reasoning models show a partial and improving capability, that planning is a property of a system in which a sound component guarantees validity, and that the model's useful role is to write the planner rather than the plan. The decisive evidence is a contrast: classical planners are invariant to renaming, while every language model studied loses validity under obfuscated names. Standard LLMs collapse to near zero on Mystery Blocksworld; reasoning models keep between about a third and nine-tenths of their ordinary performance. We conclude that current models are not planners in the classical sense, but that "copying" no longer describes the best of them. We propose a renaming-invariance battery that can be run on outputs alone, and we compare this review with our earlier report.

CCS Concepts

• **Computing methodologies** → **Planning and scheduling**; *Natural language processing*.

Keywords

LLM planning, PDDL, PlanBench, Mystery Blocksworld, obfuscation, LLM-Modulo, generalized planning, large reasoning models

1 The Central Question and its Scope

A classical planner takes a PDDL task (objects, initial state, goal and action schemas with preconditions and effects) and searches it for a sequence of actions that provably reaches the goal [Helmert 2006; McDermott et al. 1998]. A language model instead completes a natural-language description into steps that *look* like a plan, with no guarantee that they are executable. Validity is therefore checked separately, by a plan validator such as VAL [Howey et al. 2004]. This separates two questions that are easy to conflate: does a system produce plan-shaped text, and is that text a valid plan?

Central question. *Judged only from their outputs (whether a plan is valid, whether it stays valid when names are scrambled, and whether it holds up as problems grow), can today's language and reasoning models be called planners in the classical sense? And what output-only test would separate a real plan from plan-shaped text?*

```
(:action pick-up
 :parameters (?x)
 :precondition (and
   (clear ?x) (ontable ?x)
   (handempty))
 :effect (and (holding ?x)
   (not (ontable ?x))
   (not (clear ?x))
   (not (handempty))))
```

```
(:action overcome
 :parameters (?x)
 :precondition (and
   (province ?x) (harmony ?x)
   (craves))
 :effect (and (pain ?x)
   (not (harmony ?x))
   (not (province ?x))
   (not (craves))))
```

Figure 1: The obfuscation probe. Left: Blocksworld `pick-up`. Right: the same schema in Mystery Blocksworld (deceptive names; example from Du et al. [2026]). The structure is identical, so a planner's search and VAL's verdict are unchanged; only the lexical cues a language model could exploit are gone.

In scope. Deterministic, fully observable classical planning, with PDDL or a faithful natural-language rendering of it. We cover direct plan generation by LLMs and large reasoning models (LRMs), with zero-shot, few-shot and chain-of-thought prompts. We also cover three hybrids in which a model produces a formal artifact (a PDDL problem file or a program) or works with an external validator. The evidence is validity scored by an external validator on ordinary and obfuscated domains, on unseen instances and at larger sizes. **Out of scope.** Claims about what happens inside the model (mechanistic interpretability); embodied, web or tool-use agents whose dynamics are not formally specified; and planning under uncertainty. Where a paper's claim needs evidence about internal mechanisms, we say so rather than treat it as settled by outputs.

This is why obfuscation is the central probe (Figure 1): renaming leaves every precondition and effect intact, so any change in validity must come from the model's reliance on names.

2 The Evidence Base

Core readings. PlanBench [Valmeekam et al. 2023a] is the benchmark suite: 600 Blocksworld instances, Logistics, and Mystery Blocksworld, in which action and predicate names are replaced by unrelated words ("deceptive") or random strings ("randomized"). The *Critical Investigation* [Valmeekam et al. 2023b] uses it to test GPT-family models autonomously and in "LLM-Modulo" roles. Kambhampati et al. [2025] (we cite and read arXiv v1) report the

same ordinary-versus-obfuscated contrast for LRMs (o1, DeepSeek R1).

Further papers (eight). These are contrasting architectures: LLM-Modulo [Kambhampati et al. 2024], LLM+P [Liu et al. 2023] and generalized planning by program synthesis [Silver et al. 2024]. There are two independent tests of the same claim on fresh tasks: Corrêa et al. [2025] and Goebel and Zips [2025]. Two are follow-ups that attack obfuscation directly or reframe the LLM's role: Du et al. [2026] and Katz et al. [2026]. One is a survey that standardises metrics: Wei et al. [2025]. We also use Stechly et al. [2024] on chain-of-thought (CoT) prompting. Table 1 lists the conditions behind every number we compare, as the topic brief requires.

