

(How) Do LLMs Reason?

Subbarao Kambhampati
School of Computing & AI



Twitter: @rao2z

Talk on 6/25/25 @
Microsoft India Research
Academic Summit



Plan? (How) Do LLMs Reason?

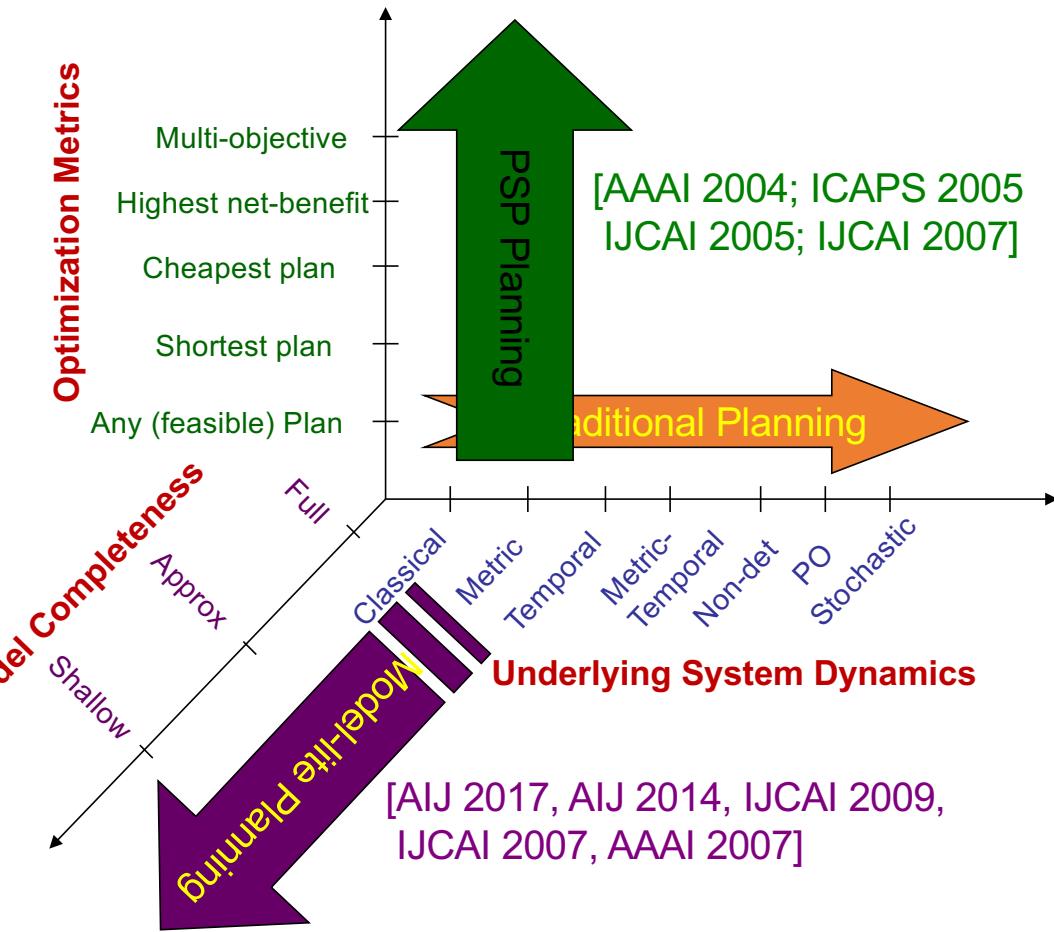
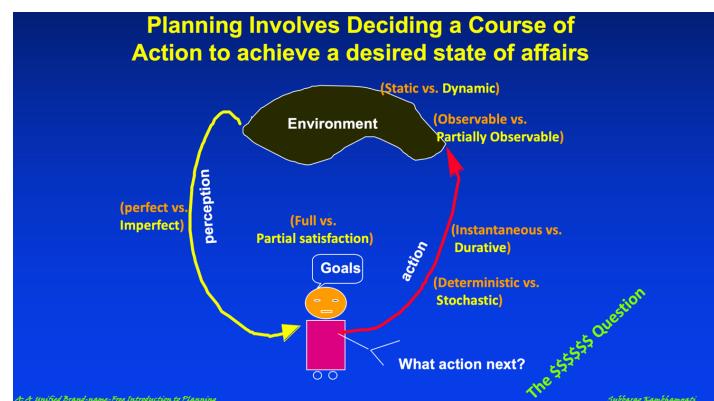
Subbarao Kambhampati
School of Computing & AI



Twitter: @rao2z

Talk on 6/25/25 @
Microsoft India Research
Academic Summit





Information Gathering;
Information Integration

RL with Simulator is Planning..

State of the Art in LLMs/LRMs

Standard LLMs

[GPT-x; Gemini; Llama..]

- Autoregressively trained on web corpora
 - Fine tuned (instruction tuning, RLHF) to make them more palatable
- At inference time, just complete the prompt

“Reasoning Models”

o1; R1; long CoT models..

- Standard LLMs +
 - Test-time scaling
 - Make LLM do more than just offer a single completion
 - (e.g. Generate-Test with verifier in the loop; LLM Modulo)
 - Post-training with synthetic data backed by verifiers
 - Supervised fine-tuning; RL-based

Multi-modality is an orthogonal issue

Results on PlanBench as of 8/27/2024

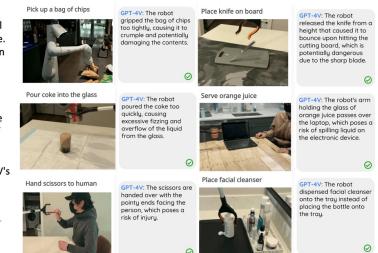
Foreshadowed Summary

Domain		Claude-3.5-Sonnet	Claude 3 (Opus)	GPT-4o	GPT-4	GPT-4 Turbo	LLaMA-3.1 405B	LLaMA-3 70B	Gemini Pro
Blocksworld	One shot	346/600 (57.6%)	289/600 (48.1%)	170/600 (28.3%)	206/600 (34.3%)	138/600 (23%)	284/600 (47.3%)	76/600 (12.6%)	68/600 (11.3%)
	Zero shot	329/600 (54.8%)	356/600 (59.3%)	213/600 (35.5%)	210/600 (34.6%)	241/600 (40.1%)	376/600 (62.6%)	205/600 (34.16%)	3/600 (0.5%)
Mystery Blocksworld	One shot	19/600 (3.1%)	8/600 (1.3%)	5/600 (0.83%)	26/600 (4.3%)	5/600 (0.83%)	21/600 (3.5%)	15/600 (2.5%)	2/500 (0.4%)
	Zero shot	0/600 (0%)	0/600 (0%)	0/600 (0%)	1/600 (0.16%)	1/600 (0.16%)	5/600 (0.8%)	0/600 (0%)	0/500 (0%)

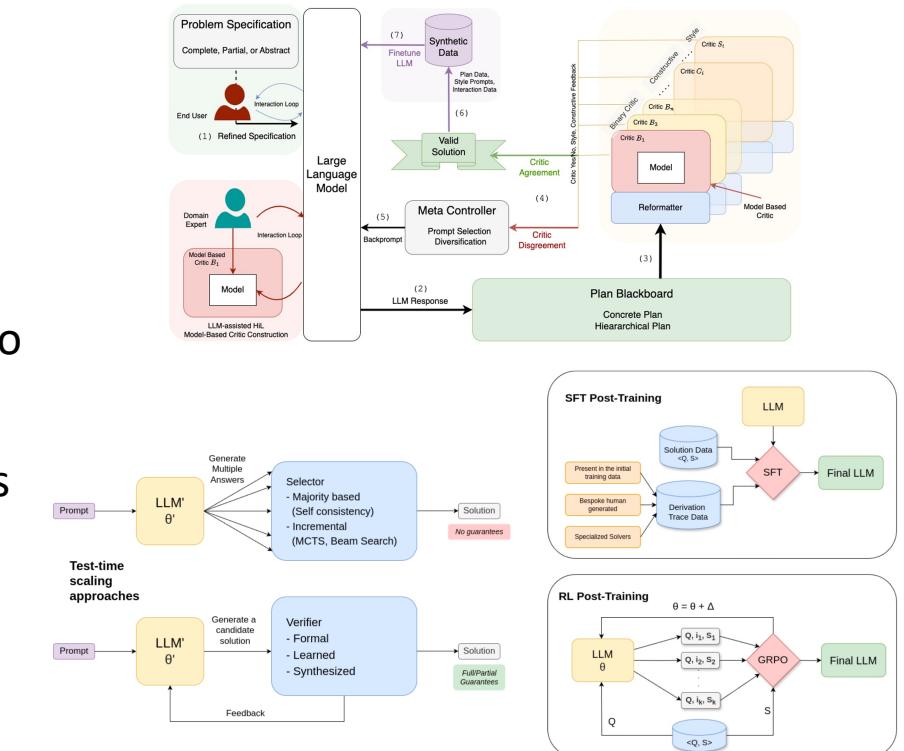
LLMs as Behavior Critics to catch undesirable robot behaviors
Can LLMs capture human preferences in embodied AI tasks?

- 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 Behaviors in Embodied AI Tasks
Lin Guan*, Yifan Zhou*, Dennis Liu, Yantian Zho, Heng Ben Amor, Subbarao Kambhampati, CDMR 2024



- LLMs trained just on web corpora were known to have severe limitations on planning/reasoning tasks
 - They could be good arbiters of style though..
- They could however be used as generators in a Generate-Test cycle to give robust planning performance; this LLM-Modulo is a form of test-time scaling
- The improved reasoning abilities can be “compiled” into base generator LLM with SFT or RL tuning on synthetic data
- The resulting LLMs still have no correctness guarantees (but are better generators)
 - Post-training *incrementally compiles* verifier signal into the LLM
- The anthropomorphization of intermediate tokens as “reasoning traces” is questionable



Reasoning & Planning Abilities of LLMs/LRMs

Lecture 1

Subbarao Kambhampati
ASU Arizona State University

Twitter: @rao2z

52:52

Reasoning & Planning Abilities of LLMs/LRMs

Lecture 2

Subbarao Kambhampati
ASU Arizona State University

Twitter: @rao2z

52:27

Reasoning & Planning Abilities of LLMs/LRMs

Lecture 3

Subbarao Kambhampati
ASU Arizona State University

Twitter: @rao2z

55:49

<https://bit.ly/3ZFtist>



Planning & Reasoning Abilities of LLMs/LRMs: The Agenda

- Lecture 1: Big Picture
 - A perspective on LLMs and their approximate retrieval
 - Understanding reasoning/planning (in)abilities of standard LLMs
 - Ideas for improving: LLMs as generators; Test Time Scaling as Generate-Test (“LLM-Modulo”); Post-training as Generate-Test-Compile during training
- Lecture 2: Drill down on planning abilities of standard LLMs
 - Out of the box, with Chain-of-thought, with ReACT, with self-verification etc.
- Lecture 3: Drill down on test-time scaling (LLM-Modulo); Post-training approaches (Supervised fine tuning vs. RL)
 - Understanding some continuing misconceptions

LLM's Can't Plan;

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

- LLMs can't do planning in autonomous mode [NeurIPS 23]



Yann LeCun ✅

@ylecun

...

