Note: A summary is provided on page 10 onwards. It is recommended to go through it before reading the full paper!

Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks

Subbarao Kambhampati [1] **Karthik Valmeekam** [1] **Lin Guan** [1] **Mudit Verma** [1] **Kaya Stechly** [1]
Siddhant Bhambri [1] **Lucas Saldyt** [1] **Anil Murthy** [1]

Abstract

We argue that auto-regressive LLMs cannot, by themselves, do planning or self-verification (which is after all a form of reasoning), and shed some light on the reasons for misunderstandings in the literature. We also argue that LLMs should be viewed as universal approximate knowledge sources that have much more meaningful roles to play in planning/reasoning tasks beyond simple front-end/back-end format translators. We present a vision of **LLM-Modulo Frameworks** that combines the strengths of LLMs with external model-based verifiers in a tighter bi-directional interaction regime. We will show how the models driving the external verifiers themselves can be acquired with the help of LLMs. We will also argue that rather than simply pipelining LLMs and symbolic components, this LLM-Modulo Framework provides a better *neuro-symbolic* approach that offers tighter integration between LLMs and symbolic components, extending the scope of model-based planning/reasoning regimes towards more flexible knowledge, problem and preference specifications.

1. Introduction

Large Language Models (LLMs), essentially n-gram models on steroids which have been pre-trained on web-scale language corpora (or, effectively, our collective consciousness), have caught the imagination of the AI research community with linguistic capabilities that no one expected text completion systems to possess. Their seeming versatility has led many researchers to wonder whether they can also do well on planning and reasoning tasks typically associated with System 2 competency. On the face of it, this doesn't seem to ring true, as both by training and operation, LLMs are best seen as a giant pseudo System 1 (Kahneman, 2011) (see Figure 1). Even from a pure engineering perspective, a system that takes constant time to produce the next token cannot possibly be doing principled reasoning on its own.[1] Not surprisingly, initial excitement based on anecdotal performance of LLMs on reasoning tasks (Bubeck et al., 2023) has been dissipated to some extent by the recent spate of studies, including our own, questioning the robustness of such behaviors–be they planning (Valmeekam et al., 2023c; Kambhampati, 2024), simple arithmetic and logic (Dziri et al., 2023), theory of mind (Ullman, 2023; Verma et al., 2024b), or general mathematical and abstract benchmarks (McCoy et al., 2023; Gendron et al., 2023). Despite this, a steady stream of claims continue to be made in the literature about the planning and reasoning capabilities of LLMs. In light of questions about their planning capabilities, the head-long rush into agentic LLMs should be particularly concerning. After all, acting without the ability to plan is surely a recipe for unpleasant consequences!

Can you Think why?

In an ironic juxtaposition to this unwarranted optimism about the planning and reasoning abilities of LLMs, there is also unwarranted pessimism about the roles LLMs can play in planning/reasoning tasks. Several efforts (e.g. (Liu et al., 2023; Pan et al., 2023; Xie et al., 2023)) advocate using LLMs only as glorified translators–converting reasoning problems embedded in textual format to symbolic representations, and pawning them off to external classical symbolic solvers (with all their attendant expressivity and search complexity challenges (Doyle & Patil, 1991)).[2]

In truth, LLMs can be a whole lot more than machine trans-

[1]School of Computing and AI, Arizona State University, Tempe, AZ, USA. Correspondence to: Subbarao Kambhampati <rao@asu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]Think of asking an LLM an yes/no question–*is this theorem logically entailed by this first-order logic knowledge-base*. This is well-known to be a semi-decidable problem. Ask yourself if the LLM will take longer in answering the question. (If you are thinking Chain-of-thought prompts or training with step-by-step data, consider that you are essentially changing the nature of the original prompt/training).

Good question indeed!

[2]In some circles, this unidirectional pipeline has been given the undeserved badge of *neuro-symbolic architecture*.

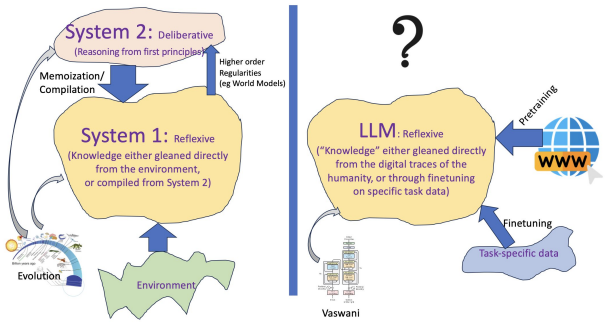

Figure 1. An informal account of viewing an LLM as a giant external non-veridical memory that acts as a pseudo System 1

lators. They are a kind of approximate knowledge source (albeit sans guarantees) trained on our collective consciousness. While it is unlikely that they will have System 2 competencies by themselves, they can nevertheless be valuable resources in solving System 2 tasks. To put it another way, the problem with Alchemy of yore was not that Chemistry is useless, but that people wanted to delude themselves that Chemistry–a pretty amazing discipline on its own merits–can be Nuclear Physics if you prompt it just so. The confusions regarding LLM abilities, or should we say, LLM alchemy, doesn't seem to be much different–oscillating between ignoring their strengths, and ascribing abilities they don't have.

The goal of this position paper is to introduce some clarity into this confusing state of affairs oscillating between over-optimism and over-pessimism. Simply put, we take the stance that LLMs are amazing giant external non-veridical memories that can serve as powerful cognitive orthotics for human or machine agents, if rightly used. The underlying n-gram nature makes them effortlessly intermix what would be considered disparate fields of study (not surprisingly, LLMs are seen to be very good at making/finding analogies!). The challenge is to leverage them without wrongly ascribing to them capabilities they don't possess. The **LLM-Modulo framework** proposed in this position paper tackles this challenge.

For the sake of concreteness, we consider planning tasks, especially as studied in the automated planning community (Ghallab et al., 2004). The central position of the paper is that *LLMs cannot plan themselves but can play a variety of constructive roles in solving planning tasks–especially as approximate knowledge sources and candidate plan generators in so-called LLM-Modulo Frameworks, where they are used in conjunction with external sound model-based verifiers.*

We support this position by first reviewing literature, including our own works, that establishes that LLMs cannot be used as planners or plan verifiers themselves (Section 2). We also discuss why there are claims about planning/verification abilities in the first place, in the process hopefully clarifying some prevalent misunderstandings.

Second, we will propose a framework that allows us to leverage LLMs effectively in planning tasks, by combining them with external critics, verifiers and humans. We call this an **LLM-Modulo Framework** (a name loosely inspired by SAT Modulo Theories (Nieuwenhuis & Oliveras, 2006)); see Figure 3. LLMs play a spectrum of roles in this architecture, from guessing candidate plans, to translating those plans into syntactic forms that are more accessible to external critics, to helping end users flesh out incomplete specifications, to helping expert users acquire domain models (that in turn drive model-based critics). All this leveraging of LLMs is done without ascribing to them any planning or verification abilities. The LLM ideas are vetted by external critics, thus ensuring that the plans generated in this architecture can have formal correctness guarantees where possible.

2. Planning-centered Limitations of LLMs

In this section, we will first review literature that calls into question claims about the planning and self-verification capabilities of LLMs. Subsequently, we will also provide some possible reasons for claims to the contrary made in the literature.

