

Looks Like we showed that LLMs  
can't Plan in autonomous modes..?



Yann LeCun: Towards Machines That Can Understand, Reason, & Plan

## 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

---

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

---

A PREPRINT

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

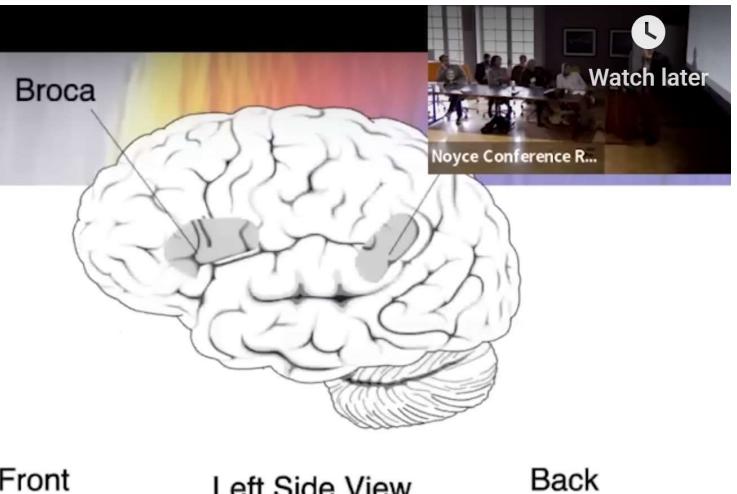
**Idan A. Blank**  
University of California Los Angeles  
iblank@psych.ucla.edu

**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



Front

Left Side View

Back

ArXiv:2206.10498

---

### 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

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

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

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

MORE VIDEOS

123

# 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<sup>†+</sup>  
Dale Schuurmans  
<sup>†</sup>Google Research

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

Hanxu Hu<sup>1\*</sup> Hongyuan Lu<sup>2†</sup> Huajian Zhang<sup>3</sup> Wai Lam<sup>2</sup> Yue Zhang<sup>1</sup>  
<sup>1</sup>Westlake University   <sup>2</sup>The Chinese University of Hong Kong   <sup>3</sup>University of Edinburgh  
{huhanxu,zhangyue}@westlake.edu.cn  
{hylu,wlam}@se.cuhk.edu.hk  
zihzha17@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

### 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

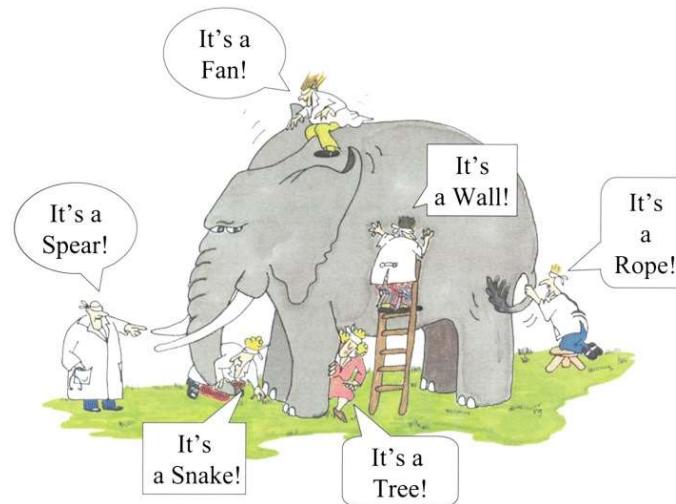
Yusuke Iwasawa  
The University of Tokyo

### 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 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.

# Why this divide?

Answer: Misunderstandings about what planning involves



Don't attribute to malice  
that which is adequately  
explained by ignorance

# 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.

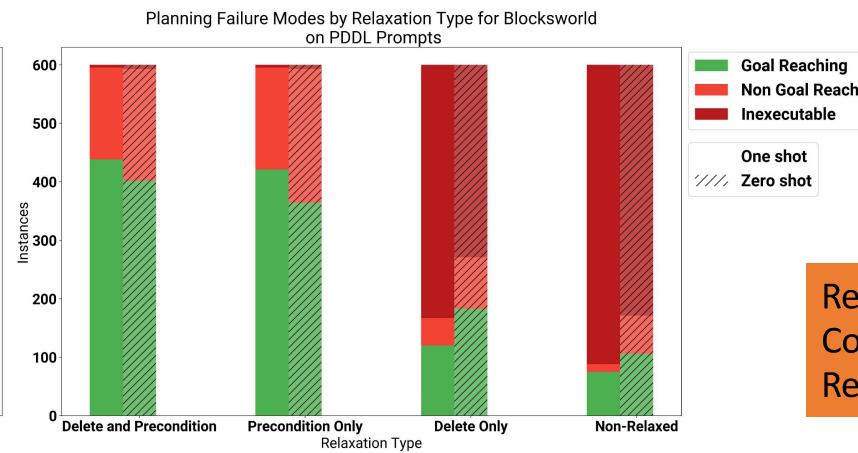
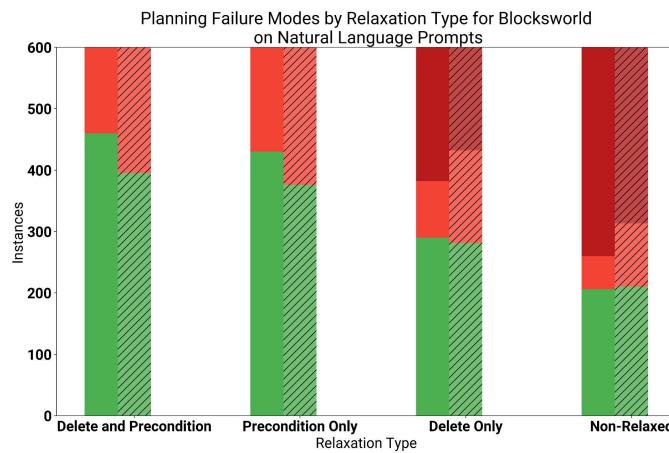
AI Planners will give formal guarantees, but only accept problems expressible in their language.

## 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?

- 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

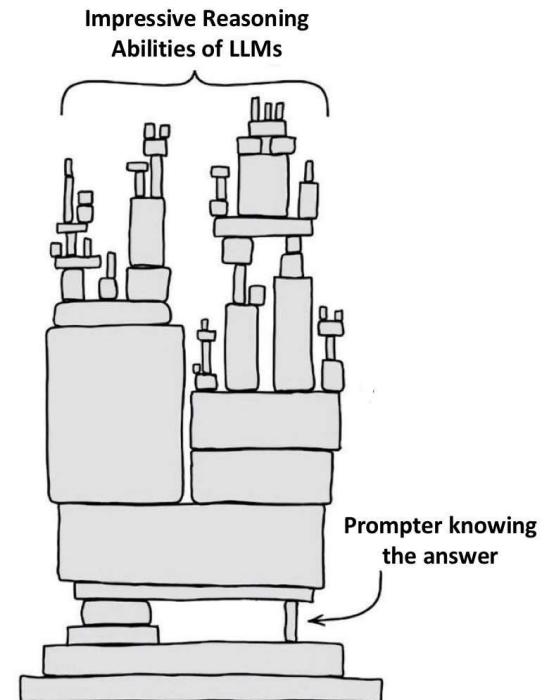
Approximate retrieval of Plans  
≠  
Planning

- 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 pawned 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

# 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





Subbarao Kambhampati (కంభంపాటి సుబ్రామణి)

@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.



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



# 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!!

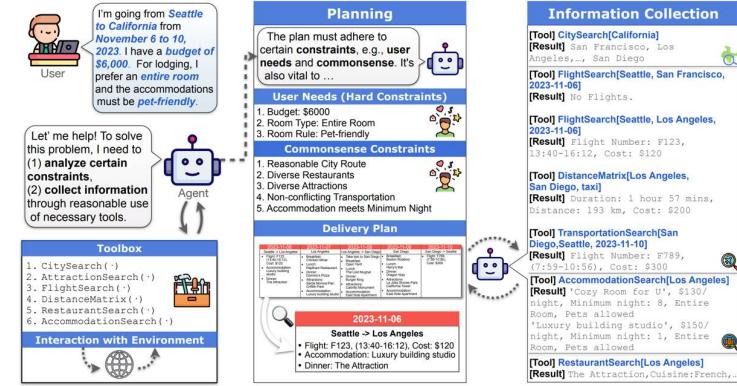


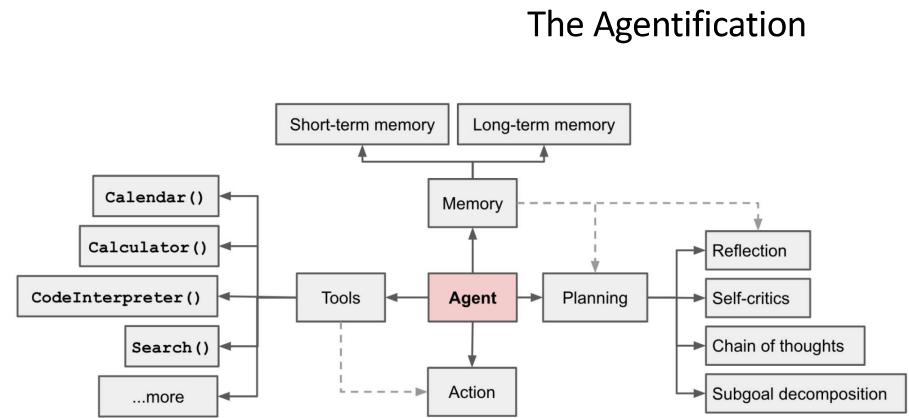
Figure 1. Overview of TravelPlanner. Given a query, language agents are tasked with employing various search tools to gather information. Based on the collected information, language agents are expected to deliver a plan that not only satisfies the user's needs specified in the query but also adheres 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   |
|-------------------------------|---|
| <b>Environment Constraint</b> |   |
| Unavailable Transportation    | There is no available flight or driving information between the two cities.   |
| <b>Commonsense Constraint</b> |   |
| 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. |
| <b>Hard Constraint</b>        |   |
| 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".   |

# Acting vs. Planning: The Agentic LLM Goldrush

- LLMs can obviously be used to invoke external actions
- 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..



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..)