LLMs still can't plan.

Llama-3.1-405b and Claude can plan a bit on Blocksworld.

GPT4 and Gemini not so much.

Performance is abysmal for everyone on Mystery Blocksworld.



Subbarao Kambhampati (కంభంపాటి సుబ్రామణి) ✅ @rao2z · 21h

The latest LLM performance on PlanBench (arxiv.org/abs/2206.10498) via @karthikv792. While they all bite the dust on Mystery BW, it is notable that the 400bn free open source **model** LLaMA3.1 does quite well compared to the costly closed models marketed as masters of reasoning..

[Show this thread](#)

Results on PlanBench as of 8/27/2024

Domain		Claude-3.5-Sonnet	Claude 3 (Opus)	GPT-4o	GPT-4	GPT-4 Turbo	LLaMA-3.1 405B	LLaMA-3 70B	Gemini Pro
Blocksworld	One shot	346/600 (57.6%)	289/600 (48.1%)	170/600 (28.3%)	206/600 (34.3%)	138/600 (23%)	284/600 (47.3%)	76/600 (12.6%)	68/600 (11.3%)
	Zero shot	329/600 (54.8%)	356/600 (59.3%)	213/600 (35.5%)	210/600 (34.6%)	241/600 (40.1%)	376/600 (62.6%)	205/600 (34.16%)	3/600 (0.5%)
Mystery Blocksworld	One shot	19/600 (3.1%)	8/600 (1.3%)	5/600 (0.83%)	26/600 (4.3%)	5/600 (0.83%)	21/600 (3.5%)	15/600 (2.5%)	2/500 (0.4%)
	Zero shot	0/600 (0%)	0/600 (0%)	0/600 (0%)	1/600 (0.16%)	1/600 (0.16%)	5/600 (0.8%)	0/600 (0%)	0/500 (0%)

12:14 PM · Sep 8, 2024 · 131.4K Views

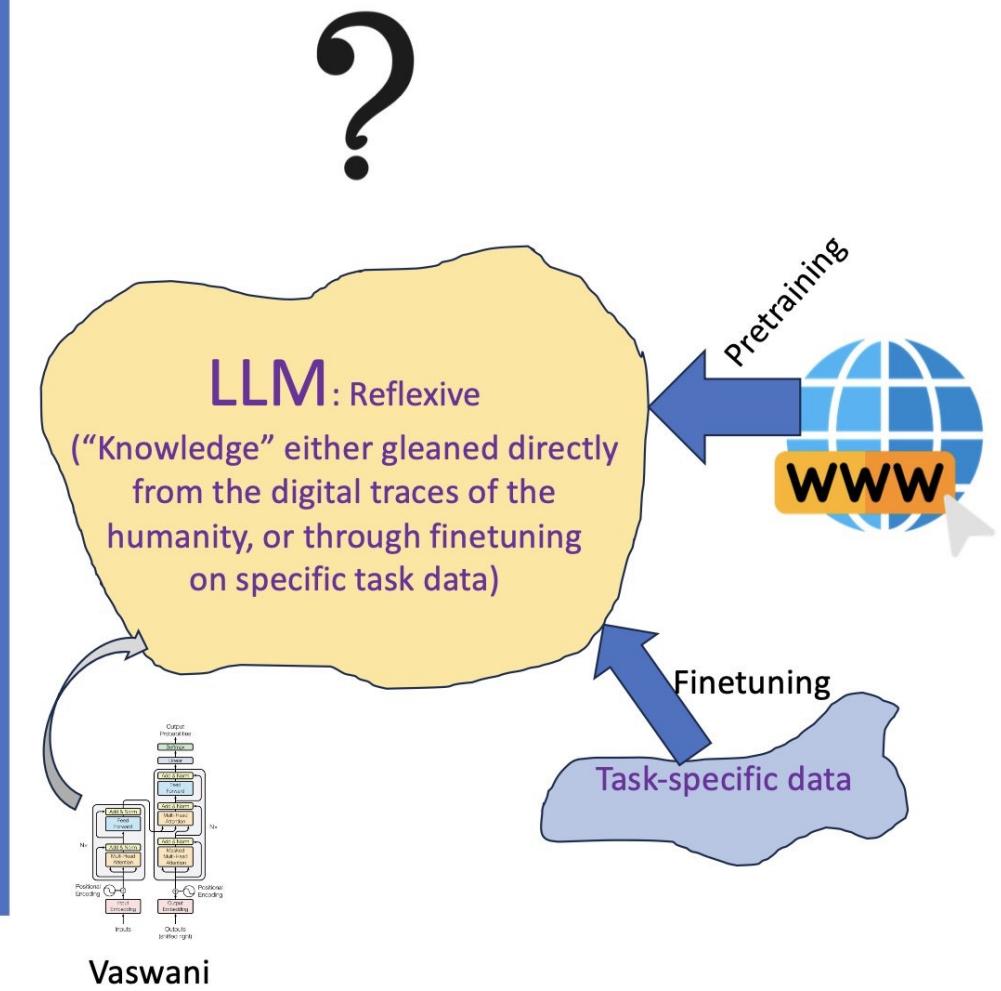
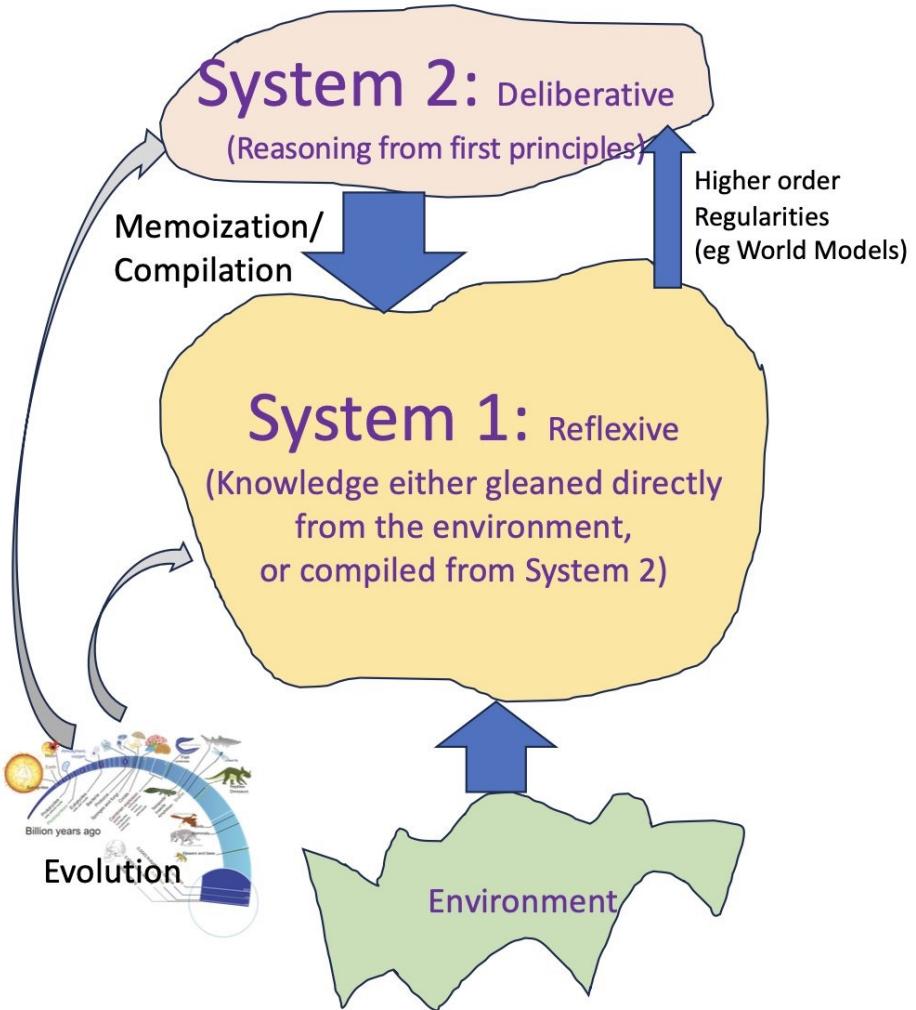
63

98

721

285

↑



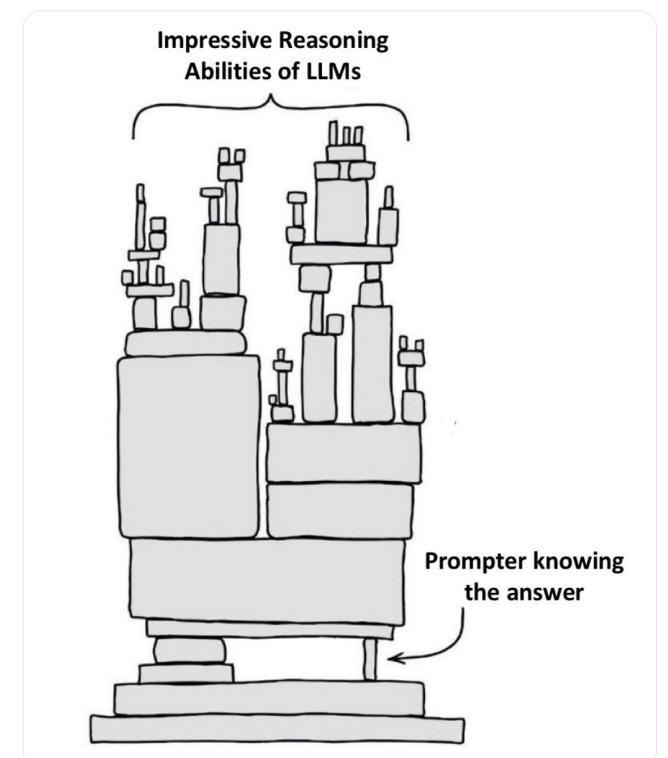
Style vs. Content Form vs. Factuality

- 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*..)
 - [A big part of the success of foundation models in Robotics has really been the ability to learn style from demos to do *trajectory planning*..]
- 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 సుబ్రామణ్ణు) ✅ @rao2z

The tl;dr 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/171000000000000000 ...)