3 Synthesis: Four Claims About the Question

We group the literature by what it claims about the central question, not by paper. Figure 2 shows the evidence that all four claims must explain, and Figure 3 shows which component guarantees validity under each.

3.1 Claim A: LLMs retrieve plan-shaped text

Valmeekam et al. [2023b] and Kambhampati et al. [2024] argue that LLMs do "approximate retrieval" rather than planning. The argument rests on a manipulation that cannot affect a sound planner: renaming. GPT-4 produces valid plans for 34.6% of zero-shot Blocksworld instances but 0.16% under deceptive names and 0% under random strings. Newer models in Kambhampati et al. [2024] show the same collapse (Claude-3 Opus falls from 59.3% to 0%). Three supporting observations come from the same studies. Chain-of-thought barely moves the result (35.6% in Blocksworld), and Stechly et al. [2024] find that CoT gains depend on demonstrations that closely match the problem class. Even under delete- and precondition-relaxed checking, many GPT-4 plans still fail to reach the goal [Valmeekam et al. 2023a]. Success does not track optimal plan length the way a planner's would. Validity on ordinary Blocksworld is therefore not sufficient evidence of planning: the same model fails the isomorphic task.

3.2 Claim B: reasoning models have a partial, improving capability

The second claim comes, in part, from the same authors. Kambhampati et al. [2025] report that o1-preview and R1 "saturate" Blocksworld (97.8%, 97%) and are "the first models to make non-trivial progress" on Mystery Blocksworld (52.8%, 42.6%), at a much higher cost ($42.12 per 100 instances for o1-preview versus $0.65 for GPT-4o). They do not conclude that LRMs plan. LRMs are "better generators with a higher density of correct solution guesses", and post-training is described as "compiling the verification signal" into the generator. Larger instances and unsolvable problems still defeat them. Corrêa et al. [2025] go further on freshly generated tasks (which reduces contamination): GPT-5 solves 205 of 360 tasks against LAMA's 204, and under random-string names still solves 152, with Gemini 2.5 Pro dropping only from 155 to 146. A model that merely copied plans it had seen should not keep 59–94% of its performance on fresh, renamed tasks. The claim that best fits the data is therefore not "copying" but partial, renaming-sensitive competence.

3.3 Claim C: planning is a property of the system

A third group stops asking whether the LLM plans and makes validity a property of the whole system. In LLM+P [Liu et al. 2023] the LLM writes only the PDDL problem file; Fast Downward searches. Success rises from 9.3% (LLM as planner) to 45.7%, but falls to 0% without an in-context example, and the domain file is written by humans. Failures are translation errors, such as missing initial conditions, not search errors. LLM-Modulo [Kambhampati et al. 2024] keeps the LLM as the plan proposer and puts sound critics (for example VAL) in a loop. Back-prompting lifts GPT-4 to 82% on previously failed Blocksworld instances, but only to 10% on Mystery Blocksworld, because the critic can reject but not create good candidates. Du et al. [2026] push this further with a symbol-agnostic pipeline (constraint extraction from the PDDL, STRIPS simulation, VAL, four targeted repair strategies) and reach 46–48% on Mystery Blocksworld for GPT-4, GPT-4o and o1-mini alike. In all three, every returned plan is guaranteed valid *by the external component*, so the success rate measures how often the LLM supplies usable material, not whether it plans.

3.4 Claim D: the LLM should write the planner, not the plan

Silver et al. [2024] ask GPT-4 for a Python program that solves a whole domain and score it on held-out tasks that are larger than the training tasks. This is a stronger output-only test of generalisation than single instances. GPT-4 is "a surprisingly powerful generalized planner" (mean 62% of held-out tasks over seven domains; results are typically all-or-nothing per run). The same paper also contains the most direct obfuscation evidence for program synthesis: with all names replaced by identifiers, the mean falls to 4%. Classical planners and the PG3 generalized planner would be unaffected. Katz et al. [2026] generalise this into a research agenda: use LLMs at *construction time* to generate search components, PDDL models or policies, verify them, and then run them without the LLM. They note that frontier models "approach the coverage of LAMA" yet still degrade under obfuscation. On this view the question "can the LLM plan?" is replaced by "can it produce a verifiable planner?"