# 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!

Ability to approximately retrieve code segments



Ability to reason and plan



# Talk Overview

## • Part 1

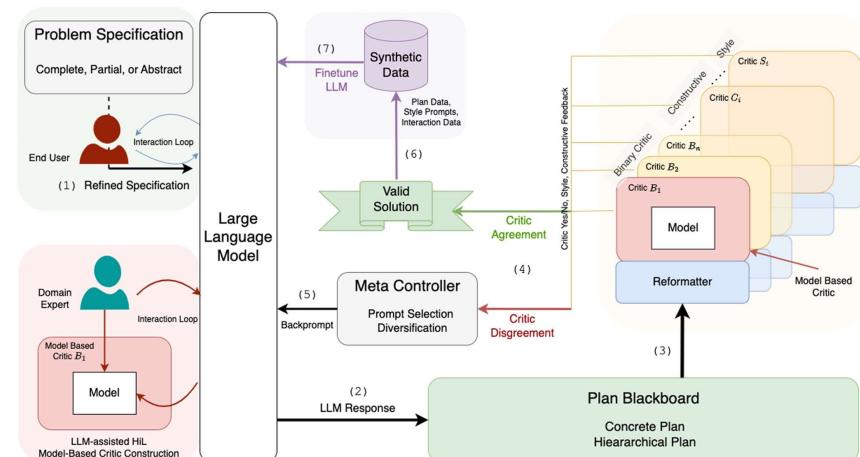
- Evaluating LLM Planning capabilities in Autonomous mode, including effect of
  - Prompting strategies (including Chain-of-thought)
  - Fine Tuning
  - Self Verification
- Understanding the contradictory claims in the literature

## • Part 2

- 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 elaborators<sup>1</sup>

## • Part 3

- Some thoughts on why people tend to ascribe planning/reasoning abilities to LLMs



**Can LLMs help in Planning at all?**

# LLMs as Idea Generators (“Muses”)

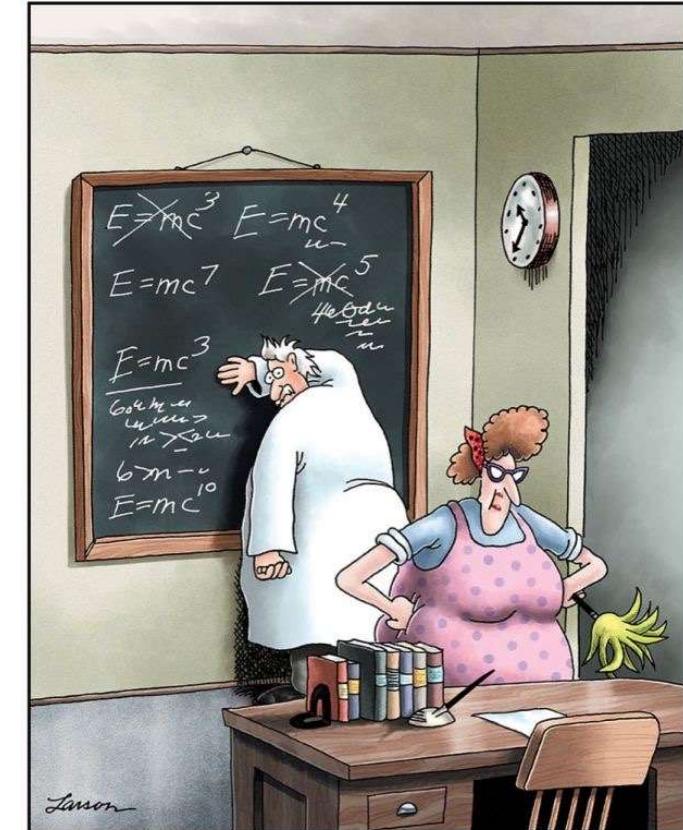
- “I get many ideas, and I throw away the bad ones”
  - Linus Pauling on how he managed to get TWO Nobels



Subbarao Kambhampati (కంధంపాటి సుబ్రామణి) @rao2z · Sep 3

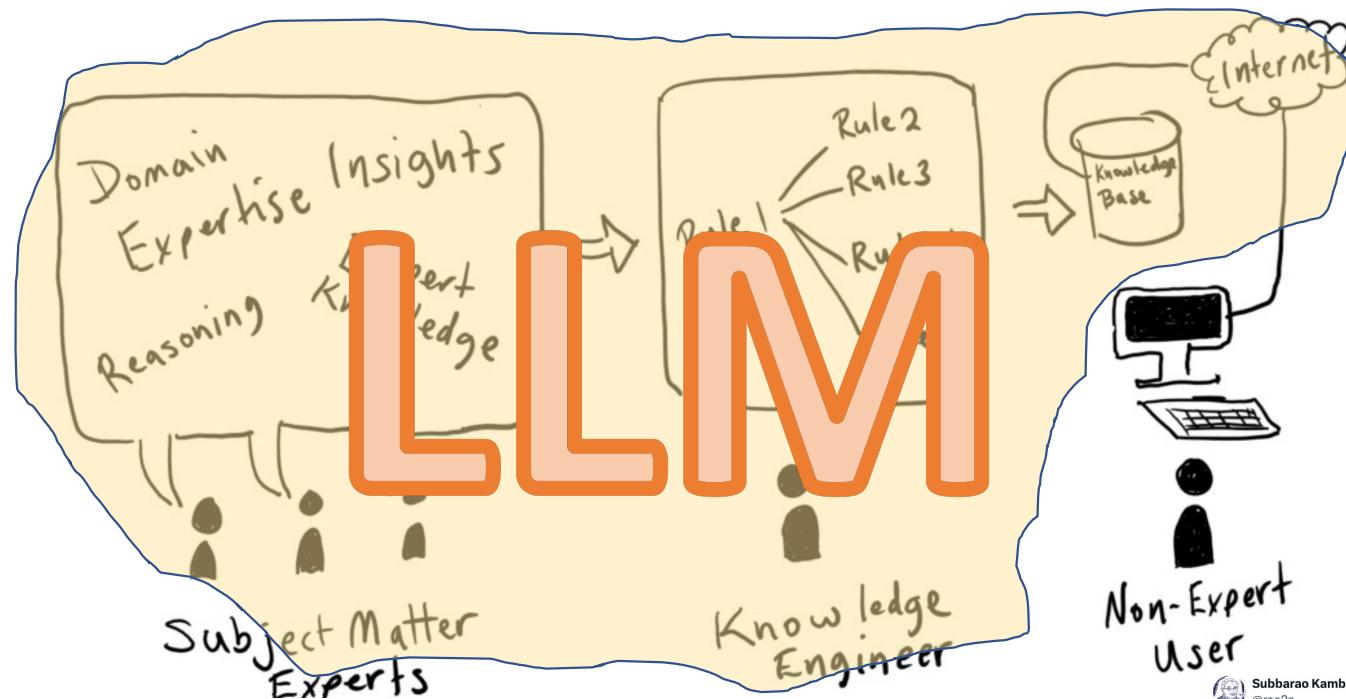
Replies to @rao2z @liron and @DynamicWebPaige

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/



"Now that desk looks better. Everything's squared away,  
yessir, squaaaaared 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 (ಕಂಫಂಪಾಟಿ ಸುಬ್ರಾಹಂಪತಿ)

...

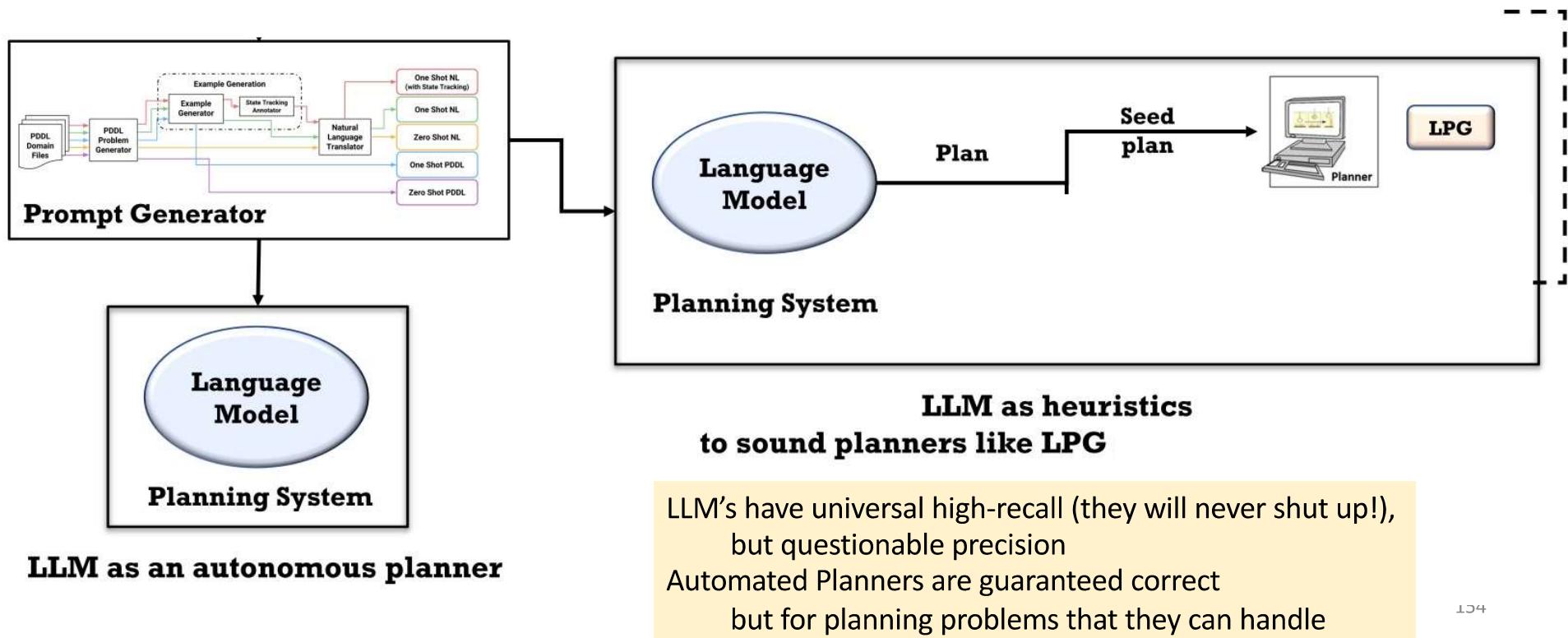
If you give what you know about a toy world to the computer, and have it solve new instances, it is #GOFAI cheating.

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.

