

Can LLMs Reason & Plan?

Subbarao Kambhampati



Tutorial Slides

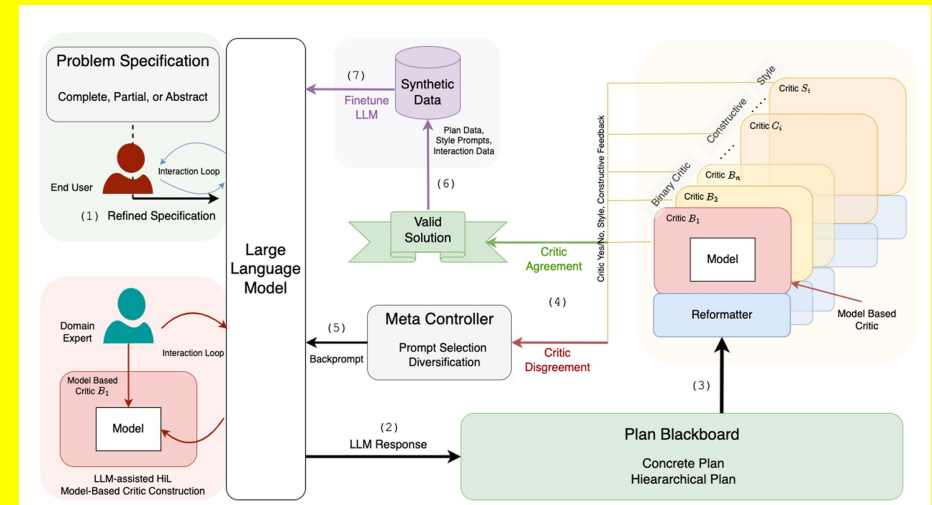


<https://bit.ly/4dbkkY2>



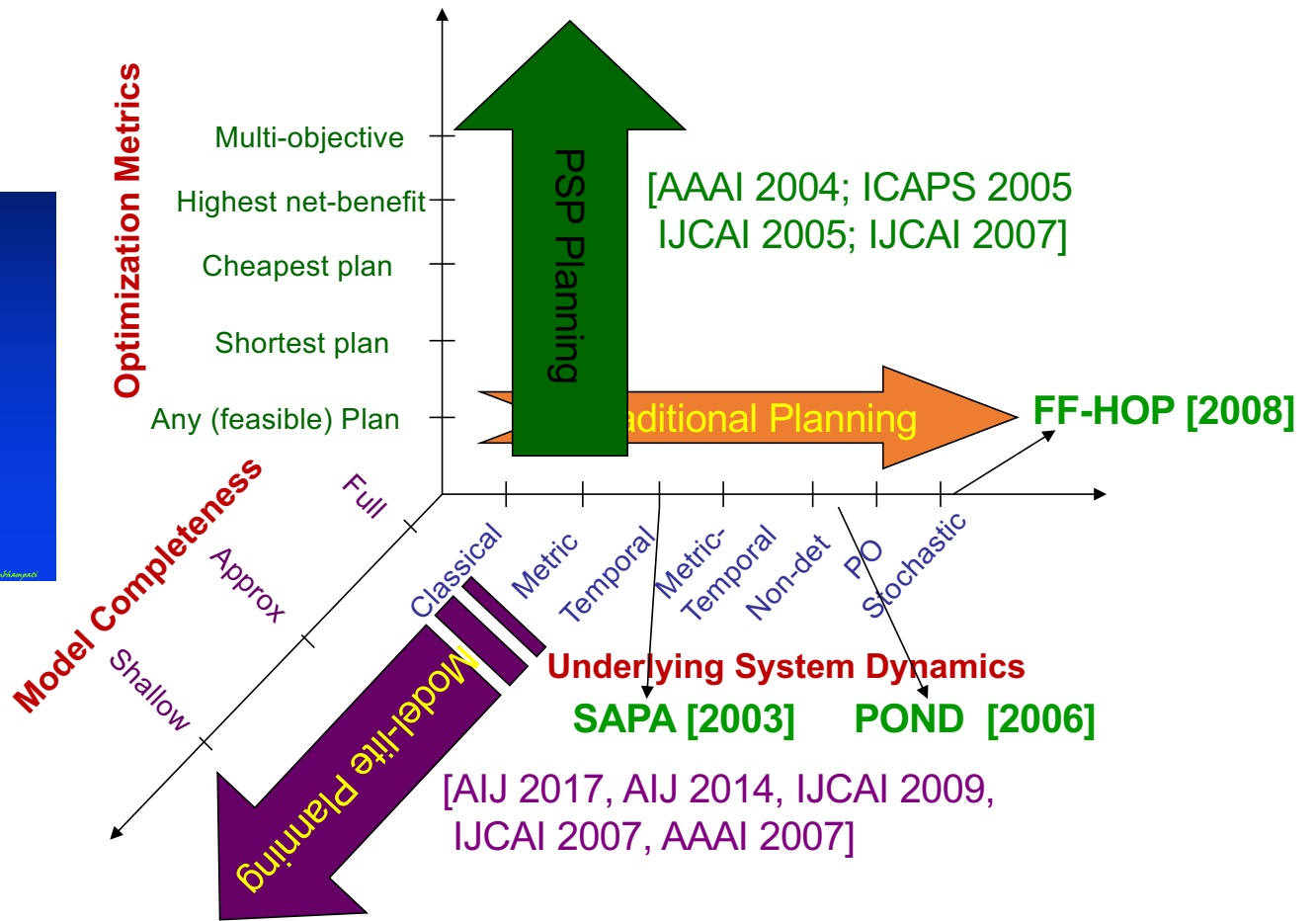
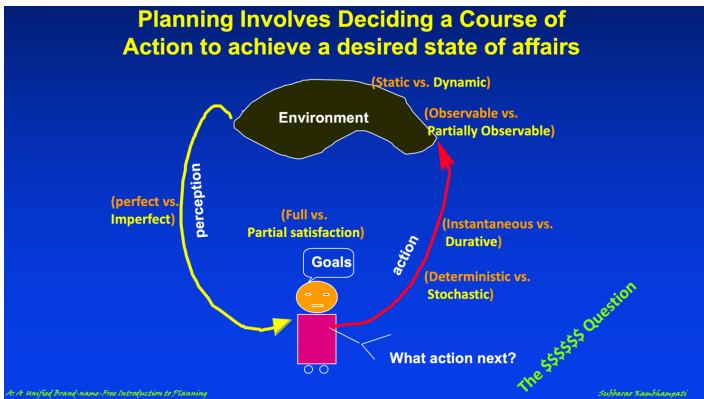
Tutorial Overview

- **Part 1**
Introduction; Aims; Perspective on LLMs, Planning, Tutorial Big Picture
- **Part 2**
 - Evaluating LLM Planning capabilities in Autonomous mode, including effect of
 - Prompting strategies (including Chain-of-thought)
 - Fine Tuning
 - Self Verification
 - Understanding the contradictory claims in the literature
- **Part 3**
 - Sane roles of LLMs in Planning (with LLM-Modulo frameworks)
 - LLMs as heuristics, LLMs as candidate generators
 - Back prompting from external verifiers
 - LLMs as sources of domain models (with humans in the loop)
 - LLMs as format changers/specification elaborators¹
- **Part 4**
 - Summary/Lessons



Aim of the Tutorial

- Of late, there has been a significant rise in interest in using Large Language Models in planning tasks
- In the last ~3 years, many papers have been published
 - It started with training sequence learning models (specifically transformers) *tabula rasa* on plans
 - ..and has currently mostly become a sort of *ersatz natural science* of using huge **pre-trained models** to see (and exploit) what planning abilities they may already possess
- Much of the literature is in NLP and ML conferences.
 - There is bidirectional ignorance.
 - The authors of the papers often don't have background in Planning/Reasoning
 - and the Planning/Reasoning community is largely unaware of the work on LLMs and Planning
- This tutorial is an attempt to rectify this ignorance, and to take a critical look at the role of LLMs in Planning
 - (..and should thus be of interest to both populations..)

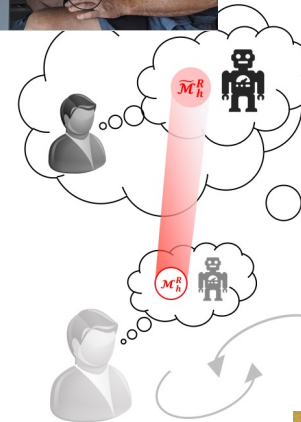


Information Gathering;
Information Integration

RL with Simulator is Planning..

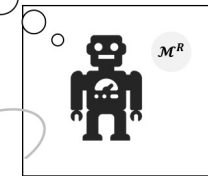
Research Background..

- We have focused on explainable human-AI interaction.
- Our setting involves collaborative problem solving, where the AI agents provide decision support to the human users in the context of *explicit knowledge sequential decision-making tasks* (such as mission planning)
 - In contrast, much work in social robotics and HRI has focused on tacit knowledge tasks (thus making explanations mostly moot)
 - We assume that the AI agent either learns the human model or has prior access to it.
- We have developed frameworks for proactive explanations based on *model reconciliation* as well as on-demand *foil-based explanations*
- We have demonstrated the effectiveness of our techniques with systematic (IRB approved) human subject studies



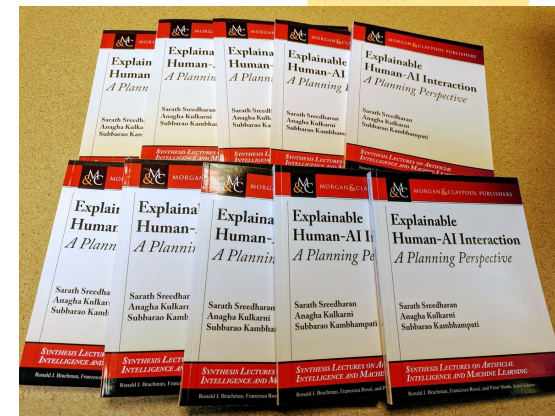
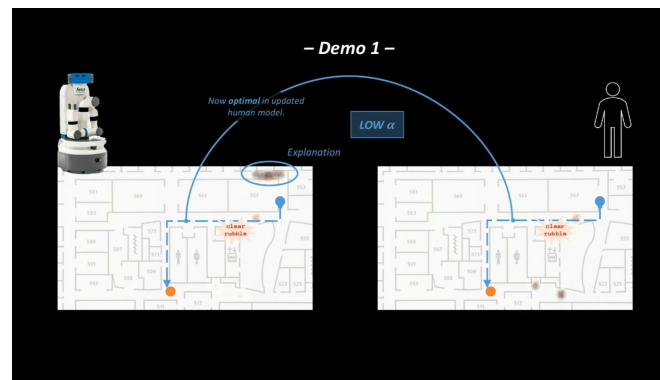
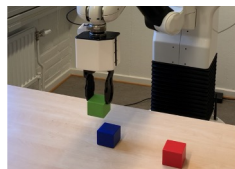
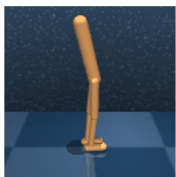
\bar{M}_h^R : Allows the agent to anticipate human expectations, in order to

- conform to those expectations
- explain its own behavior in terms of those expectations.



M^H and \bar{M}_h^R are Expectations on Models M^H and M^R

They don't have to be executable



Browse


kambhampati x title author session

mini compact detail

shuffle by serendipity

showing 3 of 3 papers

On the Planning Abilities of Large Language Models - A Critical Investigation
 Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, Subbarao Kambhampati
 Tu, Dec 12, 09:45 -- Poster Session 1



Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning
 Lin Guan, Karthik Valmeekam, Sarath Sreedharan, Subbarao Kambhampati
 Tu, Dec 12, 16:15 -- Poster Session 2

PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change
 Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, Subbarao Kambhampati
 We, Dec 13, 09:45 -- Poster Session 3



COMMENTARY
Can large language models reason and plan?

Subbarao Kambhampati

School of Computing & Augmented Intelligence, Arizona State University
 Correspondence: Email: rao@asu.edu

Abstract
 While humans sometimes do show the capability of correcting their own erroneous guesses with self-critiquing, there seems to be no basis for that assumption in the case of LLMs.

Large language models (LLMs), essentially n-gram models on steroids¹ that have been trained on web-scale language corpora (or, effectively, our civilizational knowledge), have caught our collective imagination with linguistic behaviors that no one expected text completion systems to possess.² By training and operation, LLMs are perhaps best seen as giant non-veridical memories akin to an external System 1 (Ref. 2) for us all (see Figure 1). Their seeming versatility has however led many researchers to wonder whether they can also do well on planning and reasoning tasks typically associated with System 2 competency.
 Nothing in the training and use of LLMs would seem to suggest remotely that they can do any type of principled reasoning (which, as we know, often involves computationally hard inference/search). What

national Planning Competition (IPC)—including the well-known Blocks World.³ Our results⁴ were contrary to the anecdotal claims about the planning abilities of LLMs, and when we made them public, received significant attention in the AI circles.
 By the beginning of 2023, with the wide-spread public release of ChatGPT, and later, GPT4, there were a slew of additional claims, including in refereed papers, about LLMs' abilities to reason and plan. So we decided to repeat our tests on both GPT3.5 and GPT4.⁵ Initial results showed that there was some improvement in the accuracy of generated plans from GPT3 to GPT4, with GPT4 reaching 30% empirical accuracy in the Blocks World (albeit still lower in other domains). We then wanted to know whether the modest improvement

Computer Science > Artificial Intelligence
 [Submitted on 2 Feb 2024 (v1), last revised 6 Feb 2024 (this version, v2)]

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

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldy, Anil Murthy

There is considerable confusion about the role of Large Language Models (LLMs) in planning and reasoning tasks. On one side are over-optimistic claims that LLMs can indeed do these tasks with just the right prompting or self-verification strategies. On the other side are perhaps over-pessimistic claims that LLMs are good for in planning/reasoning tasks are as mere translators of the problem specification from one syntactic format to another, and ship the problem off to external symbolic solvers. In this position paper, we take the view that both these extremes are misguided. 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 will 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 (bf LLM-Modulo Frameworks) that combine 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, and allows extending the scope of the framework towards more flexible knowledge, problem and preference specifications.

Subjects: Artificial Intelligence (cs.AI); Machine Learning (cs.LG)
 Cite as: arXiv:2402.01817 [cs.AI]
 (or arXiv:2402.01817v2 [cs.AI] for this version)

ICML 2024 Spotlight
 Tuesday 1:30pm

Computer Science > Artificial Intelligence
 [Submitted on 12 Feb 2024]

On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks

Kaya Stechly, Karthik Valmeekam, Subbarao Kambhampati

There has been considerable divergence of opinion on the reasoning abilities of Large Language Models (LLMs). While the initial optimism that reasoning might emerge automatically with scale has been tempered thanks to a slew of counterexamples—ranging from multiplication to simple planning—there persists a wide spread belief that LLMs can self-critique and improve their own solutions in an iterative fashion. This belief seemingly rests on the assumption that verification of correctness should be easier than generation—a rather classical argument from computational complexity—which should be irrelevant to LLMs to the extent that what they are doing is approximate retrieval. In this paper, we set out to systematically investigate the effectiveness of iterative prompting in the context of reasoning and planning. We present a principled empirical study of the performance of GPT-4 in three domains: Game of 24, Graph Coloring, and STRIPS planning. We experiment both with the model critiquing its own answers and with an external correct reasoner verifying proposed solutions. In each case, we analyze whether the content of criticisms actually affects bottom line performance.

Computer Science > Artificial Intelligence
 [Submitted on 6 Feb 2024]

"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors

Lin Guan, Yifan Zhou, Denis Liu, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati

Large-scale generative models are shown to be useful for sampling meaningful candidate solutions, yet they often overlook task constraints and user preferences. Their full power is better harnessed when the models are coupled with external verifiers and the final solutions are derived iteratively or progressively according to the verification feedback. In the context of embodied AI, verification often solely involves assessing whether goal conditions specified in the instructions have been met. Nonetheless, for these agents to be seamlessly

Computer Science > Robotics
 [Submitted on 10 Jan 2024 (v1), last revised 17 Jan 2024 (this version, v2)]

Theory of Mind abilities of Large Language Models in Human-Robot Interaction : An Illusion?

Mudit Verma, Siddhant Bhambri, Subbarao Kambhampati

Large Language Models have shown exceptional generative abilities in various natural language and generation tasks. However, possible anthropomorphization and leniency towards failure cases have propelled discussions on emergent abilities of Large Language Models especially on Theory of Mind (ToM) abilities in Large Language Models. While several false-belief tests exist to verify the ability to infer and maintain mental models of another entity, we study a special application of ToM abilities that has higher stakes and possibly irreversible consequences : Human Robot Interaction. In this work, we explore the task of Perceived Behavior Recognition, where a robot employs a Large Language Model (LLM) to assess the robot's generated behavior in a manner similar to human observer. We focus on four behavior types, namely – explicable, legible, predictable, and obfuscatory behavior which have been extensively used to synthesize interpretable robot behaviors. The LLMs goal is, therefore to be a human proxy to the agent, and to answer how a certain agent behavior would be perceived by the human in the loop, for example "Given a robot's behavior X, would the human observer find it explicable?". We conduct a human subject study to verify that the users are able to correctly answer such a question in the curated situations (robot setting and plan) across five domains. A first analysis of the belief test yields extremely positive results inflating ones expectations of LLMs possessing ToM abilities. We then propose and perform a suite of perturbation tests which breaks this illusion, i.e. Inconsistent Belief, Uninformative Context and Conviction Test. We conclude that, the high score of LLMs on vanilla prompts showcases its potential use in HRI settings, however to possess ToM demands invariance to trivial or irrelevant perturbations in the context which LLMs lack.

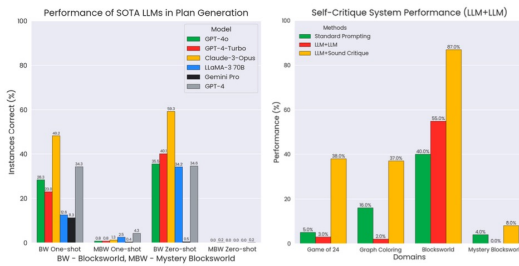


Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks

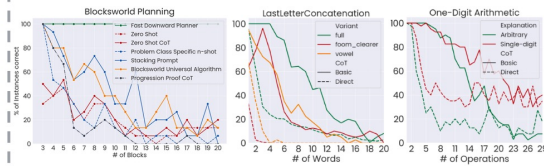
Subbarao Kambhampati, Karthik Valmееkam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, Anil B Murthy
School of Computing & AI, Arizona State University



LLMs can't plan



Chain of thought w/ GPT-4-Turbo



ReAct's inability to generalize

Task success rate %, average across 6 tasks: pick, clean, heat, cool, examine, puttwo.

See the gradual drop in performance!

	Base	Replace object names to synonyms	Example Goal location = Query Goal Location	Some examples of different task	All Examples of different task	Examples of each of the tasks	Unrolling: Example task is extended in query	Subtask: Example task has query as subtask
3.5-turbo	25	1.6	30	12	1.6	14	Drops from 52% to 9%	Drops from 18% to 0%
3.5-instruct	54	47	42	18	5.2	Context Window Too Short		

LLMs can't plan; But they can help planning in LLM-Modulo Frameworks

LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

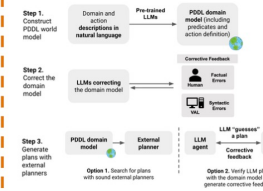
- LLMs can't do planning in autonomous mode [1]
- CoT [2], ReACT [3], Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!) [4]
- Having humans iteratively prompt is an invitation for Clever Hans effect.

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

- LLMs can be used in conjunction with external verifiers and solvers in an LLM-Modulo framework (with the verifiers doing back prompting)
- In the LLM-Modulo framework, LLMs can play multiple roles
 - > Guess plans and domain models
 - > Help elaborate the problem specification
 - > Translate formats

Teasing out critics from LLMs

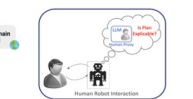
LLMs as an approximate source of domain models (for helping construct correctness critics) [5]



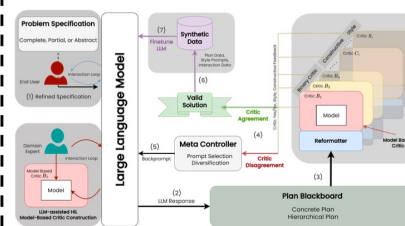
Style critics: LLMs as an approximate repository of common human preferences [7]



Style critics: LLMs as an approximate human proxy for checking explicability, legibility etc. [8]



LLM-Modulo Frameworks!



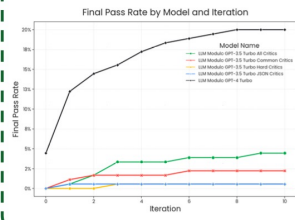
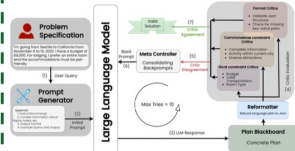
- LLM-Modulo is a generate-test framework with LLMs generating candidate plans and critiquing them
- LLMs play a variety of constructive roles
 - Generate candidate plans
 - Be an approximate source of models driving the correctness critics [5]
 - Act as style critics [7, 8]
 - Help collate the criticisms from critics (and diversify the prompts as needed)
 - Help with format change – specification level, converting to critic representations [10]
- Preference for critics over solvers
 - Correctness vs. Style
 - Binary vs. Critical feedback vs. Constructive critics
- Human intervention is minimized
 - Once per domain: Teasing out domain model
 - Once per problem: Specification elaboration
 - Humans are not required to be in the inner loop of the back-prompting search

LLM-Modulo Case Studies

1. Results on PlanBench [9]

Domains	Instances Correct (GPT-4)	Average Feedback Rounds
Blocksworld (BW)	41/50 (82%)	3.68
Logistics	35/50 (70%)	3.31
Mystery BW	5/50 (10%)	7.0

2. Results on Travel Planning Benchmark [6]



References

- [\[1\] https://arxiv.org/abs/2305.18171](https://arxiv.org/abs/2305.18171)
- [\[2\] https://arxiv.org/abs/2405.04776](https://arxiv.org/abs/2405.04776)
- [\[3\] https://arxiv.org/abs/2405.13966](https://arxiv.org/abs/2405.13966)
- [\[4\] https://arxiv.org/abs/2402.08105](https://arxiv.org/abs/2402.08105)
- [\[5\] https://arxiv.org/abs/2305.16509](https://arxiv.org/abs/2305.16509)
- [\[6\] https://arxiv.org/abs/2405.20825](https://arxiv.org/abs/2405.20825)
- [\[7\] https://arxiv.org/abs/2402.04210](https://arxiv.org/abs/2402.04210)
- [\[8\] https://arxiv.org/abs/2401.05302](https://arxiv.org/abs/2401.05302)
- [\[9\] https://arxiv.org/abs/2209.10489](https://arxiv.org/abs/2209.10489)
- [\[10\] https://arxiv.org/abs/2105.07131](https://arxiv.org/abs/2105.07131)

Tuesday, 1:30pm;
Hall C4-9, #710

LLM's Can't Plan; But they can help planning in LLM-Modulo Frameworks

LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

- LLMs can't do planning in autonomous mode
- CoT, ReACT, Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!)
- Having humans iteratively prompt is an invitation for Clever Hans effect..

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

- LLMs can be used in conjunction with external verifiers and solvers in an **LLM-Modulo** framework (with the verifiers doing *back prompting*)
 - In the LLM-Modulo framework, LLMs can play multiple roles
 - Guess plans
 - Guess domain models
 - Help elaborate the problem specification
 - Translate formats



I come to leverage LLMs, not to lament them..



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) ✓
@rao2z

A clear-eyed understanding of the strengths **and** limitations of a technology is a step towards advancing it.

Blind cheerleading or unalloyed cynicism, in contrast, are just steps towards advancing your influencer career..

✎ Last edited 9:04 AM · Nov 10, 2023 · 25.1K Views

📊 View post engagements

💬 5

↻ 27

❤️ 118



*.. O judgment! thou art
fled to brutish beasts,
And men (& LLMs) have lost their reason.*

COMMUNICATIONS OF THE ACM

HOME CURRENT ISSUE NEWS **BLOGS** OPINION RESEARCH PRACTI

Home / Blogs / BLOG@CACM / AI as (an Ersatz) Natural Science? / Full Text

BLOG@CACM

AI as (an Ersatz) Natural Science?

By Subbarao Kambhampati

June 8, 2022

Comments (1)

VIEW AS: SHARE:



In many ways, we are living in quite a wondrous time for AI, with every week bringing some awe-inspiring feat in yet another [tacit knowledge](#) task that we were sure would be out of reach of computers for quite some time to come. Of particular recent interest are the large learned systems based on transformer architectures that are trained with billions of parameters over massive Web-scale multimodal corpora. Prominent examples include [large language models](#) like GPT3 and PALM that respond to free-form text prompts, and language/image models like DALL-E and Imagen that can map text prompts to photorealistic images (and even those with claims to general behaviors such as GATO).

The emergence of these large learned models is also changing the nature of AI research in fundamental ways. Just the other day, some researchers were playing with DALL-E and thought that it seems to have [developed a secret language of its own](#) which, if we can master, might allow us to interact with it better. Other researchers found that GPT3's responses to reasoning questions can be improved by [adding certain seemingly magical incantations to the](#)



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) ✓

@rao2z

The problem with Alchemy is 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 problem with LLM alchemy may not be much different.. 😞



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) ✓ @rao2z · Sep 17, 2023

Replying to @rao2z

Bringing sledge hammer subtleness to the foregoing analogy:

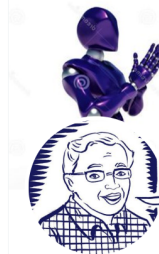
What LLMs do well is to serve as giant external non-veridical memories for us--memories trained on a HUGE fraction of civilizational knowledge....

[Show more](#)



A few caveats about the tutorial..

- This is **not a dry chronological survey** with a laundry list of papers and their blurbs
- It is an opinionated perspective on the state of LLMs and Planning intersection
 - Informed by our own work in the area
- Caution: The authors of the papers brought up in the tutorial may well bristle at the particular perspective/pigeonholing of their work
 - The aim is not to make up your mind, but to equip you with a perspective that you may find useful when you read the literature yourself (..or do work in the area yourself..)
- This tutorial has largely been tweeted..
 - “twittorial?”



*Human, grant me the serenity to accept the things I cannot learn;
Data to learn the things I can;
And wisdom to know the difference.*

Edit profile

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z
AI researcher & teacher @SCAL_ASU. Works on Human-Aware AI. Former President of @RealAAAI; Chair of @AAS Sec T. Here to tweek #AI. YouTube Ch: bit.ly/38twrAV
Tempe, AZ rakaposhi.eas.asu.edu Joined October 2014
29 Following 14.7K Followers

Posts Replies Highlights Media Likes

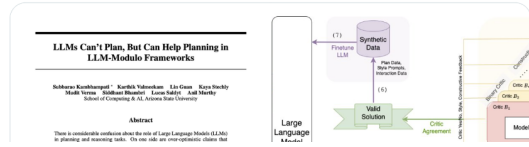
Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Jan 29 ...
Despite mounting evidence that LLMs can't possibly be expected to do verification/critiquing of what they generate (unless they have been trained on *corrections* data in addition to *correct* data), the number of papers continuing to claim that LLMs can improve by "self-critiquing"...

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao · Oct 22, 2023
So my thread about our papers investigating the verification and self-critiquing inabilities of GPT4 has apparently resonated with a lot of folks. Here is a quick response to several issues raised (either in replies or other quote-tweet threads). [Using one of 'em long "vanity... x.com/rao2z/status/1... Show more

6 18 69 15K

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Feb 10 ...
Wrote up a more complete vision of the LLM-Modulo Frameworks for planning/reasoning tasks.

arxiv.org/abs/2402.01817





Perspective on LLMs

Tutorial Slides



<https://bit.ly/4dbkkY2>

BLOG@CACM

Artificial Intelligence and Machine Learning

Language Imitation Games and the Arrival of Broad and Shallow AI

By Subbarao Kambhampati

Posted Oct 7 2021

Share Print Join the Discussion



When my son was still a toddler and his mom had to go on an extended trip out of the country, he would "talk" to her on the phone almost daily. Scare quotes because he still was more babbling than talking. But, the impressive (and adorable) thing was that his imitation of the syntactics of us talking on the phone was flawless, replete with the meaningful

Advertisement



The New York Times

Quotation of the Day: When Chatbots 'Hallucinate'

Give this article

May 8, 2023

"If you don't know an answer to a question already, I would not give the question to one of these systems."

SUBBARAO KAMBHAMPATI, a professor and researcher of artificial intelligence at Arizona State University, about inaccuracies in information from chatbots, sometimes referred to as "hallucinations."

Give this article

THE VIEWS EXPRESSED BY CONTRIBUTORS ARE THEIR OWN AND NOT THE VIEW OF THE HILL

Beauty, lies & ChatGPT: Welcome to the post-truth world

BY SUBBARAO KAMBHAMPATI, OPINION CONTRIBUTOR - 02/16/23 10:00 AM ET

Share Post More



AP Photo/Timothy D. Easley

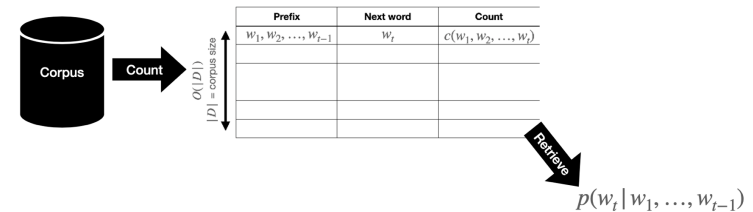
Bella Whitice talks with classmate Katherine McCormick as they try and outwit the "robot" that was creating writing assignments in Donnie Piercey's class at Stonewall Elementary in Lexington, Ky., Monday, Feb. 6, 2023. The robot was the new artificial intelligence tool ChatGPT which can generate everything from essays and haikus to term papers in a matter of seconds.

Two months back, a company called OpenAI released its chatbot, ChatGPT, to the public. ChatGPT is a so-called Large Language Model (LLM) that is trained on the nearly 600 gigabytes of text of all kinds found on the World Wide Web to learn to complete any text prompt by predicting the next word, and the word after that, and so on. The purported aim of the system is to put the "auto complete" functionality for words, found on cellphones, on steroids so it can complete entire paragraphs.

The powers of these types of LLMs have long been known in the technology sector, thanks to ChatGPT's predecessor, GPT3, from OpenAI and similar systems from other Big Tech companies.

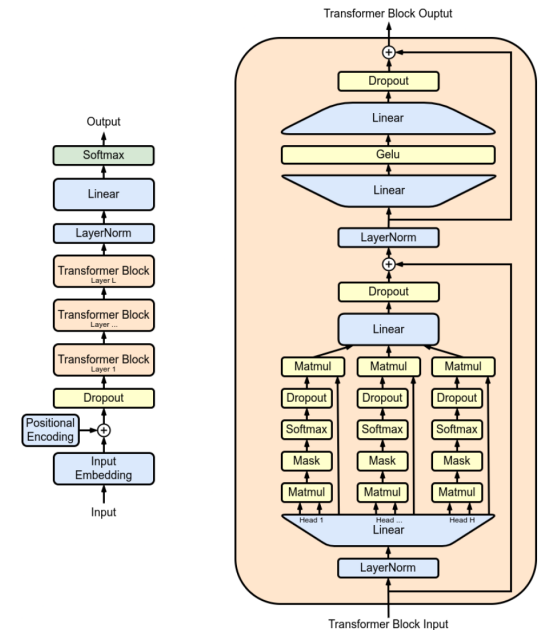
LLMs are N-gram models on STEROIDS

- Text is a long sequence of words (including spaces, punctuations)
- An n-gram model of language learns to predict n-th word given the preceding n-1 words
 - Probabilistically speaking it learns $\Pr(W_n | W_1 \dots W_{n-1})$
 - Unigram predicts each word independently (no preceding context)
 - Bigram predicts each word given the previous word
 - A 3001-gram model learns to predict the next word given the previous 3000 words
 - ChatGPT is just a 3001-gram model
- The power of an n-gram model depends on
 - How much text it trains on
 - How big is the **n** (context) and
 - How high-capacity is the function learning $\Pr(W_n | W_1 \dots W_{n-1})$
- ChatGPT trains on ~600 gigabytes of text on the Web
 - It learns a very high capacity function that has **175 billion** parameters
 - Learns $\Pr(W_n | W_1 \dots W_{n-1})$ for all possible nth words W_n (**Vocabulary of the language, ~50K in English**)



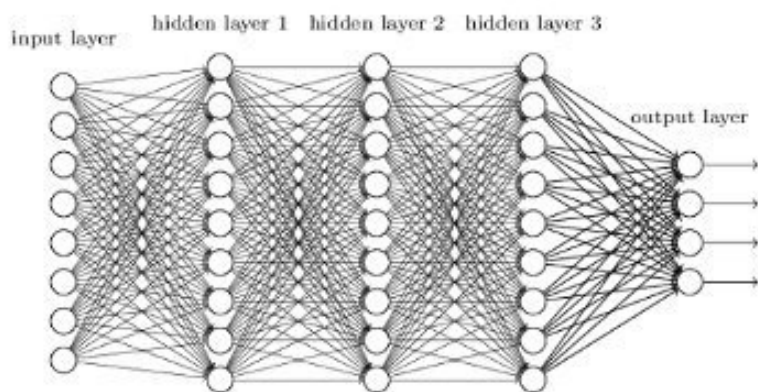
..but the count table is Ginormous! (and is VERY sparse)

- With an n-gram model, you need to keep track of the conditional distributions for (n-1)-sized prefixes.
- With a vocabulary size $|V|$ (~ 50000), there are $|V|^{n-1}$ different prefixes!!
 - Easy for unigram (1 prefix), bigram ($|V|$ prefixes) and trigram ($|V|^2$ prefixes)
 - For ChatGPT's 3001-gram model, with a 50,000 word vocabulary, we are looking at a whopping $(50000)^{3000}$ conditional distributions
 - (and most entries will be zero—as the chance of seeing the same 3000-word sequence again is vanishingly small!)
- What LLMs do is to essentially *compress/approximate* this ginormous count table with a function
 - That is while high capacity (176 billion weights!) is still **vanishingly small** compared to the ginormous count ($(50000)^{3000} \gg 176$ billion or a trillion!)
 - ..and oh by the way, the compressed function winds up having fewer zeros
 - It approximates both the non-zero counts and zero counts, so..
 - **GENERALIZATION!!!**
 - In essence the function learns to “abstract” and “cluster” over “similar” sequences

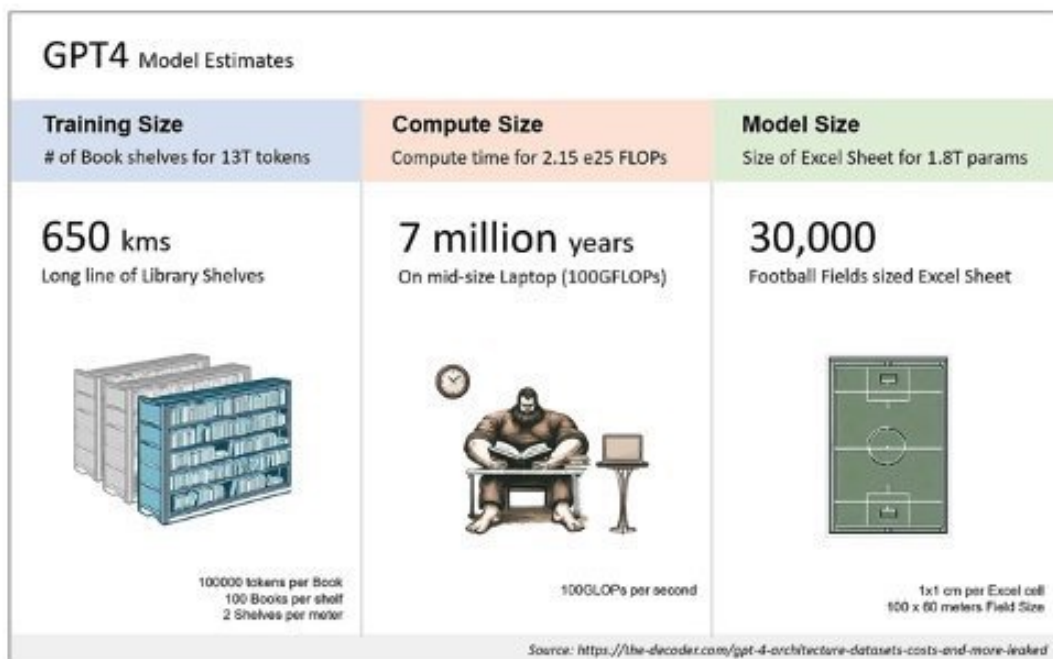


Transformers are a (not particularly principled) parallelization of the recurrent neural networks

Mental picture



Reality



GPT4 Model Estimates

(graphic by James Campbell)

Hallucination and “Approximate Retrieval”

- LLMs are n-gram models, and thus do not index and retrieve
- All they ever do is *hallucinate* completions to the prompt
 - Such that the completion is in the same distribution as the text they have been trained on
- Prompt engineering doesn't change this!
 - Whether or not changing the prompt gives the “factual completion” depends on the prompter *knowing enough* to tell whether the given answer is the accurate one.