4 Assessment of the Evidence

Table 2 traces each main claim to its experiment and baseline, and states what the result would look like if the claim were wrong.

Disagreement 1: how large is the obfuscation gap? Kambhampati et al. [2025] call 52.8% on Mystery Blocksworld "a long way" from robustness. Corrêa et al. [2025] find frontier models "competitive" after renaming. The gap is partly in the setups. PlanBench has been public since 2022, while Corrêa et al. generate new tasks. Deceptive renaming (meaningful but misleading words) is harsher than random strings. The domains differ (Blocksworld only versus eight IPC domains), and so do the models (2024 versus 2025). The two results agree on the direction (every model loses validity and the planner does not) and differ on the size of the drop.

Disagreement 2: does verification beat model capacity? Du et al. [2026] conclude that "systematic validation and repair mechanisms play a more decisive role than raw model capacity", citing the convergence of three models to 46–48%. LLM-Modulo with VAL alone reached only 10% on Mystery Blocksworld. The difference

Table 1: Evidence base. "Ord." and "Obf." give the percentage of valid plans (or tasks solved) under ordinary and obfuscated names for the same setup. All rates are external-validator scores. Numbers are from the cited version and are not comparable across rows with different benchmarks.

Source	Model / system	Domain, condition, prompting	Ord.	Obf.	External help in the result
Valmeekam et al. [2023b]	GPT-4	PlanBench BW vs. Mystery BW (deceptive), NL, zero-shot, 600 inst.	34.6	0.16	None; VAL scores only. Randomized names: 0%. CoT: 35.6 vs. 9.0
Kambhampati et al. [2024]	Claude-3 Opus; GPT-4o	Same, zero-shot	59.3; 35.5	0; 0	None
Kambhampati et al. [2025] (v1)	o1-preview; R1; o1-mini	Same, zero-shot	97.8; 97; 56.6	52.8; 42.6; 19.1	None at inference; verifiers used in post-training
Valmeekam et al. [2023b], reported in Kambhampati et al. [2024]	GPT-4 + VAL back-prompting	50 *previously failed* inst. per domain, ≤15 rounds; BW / Logistics / Mystery BW	82 / 70	10	VAL finds errors; LLM proposes all plans
Corrêa et al. [2025] (v1)	GPT-5; Gemini 2.5 Pro; R1; LAMA	360 freshly generated IPC-2023 tasks, 8 domains; random-string names	56.9; 43.1; 43.6; 56.7	42.2; 40.6; 25.8; 56.7	None for LLMs; LAMA is a planner
Du et al. [2026]	GPT-4; GPT-4o; o1-mini + pipeline	PlanBench Mystery BW, 600 inst., budget b=3	–	46.2; 46.2; 48.0	STRIPS simulator, VAL, targeted repair. Baselines 0.16–19.1 taken from earlier work
Liu et al. [2023]	GPT-4 → PDDL → Fast Downward	7 IPC domains, 20 tasks each, with one example (problem, PDDL)	45.7 (69.3†)	n/a	Planner does all search; human-written domain file. LLM-as-planner: 9.3
Silver et al. [2024]	GPT-4 writes Python	7 PDDL domains, held-out tasks larger than training; "No Names" ablation	61.6	4.4	Debugging feedback on training tasks only
Goebel and Zips [2025]	o1 vs. Fast Downward	5 PDDL domains, 93 tasks	63.4	n/a	None; planner reaches 97.9

†Counting the sub-optimal planner alias where the optimal one timed out. Mean of the seven per-domain rates in Table I of Liu et al. [2023]; Silver means are computed from their Table 1.