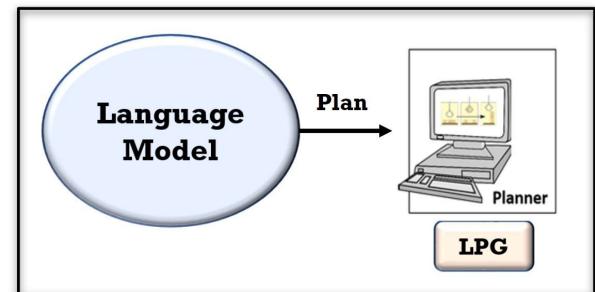
#AI Aphorisms

8:44 PM · Jul 23, 2022

# Workflow for using LLM's as Idea Generators (for External Sound Planners)



# LLMs as heuristics to sound planners



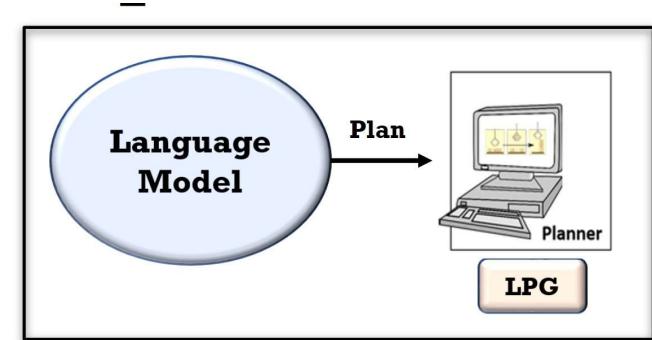
LLM generated plan as a heuristic to  
a sound planner like LPG

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.<br>Distance |
|------------|---------|-----------------------|------------------------|---------------------|-----------------------|------------------------|---------------------|-----------------------|
|            |         | Empty<br>Seed<br>Plan | Random<br>Seed<br>Plan | LLM<br>Seed<br>Plan | Empty<br>Seed<br>Plan | Random<br>Seed<br>Plan | LLM<br>Seed<br>Plan |                       |
|            |         | 15.8                  | 20.07                  | 14.5                | 8.45                  | 9.62                   | 11.7                | 7.22                  |
| 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 generated plan as a heuristic to  
a sound planner like LPG

## Computer Science &gt; 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

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 all 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 model-based planning/reasoning regimes towards more flexible knowledge, problem and preference specifications.

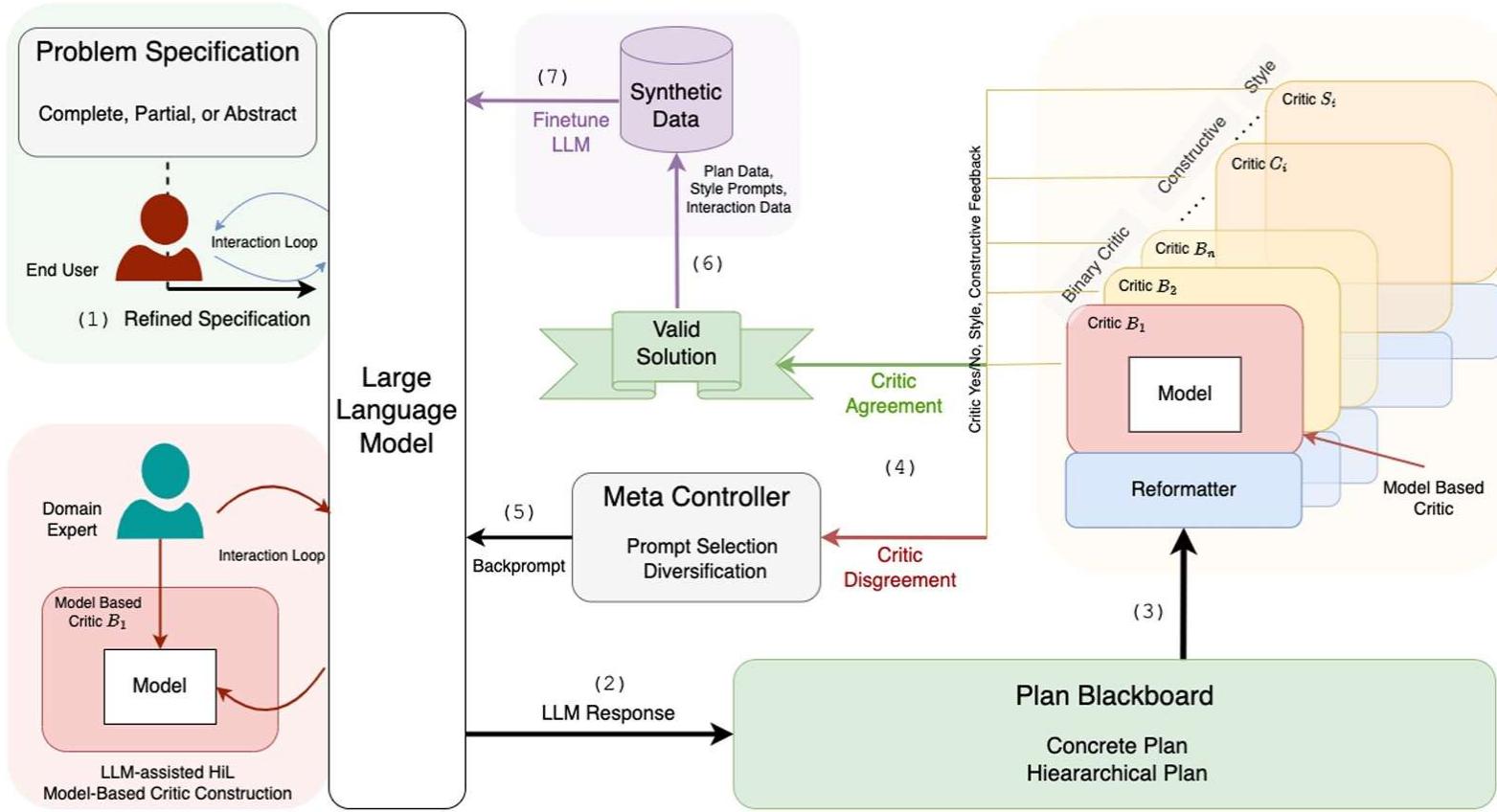
Subjects: [Artificial Intelligence \(cs.AI\)](#); [Machine Learning \(cs.LG\)](#)

Cite as: [arXiv:2402.01817 \[cs.AI\]](#)

(or [arXiv:2402.01817v2 \[cs.AI\]](#) for this version)

<https://doi.org/10.48550/arXiv.2402.01817> 

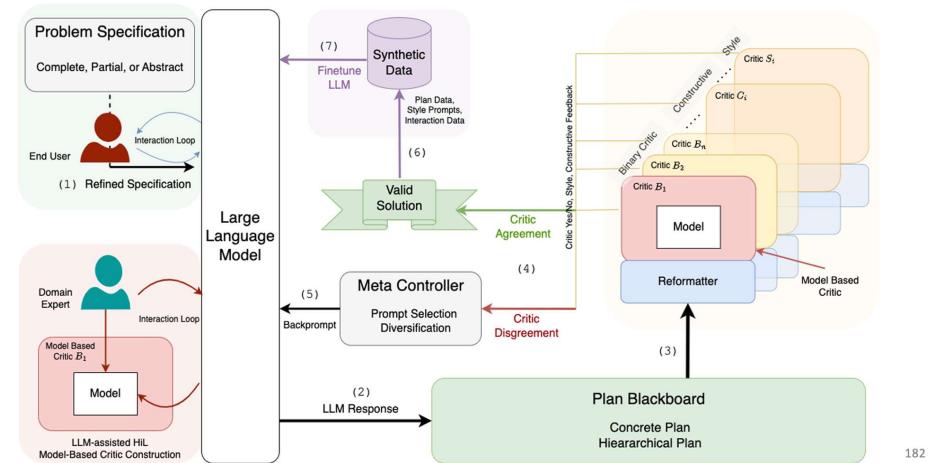
# LLM-Modulo: a principled framework for Planning wherein LLMs can play multiple constructive roles



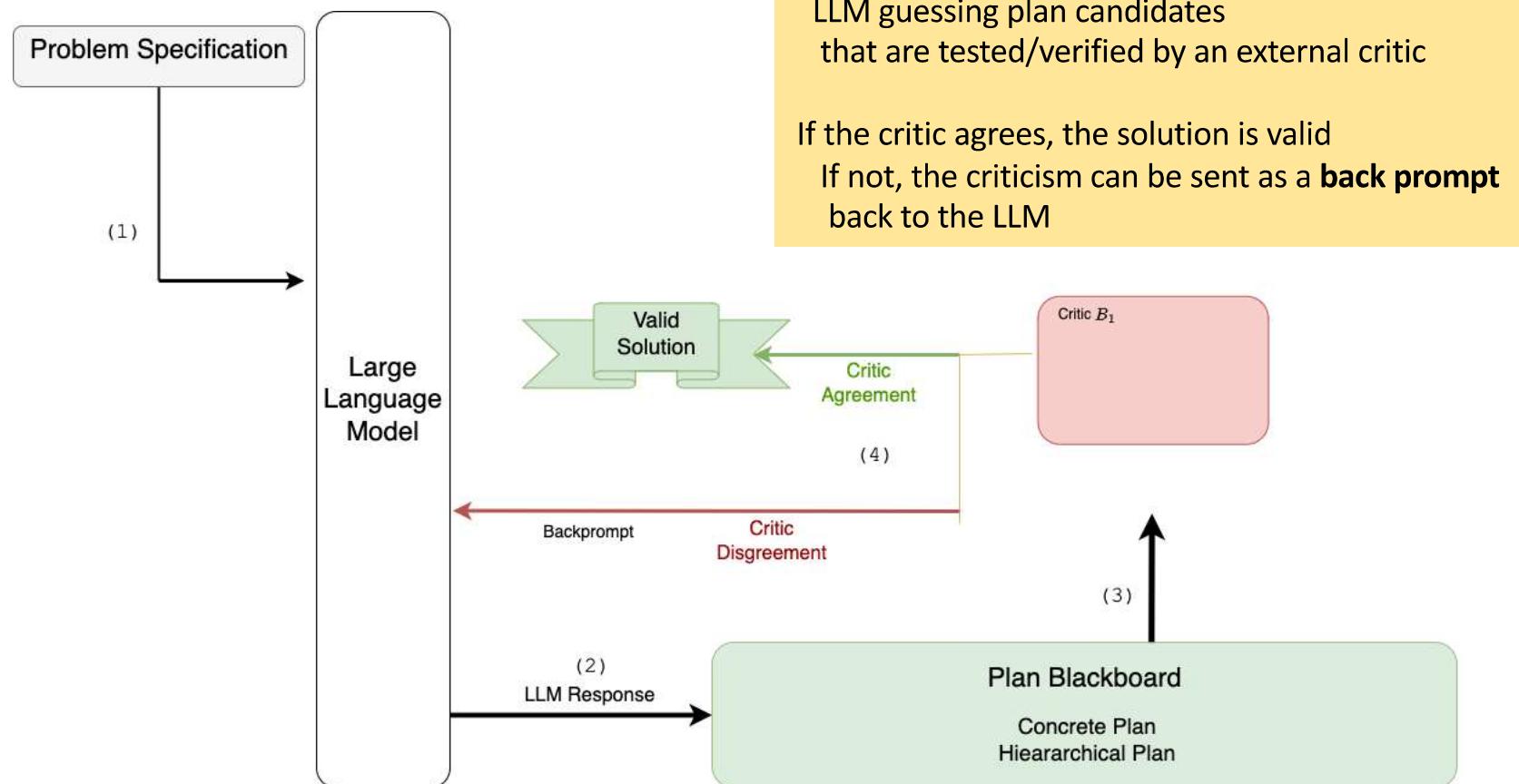
# Prefer Verifiers to Solvers! [Solver $\approx$ Verifier + Search]