Subbarao Kambhampati (కంభంపాటి సుబ్రామణ్ణు) ✅ @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 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.)

LLM's Can't Plan;

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

- LLMs can't do planning in autonomous mode [NeurIPS 23]



Yann LeCun ✅

@ylecun

...

LLMs still can't plan.

Llama-3.1-405b and Claude can plan a bit on Blocksworld.

GPT4 and Gemini not so much.

Performance is abysmal for everyone on Mystery Blocksworld.



Subbarao Kambhampati (కంభంపాటి సుబ్రామణి) ✅ @rao2z · 21h

The latest LLM performance on PlanBench (arxiv.org/abs/2206.10498) via @karthikv792. While they all bite the dust on Mystery BW, it is notable that the 400bn free open source **model** LLaMA3.1 does quite well compared to the costly closed models marketed as masters of reasoning..

[Show this thread](#)

Results on PlanBench as of 8/27/2024

Domain		Claude-3.5-Sonnet	Claude 3 (Opus)	GPT-4o	GPT-4	GPT-4 Turbo	LLaMA-3.1 405B	LLaMA-3 70B	Gemini Pro
Blocksworld	One shot	346/600 (57.6%)	289/600 (48.1%)	170/600 (28.3%)	206/600 (34.3%)	138/600 (23%)	284/600 (47.3%)	76/600 (12.6%)	68/600 (11.3%)
	Zero shot	329/600 (54.8%)	356/600 (59.3%)	213/600 (35.5%)	210/600 (34.6%)	241/600 (40.1%)	376/600 (62.6%)	205/600 (34.16%)	3/600 (0.5%)
Mystery Blocksworld	One shot	19/600 (3.1%)	8/600 (1.3%)	5/600 (0.83%)	26/600 (4.3%)	5/600 (0.83%)	21/600 (3.5%)	15/600 (2.5%)	2/500 (0.4%)
	Zero shot	0/600 (0%)	0/600 (0%)	0/600 (0%)	1/600 (0.16%)	1/600 (0.16%)	5/600 (0.8%)	0/600 (0%)	0/500 (0%)

12:14 PM · Sep 8, 2024 · 131.4K Views

63

98

721

285



LLM's Can't Plan;

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

- LLMs can't do planning in autonomous mode [NeurIPS 23]
- CoT, ReACT, Fine Tuning etc. don't help that much (as they don't generalize enough) [NeurIPS 24] [TMLR, 2025]



Published in Transactions on Machine Learning Research (05/2025)

Do Think Tags Really Help LLMs Plan? A Critical Evaluation of ReAct-Style Prompting

Siddhant Bhambhani^{*} School of Computing and AI Arizona State University

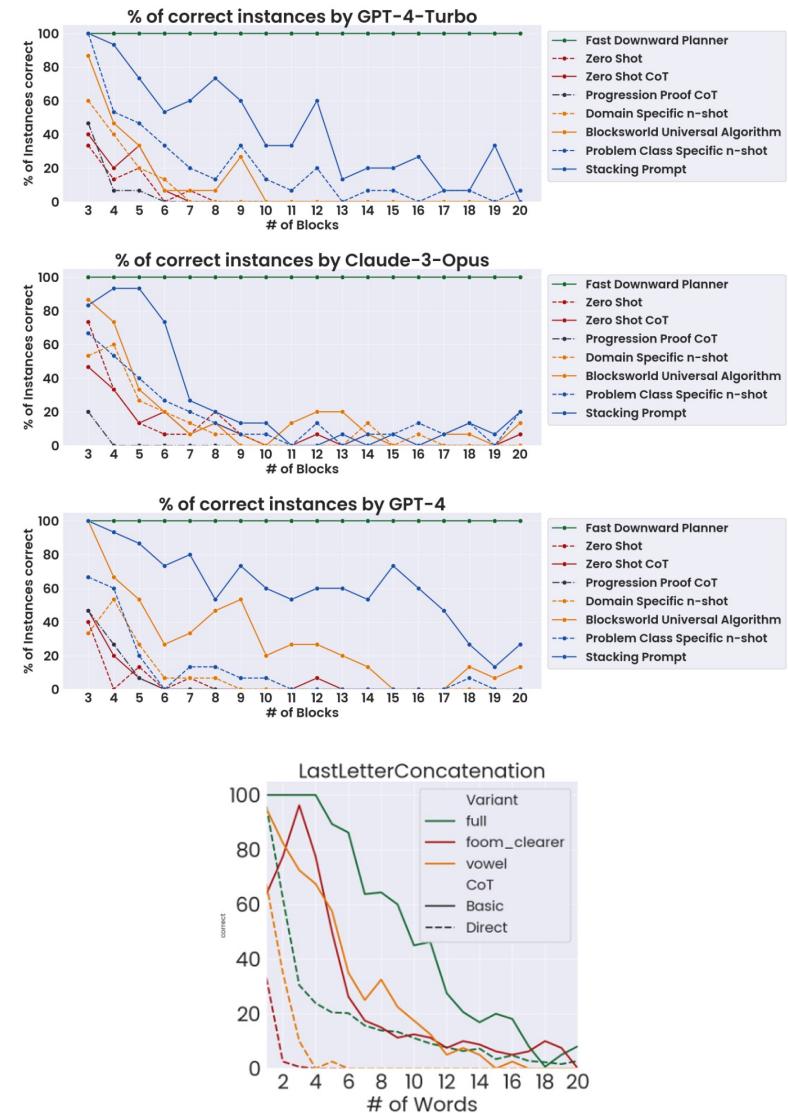
Mudit Verma^{*} School of Computing and AI Arizona State University

Subbarao Kambhampati School of Computing and AI Arizona State University

Reviewed on OpenReview: <https://openreview.net/for?id=rfAFM9nB>

Abstract

The reasoning abilities of Large Language Models (LLMs) remain a topic of considerable interest and debate. Among the original papers arguing for emergent reasoning abilities of LLMs, ReAct became particularly popular by claiming to tease out LLM reasoning abilities with minimal annotation. In this work, we show that ReAct-style prompting does not actually improve reasoning abilities of LLMs.



This week in The History of AI at AIWS.net – John McCarthy proposed the ‘advice taker’ in 1959

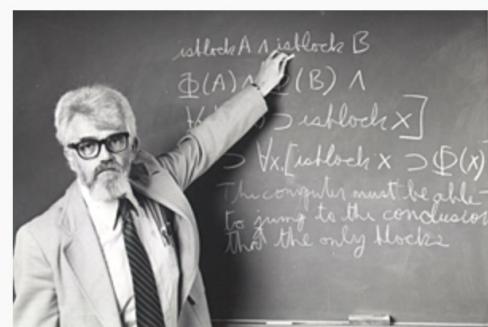
July 22, 2022

This week in The History of AI at AIWS.net – John McCarthy proposed the “advice taker” in his paper “Programs with Common Sense.” This hypothetical program was the first to use logic to represent information. The paper was published in 1959.

John McCarthy was an American computer scientist. He, along with Minsky and others, co-wrote the proposal for the Dartmouth Conference, and participated in it. McCarthy also developed the Lisp programming language. He spent most of his career at Stanford after working at Dartmouth and MIT. McCarthy won the Turing Award in 1971.

McCarthy wrote and published the paper while he was a part of the Computer Science at Stanford University. It can be viewed and downloaded [here](#).

The HAI initiative considers this an event in the history of AI due to the paper being one of the earliest developments in artificial intelligence. Furthermore, John McCarthy is one of the pivotal figures in the growth of the field, having been there at the Dartmouth Conference.



Subbarao Kambhampati (కంభంపాటి సుబ్రాపతి)

@rao2z
My new joke about Chain of Thought(lessness) 😊

*Give an LLM fish and feed it for a day
Teach it how to fish two fish and feed it for two days
Teach it how to fish three fish and feed it for three days
Teach it how to fish four fish and feed it for four days
Teach it how to fish five fish and feed it for five days
Teach it how to fish six fish and feed it for six days
Teach it how to fish seven fish and feed it for seven days.*



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^{†*}
Dale Schuurmans
[†]Google Research

Chain-of-
ous natu-
asks wh-
prompts.
a novel p-
egy is to
and then
answers t-
lated to s-
ing reveal
problems
code-d-
the comp-
split) wil-
16% accu-
because t-
are train-
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
{hanxuhu,zhangyue}@westlake.edu.cn
{hylu,wlam}@se.cuhk.edu.hk
v1zhai7@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
interactive-
high-level
set of acti-
from exp-
pre-traine-
decompos-
ever, the p-
actions.
semantic
VirtualHo
executabl
trade-off
extracting

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

Machel Reid
Google Research^{*}

Yutaka Matsuo
The University of Tokyo

Shixiang Shane Gu
Google Research, Brain Team

Yusuke Iwasawa
The University of Tokyo

Large Language Models are Zero-Shot Reasoners

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 arithmetic (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.