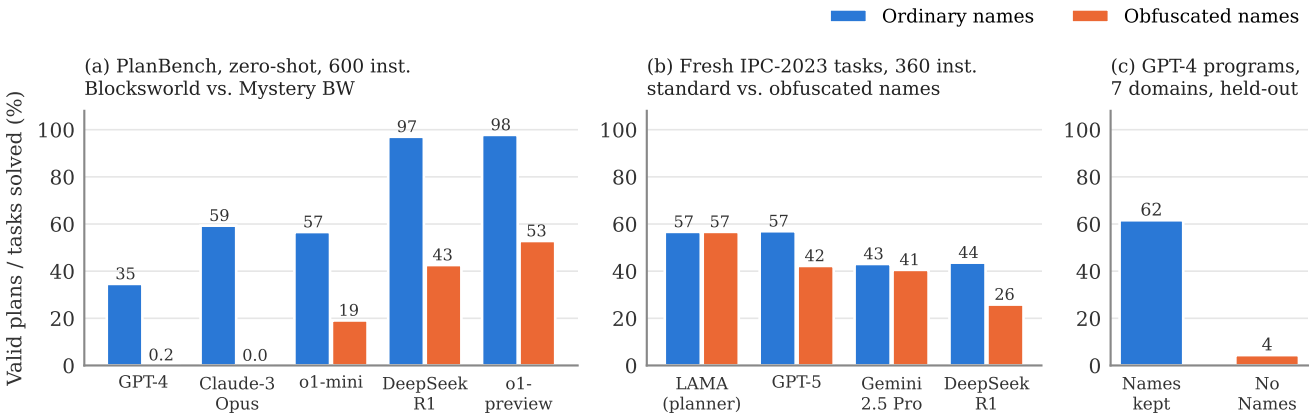


Figure 2: The same output-only test, run three ways. (a) PlanBench zero-shot plan generation [Kambhampati et al. 2025, 2024; Valmeekam et al. 2023b]. (b) Fresh tasks with random-string names; the classical planner LAMA is unaffected by renaming [Corrêa et al. 2025]. (c) Generalized programs with all PDDL names replaced by identifiers [Silver et al. 2024]. Panels use different benchmarks and must not be compared across panels.

lies in what the loop supplies: Du et al. extract structural constraints from the PDDL and use repair strategies chosen by failure position, so more of the search is done outside the model. Their baselines are also copied from earlier papers, and they flag a discrepancy between paper and repository (GPT-4 at 4.3% vs. 0%). The improvement is real, but much of it is the pipeline's, which is Claim C rather than evidence that the model plans.

Disagreement 3: is GPT-4 a generalized planner? Silver et al. [2024] answer "surprisingly powerful"; their own No Names column (62% → 4%) is as stark as Mystery Blocksworld. Both readings hold. The programs generalise to larger instances, which single-plan tests

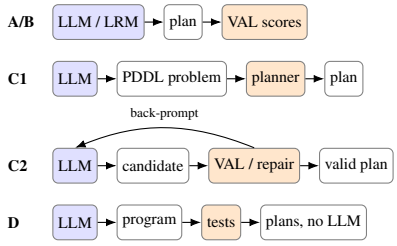


Figure 3: Where validity comes from under each claim. Blue: the language model; orange: the sound component. Only in A/B is the model alone responsible for the plan, so only there does a success rate measure the model's planning.

Table 2: What each claim rests on and how it could fail.

Claim	Rests on (experiment, baseline, benchmark)	If the claim were wrong we would see
A: retrieval	PlanBench BW vs. Mystery BW, VAL; implicit baseline is a renaming-invariant planner	Obfuscated validity close to ordinary validity for the same model and prompt
B: partial capability	Same contrast for LRMs; fresh IPC-2023 tasks with LAMA as the baseline	A collapse to near zero on fresh random renamings, or performance that follows familiarity rather than difficulty
C: system property	LLM+P vs. LLM-as-planner; back-prompting vs. one shot; pipeline vs. direct generation	No gain over a random or empty proposer when the same sound component is used
D: write the planner	Programs scored on held-out larger tasks; No Names ablation; PG3 baseline	Programs that fit the training tasks but fail on larger held-out ones; no drop without names

cannot show, but the strategy is recovered from the meaning of the names. Silver et al. also note that a correct Sokoban answer is "likely parroting pretraining data".