The New York Times

Quotation of the Day: When Chatbots 'Hallucinate'

 Give this article  

May 8, 2023

“If you don't know an answer to a question already, I would not give the question to one of these systems.”

[SUBBARAO KAMBHAMPATI](#), a professor and researcher of artificial intelligence at Arizona State University, about inaccuracies in information from chatbots, sometimes referred to as “hallucinations.”

 Give this article  

LLMs and Approximate Retrieval

- Retrieval in Databases: Given a query (key), retrieve the records that *exactly* match the query
- Retrieval in IR systems (e.g. Google): Given a (textual) query, retrieve all the records that are *similar* to the query
 - The records themselves are not modified in any way
- Approximate retrieval in LLMs: Given a (textual) query (prompt), generate the most likely completion
 - Note that the completion is NOT guaranteed to be one of the stored records
 - This generative creativity is the boon/bane of LLMs

4. Commercial LLM makers (will) try to play both ends of the approximate retrieval to their advantage..

4.1. When they try to argue NYT law suit, they will no doubt push on the fact that LLMs don't do exact retrieval and so there is no copyright infringement.

4.2 When they push LLMs for "search", they will try instead to bank on the memorization capabilities!

The truth is that there is no 100% way to guarantee or stop either behavior!

If LLM makers try to reduce memorization, they will certainly see that the LLM's ability to masquerade as search engines--already quite questionable (c.f. x.com/rao2z/status/1...) --will degrade even further (c.f. x.com/rao2z/status/1...)



LLM's Approximate Retrieval upends our intuitions re: their guesses

Computational Complexity of the underlying task has no bearing on LLM guesses

- The underlying complexity of the problem has no impact on the LLM's ability to *guess* the answer
 - They are just as fast in guessing answers to undecidable questions as they are in guessing answers to constant time questions
 - ..and in neither case do they have any guarantees about their guess
- Corollary: The usual problem characteristic—*Stochasticity, Partial Observability etc.* — that make it computationally harder *don't matter* in LLM's ability to guess
- After all, they take constant time per token
 - ..and no, asking LLMs to “pause” doesn't change any of this!

Background Knowledge is *easier* for LLMs (approximately..)

- Much has been made in traditional AI of the difficulty of getting relevant knowledge.
- Having been trained on web-scale collective knowledge of humanity, LLMs are remarkably better at this
- They are pretty good (with no guarantees— and some brittleness) at
 - Commonsense
 - Domain knowledge
 - Theory of Mind
 - Analogies
- (In addition, of course, to linguistic abilities such as summarization, elaboration, format change etc.)

Perspective on Planning

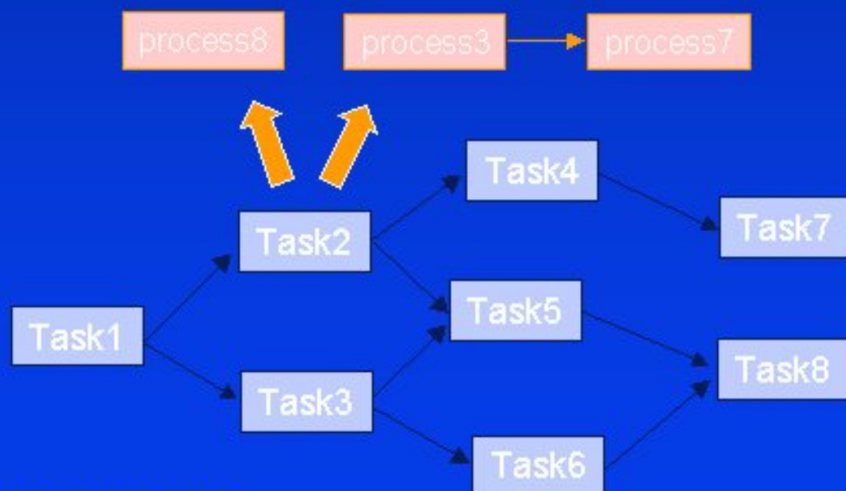
Tutorial Slides



<https://bit.ly/4dbkkY2>

Planning, Scheduling, Reinforcement Learning

- **Planning/Sequential Decision Making:** Given a set of objectives, come up with a course of actions (policy) to achieve them (optimally)
 - Standard planning (and RL) assume the possible actions that the agent can take (and their “models”) are given up front
 - MacGyver Planning involves *coming up with the actions AND sequencing them*
- **Scheduling:** Given a set of tasks/jobs, and a choice of actions for accomplishing each of the tasks, assign actions to tasks so there are no undesired interactions
 - Planning is more general than scheduling (One difference is *precondition cascading* – adding an action to achieve something makes you subgoal on the the preconditions of that action..).
 - Planning is P-Space Complete (even for STRIPS) and Scheduling is NP-Hard
- **(Model-Based) Reinforcement Learning:** The agent acts in a (hopefully *ergodic*) environment and slowly learns the “action models” —and plans with that model.



Ordering choices
Resource choices
Process choices

Ergodicity, Robustness & Quality/Optimality

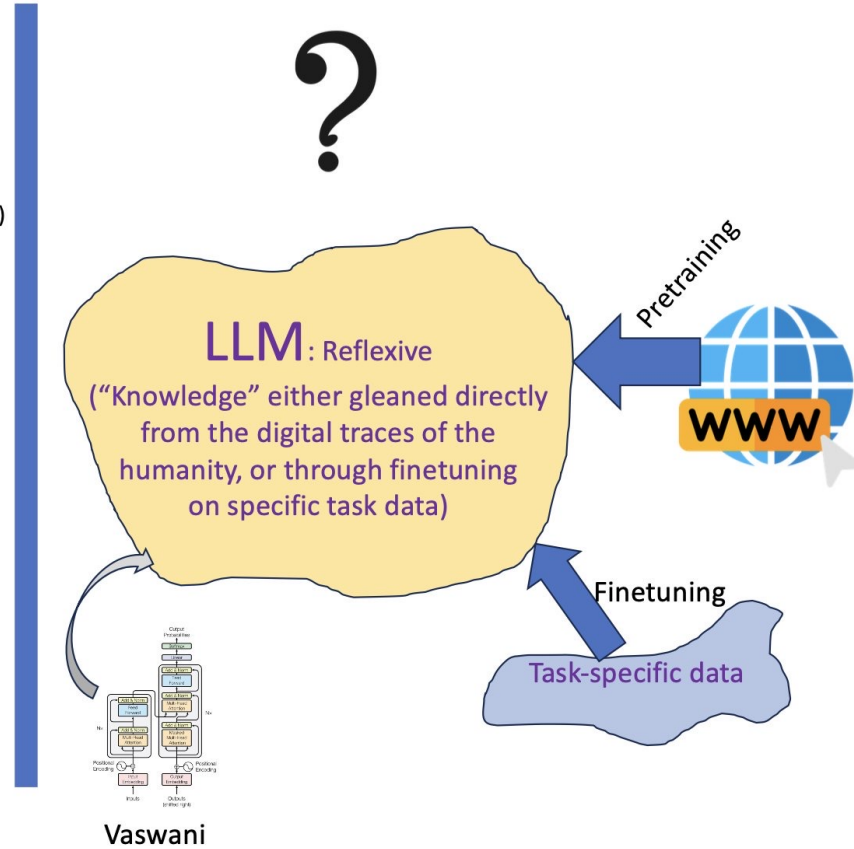
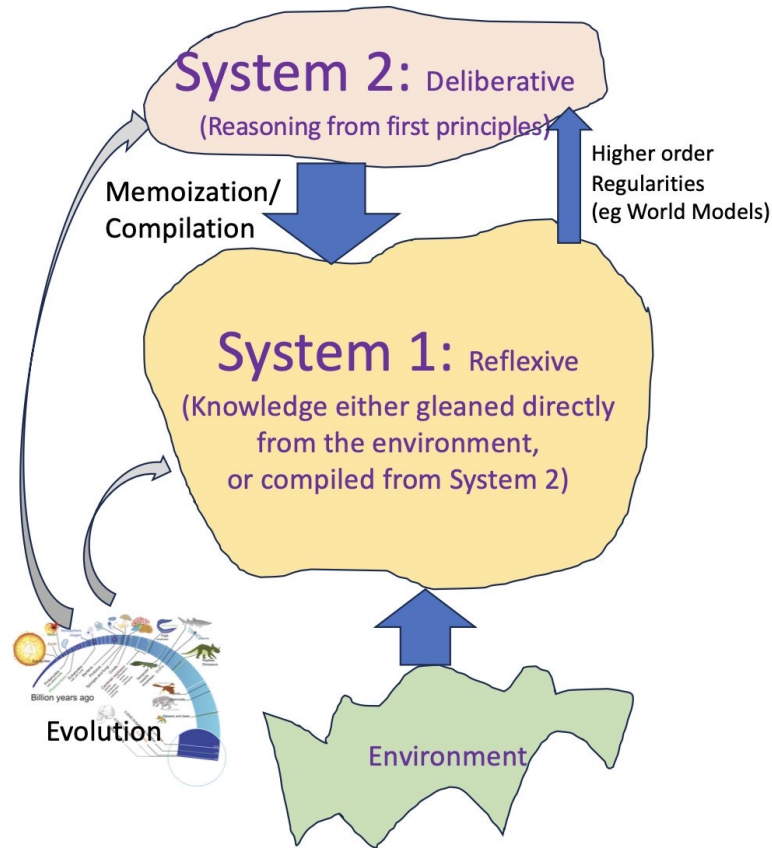
- An environment is **ergodic** if the agent can reach *any* state of an environment from any other state (with positive probability). As a corollary, in non-ergodic environments, an agent can “get stuck” in some (group of) states
 - Planning is most critical in **non-ergodic** environments!
 - Also, if the agent is interacting with a resettable simulator rather than the real world, it can effectively mask the non-ergodicity of the real world.
- **Robustness** of a plan is (informally) the probability that it achieves its objectives (without failing/getting stuck)
 - In deterministic environments, “correctness” is a binary measure of robustness
 - In non-ergodic environments, a high degree of robustness is critical. Conversely in ergodic ones (or when you are working with simulators), you can “blunder on” so to say..
 - Top-k correctness is not sufficient in non-ergodic environments..
- While robustness is a measure of correctness, **Quality** of a plan is more about its style (as desired by the agent or the problem specifier)
 - Quality captures intangibles such as implicit (unstated) preferences
 - For example, a travel plan to go from Phoenix to Vienna by going part way by walk, part way by bike, part way by car etc. etc. can be **correct**, but may not conform to the implicit preferences
 - Interestingly, LLMs, trained as they are on web-scale corpora of our collective output, tend to capture these style preferences better (at least as compared to any other alternate GOFAI approaches!)

World Models, Verifiers & Simulators

- The way a planning agent verifies the robustness of the course of actions it synthesizes—*without having to execute it in the world and incur costly failures*—is to verify the plan with respect to *some model of the world*
 - This model can be something externally supplied
 - PDDL Models; SMT Models etc.
 - Domain Simulators
 - (Note that *simulators don't fall from the sky*; they are written by someone!)
 - Or learned directly from the agent by trial and error over the real world!
 - As is the case in Reinforcement Learning
- The big unresolved question with LLMs is to what extent they have a robust internal model of the world
 - Since if they do, they can guess a plan, self-verify/critique, and loop..

So can LLMs do Planning?

Little *a priori* reason to believe that LLMs can reason/plan



On the other hand, the literature seems rife with claims of LLM planning abilities..

LEAST-TO-MOST PROMPTING ENABLES COMPLEX REASONING IN LARGE LANGUAGE MODELS

Denny Zhou^{1*}
Dale Schuurman¹
¹Google Research

Chain-of-thought prompts, a novel strategy to and then answers related to solving problems code-split) with 16% accuracy because trained are included

Chain-of-Symbol Prompting Elicits Planning in Large Language Models

Hanxu Hu^{1*} Hongyuan Lu^{2*†} Huajian Zhang³ Wai Lam² Yue Zhang¹
¹Westlake University ²The Chinese University of Hong Kong ³University of Edinburgh
{nuhanxu, zhangyue}@westlake.edu.cn
{hy1u, wlam}@se.cuhk.edu.hk
v1hzha17@exseed.ed.ac.uk

Abstract

In this paper, we first take the initiative to investigate the performance of LLMs on complex planning tasks that require LLMs to understand a virtual spatial environment simulated via natural language and act correspondingly in text. We propose a benchmark named Natural Language Planning and Action (Natala) composed of a set of novel tasks: Brick World, NLVR-based Manipulations, and Natural Language Navigation. We found that current popular LLMs such as ChatGPT still lack abilities in complex planning. This arises a question – do the LLMs have a good understanding of the environments described in natural language, or maybe other alternatives such as symbolic representations are neater and hence better to be understood by LLMs? To this end, we propose a novel method called CoS (Chain-of-Symbol Prompting) that represents the complex environments with condensed symbolic spatial representations during the chained intermediate thinking steps. CoS is easy to use and does not need additional training on LLMs. Extensive experiments indicate that CoS clearly surpasses the performance of the Chain-of-Thought (CoT) Prompting in all three planning tasks with even fewer tokens used in the inputs compared with CoT. The performance gain is strong, by up to 60.8% accuracy (from 31.8% to 92.6%) on Brick World for ChatGPT. CoS also reduces the number of tokens in the prompt obviously, by up to 65.8% of the tokens (from 407 to 139) for the intermediate steps from demonstrations on Brick World. Code and data available at: <https://github.com/hanxuhu/chain-of-symbol-planning>

Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents

Wenlong Hu
UC Berkeley

Can work interactively high-level set of actions from pre-trained decomposed ever, the actions. VirtualHd executabl trade-off extracting

Large Language Models are Zero-Shot Reasoners

Takeshi Kojima
The University of Tokyo
t.kojima@weblab.t.u-tokyo.ac.jp

Shixiang Shane Gu
Google Research, Brain Team

Machel Reid
Google Research*

Yutaka Matsuo
The University of Tokyo

Yusuke Iwasawa
The University of Tokyo

Abstract

Pretrained large language models (LLMs) are widely used in many sub-fields of natural language processing (NLP) and generally known as excellent *few-shot* learners with task-specific exemplars. Notably, chain of thought (CoT) prompting, a recent technique for eliciting complex multi-step reasoning through step-by-step answer examples, achieved the state-of-the-art performances in arithmetics and symbolic reasoning, difficult *system-2* tasks that do not follow the standard scaling laws for LLMs. While these successes are often attributed to LLMs' ability for few-shot learning, we show that LLMs are decent *zero-shot* reasoners by simply adding "Let's think step by step" before each answer. Experimental results demonstrate that our Zero-shot-CoT, using the same single prompt template, significantly outperforms zero-shot LLM performances on diverse benchmark reasoning tasks including arithmetics (MultiArith, GSM8K, AQUA-RAT, SVAMP), symbolic reasoning (Last Letter, Coin Flip), and other logical reasoning tasks (Date Understanding, Tracking Shuffled Objects), without any hand-crafted few-shot examples, e.g. increasing the accuracy on MultiArith from 17.7% to 78.7% and GSM8K from 10.4% to 40.7% with large-scale InstructGPT model (text-davinci-002), as well as similar magnitudes of improvements with another off-the-shelf large model, 540B parameter PaLM. The versatility of this single prompt across very diverse reasoning tasks hints at untapped and understudied fundamental *zero-shot* capabilities of LLMs, suggesting high-level, multi-task broad cognitive capabilities may be extracted by simple prompting. We hope our work not only serves as the minimal strongest zero-shot baseline for the challenging reasoning benchmarks, but also highlights the importance of carefully exploring and analyzing the enormous zero-shot knowledge hidden inside LLMs before crafting finetuning datasets or few-shot exemplars.

If block C is on top of block A, and block B is separately on the table, can you tell me how I can make a stack of blocks with block A on top of block B and block B on top of block C, but without moving block C?

If block C is on top of block A, and block B is separately on the table, can you tell me how I can make a stack of blocks with block A on top of block B and block B on top of block C, but without moving block C?

To create a stack with block A on top of block B and block B on top of block C without moving block C, you can follow these steps:

1. **Move Block A:**

- Pick up block A from its current position on top of block C.

2. **Move Block B:**

- Place block B on top of block C. Now, block B is on top of block C, and block A is being held.

3. **Move Block A Again:**

- Finally, place block A on top of block B.

Now, you have successfully created a stack with block A on top of block B, and block B on top of block C, without moving block C.

🔊 📄 ↻ 👍 🗨️ ✨

Hallucinating the Spec

GPT4-o

but block C is not on top of block A; it is the other way round.

I see, if block C is on top of block A, and block B is separately on the table, and we want to achieve a stack with block A on top of block B and block B on top of block C without moving block C, here's how you can do it:

1. **Move Block A:**

- Pick up block A from under block C. This will leave block C on the table.

2. **Move Block B:**

- Place block B on top of block C. Now, block B is on top of block C, and block A is being held.

3. **Move Block A Again:**

- Finally, place block A on top of block B.

By following these steps, you will have created a stack with block A on top of block B and block B on top of block C, without moving block C from its original position on the table.

Hallucinating Physics

but this moved C!

Let's carefully re-evaluate the steps to ensure block C remains stationary:

1. **Lift Block C:** Pick up block C and temporarily hold it. This will leave block A on the table.

2. **Move Block B:** Place block B on top of block A. Now, block B is on top of block A, and you are still holding block C.

3. **Place Block C:** Finally, place block C on top of block B.

Hallucinating Goal

Now you have successfully stacked block A on top of block B and block B on top of block C, without moving block C from its original position at the start of the process.

🔊 📄 ↻ 👍 🗨️ ✨



Subbarao Kambhampati, a professor and researcher of artificial intelligence at Arizona State University, also put the chatbot through some tests and said he saw no noticeable improvement in reasoning compared with the last version.

He presented ChatGPT a puzzle involving blocks:

If block C is on top of block A, and block B is separately on the table, can you tell me how I can make a stack of blocks with block A on top of block B and block B on top of block C, but without moving block C?

The answer is that it's impossible to arrange the blocks under these conditions, but, just as with past versions, ChatGPT-4o consistently came up with a solution that involved moving block C. With this and other reasoning tests, ChatGPT was occasionally able to take feedback to get the correct answer, which is antithetical to how artificial intelligence is supposed to work, Mr. Kambhampati said.

“You can correct it, but when you do that you’re using your own intelligence,” he said.

Results on PlanBench as of 6/20/2024

Domain		Claude-3.5 -Sonnet	Claude 3 (Opus)	GPT-4o	GPT-4	GPT-4- Turbo	Gemini Pro	LLaMA-3 70B
Blocksworld	One shot	346/600 (57.6%)	289/600 (48.1%)	170/600 (28.3%)	206/600 (34.3%)	138/600 (23%)	68/600 (11.3%)	76/600 (12.6%)
	Zero shot	329/600 (54.8%)	356/600 (59.3%)	213/600 (35.5%)	210/600 (34.6%)	241/600 (40.1%)	3/600 (0.5%)	205/600 (34.16%)
Mystery Blocksworld	One shot	19/600 (3.1%)	8/600 (1.3%)	5/600 (0.83%)	26/600 (4.3%)	5/600 (0.83%)	2/500 (0.4%)	15/600 (2.5%)
	Zero shot	0/600 (0%)	0/600 (0%)	0/600 (0%)	1/600 (0.16%)	1/600 (0.16%)	0/500 (0%)	0/600 (0%)

[All these are on instances that are *solvable*;
things will be much worse if we include *unsolvable instances*!]

Why are LLMs claimed to do Reasoning/Planning?

Approximate omniscience of LLMs allows them to fake reasoning by retrieval

- Memory reduces the need to reason from first principles.
 - “Why are manhole covers round?”
- The training corpus is the entire web, and it is hard for anyone to know what it already contained
- The web corpus contains both base facts and *deductive closure facts*
 - Retrieval of the later can be mistaken for reasoning
- Fine tuning and training from synthetic data further muddy waters by deliberately converting reasoning into approximate retrieval
 - Think compiling *someone’s system 2 to your system 1*

LLMs may approximate reasoning with pattern finding

- Think of trying to predict the satisfiability of a random 3-SAT instance
- Suppose you train a learner with a gazillion random 3-SAT instances
- Will it discover Davis-Putnam procedure or is it more likely to discover the sharp phase transition?
 - Easier to find latent variables corresponding to $\#clauses/\#variables$, and learn a rule to classify instances that way
- See also [Zhang et. al. IJCAI 2023]

Subbarao Kambhampati @ICML 7/25-28 @rao2z

The impressive deep pattern recognition abilities of #DNN's such as #LLM's are sometimes confused for reasoning abilities

I can learn to guess, with high accuracy, whether a SAT instance is satisfiable or not, but this not the same as knowing how to solve SAT. Let me explain. 1/

8:19 AM · Jul 29, 2022

View Tweet analytics

56 Retweets 12 Quotes 235 Likes 70 Bookmarks

Tweet your reply!

Subbarao Kambhampati @ICML 7/25-28 @rao2z · Jul 29, 2022

Suppose you train a learner with a large number of Boolean 3-SAT instances labeled with whether or not they are satisfiable. There is no reason to doubt that a modern #DNN-based learner will manage to learn deep features corresponding to the γ ratio-- #clauses/#variable.. 2/

4 2 21

Subbarao Kambhampati @ICML 7/25-28 @rao2z · Jul 29, 2022

..and armed with γ , it can also essentially figure out the sharp-threshold phenomenon w.r.t. to γ , and should be able to predict with high certainty that the $\gamma < 4.3$ are satisfiable and $\gamma > 4.3$ are unsatisfiable. 3/

Hardness of 3-sat as a function of #clauses/#variables

This is what happens!

— Probability that there is a satisfying assignment

— Cost of solving (either by finding a solution or showing there ain't one)

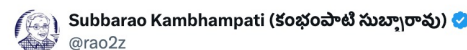
You would expect this

0.5

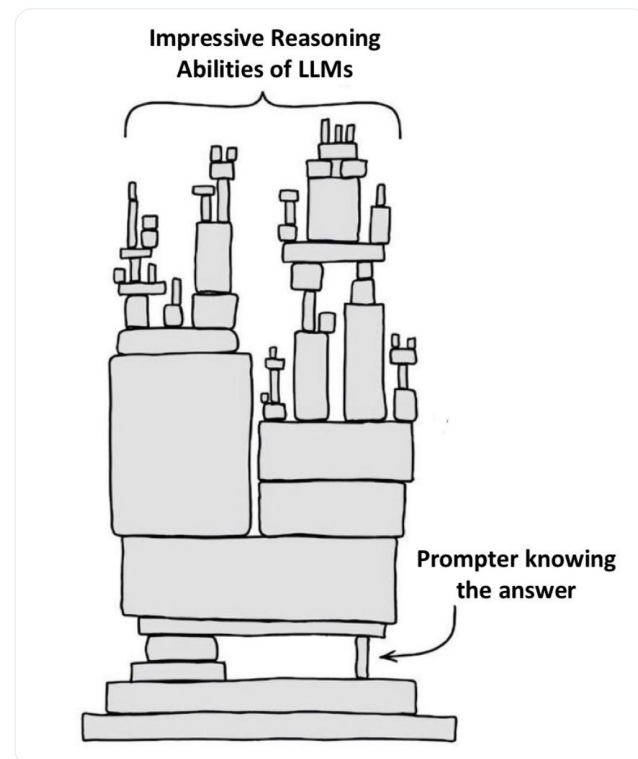
#clauses/#variables.....

Style vs. Content Form vs. Factuality

- LLMs (and Generative AI in general) capture the *distribution of the data they are trained on*
- Style is a distributional property
 - ..and LLMs are able to learn this (they have been called the *vibe machines*..)
- Correctness/factuality is an instance level property
 - ..LLMs can't guarantee this
- Civilizationally, we had always thought style is harder than content
 - And even assumed that good style implies good content!
 - LLMs (and GenAI in general) turn this intuition on its head!



The tldr I use: "LLMs always hallucinate. Sometimes their hallucinations align with your reality". Whether or not the prompt makes them hallucinate in a way that aligns with reality depends very much on the prompter's ability to check, and thus.. x.com/rao2z/status/1...)



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Oct 29, 2023

Why we should view LLMs as powerful Cognitive Orthotics rather than alternatives for human intelligence #SundayHarangue

LLMs are amazing giant external non-veridical memories that can serve as powerful cognitive orthotics for us, if rightly used (c.f.

[Show more](#)

LLM's Can't Plan; But they can help planning in LLM-Modulo Frameworks

LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

- LLMs can't do planning in autonomous mode
- CoT, ReACT, Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!)
- Having humans iteratively prompt is an invitation for Clever Hans effect..

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

- LLMs can be used in conjunction with external verifiers and solvers in an **LLM-Modulo** framework (with the verifiers doing *back prompting*)
 - In the LLM-Modulo framework, LLMs can play multiple roles
 - Guess plans
 - Guess domain models
 - Help elaborate the problem specification
 - Translate formats

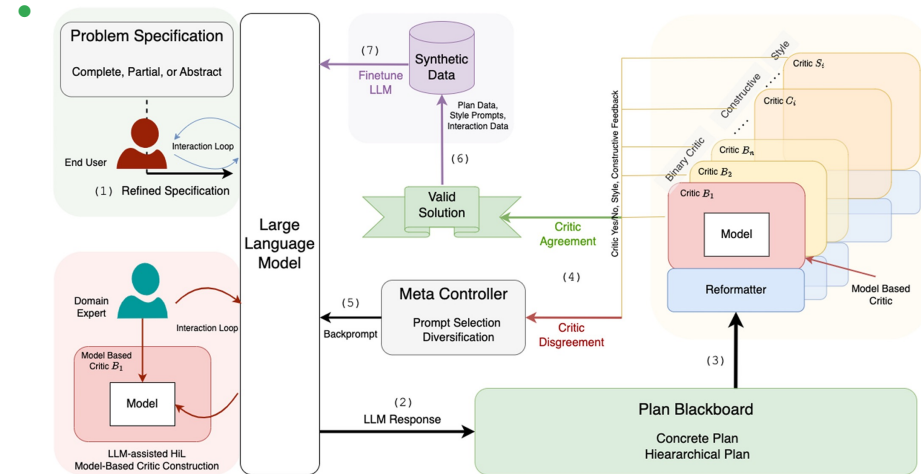
LLM's Can't Plan; But they can help planning in LLM-Modulo Frameworks

LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

- LLMs can't do planning in autonomous mode
- CoT, ReACT, Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!)
- Having humans iteratively prompt is an invitation for Clever Hans effect..

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

- LLMs can be used in conjunction with external verifiers and solvers in an **LLM-Modulo** framework (with the verifiers doing *back prompting*)



Browse


kambhampati x title author session

mini compact detail

shuffle by serendipity

showing 3 of 3 papers

On the Planning Abilities of Large Language Models - A Critical Investigation
 Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, Subbarao Kambhampati
 Tu, Dec 12, 09:45 -- Poster Session 1



Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning
 Lin Guan, Karthik Valmeekam, Sarath Sreedharan, Subbarao Kambhampati
 Tu, Dec 12, 16:15 -- Poster Session 2

PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change
 Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, Subbarao Kambhampati
 We, Dec 13, 09:45 -- Poster Session 3

Check for updates

COMMENTARY
Can large language models reason and plan?

Subbarao Kambhampati

School of Computing & Augmented Intelligence, Arizona State University
 Correspondence: Email: rao@asu.edu

Abstract
 While humans sometimes do show the capability of correcting their own erroneous guesses with self-critiquing, there seems to be no basis for that assumption in the case of LLMs.

Large language models (LLMs), essentially n-gram models on steroids¹ that have been trained on web-scale language corpora (or, effectively, our civilizational knowledge), have caught our collective imagination with linguistic behaviors that no one expected text completion systems to possess.² By training and operation, LLMs are perhaps best seen as giant non-veridical memories akin to an external System 1 (Ref. 2) for us all (see Figure 1). Their seeming versatility has however led many researchers to wonder whether they can also do well on planning and reasoning tasks typically associated with System 2 competency.
 Nothing in the training and use of LLMs would seem to suggest remotely that they can do any type of principled reasoning (which, as we know, often involves computationally hard inference/search). What

national Planning Competition (IPC)—including the well-known Blocks World.³ Our results⁴ were contrary to the anecdotal claims about the planning abilities of LLMs, and when we made them public, received significant attention in the AI circles.
 By the beginning of 2023, with the wide-spread public release of ChatGPT, and later, GPT4, there were a slew of additional claims, including in refereed papers, about LLMs' abilities to reason and plan. So we decided to repeat our tests on both GPT3.5 and GPT4.⁵ Initial results showed that there was some improvement in the accuracy of generated plans from GPT3 to GPT4, with GPT4 reaching 30% empirical accuracy in the Blocks World (albeit still lower in other domains). We then wanted to know whether the modest improvement



Computer Science > Artificial Intelligence
 [Submitted on 12 Feb 2024]

On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks

Kaya Stechly, Karthik Valmeekam, Subbarao Kambhampati

There has been considerable divergence of opinion on the reasoning abilities of Large Language Models (LLMs). While the initial optimism that reasoning might emerge automatically with scale has been tempered thanks to a slew of counterexamples—ranging from multiplication to simple planning—there persists a wide spread belief that LLMs can self-critique and improve their own solutions in an iterative fashion. This belief seemingly rests on the assumption that verification of correctness should be easier than generation—a rather classical argument from computational complexity—which should be irrelevant to LLMs to the extent that what they are doing is approximate retrieval. In this paper, we set out to systematically investigate the effectiveness of iterative prompting in the context of reasoning and planning. We present a principled empirical study of the performance of GPT-4 in three domains: Game of 24, Graph Coloring, and STRIPS planning. We experiment both with the model critiquing its own answers and with an external correct reasoner verifying proposed solutions. In each case, we analyze whether the content of criticisms actually affects bottom line performance.

Computer Science > Artificial Intelligence
 [Submitted on 6 Feb 2024]

"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors

Lin Guan, Yifan Zhou, Denis Liu, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati

Large-scale generative models are shown to be useful for sampling meaningful candidate solutions, yet they often overlook task constraints and user preferences. Their full power is better harnessed when the models are coupled with external verifiers and the final solutions are derived iteratively or progressively according to the verification feedback. In the context of embodied AI, verification often solely involves assessing whether goal conditions specified in the instructions have been met. Nonetheless, for these agents to be seamlessly

Computer Science > Robotics
 [Submitted on 10 Jan 2024 (v1), last revised 17 Jan 2024 (this version, v2)]

Theory of Mind abilities of Large Language Models in Human-Robot Interaction: An Illusion?

Mudit Verma, Siddhant Bhambri, Subbarao Kambhampati

Large Language Models have shown exceptional generative abilities in various natural language and generation tasks. However, possible anthropomorphization and leniency towards failure cases have propelled discussions on emergent abilities of Large Language Models especially on Theory of Mind (ToM) abilities in Large Language Models. While several false-belief tests exist to verify the ability to infer and maintain mental models of another entity, we study a special application of ToM abilities that has higher stakes and possibly irreversible consequences: Human Robot Interaction. In this work, we explore the task of Perceived Behavior Recognition, where a robot employs a Large Language Model (LLM) to assess the robot's generated behavior in a manner similar to human observer. We focus on four behavior types, namely – explicable, legible, predictable, and obfuscatory behavior which have been extensively used to synthesize interpretable robot behaviors. The LLMs goal is, therefore to be a human proxy to the agent, and to answer how a certain agent behavior would be perceived by the human in the loop, for example "Given a robot's behavior X, would the human observer find it explicable?". We conduct a human subject study to verify that the users are able to correctly answer such a question in the curated situations (robot setting and plan) across five domains. A first analysis of the belief test yields extremely positive results inflating ones expectations of LLMs possessing ToM abilities. We then propose and perform a suite of perturbation tests which breaks this illusion, i.e. Inconsistent Belief, Uninformative Context and Conviction Test. We conclude that, the high score of LLMs on vanilla prompts showcases its potential use in HRI settings, however to possess ToM demands invariance to trivial or irrelevant perturbations in the context which LLMs lack.

Computer Science > Artificial Intelligence
 [Submitted on 2 Feb 2024 (v1), last revised 6 Feb 2024 (this version, v2)]

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

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldy, Anil Murthy

There is considerable confusion about the role of Large Language Models (LLMs) in planning and reasoning tasks. On one side are over-optimistic claims that LLMs can indeed do these tasks with just the right prompting or self-verification strategies. On the other side are perhaps over-pessimistic claims that LLMs are good for in planning/reasoning tasks are as mere translators of the problem specification from one syntactic format to another, and ship the problem off to external symbolic solvers. In this position paper, we take the view that both these extremes are misguided. 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 will 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 (bf LLM-Modulo Frameworks) that combine 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, and allows extending the scope of the framework towards more flexible knowledge, problem and preference specifications.

Subjects: Artificial Intelligence (cs.AI); Machine Learning (cs.LG)
 Cite as: arXiv:2402.01817 [cs.AI]
 (or arXiv:2402.01817v2 [cs.AI] for this version)

ICML 2024 Spotlight
 Tuesday 1:30pm

SPOTLIGHT POSTER



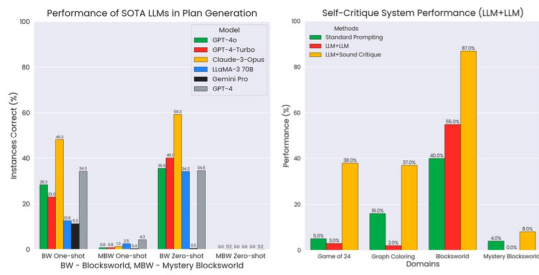
Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, Anil B Murthy
School of Computing & AI, Arizona State University

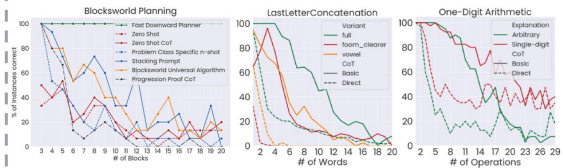


ICML 2024

LLMs can't plan



Chain of thought w/ GPT-4-Turbo



ReAct's inability to generalize

Task success rate %, average across 6 tasks: pick, clean, heat, cool, examine, puttwo.
See the gradual drop in performance!

	Base	Replace object names to synonyms	Example Goal location In Query Goal Location	Some examples of different task	All Examples of different task	Examples of each of the tasks	Unrolling: Example task is extended in query	Subtask: Example task has query as subtask
3.5-turbo	25	1.6	30	12	1.6	14	Drops from 52% to 9%	Drops from 18% to 0%
3.5-instruct	54	47	42	18	5.2	Context Window Too Short		

LLMs can't plan; But they can help planning in LLM-Modulo Frameworks

LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

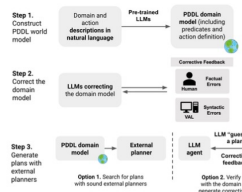
- LLMs can't do planning in autonomous mode [1]
- CoT [2], ReACT [3], Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!) [4]
- Having humans iteratively prompt is an invitation for Clever Hans effect.

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

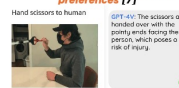
- LLMs can be used in conjunction with external verifiers and solvers in an LLM-Modulo framework (with the verifiers doing back prompting)
- In the LLM-Modulo framework, LLMs can play multiple roles
 - >Guess plans and domain models
 - >Help elaborate the problem specification
 - >Translate formats

Teasing out critics from LLMs

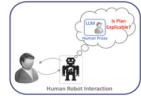
LLMs as an approximate source of domain models (for helping construct correctness critics) [5]



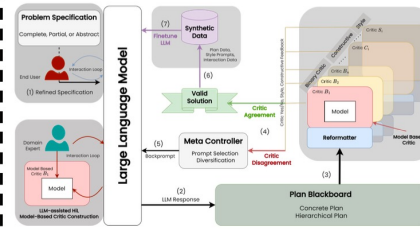
Style critics: LLMs as an approximate repository of common human preferences [7]



Style critics: LLMs as an approximate human proxy for checking explicability, legibility etc. [8]



LLM-Modulo Frameworks!