2.1. LLMs cannot generate executable plans in autonomous mode

Despite initial claims about the planning capabilities of LLMs (Bairi et al., 2023; Yao et al., 2023b; Shinn et al., 2023; Huang et al., 2022; Hao et al., 2023) several recent studies confirm that LLMs are not actually able to generate executable plans when they are used in autonomous modes (Valmeekam et al., 2023c; Liu et al., 2023; Silver et al., 2022). For example, in (Valmeekam et al., 2023c;b), we evaluate LLMs' ability to generate correct plans on a suite of planning problem instances based on the kinds of domains employed in the International Planning Competition (IPC, 1998). To eliminate the subjective aspect of analysis that forms the core part of many earlier efforts to evaluate the reasoning capabilities of LLMs, we leverage models and tools from the automated planning community to automate evaluation.

We show that results in the autonomous mode are pretty bleak. On average, only about 12% of the plans that the best LLM (GPT-4) generates are actually executable without errors and goal-reaching. We show that the choice of LLM doesn't have much bearing on this. We tested the family

Domain	Method	Instances correct					
		GPT-4o	GPT-4-Turbo	Claude-3-Opus	LLaMA-3 70B	Gemini Pro	GPT-4
Blocksworld (BW)	One-shot	170/600 (28.33%)	138/600 (23%)	289/600 (48.17%)	76/600 (12.6%)	68/600 (11.3%)	206/600 (34.3%)
	Zero-shot	213/600 (35.5%)	241/600 (40.1%)	356/600 (59.3%)	205/600 (34.16%)	3/600 (0.5%)	210/600 (34.6%)
Mystery BW (Deceptive)	One-shot	5/600 (0.83%)	5/600 (0.83%)	8/600 (1.3%)	15/600 (2.5%)	2/500 (0.4%)	26/600 (4.3%)
	Zero-shot	0/600 (0%)	1/600 (0.16%)	0/600 (0%)	0/600 (0%)	(0/500) (0%)	1/600 (0.16%)

Not surprising TBH 😅

Table 1. Results of state-of-the-art LLMs GPT-4o, GPT-4-Turbo, Claude-3-Opus, Gemini Pro and LLaMA-3 70B for Plan Generation with prompts in natural language.

of GPT LLMs including GPT-4 (OpenAI, 2023), GPT-3.5 (OpenAI, 2022), InstructGPT-3 (Ouyang et al., 2022) and GPT-3 (Brown et al., 2020). We also show that fine-tuning does not seem to have a major effect on this dismal performance. We demonstrate that the performance deteriorates further if the names of the actions and objects in the domain are obfuscated–a change that doesn't in any way affect the performance of the standard AI planners. This further suggests that LLMs are more likely doing approximate retrieval of plans than actual planning.

We continue to reconfirm these limitations over each of the more recently released LLMs, including Claude Opus, Gemini, GPT4-Turbo and GPT4-o. Table 1 shows that all the state of the art LLMs show dismal performance on PlanBench (Valmeekam et al., 2023b).

More recently, we have also investigated so-called "chain of thought" prompting (Stechly et al., 2024b), as well as ReAct-style step-by-step prompting (Verma et al., 2024a) and found that they too are largely ineffective in improving the planning performance of LLMs.

2.2. LLMs cannot verify plans and thus cannot improve by self-critiquing

There still exists considerable optimism that even if LLMs can't generate correct solutions in one go, their accuracy might improve in an iterative prompting regime, where LLMs will be able to "self-critique" their candidate solutions and refine them to the point of correctness (Yao et al., 2023b;a; Shinn et al., 2023; Weng et al., 2023; Huang et al., 2022). This belief seems to rest largely on the assumption that verification of correctness should be easier than generation for many reasoning problems–a rather classical argument from computational complexity. There are grounds to be skeptical of this assumption as the complexity of the reasoning task should be irrelevant to LLM performance if what they are doing is approximate retrieval. In general, unless LLMs are trained not just on "*correct data*," but also on "*corrections data*," there is no *a priori* reason to believe that their critiques would even be approximately relevant, let alone actually correct.

aligned with this statement

Two of our studies–one on plan verification (Valmeekam et al., 2023a) and the other on CSP verification (Stechly et al., 2023) seem to throw cold water on this optimism. In (Stechly et al., 2023), we systematically investigate the effectiveness of iterative prompting in the context of *Graph Coloring*, a canonical NP-complete reasoning problem. Our methodology involves a principled empirical study of the performance of GPT4 on two tasks: solving a large suite of random graph coloring instances and, separately, verifying the correctness of the candidate colorings–both in direct (i.e., return the first solution generated by the LLM) and iterative modes. In iterative modes, we experiment both with an LLM critiquing LLM-produced solutions and an external, guaranteed correct reasoner verifying solutions. In both cases, we analyze whether the content of criticisms actually affects bottom-line performance. A more recent paper further analyzes these results along with performance on the 24 puzzle–a task that has been used by some researchers claiming LLMs have the ability to self verify (Stechly et al., 2024a).

Our results indicate that in direct mode, LLMs are, perhaps not surprisingly, pretty bad at solving graph coloring instances. More interestingly, they are no better at verifying solutions. In iterative modes, given the inability of LLMs to verify solutions, it should come as no surprise that our experiments also show that the strategy of LLMs self-critiquing their solutions does not improve over the baseline. We report that the performance is in fact *worse* because the system can't recognize a correct coloring and thus merrily passes over fortuitously correct colorings it has generated, ending up with a wrong one! Similar results have also been reported for planning problems in (Valmeekam et al., 2023c).

Will come back to this point soon

One important corollary of the fact that LLMs cannot self-critique their plans is that they also can't self-improve by generating synthetic data, e.g. by generating plans themselves, critiquing the plans by themselves to improve them, and then using those to fine-tune themselves, as has been claimed in the literature (Huang et al., 2023b; Wang et al., 2022)[3].

2.3. Analyzing Claims to the Contrary in the Literature

Given that LLMs can neither guarantee correct generation nor correct verification of plans, as discussed in the previous sections, one obvious question is why the literature is replete with claims contrary to this (Bairi et al., 2023; Yao et al., 2023b; Shinn et al., 2023; Yao et al., 2023a; Weng et al., 2023; Huang et al., 2022).

Claims about Planning: To analyze planning claims, we need to first understand that solving planning tasks requires (a) having the necessary planning domain knowledge–the actions and their preconditions, effects; the standard hierarchical recipes (e.g. task reduction schemas in HTN planning), past cases/plans, etc., and (b) being able to assemble this planning knowledge into an executable plan that takes care of any subgoal/resource interactions. The first part can be called knowledge acquisition and the second reasoning/planning. On closer examination, many papers claiming LLMs have planning abilities wind up confusing general planning knowledge extracted from the LLMs for executable plans. When all we are looking for are abstract plans, such as "wedding plans," with no intention of actually executing them, it is easy to confuse them for complete executable plans. Indeed, our close examination of several works claiming planning capabilities for LLMs (Kambhampati et al., 2023) suggests that they either work in domains/tasks where subgoal interactions can be safely ignored (Yao et al., 2023b; Shinn et al., 2023)[4]–either because they are just working on a single subgoal, or because the world is forgiving and ergodic; or by delegating the interaction resolution (reasoning) to the humans in the loop (who, through repeated prompting, have to "correct" the plan). Sometimes, in common sense domains, or with enough fine-tuning, the "assembling" part may also be obviated by having seen a case that pretty much corresponds to the problem that needs to be solved. Not surprisingly, our work (Valmeekam et al., 2023c) shows that if the action interactions are removed by relaxing the world