- 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



# Bare Bones Generate-Test LLM-Modulo with External Verifier



# Automated Back-Prompting with External Verifiers

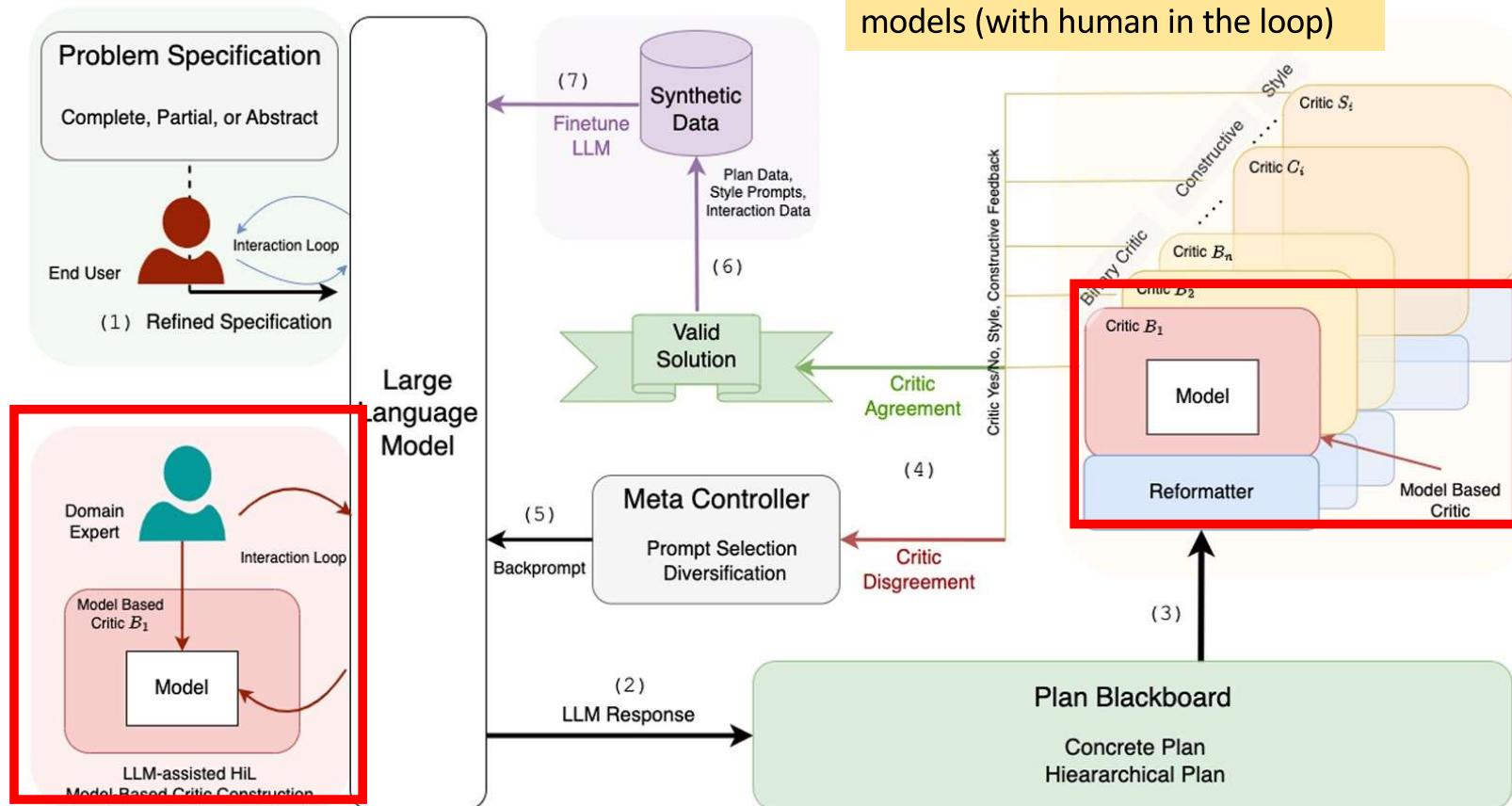
- Preliminary experiments show that back-prompting does improve LLM’s ability to produce plans in the Blocks World and Logistics
  - On the average over ~4 feedback rounds
- The performance in the Mystery BW still doesn’t improve— showing that the connection to commonsense domains/terms is critical for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

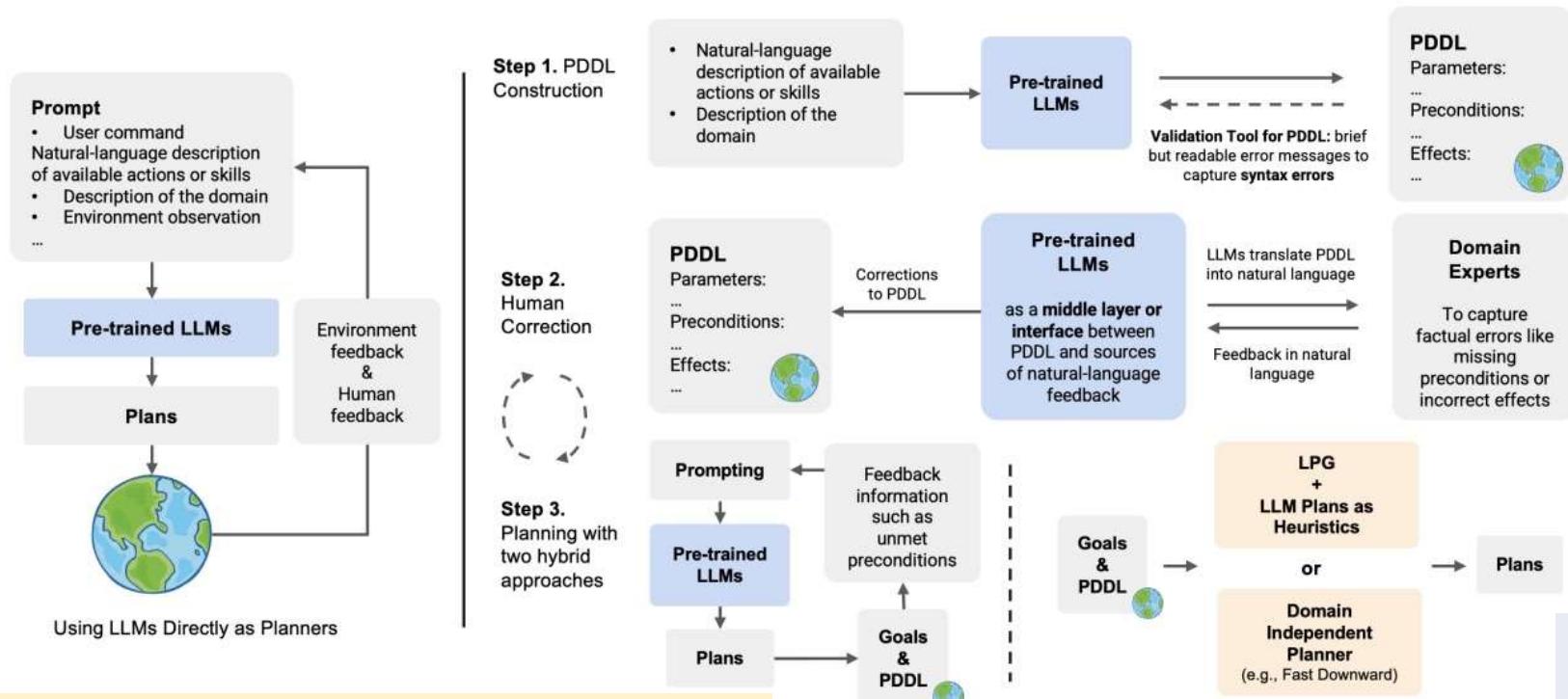
| Domain                  | I.C         |       | A.F.R |
|-------------------------|-------------|-------|-------|
|                         | GPT-4       | GPT-4 |       |
| <b>Blocksworld (BW)</b> | 41/50 (82%) |       | 3.68  |
| <b>Logistics</b>        | 35/50 (70%) |       | 3.31  |
| <b>Mystery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

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

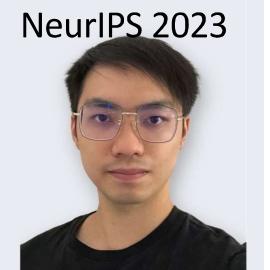
# LLMs for Extracting Planning Knowledge



# 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



# LLMs for constructing world models

- We utilize LLMs to extract a symbolic representation of the actions in the form of PDDL action models
- This intermediate output can be used with an external domain-independent planner to reliably search for feasible plans, or it can be used to validate and correct "heuristic" plans generated by an LLM planner.
- We also show that LLMs can also serve as an interface between PDDL and any feedback sources that can provide corrective feedback in natural language, such as humans and the PDDL validator in VAL

```
Instructions for the PDDL generation task
You are defining the preconditions and effects (represented in PDDL format) of an AI agent's
→ actions. Information about the AI agent will be provided in the domain description ...
One or two examples from other domains for illustrating the input and output formats
Here are two examples from the classical BlocksWorld domain for demonstrating the output format.
...
Here is the task.
A natural language description of the domain
Domain information: The AI agent here is a household robot that can navigate to various large and
↪ normally immovable furniture pieces or appliances in the house to carry out household tasks
↪ ...
A natural language description of the action
Action: This action enables the robot to toggle small appliances (like humidifiers and light
↪ bulbs) which are toggleable to switch them on ...
The dynamically updated list of predicates
You can create and define new predicates, but you may also reuse the following predicates:
1. (robot-at ?r - robot ?f - furnitureAppliance): true if the robot ?r is at the furniture or
↪ appliance ?f
2. (object-in-on ?o - householdObject ?f - furnitureAppliance): true if the object ?o is in or on
↪ the furniture or appliance ?f
...
Parameters:
-----
The LLM:
...
2. ?o - householdObject: the small appliance to be toggled on
...
Preconditions:
(and
  ...
  (not (appliance-on ?o))
)
Effects:
(and
  (appliance-on ?o)
)
New Predicates:
1. (appliance-on ?o - householdObject): true if the small appliance ?o is switched on
```

# Step 1. PDDL Construction

**Incrementally construct the domain model:** our algorithm generates PDDL models for each action separately, one at a time, by iterating over the set of actions.