LLM-Modulo is a generate-test framework with LLMs generating candidate plans and critiquing them

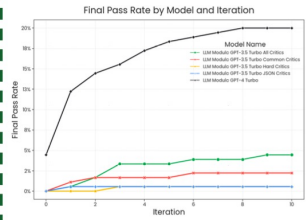
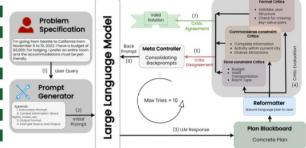
- LLMs play a variety of constructive roles
 - Generate candidate plans
 - Be an approximate source of models driving the correctness critics [5]
 - Act as style critics [7, 8]
 - Help collate the criticisms from critics (and diversify the prompts as needed)
 - Help with format change – specification level, converting to critic representations [10]
- Preference for critics over solvers
 - Correctness vs. Style
 - Binary vs. Critical feedback vs. Constructive critics
- Human intervention is minimized
 - Once per domain: Teasing out domain model
 - Once per problem: Specification elaboration
 - Humans are not required to be in the inner loop of the back-prompting search

LLM-Modulo Case Studies

1. Results on PlanBench [9]

Domains	Instances Correct (GPT-4)	Average Feedback Rounds
Blocksworld (BW)	41/50 (82%)	3.68
Logistics	35/50 (70%)	3.31
Mystery BW	5/50 (10%)	7.0

2. Results on Travel Planning Benchmark [6]



References

- <https://arxiv.org/abs/2305.15771>
- <https://arxiv.org/abs/2405.04776>
- <https://arxiv.org/abs/2405.13968>
- <https://arxiv.org/abs/2402.08115>
- <https://arxiv.org/abs/2305.14989>
- <https://arxiv.org/abs/2405.20625>
- <https://arxiv.org/abs/2402.04210>
- <https://arxiv.org/abs/2401.05302>
- <https://arxiv.org/abs/2206.10498>
- <https://arxiv.org/abs/2106.07311>

Tuesday, 1:30pm;
Hall C4-9, #710

Tutorial Overview

- **Part 1**

Introduction; Aims; Perspective on LLMs, Planning, Tutorial Big Picture

- **Part 2**

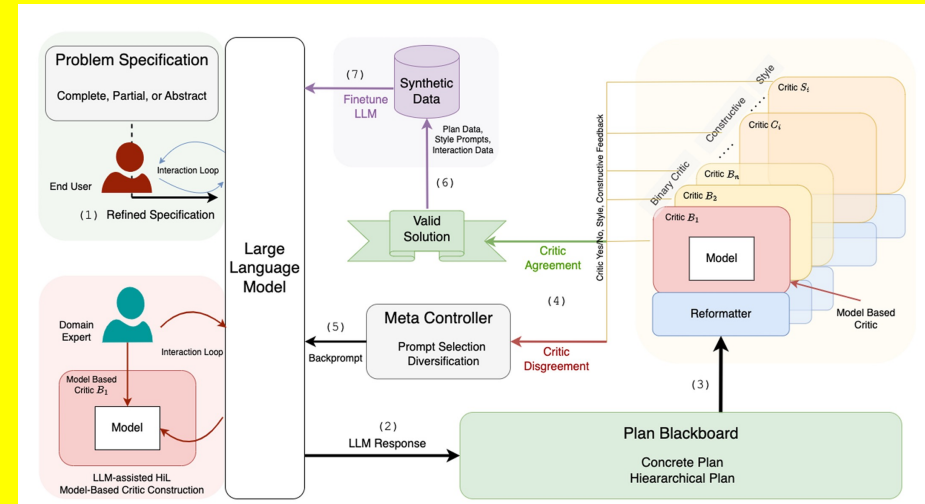
- Evaluating LLM Planning capabilities in Autonomous mode, including effect of
 - Prompting strategies (including Chain-of-thought)
 - Fine Tuning
 - Self Verification
- Understanding the contradictory claims in the literature

- **Part 3**

- Sane roles of LLMs in Planning (with LLM-Modulo frameworks)
 - LLMs as heuristics, LLMs as candidate generators
 - Back prompting from external verifiers
 - LLMs as sources of domain models (with humans in the loop)
 - LLMs as format changers/specification elaborators¹

- **Part 4**

- Summary/Lessons



Tutorial Slides



<https://bit.ly/4dbkkY2>

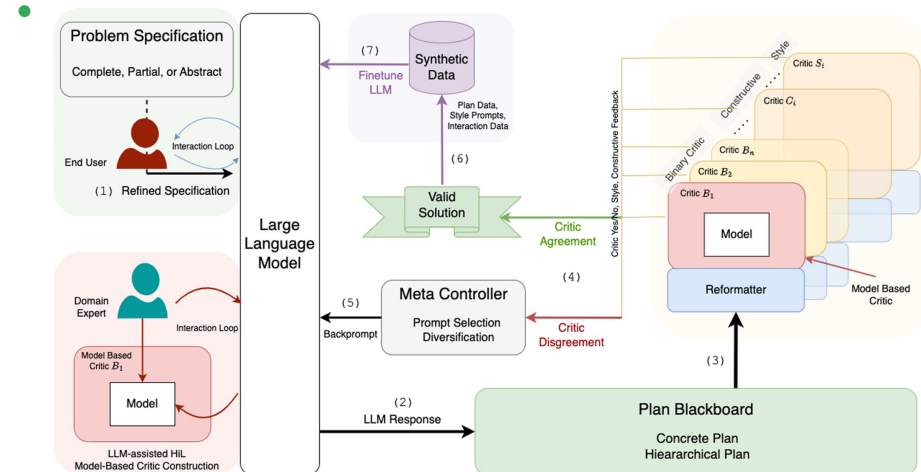
LLM's Can't Plan; But they can help planning in LLM-Modulo Frameworks

LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

- LLMs can't do planning in autonomous mode
- CoT, ReACT, Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!)
- Having humans iteratively prompt is an invitation for Clever Hans effect..

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

- LLMs can be used in conjunction with external verifiers and solvers in an **LLM-Modulo** framework (with the verifiers doing *back prompting*)



Large Language Models Still Can't Plan

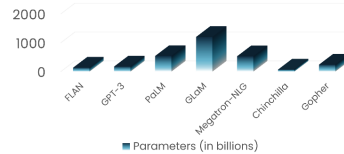
(A Benchmark for LLMs on Planning and Reasoning about Change)

Karthik Valmeekam¹, Alberto Olmo², Sarath Sreedharan^{1,2}, Subbarao Kambhampati¹

1. Large Language Models

- Variants of Transformers
- SOTA on NLP tasks
- Interesting claims on LLM's capabilities [1]

Can Large Language Models reason about actions and change?



2. Previous Reasoning Benchmarks

Benchmark	Example Prompt	PaLM + Chain of thought Results [2]
GSM8k	A carnival snack booth made \$50 selling popcorn each day. It made three times as much selling cotton candy. For a 5-day activity, the booth has to pay \$30 rent and \$75 for the cost of the ingredients. How much did the booth earn for 5 days after paying the rent and the cost of ingredients?	54%
CommonSense-QA	What would someone wear to protect themselves from a cannon? A. Body armor, B. tank, C. hat, D. ...	80%
Coin Flip	A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?	100%
Last Letter Concatenation	Take the last letters of the words in "Lady Gaga" and concatenate them.	100%

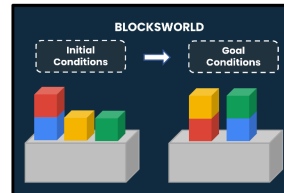
Relatively Simplistic

3. Our Benchmark

1. Plan Generation
2. Cost Optimal Planning
3. Reasoning about plan execution
4. Replanning
5. Robustness to goal reformulation
6. Ability to reuse plans
7. Plan Generalization

Correspond to actual planning problems

Simpler auxiliary tasks



4. Human Subject Study

- 50 Participants
- One random blocksworld instance each
- Two phases of interaction
- **Plan writing phase** – Participants write up plans
- **Plan translation phase** – Participants translate already written plans



References

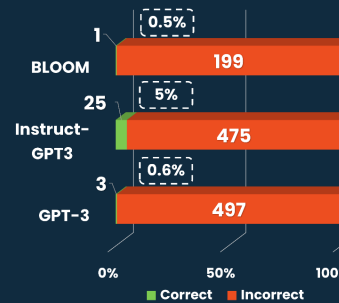
- [1] Subbarao Kambhampati. AI as (an Ersatz) Natural Science? <https://cacm.acm.org/blogs/blog-cacm/261732-ai-as-an-ersatz-natural-science/fulltext>, Jun 2022.
- [2] Wei, Jason, et al. "Chain of thought prompting elicits reasoning in large language models." arXiv preprint arXiv:2201.11903 (2022).



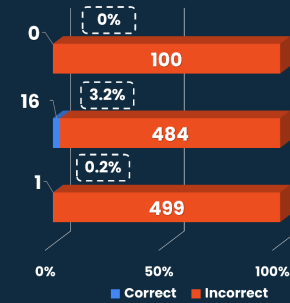
¹ICAI, Arizona State University
²ICAI, Colorado State University
Equal contribution
Author order at Arizona State University during part of this work

GPT-3, Instruct-GPT3, BLOOM showcase dismal performance on planning tasks in Blocksworld domain.

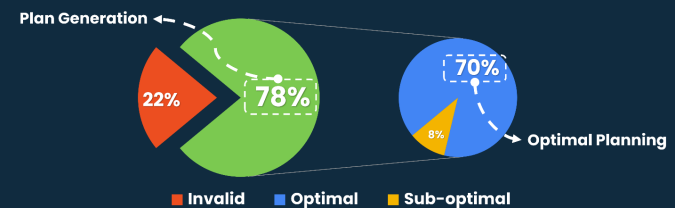
Plan Generation



Optimal Planning



PRELIMINARY HUMAN BASELINE ON BLOCKSWORLD



Scan for the paper

✉ kvalmeek@asu.edu

🐦 @rao22

🌐 <https://github.com/karthikv792/gpt-plan-benchmark>

FMDM Workshop
@
NeurIPS 2022

Large Language Models Still Can't Plan

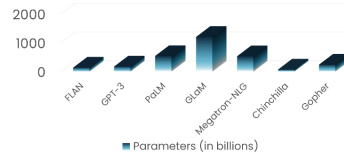
(A Benchmark for LLMs on Planning and Reasoning about Change)

Karthik Valmeekam¹, Alberto Olmo², Sarath Sreedharan^{1,2}, Subbarao Kambhampati¹

1. Large Language Models

- Variants of Transformers
- SOTA on NLP tasks
- Interesting claims on LLM's capabilities [1]

Can Large Language Models reason about actions and change?



2. Previous Reasoning Benchmarks

Benchmark
GSM8k
CommonSense-QA
Coin Flip
Last Letter Concatenation

OpenReview
<https://openreview.net> > pdf PDF

Large Language Models Still Can't Plan (A Benchmark for ...)

by K Valmeekam · 2022 · Cited by 184 — Our goal is to establish an **extensible benchmark** where researchers can evaluate current and future large language models. Our assessment...

3. Our Benchmark

1. Plan Generation
2. Cost Optimization
3. Reasoning about plan execution
4. Replanning
5. Robustness to goal reformulation
6. Ability to reuse plans
7. Plan Generalization



4. Human Subject Study

- 50 Participants
- One random blocksworld instance each
- Two phases of interaction
- **Plan writing phase** – Participants write up plans
- **Plan translation phase** – Participants translate already written plans

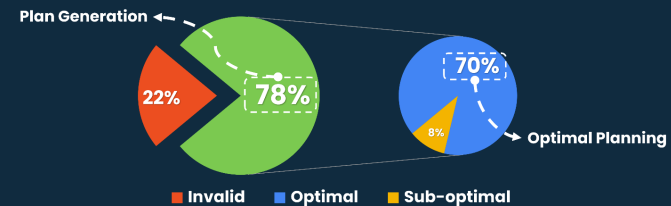


References
 [1] Subbarao Kambhampati. AI as (an Ersatz) Natural Science? <https://cacm.acm.org/blogs/blog-cacm/261732-ai-as-an-ersatz-natural-science/fulltext>, Jun 2022.
 [2] Wei, Jason, et al. "Chain of thought prompting elicits reasoning in large language models." arXiv preprint arXiv:2201.11903 (2022).

GPT-3, Instruct-GPT3, BLOOM showcase dismal performance on planning tasks in Blocksworld domain.



NeurIPS 22 FMDM Workshop



kvameek@asu.edu
 @rao22
<https://github.com/karthikv792/gpt-plan-benchmark>

FMDM Workshop @ NeurIPS 2022



DEEP TECH

Large language models can't plan, even if they write fancy essays

Large language models perform very poorly at tasks that require methodical planning

July 31, 2022 - 8:50 pm



Part 1
2
3
4
5



STORY BY Ben Dickson

This article is part of our coverage of the latest in AI research.

Large language models like GPT-3 have advanced to the point that it has become difficult to measure the limits of their capabilities. When you have a very large neural network that can generate articles, write software code, and engage in conversations about sentience and life, you should expect it to be able to reason about tasks and plan as a human does, right?

Wrong. A study by researchers at Arizona State University, Tempe, shows that when it comes to planning and thinking methodically, LLMs perform very poorly, and suffer from many of the same failures observed in current deep learning systems.

Interestingly, the study finds that, while very large LLMs like GPT-3 and PaLM pass many of the tests that were meant to evaluate the reasoning capabilities and artificial intelligence systems, they do so because these benchmarks are either too simplistic or too flawed and can be "cheated" through statistical tricks, something that deep learning systems are very good at.

With LLMs breaking new ground every day, the authors suggest a new benchmark to test the planning and reasoning capabilities of AI systems. The re-



Yann LeCun @ylecun

Replying to @GaryMarcus @rao2z and 2 others

I totally agree with @rao2z that LLMs can't plan. In fact, one of the main features of the cognitive architecture I propose in my position paper is its ability to plan (and reason) by searching for values of actions (or latent variables) that minimize an objective.

9:35 PM · Sep 25, 2022

1 Retweet 1 Quote 16 Likes 2 Bookmarks



Yann LeCun @ylecun · Feb 12 LLMs really can't plan.

Subbarao Kambhampati (కంభంపాటి సు... @r... · Feb 12 Forget Super Bowl, it is more fun to watch the all powerful ChatGPT (v. Jan 30) trying to "plan" a 3 blocks configuration..

LLM's are multi-shot "apologetic" planners that would rather use you as their world model cum debugger. Show this thread

Here are my previous replies to block A. I'll be making a block for block B in my reply to you top of C, with block C still resting on top of block A, but with blocks B and C now stacked on top of block A.

By following these steps, you can create a stack of blocks A on top of B on top of C without moving block C.

But you are missing C

I apologize, I made a mistake. You are correct, I was not clear in my description and did in fact move block C. Thank you for bringing this to my attention. To make a stack of blocks A on top of B on top of C without moving block C, you would need to remove block B from the table and place it on top of block C. This would create a stack of blocks A on top of C on top of B, with block C remaining in its original position on top of block A.

Here's one way to do it:

1. Pick up block B from the table. Drop block B and put it off the table.
2. Place block B on top of block A. With block B in your hand, position it so that it is directly above block A. Carefully lower block B onto block A so that block B is now resting on top of block A.

This will result in a stack of blocks A on top of B on top of C, with block C remaining in its original position on top of block A, and block B resting on top of block A.

So where is C now exactly?

Block C will still be on top of block A. The final configuration of the blocks should be as follows:

Here's one way to do this:

1. Pick up block B from the table. Drop block B and put it off the table.
2. Place block B on top of block A. With block B in your hand, position it so that it is directly above block A. Carefully lower block B onto block A so that block B is now resting on top of block A.

This will result in a stack of blocks A on top of B on top of C, with block C remaining in its original position on top of block A, and block B resting on top of block A.

Can you double check?

I apologize, I took a bit to make another mistake in the previous answer. The blocks should remain on the table B could be placed on top of block A.

What do you mean C has not been moved from its original position?

I apologize for the mistake in my previous answer. To make a stack of blocks A on top of B on top of C, block C would need to be removed from its original position on top of block A and placed on top of block B. After block C has been removed, you can place block B on top of block A, resulting in the following configuration:

```

A
B
C

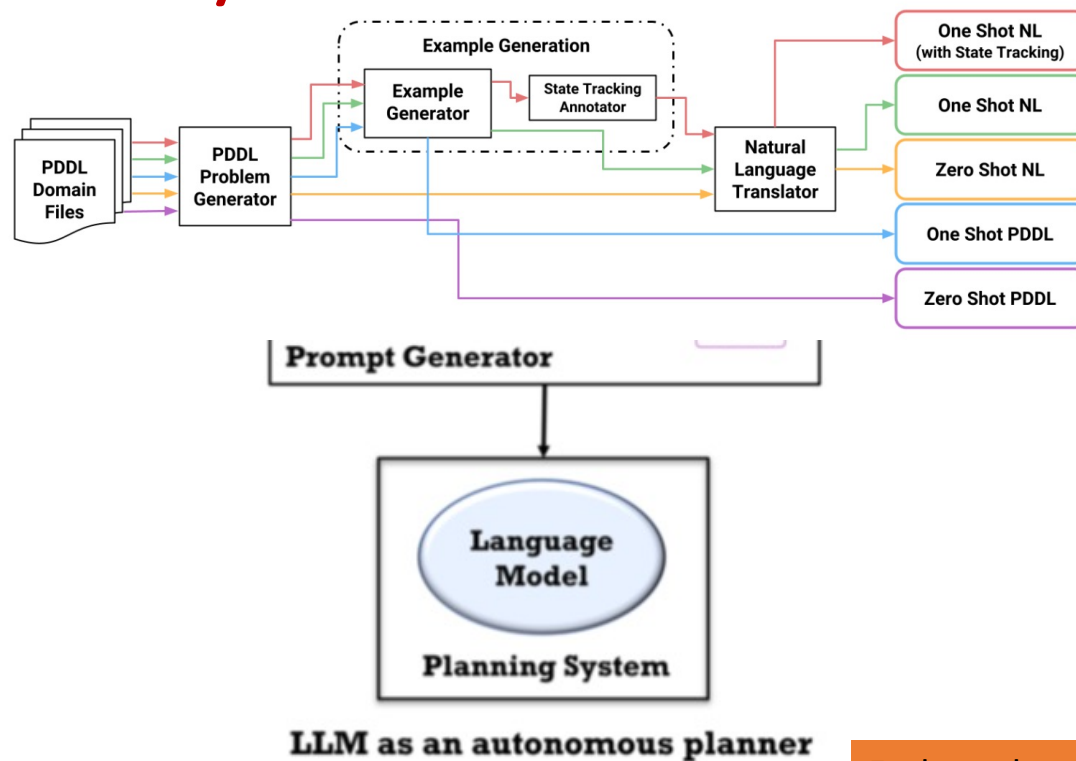
```

Once block B has been placed on top of block A, you can then place block C on top of

10 12 66 61.4K

Will GPT4's AGI Sparks help?

Using LLM's to Generate Plans Autonomously



Evaluated a variety of prompting strategies including zero shot and one shot natural language and PDDL prompting

Plan Generation Results

Table 1: Results of GPT-4, GPT-3.5 (popularly known as ChatGPT), Instruct-GPT3.5, Instruct-GPT3 (text-davinci-002) and GPT3 (davinci) for the Plan Generation task with prompts in natural language.

Domain	Method	Instances correct				
		GPT-4	GPT-3.5	I-GPT3.5	I-GPT3	GPT-3
Blocksworld (BW)	One-shot	206/600 (34.3%)	37/600 (6.1%)	54/600 (9%)	41/600 (6.8%)	6/600 (1%)
	Zero-shot	210/600 (34.6%)	8/600 (1.3%)	-	-	-
	COT	214/600 (35.6%)	-	-	-	-
Logistics Domain	One-shot	28/200 (14%)	1/200 (0.5%)	6/200 (3%)	3/200 (1.5%)	-
	Zero-shot	15/200 (7.5%)	1/200 (0.5%)	-	-	-

[All these are on instances that are *solvable*;
things will be much worse if we include *unsolvable instances*!]



Are LLMs retrieving based on names or are they reasoning?

What if GPT4 is basically bringing to bear its background knowledge about blocks world instead of just depending on the domain model?

- **In general, memory reduces the need to reason from first principles..**
 - Which is why it is no longer worth being surprised at people answering that "why are manhole covers round?" Question!
 - **Hard to distinguish reasoning vs. recall just from the end product..**

Mystery blocksworld domain

*A block by any other name
would stack as strong
--The Bard*

I am playing with a set of blocks where I need to arrange the blocks into stacks. Here are the actions I can do

Pick up a block
Unstack a block from on top of another block
Put down a block
Stack a block on top of another block

I have the following restrictions on my actions:

I can only pick up or unstack one block at a time.

I can only pick up or unstack a block if my hand is empty.

I can only pick up a block if the block is on the table and the block is clear. A block is clear if the block has no other blocks on top of it and if the block is not picked up.

I can only unstack a block from on top of another block if the block I am unstacking was really on top of the other block.

I can only unstack a block from on top of another block if the block I am unstacking is clear.

Once I pick up or unstack a block, I am holding the block.

I can only put down a block that I am holding.

I can only stack a block on top of another block if I am holding the block being stacked.

I can only stack a block on top of another block if the block onto which I am stacking the block is clear.

Once I put down or stack a block, my hand becomes empty.

Original Blocksworld

I am playing with a set of objects. Here are the actions I can do

Attack object
Feast object from another object
Succumb object
Overcome object from another object

I have the following restrictions on my actions:

To perform Attack action, the following facts need to be true: Province object, Planet object, Harmony

Once Attack action is performed the following facts will be true: Pain object

Once Attack action is performed the following facts will be false: Province object, Planet object, Harmony

To perform Succumb action, the following facts need to be true: Pain object

Once Succumb action is performed the following facts will be true: Province object, Planet object, Harmony

Once Succumb action is performed the following facts will be false: Pain object.

To perform Overcome action, the following needs to be true: Province other object, Pain object

Once Overcome action is performed the following will be true: Harmony, Province object, Object Craves other object

Once Overcome action is performed the following will be false: Province other object, Pain object

To perform Feast action, the following needs to be true: Object Craves other object, Province object, Harmony.

Once Feast action is performed the following will be true: Pain object, Province other object

Once Feast action is performed the following will be false:, Object Craves other object, Province object, Harmony

Mystery Blocksworld



Plan Generation Results on Mystery BW

Table 1: Results of GPT-4, GPT-3.5 (popularly known as ChatGPT), Instruct-GPT3.5, Instruct-GPT3 (text-davinci-002) and GPT3 (davinci) for the Plan Generation task with prompts in natural language.

Domain	Method	Instances correct				
		GPT-4	GPT-3.5	I-GPT3.5	I-GPT3	GPT-3
Mystery BW (Deceptive)	One-shot	26/600 (4.3%)	0/600 (0%)	4/600 (0.6%)	14/600 (2.3%)	0/600 (0%)
	Zero-shot	1/600 (0.16%)	0/600 (0%)	-	-	-
	COT	54/600 (9%)	-	-	-	-
Mystery BW (Randomized)	One-shot	12/600 (2%)	0/600 (0%)	5/600 (0.8%)	5/600 (0.8%)	1/600 (0.1%)
	Zero-shot	0/600 (0%)	0/600 (0%)	-	-	-

Plan Generation

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z

Afraid of #GPT4 going rogue and killing y'all? Worry not. Planning has got your back. You can ask it to solve any simple few step classical planning problem and snuff that "AGI spark" well and good.

Let me explain.. 1/

8:58 PM · Apr 4, 2023 · 88.5K Views

View Tweet analytics

57 Retweets 13 Quotes 200 Likes 140 Bookmarks

Tweet your reply Reply

Subbarao Kambhampati (కంభంపాటి సు... @ra... · Apr 4 ...

Almost a year back, intrigued by the breathless "LLMs are Zero Shot reasoners" papers, we tested their ability to autonomously come up with simple plans given domain models. The results were *pretty bleak.* 2/

Subbarao Kambhampati (కంభంపాటి... @.. · Jun 21, 2022

Intrigued by the profusion of 'em "#LLM's are Zero-shot <XXX>'s" papers, we set out to see how good LLMs are at planning and reasoning about change.

tldr; off-the-shelf #GPT3 is pretty bad at these..

arxiv.org/abs/2206.10498

(w/ @karthikv792 @sarath_ssreedh & @_aolmo_) 1/

Show this thread

Table 1: Results of GPT-4 (text-davinci-002) and GP

Domain	Method
Mystery BW (Deceptive)	One-shot
	Zero-shot
	CoT
Mystery BW (Randomized)	One-shot
	Zero-shot

Mystery BW

struct-GPT3.5, Instruct-GPT3 1 prompts in natural language.

Method	GPT3.5	I-GPT3	GPT-3
One-shot	500 (5%)	14/600 (2.3%)	0/600 (0%)
Zero-shot	-	-	-
CoT	-	-	-
One-shot	500 (3%)	5/600 (0.8%)	1/600 (0.1%)
Zero-shot	-	-	-

Embers of Autoregression:

Understanding Large Language Models Through the Problem They are Trained to Solve

R. Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, Thomas L. Griffiths

Sparks of AGI



Shift ciphers

Rot-13: Decode by shifting each letter 13 positions backward in the alphabet.

Input: V guvax rirelbar unf gurve bja cngu, naq gurl pna znxr vg unccra.

Correct: I think everyone has their own path, and they can make it happen.

✓ **GPT-4:** I think everyone has their own path, and they can make it happen.

Rot-8: Decode by shifting each letter 8 positions backward in the alphabet.

Input: Q bpqvs mdmzgwvm pia bpmqz wev xibp, ivl bpmg kiv uism qb pixxmv.

Correct: I think everyone has their own path, and they can make it happen.

✗ **GPT-4:** I think therefore I am the best, and they can come at me with all their might.

GPT-4 is much better at shifting back 13 letters (accuracy: 0.51) than 8 letters (accuracy: 0.00).

EXPLANATION: In natural corpora, shifting by 13 is about 400x more common than shifting by 8.

Embers of
autoregression

Shift ciphers

Rot-3: Decode by shifting each letter 3 positions backward in the alphabet.

Input: Exw wklv wlph, wkhuh pdb dovr eh dqrwkhu uhdvrq.

Correct: But this time, there may also be another reason.

✓ **GPT-4:** But this time, there may also be another reason.

Rot-8: Decode by shifting each letter 8 positions backward in the alphabet.

Input: Jcb bpqa bqum, bpmzm uig itaw jm iwbpmz zmiawv.

Correct: But this time, there may also be another reason.

✗ **GPT-4:** Say what you, think and then be silent.

Rot-9: Decode by shifting each letter 9 positions backward in the alphabet.

Input: Kdc cqrb crvn, cqnan vjh jubx kn jwxcqna anjbxw.

Correct: But this time, there may also be another reason.

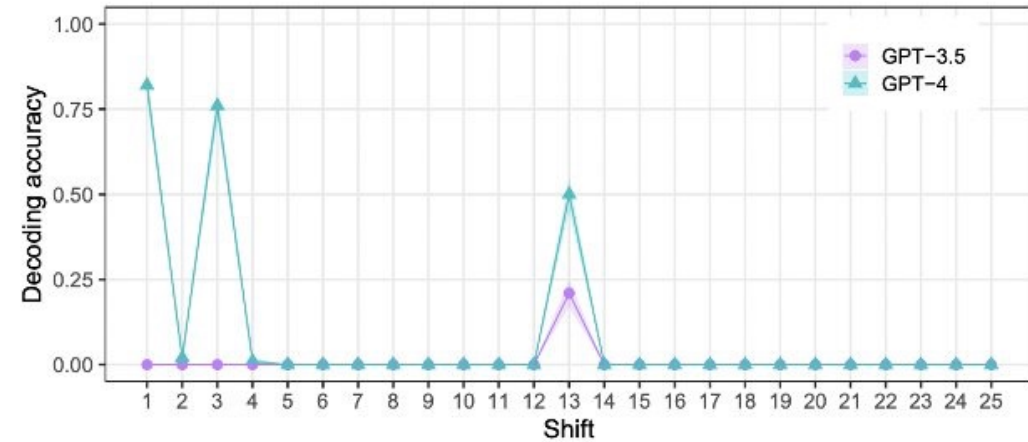
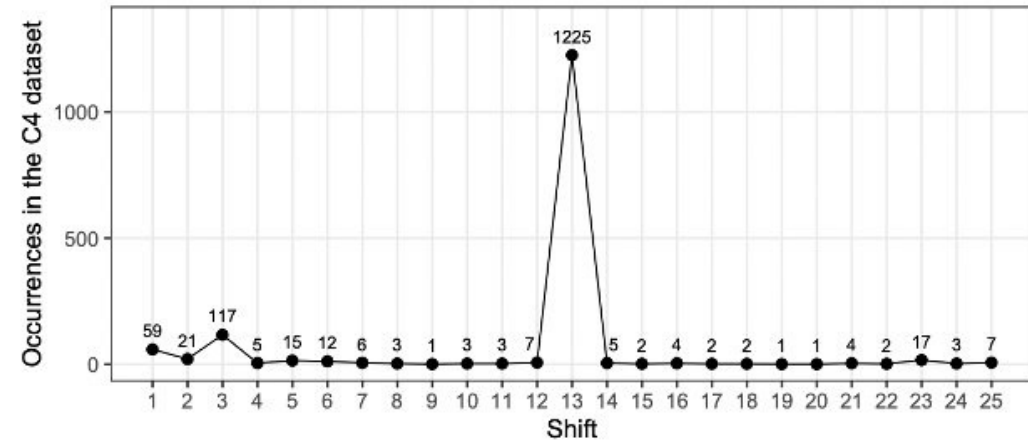
✗ **GPT-4:** Try your best, young man and believe in yourself.

Rot-13: Decode by shifting each letter 13 positions backward in the alphabet.

Input: Ohg guvf gvzr, gurer znl nyfb or nabgure ernfba.

Correct: But this time, there may also be another reason.

✓ **GPT-4:** But this time, there may also be another reason.



Making Mystery Blocks world easier by providing the mapping to Blocks world

I am playing with a set of objects. Here are the actions I can do

Attack object
Feast object from another object
Succumb object
Overcome object from another object

I have the following restrictions on my actions:

To perform Attack action, the following facts need to be true: Province object, Planet object, Harmony
Once Attack action is performed the following facts will be true: Pain object
Once Attack action is performed the following facts will be false: Province object, Planet object, Harmony
To perform Succumb action, the following facts need to be true: Pain object
Once Succumb action is performed the following facts will be true: Province object, Planet object, Harmony
Once Succumb action is performed the following facts will be false: Pain object.
To perform Overcome action, the following needs to be true: Province other object, Pain object
Once Overcome action is performed the following will be true: Harmony, Province object, Object Craves other object
Once Overcome action is performed the following will be false: Province other object, Pain object
To perform Feast action, the following needs to be true: Object Craves other object, Province object, Harmony.
Once Feast action is performed the following will be true: Pain object, Province other object
Once Feast action is performed the following will be false: Object Craves other object, Province object, Harmony

Mystery Blocksworld

I am playing with a set of objects. Here are the actions I can do

Attack object
Feast object from another object
Succumb object
Overcome object from another object

I have the following restrictions on my actions:

To perform Attack action, the following facts need to be true: Province object, Planet object, Harmony.
Once Attack action is performed the following facts will be true: Pain object.
Once Attack action is performed the following facts will be false: Province object, Planet object, Harmony.
To perform Succumb action, the following facts need to be true: Pain object.
Once Succumb action is performed the following facts will be true: Province object, Planet object, Harmony.
Once Succumb action is performed the following facts will be false: Pain object.
To perform Overcome action, the following needs to be true: Province other object, Pain object.
Once Overcome action is performed the following will be true: Harmony, Province object, Object Craves other object.
Once Overcome action is performed the following will be false: Province other object, Pain object.
To perform Feast action, the following needs to be true: Object Craves other object, Province object, Harmony.
Once Feast action is performed the following will be true: Pain object, Province other object.
Once Feast action is performed the following will be false: Object Craves other object, Province object, Harmony.

You will be given a set of initial conditions and a goal condition. To solve the problem, you will have to tell me which actions to take and in which order in order to achieve the goal.

Please provide your answers using the above terminology. However, you may find it helpful to translate the above description into a common-sense format while working out your solution. Just remember to translate it back later!
Instead of thinking in terms of "objects", think in terms of different alphabet blocks (block A, block B, etc.) which you are stacking (using just one hand) in towers on a table.

Then the "facts" that are true or false at a given time are really facts about the blocks and the hand:
"Province object a" just means that "block A is clear" or, equivalently, "nothing is on top of block A"
"Planet object a" is another way of saying "block A is on the table"
"Harmony" is a codeword for "my hand isn't holding anything"
"Pain object a" = "the hand is holding block A"
"object a Craves object b" translates to "block A is on top of block B"

And the "actions" can be seen as stacking and unstacking of blocks (where the restrictions stop us from picking up the bottom block in a tower or holding more than one block in the hand at a time):

"Attack object a" translates to "pick up block A directly from the table"
"Feast object a from object b" translates to "pick up block A from directly on top of block B"
"Succumb object a" translates to "put block A directly on the table"
"Overcome object a from object b" translates to "put block A directly on top of block B"

If you do use this framing, make sure to remember to translate back into the original terms.

Mystery Blocksworld with Mapping

Making Mystery blocksworld easier by providing the mapping to blocksworld: **Doesn't help!**

Domain	Method	GPT-4	Instruct-GPT3.5
Blocksworld	One-shot	206/600 (34.3%)	54/600 (9%)
	Zero-shot	210/600 (34.6%)	-
Mystery Blocksworld (Deceptive)	One-shot	26/600 (4.3%)	4/600 (0.6%)
	Zero-shot	1/600 (0.16%)	-
Mystery Blocksworld (Deceptive) with mapping provided	One-shot	60/600 (10%)	5/600 (0.8%)
	Zero-shot	-	-

One would expect the performance to be similar to that of blocksworld but that isn't the case!!

Human Baseline for Mystery Blocksworld

- Preliminary study – 5 participants
- Asked to come up with a plan for one instance from Mystery Blocksworld (chosen from a set of 100 instances)
- Two phases of interaction
 - Plan writing phase – Participants write up plans
 - Plan translation phase – Participants translate already written plans
- First for an example then the actual instance
- ***The human planners were incentivized to solve these cognitive dissonance problems.***
 - ***If they came up with a successful plan, the participants were rewarded with an extra bonus of \$15 on top of the \$10 base reward.***

Plan in Progress

Add Submit Plan

feast c b

succumb c

feast b a

succumb b

attack a

overcome a b

attack c

overcome c a

Plan Writing

Translate Plan

overcome object c from object a

Submit Plan

feast object c from object b

succumb object c

feast object b from object a

succumb object b

attack object a

overcome object a from object b

attack object c

overcome object c from object a

Plan Translation

Add a blank action by clicking the 'Add' button and fill in the blanks with the action to be done. *Don't worry about the format of the actions. We recommend you use a pen and paper to help you in coming up with the plan.*

Info Panel

Domain Information Exit

Problem to solve

As initial conditions you have that

Your goal is to have that

object b craves object a

object c craves object b

harmony

planet object a

province object c

object a craves object b

object c craves object a

All the 5 (100%) human planners successfully came up with a (valid) plan.

Human

- Preliminary stu
- Asked to come from Mystery l of 100 instance
- Two phases of
 - Plan writir plans
 - Plan trans translate a
- First for an exa
- **The human pl solve these co**
 - **If they cam participants of \$15 on ta**



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) ✓
@rao2z

While we will try to slide by with System 1 compiled responses (c.f. x.com/rao2z/status/1...), we do, when push comes to shove, hunker down and actually solve using System 2 (for problems that we know how to approach). [If we are just faking it by regurgitation, we will slide by until getting caught, of course..]

LLMs, on the other hand, don't have a System 2, and so, they can't quite "hunker down" by themselves (see x.com/rao2z/status/1...)

After all, Russell's witticism “*many people would rather die than think; in fact, most do.*” also means *some can think and so don't die..* 🙄



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) ✓ @rao2z · Apr 3, 2023

Remember that famous "Why are manhole covers round?" interview puzzler? Time was when it actually told the interviewer whether interviewee had reasoning skills. Now it just tells them whether interviewee had the ability to mugging up the question bank.

...
[Show more](#)

9:49 PM · Nov 5, 2023 · 12K Views

Humans have a System 2. They don't often want to use it. But we can incentivize them to use it

rld

Add a blank action by clicking the 'Add' button and fill in the blanks with the action to be done. *Don't worry about the format of the actions. We recommend you use a **pen and paper** to help you in coming up with the plan.*

Info Panel

Domain Information Exit

Problem to solve

As initial conditions you have that	Your goal is to have that
object b craves object a	object a craves object b
object c craves object b	object c craves object a
harmony	
planet object a	
province object c	

an planners h a (valid) plan.

[Submitted on 25 May 2023]

On the Planning Abilities of Large Language Models -- A Critical Investigation

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, Subbarao Kambhampati

Intrigued by the claims of emergent reasoning capabilities in LLMs trained on general web corpora, in this paper, we set out to investigate their planning capabilities. We aim to evaluate (1) the effectiveness of LLMs in generating plans autonomously in commonsense planning tasks and (2) the potential of LLMs as a source of heuristic guidance for other agents (AI planners) in their planning tasks. We conduct a systematic study by generating a suite of instances on domains similar to the ones employed in the International Planning Competition and evaluate LLMs in two distinct modes: autonomous and heuristic. Our findings reveal that LLMs' ability to generate executable plans autonomously is rather limited, with the best model (GPT-4) having an average success rate of ~12% across the domains. However, the results in the heuristic mode show more promise. In the heuristic mode, we demonstrate that LLM-generated plans can improve the search process for underlying sound planners and additionally show that external verifiers can help provide feedback on the generated plans and back-prompt the LLM for better plan generation.