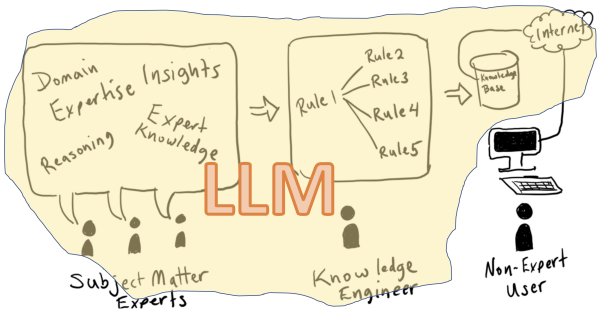

Figure 2. Viewing LLMs as an approximate knowledge source trained over civilizational knowledge

models, then the ability of LLMs to guess executable plans improves. Without these assumptions or mitigations, the plans that come out of LLMs may look reasonable to the lay user, and yet lead to execution time interactions and errors.[5]

Claims about Self-Verification: Coming to the claims about LLM's self-verification abilities, a closer look at the literature (Yao et al., 2023a; Huang et al., 2023a) shows that those claims are either (i) made in the context of tacit knowledge tasks for which there is little possibility of a verifier (e.g. essay writing)–making it hard to evaluate whether LLM's critiquing actually helped or (ii) the external verification is carried out either by simulators (Wang et al., 2023b; Yao et al., 2023b) or simple calls to the underlying operating system.

In a related vein, there is the recent Tree of Thoughts (ToT) paper (Yao et al., 2023a), which has been pitched as a way to convert LLMs into some type of systematic search with self-verification. Specifically, ToT employs a problem-specific prompt priming method. The "tree" in ToT is essentially a way to generate diverse priming prompts (that the authors set up in a *problem specific way*). In other words, despite the use of terminology of problem-solving agents (Russell & Norvig, 2010)–search tree, expansion etc., there is really no deeper connection to search-based agents.

The guarantees–if any–are coming in terms of soundness of the external verifier. The one clear reasoning problem used in the ToT paper is the 24 puzzle–for which the external verifier can be easily implemented in terms of arithmetic operations (thankfully not done by the numerically challenged LLM!). Here, our experiments show that the LLM's own criticisms are often quite off the mark.[6] Because the

[3]Contrary to their claim of "self-improvement", works like (Wang et al., 2022) actually heavily depend on external knowledge (crafted seed examples) and critics (filtering step).

[4]Although domains like AlfWorld (Shridhar et al., 2021) do have sub-goal interactions for successful task completion, (Yao et al., 2023b) and (Shinn et al., 2023) largely ignore these interactions by either focusing on single subgoals or relying on the ergodic nature of the domain when prompting LLMs for generating plans (Verma et al., 2024a).

[5]These issues are illustrated in part by a recent news story (Kugel & Hiltner, 2023) about the proliferation of travel planning books, mostly auto-extracted from LLMs, and the ensuing disappointment of the unsuspecting end users who buy them mistaking them for usable plans!

[6]Note that we can do this check easily because of the formal specification of correctness. For the "improving writing task" also

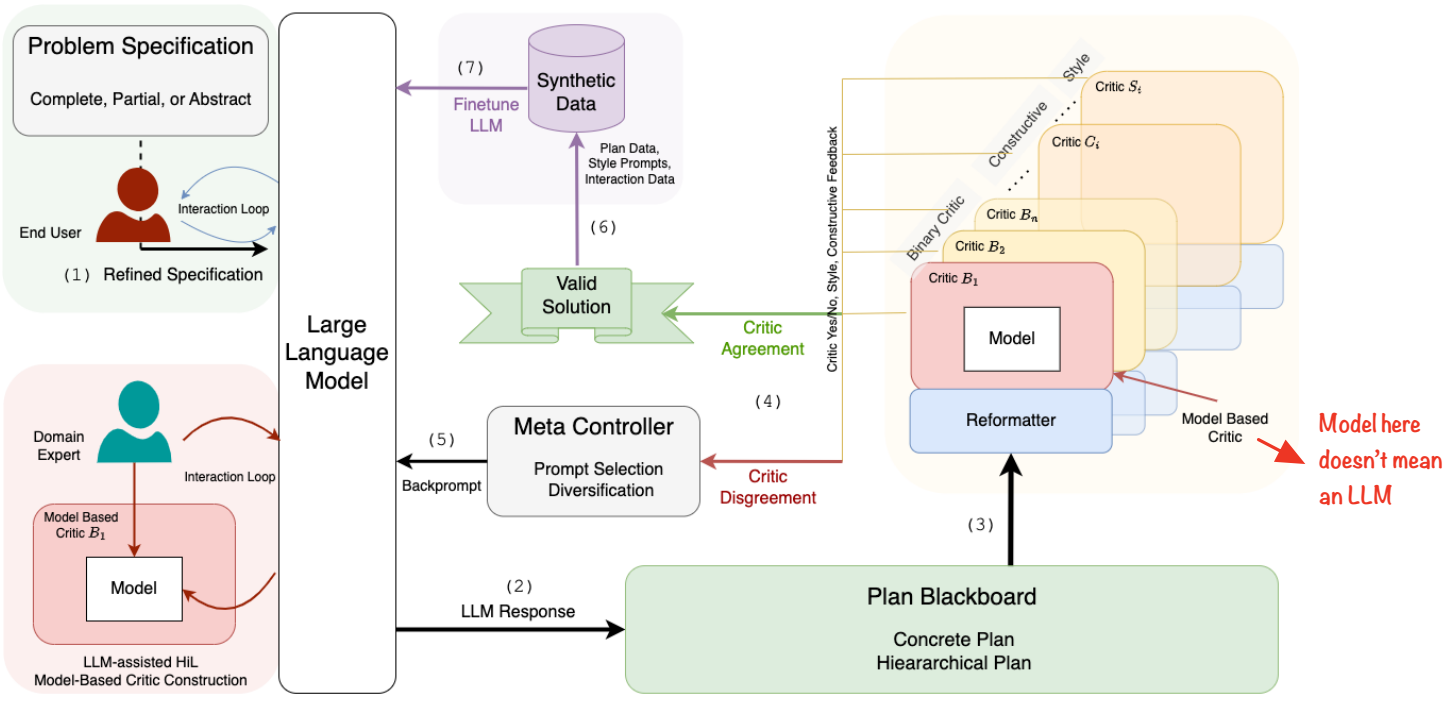

Figure 3. The proposed LLM-Modulo framework where LLMs act as idea generators and various external critics that specialize in different aspects, critique the candidate plan.

24 puzzle's solutions can be verified by simple arithmetic operations, it is trivial to implement an external verifier for these problems. In general though, the verifier may be more complex and can involve substantial work (you can substitute a simulator for the verifier–but someone has to write that simulator too!).

LLMs as Approximate Knowledge Sources: The fact that LLMs are often good at extracting planning knowledge can indeed be gainfully leveraged. As shown in recent works (Guan et al., 2023), LLMs can be a rich source of approximate models of world/domain dynamics and user preferences, as long as the humans (and any specialized critics) in the loop verify and refine those models, and give them over to model-based solvers. This way of using LLMs has the advantage that the humans need only be present when the dynamics/preference model is being teased out and refined, and the actual planning after that can be left to sounder planning frameworks with correctness guarantees, such as the LLM-Modulo framework we propose.