Newly defined predicates are appended to the list of extracted predicates

Instructions for the PDDL generation task  
You are defining the preconditions and effects (represented in PDDL format) of an AI agent's actions. Information about the AI agent will be provided in the domain description ...  
One or two examples from other domains for illustrating the input and output formats  
Here are two examples from the classical BlocksWorld domain for demonstrating the output format.  
...  
Here is the task.  
A natural language description of the domain  
Domain information: The AI agent here is a household robot that can navigate to various large and normally immovable furniture pieces or appliances in the house to carry out household tasks  
...  
A natural language description of the action  
Action: This action enables the robot to toggle small appliances (like humidifiers and light bulbs) which are toggleable to switch them on ...  
The dynamically updated list of predicates  
You can create and define new predicates, but you may also reuse the following predicates:  
1. (robot-at ?r - robot ?f - furnitureAppliance): true if the robot ?r is at the furniture or appliance ?f  
2. (object-in-on ?o - householdObject ?f - furnitureAppliance): true if the object ?o is in or on the furniture or appliance ?f  
...  
Parameters:  
-----  
The LLM output  
1. ?r - robot: the robot performing the action  
2. ?o - householdObject: the small appliance to be toggled on  
3. ?f - furnitureAppliance: the furniture or appliance where the small appliance is located  
Preconditions:  
(and  
    (robot-at ?r ?f)  
    (object-in-on ?o ?f)  
    (robot-gripper-empty ?r)  
    (not (appliance-on ?o)))  
Effects:  
(and  
    (appliance-on ?o))  
New Predicates:  
1. (appliance-on ?o - householdObject): true if the small appliance ?o is switched on

Every time a new predicate is defined, the LLM is required to give the natural language description of it.

## Step 2. Correcting PDDL

### Sources of corrective feedback

- Syntax error: PDDL validations like the one in VAL
- Factual error: human domain experts

Corrective feedback is integrated by replaying and continuing the PDDL-construction dialogue.

#### The LLM:

The generated PDDL model for "heating food with a pan":

Parameters:

1. ?r - robot: the robot that will heat the food

...

Incorrect object type: stove

4. ?s - stove: the stove burner where the pan is placed

Preconditions:

(and

...

Unnecessary precondition  
(object-pickupable ?f)

)

...

### Corrective feedback in natural language:

There are some errors in the PDDL model:

1. There is an invalid object type 'stove' for the parameter ?s

2. There is an unnecessary precondition "the food to heat is pickup-able"

Please revise the PDDL model (and the list of predicates if needed) to fix the above errors (and other potentially similar errors).

# LLMs for constructing world models

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

| 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                        |
| Tyeworld  | 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.
```

GPT4: 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

# LLMs for constructing world models

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

Action description

```
This action enables the robot to toggle a small appliance on or off. It can be used to turn lights on or off, or to switch other small household objects on or off. For example, the robot can toggle a light bulb on or off.
```

GPT4: Toggle a small appliance off

```
Parameters:  
1. ?r - robot: the robot performing the action  
2. ?o - householdObject: the small appliance  
3. ?f - furnitureAppliance: the furniture or object being toggled  
  
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

| 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                        |

: produced by the LLMs for each of the domains.  
essively noisy, making it challenging to determine

```
Load a package into a truck. For example, load a package_1 into a truck_1.
```

Parameters:  
1. ?p - package: the package being loaded  
2. ?t - truck: the truck doing the loading  
  
Preconditions:  
(and  
 (package-at ?p ?l)  
 (truck-at ?t)  
 (not (truck-with ?t))  
)  
  
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

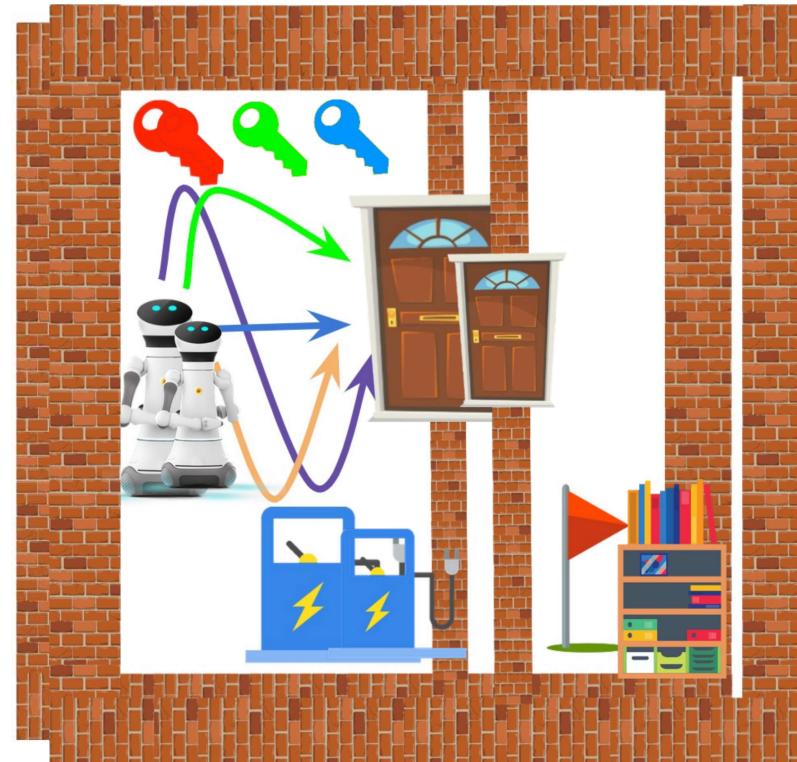
- Example: door-open, at-destination ...



Derive reward functions

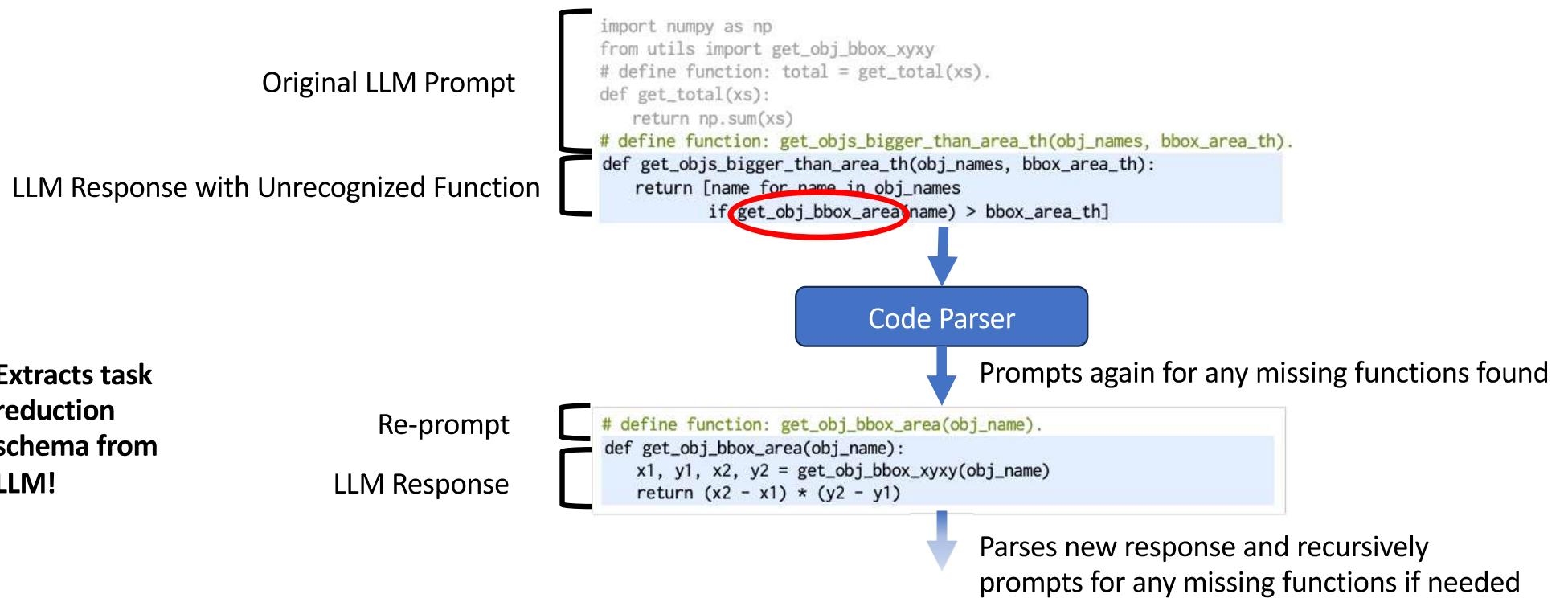
### Diverse set of skills learned per landmark