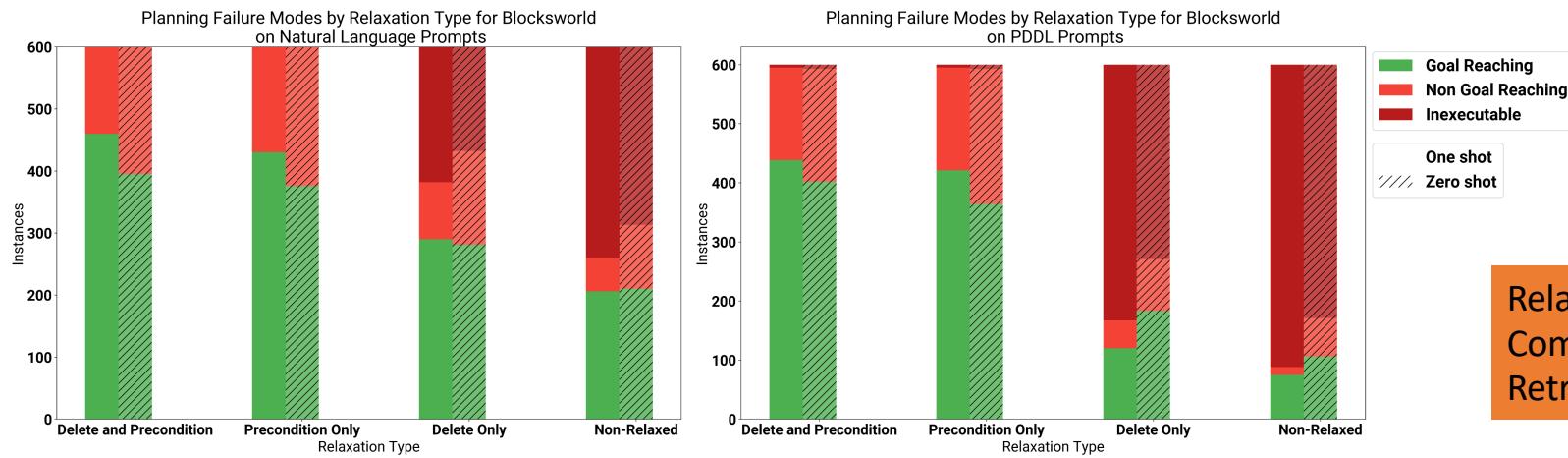
AI Planners will give formal guarantees, but only accept problems expressible in their language.

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..

How do we go from System 1 LLMs to System 1+2 LRM^s?

Test Time Inference (Test-time Scaling; adaptive computation...)

Improve the generator by post-training

Spectrum of Ideas for Test-Time Scaling

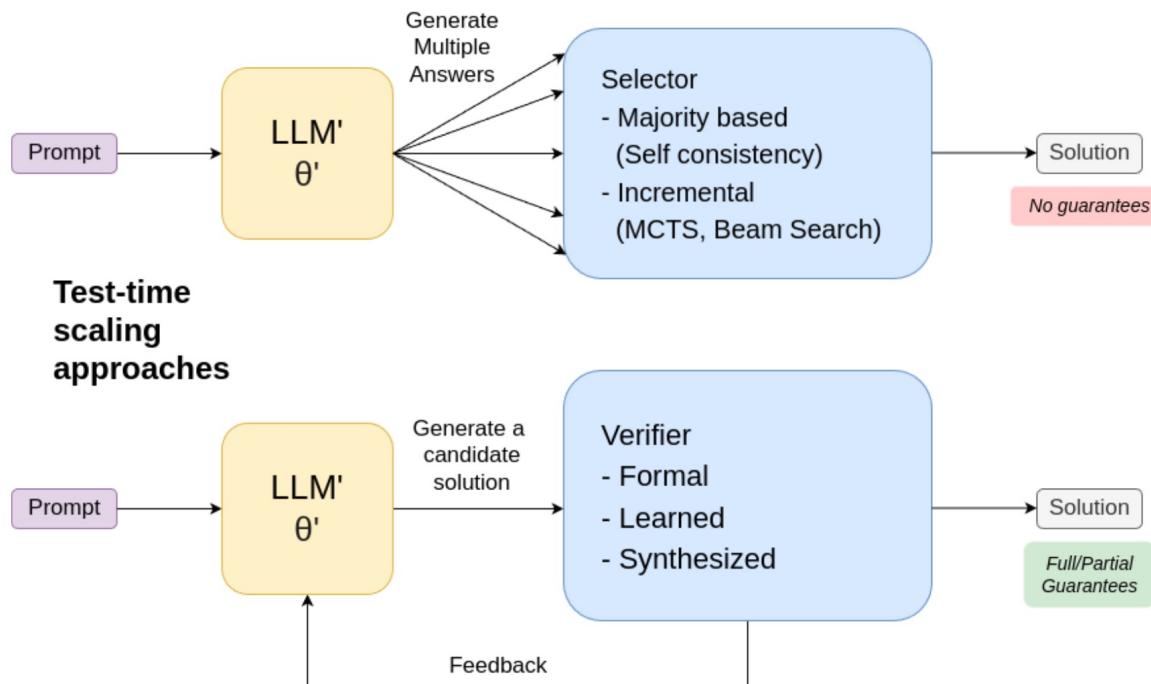
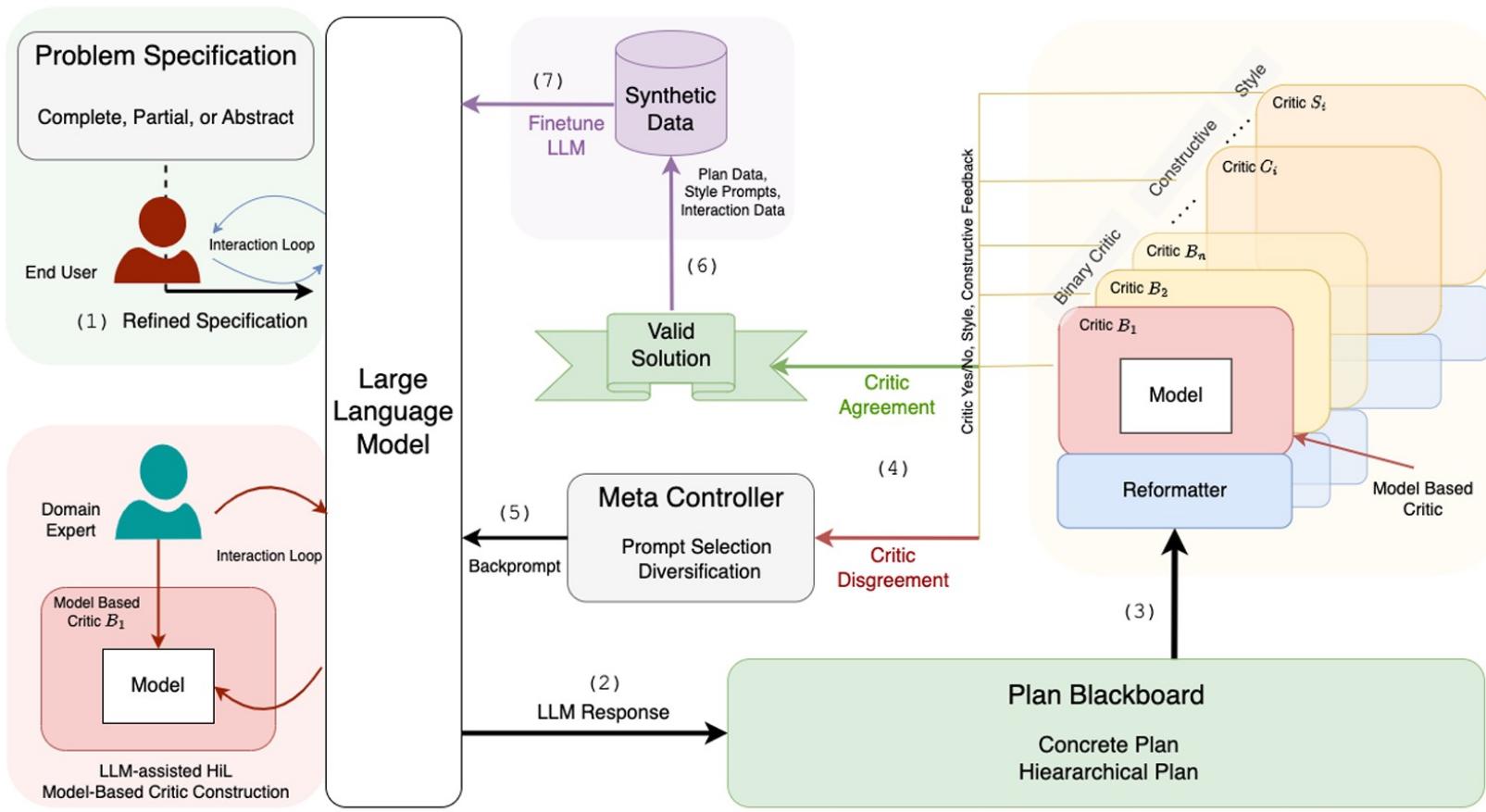
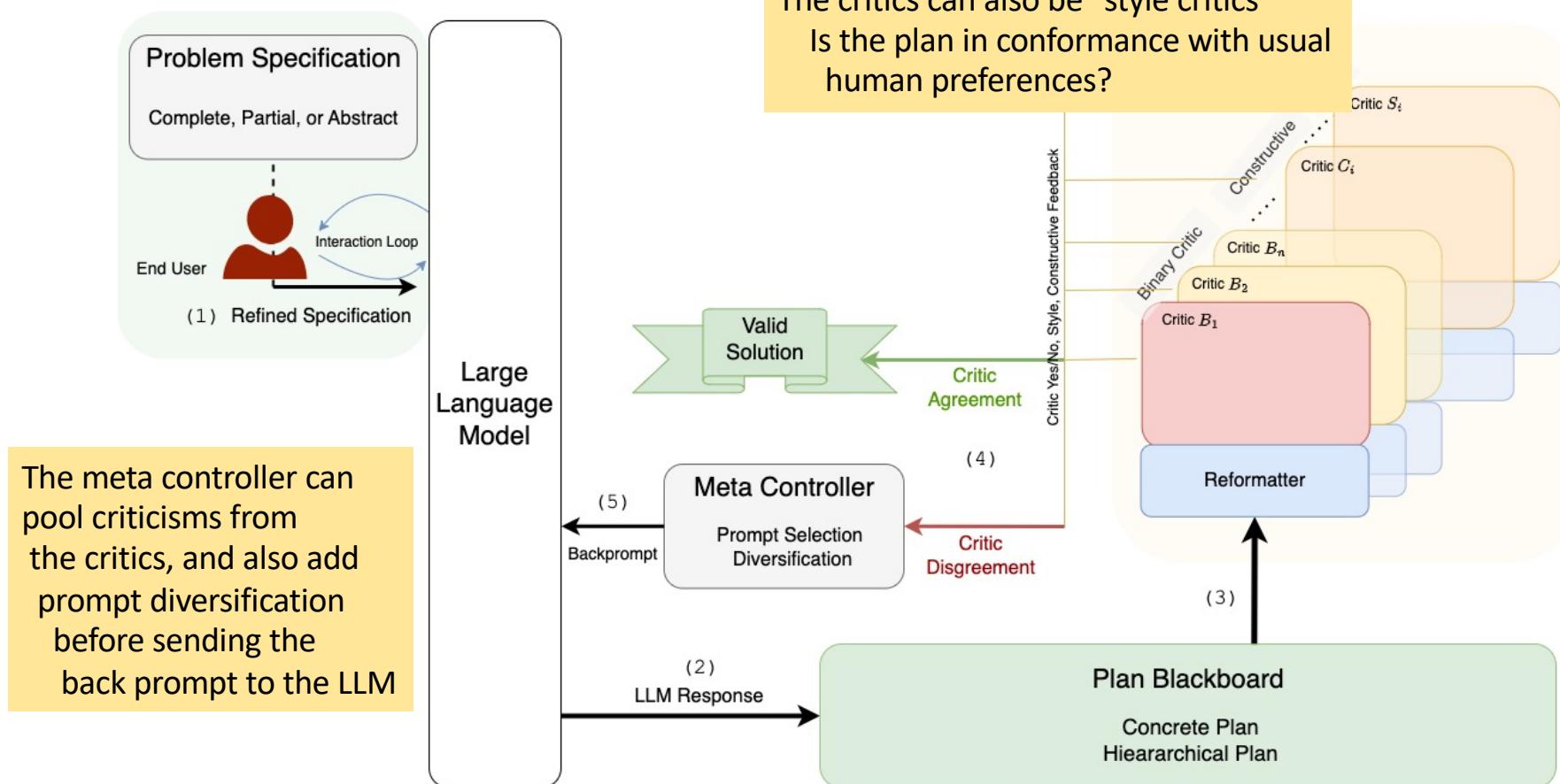