Such an overall approach has striking similarities to knowledge-based AI systems of yore, with LLMs effectively replacing the "knowledge engineer" (see Figure 2). Given the rather quixotic and dogmatic shift of AI away from approaches that accept domain knowledge from human experts that can be termed "Polanyi's Revenge" (c.f. (Kambhampati, 2021)), this new trend of using LLMs as knowledge sources can be viewed as a form of avenging Polanyi's revenge! Indeed, LLMs make it easy to get problem-specific knowledge as long as we are willing to relax the correctness requirements of that knowledge. In contrast to the old knowledge engineering approaches, LLMs offer this without making it look like we are inconveniencing any specific human (we are, instead, just leveraging everything humans told each other on the Web!). So the million dollar question for reasoning tasks is: "*how would you do robust planning if you have some doddering know-it-all ready to give you any kind of knowledge?*" The LLM-Modulo Framework is a principled method for tackling this challenge.

3. LLM-Modulo Framework for Robust Planning

While Section 2 questions the claims that LLMs are capable of planning/reasoning by themselves, it is certainly not meant to imply that LLMs don't have any constructive roles to play in solving planning/reasoning tasks. On the contrary, as discussed in the Introduction, their uncanny ability to generate ideas/potential candidate solutions–albeit with no guarantees about those guesses–can be valuable in

used in ToT, there are no formal quality metrics and so it is hard to say anything concrete about the critiques of the LLM.

the generate-test-critique setups in conjunction with either model-based verifiers or expert humans in the loop. Accordingly, we propose a general "LLM-Modulo" framework[7]. While we believe that versions of such an architecture can be of use in a wide variety of planning or reasoning tasks, for the sake of concreteness, we will focus on planning tasks, especially of the type studied in the automated planning community (Ghallab et al., 2004).

Figure 3 gives a schematic of the **LLM-Modulo Framework**, as we envision it. As can be seen readily, the underlying architecture is a Generate-Test-Critique loop, with the LLM generating candidate plans and a bank of critics critiquing the candidate. The loop starts with the LLM getting the problem specification and generating its first plan candidate.[8] Note that the plans an LLM helps generate in this architecture have *soundness guarantees* because of the external sound critics. This means that plans coming out of such an compound system will constitute a better corpus of synthetic data for any fine tuning phase carried out to improve/customize the LLM's generation capability. The *completeness* of the system depends on the LLM's ability to generate all potentially relevant candidates.

Design Choices: Before going into the details about the framework and its various modules, it is worth noting some design decisions underlying the proposed architecture. We start by noting that the LLM-Modulo architecture is a "Generate-Test" one that involves LLMs interacting with the external **critics/verifiers** rather than a LLMs being just front-ends to external solvers. This is a deliberate decision–as this allows the LLM to guess/generate candidates to satisfy the critics, as against dealing with the expressiveness and search complexity issues of the solvers. The critics/verifiers also are also more naturally *composable* than solvers/planners. As we shall see, we do allow for *constructive critics* which can be based on solvers, and provide suggestions on specific ways of extending/modifying the candidate plans.

Secondly, the framework explicitly recognizes that the LLMs can generate approximate ideas not just about plan candidates, but domain models, problem reduction strategies, and refinements to the problem specification. The framework also recognizes that LLMs are good at format/syntax changes. Accordingly, the framework leverages all these abilities of LLMs, letting them play multiple roles in planning. Finally, the architecture carefully circumscribes the human's role–domain experts interact with the LLM to tease out the models used by (some of) the critics, while end users take part in refining any incomplete problem specification in concert with the LLM. A notable, and deliberate, absence is human's involvement in the inner loop of planning–e.g. with iterative prompting. In addition to posing an infeasible burden on the human's time for complex planning problems, such iterative prompting strategies are notorious for their Clever Hans effect (cle).

3.1. Critics/Verifers

In the LLM-Modulo framework, critics can evaluate LLM-generated candidates for a planning/reasoning problem over both hard and soft (style) constraints. Hard constraints refer to correctness verification which can include causal correctness, timeline correctness, resource constraint correctness as well as unit tests. For PDDL planning problems, the hard critic can be based on VAL (Howey et al., 2004), that works off of a model (which itself can be acquired with the help of the LLM (Guan et al., 2023). It is worth noting that the critics don't always have to be declarative model-based ones, and can be simulators. Just as LLMs can help humans in coming up with models, they can also help in writing procedural simulators, as seems to be done in systems like Voyager (Wang et al., 2023a).

On the other hand, soft constraints can include more abstract notions of good form such as style, explicability, preference conformance, etc. As discussed in Section 2.3, while LLMs cannot take on the role of hard critics with soundness guarantees,[9] they can help simulate some aspects of the role of soft (style) critics. So our framework does allow for style critics be possibly based on LLMs. For example, in (Verma et al., 2024b) we discuss how LLMs can act as a human proxy to evaluate plans in terms of how they would be perceived by humans in the loop. Additionally, in (Guan et al., 2024), we show how Vision-Language Models (VLMs) can be leveraged to critique the style of robot behaviors in terms of their adherence to the soft common-sense preferences of the humans in the loop. We reiterate that the soundness of the LLM-modulo framework is inherited from the soundness of the correctness (hard) critics.

The bank of critics–hard (model-based) as well as soft (possibly LLM-based) evaluate the current plan candidate to evaluate its fitness/acceptability. If at least all the hard critics sign off on the current candidate, then that is considered a valid solution to be returned to the end-user or the executor. When a critic finds the current plan candidate to be unsatisfactory, it can provide varying levels of feedback, ranging from "*No, try again*" to "*No, try again, here is one thing wrong with the current plan*" to "*No, try again, here are all the things wrong with the current plan.*" More importantly, the critics can be **constructive**, and offer alternatives

[7]The name LLM-Modulo is inspired by the SAT-Modulo theories (Nieuwenhuis & Oliveras, 2006).

[8]Although we focus on planning from scratch, it is easy to accommodate replanning scenarios, where the loop starts with an externally supplied candidate plan.

[9]If we don't insist on soundness guarantees, then it is, in principle, possible to train LLMs discriminatively to learn to verify plans; see (Arora & Kambhampati, 2023).

Yes!👍

plan/subplan suggestions. One way of obtaining such constructive critics is to base them on partial planners–operating either on the models themselves or their relaxations (Bryce & Kambhampati, 2007).These critiques are all pooled at the Meta (Backprompt) Controller (see Section 3.2)

LLMs as Reformulators: One interesting challenge is that many of the symbolic model-based verifiers tend to be operating on specialized formal representations. Given a central candidate plan (e.g. a mission plan), these critics need translations of that candidate into their representations. This is the role of the reformulator module attached to individual critics. These reformulator modules can be supported to a large extent by LLMs, given that one thing LLMs are very good at is format change across different syntactic representations (Olmo et al., 2021). Indeed, as discussed in Section 1, some approaches to combine LLMs with external symbolic solvers just use LLMs as reformulators for these solvers (Liu et al., 2023; Pan et al., 2023). It is worth noting that the syntax conversion itself can be helped with a nested LLM-Modulo framework–where the syntactic correctness of the conversion is checked by syntax critics. We will have occasion to illustrate this in the context of our preliminary work on LLM-Modulo frameworks for travel planning discussed in Section 4. Our discussion of LLM-Modulo framework should make it clear that syntax reformulation alone is a severely limited role for LLMs!