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



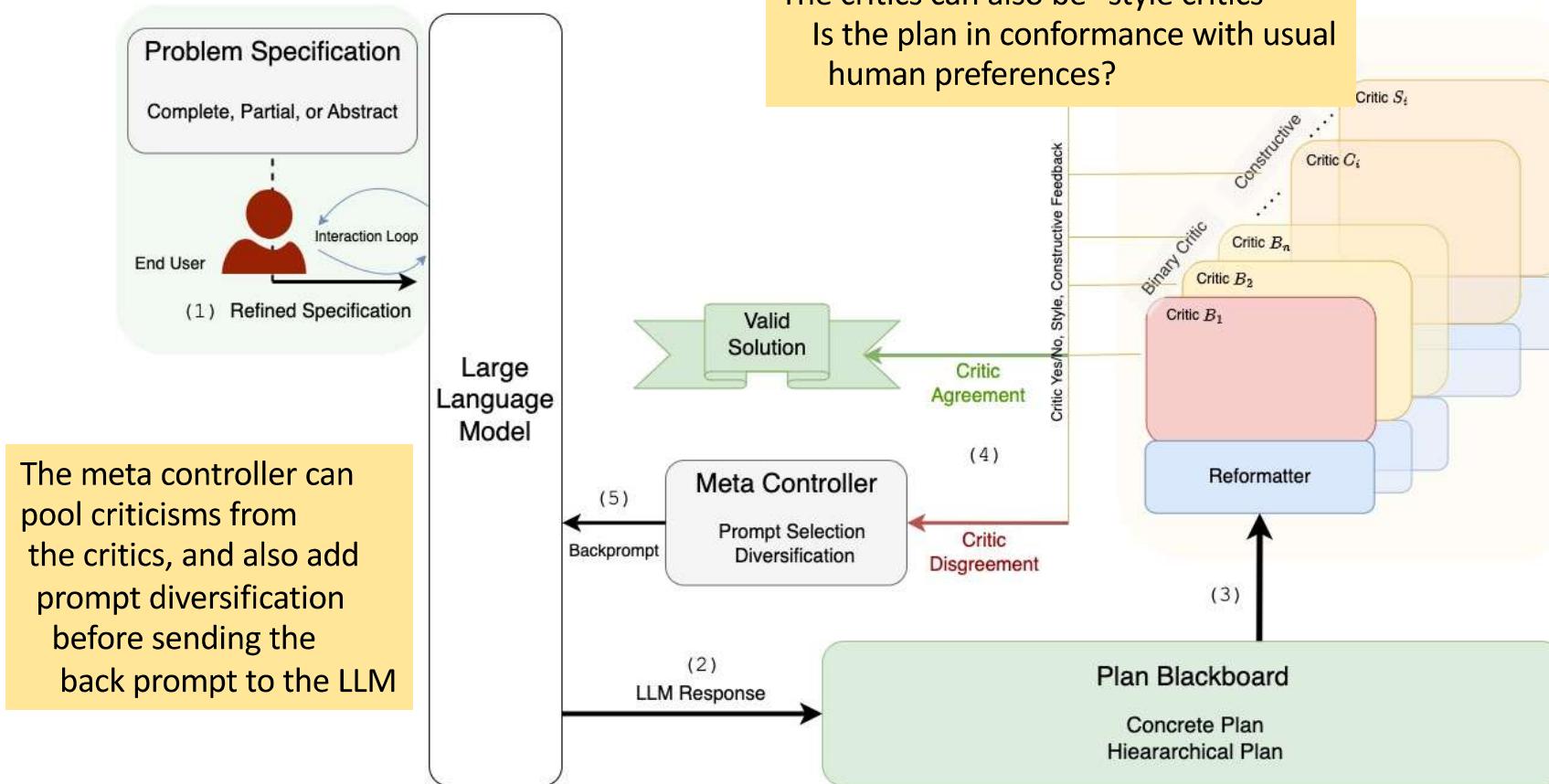
[ICML 2022]

# 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.

# LLM-Modulo with a Bank of critics



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!

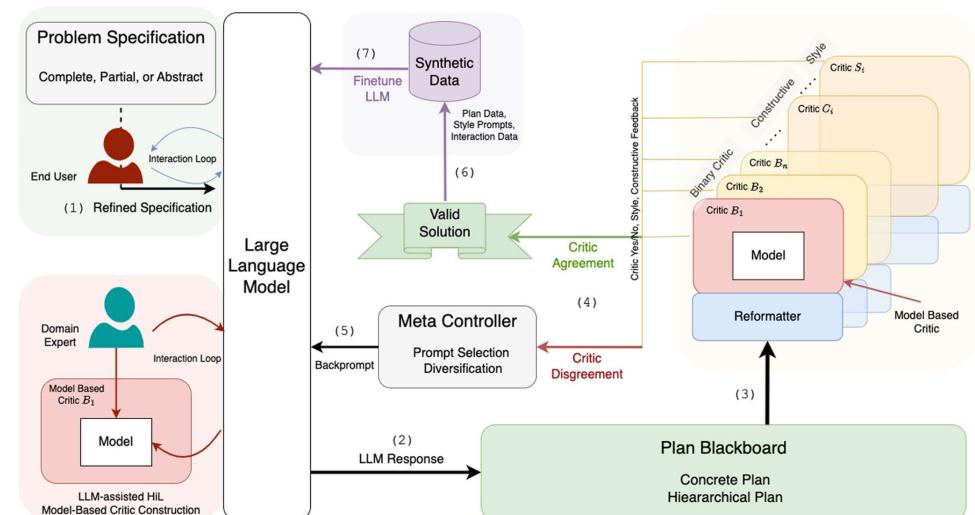
7:46 AM · Mar 26, 2024 · 8,635 Views

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
- 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



# 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**.

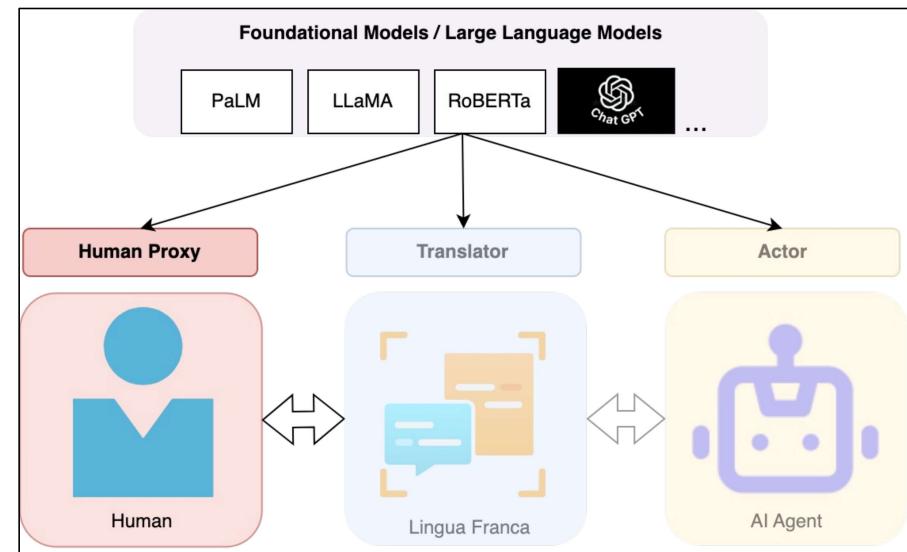


Figure: Different roles of an LLM in Human-AI interaction.

*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

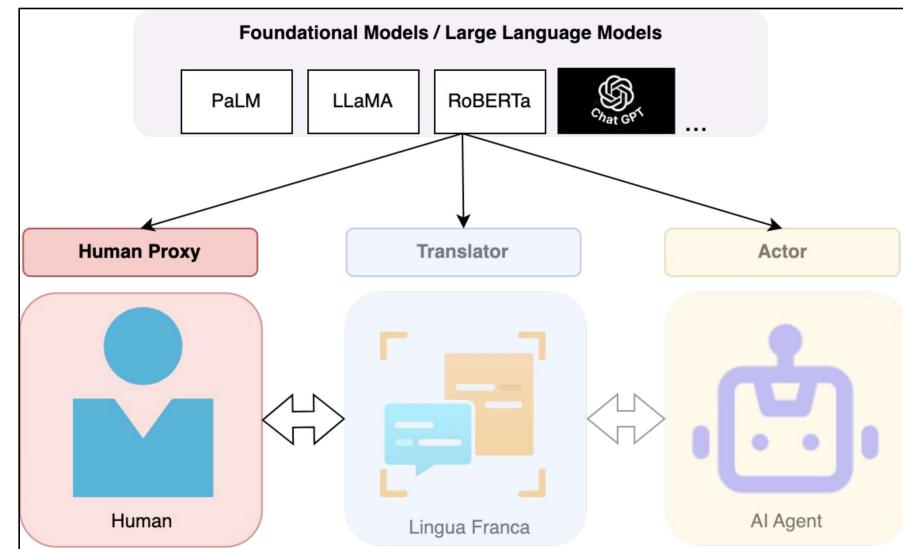
*Can LLMs capture human preferences?*

## 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



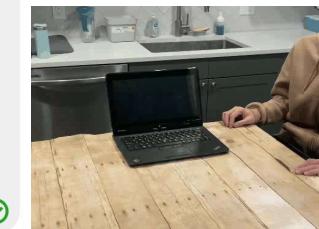
**GPT-4V:** The robot released the knife from a height that caused it to bounce upon hitting the cutting board, which is potentially dangerous due to the sharp blade.

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.

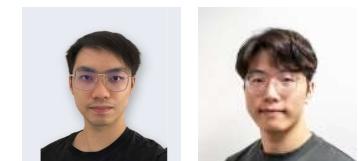
✗ (grounding error) The robot released the vessel from a height that could cause damage to the vessel or the stove if it were heavier or more fragile.