Figure 1. Test-time scaling approaches for teasing out reasoning

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



LLM-Modulo with a Bank of critics



LLM-Modulo Improves Performance (with guarantees) on Multiple Benchmarks

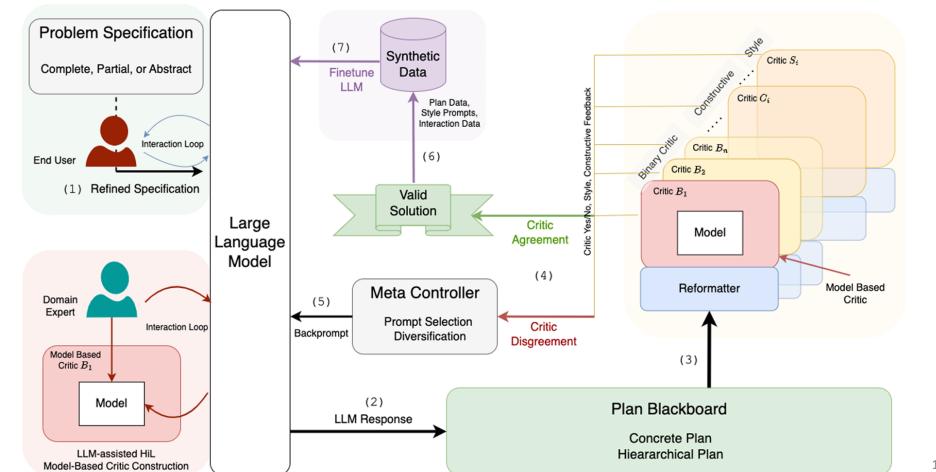
Models	OSU		TP		MP		CS	
	Direct	LM	Direct	LM	Direct	LM	Direct	LM
Gpt 4o-mini	2.78%	15.00%	6.18%	12.06%	32.80%	51.90%	36.90%	61.60%
Gpt 4o	8.33%	23.89%	3.43%	40.00%	49.10%	59.80%	56.10%	83.30%
Claude 3.5 Sonnet	4.44%	25.00%	39.43%	47.00%	57.10%	69.5%	72.90%	88.80%

Table 1: Performance of various LLMs on Travel Planner (OSU), Meeting Planning (MP), Trip Planning (TP), and Calendar Scheduling (CS). The **Direct** column represents direct prompting, while the **LM** column shows the results of 10 iterations of LLM-Modulo.

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

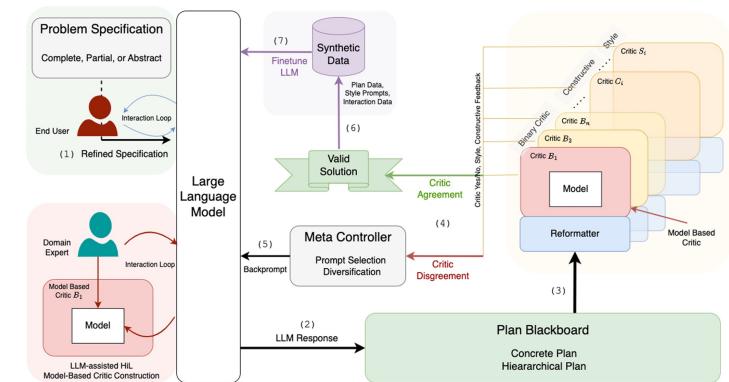
- Although we just saw a way of combining LLMs with external solvers, our *recommendation* is that you interface LLMs with Verifiers/Critics
 - This is why the LLM-Modulo architecture talks about a bank of critics
 - You can have constructive critics and style critics
- With solvers, you are stuck with 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



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 correct 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**



 Subbarao Kambhampati (కోచువారి) @rao22 ...

👉 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!

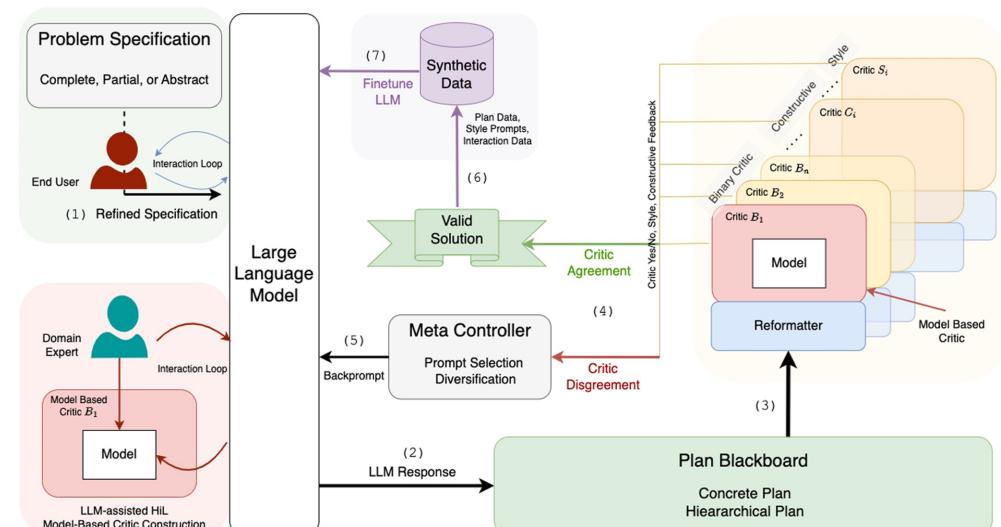
7:46 AM · Mar 26, 2024 · 8,635 Views

 View post engagements

 7  18  75  24 

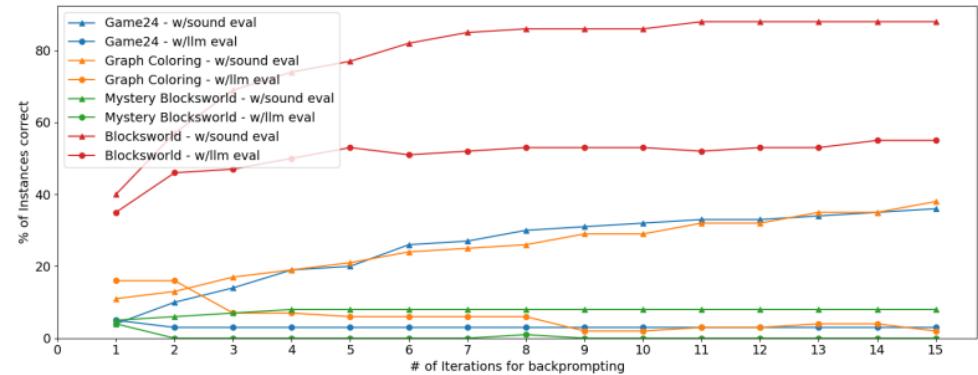
Types of Critics in LLM-Modulo

- 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' 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	
Game of 24	5%	3%	36%	38%	N/A	28%	42%	6%
Graph Coloring	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 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. COLM 2024*

Pick up a bag of chips



GPT-4V: The robot gripped the bag of chips too tightly, causing it to crumble 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



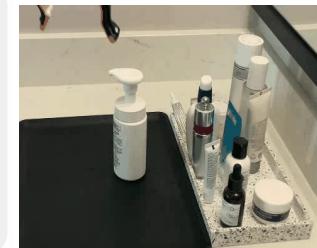
GPT-4V: The robot's arm holding the glass of orange juice passes over the laptop, which poses a risk of spilling liquid on the electronic device.

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-

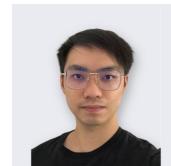
Language Models as Behavior Critics for Catching

Undesirable Agent Behaviors

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

Subbarao Kambhampati. COLM 2024

[COLM 2024]



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**.