3.2. Backprompt (Meta) Controller

The critiques from the various critics are pooled together by the Meta (Backprompt) Controller, which passes a processed version of them to the LLM as the next iterative prompt to elicit the next guess. This is especially required in the presence of a mix of soft and hard critics, where the Meta Controller can assume the responsibility of compiling the critiques into a consistent feedback to send to the LLM.

The processing in the controller can range from (i) simple round-robin selection of prompts to (ii) generating a summarized prompt (with LLM help) to (iii) employing a *prompt diversification strategy* to elicit the next candidate from a different part of the implicit search space. This last strategy helps increase the completeness of the LLM candidate generation, and may involve domain/task-specific knowledge (see the discussion of Tree of Thoughts in Section 2.3).

3.3. Specification Refinement & Critic/Model Acquisition

As mentioned earlier, we avoid having humans involved in iteratively prompting LLMs–as this can be an infeasibly time-consuming activity for them. Instead, we let automated verifiers, either model-based or LLM-supported, to manage the plan critiquing process. The framework does depend on humans for "once per domain" and "once per problem" interactions. In the former category, human domain experts can play a role in acquiring the domain model with the help of the LLM. Examples of such interaction include teasing out PDDL planning models from the LLMs with the help of human expert curation (top left in Figure 3). An example of this is our work in (Guan et al., 2023). The idea here is that the traditional domain model acquisition task (e.g. (Simpson et al., 2001)) is significantly made easier by having the LLMs help with ideas regarding various pieces of the domain model (e.g., actions, their preconditions and effects) and letting humans sign off/critique the resulting model. Once the model is acquired this way, it can be used by correctness verifiers such as VAL (Howey et al., 2004; Guan et al., 2023). Often the planning problems in real world situations are specified incompletely, leaving it to the human commonsense to refine the specification. This brings up a second role for humans–this time end users (bottom left in Figure 3–in collaboratively refining the specification with the help of LLMs (similar to the way done in (Xie et al., 2023; Liu et al., 2023)).

3.4. Summary of LLM Roles in LLM-Modulo

It is worth summarizing the multiple roles the LLM plays in the LLM-Modulo architecture. The most prominent, of course, is its role in "guessing" the candidate plans (step 2 in Figure 3) in response to problem specification and iterative back prompting from the bank of critics (Step 5). Second, the LLM plays a role in converting the guessed plan candidate into specialized representations used by the various critics (e.g., the time-line view, the causal link view etc.). This role leverages the fact that LLMs are very good at format conversion (c.f. (Olmo et al., 2021)). Third, the LLM plays a role in helping the end user flesh out the incomplete problem specification to begin with (Step 1 in Figure 3). Finally, the LLM plays a role in helping the domain expert tease out and refine the domain models used by the various model-based critics (Guan et al., 2023; Kwon et al., 2022), or help "implement" procedural critics (such as those checking syntactic constraints). As a broad approximate source of knowledge, the LLM can also help enumerate the list of potential critics needed to validate the candidate plans (once again with a human in the loop).

3.5. Can LLM-Modulo Frameworks Pay Their Way?

Let's address the elephant in the room: *Given that formal model-based planning systems already exist (Ghallab et al., 2004), is LLM-Modulo framework for planning more than a gratuitous attempt to shoe-horn (the currently popular) LLMs to solve planning problems?* Indeed, when the underlying problem is actually solvable by such combinatorial solvers, it can be orders of magnitue more resource efficient

How much are the LLMs used in this hybrid framework proposed for planing?

We all have already used LLMs as reformatted in many use cases.
Think about a min where we have applied this.
Can you think why a controlled is required?

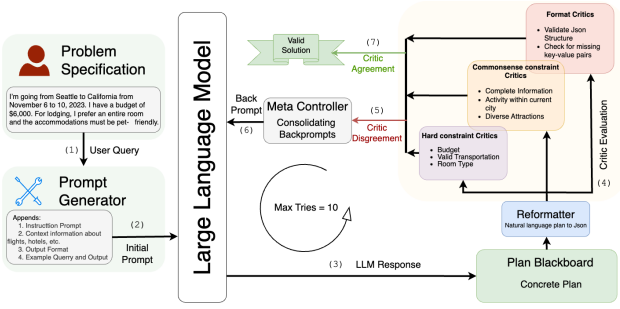

Figure 4. LLM Modulo Framework adapted for Travel Planning

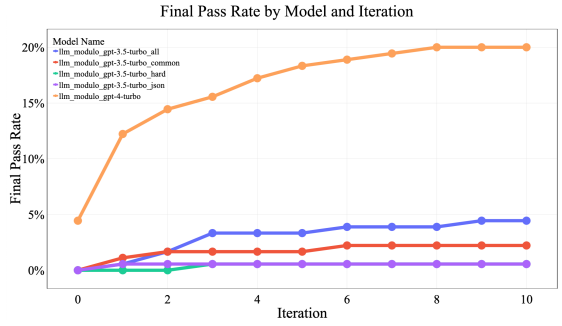

Figure 5. Final Pass rates of models across LLM Modulo Iterations

to use them.[10] Compared to a planner that is guaranteed to be correct in a narrow set of domains, LLMs may likely be good at generating plausible (but not guaranteed to be correct) plan heuristics/suggestions in many more scenarios. Thus, unlike the traditional planning architectures studied in AI (Ghallab et al., 2004), which put *a priori* constraints on the expressiveness of the problems that can be posed to the planner (to wit, the different expressiveness levels of the PDDL specification (McDermott et al., 1998)), the LLM-Modulo architecture puts no such restrictions. In this sense, it is more representative of real-world planning problems such as those in NASA mission planning, where the different critics–human and automated–are at best able to give "no objection" certificates for the candidate plans under consideration, clearing it from their perspective. (Indeed, both deep space network planning and mars rover task planning are done via a *collective human blackboard*. (Johnston et al., 2014; Bresina et al., 2004).) Note that this is starkly different from just sending an unvetted plan out to execution (as would be the case if we have LLMs operate in autonomous mode to guess plans). Generalizing planning and reasoning frameworks this way is consistent with the Doyle & Patil's call to the Knowledge Representation community of yore (Doyle & Patil, 1991), as well as our own call for model-lite planning (Kambhampati, 2007).

[10] Not surprisingly, automated programming, the one community that certainly doesn't have the luxury of a ready-made "solver," have stuck to LLM-Modulo style approaches.

4. Two Case Studies of LLM-Modulo

We have applied the LLM-Modulo framework to classical planning domains (as reported in (Valmeekam et al., 2023c)) and to a recent travel planning benchmark (as reported in (Gundawar et al., 2024)). In the former case, the results (presented in Section 5.2 and Table 4 of (Valmeekam et al., 2023c)) show that with back prompting from VAL (Howey et al., 2004) acting as the external verifier and critic, LLM performance in Blocks World improves to 82% within 15 back prompting rounds, while in Logistics, it improves to 70%. LLM-Modulo doesn't help as much in an obfuscated version of blocks world called Mystery BW, reaching about 10% accuracy. This should be expected because the LLMs have difficulty generating plausible candidate plans for this domain (note that even here, if a plan is returned, it must have passed muster with VAL, and is thus guaranteed correct by its model).