Disagreement 4: verify at inference time or at construction time? LLM-Modulo keeps the LLM in the inference loop as an "approximate knowledge source" whose every proposal is checked by critics [Kambhampati et al. 2024]. Katz et al. [2026] argue that such loops are costly and wasteful, because "each problem is solved separately and no computation is reused", and that bounded LLM-driven search loses soundness and completeness without gaining precision. They prefer to call the LLM once per domain to produce search components, PDDL models or policies that are then validated and reused. The two positions agree that the LLM is not the planner. They differ on where its unreliability is absorbed: per instance by a verifier, or per domain by testing generated code. The cost data support the second view: o1-preview costs over 60 times GPT-4o per instance [Kambhampati et al. 2025], whereas a synthesised generalized program ran faster than LAMA on large tasks [Silver et al. 2024].

Threats to cross-paper comparison. Four factors make numbers fragile, following the brief's checklist. *Contamination:* PlanBench instances and their solutions are public, so rising scores on ordinary Blocksworld may partly reflect exposure; fresh generation [Corrêa et al. 2025] is the only control we found. *Prompt and representation:* natural-language and PDDL prompts give different rates for the same model (GPT-4 solves 210 Blocksworld instances zero-shot in natural language but 106 in PDDL) [Valmeekam et al. 2023b]. *Kind of obfuscation:* deceptive words, random strings and generic identifiers (`predicate1`) are not interchangeable. *Budget:* LRMs, back-prompting and repair pipelines spend far more computation than single-shot generation, which a success rate alone hides; the survey by Wei et al. [2025] accordingly treats efficiency as a separate criterion alongside completeness, executability, optimality, representation and generalisation.

What outputs alone can and cannot support. Validity, renaming invariance, size generalisation and detection of unsolvable tasks are all output claims and are supported or refuted directly. Statements that a model "reasons", "searches internally" or that its intermediate tokens are "thoughts" are claims about internals. Kambhampati et al. [2025] show that formally checked traces from search-trained models are often invalid even when the final answer is right. Output-only work cannot settle these claims, so we do not draw them.

5 Open Questions and Our Position

Table 3 collects our answers to the brief's discussion questions; the questions below remain open.

Open questions. (1) When obfuscated performance stays well above chance, as with o1-preview at 52.8%, is it partial search or a broader training distribution that covers more surface forms? No study separates the two. (2) No paper reports *consistency across renamings*: whether the same instance, under several independent random mappings, gets the same validity verdict. A searcher should be consistent; a pattern-matcher need not be. (3) Size scaling is reported only coarsely ("larger instances still trip up LRMs"). Validity plotted against optimal plan length under obfuscation is missing. (4) For hybrids, the share of the result due to the LLM is rarely isolated. A random or empty-proposer control with the same verifier or repair loop, like the empty and random seed plans used for LPG by Valmeekam et al. [2023b], should be standard. (5) For generalized planning, does a program synthesised from obfuscated PDDL generalise to held-out larger tasks? Only a handful of No-Names successes exist [Silver et al. 2024].

Proposed output-only test. A system earns the label "planner" on a domain if, on freshly generated instances scored by VAL, (i) its validity under $k \geq 3$ independent random bijective renamings is statistically indistinguishable from ordinary names, with per-instance agreement across renamings, (ii) its validity degrades with optimal plan length no faster than a classical planner under the same budget, and (iii) it reports unsolvable instances as unsolvable. For a generated program, (i)–(iii) apply to held-out tasks larger than any training task. When a sound component takes part, the label belongs to the system, and the LLM's share is measured against a null proposer.

On the brief's advanced questions. Splitting a system into an LLM that writes a formal artifact and a solver that executes it is, in our reading, mainly a *decomposition* rather than a new notion of planning capability: the planning guarantee is the solver's, and the LLM's contribution is a modelling or translation capability that should be evaluated as such (fidelity of the PDDL, robustness to paraphrase), not by end-to-end plan validity. A generalized program scored on many held-out, larger instances does give stronger evidence than one valid plan, in the specific sense that it rules out instance-level copying: a fixed program cannot have memorised plans for tasks it never saw. It does not rule out retrieval of a known

Table 3: The topic brief's discussion questions, answered from the evidence reviewed.