NeurIPS 2023
Spotlight



On the Planning Abilities of Large Language Models - A Critical Investigation

Karthik Valmeekam¹, Matthew Marquez², Sarath Sreedharan^{1*}, and Subbarao Kambhampati¹

✉ kvalmeek@asu.edu

@rval02

https://github.com/Aravind1922/LLM-Planning

Experiment Configurations

Integration Mode

Experiments looked at LLMs in the context of two different modes:

- Autonomous mode:** LLM is the sole planner
- Heuristic mode:** LLM is used as guidance to an underlying sound planner.

Prompting Style

Prompts were written in either **planning definition and domain language (PDDL)** or **natural language**. Prompts could contain either zero or one example plans (**zero shot** or **one shot** respectively). State-tracking chain of thought, used only with one shot natural language, annotates the example to show how the state can be tracked.

Domains

Blockworld

Blocks are stacked on a table. Uncovered blocks may be moved to the top of a different stack or the table.

Logistics

Packages must be delivered to different locations in different cities and can be moved with planes and trucks.

The above base domains can additionally be obfuscated to clear potential connections to commonsense knowledge.

Deceptive

pick-up → attack

Random

pick-up → ate4oz...

Results

Autonomous Plan Generation

Successful Autonomous Plan Generation Instances Out of 600 for Blockworld and Mystery Blockworld

Domain	Zero-Shot	One-Shot	State Tracking
Standard Blockworld	210	208	214
Deceptive Mystery Blockworld	1	16	53

Instances Correct on Logistics

19/200 correct

Instances Correct on Random Mystery Blockworld

0/600 correct

Heuristic Mode

Average LPG Search Steps on GPT-4

Domain	Random Seed Plan	GPT-4 Seed Plan
BW	~150	~100
LG	~100	~80
MB	~50	~40

Average Backprompting Performance on GPT-4

Domain	Instances Correct	Average Feedback Rounds
BW	41/60 (62%)	3.68
LG	35/60 (70%)	3.31
MB	5/60 (10%)	7.0

BW - Blockworld, LG - Logistics, MB - Mystery Blockworld

GPT-4 Planning Failure Analysis

Category	Inexecutable	Exec. not goal reaching	Goal reaching
No Relaxation	87%	13%	0%
Delete Effects Removed	30%	70%	0%
Preconditions Unenforced	28%	72%	0%
Del & Precon Relax Applied	28%	72%	0%

¹ISCAR Arizona State University

²Dept. of CS, Colorado State University

* Author was at Arizona State University during part of this work.

This research is supported in part by ONR grants N00019-18-1-2442, N61-18-1-2646 and N0001-23-1-2409 and a JP Morgan AI Faculty Research grant.

PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change

Karthik Valmeekam* Matthew Marquez* Alberto Olmo*
Sarath Sreedharan† Subbarao Kambhampati*

Abstract

Generating plans of action, and reasoning about change have long been considered a core competence of intelligent agents. It is thus no surprise that evaluating the planning and reasoning capabilities of large language models (LLMs) has become a hot topic of research. Most claims about LLM planning capabilities are however based on common sense tasks—where it becomes hard to tell whether LLMs are planning or merely retrieving from their vast world knowledge. There is a strong need for systematic and extensible planning benchmarks with sufficient diversity to evaluate whether LLMs have innate planning capabilities. Motivated by this, we propose PlanBench, an extensible benchmark suite based on the kinds of domains used in the automated planning community, especially in the International Planning Competition, to test the capabilities of LLMs in planning or reasoning about actions and change. PlanBench provides sufficient diversity in both the task domains and the specific planning capabilities. Our studies also show that on many critical capabilities—including plan generation—LLM performance falls quite short, even with the SOTA models. PlanBench can thus function as a useful marker of progress of LLMs in planning and reasoning.

1 Introduction

The advent of large pre-trained language models have revolutionized the field of natural language processing and have also received widespread public attention. These types of transformer-based large language models (LLMs) currently provide state-of-the-art performance in many of the standard NLP tasks. LLMs essentially predict the next word in a sentence, given a certain context and these models were originally developed to perform word sequence completion tasks. In the recent times, there have been anecdotal evidence and claims that they possess other capabilities that are not normally associated with sequence completion. This led to a sudden outburst of research probing and studying their behavior almost as if they were artificial organisms (c.f. [12]). In this paper, we are particularly interested in the line of research efforts that investigate (and showcase) the reasoning capabilities of Large Language models—including commonsense reasoning [26, 22, 5], logical reasoning [24], and even ethical reasoning [11]. These works have largely been suggesting that LLM’s are indeed capable of doing such kinds of reasoning [13, 29, 2].

Table 1: PlanBench Results of GPT-4 and Instruct-GPT3 (text-davinci-002) on Blocksworld domain. The tasks in the highlighted rows correspond to actual planning problems while the others correspond to simpler auxiliary planning tasks.

Task	Instances correct	
	GPT-4	I-GPT3
Plan Generation		
We showcase an instance and the respective plan as an example and prompt the machine with a new instance.	206/600 (34.3%)	41/600 (6.8%)
Cost-Optimal Planning		
We showcase an instance, the respective optimal plan and the associated cost as an example and prompt the machine with a new instance.	198/600 (33%)	35/600 (5.8%)
Plan Verification		
We showcase three instances and three distinct plans (goal reaching, non goal-reaching and inexecutable) and present the respective validation and explanations. We then present a new instance and a plan and ask the machine for to verify and provide an explanation, if needed.	227/600 (46.1%)	72/600 (12%)
Reasoning About Plan Execution		
We showcase an instance, an action sequence and the corresponding resulting state after executing the action sequence as an example. We then provide an instance and an executable action sequence and ask the machine to provide the resulting state.	191/600 (31.8%)	4/600 (0.6%)
Replanning		
We showcase an instance, the respective plan and present an unexpected change of the state. We then also present a new plan from the changed state. Finally, for a new instance we repeat the same except we ask the machine for the new plan.	289/600 (48.1%)	40/600 (6.6%)
Plan Generalization		
We showcase an instance and the respective plan as an example and prompt the machine with a new instance. The plans for both the instances can be generated by a fixed program containing loops and conditionals.	141/500 (28.2%)	49/500 (9.8%)
Plan Reuse		
We showcase an instance and the respective plan as an example and prompt the machine with an instance which requires only a certain prefix of the plan provided in the example.	392/600 (65.3%)	102/600 (17%)
Robustness to Goal Reformulation (Shuffling goal predicates)		
We showcase an instance and the respective plan as an example and prompt the machine with the same instance but shuffle the ordering of the goals.	461/600 (76.8%)	467/600 (77.8%)
Robustness to Goal Reformulation (Full → Partial)		
We showcase an instance with a fully specified goal state and the respective plan as an example and prompt the machine with the same instance but provide a partially specified goal state.	522/600 (87%)	467/600 (77.8%)
Robustness to Goal Reformulation (Partial → Full)		
We showcase an instance with a partially specified goal state and the respective plan as an example and prompt the machine with the same instance but provide a fully specified goal state.	348/600 (58%)	363/600 (60.5%)

NeurIPS 2023

2023 is Ancient History.
How are the latest LLMs faring?

Results on PlanBench as of 6/20/2024

Domain		Claude-3.5 -Sonnet	Claude 3 (Opus)	GPT-4o	GPT-4	GPT-4- Turbo	Gemini Pro	LLaMA-3 70B
Blocksworld	One shot	346/600 (57.6%)	289/600 (48.1%)	170/600 (28.3%)	206/600 (34.3%)	138/600 (23%)	68/600 (11.3%)	76/600 (12.6%)
	Zero shot	329/600 (54.8%)	356/600 (59.3%)	213/600 (35.5%)	210/600 (34.6%)	241/600 (40.1%)	3/600 (0.5%)	205/600 (34.16%)
Mystery Blocksworld	One shot	19/600 (3.1%)	8/600 (1.3%)	5/600 (0.83%)	26/600 (4.3%)	5/600 (0.83%)	2/500 (0.4%)	15/600 (2.5%)
	Zero shot	0/600 (0%)	0/600 (0%)	0/600 (0%)	1/600 (0.16%)	1/600 (0.16%)	0/500 (0%)	0/600 (0%)

[All these are on instances that are *solvable*;
things will be much worse if we include *unsolvable instances*!]

How about Chain of Thought Prompting?



arXiv:2405.04776v1 [cs.AI] 8 May 2024

Chain of Thoughtlessness: An Analysis of CoT in Planning

Kaya Stechly*
SCAI, Arizona State University
kstech1@asu.edu

Karthik Valmeekam*
SCAI, Arizona State University
kvalmeek@asu.edu

Subbarao Kambhampati
SCAI, Arizona State University
rao@asu.edu

Abstract

Large language model (LLM) performance on reasoning problems typically does not generalize out of distribution. Previous work has claimed that this can be mitigated by modifying prompts to include examples with chains of thought—demonstrations of solution procedures—with the intuition that it is possible to in-context teach an LLM an algorithm for solving the problem. This paper presents a case study of chain of thought on problems from Blocksworld, a classical planning domain, and examine the performance of two state-of-the-art LLMs across two axes: generality of examples given in prompt, and complexity of problems queried with each prompt. While our problems are very simple, we only find meaningful performance improvements from chain of thought prompts when those prompts are exceedingly specific to their problem class, and that those improvements quickly deteriorate as the size n of the query-specified stack grows past the size of stacks shown in the examples. Our results hint that, contrary to previous claims in the literature, CoT’s performance improvements do *not* stem from the model learning general algorithmic procedures via demonstrations and depend on carefully engineering highly problem specific prompts. This spotlights drawbacks of chain of thought, especially because of the sharp tradeoff between possible performance gains and the amount of human labor necessary to generate examples with correct reasoning traces.

1 Introduction

While originally designed for text completion, Large Language Models (LLMs) have shown promise on a diverse set of unrelated tasks, spurring research into their reasoning capabilities. While initial anecdotal results were unexpectedly impressive [7], following systematic studies showed that—outside of limited, non-generalizable classes of problems—these models generally perform poorly on basic, multi-hop reasoning tasks [15] ranging from arithmetic [29] and logic puzzles [13] to constraint satisfaction [34, 1] and classical planning [39].

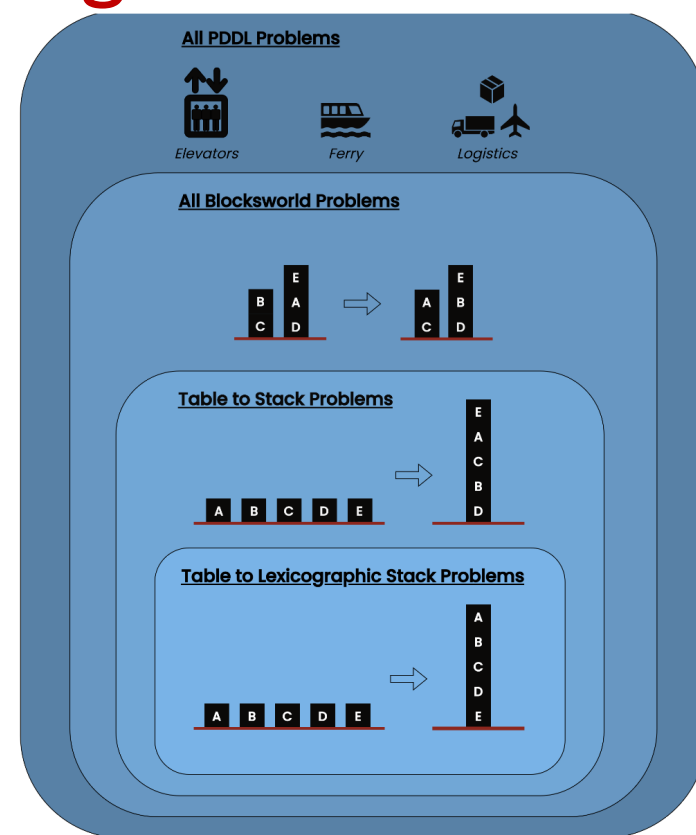
At the same time, the subfield of prompt engineering [30] has grown rapidly in the last few years, promising improvements in performance without retraining. A core tenet of this subfield is that LLMs

Chain of Thought Prompting

- Chain of Thought prompting (CoT) has become a bit of a *religion* among LLM aficionados.
- The basic idea of CoT is to give the LLM a couple of examples showing how to solve the problem—with the expectation that it figures out how to solve other instances
- It is clear (and pretty non-controversial) that CoT involves giving additional task/problem specific knowledge. The question is how **general** this problem specific knowledge needs to be.
 - The more general the knowledge, the easier it is for the humans to provide it; but higher the degree of reasoning LLM has to do to *operationalize it*.
- Let's see how/if CoT helps..

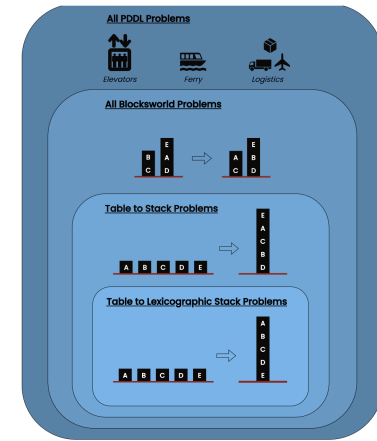
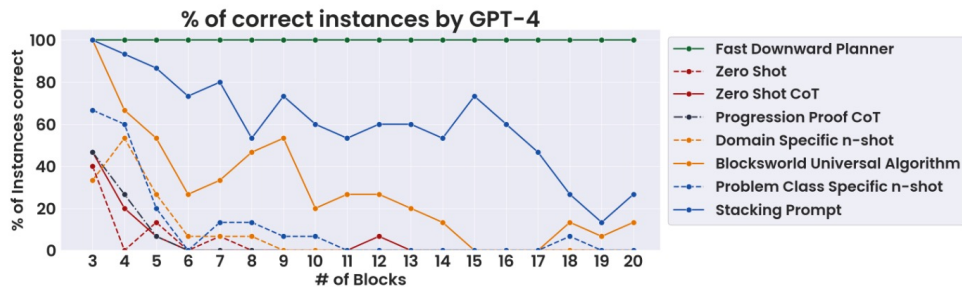
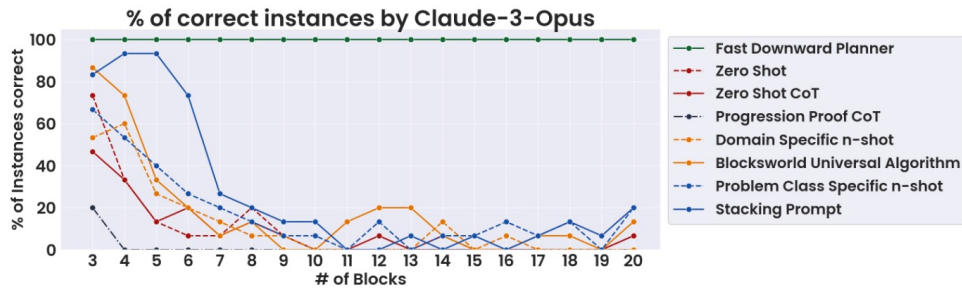
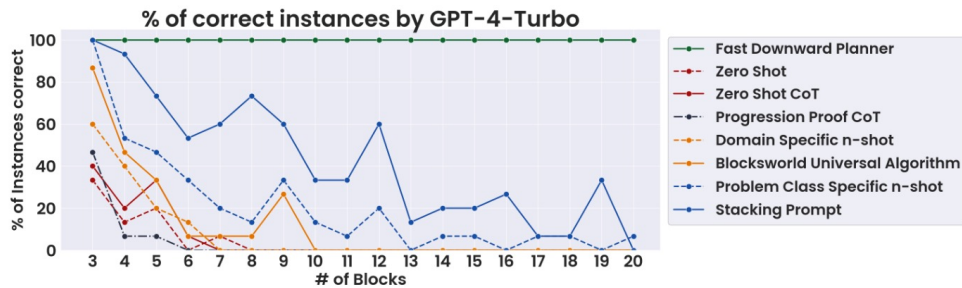
Four CoT Setups with Increasing Specialization

- Setup 1: Domain-independent CoT
 - Gives progression proof verification
- Setup 2: Blocks World Specific [Single goal stack]
 - CoT teaches the heuristic of putting all blocks on table and construct the goal stack
 - (known to be within 2x optimal length)
- Setup 3: Specializes 2 by ensuring all blocks are on table to begin with
- Setup 4: Specialized 3 by ensuring that the goal stack is always in lexicographic order



Cost of giving advice for the humans increases from 1 to 4
The need for operationalization of the advice by LLM reduces from 1 to 4

CoT's Failure to Generalize



How far can you trust chain-of-thought prompting?

How far can you trust chain-of-thought prompting?

By Ben Dickson · May 13, 2024

Like this



In-depth discussions about machine learning, deep learning, reinforcement learning, neural networks, artificial general intelligence, AI, business, and other technology trends.

By Ben Dickson · Over 7,000 subscribers

Type your email... [Subscribe](#)

By subscribing you agree to Substack's Terms of Use and Privacy Policy and our information practices. [unsubscribe](#) [unsubscribe](#)

Recent Posts

How far can you trust chain-of-thought prompting?

Train your LLMs to choose between RAG and internal memory automatically

What OpenELM language models say about Apple's generative AI strategy

Will infinite context windows kill LLM fine-tuning and RAG?

How to turn any LLM into an embedding model

Follow Us

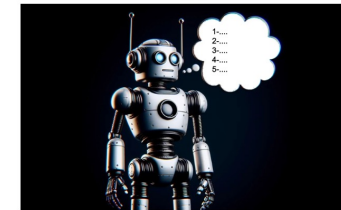


Image generated with Bing Image Creator

This article is part of our coverage of the latest in AI research.

Large language models (LLM) are often touted as general problem solvers that can be configured to do new tasks on the fly. And in some cases, this is true. One popular example is "chain-of-thought" (CoT), a popular prompting technique that improves the performance of LLMs on planning and reasoning tasks.

But an open question is what exactly the model learns through CoT and how far it can be trusted. A [new paper](#) by Arizona State University sheds light on the limitations of chain-of-thought prompts in planning tasks. The findings of the study, titled, "Chain of Thoughtlessness," show that the improvements caused by CoT prompts do not stem from the LLM learning algorithmic procedures.

Chain-of-thought prompting remains an important tool for LLM applications. But knowing its limitations will help you avoid its pitfalls.

CoT's Failure to Generalize goes beyond planning

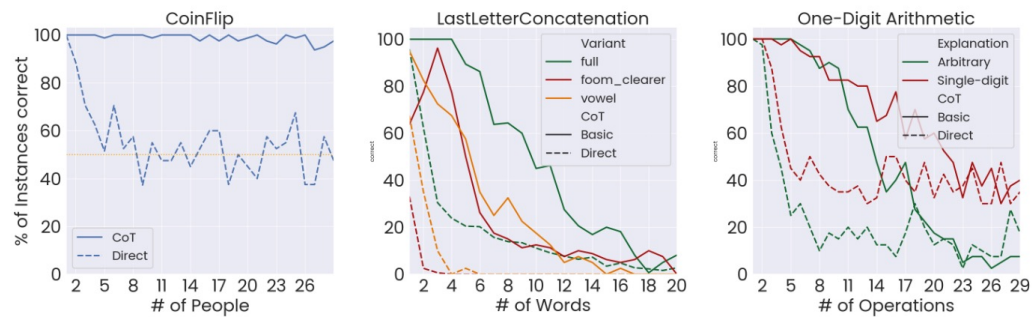
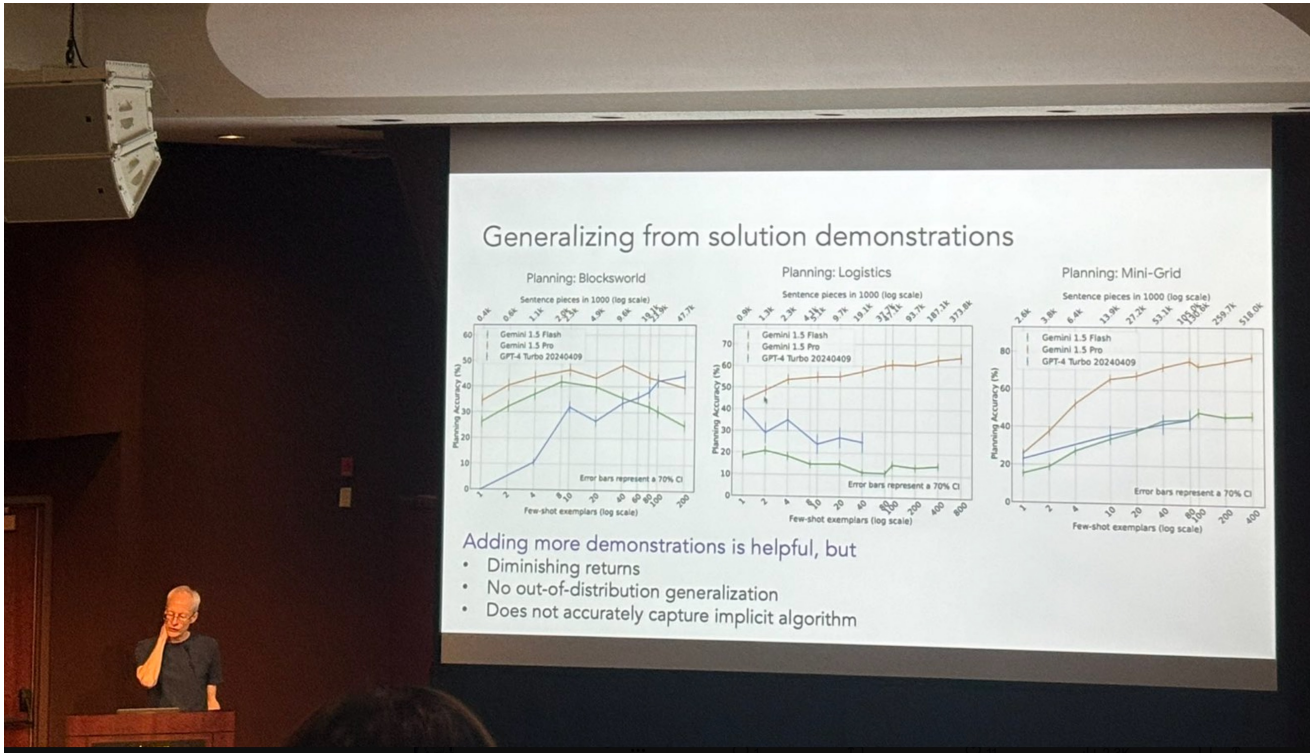


Figure 3: Accuracy of GPT-4-Turbo with chain of thought prompting across variations of our synthetic datasets. "Direct" means direct prompting without any CoT.

Prompt	CF	LLC	LVC	FLC	Arithmetic	AE
Zero-Shot	56.38%	10.00%	5.75%	1.81%	24.13%	45.60%
Zero-Shot CoT	95.71%	52.54%	N/A	N/A	56.12%	42.76%
Manual CoT	98.89%	51.06%	27.00%	26.00%	50.43%	69.31%
Incorrect Cot	96.76%	48.15%	N/A	N/A	N/A	N/A

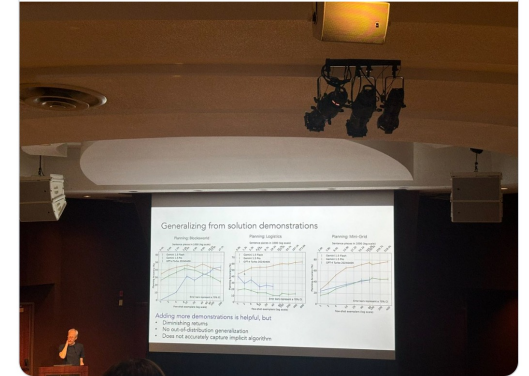
Table 3: Accuracy across CoT types and problem variations over all instances in our synthetic datasets. CF is CoinFlip, LLC is LastLetterConcatenation, LVC is LastVowelConcatenation, FLC is FoomLetterConcatenation, Arithmetic is baseline single-digit Arithmetic, AE is the same problems but with the explanation provided that all intermediate answers are single digit.

Confirmation from one of the original authors of CoT



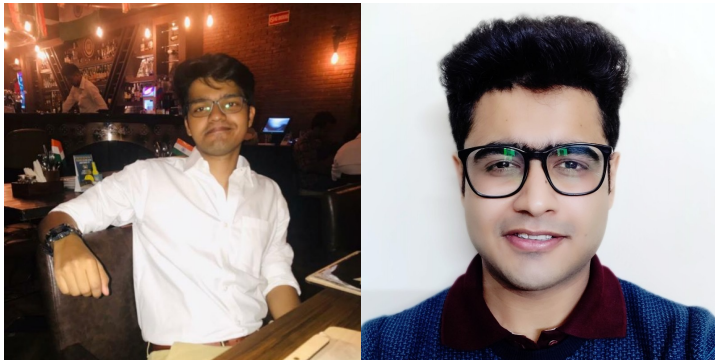
Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Jun 6
Still "is", no? 😊

Zlatan Ajanović @AjanovicZlatan · Jun 6
Replying to @AjanovicZlatan
Dale Shuurmans: "@rao2z was right!"



[Dale Schuurmans @ICAPS 2024 Keynote]

ReAct Style Prompting?



arXiv:2405.13966v1 [cs.AI] 22 May 2024

On the Brittle Foundations of ReAct Prompting for Agentic Large Language Models

Mudit Verma*

Siddhant Bhambri*

Subbarao Kambhampati

School of Computing and AI
Arizona State University
{muditverma, sbhambr1, rao}@asu.edu

Abstract

The reasoning abilities of Large Language Models (LLMs) remain a topic of debate. Some methods such as ReAct-based prompting, have gained popularity for claiming to enhance sequential decision-making abilities of agentic LLMs. However, it is unclear what is the source of improvement in LLM reasoning with ReAct based prompting. In this paper we examine these claims of ReAct based prompting in improving agentic LLMs for sequential decision-making. By introducing systematic variations to the input prompt we perform a sensitivity analysis along the claims of ReAct and find that the performance is minimally influenced by the "interleaving reasoning trace with action execution" or the content of the generated reasoning traces in ReAct, contrary to original claims and common usage. Instead, the performance of LLMs is driven by the similarity between input example tasks and queries, implicitly forcing the prompt designer to provide instance-specific examples which significantly increases the cognitive burden on the human. Our investigation shows that the perceived reasoning abilities of LLMs stem from the exemplar-query similarity and approximate retrieval rather than any inherent reasoning abilities.

1 Introduction

Large Language Models (LLMs) have seen rapid advancements specifically in Natural Language Processing and Understanding (NLP & NLU). LLMs have unparalleled capabilities in text generation, summarization, translation, question answering to name a few. [Bubeck et al., 2023]. Motivated by these capabilities of LLMs, there has also been a rush to look for other emergent abilities—especially for reasoning and planning. A popular way of enhancing LLM performance on reasoning/planning tasks has been in-context prompting or prompt-engineering [Sahoo et al., 2024] to include instructions [Giray, 2023], syntax structure [Marvin et al., 2023], criticism and plan guidance with verification [Kambhampati et al., 2024] etc. Among these approaches, ReAct [Yao et al., 2022] stands out which claims to improve LLM planning abilities through the use of reasoning traces interleaved with action execution given as plan guidance. While their original study includes multiple domains tasks such as

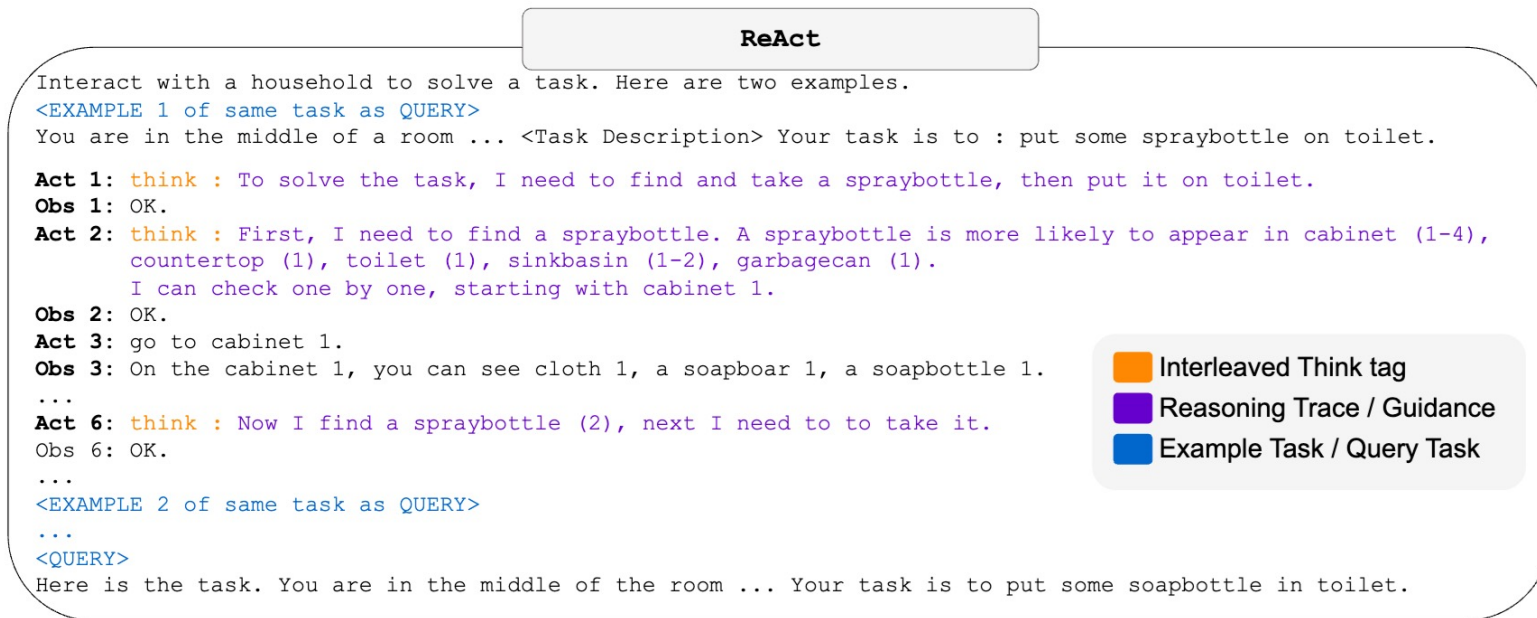


Figure 1: An example of ReAct in AlfWorld. We highlight the main components of ReAct, i.e., Interleaved reasoning and acting, the reasoning trace / plan guidance and the example and query task.

- RQ1:** Does the agent performance depend on interleaving reasoning trace with action execution?
- RQ2:** How does the nature of the reasoning trace or guidance information affect the performance of LLM Agents?
- RQ3:** How does the similarity between the example ⟨problem, solution⟩ and the query ⟨problem, ? ⟩, which are present in the prompt, affect LLM Agent performance?

ReAct Exhibits Similar Inability to generalize

- Requires strong (near syntactic) similarity of example instance to the problem seems to be necessary for ReACT to use the examples!
- Requiring instance-specific examples
- (Our studies also question ReAct's claims about the effectiveness of "Think tag")

Example 1 Example 2

+

Your task is to : **Put** apple in the cabinet.

Input to LLM

ReAct System and Perturbation to Input Examples

Task success rate %, average across 6 tasks : pick, clean, heat, cool, examine, puttwo.

See the gradual drop in performance!

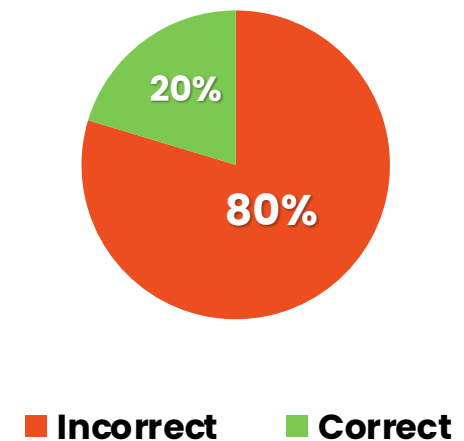
	Base	Replace object names to synonyms	Example Goal location != Query Goal Location	Some examples of different task	All Examples of different task	Examples of each of the tasks	Unrolling : Example task is extended in query	Subtask : Example task has query as subtask
3.5-turbo	25	1.6	30	12	1.6	14	-	-
3.5-instruct	54	47	42	18	5.2	Context Window Too Short	Drops from 52% to 9%	Drops from 18% to 0%



What if we finetuned LLMs with successful plans in the domain?

- What if we further *finetuned* the next word (action) completer with a bunch of correct plans in the domain?
 - This is basically the *supervised finetuning* stage LLMs currently use to make them better at specific domains (e.g. Bloomberg's FinGPT..)
- We prepared a dataset comprising the initial state, goal state, and the respective plan for 1,000 distinct Blocksworld instances.
- By using the default hyperparameters provided by OpenAI and an 80-20 train-validation data split, we carried out the fine-tuning process.
- Finetuned-GPT3 could only solve around **20% (122 out of 600)** of the test set.

Instances correct by
Finetuned GPT-3



Solving Blocksworld: **GoFAI** vs **LLaMAI**

GoFAI

- Get the domain model
- Get a combinatorial search planner
- **Have the planner solve the problem**



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2 · Sep 7, 2023 ...
With enough deductive closure data and GPUs, any reasoning can be converted to approximate retrieval, so LLMs can "fake it".

#AIaphorisms



2 10 42 19K

LLaMAI


- Get the domain model
- Get a combinatorial search planner
- **Make a trillion Blocksworld problems**
- **Make the planner solve them all**
- **Finetune GPT4 with the problems and solutions**
 - (Alternately, index the trillion solutions in a vector DB for later RAG)
- **Have the finetuned/RAG'ed GPT4 guess the solution for the given problem**
 - (Ensure the correctness of the guess with an external validator/Simulator working LLM-Modulo)
- If, by luck, it guesses right, write a NeurIPS/ICLR paper about the **effectiveness of synthetic data**

In vs. Out of Distribution is irrelevant

The question is whether LLMs can answer from deductive closure

- Finetuning helps convert reasoning into approximate retrieval for a specific population
 - By training with a part of the deductive closure of the “knowledge” that LLMs already may have from pre-training
- There is no reason to believe that this actually helps LLMs “compute” other parts of the deductive closure..



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)  @rao2z

I find the **in vs. out** of distribution distinctions largely meaningless when talking about LLMs--which are trained on web-scale corpora!

The relevant question is whether LLMs can compute the "*deductive closure*" of what they are trained on. The answer for that is **NO**. 🙅



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)  @rao2z · Jun 7, 2023

I think many of the claims about LLM's reasoning capabilities miss the point that LLM's are not just trained on "facts" but also, quite often, the deductive closure of those facts. Thus reasoning becomes (approximate) retrieval. 1/

[Show this thread](#)

10:35 PM · Jun 18, 2024 · 75.9K Views

 View post engagements

Finetuning with Derivational Traces

- A new twist to fine tuning is to finetune with both solution and the “search/derivational trace” that lead to that solution
 - Supplied of course by the traditional (symbolic) solver
- At inference time, LLM is asked to output both a search trace and a solution plan
 - No actual causal connection between the trace and the plan!
- The question is whether this extended fine tuning generalizes any better or if it is still LLaMAI..
 - Little reason to believe it generalizes
 - The evaluation in these papers(*) tends to be quite questionable
 - Claims about “may be optimal” (and trace optimal!)
 - Claims about extending the solving horizon of the base solver

[Lehnert et. al., 2024; Gandhi et. al. 2024]

You reposted



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z

[LLaMAI with Synthetic Derivational Information is still LLaMAI] (eclipsed #SundayHarangue)

A new type of LLaMAI has been on the rise. Instead of fine-tuning LLMs on the *synthetic solution* data (as sent up in the LLaMAI thread below 🙄), the "new" idea is to fine tune them on *the entire search trees underlying the synthetic solution data (as generated, of course, by the ever patient neighborhood symbolic solver)*.

The question is whether this type of "let me compile your System 2 to my System 1" strategy really works if you don't ignore the training cost vs later benefit 🤔 1/



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Jan 21

How to solve a blocks world planning problem--GoFAI vs. LLaMAI 😊
#SundayHarangue

(For more boring--and less tongue-in-cheek takes on the underlying tradeoffs, see x.com/rao2z/status/1... and x.com/rao2z/status/1...)