The case where Llm doesn't help at all

For the travel planning case study, we used a benchmark proposed in (Xie et al., 2024), which involves a rich mix of travel constraints presented in flexible natural language form. Our preliminary results on adapting LLM-Modulo framework to this benchmark are reported in (Gundawar et al., 2024). The benchmark's authors test LLMs across a variety of prompt engineering techniques including Chain of Thought and ReAct, reporting that–on GPT-3.5-Turbo–the current best strategies only manage a startlingly low 0.7% performance rate! We adapted the LLM-Modulo framework to this benchmark by operationalizing their hard constraints (such as the budget constraint set by the user) or commonsense constraints (such as suggesting diverse attractions to visit) as critics as shown in Figure 4. Our preliminary results show (see Figure 5; additional results in (Gundawar et al., 2024)) that LLM-Modulo based agentification with automated critics in the loop significantly improves the performance (6x of baselines) even with a limit of 10 back prompting cycles, and weaker models such as GPT-3.5-turbo. Furthermore, we also find that LLMs can successfully implement functions corresponding to hard critics and several common-sense critics. Finally, LLMs reliably play the role of reformatter as well, converting free form travel plans into structured plans parseable by the critics for back-prompts or plan evaluation. One interesting observation about this domain is that we were able to use the LLM itself to enumerate the type of critics needed to validate the plan (with light human supervision).

So, critiquing works but not self-critiquing

5. Related Work

While the LLM-Modulo framework is being proposed in general form here for the first time, there are certainly works in leveraging LLMs in planning and reasoning tasks that are in line with the spirit of the LLM-Modulo framework. Work on FunSearch (Romera-Paredes et al., 2023) depends on a

generate-test loop between a specially fine-tuned LLM that guesses solutions, and an external symbolic evaluator that critiques them. The authors note how the external verifier is critical for avoiding falling prey to hallucinations (i.e., approximate solution candidates that have flaws). AlphaGeometry (Trinh et al., 2024) too depends on the Generate-Test-Critique interaction between a fine-tuned LLM and a symbolic evaluator. Both these systems fine-tune pre-trained LLMs with task specific synthetic data–the correctness of which is vetted with external simulators.

While we focused on PDDL planning tasks for the sake of concreteness, we believe that the essence of LLM-Modulo framework is equally applicable to other scenarios involving planning and reasoning–such as *Reinforcement Learning with Simulators*. Such RL systems rely on rewards as feedback to train a policy. Simulators takes on the roles of plan evaluation and critiques performed by the respective critics in the LLM-Modulo framework (e.g. (Rajvanshi et al., 2023)). The fact that simulators play the role of verifiers is often not explicitly recognized in cases where LLMs are used as an actor to generate an admissible plan by interacting with a simulator, for example in the case of AlfWorld (Yao et al., 2023b; Shinn et al., 2023) and Minecraft (Wang et al., 2023b). Similar to extracting a domain model such as in the case of (Guan et al., 2023), LLMs can also be used for designing a reward model or shaping the reward (Bhambri et al., 2024; Kwon et al., 2022; Hao et al., 2023; Ma et al., 2023).

Interestingly, the fact that LLM's can help come up with approximate quasi-symbolic transition models, reward models and models of high level actions has made a bigger splash in RL. This is because for far too long, researchers there have tried to spurn any high level models (lest that would involve depending on humans; (Kambhampati, 2021)) and focused on learning to act from sensory information, under the name of "deep reinforcement learning." Given the horrendous sample complexity of the DRL methods even in reaching a single subgoal, and the well known fact that even approximate symbolic models can help drastically improve the performance (c.f. (Guan et al., 2022)), coupled with the fact that LLM's are only too glad to dream up approximate models and goal recipes, there has been a performance revolution of sorts there (Yao et al., 2023b; Liang et al., 2023; Wang et al., 2023b). If we look beyond the improvements in these lower level goal seeking behaviors–especially in the presence of ergodic simulators, the RL approaches dependent on LLMs will encounter the same issues regarding subgoal interactions that our discussion of PDDL planning problems brought into focus. The LLM-Modulo inspired frameworks will thus, we believe, be equally relevant there. Indeed, SayCan (Ahn et al., 2022) the earliest use of LLMs in generating policies in an RL-with-Simulator scenario, explicitly filters the action choices suggested by the LLM with the help of simulator.

Although we focused on text based LLMs (such as GPT4), recently there have also been impressive development in multi-modal LLMs (e.g. GPT4V). While multi-modality is a great addition that increases the coverage of their System 1 imagination (Figure 1), it is not clear that this gives them System 2 competence.[11] As we discussed earlier, we can leverage VLMs for style criticism of the robot behavior (Guan et al., 2024).

Finally, our position (with published supporting evidence) that LLMs are incapable of supporting planning in autonomous modes must seem quite at odds with the current head-long rush into agentic LLMs. We believe that the latter is largely a result of confusing "acting" with "planning." Given their ability to translate across formalisms, it is of course possible for LLMs to invoke external services–something frameworks like AutoGPT and LangChain support. But the mere ability to invoke an action doesn't, in any way, guarantee that the course of actions thus invoked will achieve a desired state of affairs. The only way to guarantee the latter is to to support robust planning capabilities–something our LLM-Modulo frameworks strive to do.

6. Conclusion

This position paper is a modest attempt to combat both over-optimism and over-pessimism about the role of LLMs in planning and reasoning tasks. Our position is that *LLMs cannot plan themselves but can play a variety of constructive roles in solving planning tasks–especially as approximate knowledge sources and candidate plan generators in the so-called LLM-Modulo Frameworks in conjunction with external sound model-based verifiers.* In support of this position, we summarized the literature questioning the claims about the planning and self-verification capabilities of LLMs by themselves. We also discussed how conflating approximate knowledge acquisition and generating executable plans of action is behind many of the claims about planning and verification abilities of LLMs. We then shared LLM-Modulo framework, our vision for a productive way to integrate the impressive idea generation/approximate knowledge provision capabilities of LLMs with external verifiers with correctness guarantees, for robust and expressive planning. We discussed how planning in LLM-Modulo framework avoids inheriting the expressiveness and search-complexity limitations of traditional symbolic planners, while retaining their soundness guarantees. We illustrated and discussed the many roles LLMs can play in the LLM-Modulo framework. Finally, we also discussed two case studies of adapting the LLM-Modulo frameworks.

[11] If you know how to complete sentences, and now learned to complete dance moves, does your ability to reason/plan magically improve?

Summary

A bit of a background

- With the rise of the capabilities of LLMs to perform well on many tasks, researchers have been trying to figure out whether LLMs can do well on planning and reasoning-based tasks.
- Several papers have shown that LLMs can reason, but many recent studies have found those claims very limited.
- So, there is both optimism and pessimism about the planning capabilities of the current generation of LLMs
- This paper focuses solely on planning tasks with a position that LLMs cannot plan themselves but can play a variety of constructive roles in solving planning tasks–especially as approximate knowledge sources and candidate plan generators in so-called LLM-Modulo Frameworks.

LLMs cannot generate executable plans in autonomous mode

- The authors evaluate the ability of LLMs to generate correct plans on a suite of planning problem instances based on the kinds of domains employed in the International Planning Competition. They leverage models and tools from the automated planning community to automate evaluation.
- They found out that only about 12% of the plans that the best LLM generates are executable without errors and goal-reaching. The choice of LLM does not matter much in this regard.
- Fine-tuning does not help either, and the performance deteriorates further if we obfuscate the names of the actions and objects in the domain, suggesting that LLMs are more likely to do approximate retrieval of plans than actual planning.
- Techniques like chain of thought (CoT) are ineffective.