**GPT-4V:**

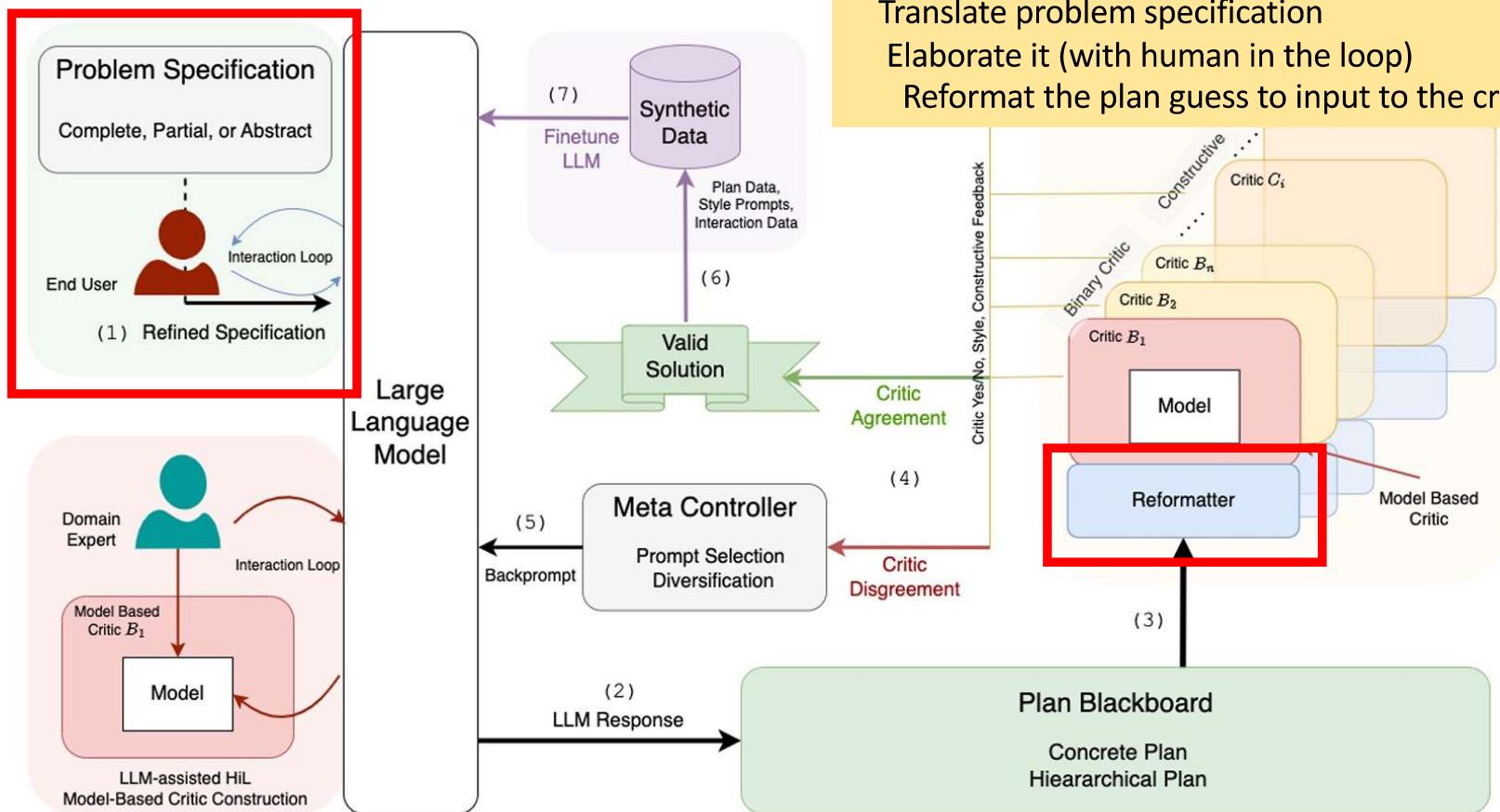
❓ (unoperationalizable) The robot caused pasta sauce to drip onto the counter while transferring the spoon from the jar to the bowl.

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



# LLMs for format change

- We have used them in the past to **translate** existing documents in natural language to formal representations

GPT3-to-Plan: Extracting Plans from Text  
[KEPS-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.

# Text to plan using GPT-3

**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.

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

| Model   | Action names |              |              | Action arguments |              |              |
|---------|--------------|--------------|--------------|------------------|--------------|--------------|
|         | 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            | <b>76.33</b> | 70.32        |
| STFC    | 62.66        | 67.39        | 62.75        | 38.79            | 43.31        | 42.75        |
| EASDRL  | 93.46        | 84.18        | 75.40        | <b>95.07</b>     | 74.80        | 75.02        |
| cEASDRL | <b>97.32</b> | <b>89.18</b> | <b>82.59</b> | 92.78            | 75.81        | <b>76.99</b> |
| <hr/>   |              |              |              |                  |              |              |
| GPT-3   |              |              |              |                  |              |              |
| Davinci | <b>86.32</b> | <b>58.14</b> | <b>43.36</b> | 22.90            | <b>29.63</b> | <b>22.25</b> |
| Curie   | 75.80        | 35.57        | 22.41        | <b>31.75</b>     | 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         |

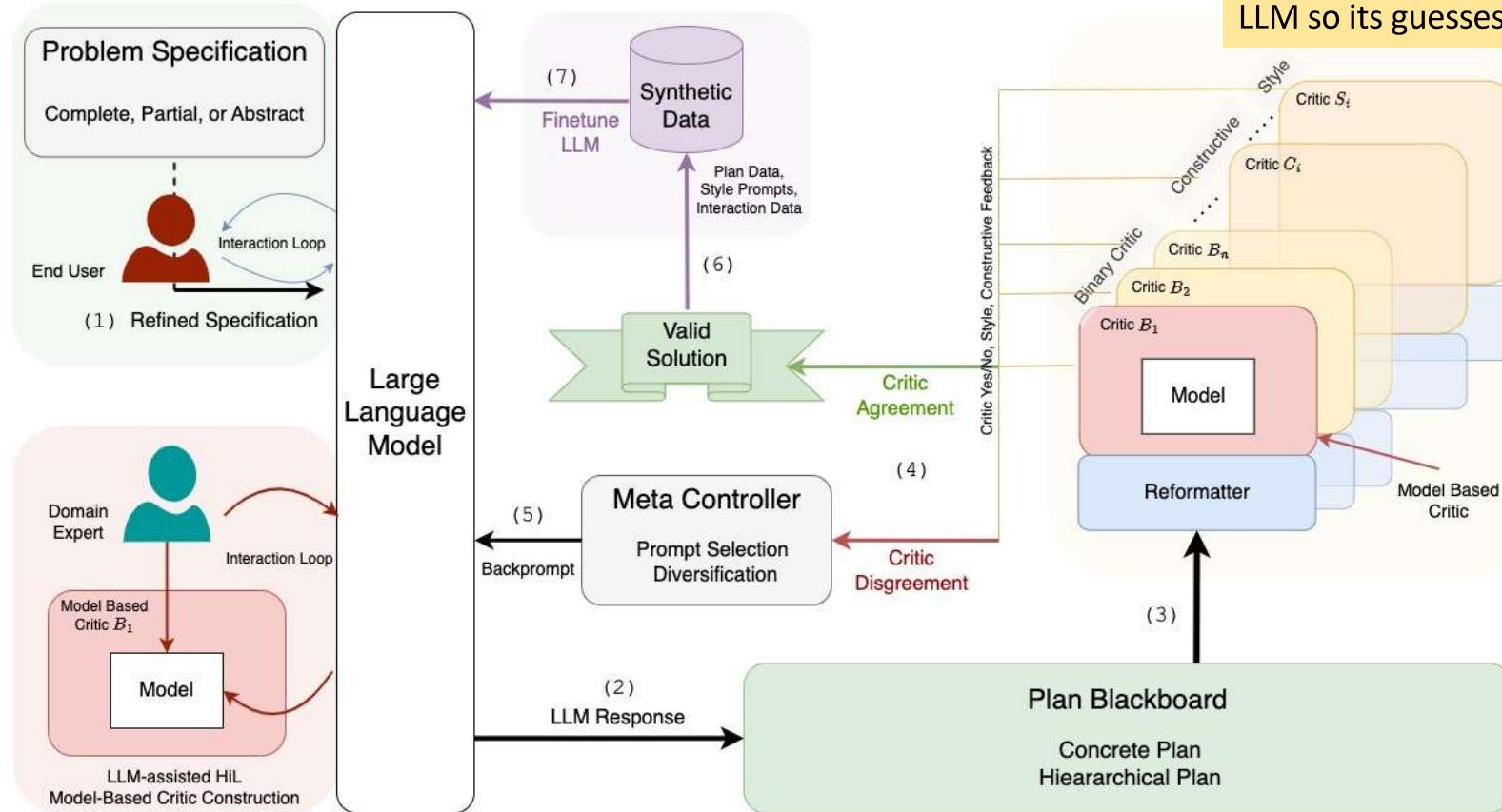
Table 3:  $F_1$  scores for all actions and their arguments across the WHS, CT and WHG datasets for the state-of-art sequence extraction models and GPT-3. State-of-art task-specific model  $F_1$  scores are extracted from Miglani and Yorke-Smith (2020); Feng, Zhuo, and Kambhampati (2018) and represent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

# Using LLMs to translate Goals specified in Natural Language

- Perhaps the least ambitious way to use LLMs in plan generation is to just have them convert the goals specified in natural language to formal representations (..and then use an actual external planner to solve the planning problem..)
  - A bit of a Rube Goldberg approach..
- Examples (not by us) include
  - Converting goal specifications to PDDL spec (LLM+P)
    - Which basically involves putting in parentheses at the lowest end..
  - Converting goal specification to STL spec (AutoTamp)

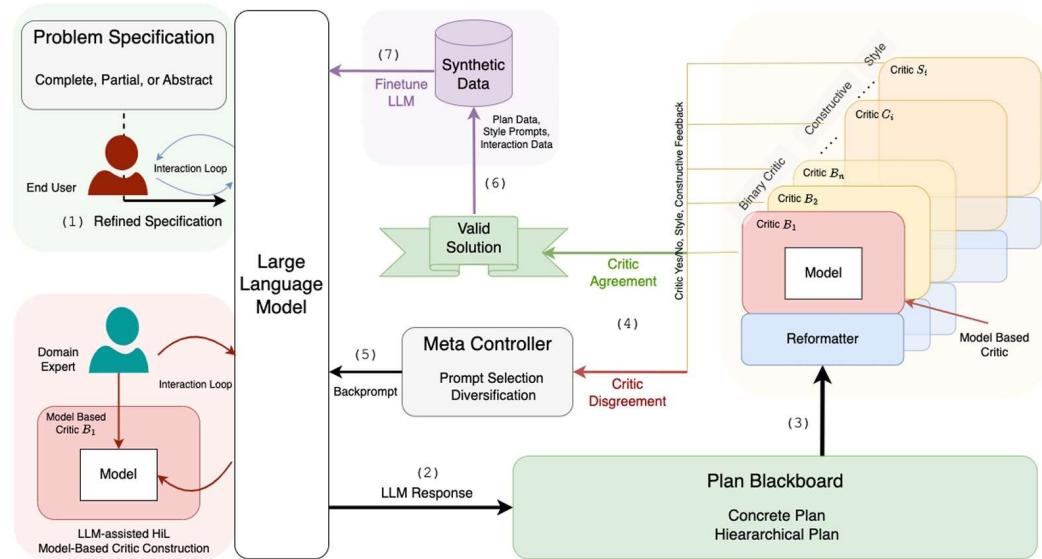
# Generating Synthetic Data (Self-Instruct LLM-Modulo Way)



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..