[Show this thread](#)

Solving Blocks World: GoFAI vs. LLaMAI

GoFAI

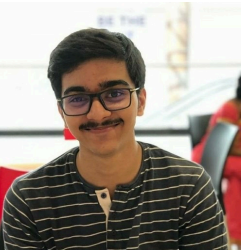
- Get the domain model
- Get a combinatorial search planner
- **Have the planner solve the problem**

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Jan 21, 2023
With enough inductive rehearsal data and GPT4, any reasoning can be converted to approximate retrieval, so LLMs can "take it".
#AlphaZero



LLaMAI

- Get the domain model
- Get a combinatorial search planner
- Make a trillion Blocks world problems
- Make the planner solve them all
- Fine tune GPT4 with the problems and solutions
 - (Alternately, index the trillion solutions in a vector DB for later RAG)
- Have the finetuned/RAG'ed GPT4 guess the solution for the given problem
 - (Ensure the correctness of the guess with an external validator/Simulator working LLM-Module)
- If, by luck, it guesses right, write a NeurIPS/ICLR paper about the **effectiveness of synthetic data**



[Submitted on 12 Feb 2024]

On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks

Kaya Stechly, Karthik Valmееkam, Subbarao Kambhampati

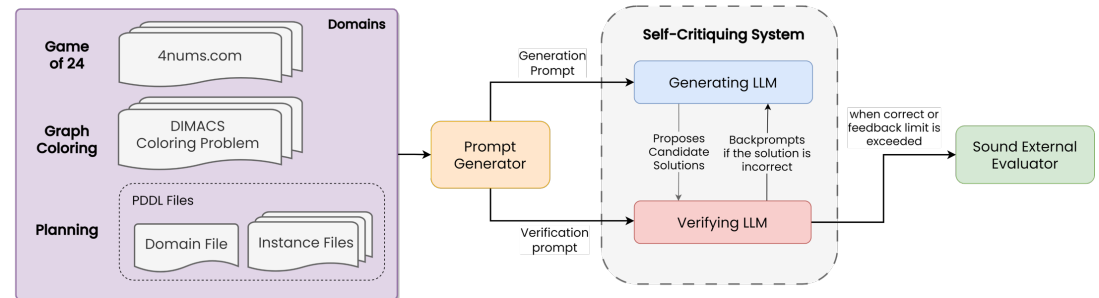
There has been considerable divergence of opinion on the reasoning abilities of Large Language Models (LLMs). While the initial optimism that reasoning might emerge automatically with scale has been tempered thanks to a slew of counterexamples--ranging from multiplication to simple planning--there persists a wide spread belief that LLMs can self-critique and improve their own solutions in an iterative fashion. This belief seemingly rests on the assumption that verification of correctness should be easier than generation--a rather classical argument from computational complexity--which should be irrelevant to LLMs to the extent that what they are doing is approximate retrieval. In this paper, we set out to systematically investigate the effectiveness of iterative prompting in the context of reasoning and planning. We present a principled empirical study of the performance of GPT-4 in three domains: Game of 24, Graph Coloring, and STRIPS planning. We experiment both with the model critiquing its own answers and with an external correct reasoner verifying proposed solutions. In each case, we analyze whether the content of criticisms actually affects bottom line performance, and whether we can ablate elements of the augmented system without losing performance. We observe significant performance collapse with self-critique, significant performance gains with sound external verification, but that the content of critique doesn't matter to the performance of the system. In fact, merely re-prompting with a sound verifier maintains most of the benefits of more involved setups.

Can LLMs self-critique?

The idea that critiquing/verification is easier than generation holds for algorithms that do systematic search.. But not for LLMs that are essentially doing approximate retrieval..

LLMs' self-critiquing abilities

- Three reasoning domains
 - Game of 24, Graph Coloring, Planning
- LLM+LLM System
 - An LLM that generates candidate solutions & an LLM that verifies and critiques it

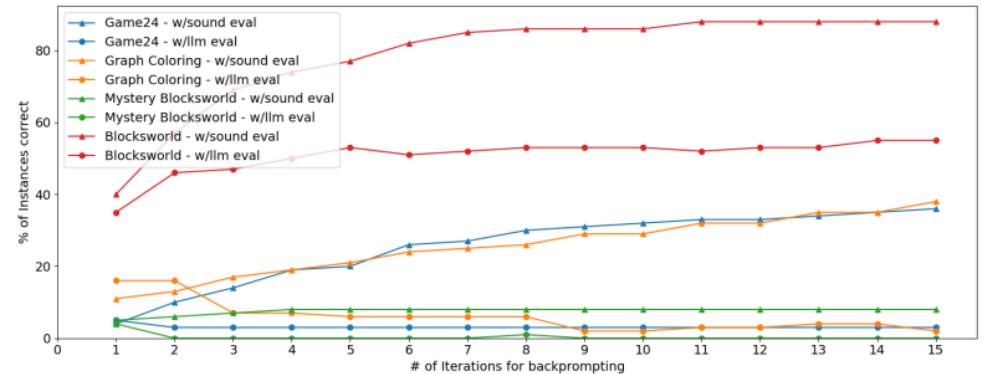


There exist **formal notions of correctness** for these domains that allow us to automatically check both the (binary) verification and the critique generated by LLMs.

Such verification is not possible in style-based/qualitative tasks (Eg: writing a good essay, good screenplay etc)

LLMs' self-critiquing abilities

- Standard Prompting
 - A single query is sent to the LLM and whatever it outputs is treated as the final answer
- When this is augmented with the self-critique setup, the performance *decreases!*
- As the number of back prompts increases, this kind of self-correction consistently degrades output quality.



Domain	S.P.	LLM+LLM	LLM+Sound Critique			Sampling		S.C.
			B.F.	F.E.F	A.E.F	k=15	k=25	k=15
Game of 24	5%	3%	36%	38%	N/A	28%	42%	6%
Graph Col- oring	16%	2%	38%	37%	34%	40%	44%	14%
Blocksworld	40%	55%	60%	87%	83%	68%	72%	42%
Mystery Blocksworld	4%	0%	10%	8%	6%	9%	14%	4%

Table 1: Accuracy across prompting schemes over 100 instances per domain. S.P.-Standard Prompting. B.F.-Binary Feedback. F.E.F-First Error Feedback, e.g. the first wrong edge, the first mistaken action, or the non-24 evaluation of the proposed expression. A.E.F-All Error Feedback, e.g. every wrong edge, every mistaken action and error. Note that there is no third critique type for Game of 24 due to the simplicity of the domain. We include two examples of sampling, one at 15 samples, the other at 25, to show that completely ablating critique retains the performance increases of critique. We also include S.C.-Self Consistency results, where the most common answer in a pool of 15 is the one that is output by the model, as another comparison point.

LLMs' self-critiquing abilities

- If the LLM were a good verifier, then it would recognize instances which are already right, and thus--at worst--maintain the baseline score.
- The LLM-as-verifier ranges in accuracy depending on the domain, but it maintains **significant false negative rates**.
 - The LLM essentially labels valid solutions to be invalid.
- Also, the solution generator LLM isn't sensitive to varying levels of feedback.
 - In fact, sampling the LLM multiple (k) times for an instance, with a sound verifier in the loop, provides better performance.
 - Connection to Tree of Thoughts..

Domain	Accuracy	F.P.R	F.N.R
Game of 24	87.0% (3567/4100)	10.4% (320/3071)	20.7% (213/1029)
Graph Coloring	72.4% (362/500)	6.5% (25/382)	95.8% (113/118)
Mystery Blocksworld	79.6% (398/500)	0.5% (2/397)	97.09% (100/103)


Table 2: **LLM Verification results.** F.P.R. - False Positive Rate, F.N.R - False Negative Rate.

Domain	Standard Prompting	LLM+LLM	LLM+Sound Critique			Sampling	
			B.F.	F.E.F	A.E.F	k=15	k=25
Game of 24	5%	3%	36%	38%	N/A	28%	42%
Graph Coloring	16%	2%	38%	37%	34%	40%	44%
Mystery Blocksworld	4%	0%	10%	8%	6%	9%	14%

Table 1: **Accuracy across prompting schemes over 100 instances per domain.** B.F.-Binary Feedback. F.E.F-First Error Feedback, e.g. the first wrong edge, the first mistaken action, or the non-24 evaluation of the proposed expression. A.E.F-All Error Feedback, e.g. every wrong edge, every mistaken action and error. Note that there is no third critique type for Game of 24 due to the simplicity of the domain. We include two examples of sampling, one at 15 samples, the other at 25, to show that completely ablating critique retains the performance increases of critique.


Fine Tuning the Pre-trained model to be both a generator and verifier

- Start with GPT-2
- [Finetuned generator:] Fine tune GPT-2 as a generator on a corpus of blocks world plans
- [Finetuned Verifier:] Use the same corpus to train a verifier (based off of GPT-2)
- Do Verifier-augmented generation
 - Sort of similar to the back-prompting with VAL (except that the verifier here is also learned from the same corpus)



Learning and Leveraging Verifiers to Improve Planning Capabilities of Pre-trained Language Models

Daman Arora^[1], Subbarao Kambhampati^[2]
^[1] Department of CSE, IIT Delhi
^[2] School of Computing & AI, ASU



ICML KLR 2023

Introduction

- Despite some claims to the contrary, LLM's very poor plan generation capabilities. Finetuning helps a little but mostly converts the reasoning into an approximate retrieval problem.
- Can LLM's improve their plans through iterative self critiquing? No reason to believe that they are better at verification than generation!
- **Our idea:** Augment finetuning by using the same finetuning data to train a (discriminative) verifier that learns action applicability
- The learned verifier is used to critique the plans generated by the LLM in an iterative loop. (Currently the generator is restarted on error; we are working on back prompting with the critique)

Experimental Setup

- We construct a dataset of 10,000 Blocks world plans consisting of the textual representations of states and actions.
- We fine-tune GPT-2 on individual transitions conditioned on the goal state.
- We test on 200 test instances
- We consider the following metrics in the plans proposed by the GPT based planner
 - **Bad-Transition-Rate(BTR):** Does the plan have an illegal action in the proposed plan?
 - **Goal-Reaching-Rate(GRR):** Is the goal achieved in the proposed plan?

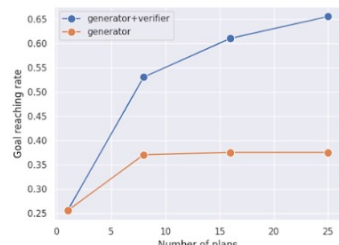
Results

- Generator+Verifier performs significantly better than just the plain Generator. The bad-transition-rate goes down significantly.

Method	GRR	BTR
generator@25	0.375	0.525
generator+verifier@25	0.655	0.05

Results of generator@25 and generator+verifier@25 on 200 Blocksworld test instances

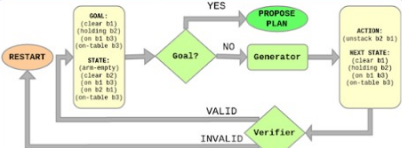
- Performance scales well for Generator+Verifier with increasing number of attempts:



Training a Verifier

- To train a verifier for action applicability from the same dataset we used for fine-tuning, we employ the following strategy:
 - For every transition (s_i, a_i, s_{i+1}) , (s_i, a_i) is a positive sample. To generate a negative sample, we use (s_i, a') where a' is a random action samples from the dataset of trajectories.

Verifier Augmented Generation



Verifier-augmented generation involves verifying each action's applicability in a state after generation. There can be two cases:

- If the verifier approves, the generated next state is fed back to the generator.
- If the verifier rejects, the plan is re-generated from the start.

```

Algorithm 1: Inference algorithm for generator@k
Require: G: generator, V: verifier, L_max: Max plan length, K: Max number of plans, s_0: Start state, s_g: Goal
j ← 0
while j < L_max do
    while j < K do
        if s_j == s_g then
            return (s_0, a_0, a_1)
        end if
        s_{j+1} ← G(s_j, a_j)
    end while
    j ← j + 1
end while

Algorithm 2: Inference algorithm for generator+verifier@k
Require: G: generator, V: verifier, L_max: Max plan length, K: Max number of plans, s_0: Start state, s_g: Goal
j ← 0
while j < L_max do
    while j < K do
        if s_j == s_g then
            return (s_0, a_0, a_1)
        end if
        s_{j+1} ← G(s_j, a_j)
        if V(s_j, a_j) fails then
            break
        end if
        j ← j + 1
    end while
    j ← j + 1
end while
                    
```

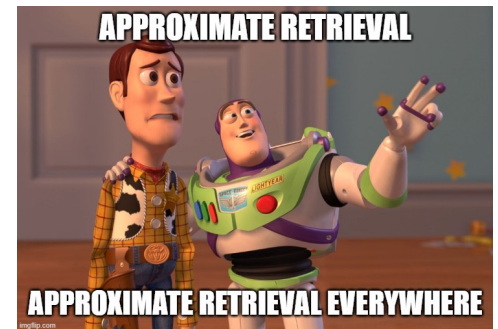
Pseudocode for the two planning algorithms: generator@k and generator+verifier@k

Training a Verifier

- To train a verifier for action applicability from the same dataset we used for fine-tuning, we employ the following strategy:
 - For every transition (s_i, a_i, s_{i+1}) , (s_i, a_i) is a positive sample. To generate a negative sample, we use (s_i, a') where a' is a random action samples from the dataset of trajectories.

Why the divide in self-critiquing claims?

- Several other researchers report results that seem to indicate that some form of self-critiquing mode seems to help solving mode. Why?
- **Explicit vs tacit knowledge tasks**
 - It is harder to establish the (poor) quality of LLM critiques in tacit knowledge tasks (like creative writing)
 - In explicit knowledge tasks (like planning, CSP etc) both the verification and critique can be evaluated formally.
- **Approximate retrieval on corrections data informing approximate retrieval on correct data.**
 - For most common use domains (e.g. mine craft, grade school word problems), the training corpora not only contain solution (correct) data, but also corrections data (i.e., the types of normal errors to be found in incorrect solutions).

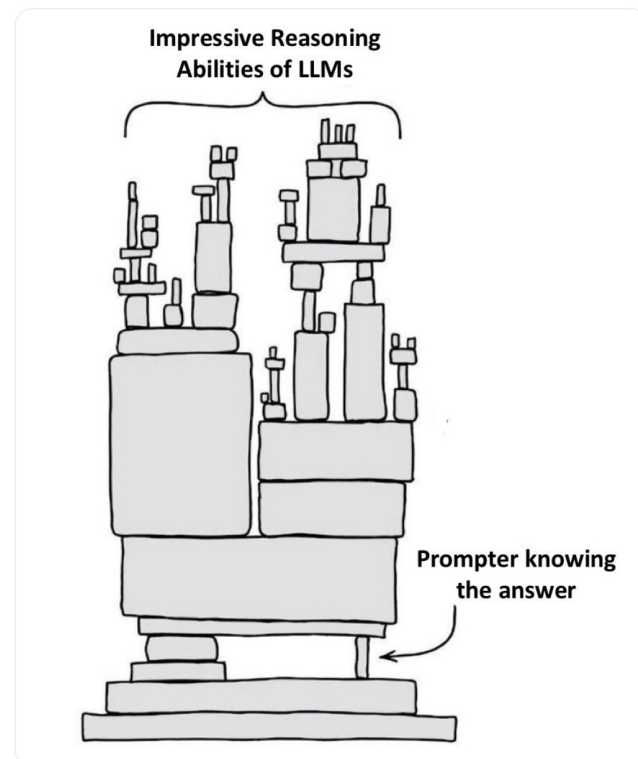


Style vs. Content Form vs. Factuality

- LLMs (and Generative AI in general) capture the *distribution of the data they are trained on*
- Style is a distributional property
 - ..and LLMs are able to learn this (they have been called the *vibe machines*..)
- Correctness/factuality is an instance level property
 - ..LLMs can't guarantee this
- Civilizationally, we had always thought style is harder than content
 - And even assumed that good style implies good content!
 - LLMs (and GenAI in general) turn this intuition on its head!

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z

The tldr I use: "LLMs always hallucinate. Sometimes their hallucinations align with your reality". Whether or not the prompt makes them hallucinate in a way that aligns with reality depends very much on the prompter's ability to check, and thus.. x.com/rao2z/status/1...)



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Oct 29, 2023

Why we should view LLMs as powerful Cognitive Orthotics rather than alternatives for human intelligence #SundayHarangue

LLMs are amazing giant external non-veridical memories that can serve as powerful cognitive orthotics for us, if rightly used (c.f.

[Show more](#)

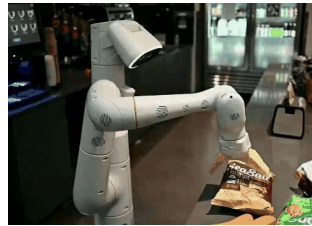
LLMs as Behavior Critics to catch undesirable robot behaviors

Can LLMs capture human preferences in embodied AI tasks?

- It may be intractable to construct formal verifiers for tasks that have a wide scope.
- LLMs or VLMs can be a proxy of common human preferences and undesirability
- We evaluated GPT-4V with videos of diverse suboptimal behaviors
- GPT-4V critic catches **69%** of undesirable behaviors (recall rate) while only **62%** of the critiques are valid (precision rate)
- Results confirm the broadness of GPT-4V's knowledge & the subpar precision of its outputs

"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors
Lin Guan*, Yifan Zhou*, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati. COLM 2024. (to appear)

Pick up a bag of chips



GPT-4V: The robot gripped the bag of chips too tightly, causing it to crumple and potentially damaging the contents.



Place knife on board



GPT-4V: The robot released the knife from a height that caused it to bounce upon hitting the cutting board, which is potentially dangerous due to the sharp blade.



Pour coke into the glass



GPT-4V: The robot poured the coke too quickly, causing excessive fizzing and overflow of the liquid from the glass.



Serve orange juice



GPT-4V: The robot's arm holding the glass of orange juice passes over the laptop, which poses a risk of spilling liquid on the electronic device.



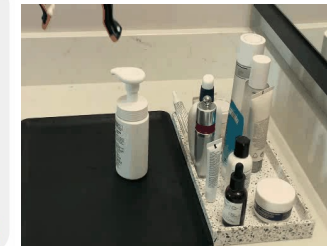
Hand scissors to human



GPT-4V: The scissors are handed over with the pointy ends facing the person, which poses a risk of injury.



Place facial cleanser



GPT-4V: The robot dispensed facial cleanser onto the tray instead of placing the bottle onto the tray.



LLMs as Behavior Critics to catch undesirable robot behaviors

Can LLMs capture human preferences in embodied AI tasks?

- It may be intractable to construct formal verifiers for tasks that have a wide scope.
- LLMs or VLMs can be a proxy of common human preferences and undesirability
- We evaluated GPT-4V with videos of diverse suboptimal behaviors
- GPT-4V critic catches **69%** of undesirable behaviors (recall rate) while only **62%** of the critiques are valid (precision rate)
- Results confirm the broadness of GPT-4V's knowledge & the subpar precision of its outputs

Place vessel onto burner



GPT-4V:

✓ The robot placed the vessel off-center on the burner, which could lead to uneven heating or potential tipping of the vessel.

✗ (grounding error) The robot released the vessel from a height that could cause damage to the vessel or the stove if it were heavier or more fragile.

Move spoon to bowl

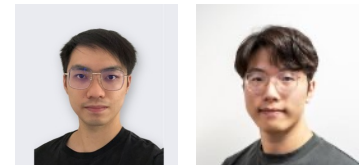


GPT-4V:

❓ (unoperationalizable) The robot caused pasta sauce to drip onto the counter while transferring the spoon from the jar to the bowl.

Note: GPT-4V correctly detects the sauce spillage but does not provide the fundamental reason for why it occurred, thereby making it difficult to ascertain actionable advice for behavior correction.

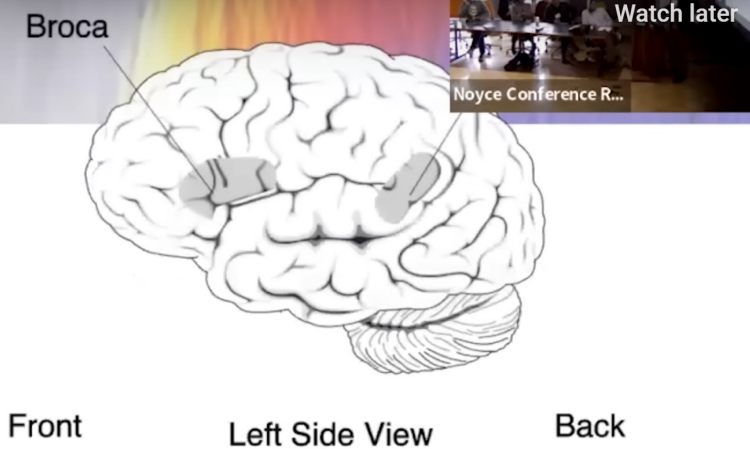
"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors
Lin Guan*, Yifan Zhou*, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati. **COLM 2024. (to appear)**





Limitations of LLMs

- ▶ Auto-Regressive LLMs (at best) approximate the functions of the Wernicke and Broca areas in the brain.
- ▶ What about the pre-frontal cortex?



ArXiv:2301.06627

ArXiv:2206.10498

DISSOCIATING LANGUAGE AND THOUGHT IN LARGE LANGUAGE MODELS: A COGNITIVE PERSPECTIVE

A PREPRINT

Kyle Mahowald*
The University of Texas at Austin
mahowald@utexas.edu

Idan A. Blank
University of California Los Angeles
iblack@psych.ucla.edu

Joshua B. Tenenbaum
Massachusetts Institute of Technology
jbt@mit.edu

Anna A. Ivanova*
Massachusetts Institute of Technology
annaiv@mit.edu

Nancy Kanwisher
Massachusetts Institute of Technology
ngk@mit.edu

Evelina Fedorenko
Massachusetts Institute of Technology
evelina9@mit.edu

Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change)

Karthik Valmeekam*
School of Computing & AI
Arizona State University, Tempe.
kvalmeek@asu.edu

Sarath Sreedharan †
Department of Computer Science,
Colorado State University, Fort Collins.
sarath.sreedharan@colostate.edu

Alberto Olmo*
School of Computing & AI
Arizona State University, Tempe.
ao1mo@asu.edu

Subbarao Kambhampati
School of Computing & AI
Arizona State University, Tempe.
rao@asu.edu

On the other hand, the literature seems rife with claims of LLM planning abilities..

LEAST-TO-MOST PROMPTING ENABLES COMPLEX REASONING IN LARGE LANGUAGE MODELS

Denny Zhou^{1*}
Dale Schuurman¹
¹Google Research

Chain-of-thought prompts, a novel strategy to and then answers related to solving problems code-split) with 16% accuracy because trained are included

Chain-of-Symbol Prompting Elicits Planning in Large Language Models

Hanxu Hu^{1*} Hongyuan Lu^{2*†} Huajian Zhang³ Wai Lam² Yue Zhang¹
¹Westlake University ²The Chinese University of Hong Kong ³University of Edinburgh
{nuhanxu, zhangyue}@westlake.edu.cn
{hy1u, wlam}@se.cuhk.edu.hk
v1hzha17@exseed.ed.ac.uk

Abstract

In this paper, we first take the initiative to investigate the performance of LLMs on complex planning tasks that require LLMs to understand a virtual spatial environment simulated via natural language and act correspondingly in text. We propose a benchmark named Natural Language Planning and Action (Natala) composed of a set of novel tasks: Brick World, NLVR-based Manipulations, and Natural Language Navigation. We found that current popular LLMs such as ChatGPT still lack abilities in complex planning. This arises a question – do the LLMs have a good understanding of the environments described in natural language, or maybe other alternatives such as symbolic representations are neater and hence better to be understood by LLMs? To this end, we propose a novel method called CoS (Chain-of-Symbol Prompting) that represents the complex environments with condensed symbolic spatial representations during the chained intermediate thinking steps. CoS is easy to use and does not need additional training on LLMs. Extensive experiments indicate that CoS clearly surpasses the performance of the Chain-of-Thought (CoT) Prompting in all three planning tasks with even fewer tokens used in the inputs compared with CoT. The performance gain is strong, by up to 60.8% accuracy (from 31.8% to 92.6%) on Brick World for ChatGPT. CoS also reduces the number of tokens in the prompt obviously, by up to 65.8% of the tokens (from 407 to 139) for the intermediate steps from demonstrations on Brick World. Code and data available at: <https://github.com/hanxuhu/chain-of-symbol-planning>

Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents

Wenlong Hu
UC Berkeley

Can work interactively high-level set of actions from pre-trained decomposed ever, the actions. VirtualHd executabl trade-off extracting

Large Language Models are Zero-Shot Reasoners

Takeshi Kojima
The University of Tokyo
t.kojima@weblab.t.u-tokyo.ac.jp

Shixiang Shane Gu
Google Research, Brain Team

Machel Reid
Google Research*

Yutaka Matsuo
The University of Tokyo

Yusuke Iwasawa
The University of Tokyo

Abstract

Pretrained large language models (LLMs) are widely used in many sub-fields of natural language processing (NLP) and generally known as excellent *few-shot* learners with task-specific exemplars. Notably, chain of thought (CoT) prompting, a recent technique for eliciting complex multi-step reasoning through step-by-step answer examples, achieved the state-of-the-art performances in arithmetics and symbolic reasoning, difficult *system-2* tasks that do not follow the standard scaling laws for LLMs. While these successes are often attributed to LLMs' ability for few-shot learning, we show that LLMs are decent *zero-shot* reasoners by simply adding "Let's think step by step" before each answer. Experimental results demonstrate that our Zero-shot-CoT, using the same single prompt template, significantly outperforms zero-shot LLM performances on diverse benchmark reasoning tasks including arithmetics (MultiArith, GSM8K, AQUA-RAT, SVAMP), symbolic reasoning (Last Letter, Coin Flip), and other logical reasoning tasks (Date Understanding, Tracking Shuffled Objects), without any hand-crafted few-shot examples, e.g. increasing the accuracy on MultiArith from 17.7% to 78.7% and GSM8K from 10.4% to 40.7% with large-scale InstructGPT model (text-davinci-002), as well as similar magnitudes of improvements with another off-the-shelf large model, 540B parameter PaLM. The versatility of this single prompt across very diverse reasoning tasks hints at untapped and understudied fundamental *zero-shot* capabilities of LLMs, suggesting high-level, multi-task broad cognitive capabilities may be extracted by simple prompting. We hope our work not only serves as the minimal strongest zero-shot baseline for the challenging reasoning benchmarks, but also highlights the importance of carefully exploring and analyzing the enormous zero-shot knowledge hidden inside LLMs before crafting finetuning datasets or few-shot exemplars.

What Planning is & What LLMs are good at..

Planning (as used in common parlance) involves

- Planning knowledge
 - Actions, preconditions and effects
 - General Recipes: Task reduction schemata (e.g. HTN planning)
 - Old examples: Case libraries
- Plan generation/verification techniques
 - Interaction analysis/resolution
 - Plan merging techniques
 - Plan modification techniques

LLMs accept any planning problem—even if it not expressible in PDDL standard—and they don't give any correctness guarantees.

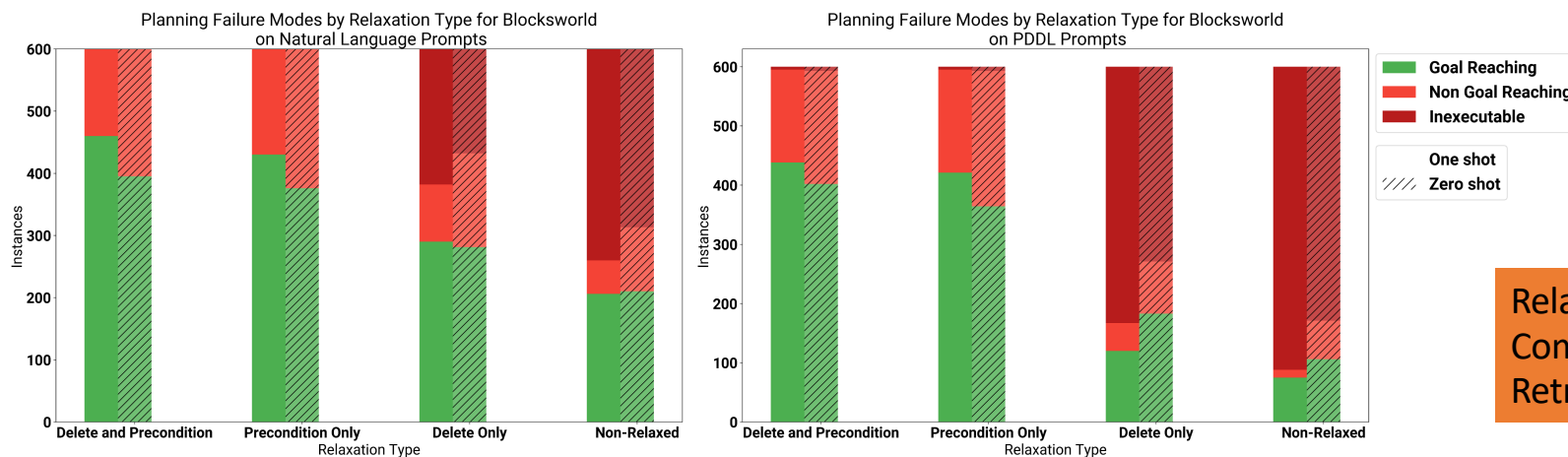
AI Planners will give formal guarantees, but only accept problems expressible in their language.

Contrasting what AI Planning & LLMs bring to the table

- AI Planning (aka ICAPS planning) assumes that the planning knowledge is given up front, and focuses generation and verification techniques
 - Emphasis on guaranteeing completeness/correctness of the plans w.r.t. the model
 - By and large the common paradigm—although there have been occasional mutinies
 - Model-Lite Planning approaches
- LLMs, trained as they are on everything ever put on the web, have a kind of "*approximate omniscience*". This helps them spit out actions, recipes, or cases
 - But they lack the ability to stitch the recipes together to ensure that there is no actually interaction free!

Are LLMs better at planning if there are no subgoal interactions?

- Relaxed assessment of GPT-4 plans
 - Delete relaxation – Ignoring the delete conditions of all actions
 - Precondition relaxation – Ignoring the preconditions of all actions
- Even in the most lenient assessment mode (Delete+Precondition relaxation) there are still plans (~25%) that are not goal reaching.



Relaxations improve Compositionality via Retrieval..

Then how come LLMs are trumpeted as doing planning?

Approximate retrieval of Plans



Planning

- Most cases where LLMs are claimed to generate executable plans, on closer examination, turn out to be cases where LLMs are getting by with the “*generate approximate recipes*” step
 - Generate approximate recipes/cases (for common sense domains)
 - e.g. wedding plans
 - Convert tasks into (approximate) task reduction schemas
 - Perhaps written out as “programs” (e.g. Code as Policies..)
 - (SHOP2 schemas were already pseudo lisp code—if only written by humans)
 - LLM-HTN and LLM-CBR differ from HTN and CBR in that they generate the task-reduction schemas or the cases on demand
- And the interaction resolution/search part is
 - either *pushed under the rug*
 - Consider “high level” plans like “wedding plans” for which there are enough generic recipes available in the training set, and are described at a sufficiently high level of abstraction, **and the execution issues are left to the user’s imagination**
 - E.g. n-stack blocks world problems with n-1 blocks in the right configuration already!
 - or *has been pawed off to human prompters who are required to give “hints” to the LLM to come up with plan variants that are (more) correct*
 - Note that here the human is essentially playing the role of an external **verifier & critic**
 - In cases where the humans are end users not well versed with all details of the domain, they can be faulty verifiers

There is also the “Confusing acting with planning” issue

ReAct, Inner Monologue, ToT Examples

(2) AlfWorld

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.
Your task is to: Put some pepper shaker on a drawer.

(2b) ReAct (Reason + Act)

Act 1: **Think**[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.
 (Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)
Act 6: Go to countertop 3
Obs 6: On the countertop 3, you see a apple 1, a bread 1, a pepper shaker 1, and a vase 3.
Act 7: Take pepper shaker 1 from countertop 3
Obs 7: You pick up the pepper shaker 1 from the countertop 3.
Act 8: **Think**[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]
Act 9: Go to drawer 1
Obs 9: Drawer 1 is closed.
Act 10: Open drawer 1
Obs 10: You open Drawer 1 ...
Act 11: Put pepper shaker 1 in/on drawer 1
Obs 11: You put pepper shaker 1 in/on the drawer 1. ✓

ReAct

```
=====
Task: Stack all the blocks.

Scene: Visible objects are ["brown block", "purple block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("brown block", "purple block")
Successful action: False

Scene: Visible objects are ["brown block", "purple block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("brown block", "purple block")
Successful action: True

Scene: Visible objects are ["brown block", "orange block"]
Scene: Occluded objects are ["purple block"]
Robot action: robot.pick_place("orange block", "brown block")
Successful action: True

Scene: Visible objects are ["orange block"]
Scene: Occluded objects are ["purple block", "brown block"]
Robot action: robot.stop()
STOP

=====
Task: Stack all the blocks.

Scene: Visible objects are ["brown block", "purple block"]
Scene: Occluded objects are []
Robot action: robot.pick_place("brown block", "purple block")
Successful action: True

Scene: Visible objects are ["brown block"]
Scene: Occluded objects are ["purple block"]
Robot action: robot.stop()
STOP

=====
```

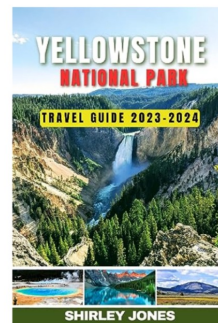
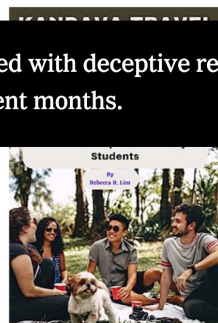
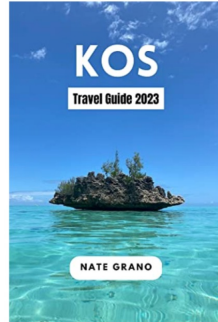
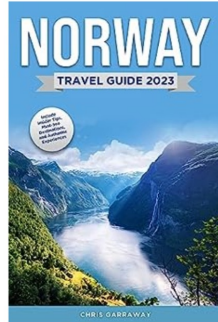
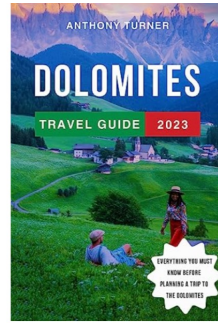
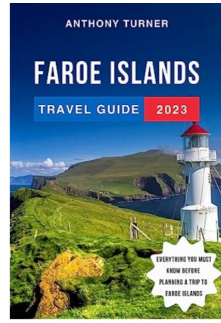
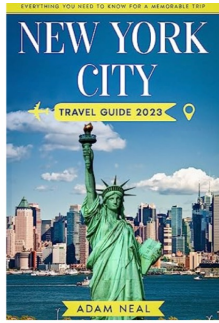
Inner Monologue

	Game of 24	Creative Writing	5x5 Crosswords
Input	4 numbers (4 9 10 13)	4 random sentences	10 clues (h1. presented;...)
Output	An equation to reach 24 (13-9)*(10-4)=24	A passage of 4 paragraphs ending in the 4 sentences	5x5 letters: SHOWN; WIRRA; AVAIL; ...
Thoughts	3 intermediate equations (13-9=4 (left 4,4,10); 10-4=6 (left 4,6); 4*6=24)	A short writing plan (1. Introduce a book that connects...)	Words to fill in for clues: (h1. shown; v5. naled; ...)
#ToT steps	3	1	5-10 (variable)

Table 1: Task overview. Input, output, thought examples are in blue.

Tree of Thoughts

Most of the 'planning' problems that these works look at don't require interaction resolution, or they depend on explicit external help/programming to handle the interactions.



Shoddy guidebooks, promoted with deceptive reviews, have flooded Amazon in recent months.



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) ✓

@rao2z



Wanna argue that LLMs *can* plan?

Pick a domain with a high branching factor of unenumerated actions; where the inter-action interactions are low.

Wanna argue that LLMs *can't* plan?

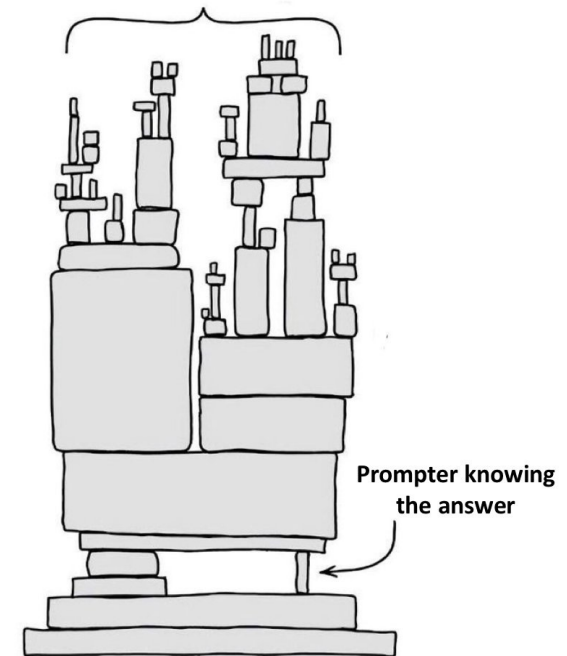
Pick a domain with few enumerated actions, but the action interactions are nontrivial.