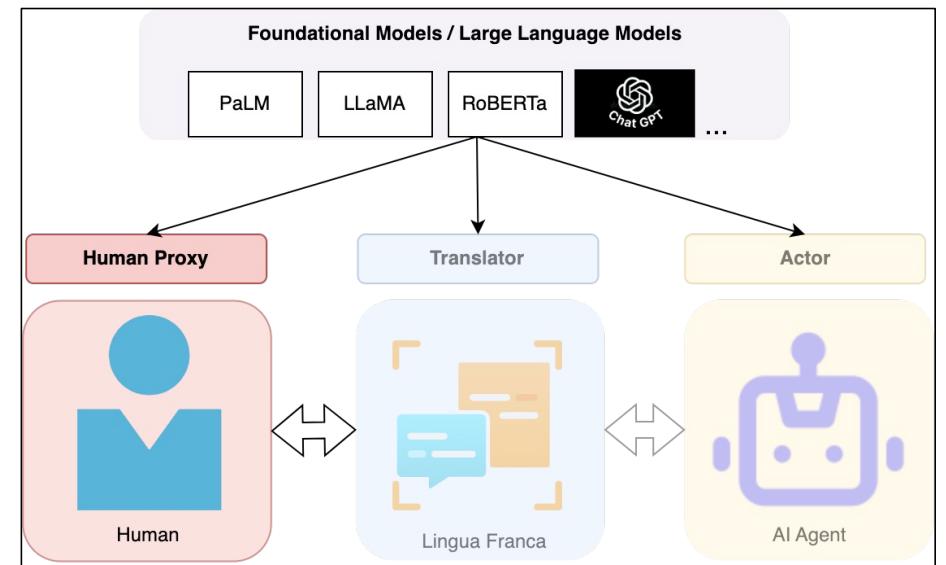


Figure: Different roles of an LLM in Human-AI interaction.

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.

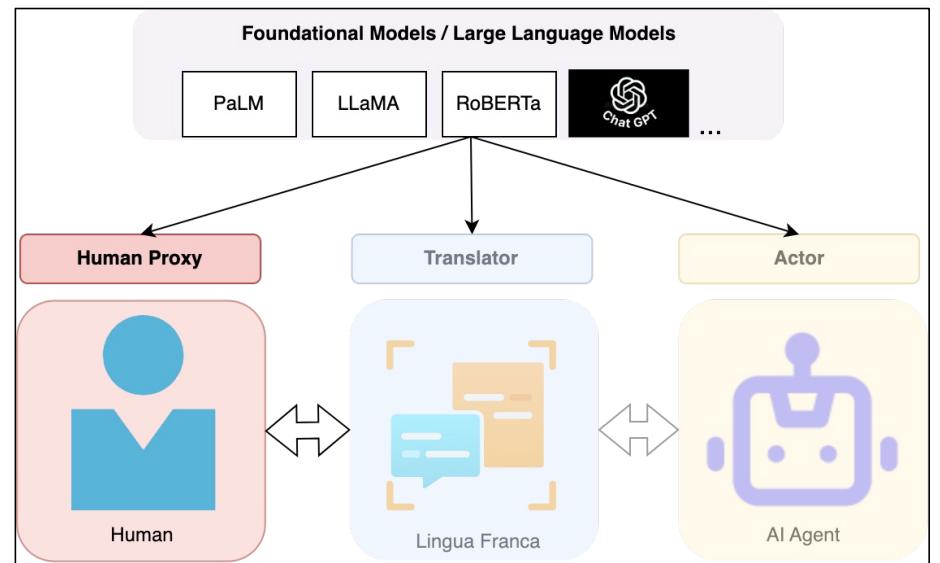


Figure: Different roles of an LLM in Human-AI interaction.

LLM-Modulo is the sane way to use LLMs in Reasoning problems when correctness guarantees are needed..

- The AlphaGeometry, AlphaProof, FunSearch & AlphaEvolve all use LLMs as idea generators—with external sound solvers/verifiers
 - They spend enormous resources finetuning the LLMs to improve the density of good guesses
- The Program Synthesis community uses LLM generated code guesses in conjunction with unit testing over Python interpreters
 - Unit testing is a partial verification



Spectrum of Ideas for Test-Time Scaling

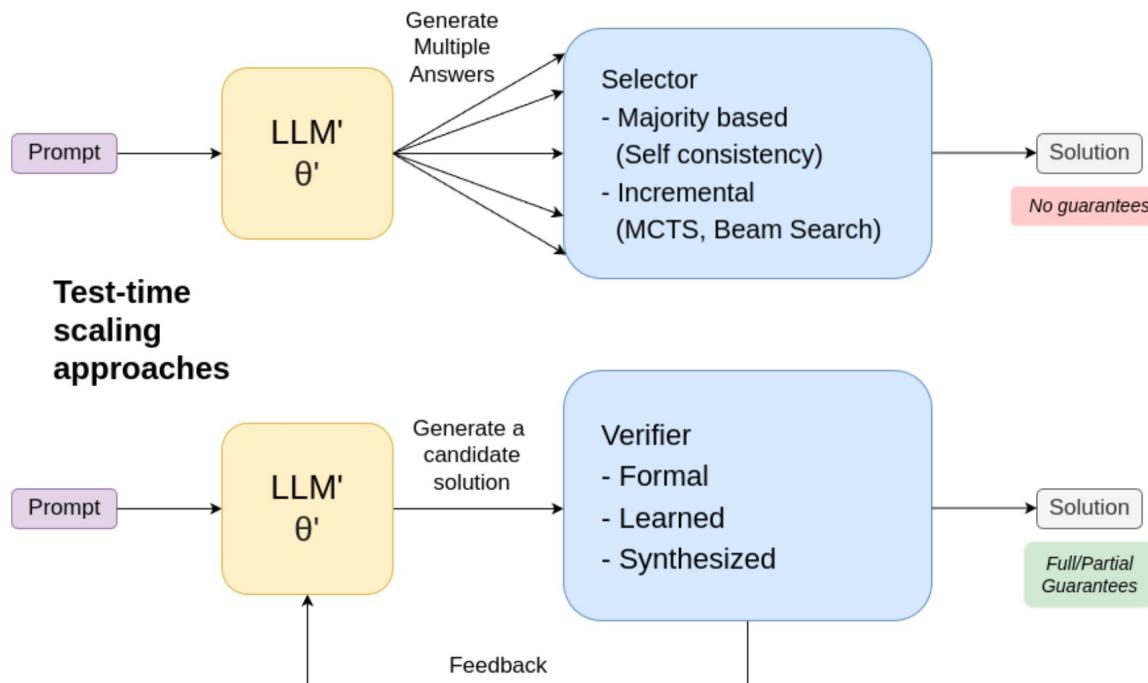
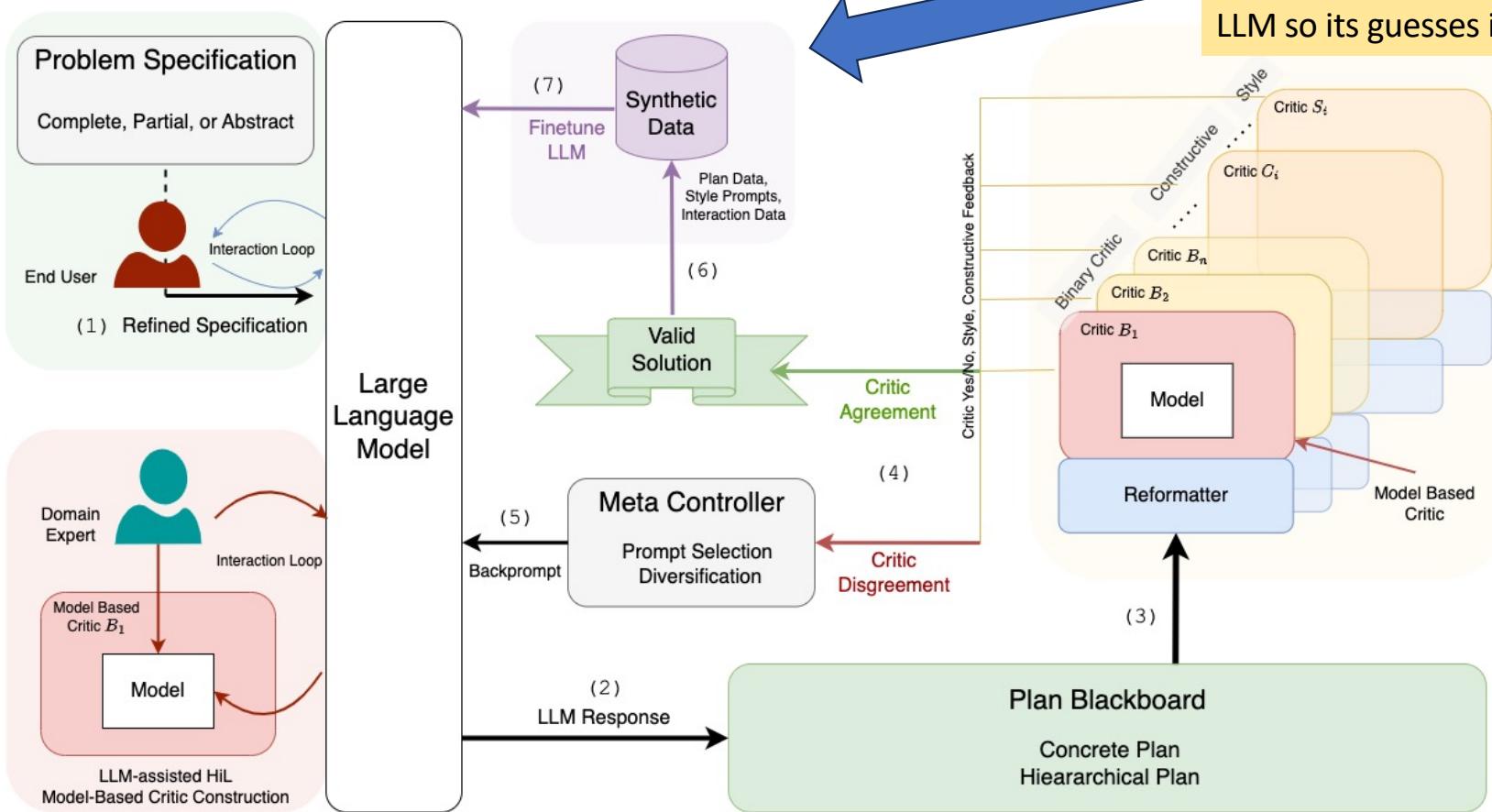


Figure 1. Test-time scaling approaches for teasing out reasoning

“Intelligence is
integrating Test
part of Generate-
Test into Generate
part”
--Marvin Minsky



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..