Question	Answer and evidence
What counts as evidence that a model is planning?	A plan that a validator accepts, which stays valid under renaming and on fresh, larger instances, and a correct "unsolvable" verdict when no plan exists (Sec. 6). Plan-like form and a plausible trace do not count [Kambhampati et al. 2025; Valmeekam et al. 2023b].
Is validity on ordinary Blocksworld sufficient?	No. GPT-4 and Claude-3 Opus reach 35–59% on Blocksworld and about 0% on the isomorphic Mystery version [Kambhampati et al. 2024]. Validity there is consistent with retrieval.
What does Mystery Blocksworld test that the ordinary version does not?	Dependence on lexical cues. Structure is held fixed, so the planner baseline is unchanged and any drop is attributable to names (Fig. 1).
What should and should not be inferred from a drop under obfuscation?	It shows reliance on names and limited generalisation. It does not prove verbatim memorisation, and survival does not prove search: deceptive names may mislead more than random ones, and a broad training distribution can cover many surface forms [Corrêa et al. 2025].
What changes when the model writes a PDDL problem for a planner?	Search and validity move to the planner. The claim becomes one about translation fidelity, which is not tested by obfuscation and depends on a human domain file and an in-context example [Liu et al. 2023].
What changes with a generalized program?	The output can be scored on many held-out, larger instances, which is a stronger test of generalisation, but the No Names ablation shows the strategy still leans on names [Silver et al. 2024].
What role does a validator or planner play?	In scoring, it is the ground truth. Inside the loop, it guarantees every returned plan, so success measures the model's contribution only against a null-proposer control [Du et al. 2026; Kambhampati et al. 2024].
Which claims are output-only, which need internals?	Validity, invariance, scaling and unsolvability are output claims. "Reasons", "searches internally" and "traces are thoughts" are claims about internals and are not settled here [Kambhampati et al. 2025].

strategy, which is exactly what the No Names ablation probes. A generated program that passes held-out tests after renaming would be the strongest output-only evidence of a planning-like capability we can currently specify.

Our position. On the evidence above, no language or reasoning model studied is a planner in the classical sense. Every one fails criterion (i), while the planner baselines pass it by construction (Figure 2). "Copying plans it has seen" is, however, no longer an adequate description of frontier reasoning models, which keep most of their performance on fresh, renamed tasks [Corrêa et al. 2025]. We therefore read current models as strong, renaming-sensitive *plan generators*. Validity guarantees come from sound components, either at inference (Claim C) or at construction time (Claim D), and that is where reliable LLM planning currently lies.

6 Comparison With Our Earlier Report

Our first report [Yakpe Trolindo et al. 2026] reached a similar practical conclusion: LLMs belong inside larger frameworks rather than as replacements for classical planners. We also flagged that an obfuscation drop shows limited generalisation rather than proof of memorisation. Rechecking it against the topic brief and the sources showed the gaps in Table 4, which this review addresses.

The earlier cross-study success-rate table (Fast Downward 97.9% vs. o1 63.4% [Goebel and Zips 2025]; LLM+P 45.7%) remains useful, but it compares different systems on different benchmarks. The ordinary-versus-obfuscated contrast within one setup is the stronger evidence for the central question, because each model serves as its own control. We dropped the online-versus-offline agent results because we could not locate the source preprint.

7 Conclusion

Plan validity on familiar domains does not show planning: the same models fail isomorphic, renamed tasks that leave a classical planner untouched. The literature disagrees on how large that failure now is, not on whether it exists. Hybrids and generated planners deliver valid plans because a sound component guarantees them. The test that would settle the central question can be run on outputs alone: renaming invariance with cross-mapping consistency, size scaling and unsolvability detection on fresh instances.

Table 4: Earlier report versus this review.

Aspect	Earlier report	This review
Question, scope	Implicit; no in/out scope	Stated precisely, with in and out of scope (Sec. 1)
Organisation	Chronological and by method (early vs. modern), largely paper by paper	Four claims, each compared on the same obfuscation evidence
Reasoning models	Kambhampati et al. [2025] listed but not discussed	LRM ordinary vs. obfuscated results are central (Claim B)
Obfuscation evidence	Only LLM-Modulo's 82/70/10%	Five sources, three settings (Fig. 2), incl. No Names
Precision	82% presented as overall success	82% is on 50 previously failed instances
Assessment	Caveats on comparability	Experiment, baseline, falsifier per claim; three disagreements explained
Citations	Logistics credited to a manufacturing-strategy paper; PlanGenLLMs cited as the FLAN paper; preprints and versions unmarked; one preprint we could not locate	Author-year; preprints and versions marked; v1 of arXiv:2504.09762; changed titles of PlanBench and Corrêa et al. noted

AI use disclosure. Claude (Anthropic) was used to extract figures from the cited papers, draft and typeset this review in LaTeX, and produce the accompanying slides from our earlier report and the readings. The authors are responsible for checking every number against the cited versions and for the conclusions.