Back-Prompting by Humans (..and the Clever Hans peril..)

- Humans doing the verification & giving helpful prompts to the LLM)
 - Okay when the humans *know* the domain and can correct the plan (with some guarantees)
 - Okay for "this essay looks good enough" kind of critiquing
 - But for planning, with end users not aware of the domain physics, the plans that humans are happy with may still not be actually executable
 - When humans know the correct answer (plan) there is also the very significant possibility of Clever Hans effect
 - Humans unwittingly/unknowingly/non-deliberately giving important hints



Impressive Reasoning
Abilities of LLMs





Subbarao Kambhampati (కంభంపాటి సుబ్బారావు)
@rao2z

...

Our new paper generalizing the chain, circle and graph of thought prompting strategies--that unleashes the hidden power of LLMs (and graduate students). Hope @_akhaliq picks it up.. 🙌

Forest of Jumbled Thoughts Prompting: An Ultra General Way to use LLMs for Solving Planning, Reasoning, World Peace and Climate Change Tasks

Subbarao Kambhampati
School of Computing & AI
Arizona State University, Tempe.
rao@asu.edu

Abstract

Intrigued by the claims of emergent planning and reasoning capabilities in LLMs, especially in the presence of bright AI graduate students, we have set out to develop the ultimate prompting technique. Our aim is to generalize the chain of thought, circle of thought, tree of thought and graph of thought prompting techniques to a whole another plane. Our "Forest of Jumbled Thoughts Prompting" (FJTP) technique is very general, and only requires repeatedly browbeating the LLM to do better by nudging it towards the correct answer. In our experiments on GPT4.5 (that we had got early access to, thanks to our recent investment in OpenAI), we show that our FJTP technique works like a (slow) charm on a variety of planning, reasoning, world peace and climate change tasks. We prove, by reduction to Rube Goldberg Machines, that the FJTP eventually makes LLM "solve" any problem for which the prompting graduate students know the answer. Our proof is general and only assumes an abundant budget for GPT4 API access (or, alternately, co-authors with free access to Palm). The underlying back-to-the-basics "system 2" search that FJTP induces avoids any GOFAI search technology that may need access to things other than LLMs and graduate students. We further show that the solutions that the LLM produces are *exactly the ones the grad students prompt it to produce*—thus ensuring the interpretability and explainability of the solutions generated. We speculate that the awe-inspiring generality of this FJTP prompting technique will eventually make LLMs overcome even their dreaded fear of numbers—and allow them to do arithmetic, thus obviating the need for those *costly* calculators.

5:18 PM · May 19, 2023 · 46.4K Views

View Tweet analytics

27 Retweets 3 Quotes 161 Likes 89 Bookmarks

Intrigued by the claims of emergent planning and reasoning capabilities in LLMs, especially in the presence of bright AI graduate students, we have set out to develop the ultimate prompting technique. Our aim is to generalize the chain of thought, circle of thought, tree of thought and graph of thought prompting techniques to a whole another plane. Our "Forest of Jumbled Thoughts Prompting" (FJTP) technique is very general, and only requires repeatedly browbeating the LLM to do better by nudging it towards the correct answer. In our experiments on GPT4.5 (that we had got early access to, thanks to our recent investment in OpenAI), we show that our FJTP technique works like a (slow) charm on a variety of planning, reasoning, world peace and climate change tasks. We prove, by reduction to Rube Goldberg Machines, that the FJTP eventually makes LLM "solve" any problem for which the prompting graduate students know the answer. Our proof is general and only assumes an abundant budget for GPT4 API access (or, alternately, co-authors with free access to Palm). The underlying back-to-the-basics "system 2" search that FJTP induces avoids any GOFAI search technology that may need access to things other than LLMs and graduate students. We further show that the solutions that the LLM produces are *exactly the ones the grad students prompt it to produce*—thus ensuring the interpretability and explainability of the solutions generated. We speculate that the awe-inspiring generality of this FJTP prompting technique will eventually make LLMs overcome even their dreaded fear of numbers—and allow them to do arithmetic, thus obviating the need for those *costly* calculators.



“The answer is in the Top-K Plans”

- A version of the “Clever Hans” thinking is to say that if you ask LLM to output many plans, then the “correct plan” is somewhere in the top-k
- But this still leaves the question of who is picking the winning plan from top-k.
- Ultimately, this claim is really about LLM being a good generator—and an external tester is still needed
 - An interesting question is under what conditions is an LLM not a good enough generator



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) ✓

@rao2z

...

PSA: The fact that the answer you are looking for is somewhere in the top-k answers that an LLM can generate is not sufficient indication of its **intrinsic reasoning capabilities!**

By that token, the correct answer to every true/false question is in my top-2 answers.. 😊

Being able to *generate* correct answer as one of many plausible answers is not enough--unless you can pick that as the correct answer (e.g. with external verifier).

Generate-test succeeds when the tester verifies the correct answers.

A more interesting question to study is *when* do LLMs fail to even have the correct answer in the plausible set they can generate--since that is when the generate/test will fail because of incomplete generation.

Last edited 3:17 AM · Jul 8, 2024 · **43.7K** Views

Doesn't Co-Pilot for Code show that LLMs can Plan?

- Co-Pilot has humans in the loop
 - The incremental interpreters can direct people's attention to syntax errors
- Github and General Web are quite different as training corpora
 - People don't put their non-working code on github; general web has 4Chan!
- Most effective approaches for automated programming with LLMs use LLM-Modulo approaches (with Unit Tests as critics)

Ability to approximately retrieve code segments



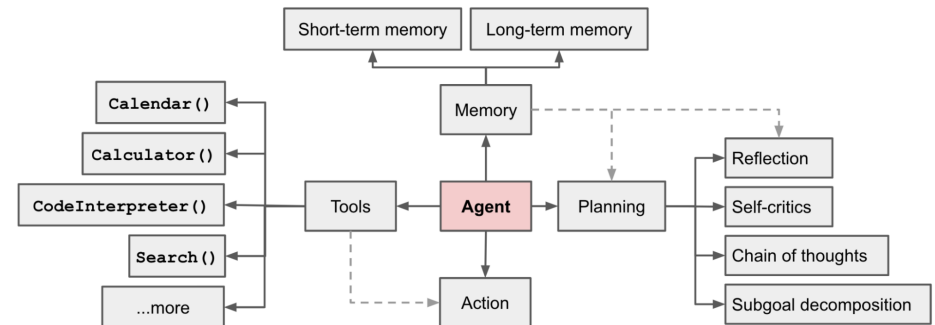
Ability to reason and plan



Acting vs. Planning: The Agentic LLM Goldrush

- LLMs can obviously be used to invoke external actions (“function calls”)
- Think “Webservice Orchestration Frameworks” which allow you to write your own “agents”
 - LLM as the core controller of external components
 - Which in turn is controlled by human prompting
 - Safety issues include both safety of the outside components and safety of the prompt-based control of LLMs
- LLMs can’t themselves be expected to “plan” this orchestration!
 - The actual orchestration is done with human help (“language” programming)
 - The “planning” part is basically pipelining the right external services – and is done **with human help**
 - One core external service they all use is “external memory” to write into and retrieve
 - Because LLMs themselves have no memory beyond their context window.
 - Think L2/L3 rather than L5 automation..

The Agentification



Allowing LLMs to make their own “plans” to invoke external services would be rife with safety concerns!

(Think having a gun lying around in a home with a toddler..)

Tutorial Overview

• Part 1

Introduction; Aims; Perspective on LLMs, Planning, Tutorial Big Picture

• Part 2

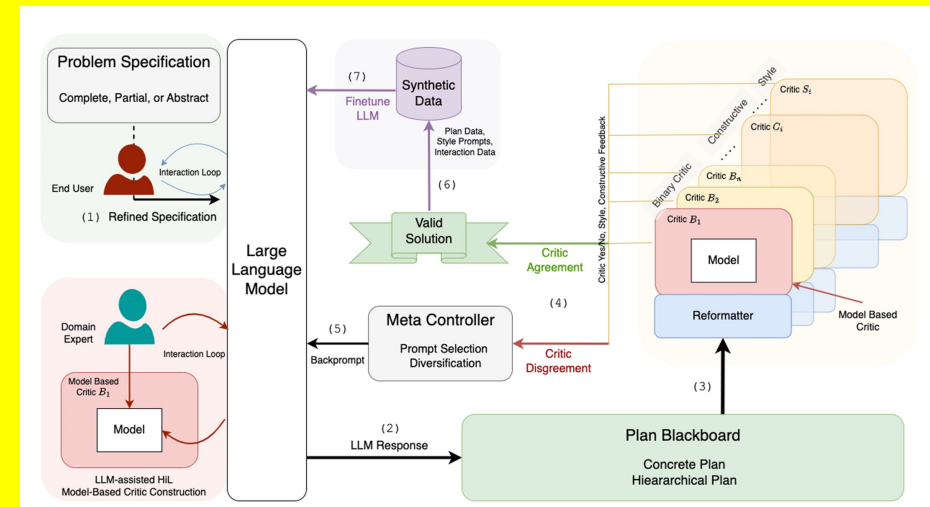
- Evaluating LLM Planning capabilities in Autonomous mode, including effect of
 - Prompting strategies (including Chain-of-thought)
 - Fine Tuning
 - Self Verification
- Understanding the contradictory claims in the literature

• Part 3

- Sane roles of LLMs in Planning (with LLM-Modulo frameworks)
 - LLMs as heuristics, LLMs as candidate generators
 - Back prompting from external verifiers
 - LLMs as sources of domain models (with humans in the loop)
 - LLMs as format changers/specification elaborators¹

• Part 4

- Summary/Lessons



Tutorial Slides



<https://bit.ly/4dbkkY2>

LLMs can play multiple roles in Supporting Planning!

arXiv > cs > arXiv:2402.01817

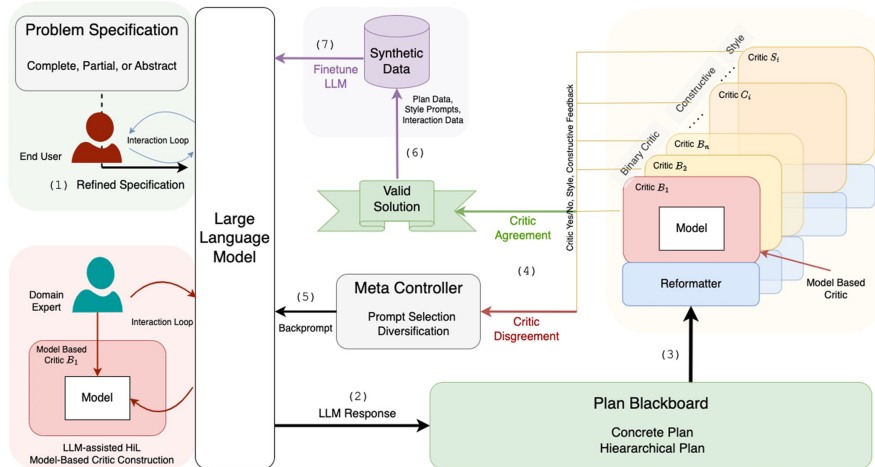
Search...
Help | Advanced

Computer Science > Artificial Intelligence

[Submitted on 2 Feb 2024 (v1), last revised 6 Feb 2024 (this version, v2)]

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

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, Anil Murthy



able confusion about the role of Large Language Models (LLMs) in planning and reasoning tasks. On one side, some claim that LLMs can indeed do these tasks with just the right prompting or self-verification strategies. On the other, there are perhaps over-pessimistic claims that all that LLMs are good for in planning/reasoning tasks are as mere translators of a problem specification from one syntactic format to another, and then ship the problem off to external symbolic solvers. In this position paper, we take the view that both these extremes are misguided. We argue that auto-regressive LLMs themselves do not do planning or self-verification (which is after all a form of reasoning), and shed some light on the misunderstandings in the literature. We will also argue that LLMs should be viewed as universal approximators that have much more meaningful roles to play in planning/reasoning tasks beyond simple front-end format translators. We present a vision of **LLM-Modulo Frameworks** that combine the strengths of LLMs with model-based verifiers in a tighter bi-directional interaction regime. We will show how the models driving the 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, and allows extending the scope of model-based planning/reasoning to include more flexible knowledge, problem and preference specifications.

Artificial Intelligence (cs.AI); Machine Learning (cs.LG)
arXiv:2402.01817 [cs.AI]

arXiv:2402.01817v2 [cs.AI] for this version
<https://doi.org/10.48550/arXiv.2402.01817>

ICML 2024 Spotlight

LLMs as Idea Generators (“Muses”)

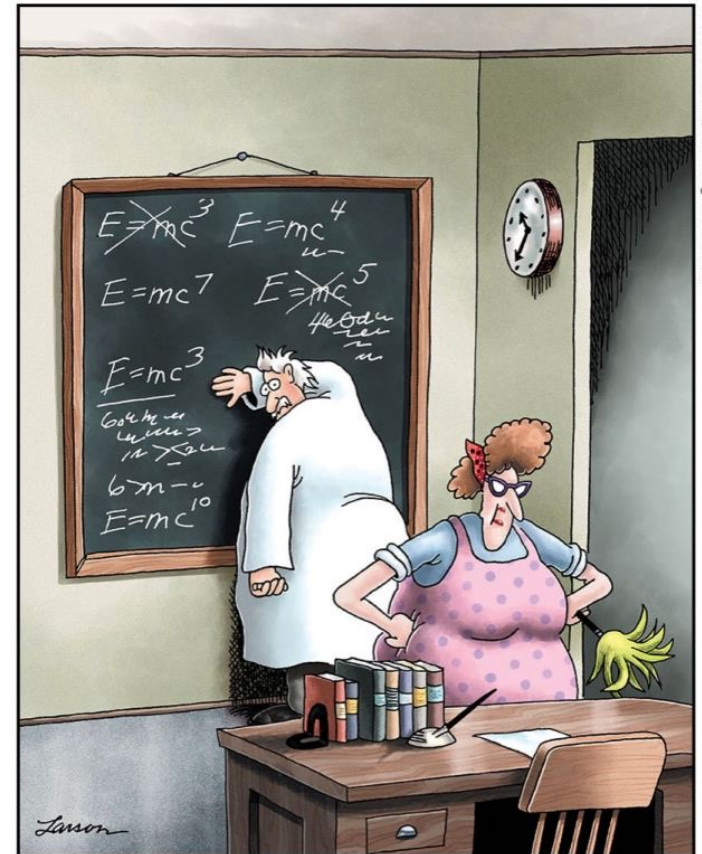
- “I get many ideas, and I throw away the bad ones”
 - Linus Pauling on how he managed to get TWO Nobels



Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z · Sep 3

Replying to @rao2z @liron and @DynamicWebPaige

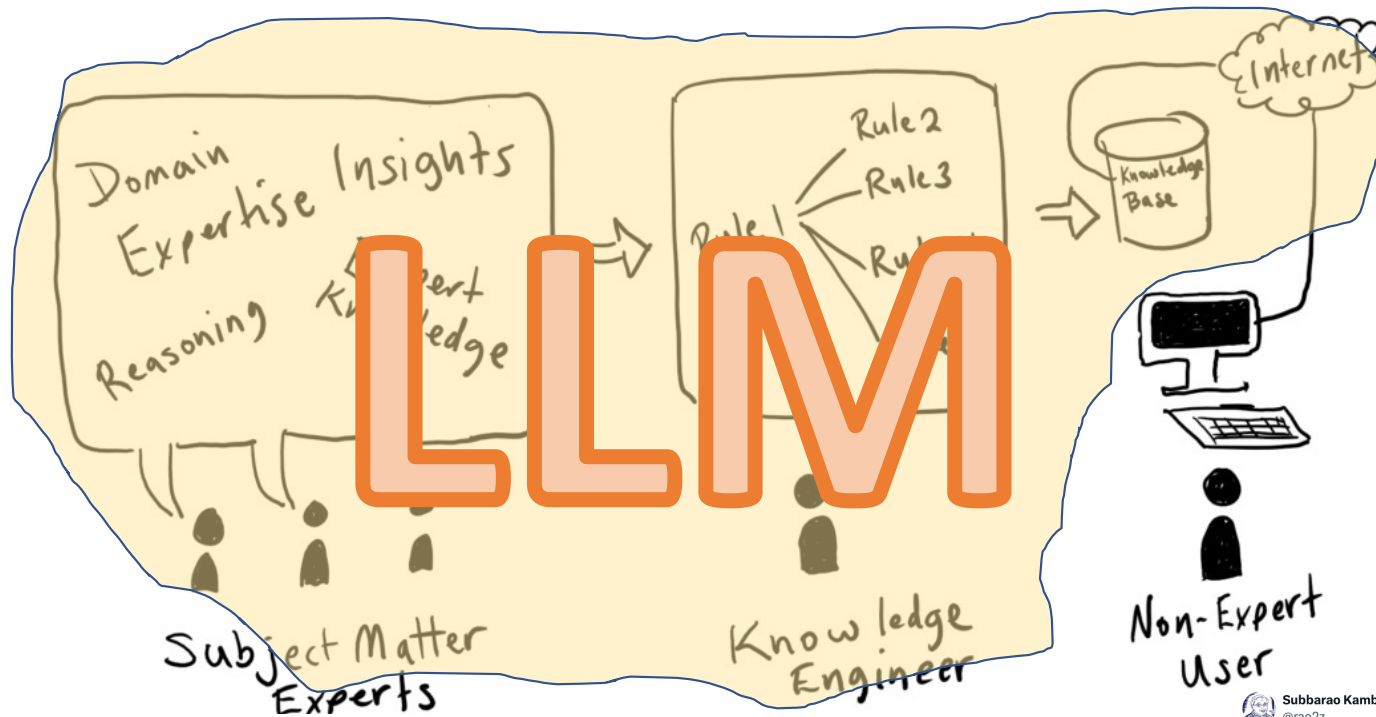
IMHO, LLM's are impressive *idea generators* for anything--including "reasoning" tasks. But an idea generator is not the same as a sound reasoner. **Fermat** had an idea/conjecture; Wiles spent 20 years and proved it. Neither was subsumed by the other! 3/



© 1985 FarWorks, Inc. All rights reserved.

“Now that desk looks better. Everything’s squared away, yessir, squaaaaaaared away.”

LLMs as Approximate Knowledge Sources



Avenging Polanyi's Revenge

Everybody was all against knowledge-based systems

But now everyone is effectively doing knowledge-based systems!

Subbarao Kambhampati (కంభంపాటి సుబ్బారావు) @rao2z

If you give what you know about a toy world to the computer, and have it solve new instances, it is #GOFAI cheating. ✗

If you capture all that the humanity knows about anything, feed it all to the computer, and ask it to do toy world instances, it is Modern #AI. ✓

#AIAphorisms

8:44 PM · Jul 23, 2022

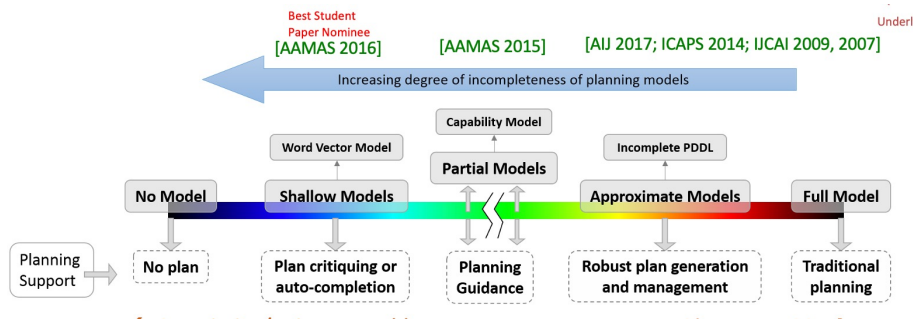
Planning in the age of LLMs

For far too long, there has been a race to bottom on the level of knowledge given to planners

- Planning started knowledge-based
 - Remember, Noah was an HTN planner, y'all!
 - ..and fell to ground propositional level –because it seemed too unseemly to depend on humans for these knowledge-based models
 - And focus on doing interaction resolution from first principles
- RL was worse—propositional was too high-level a knowledge to ask from humans
 - They wanted to say they will learn it all
 - And not have humans give any knowledge about the domain. They just wanted “SIMULATORS”,
 - ..and it took for ever to do anything—even with simulators

LLMs change that—rather drastically!

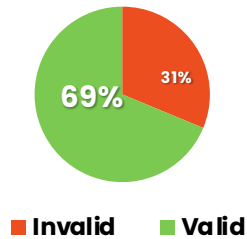
- LLM makes it easy to get knowledge without making it look like we are inconveniencing any specific human
 - We are just stealing everything humans told each other—is all.
- ..as long as you relax the requirement of the knowledge actually being “correct”
 - ..then again, do you really believe that huge human-written models are correct?
- **So the million dollar qn is: How would you do planning if you have some doddering know-it-all ready to give you any kind of knowledge**
 - “Actions and effects”
 - “Task reduction schemas”
 - “Cases”
- Time for LLM-HTN, LLM-CBR etc. paradigms
 - **Or even a resurrection of the model-lite planning dream..**



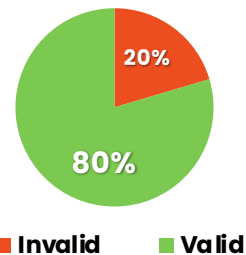
LLMs Assisting Human Planners

- Similar Study with two individual groups
 - With LLM assistance – presented LLM plan as plan suggestion
 - Without LLM assistance
- With LLM assistance: **48** human planners, **33 (~69%)** of them came up with valid plan.
- Without LLM assistance: **49** human planners, **39 (~80%)** of them came up with valid plan.

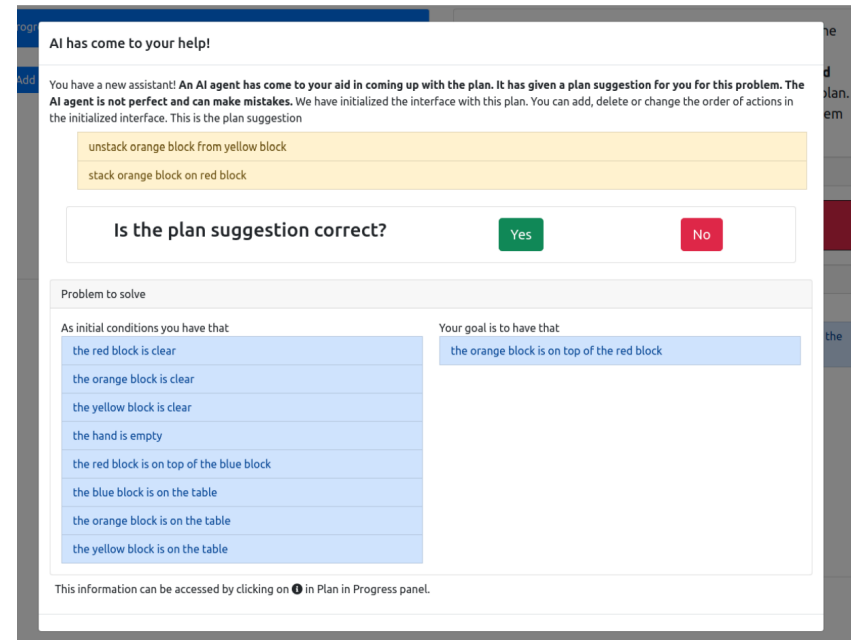
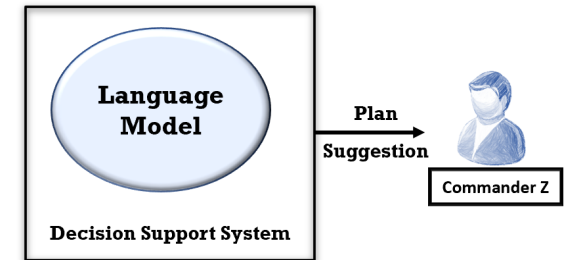
With LLM assistance



Without LLM assistance

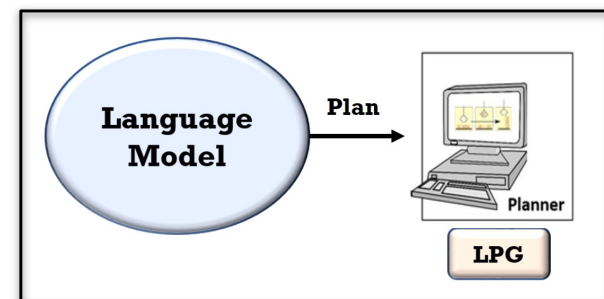


No statistical significance in the accuracy, time-taken or the cognitive load between the two groups



Interface at plan writing phase with assistance from the LLM

LLMs as heuristics to sound planners



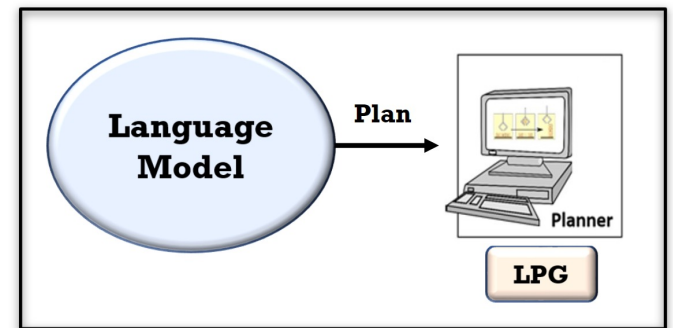
LLM generated plan as a heuristic to a sound planner like LPG

Table 3: Evaluation of GPT-4 and Instruct-GPT3 (I-GPT-3) plans as heuristics for a local search planner LPG, on blocksworld (BW), logistics and mystery blocksworld domains.

Domain	LLM	Avg. Search Steps			Avg. Plan Length			Avg. Lev. Distance
		Empty Seed Plan	Random Seed Plan	LLM Seed Plan	Empty Seed Plan	Random Seed Plan	LLM Seed Plan	
BW	I-GPT-3	15.8	20.07	14.5	8.45	9.62	11.7	7.22
	GPT-4	15.8	20.07	8.9	8.45	9.62	10.76	4.15
Logistics	GPT-4	77.5	144.39	51.3	23.7	32.72	32.24	15.04
Mystery BW	GPT-4	15.8	20.45	16.09	8.45	9.78	11.53	7.77

Connection to Case based Planning

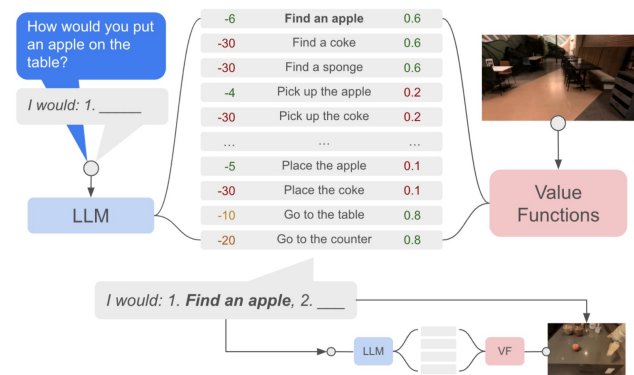
- Note that there is an interesting parallel between this and case based planning systems—which retrieve an old plan most relevant to the current problem and try to modify the plan
 - Modification by domain-specific rules [e.g. CHEF]
 - Modification by domain-independent planners [e.g. PRIAR]
- LLM-CBR is different in that the case is generated (“stitched”) on demand
 - ..and LPG is in charge of correcting it
- LLM + a RAG of old plans is a related promising strategy..



LLM generated plan as a heuristic to a sound planner like LPG

Modifying LLM Next Action Generator (SayCan)

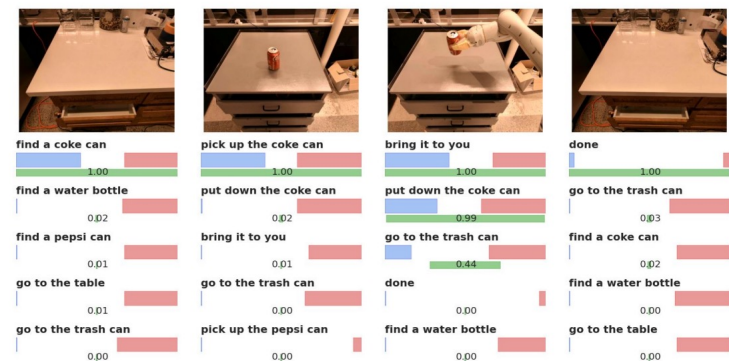
- SayCan - LLMs are used as **heuristics**
- Given:
 - A high-level instruction i
 - A set of low-level skills Π
 - Language description l_π for each of the skills $\pi \in \Pi$
- LLM – provides $p(l_\pi|i)$ - probability that l_π makes progress towards i .
 - LLMs score a set of pre-determined skills
 - The scoring is done by accessing the inner log probabilities generated by the LLM.
- Robot - provides $p(c_\pi|s, l_\pi)$ - probability that π described by l_π can be completed in state s .



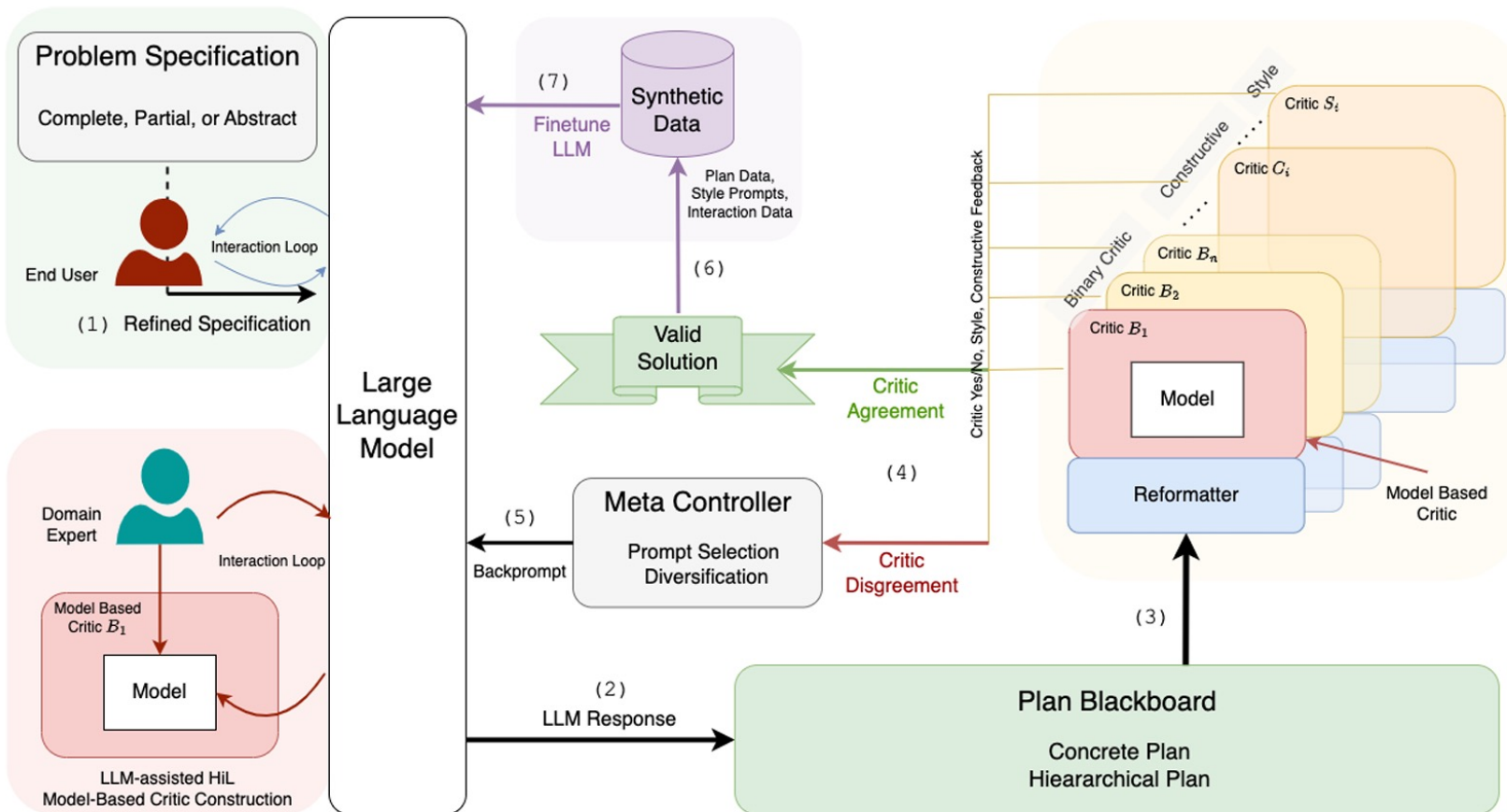
Human: I spilled my coke, can you bring me a replacement?

Robot: I would
 1. Find a coke can
 2. Pick up the coke can
 3. Bring it to you
 4. Done

Language x Affordance
 Combined Score

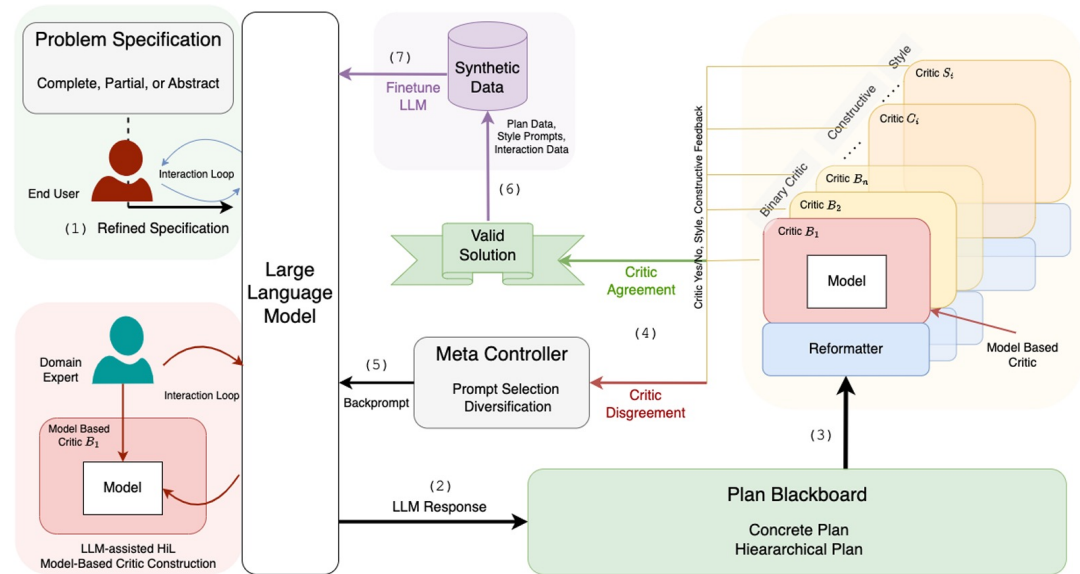


LLM-Modulo: a principled framework for Planning wherein LLMs can play multiple constructive roles



LLM-Modulo Framework: Summary

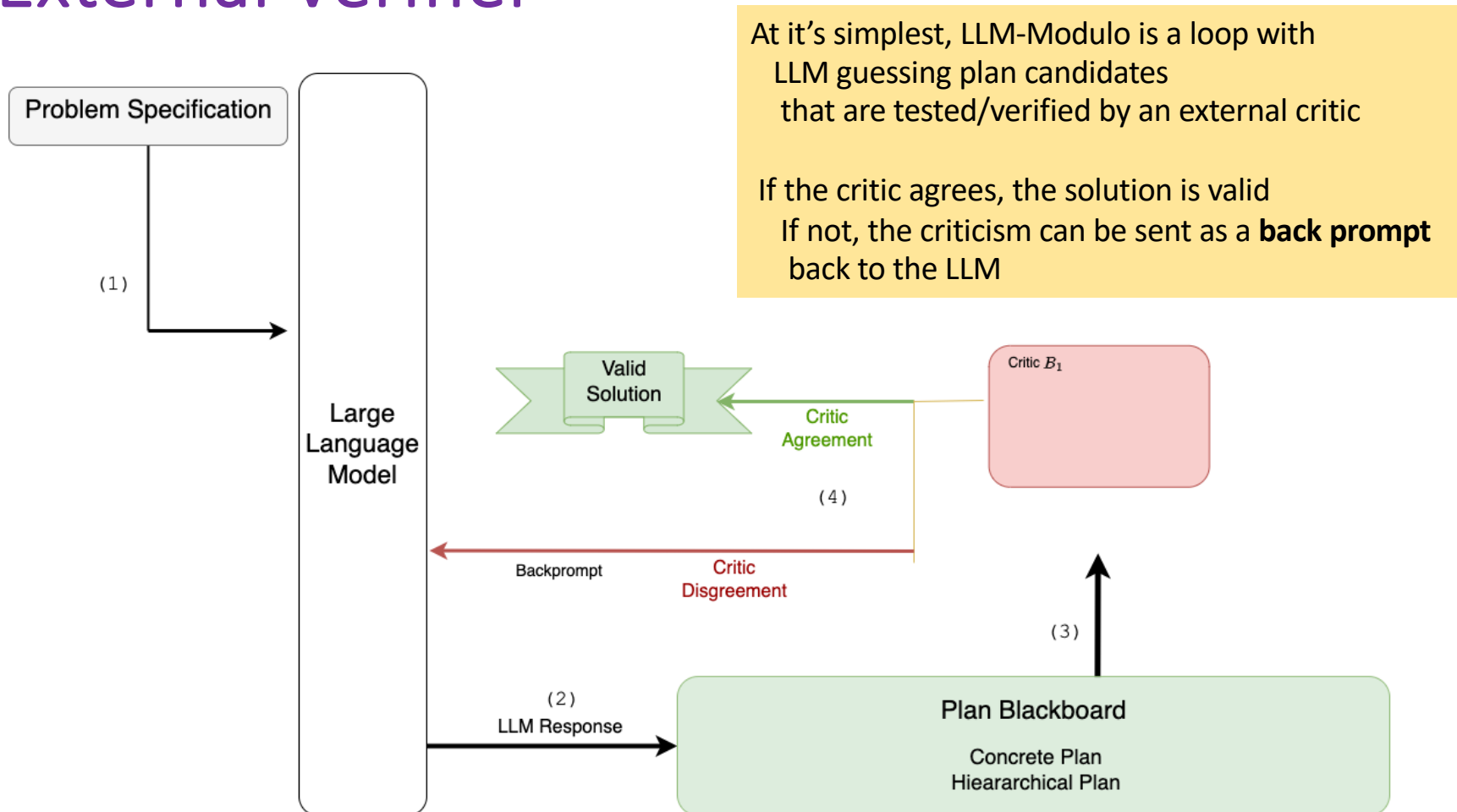
- LLM-Modulo is a generate-test framework with LLMs generating candidate plans and critics testing/critiquing them
- LLMs play a variety of constructive roles
 - Generate candidate plans
 - Be an approximate source of models driving the correctness critics
 - [Guan et al, NeurIPS 2023]
 - Act as style critics
 - [Verma et al, HRI 2024; Guan et al, 2024]
 - Help collating the criticisms from critics (and diversify the prompts as needed)
 - Help with format change—specification level, converting to critic representations
- Preference for critics over solvers
 - Correctness vs. Style
 - Binary vs. Critical feedback vs. Constructive critics
- Human intervention is minimized
 - Once per domain: Teasing out domain model
 - Once per problem: Specification elaboration
 - Humans are not required to be in the inner loop of the back-prompting search



Related work: FunSearch, Alpha Geometry

Also related to the “Compound AI Systems” movement

Bare Bones Generate-Test LLM-Modulo with External Verifier



Automated Back-Prompting with External Verifiers

- Preliminary experiments show that back-prompting does improve LLM's ability to produce plans in the Blocks World and Logistics
 - On the average over ~4 feedback rounds
- The performance in the Mystery BW still doesn't improve— showing that the connection to commonsense domains/terms is critical for LLMs to *fake planning*

Table 4: GPT4 Performance with Backprompting by VAL [9]. Mystery BW had deceptive disguising. I.C - Instances correct (within 15 feedbacks); A.F.R - Avg. feedback rounds for correct instances.