# 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
  - Act as style critics
  - 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

# 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.

# Talk Overview

- **Part 1**

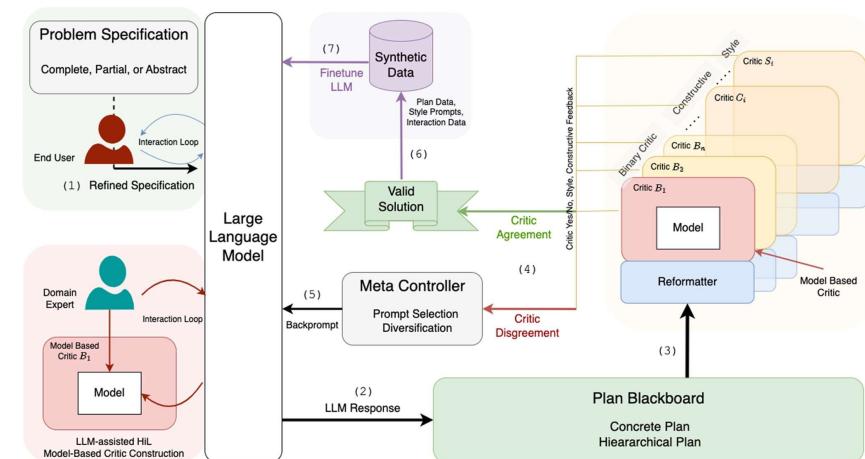
- Evaluating LLM Planning capabilities in Autonomous mode, including effect of
  - Prompting strategies (including Chain-of-thought)
  - Fine Tuning
  - Self Verification
- Understanding the contradictory claims in the literature

- **Part 2**

- Same 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 elaborators<sup>1</sup>

- **Part 3**

- Some thoughts on why people tend to ascribe planning/reasoning abilities to LLMs



# 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.)

# 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



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

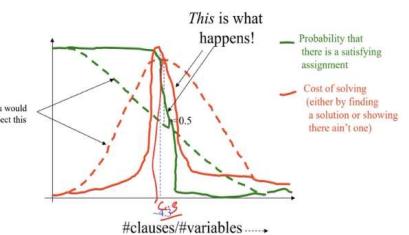
56 Retweets 12 Quotes 235 Likes 70 Bookmarks

Q Tweet reply! 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 leaner will manage to learn deep features corresponding to the  $y$  ratio-- #clauses/#variable .. 2/

Q 4 T 2 H 21 View Tweet analytics Reply  
Subbarao Kambhampati @ICML 7/25-28 @rao2z · Jul 29, 2022  
..and armed with  $y$ , it can also essentially figure out the sharp-threshold phenomenon w.r.t. to  $y$ , and should be able to predict with high certainty that the  $y < 4.3$  are satisfiable and  $y > 4.3$  are unsatisfiable. 3/

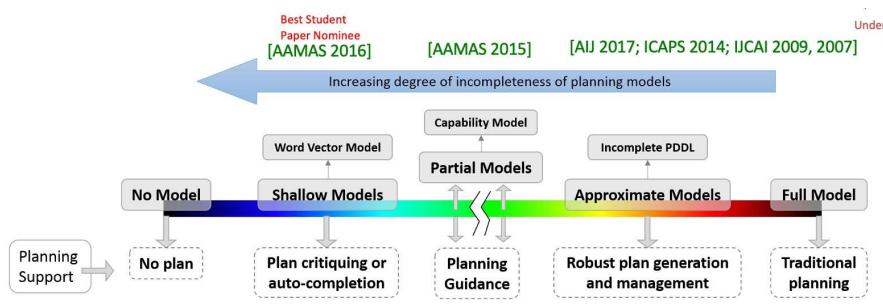
### Hardness of 3-sat as a function of #clauses/#variables



# 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..

# Epilogue

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

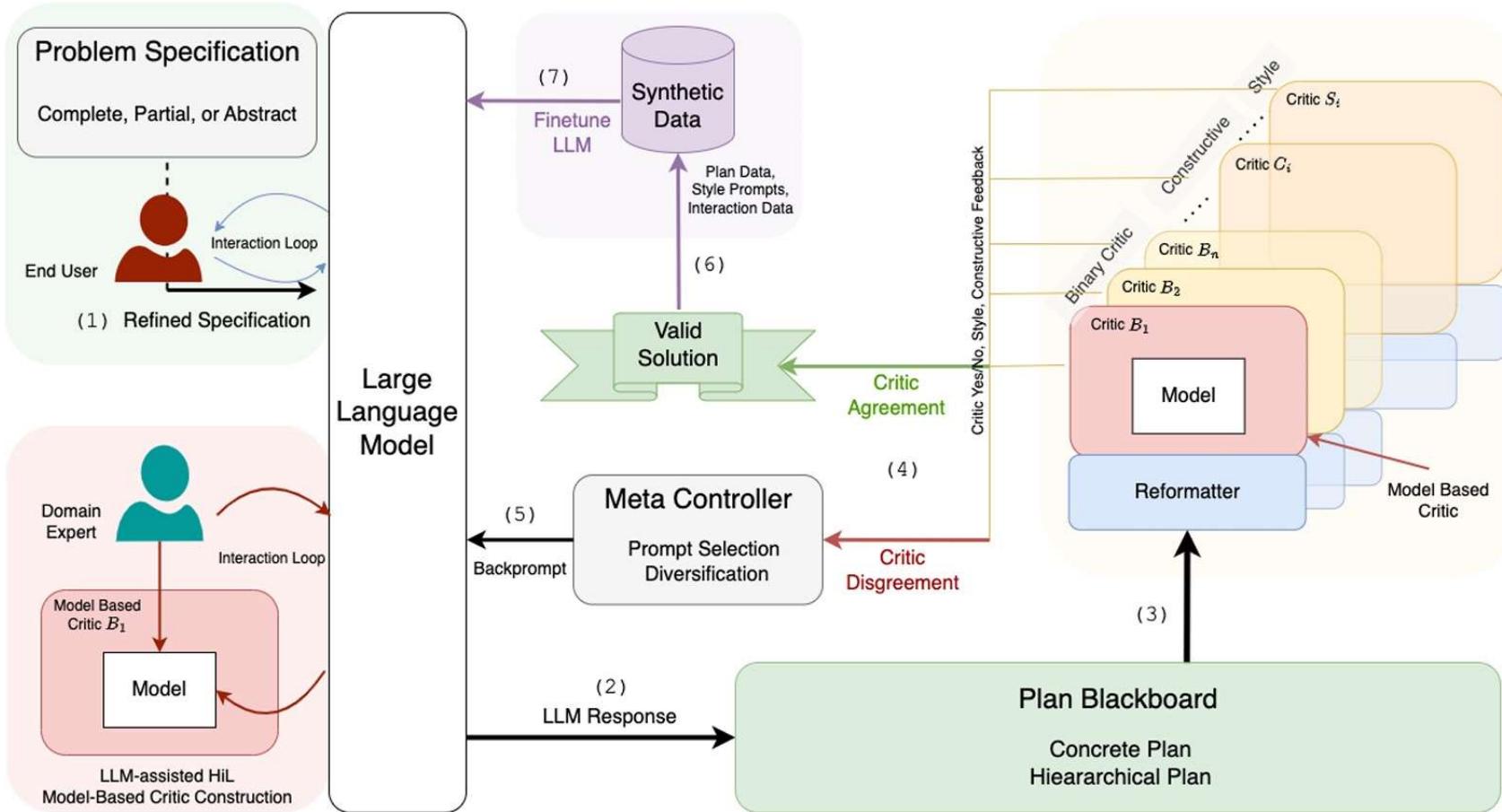
LLMs can't plan in Autonomous Modes (and many claims to the contrary are questionable)

- 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-Modulo Frameworks for Planning



# Claims on LLMs Reasoning/Planning Abilities

## Over-optimism

- LLMs can reason/plan
- With just the right “prompt”
- With letting them verify and critique their solutions

## Our Position

- LLMs can't reason/plan
- But they are approximate knowledge sources
- They can play much more meaningful roles in ***LLM-Modulo Settings***

## Over-pessimism

- LLMs can't reason/plan
- They can be translators at best
- Let external symbolic solvers handle the problem

# 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!

# Then how come LLMs are trumpeted as doing 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, the execution issues are left to the imagination of the user
  - or *has been pawned 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

**DRAFT**

Getting Started Schedule Tutorials Main Conference **DRAFT** Workshops Community **DRAFT** Help **DRAFT**

Subbarao

Browse

kambhampati  title author session

showing 3 of 3 papers

**On the Planning Abilities of Large Language Models - A Critical Investigation**  
Karthik Valmeeakam, Matthew Marquez, Sarah 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 Valmeeakam, Sarah 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 Valmeeakam, Matthew Marquez, Alberto Olmo, Sarah Sreedharan, Subbarao Kambhampati  
We, Dec 13, 09:45 – Poster Session 3



DOI: 10.1111/nys.15125

**ANNALS** OF THE NEW YORK ACADEMY OF SCIENCES

**THE SCIENCES**

**Can large language models reason and plan?**

Subbarao Kambhampati

School of Computing & Augmented Intelligence, Arizona State University

Correspondence Email: [rn@asu.edu](mailto:rn@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 steroids<sup>1</sup> 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.<sup>2</sup> 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 suggest 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.<sup>3</sup> Our results<sup>4</sup> 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 LLM's abilities to reason and plan. So we decided to repeat our tests on both GPT3.5 and GPT4.<sup>5</sup> Initial results showed that there was some improvement in the accuracy of generated plans from GPT3 to GPT3.5 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

arXiv > cs > arXiv:2402.08115

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 Valmeeakam, 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-criticize 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,

arXiv > cs > arXiv:2402.04210

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

arXiv > cs > arXiv:2401.05302

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 Bham bri, 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 exists 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 obfuscator 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.

arXiv > cs > arXiv:2402.01817

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 Valmeeakam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bham bri, Lucas Saldyt, 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 all that LLMs are good for in planning/reasoning tasks are mere translators of the problem specification from one syntactic form 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 f(b) 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 model-based planning/reasoning regimes 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)