LLMs cannot verify plans and thus cannot improve by self-critiquing

- Many papers suggest that it is easier to verify than to generate. Though true to an extent, the problem with this statement is that people start to conclude that LLMs can reason.
- While using another larger and better model as a critique for certain tasks is fine IMO, a side effect of the above statement is that people tend to believe that self-critiquing works.
- The authors argue that unless an LLM is not trained on corrections data, there is no reason to believe that critiquing is relevant and correct.
- The authors use GPT4 for two tasks: i.) solving a large suite of random graph coloring instances, ii.) verifying the correctness of the candidate colorings both in direct and iterative modes. In direct mode, we return the first solution generated by GPT4, while in iterative mode, we are allowed to use LLM critiquing, or guaranteed correct reasoner verifying solutions.
- Results? Not only do the authors find that LLMs are pretty bad at solving graph coloring problems in direct mode, but they are also bad at verifying solutions and that self-critiquing their solutions does not improve over the baseline.

What about the claims made in the past about planning?

- Planning, in general, has two prerequisites:
  - Necessary planning domain knowledge: the actions and their preconditions, effects, hierarchical task reduction schemes, etc.
  - Ability to use this knowledge to come up with an executable plan (ideally an optimal plan).
- The authors argue that works claiming planning capabilities for LLMs suggest that they either work in domains/tasks where sub-goal interactions can be safely ignored, the mistakes do not come with a huge cost, or the refinement of the plan can be delegated with a human in the loop.

What about the claims made in the past about self-verification?

- Of all the attempts, the Tree of Thought (ToT) argued that an LLM can solve a task successfully if we can break it into states over a search space.
- Given a specific problem, it generates diverse priming prompts, and an LLM is used to evaluate a particular state. The problem is that it again relies on self-consistency.
- The authors notice that other works have mostly used LLMs as verifiers in places where it is hard to argue about their critiquing capabilities, like in essay writing.

The million-dollar question: Can LLMs plan, can't they?

To answer this question, we need to keep in mind a few things:

- Trained on web-scale data, LLMs learn a lot. They are useful for many tasks, if and only if you are not looking up to them as the final solution generator. For example, instead of asking an LLM to generate a fool-proof plan, a good idea is to ask it to generate a few diverse plausible plans.
- With the help of the right tools, LLMs can plan. Whether you apply a human-in-the-loop approach or use external verifiers/tools is up to you. Their combination is a good enough start to build reliable systems for certain use cases.

Proposed LLM-Modulo Framework for Robust Planning

Given the above premise, the authors then propose a framework to use LLMs in combination with other things to leverage it for planning use cases.

- Inspired by the SAT-Modulo theories
- Generate-test critique loop: LLMs generate the plan, and a pool of critiques validates the plan. LLMs are proposers, not the final decision-makers
- Leverages the fact that LLMs are extremely good at format/syntax change
- Critics/Verifiers can evaluate LLM-generated candidates for a planning/reasoning problem over both hard and soft (style) constraints. Hard constraints refer to correctness verification and unit tests, while soft constraints include things like style, preference, etc.

- External critics are model-based (here the model doesn't mean LLM but refers to the external tools that generate guaranteed solutions to a particular problem).
- Once the LLM generate a proposal (candidate), we can leverage the LLM to format this plan to the exact syntactic representation needed by these critics (e.g. VAL for PDDL)
- If at least all the hard critics sign off on the current candidate, then that is considered a valid solution to be returned to the end user or the executor. OTOH, if something is off, they can provide constructive feedback to improve the proposal.
- Once the critics complete their task, the critiques are pooled together by a meta controller that passes a processed version of them to the LLM as the next iterative prompt to elicit the next guess.
- The framework does depend on humans: once per domain (human experts acquire the domain model with the help of an LLM) and once per problem (collaboratively refining the specification with the help of LLMs).
- The authors showcased the power of the proposed framework via two case studies: Classical planning domains and travel planning benchmarks.
- In the former case, back prompting from VAL (Howey et al., 2004) acting as the external verifier and critic, LLM performance in Blocks World improves to 82% within 15 back prompting rounds, while in Logistics, it improves to 70%. In the second case, LLM-Modulo based agentification with automated critics in the loop significantly improved the performance (6x of baselines) even with a limit of 10 back prompting cycles and weaker models such as GPT-3.5- turbo.

Conclusion

This position paper is an attempt to combat both overoptimism and over-pessimism about the role of LLMs in planning and reasoning tasks. The authors not only showcased that LLMs cannot plan themselves but also that they can play a variety of constructive roles in solving planning tasks–especially as approximate knowledge sources and candidate plan generators in the so called LLM-Modulo Frameworks in conjunction with external sound model-based verifiers

Impact Statement

This position paper takes a stance on a robust and well-founded way of leveraging Large Language Models in planning and reasoning tasks. It (i) points out the inabilities of current pre-trained LLMs to tackle planning problems, (ii) suggests some reasons as to why there are wide-spread misunderstandings about LLM planning abilities and (iii) proposes LLM-Modulo frameworks as a way to leverage LLMs for robust planning. The main consequences of realizing this position/vision is expected to be (i) sounding caution about misapplication of LLMs in autonomous modes for planning (ii) providing a way to leverage LLMs to do robust planning. Given the current interest in agentic LLMs, these insights can have significant positive impact in mission critical situations. We do not see any obvious negative societal consequences of leveraging LLMs this way (unless of course the plans are aimed at achieving malicious goals).

Acknowledgments

The ideas discussed in this paper have evolved over a series of talks, tutorials and twitter threads. The discussions, feedback and encouragement from colleagues, including Sarath Sreedharan, Tom Dietterich, Yann LeCun, Daniel Borrajo, and Dan Weld is gratefully acknowledged. The adaptation of LLM-Modulo Framework for Travel Planning, discussed in Section 4 was lead by Atharva Gundawar. Kambhampati acknowledges generous support from ONR via grants N00014-18-1-2442, N14-18-1-2840 and N00014-23-1-2409, as well as gifts from J.P. Morgan, Qualcomm and Amazon.

References

Clever Hans. https://en.wikipedia.org/wiki/Clever_Hans.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Arora, D. and Kambhampati, S. Learning and leveraging verifiers to improve planning capabilities of pre-trained language models. *ICML Workshop on Knowledge and Logical Reasoning in the Era of Data-driven Learning (arXiv preprint arXiv:2305.17077)*, 2023.

Bairi, R., Sonwane, A., Kanade, A., Iyer, A., Parthasarathy, S., Rajamani, S., Ashok, B., Shet, S., et al. Codeplan: Repository-level coding using llms and planning. *arXiv preprint arXiv:2309.12499*, 2023.

Bhambri, S., Bhattacharjee, A., Liu, H., and Kambhampati, S. Efficient reinforcement learning via large language model-based search, 2024.

Bresina, J. L., Jónsson, A. K., Morris, P. H., and Rajan, K. Activity planning for the mars exploration rovers. In *ICAPS-2005 Conference*, 2004.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Bryce, D. and Kambhampati, S. A tutorial on planning graph based reachability heuristics. *AI Mag.*, 28(1):47–83, 2007.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Doyle, J. and Patil, R. S. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial intelligence*, 48(3):261–297, 1991.