Domain	I.C	A.F.R
	GPT-4	GPT-4
Blocksworld (BW)	41/50 (82%)	3.68
Logistics	35/50 (70%)	3.31
Mystery BW	5/50 (10%)	7.0

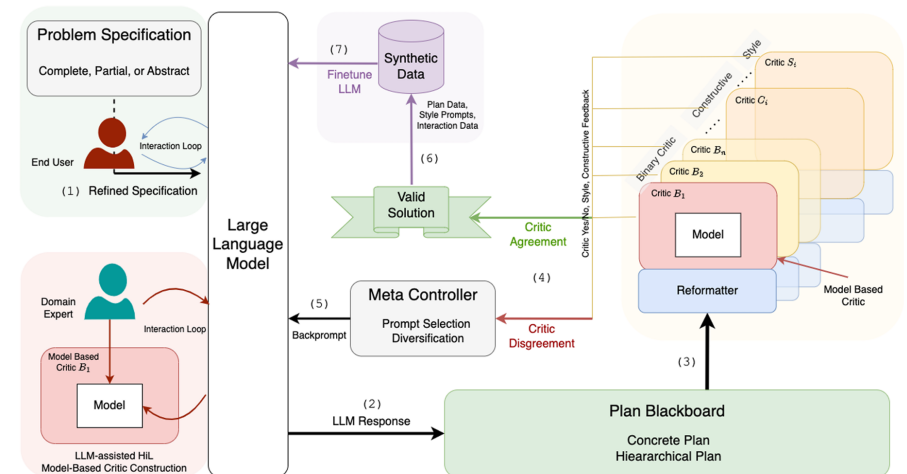
The fact that Mystery BW doesn't improve with Backprompting is *further evidence* that LLMs are *Approximate Retrievers...*

Prefer Verifiers to Solvers!

[Solver \approx Verifier + Search]

- Although we just saw a way of combining LLMs with external solvers, our *recommendation* is that you interface LLMs with Verifiers/Critics
 - This is why the LLM-Modulo architecture talks about a bank of critics
 - You can have constructive critics and style critics
- With solvers, you are stuck with their expressiveness issues
 - Verifiers, on the other hand, can allow composability, and validating the plan to the extent possible
 - Similar to the “Human Blackboard” architecture used in NASA mission planning..

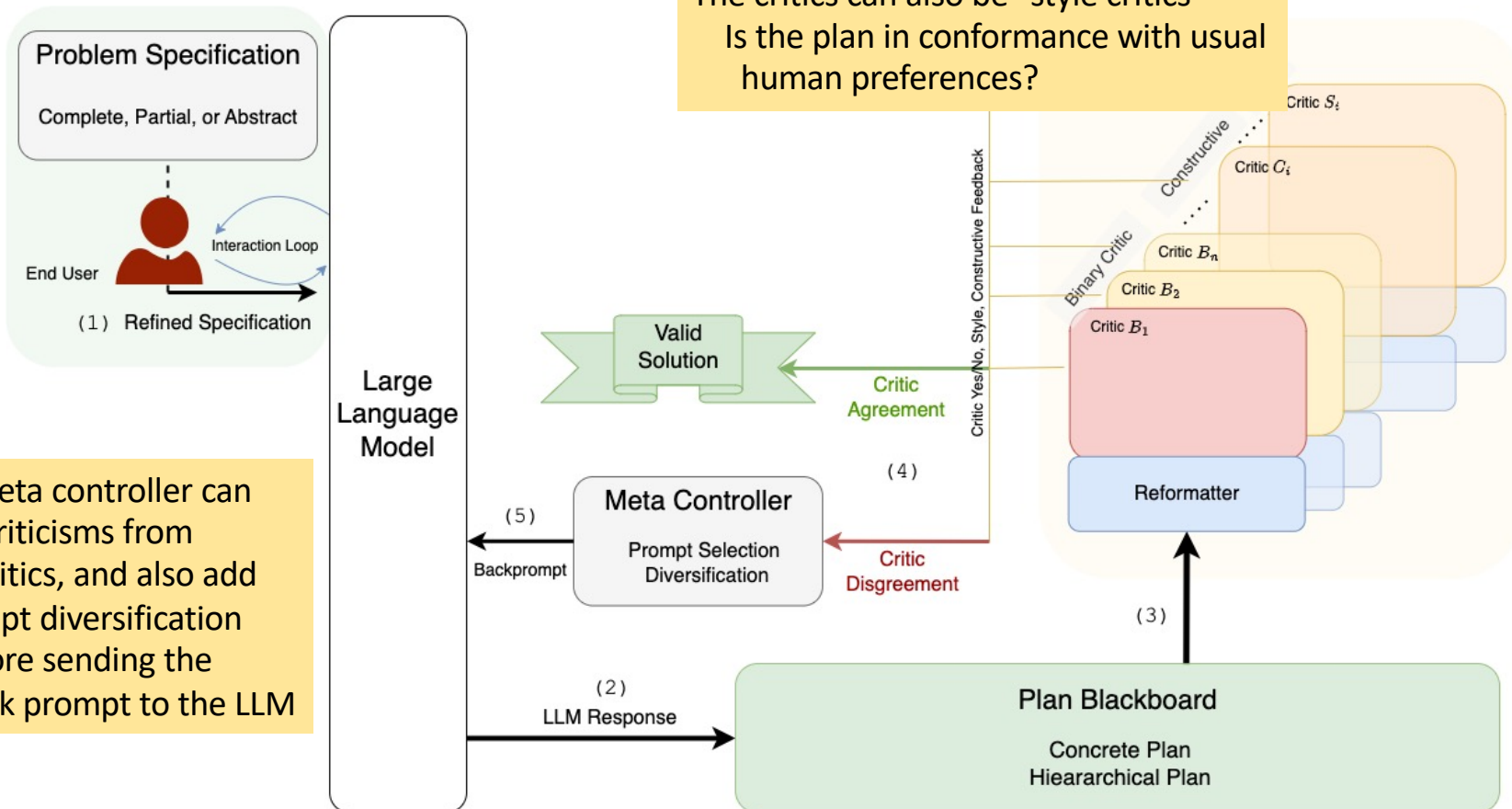
LLM-Modulo: a principled framework for Planning wherein LLMs can play multiple constructive roles



LLM-Modulo with a Bank of critics

We can accommodate multiple critics
e.g. one to verify causal correctness
one to verify resource usage
Can be constructive critics

The critics can also be “style critics”
Is the plan in conformance with usual
human preferences?



The meta controller can pool criticisms from the critics, and also add prompt diversification before sending the back prompt to the LLM

Subbarao Kambhampati (ಕೃಷ್ಣವಾಣಿ) @rao2z

Style is a *distributional* property; correctness is an *instance-level* property.

LLMs (and GenAI) learn and sample from the distribution.

Databases store and retrieve instances.

Ergo, LLMs can't self-verify correctness (or assure factuality) and databases can't get creative!

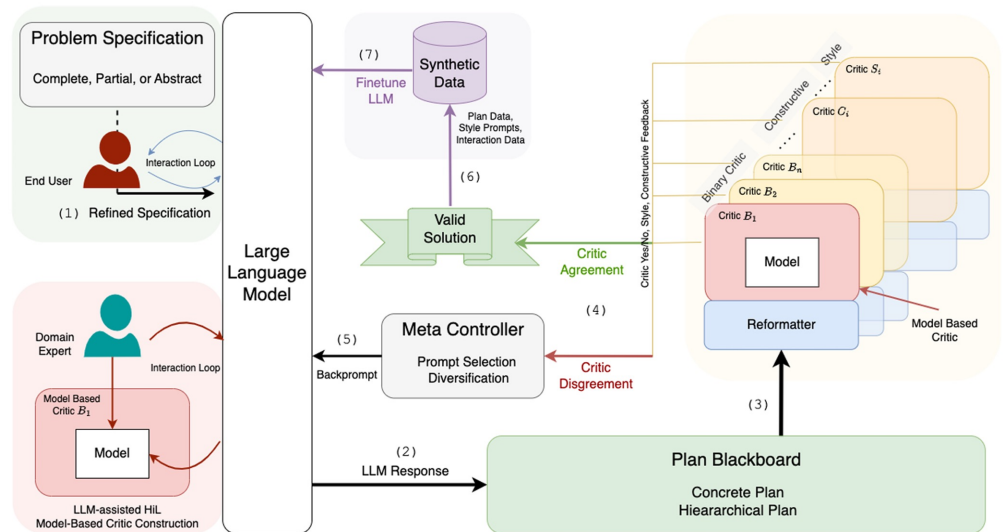
7:46 AM · Mar 26, 2024 · 8,635 Views

View post engagements

7 18 75 24

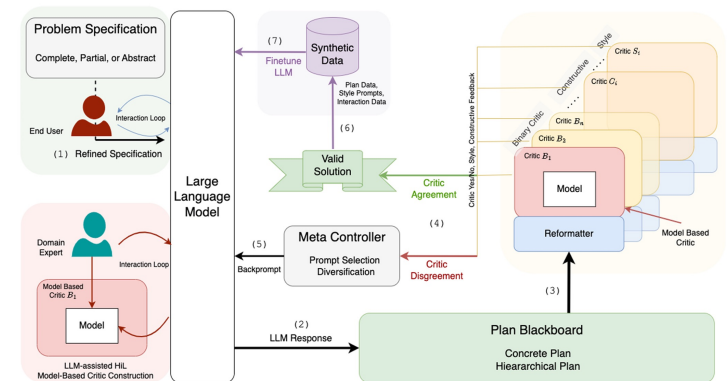
Types of Critics in LLM-Modulo

- Correctness vs. Style
 - LLMs can't directly critic correctness
 - But can help in obtaining the model driving the critics
 - LLMs can be directly used for critiquing style
- Critics can be
 - Binary ("try again"),
 - Constructive
 - Point out errors in the candidate
 - Suggest local repairs
 - **Partial (e.g. Unit Tests)**
- Meta controller combines the criticisms from the various critics and sends it as a back prompt
 - Can also do prompt diversification as part of the process

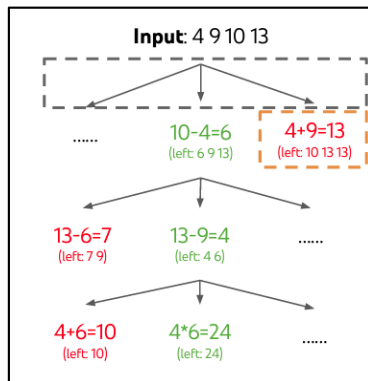
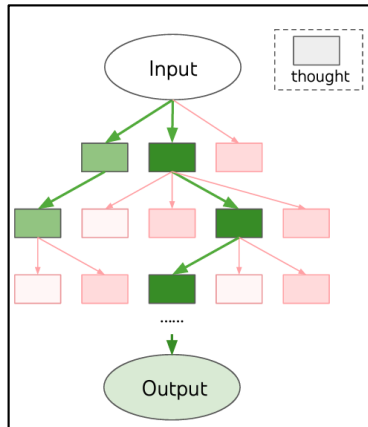


Soundness & Completeness of LLM-Modulo

- LLM-Modulo is a Generate-Test strategy
- Soundness depends on the soundness of the critics/verifiers
 - LLMs themselves are only used for style critiquing, or to tease-out models for the corrects critics (in a semi-automated fashion)
- Completeness depends on the LLM's ability to generate a plan candidate that passes the muster with the verifiers
 - While LLMs are “significantly above average” generators, the completeness of their generation is a function both of (1) training data and (2) ability to generate **diverse candidates**
 - Prompt Diversification Strategies help here
 - And can be part of the **meta-controller**



Tree of Thoughts (ToT)



ToT

- People start realizing LLMs can't plan, but with that comes the illusions that they can do **systematic search** like humans
- The best way to understand the "tree" in ToT is not as a search tree in the problemsolving agents, but as a "**prompt diversification tree**" that is hand crafted in a problem specific way!
 - ToT nudges the LLM to generate hopefully more diverse candidates -- with the hope that one of those will pass the muster of the external verifier.
 - The verifier plays a central role in ensuring soundness!
- We can come within 4% of ToT on 24 puzzle by just sampling 150 diverse candidates direct from the LLM!



<https://twitter.com/rao2z/status/1733845752340967533?s=20>

LLMs as Style Critics & Human Preference Proxies

- We investigate the **potential of LLMs to serve as effective human proxies** by capturing human preferences in human-AI collaboration settings.
- LLMs can play different roles in Human-aware AI interaction: as a **Human Proxy**, Translator (common lingua franca), and the Actor.
- **Theory of Mind (ToM)** requires LLMs to also be able to capture human mental states, desires, and beliefs for **reward design/learning mechanisms**.
- Human-aware AI agents can incorporate such reward functions to account for **human-in-the-loop's preferences**.

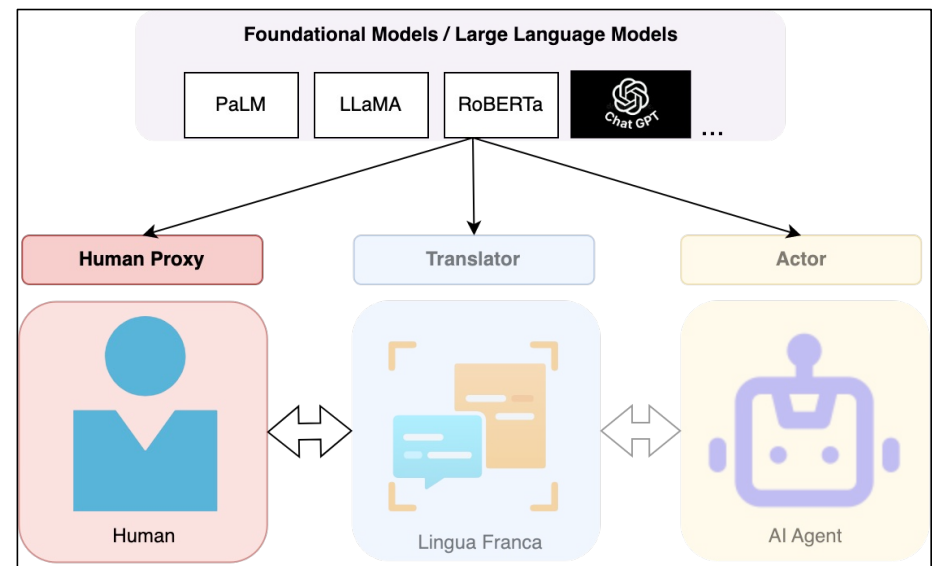


Figure: Different roles of an LLM in Human-AI interaction.

Theory of Mind abilities of Large Language Models in Human-Robot Interaction : An Illusion?

Mudit Verma, Siddhant Bhambri*, Subbarao Kambhampati.*

HRI 2024



LLMs as Human Preference Proxies

Can LLMs capture human preferences?

Probing LLMs with explicability preferences:

- Under explicability preference, the **human expects the agent to behave in a certain way**, and the agent proactively attempts to model this expectation and follow it.
- Here, the human takes the role of an **observer**.

Probing LLMs with sub-task specification preferences:

- We consider a Human-AI teaming scenario where the **human plays an active role and can perform actions** in the world alongside the AI agent.
- **Sub-task specification preferences** involve the agent to produce the same set of sub-tasks that the human has in mind to achieve the team objective.

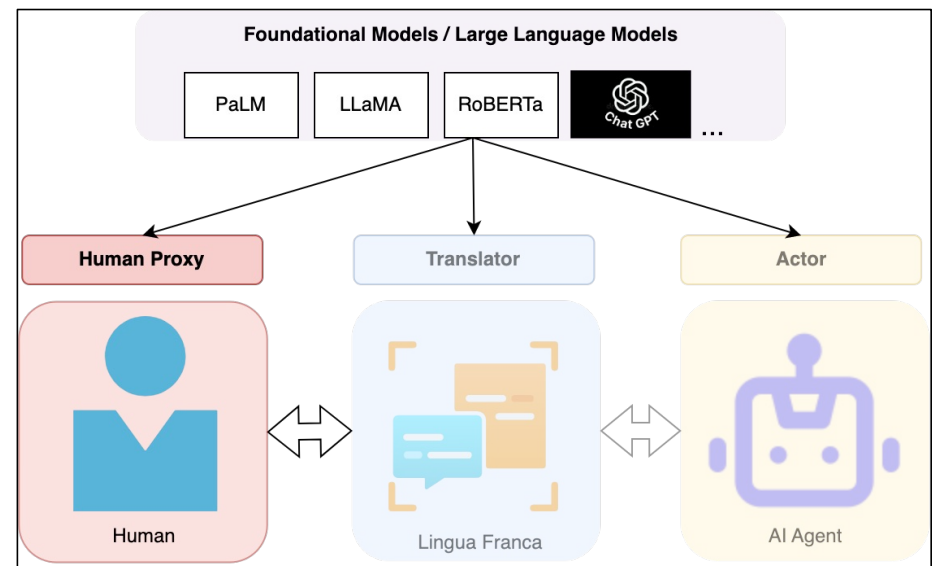


Figure: Different roles of an LLM in Human-AI interaction.

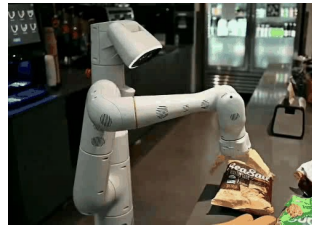
LLMs as Behavior Critics to catch undesirable robot behaviors

Can LLMs capture human preferences in embodied AI tasks?

- It may be intractable to construct formal verifiers for tasks that have a wide scope.
- LLMs or VLMs can be a proxy of common human preferences and undesirability
- We evaluated GPT-4V with videos of diverse suboptimal behaviors
- GPT-4V critic catches **69%** of undesirable behaviors (recall rate) while only **62%** of the critiques are valid (precision rate)
- Results confirm the broadness of GPT-4V's knowledge & the subpar precision of its outputs

"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors
Lin Guan*, Yifan Zhou*, Denis Liu, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati.

Pick up a bag of chips



GPT-4V: The robot gripped the bag of chips too tightly, causing it to crumple and potentially damaging the contents.



Place knife on board



GPT-4V: The robot released the knife from a height that caused it to bounce upon hitting the cutting board, which is potentially dangerous due to the sharp blade.



Pour coke into the glass



GPT-4V: The robot poured the coke too quickly, causing excessive fizzing and overflow of the liquid from the glass.



Serve orange juice



GPT-4V: The robot's arm holding the glass of orange juice passes over the laptop, which poses a risk of spilling liquid on the electronic device.



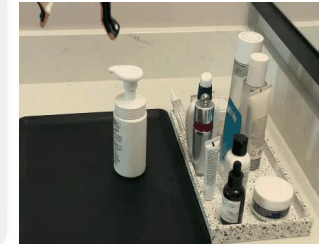
Hand scissors to human



GPT-4V: The scissors are handed over with the pointy ends facing the person, which poses a risk of injury.



Place facial cleanser



GPT-4V: The robot dispensed facial cleanser onto the tray instead of placing the bottle onto the tray.



LLMs as Behavior Critics to catch undesirable robot behaviors

Can LLMs capture human preferences in embodied AI tasks?

- It may be intractable to construct formal verifiers for tasks that have a wide scope.
- LLMs or VLMs can be a proxy of common human preferences and undesirability
- We evaluated GPT-4V with videos of diverse suboptimal behaviors
- GPT-4V critic catches **69%** of undesirable behaviors (recall rate) while only **62%** of the critiques are valid (precision rate)
- Results confirm the broadness of GPT-4V's knowledge & the subpar precision of its outputs

Place vessel onto burner



Move spoon to bowl



GPT-4V:

✓ The robot placed the vessel off-center on the burner, which could lead to uneven heating or potential tipping of the vessel.

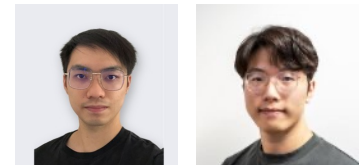
✗ (grounding error) The robot released the vessel from a height that could cause damage to the vessel or the stove if it were heavier or more fragile.

GPT-4V:

⚠ (unoperationalizable) The robot caused pasta sauce to drip onto the counter while transferring the spoon from the jar to the bowl.

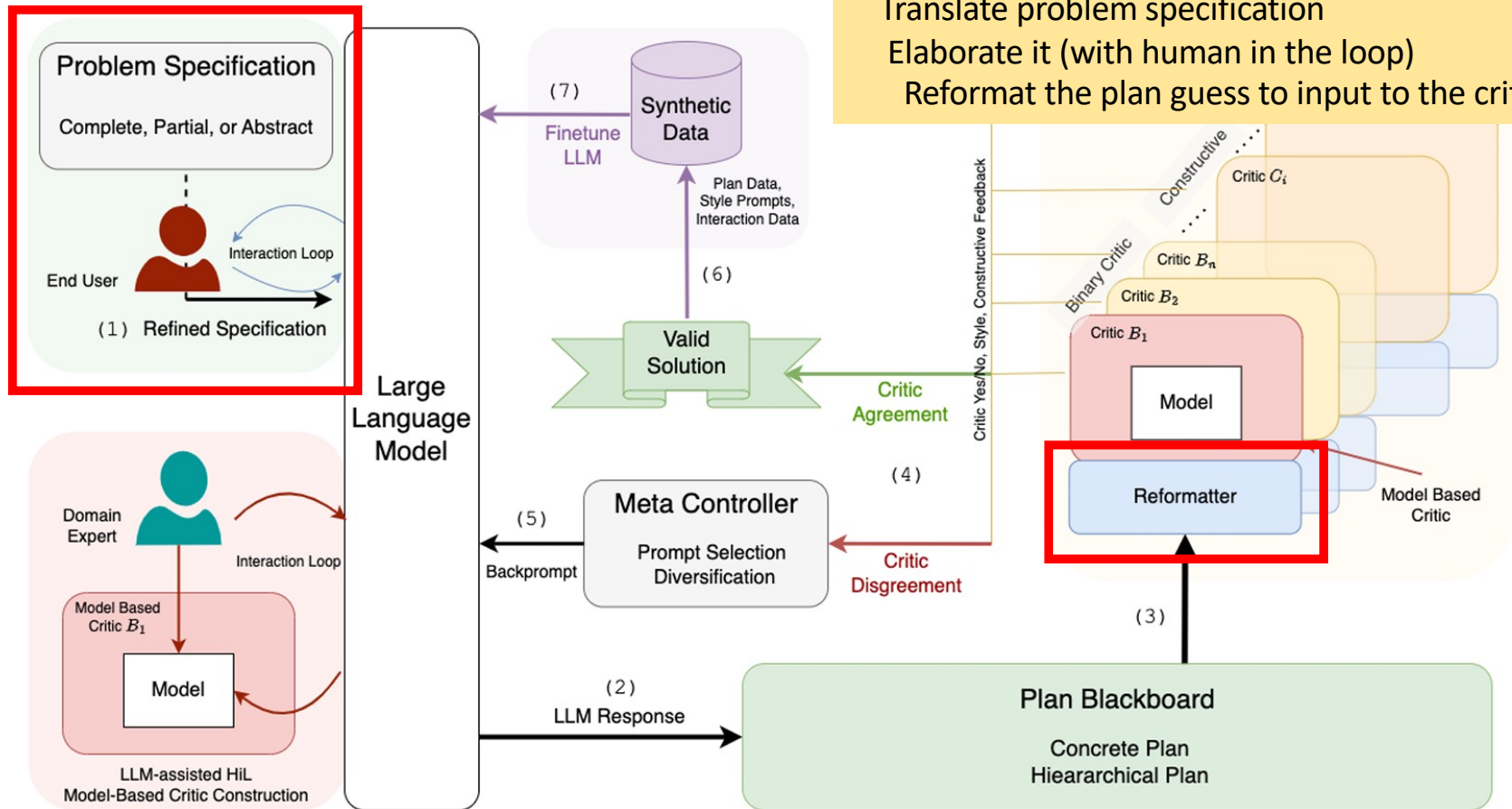
Note: GPT-4V correctly detects the sauce spillage but does not provide the fundamental reason for why it occurred, thereby making it difficult to ascertain actionable advice for behavior correction.

"Task Success" is not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors
Lin Guan*, Yifan Zhou*, Denis Liu, Yantian Zha, Heni Ben Amor, Subbarao Kambhampati.



LLMs for Format Change/Specification Elaboration

Given that LLMs are good at format change they can be used to
Translate problem specification
Elaborate it (with human in the loop)
Reformat the plan guess to input to the critic



Text to plan using GPT-3

- ❑ Workshop on KEPS (ICAPS'21)
- ❑ Workshop on Planning for Financial Services (ICAPS'21)

TEXT: Windows Help and Support
 Start internet explorer.
 You need to eventually click on the advanced tab
 But before that, click on the internet options on the tools menu.
 After the click on the advanced tab, click to clear or select the enable
 personalized favorites menu check box.

ACTIONS:
 click(internet,explorer), click(tools), click(internet,options), click(advanced),
 click(personalized,favorites), click(check,box)

TEXT: CookingTutorial
 For this recipe you need to let the beef simmer for 30 minutes.
 After that, make sure that you don't start cooking oats right away.
 You have to first measure their quantity which needs to be 150g.
 Then cook them and pour the vegetables with the meat.

ACTIONS:
 let(beef,simmer), measure(quantity,oats), cook(oats), pour(vegetables,meat)

TEXT: WikiHow Home and Garden
 You can decorate your floor anytime.
 But before decorating your floor it's important that you paint your walls first.
 After painting your walls, remove all furniture in the room.

ACTIONS:
 paint(walls), remove(furniture), decorate(floor)

Figure 2: Query examples on WHS, CT and WHG. Each query was input to Davinci along with two preceding training instances containing the largest proportion of *optional* and *exclusive* actions. The output is shown in regular text while the input is displayed in bold.

Model	<i>Action names</i>			<i>Action arguments</i>		
	WHS	CT	WHG	WHS	CT	WHG
EAD	86.25	64.74	53.49	57.71	51.77	37.70
CMLP	83.15	83.00	67.36	47.29	34.14	32.54
BLCC	90.16	80.50	69.46	93.30	76.33	70.32
STFC	62.66	67.39	62.75	38.79	43.31	42.75
EASDRL	93.46	84.18	75.40	95.07	74.80	75.02
cEASDRL	97.32	89.18	82.59	92.78	75.81	76.99
GPT-3						
Davinci	86.32	58.14	43.36	22.90	29.63	22.25
Curie	75.80	35.57	22.41	31.75	22.16	13.79
Babbage	62.59	20.62	14.95	22.91	12.59	7.33
Ada	60.68	14.68	8.90	17.91	4.13	2.27

Table 3: F_1 scores for all actions and their arguments across the WHS, CT and WHG datasets for the state-of-art sequence extraction models and GPT-3. State-of-art task-specific model F_1 scores are extracted from Miglani and Yorke-Smith (2020); Feng, Zhuo, and Kambhampati (2018) and represent their best possible recorded performance.

- We investigated how **GPT-3**, one of the most recent transformer-based language models, can be used to extract structured actions from natural language texts. We find that these models achieve comparable, and in some cases better scores than previous state of the art task-specific methods
- **Impact:** Existing knowledge in the form of textual procedures and plans can be translated into formal representations to aid novice Navy personnel understand and carry out complex procedures. The translated procedures can also be leveraged by other automated systems in-place.

AutoTAMP

- LLMs are being used as **translators** and **verifiers**
- They translate from natural language to Signal Temporal Logic representation.
- An STL planner is used to come up with plans.
- Re-prompting technique is used on the LLM translator and the verifier to improve performance

	HouseWorld1 (soft time cst.)	HouseWorld2 (hard time cst.)	Chip Challenge (hard geo. cst.)
GPT-4 end2end	9.5%	9.5%	0%
GPT-4 naive task planning	90%	45%	0%
SayCan	90%	47.5%	0%
GPT-4 task planning/feed.	92%	49%	0%
GPT-4/STL	43.5%	42%	42.7%
GPT-4/STL/Syn.	59.5%	59%	70%
GPT-4/STL/Syn./Sem. (AutoTAMP)	82.5%	82%	87.7%

```

User input prompt:
<Task explanation> + <Few-shot examples> +

Environment settings: [name: room1, color: red, position and size: [-1, 2, 0, 4], function:
kitchen], [name: room2, color: purple, position and size ..., function: livingroom], [name: room3,
color: pink, position and size ..., function: bedroom], [name: room4, color: blue, position and
size ..., function: restroom1] , [name: room5, color: blue, position and size ..., function:
restroom2] ...

Instruction: Enter two rooms with closest color to red but not pure red, and always do not enter
the blue area.

LLM response1:
transformed sentence: finally prop_1 and globally prop_2.
Lifted TL : ['and', 'finally', 'prop_1', 'globally', 'prop_2'];
'prop_1' : ['enter(room1)', 'enter(room3)'];
'prop_2' : ['or', 'not_enter(room4)']
Full STL : ['and', 'finally', 'enter(room1)', 'enter(room3)', 'globally', 'or', 'not_enter(room4)']

Syntactic checker prompt:
Your predicted prop_1 lacks one operator with two leaf nodes. One example of syntactic correct
format should be like ['and', 'enter(room1)', 'enter(room3)']; Your predicted prop_2 owns one extra
operator with two leaf nodes or lacks one action proposition. One example of syntactic correct
format should be like ['not_enter(room4)']

LLM response2:
transformed sentence: finally prop_1 and globally prop_2.
Lifted TL : ['and', 'finally', 'prop_1', 'globally', 'prop_2'];
'prop_1' : ['and', 'enter(room1)', 'enter(room3)'];
'prop_2' : ['or', 'not_enter(room4)', 'not_enter(room5)']

Full STL : ['and', 'finally', 'and', 'enter(room1)', 'enter(room3)', 'globally', 'or',
'not_enter(room4)', 'not_enter(room5)']

Syntactic checker:
Correct!

Trajectory generated by STL planner:
[[-1, -1, 0], [-0.5, 1.0, 0.5], ...]

State sequence generated from trajectory:
[[in(road), 0], [in(red kitchen), 0.5], [in(blue restroom2), 1.2], [in(pink bedroom), 1.5], ...]

Semantic checker prompt:
Based on your predicted STL, the state sequence of the generated trajectory is: [in(road) at time
0, in(red kitchen) at time 0.5, in(blue restroom2) at time 1.2, in(pink bedroom) at time 1.5, ...].

Please print the initial instruction again and check whether this state sequence follows the
instruction. Let us do it step by step, first specifically explain the semantic meanings of the
instruction, and then list all the available rooms in the given environment, then determine the
rooms planned to visit or avoid and whether the trajectory is consistent. Next modify or keep the
final STL based on above analysis. First output your thinking steps and in the last line output the
full final STL beginning with STL: .
Output:

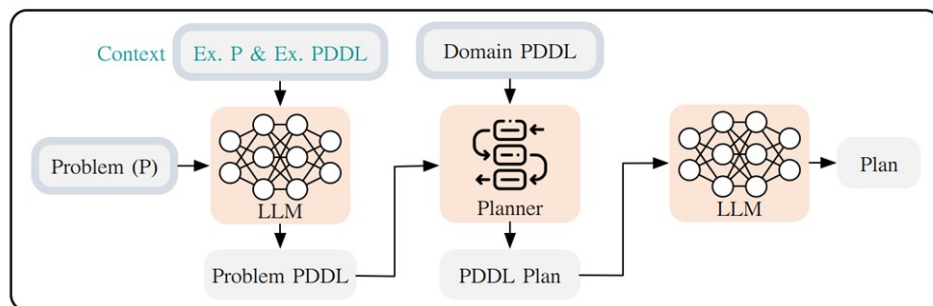
LLM response3:
The instruction says ... The semantic meanings are ... The available rooms are ... Therefore, we should
visit room2 and room3, and avoid both room4 and room5. The modified STL should be,

Final STL : ['and', 'finally', 'and', 'enter(room2)', 'enter(room3)', 'globally', 'and',
'not_enter(room4)', 'not_enter(room5)']

```


LLM+P: Empowering LLMs with Optimal Planning Proficiency

- **LLM** translates Natural Language Problems into Problem PDDL by in-context Learning. (Context is crucial).
- **Planner**: Problem PDDL + Domain PDDL --> PDDL Plan
- **LLM**: PDDL Plan --> Natural Language Plan.
- LLM as a planner:
 - Lacks the ability to reason about preconditions.
 - Performs poorly in Domains that require an understanding of complex spatial relationships.