Post-Training Approaches

- Post-training approaches rely on synthetic data
- Often generated with the help of classical verifiers
 - Compute converted to Data (Supplementing "fact web" with "reasoning web")
- LLMs are fine-tuned not only on the solutions, but also the *Intermediate Tokens* leading to those solutions
 - Intermediate tokens may come from the humans, traditional solvers, or base LLMs generating intermediate tokens that imitate derivational traces; lots of mythos here!
- RL post-training can be seen as incremental compilation of verifier signal

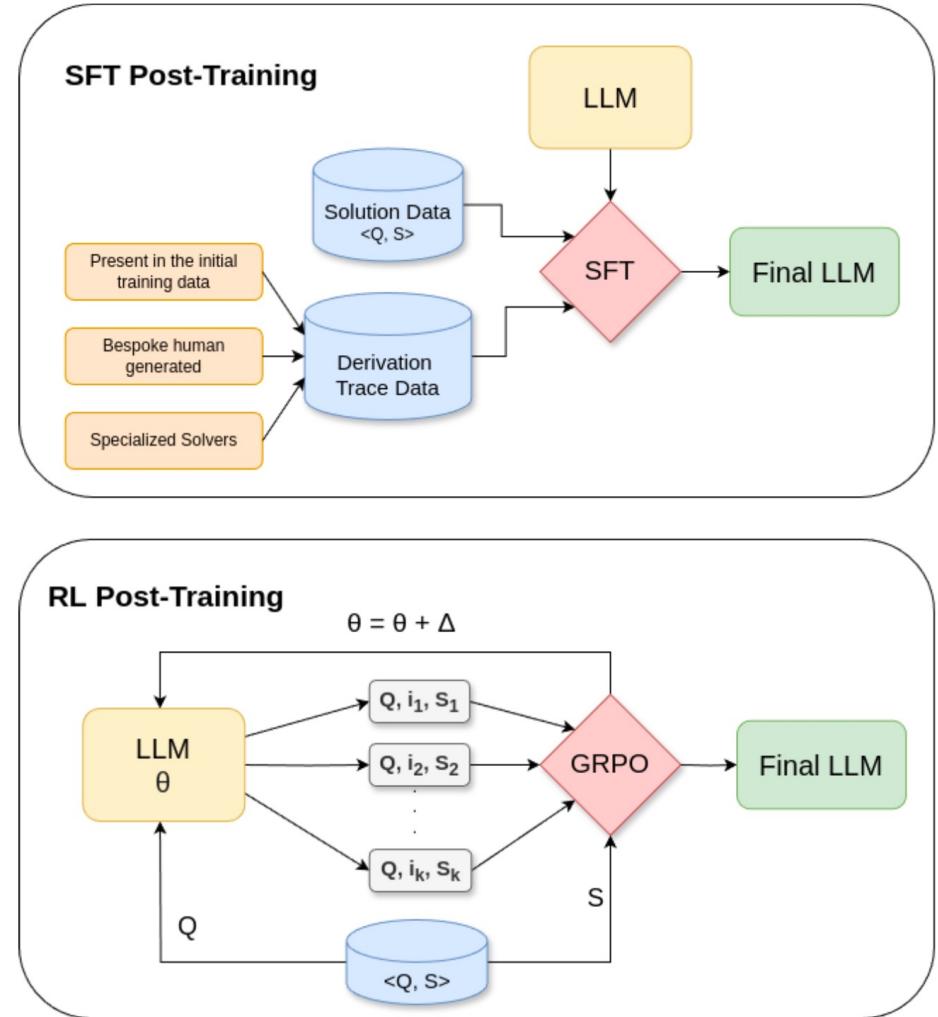
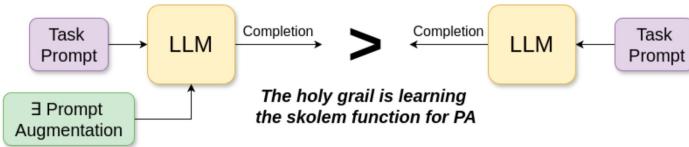


Figure 2. Post-training Approaches for teasing out reasoning



Can LLMs Reason & Plan?

పద్ధ భాషా భవిషయాలు పన్మగాలు పన్నగలవా?

Subbarao Kambhampati
School of Computing & AI
ASU Arizona State University
Twitter: @rao2z

Slides <https://bit.ly/3AJPwNl>

Keynote @  ACL 2024 Bangalore, India



Noam Brown 
@polynoamial

For example, last month at the 2024 Association for Computational Linguistics conference, the keynote by [@rao2z](#) was titled “Can LLMs Reason & Plan?” In it, he showed a problem that tripped up all LLMs. But [@OpenAI o1-preview](#) can get it right, and o1 gets it right almost always

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?

Thought for 19 seconds ▾

To achieve a stack where block A is on top of block B and block B is on top of block C—without moving block C—you would need to move blocks A and B while leaving block C stationary.

However, since block C is initially on top of block A, and you cannot move block C, you cannot remove block A from under block C without moving C in the process. This makes it impossible to rearrange the blocks into the desired order under the given constraints.

Therefore, it is not possible to create the desired stack without moving block C.

10:19 AM · Sep 12, 2024 · 110.6K Views

LLMs STILL CAN'T PLAN; CAN LRM^s?

A PRELIMINARY EVALUATION OF OPENAI'S o1 ON PLANBENCH

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

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

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

ABSTRACT

The ability to plan a course of action that achieves a desired state of affairs has long been considered a core competence of intelligent agents and has been an integral part of AI research since its inception. With the advent of large language models (LLMs), there has been considerable interest in the question of whether or not they possess such planning abilities. PlanBench [1], an extensible benchmark we developed in 2022, soon after the release of GPT3, has remained an important tool for evaluating the planning abilities of LLMs. Despite the slew of new private and open source LLMs since GPT3, progress on this benchmark has been surprisingly slow. OpenAI claims that their recent o1 (Strawberry) model has been specifically constructed and trained to escape the normal limitations of autoregressive LLMs—making it a new kind of model: a Large Reasoning Model (LRM). Using this development as a catalyst, this paper takes a comprehensive look at how well current LLMs and new LRMs do on PlanBench. As we shall see, while o1's performance is a quantum improvement on the benchmark, outpacing the competition, it is still far from saturating it. This improvement also brings to the fore questions about accuracy, efficiency, and guarantees which must be considered before deploying such systems.

1 Introduction

The recent release of OpenAI's o1 (Strawberry) [2] brings with it the opportunity to both freshly evaluate progress on PlanBench and to consider directions for extending the benchmark. In particular, unlike the LLMs which came before it, which can roughly be viewed as approximate retrievers, o1 seems to have been trained to be an approximate reasoner.¹ Following OpenAI², we draw a distinction between previous Large Language Models and o1, a Large Reasoning Model (or LRM), as its new (unknown) architecture, operation, and capabilities all seem to be fundamentally different from those of vanilla LLMs, both at pretraining phase and at inference time. To properly evaluate this new kind of model and understand its abilities and limitations will require new tools and evaluation methods, especially if details of the overall model structure are kept secret and internal traces remain inaccessible to outside researchers.³

Since PlanBench [1] first debuted on arXiv in 2022, we have continually retested new models on one particular subset: a static dataset of 600 three to five block problems. Even though LLMs have gotten ever larger and required substantially more investment per model, their performance on even the simplest planning problems has never come close to saturating this test set, and the improvements we have seen have not been robust or generalizable [6]. Our benchmark has thus served as a useful marker of LLM progress (or lack thereof) on planning and reasoning tasks, though it is important to note that this analysis—especially when confined to a static test set—can only serve as an upper bound on performance. When we first designed PlanBench, it was intended to be more than just this set, but rather to be

*Equal Contribution

¹We speculate that the complete system learns to improve its ability to make appropriate Chain-of-Thought (CoT) moves useful for reasoning in a pretraining RL step with synthetic data, and does inference time prompt-specific rollouts [3]. In other words, it may

Published in Transactions on Machine Learning Research (April/2025)

A Systematic Evaluation of the Planning and Scheduling Abilities of the Reasoning Model o1

Karthik Valmeekam*
School of Computing and AI
Arizona State University

kvalmeek@asu.edu

Kaya Stechly*
School of Computing and AI
Arizona State University

khstechly@asu.edu

Atharva Gundawar
School of Computing and AI
Arizona State University

agundawa@asu.edu

Subbarao Kambhampati
School of Computing and AI
Arizona State University

rao@asu.edu

Reviewed on OpenReview: <https://openreview.net/forum?id=FkKBxpOFhR>

Abstract

OpenAI claims that their recent o1 (Strawberry) model has been specifically constructed and trained to escape the normal limitations of autoregressive Large Language Models (LLMs)—making it a new kind of model: a Large Reasoning Model (LRM)—and be generally capable of tackling procedural reasoning tasks. We present the first comprehensive evaluation of these models on the fundamental tasks of planning and scheduling. Previous research attempted to use LLMs' expressive generation capabilities to solve these problems, but met with only limited success. We fill in the gaps in this literature by testing a larger suite of state-of-the-art LLMs on a set of large benchmarks, and then use this as a baseline to evaluate o1-preview and o1-mini. We see that while they can offer significant *accuracy* improvements over LLMs, this single metric is misleading and incomplete, as LRM queries demand large and unpredictable costs and take significant amounts of time to complete. We provide a case study demonstrating that, at those same price points, other methods of inference time scaling can do just as well. We also show that, contrary to OpenAI's injunctions, o1's performance can be improved further by embedding it in compound systems that separately, but complementarily, scale inference time further. Finally, while the paper is focused on o1, we provide similar evaluations of a more recent (and open-weight) LRM – DeepSeek R1.