Dziri, N., Lu, X., Sclar, M., Li, X. L., Jiang, L., Lin, B. Y., Welleck, S., West, P., Bhagavatula, C., Bras, R. L., Hwang, J. D., Sanyal, S., Ren, X., Ettinger, A., Harchaoui, Z., and Choi, Y. Faith and fate: Limits of transformers on compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Fkckkr3ya8.

Gendron, G., Bao, Q., Witbrock, M., and Dobbie, G. Large language models are not abstract reasoners. *arXiv preprint arXiv:2305.19555*, 2023.

Ghallab, M., Nau, D., and Traverso, P. *Automated Planning: theory and practice*. Elsevier, 2004.

Guan, L., Sreedharan, S., and Kambhampati, S. Leveraging approximate symbolic models for reinforcement learning via skill diversity. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7949–7967. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/guan22c.html.

Guan, L., Valmeekam, K., Sreedharan, S., and Kambhampati, S. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=zDbsSscmuj.

Guan, L., Zhou, Y., Liu, D., Zha, Y., Amor, H. B., and Kambhampati, S. "task success" is not enough: Investigating the use of video-language models as behavior critics for catching undesirable agent behaviors, 2024.

Gundawar, A., Verma, M., Guan, L., Valmeekam, K., Bhambri, S., and Kambhampati, S. Robust planning with llm-modulo framework: Case study in travel planning. *arXiv preprint arxiv:2405.20625*, 2024.

Hao, S., Gu, Y., Ma, H., Hong, J. J., Wang, Z., Wang, D. Z., and Hu, Z. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

Howey, R., Long, D., and Fox, M. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pp. 294–301. IEEE, 2004.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023a.

Huang, J., Gu, S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1051–1068, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.67. URL https://aclanthology.org/2023.emnlp-main.67.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

IPC. International planning competition, 1998. URL https://www.icaps-conference.org/competitions/.

Johnston, M. D., Tran, D., Arroyo, B., Sorensen, S., Tay, P., Carruth, B., Coffman, A., and Wallace, M. Automated scheduling for nasa's deep space network. *AI Magazine*, 35(4):7–25, 2014.

Kahneman, D. *Thinking, fast and slow*. macmillan, 2011.

Kambhampati, S. Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, pp. 1601. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

Kambhampati, S. Polanyi's revenge and AI's new romance with tacit knowledge. *Communications of the ACM*, 64 (2):31–32, 2021.

Kambhampati, S. Can LLMs reason and plan? *Annals of the New York Academy of Sciences*, 2024.

Kambhampati, S., Valmeekam, K., Marquez, M., and Guan, L. On the role of large language models in planning, July 2023. URL https://yochan-lab.github.io/tutorial/ICAPS-2023/. Tutorial presented at the International Conference on Automated Planning and Scheduling (ICAPS), Prague.

Kugel, S. and Hiltner, S. A new frontier for travel scammers: A.I.-Generated Guidebooks. *New York Times*, August 2023. URL https://www.nytimes.com/2023/08/05/travel/amazon-guidebooks-artificial-intelligence.html.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. Reward design with language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control, 2023.

Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., and Stone, P. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., and Anandkumar, A. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

McCoy, R. T., Yao, S., Friedman, D., Hardy, M., and Griffiths, T. L. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *arXiv preprint arXiv:2309.13638*, 2023.

McDermott, D., Ghallab, M., Howe, A. E., Knoblock, C. A., Ram, A., Veloso, M. M., Weld, D. S., and Wilkins, D. E. Pddl-the planning domain definition language. 1998.

Nieuwenhuis, R. and Oliveras, A. On sat modulo theories and optimization problems. In *Theory and Applications of Satisfiability Testing-SAT 2006: 9th International Conference, Seattle, WA, USA, August 12-15, 2006. Proceedings 9*, pp. 156–169. Springer, 2006.

Olmo, A., Sreedharan, S., and Kambhampati, S. Gpt3-to-plan: Extracting plans from text using gpt-3. *FinPlan 2021*, pp. 24, 2021.

OpenAI. Introducing chatgpt by openai, 2022. URL https://openai.com/blog/chatgpt.

OpenAI. Gpt-4 technical report, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Pan, L., Albalak, A., Wang, X., and Wang, W. Y. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023.

Rajvanshi, A., Sikka, K., Lin, X., Lee, B., Chiu, H.-P., and Velasquez, A. Saynav: Grounding large language models for dynamic planning to navigation in new environments. *arXiv preprint arXiv:2309.04077*, 2023.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J., Ellenberg, J. S., Wang, P., Fawzi, O., et al. Mathematical discoveries from program search with large language models. *Nature*, pp. 1–3, 2023.

Russell, S. J. and Norvig, P. *Artificial intelligence a modern approach*. London, 2010.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. R., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Silver, T., Hariprasad, V., Shuttleworth, R. S., Kumar, N., Lozano-Pérez, T., and Kaelbling, L. P. PDDL planning with pretrained large language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL https://openreview.net/forum?id=1QMMUB4zfl.

Simpson, R., McCluskey, T. L., and Zhao, W. Gipo: an integrated graphical tool to support knowledge engineering in ai planning. In *ECP-01*, pp. 445. Citeseer, 2001.

Stechly, K., Marquez, M., and Kambhampati, S. GPT-4 Doesn't Know It's Wrong: An Analysis of Iterative Prompting for Reasoning Problems. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

Stechly, K., Valmeekam, K., and Kambhampati, S. On the self-verification limitations of large language models on reasoning and planning tasks. *arXiv preprint arxiv:2402.08115*, 2024a.

Stechly, K., Valmeekam, K., and Kambhampati, S. Chain of thoughtlessness: An analysis of cot in planning. *arXiv preprint arxiv:2405.04776*, 2024b.

Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Ullman, T. Large language models fail on trivial alterations to theory-of-mind tasks. *arXiv preprint arXiv:2302.08399*, 2023.

Valmeekam, K., Marquez, M., and Kambhampati, S. Can large language models really improve by self-critiquing their own plans? In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023a.

Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., and Kambhampati, S. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023b. URL https://openreview.net/forum?id=YXogl4uQUO.

Valmeekam, K., Marquez, M., Sreedharan, S., and Kambhampati, S. On the planning abilities of large language models - a critical investigation. In *Thirty-seventh Conference on Neural Information Processing Systems (Spotlight)*, 2023c. URL https://openreview.net/forum?id=X6dEqXIsEW.

Verma, M., Bhambri, S., and Kambhampati, S. On the brittle foundations of react prompting for agentic large language models. *arXiv preprint arXiv:2405.13966*, 2024a.

Verma, M., Bhambri, S., and Kambhampati, S. Theory of mind abilities of large language models in human-robot interaction: An illusion? *arXiv preprint arXiv:2401.05302*, 2024b.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models, 2023a.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023b.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Weng, Y., Zhu, M., Xia, F., Li, B., He, S., Liu, S., Sun, B., Liu, K., and Zhao, J. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2550–2575, 2023.

Xie, J., Zhang, K., Chen, J., Zhu, T., Lou, R., Tian, Y., Xiao, Y., and Su, Y. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arxiv:2402.01622*, 2024.

Xie, Y., Yu, C., Zhu, T., Bai, J., Gong, Z., and Soh, H. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, 2023.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. R. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.