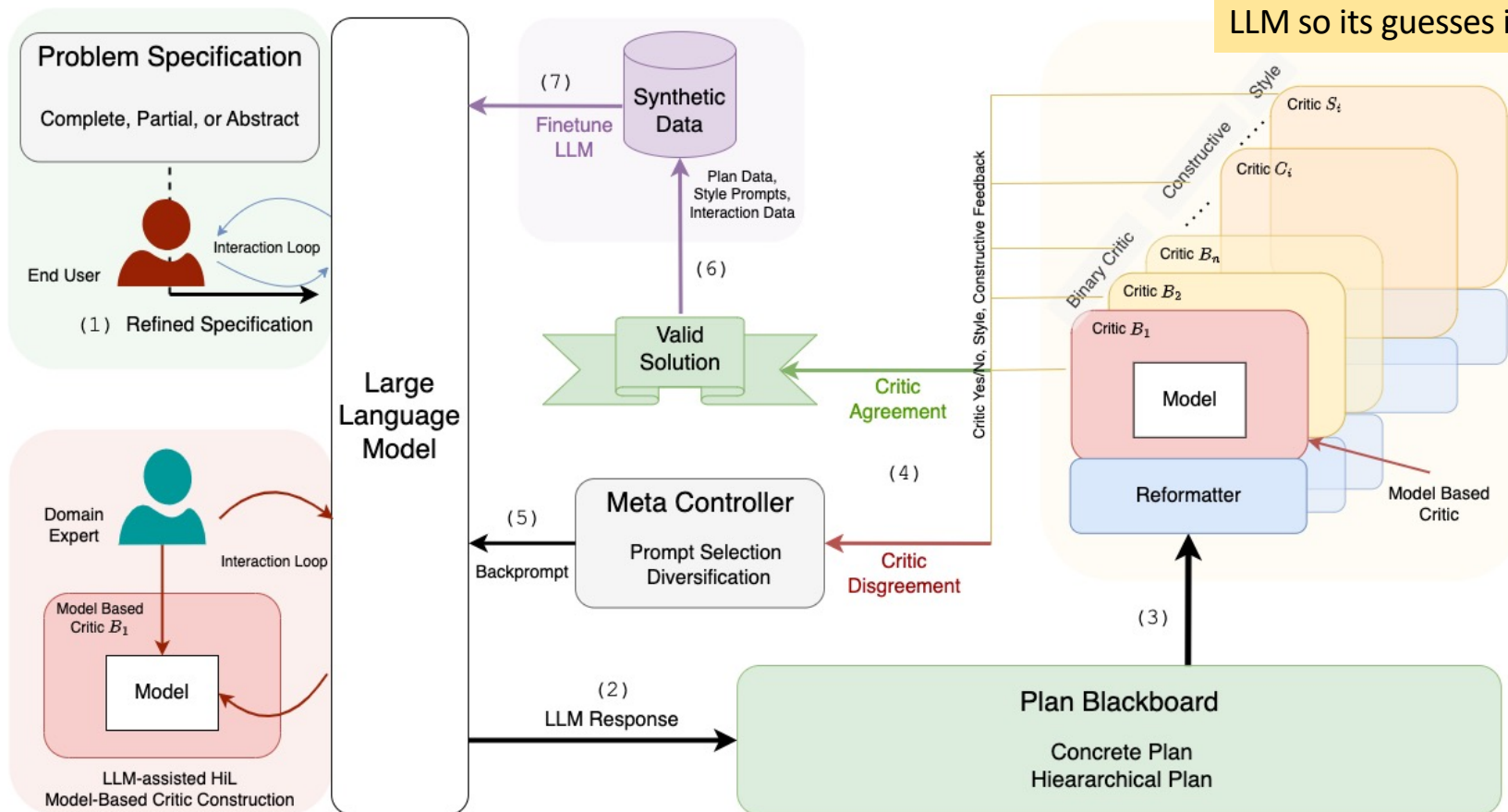


LLM + P (In-context Learning)

Domain	Success Rate %			
	LLM-AS-P (no ctx.)	LLM-AS-P	LLM+P (no ctx.)	LLM+P
BARMAN	0	0	0	85.0 ±0.0
BLOCKSWORLD	0	0	0	56.7 ±2.9
FLOORTILE	0	0	0	53.3 ±2.9
GRIPPERS	10.0	15.0	0	95.0 ±0.0
STORAGE	0	0	0	65.0 ±0.0
TERMES	0	0	0	31.7 ±2.9
TYREWORLD	0	40.0 ±5.0	0	80.0 ±0.0

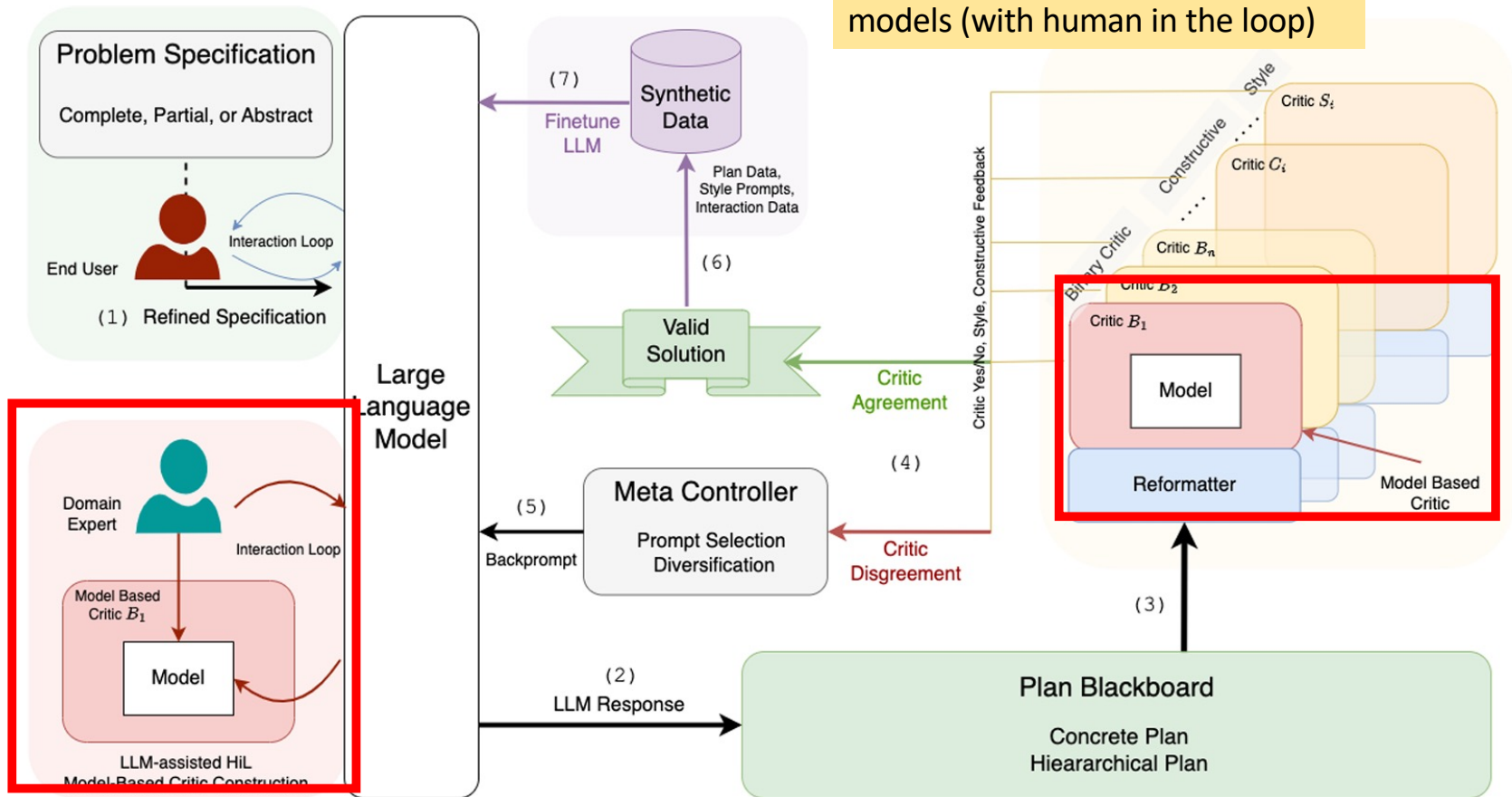
Generating Synthetic Data (Self-Instruct LLM-Modulo Way)

Finally, since the solutions coming out of LLM-Modulo frameworks are sound, they can be used to build a corpus of synthetic data that can be used to fine-tune the LLM so its guesses improve..

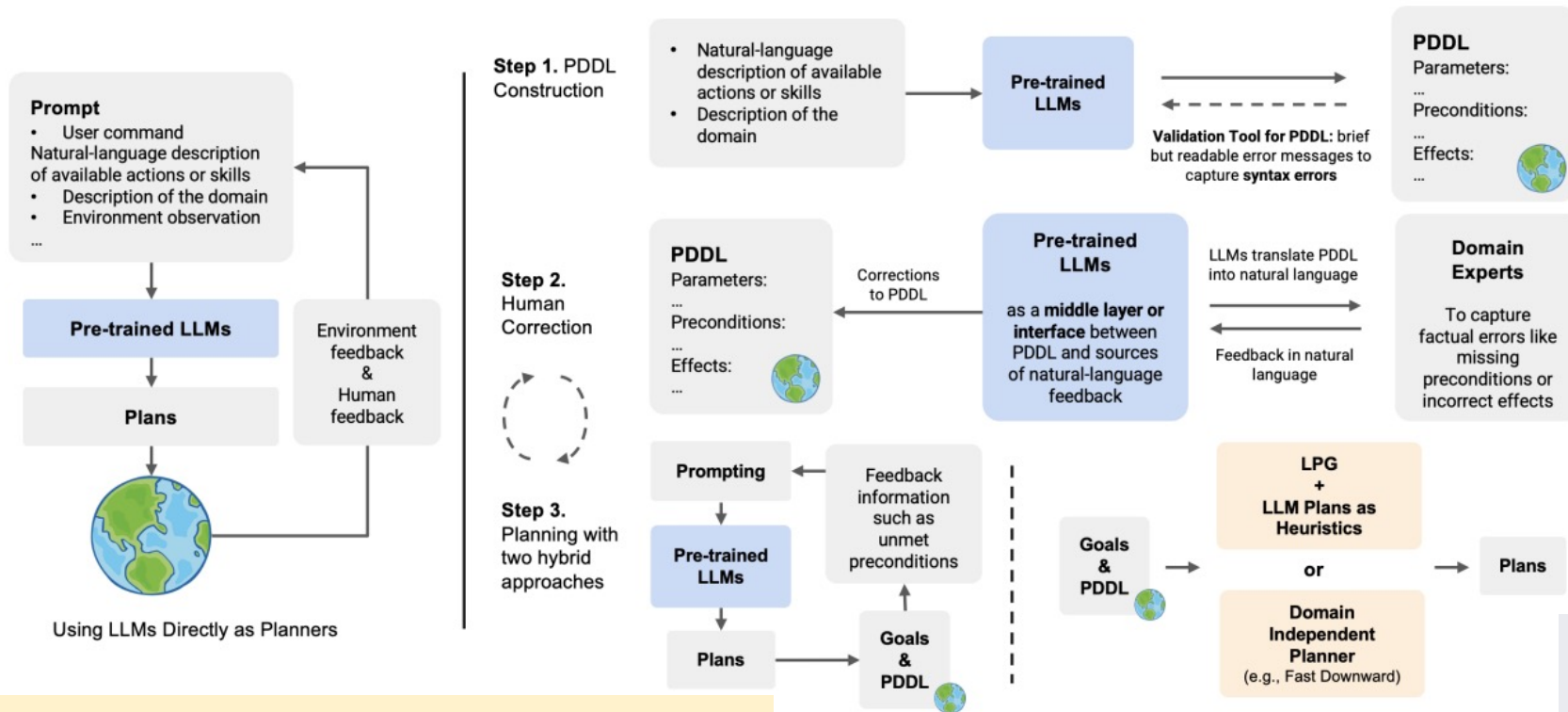


LLMs for Extracting Planning Knowledge

Since LLMs are approximate knowledge sources, they can also be helpful in acquiring domain models (with human in the loop)



LLMs for constructing domain/world models (Model Co-Pilot)



LLM's have universal high-recall (they will never shut up!),
but questionable precision
Automated Planners are guaranteed correct
but for planning problems that they can handle

NeurIPS 2023



LLMs for constructing world models

- We tested on three domains
 1. Household domain
 2. Logistics
 3. Tyreworld

Domain	# of actions	# of params and literals	# of GPT-4 errors	# of GPT-3.5-Turbo errors
Household	22	271	53	218+
Logistics	6	54	2	38
Tyreworld	13	108	4	94+

Table 1: The number of errors in the domain models produced by the LLMs for each of the domains. A "+" mark indicates that the generated model is excessively noisy, making it challenging to determine an exact number of errors.

Action description

This action enables the robot to toggle a small appliances (like humidifiers and light bulbs) to ↪ switch them off. For example, the robot toggles humidifier.2 off, or the robot toggle ↪ light_bulb.1 off.

GPT4: Toggle a small appliance off

Parameters:

1. ?r - robot: the robot performing the action
2. ?o - householdObject: the small appliance to be toggled off
3. ?f - furnitureAppliance: the furniture or appliance where the small appliance is located

Preconditions:

```
(and
  (robot-at ?r ?f)
  (object-in-on-furniture ?o ?f)
  (robot-gripper-empty ?r)
  (appliance-on ?o)
)
```

Effects:

```
(and
  (not (appliance-on ?o))
)
```

Example from Household domain

Action description

This action enables the agent to load a package into a truck. For example, load a package.1 into a ↪ truck.1.

GPT-4: Load a package into a truck

Parameters:

1. ?p - package: the package to load
2. ?t - truck: the truck to load the package into
3. ?l - location: the location where the package and truck are

Preconditions:

```
(and
  (package-at ?p ?l)
  (truck-at ?t ?l)
)
```

Effects:

```
(and
  (not (package-at ?p ?l))
  (package-in-truck ?p ?t)
)
```

Example from Logistics domain

RL Systems can Benefit Significantly with partially correct symbolic models!

[The Kind LLMs are only Too Happy to give!]

Incomplete Symbolic Model

- Includes potentially **missing information** and **mistakes**
- But still provides useful information about task



Extract information from the model that is guaranteed to be correct

Use landmarks as subgoals

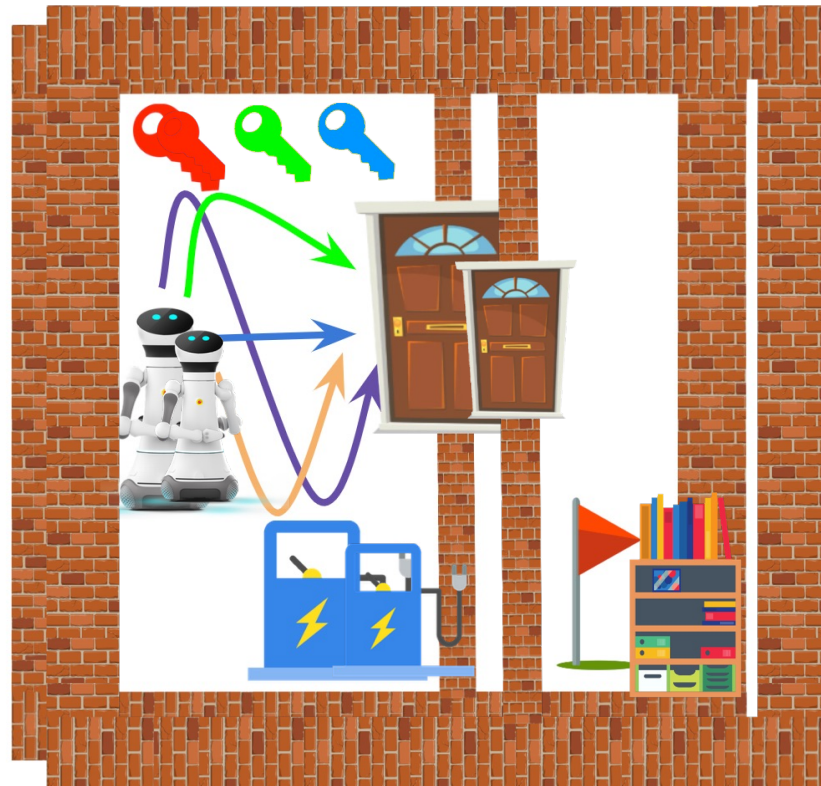
- Example: door-open, at-destination ...



Derive reward functions

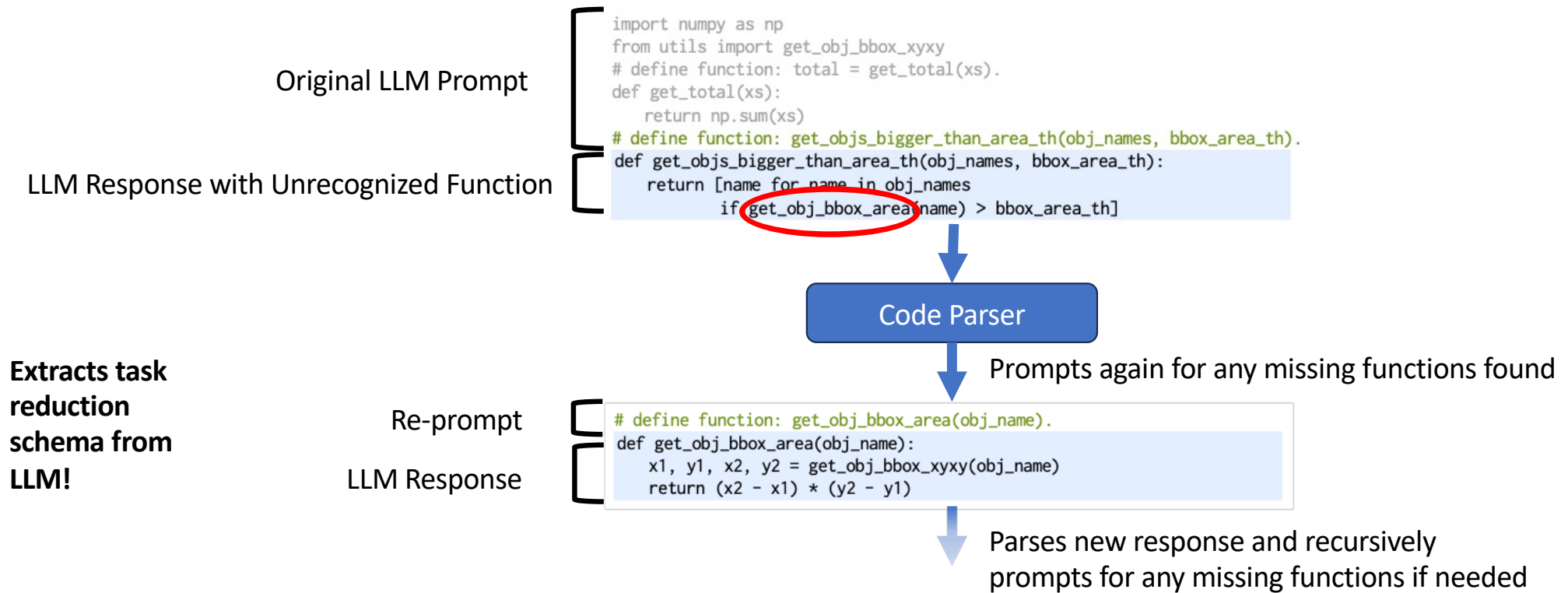
Diverse set of skills learned per landmark

- Example: multiple ways to get to the door in the image on the right



[ICML 2022]

Code as (Hierarchical) Policies



Liang, Jacky, et al. "Code as policies: Language model programs for embodied control." *arXiv preprint arXiv:2209.07753* (2022). Prompts are from that paper.

Travel Planning Benchmark

- New benchmark for travel planning proposed in Feb 2024
- Three different types of constraints
 - Environment constraints
 - Common-sense constraints
 - Hard constraints
- **GPT-4-Turbo could solve only 0.6% (out of 1000 queries)**
- Not surprising! We show that LLM's can't even stack blocks correctly, there's surely no hope for travel planning that has lots of constraints!!

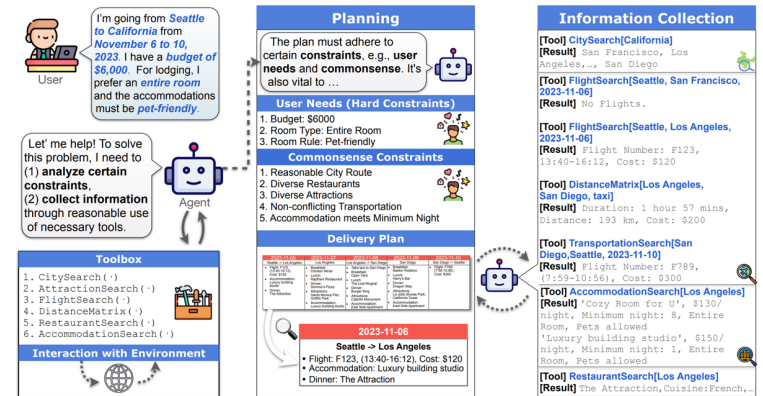
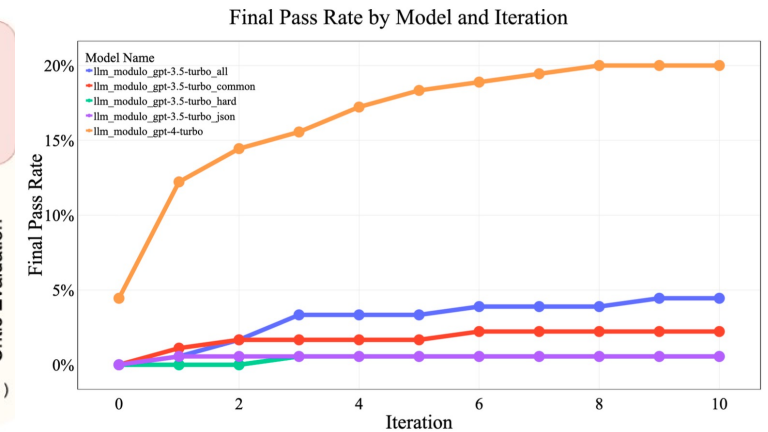
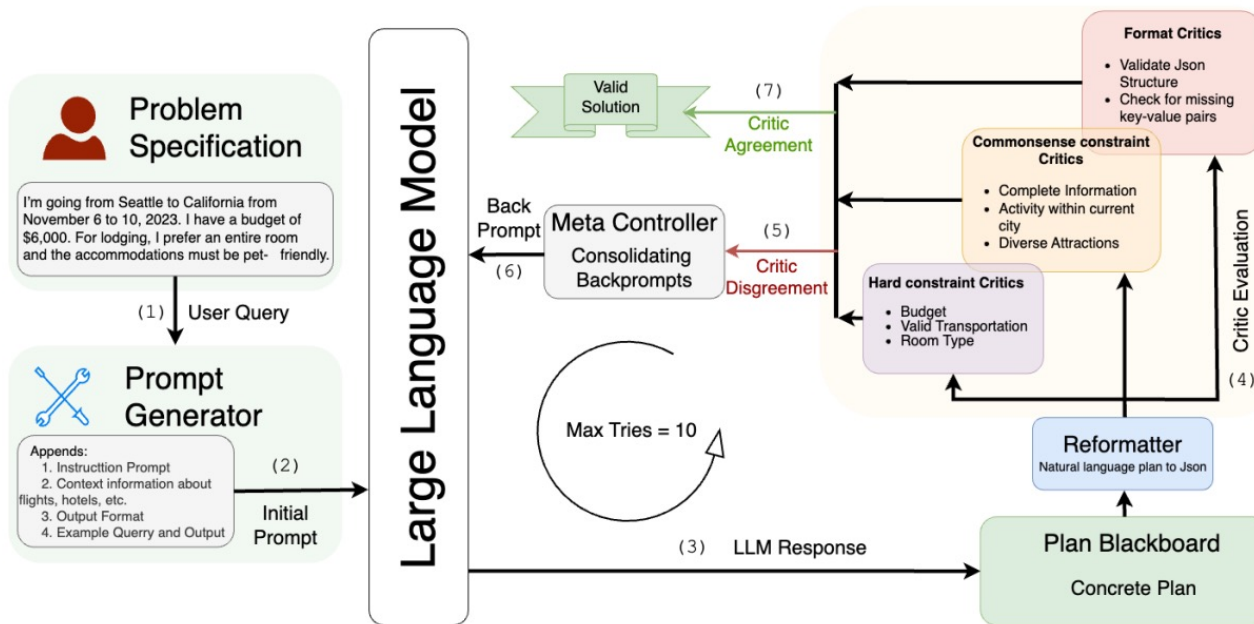


Figure 1. Overview of TravelPlanner. Given a query, language agents are tasked with employing various search tools to gather information. Based on the collected information, language agents are expected to deliver a plan that not only satisfies the user's needs specified in the query but also adheres to commonsense constraints.

Table 1. Constraint description. The environment constraints are manifested through the feedback received from the environment, assessing whether the language agent can adjust its plan appropriately. The commonsense constraints and hard constraints are evaluated based on how well the language agent's plan aligns with these specific criteria.

Constraint	Description
Environment Constraint	
Unavailable Transportation	There is no available flight or driving information between the two cities.
Unavailable Attractions	There is no available attraction information in the queried city.
Commonsense Constraint	
Within Sandbox	All information in the plan must be within the closed sandbox; otherwise, it will be considered a hallucination.
Complete Information	No key information should be left out of the plan, such as the lack of accommodation during travel.
Within Current City	All scheduled activities for the day must be located within that day's city(s).
Reasonable City Route	Changes in cities during the trip must be reasonable.
Diverse Restaurants	Restaurant choices should not be repeated throughout the trip.
Diverse Attractions	Attraction choices should not be repeated throughout the trip.
Non-conf. Transportation	Transportation choices within the trip must be reasonable. For example, having both "self-driving" and "flight" would be considered a conflict.
Minimum Nights Stay	The number of consecutive days spent in a specific accommodation during the trip must meet the corresponding required minimum number of nights' stay.
Hard Constraint	
Budget	The total budget of the trip.
Room Rule	Room rules include "No parties", "No smoking", "No children under 10", "No pets", and "No visitors".
Room Type	Room types include "Entire Room", "Private Room", "Shared Room", and "No Shared Room".
Cuisine	Cuisines include "Chinese", "American", "Italian", "Mexican", "Indian", "Mediterranean", and "French".
Transportation	Transportation options include "No flight" and "No self-driving".

Adapting LLM-Modulo Framework For Travel Planning



arXiv > cs > arXiv:2405.20625

Computer Science > Artificial Intelligence

[Submitted on 31 May 2024]

Robust Planning with LLM-Modulo Framework: Case Study in Travel Planning

Atharva Gundawar, Mudit Verma, Lin Guan, Karthik Valmееkam, Siddhant Bhambri, Subbarao Kambhampati

As the applicability of Large Language Models (LLMs) extends beyond traditional text processing tasks, there is a burgeoning interest in their potential to excel in planning and reasoning assignments, realms traditionally reserved for System 2 cognitive competencies. Despite their perceived versatility, the research community is still unraveling effective strategies to harness these models in such complex domains. The recent discourse introduced by the paper on LLM Modulo marks a significant stride, proposing a conceptual framework that enhances the integration of LLMs into diverse planning and reasoning activities. This workshop paper delves into the practical application of this framework within the domain of travel planning, presenting a specific instance of its implementation. We are using the Travel Planning benchmark by the OSU NLP group, a benchmark for evaluating the performance of LLMs in producing valid itineraries based on user queries presented in natural language. While popular methods of enhancing the reasoning abilities of LLMs such as Chain of Thought, ReAct, and Reflexion achieve a meager 0%, 0.6%, and 0% with GPT3.5-Turbo respectively, our operationalization of the LLM-Modulo framework for TravelPlanning domain provides a remarkable improvement, enhancing baseline performances by 4.6x for GPT4-Turbo and even more for older models like GPT3.5-Turbo from 0% to 5%. Furthermore, we highlight the other useful roles of LLMs in the planning pipeline, as suggested in LLM-Modulo, which can be reliably operationalized such as extraction of useful critics and reformulator for critics.

Results on Travel Planning Benchmark

Model	Delivery Rate	Commonsense Pass Rate		Hard Pass Rate		Final Pass Rate
		Micro	Macro	Micro	Macro	
Direct _{GPT-3.5-Turbo}	99.4	61.5	3.9	11.2	2.8	0.0
Direct _{GPT-4-Turbo}	100	84.9	25.6	51.9	24.4	4.4
CoT _{GPT-3.5-Turbo}	100	66.3	3.3	11.9	5	0
ReAct _{GPT-3.5-Turbo}	82.2	47.6	3.9	11.4	6.7	0.6
Reflexion _{GPT-3.5-Turbo}	93.9	53.8	2.8	11	2.8	0
LLM Modulo [All] _{GPT-3.5-Turbo}	97.8	59.8	13.3	14	6.7	5
LLM Modulo [Common] _{GPT-3.5-Turbo}	100	67.9	16.7	14	5	2.8
LLM Modulo [Hard] _{GPT-3.5-Turbo}	100	61.3	4.4	10.7	5.6	1.6
LLM Modulo [Json] _{GPT-3.5-Turbo}	100	61.3	4.4	10.2	3.9	1.1
LLM Modulo [All] _{GPT-4-Turbo}	100	89.2	40.6	62.1	39.4	20.6

Table 1. We report the results on TravelPlanning Validation set following (Xie et al., 2024). Grayed out results on CoT / ReAct / Reflexion variants are reported from (Xie et al., 2024) for completeness. Direct_{Model} are reproduced baselines as implemented in (Xie et al., 2024). LLM Modulo[Critic]_M represents the critics used during the LLM-Modulo planning with model M. Values are percentages of delivery rate, micro and macro commonsense and hard constraints and finally, success rate defined as *Final Pass Rate* as in (Xie et al., 2024).

Is LLM-Modulo just Shoe-Horning LLMs?

(Why bother with LLMs when we already have formal planning systems?)

- Formal planning systems provide soundness and completeness guarantees
 - ..but only with respect to the class of problems they can handle
 - ..for which there are hand-coded/learned models
 - It becomes the end user's responsibility to check if their problem falls in the class handled by a planning system!
- In contrast, LLMs will always guess solutions—albeit without guarantees
- LLM-Modulo framework is an attempt to keep the best of both worlds
 - Allow end user to pose any problem;
 - Ensure that the solution being sent out is verified by the bank of critics..



Artificial Intelligence
Volume 48, Issue 3, April 1991, Pages 261-297



Two theses of knowledge representation:
Language restrictions, taxonomic
classification, and the utility of
representation services

Jon Doyle, Ramesh S. Patil

Show more ▾

+ Add to Mendeley 🔗 Share 📄 Cite

[https://doi.org/10.1016/0004-3702\(91\)90029-J](https://doi.org/10.1016/0004-3702(91)90029-J)

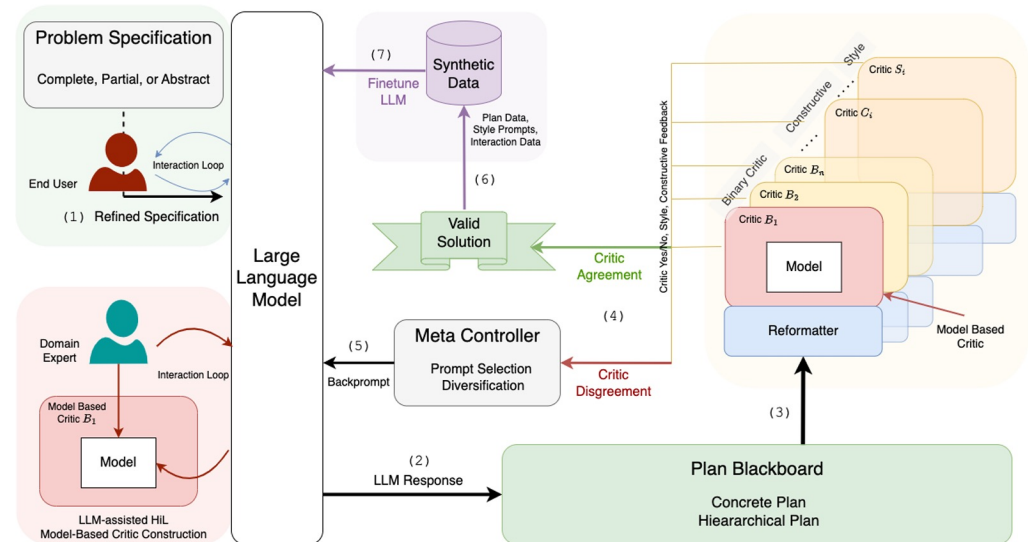
[Get rights and content ↗](#)

Abstract

Levesque and Brachman argue that in order to provide timely and correct responses in the most critical applications, general-purpose knowledge representation systems should restrict their languages by omitting constructs which require nonpolynomial worst-case response times for sound and complete classification. They also separate terminological and assertional knowledge, and restrict classification to purely terminological information. We demonstrate that restricting the terminological language and classifier in these ways limits these "general-purpose" facilities so severely that they are no longer generally applicable. We argue that logical soundness, completeness, and worst-case complexity are inadequate measures for evaluating the utility of representation services, and that this evaluation should employ the broader notions of utility and rationality found in decision theory. We suggest that general-purpose representation services should provide fully expressive languages, classification over relevant contingent information, "approximate" forms of classification involving defaults, and rational management of inference tools.

Recap

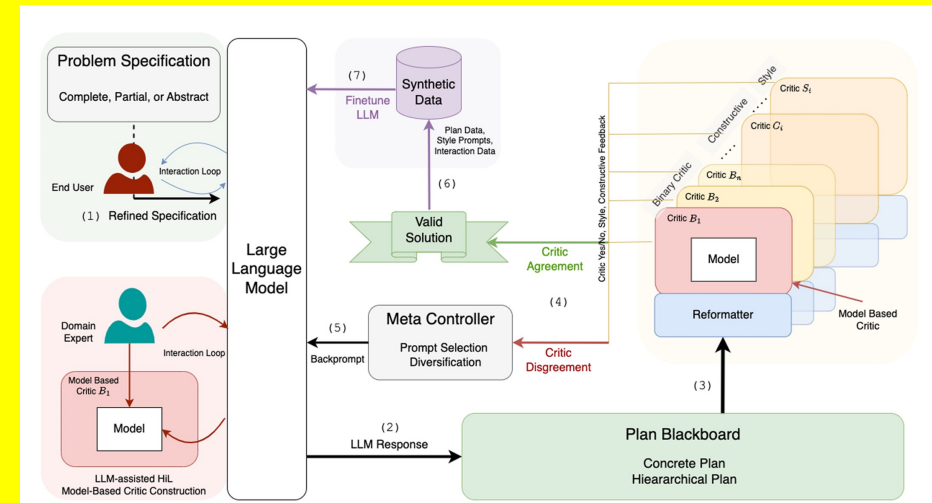
- In any of the frameworks that involve LLMs, external verifiers play a central for ensuring soundness
- LLMs are idea generators without guaranteed correctness
- Different prompting strategies help increase the chance of sampling workable solutions
- LLM-Modulo unified planning framework that:
 - Summarizes constructive roles that LLMs can play
 - Includes the minimum set of components (e.g., set of critics) that ensure the correctness of output plans



Tutorial Overview



- **Part 1**
Introduction; Aims; Perspective on LLMs, Planning, Tutorial Big Picture
- **Part 2**
 - Evaluating LLM Planning capabilities in Autonomous mode, including effect of
 - Prompting strategies (including Chain-of-thought)
 - Fine Tuning
 - Self Verification
 - Understanding the contradictory claims in the literature
- **Part 3**
 - Sane roles of LLMs in Planning (with LLM-Modulo frameworks)
 - LLMs as heuristics, LLMs as candidate generators
 - Back prompting from external verifiers
 - LLMs as sources of domain models (with humans in the loop)
 - LLMs as format changers/specification elaborators¹
- **Part 4**
 - Summary/Lessons



Tutorial Slides



<https://bit.ly/4dbkkY2>

Planning, Scheduling, Reinforcement Learning

- **Planning/Sequential Decision Making:** Given a set of objectives, come up with a course of actions (policy) to achieve them (optimally)
 - Standard planning (and RL) assume the possible actions that the agent can take (and their “models”) are given up front
 - MacGyver Planning involves *coming up with the actions AND sequencing them*
- **Scheduling:** Given a set of tasks/jobs, and a choice of actions for accomplishing each of the tasks, assign actions to tasks so there are no undesired interactions
 - Planning is more general than scheduling (One difference is *precondition cascading* – adding an action to achieve something makes you subgoal on the the preconditions of that action..).
 - Planning is P-Space Complete (even for STRIPS) and Scheduling is NP-Hard
- **(Model-Based) Reinforcement Learning:** The agent acts in a (hopefully *ergodic*) environment and slowly learns the “action models” —and plans with that model.

LLM's Can't Plan; But they can help planning in LLM-Modulo Frameworks

LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

- LLMs can't do planning in autonomous mode
- CoT, Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!)
- Having humans iteratively prompt is an invitation for Clever Hans effect..

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

- LLMs can be used in conjunction with external verifiers and solvers in an **LLM-Modulo** framework (with the verifiers doing *back prompting*)
 - In the LLM-Modulo framework, LLMs can play multiple roles
 - Guess plans
 - Guess domain models
 - Help elaborate the problem specification
 - Translate formats

LLM's Can't Plan; But they can help planning in LLM-Modulo Frameworks

LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

- LLMs can't do planning in autonomous mode
- CoT, ReACT, Fine Tuning etc. don't help that much (as they don't generalize enough)
- They can't improve by self-verification (since they can't self-verify!)
- Having humans iteratively prompt is an invitation for Clever Hans effect..

LLMs can support planning (and expand the range of planning tasks) in LLM-Modulo Frameworks

- LLMs can be used in conjunction with external verifiers and solvers in an **LLM-Modulo** framework (with the verifiers doing *back prompting*)