References

Hubert Yakpe Trolindo and CAP 6671 group members. 2026. Can a Language Model Plan, or Is It Copying Plans It Has Seen? CAP 6671 group report, University of Central Florida, October 2026. Our prior submission, compared against in Section 7.

Augusto B. Corrêa, André G. Pereira, and Jendrik Seipp. 2025. The 2025 Planning Performance of Frontier Large Language Models. Preprint, arXiv:2511.09378. https://arxiv.org/abs/2511.09378v1 Version read: v1 (12 Nov 2025). Later versions are retitled "Frontier Large Language Models Rival State-of-the-Art Planners" and report different models and results.

Zhendong Du, Hanliu Wang, and Kenji Hashimoto. 2026. Lexical-Prior-Free Planning: A Symbol-Agnostic Pipeline that Enables LLMs and LRMs to Plan under Obfuscated Interfaces. *Computers, Materials & Continua* 87, 1 (2026). doi:10.32604/cmc.2025.074520

Kai Goebel and Patrik Zips. 2025. Can LLM-Reasoning Models Replace Classical Planning? A Benchmark Study. Preprint, arXiv:2507.23589. https://arxiv.org/abs/2507.23589v1 Version: v1 (31 Jul 2025); figures as reported in our prior report.

Malte Helmert. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26 (2006), 191–246. doi:10.1613/jair.1705

Richard Howey, Derek Long, and Maria Fox. 2004. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)*. 294–301. doi:10.1109/ICTAI.2004.120

Subbarao Kambhampati, Kaya Stechly, and Karthik Valmeekam. 2025. (How) Do Reasoning Models Reason? *Annals of the New York Academy of Sciences* 1547, 1 (2025), 33–40. doi:10.1111/nyas.15339 Cited and read as arXiv:2504.09762v1 (14 Apr 2025); later arXiv versions changed substantially.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024) (PMLR, Vol. 235)*. https://arxiv.org/abs/2402.01817v3 arXiv:2402.01817. Version read: v3 (12 Jun 2024).

Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi. 2026. Planning in the LLM Era: Building for Reliability and Efficiency. Preprint (position paper), arXiv:2605.21902. https://arxiv.org/abs/2605.21902v1 Version read: v1 (21 May 2026).

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. Preprint, arXiv:2304.11477. https://arxiv.org/abs/2304.11477v3 Version read: v3 (27 Sep 2023).

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. *PDDL – The Planning Domain Definition Language*. Technical Report CVC TR-98-003. Yale Center for Computational Vision and Control.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B. Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. 2024. Generalized Planning in PDDL Domains with Pretrained Large Language Models. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI 2024)*. https://arxiv.org/abs/2305.11014v2 arXiv:2305.11014. Version read: v2 (18 Dec 2023).

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. Chain of Thoughtlessness? An Analysis of CoT in Planning. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*. https://arxiv.org/abs/2405.04776 arXiv:2405.04776.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, 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 (NeurIPS 2023), Track on Datasets and Benchmarks*. https://arxiv.org/abs/2206.10498v4 arXiv:2206.10498. Version read: v4 (26 Nov 2023). v1 (2022) had a different title, "Large Language Models Still Can't Plan," and a different author list.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. On the Planning Abilities of Large Language Models: A Critical Investigation. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Spotlight*. https://arxiv.org/abs/2305.15771v2 arXiv:2305.15771. Version read: v2 (6 Nov 2023).

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. 2025. PlanGen-LLMs: A Modern Survey of LLM Planning Capabilities. Preprint, arXiv:2502.11221. https://arxiv.org/abs/2502.11221v3 Version read: v3 (23 Jun 2025).