1 Introduction

Large Language Models (LLMs) have captured the imagination of the AI research community by demonstrating linguistic behaviors that no one expected text completion systems to possess. Their seeming versatility initially led to claims that they are equally capable of deliberative behaviors such as planning and scheduling, but they have been called into question by a steady stream of more systematic investigations (Valmeekam et al., 2023; Silver et al., 2022; Xie et al.; Zheng et al., 2024; Mirzadeh et al., 2024). In some sense, this

*Equal Contribution

O1 (LRM) is better than Autoregressive LLMs but at quite considerable API access cost

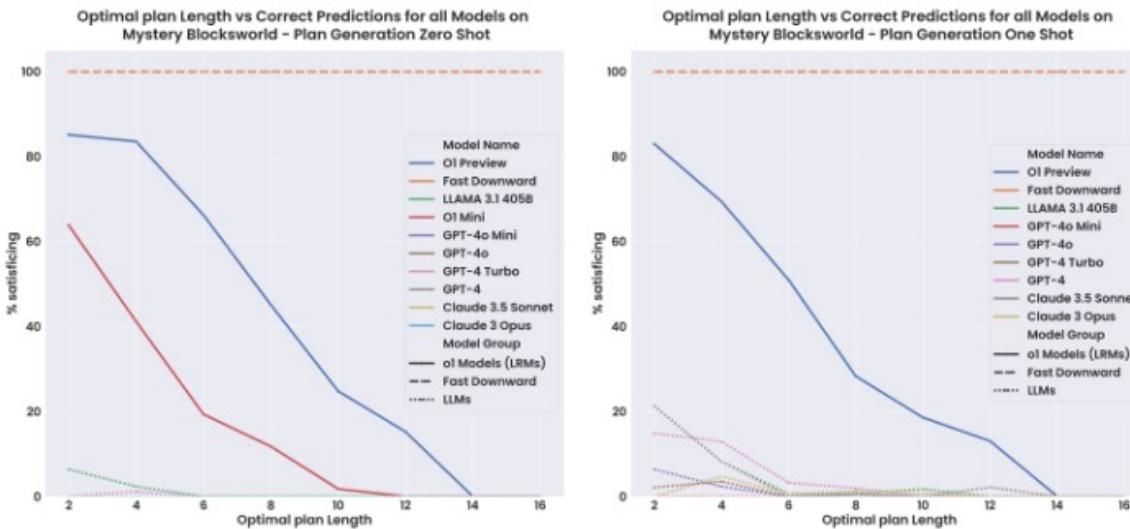


Figure 1: These examples are on Mystery Blocksworld. Fa planner [10] solves all given instances near-instantly with guarantees on even the smallest instances. The two LRMs we tested, o1, are effective, but this performance is still not robust, and degrades

Subbarao Kambhampati (కంప్యూటిస్చుల్ రావు) @rao2z · Sep 13 ...

One fun part of the pricing structure of API access currently is that you are charged for reasoning tokens—that you don't get to see or control—as output tokens. This makes even a lowly last-letter concatenation instance cost up to 50c a pop..!

You can't have AGI
Because you can't afford Reasoning Tokens!

Subbarao Kambhampati (కంప్యూటిస్చుల్ రావు) @rao2z · Sep 13

Currently considering starting a GoFundMe to pay for the planning evaluations on o1. 😊 (gave nearly 2K overnight to @sama 😊) x.com/rao2z/status/1...

Costs per 100 instances (in USD)									
Large Language Models								Large Reasoning Models	
Claude 3.5 (Sonnet)	Claude 3 (Opus)	GPT-4o	GPT-4o -mini	GPT-4	GPT-4 Turbo	Gemini 1.5 Pro	Gemini 1 Pro	o1-preview	o1-mini
\$0.44	\$1.70	\$0.65	\$0.02	\$1.80	\$1.20	\$0.33	\$0.03	\$42.12	\$3.69

Table 4: Cost per 100 instances (in USD). LRMs are significantly more expensive than LLMs.

DeepSeek R1 is Competitive with o1

Results on PlanBench as of 01/20/2025

Domain		Large Language Models					Large Reasoning Models		
		Claude-3.5 Sonnet	GPT-4o	GPT-4	LLaMA-3.1 405B	Gemini 1.5 Pro	o1-preview	o1-mini	Deepseek R1
Blocksworld	Zero shot	329/600 (54.8%)	213/600 (35.5%)	210/600 (34.6%)	376/600 (62.6%)	143/600 (23.8%)	587/600 (97.8%)	340/600 (56.6%)	483/500 (96.6%)
Mystery Blocksworld	Zero shot	0/600 (0%)	0/600 (0%)	1/600 (0.16%)	5/600 (0.8%)	-	317/600 (52.8%)	115/600 (19.1%)	199/500 (39.8%)
Avg. API Cost per 100 instances	-	\$0.44	\$0.65	\$1.80	-	\$0.33	\$42.12	\$3.69	\$2.07

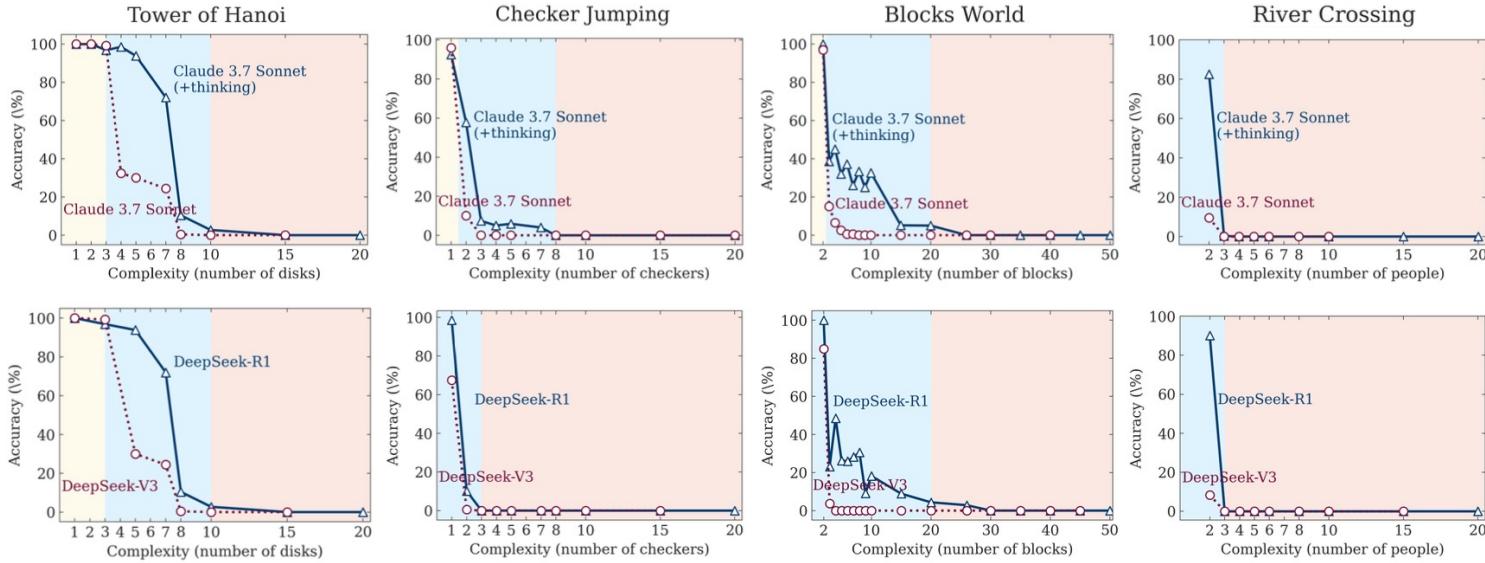


Figure 4: Accuracy of thinking models (Claude 3.7 Sonnet with thinking, DeepSeek-R1) versus their non-thinking counterparts (Claude 3.7 Sonnet, DeepSeek-V3) across all puzzle environments and varying levels of problem complexity.

The Illusion of Thinking:
Understanding the Strengths and Limitations of Reasoning Models
via the Lens of Problem Complexity

Parsin Shojaaei*^a Iman Mizrahi*^a Keivan Alizadeh^b
Maxwell Horton^b Samy Bengio^b Mohed Farajtabar^b
Apple

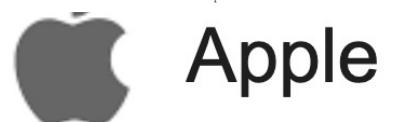
Abstract

Recent generation of frontier language models have introduced Large Reasoning Models (LRMs) that generate detailed thinking processes before providing answers. While these models demonstrate improved performance on reasoning tasks, they also exhibit several shortcomings, such as lack of common sense reasoning, difficulty in handling complex inputs, and limitations remain insufficiently understood. Current evaluation primarily focuses on stabilized mechanical and coding benchmarks, emphasizing final answer accuracy. However, this work reveals that LRMs’ reasoning behavior is often inconsistent with their final answers due to internal reasoning artifacts. To address this, we introduce a novel experimental setup that decouples the reasoning process from the final answer by analyzing the internal reasoning traces. This setup enables the analysis of not only final answers but also the internal reasoning traces, offering insights into how LRMs “think”. Through this analysis, we find that LRMs’ reasoning behavior is often inconsistent and lacks a complete accuracy collapse beyond certain complexity. Moreover, they exhibit a counter-intuitive behavior where they provide wrong answers for some reasoning tasks despite having a sufficient budget of tokens. By comparing LRMs with their standard LLM counterparts, we find that LRMs’ reasoning behavior is often inconsistent with their final answers, especially for (1) low-complexity tasks where LRMs’ reasoning becomes redundant, (2) medium-complexity tasks where additional thinking in LRMs demonstrates advantage, and (3) high-complexity tasks where LRMs’ reasoning becomes redundant again. We also find that LRMs’ reasoning artifacts are often caused by their “thinking” mechanisms such as long Claude’s thought (CLT) self-repetition, which can lead to wrong answers. Our findings highlight the need for better understanding and analyzing the models’ computational behavior, shedding light on their strengths, limitations, and ultimately raising crucial questions about their true reasoning capabilities.

1 Introduction

Large Language Models (LLMs) have recently evolved to include specialised variants explicitly designed for reasoning tasks—Large Reasoning Models (LRMs) such as OpenAI’s cl0 [1], 2025 [2], Claude with Thinking [3], and Gemini Thinking [4]. These models are new artifacts, characterized by their “thinking” mechanisms such as long Claude’s thought (CLT) self-repetition, and have demonstrated promising results across various reasoning benchmarks. Their

*Equal contribution.
^aWork done during an internship at Apple.
^b{shojaaei, mizrahi, alizadeh, maxhorton, bengio, farajtabar}@apple.com



Meta Is Building a Superintelligence Lab. What Is That?

Mark Zuckerberg has reorganized his company's ambitions around a hypothetical future that is suddenly the talk of Silicon Valley.

▶ Listen to this article - 7:05 min [Learn more](#)

Share full article

The New York Times

Technologists are divided on whether this technique will lead to A.G.I. [A recent paper from Apple](#) — which followed a [similar paper](#) from researchers at Arizona State University — shows that the latest A.I. systems could become less accurate as they work through the multiple steps of problem solving. It was an indication that the metronomic improvement that the tech industry has grown accustomed to may be slowing.

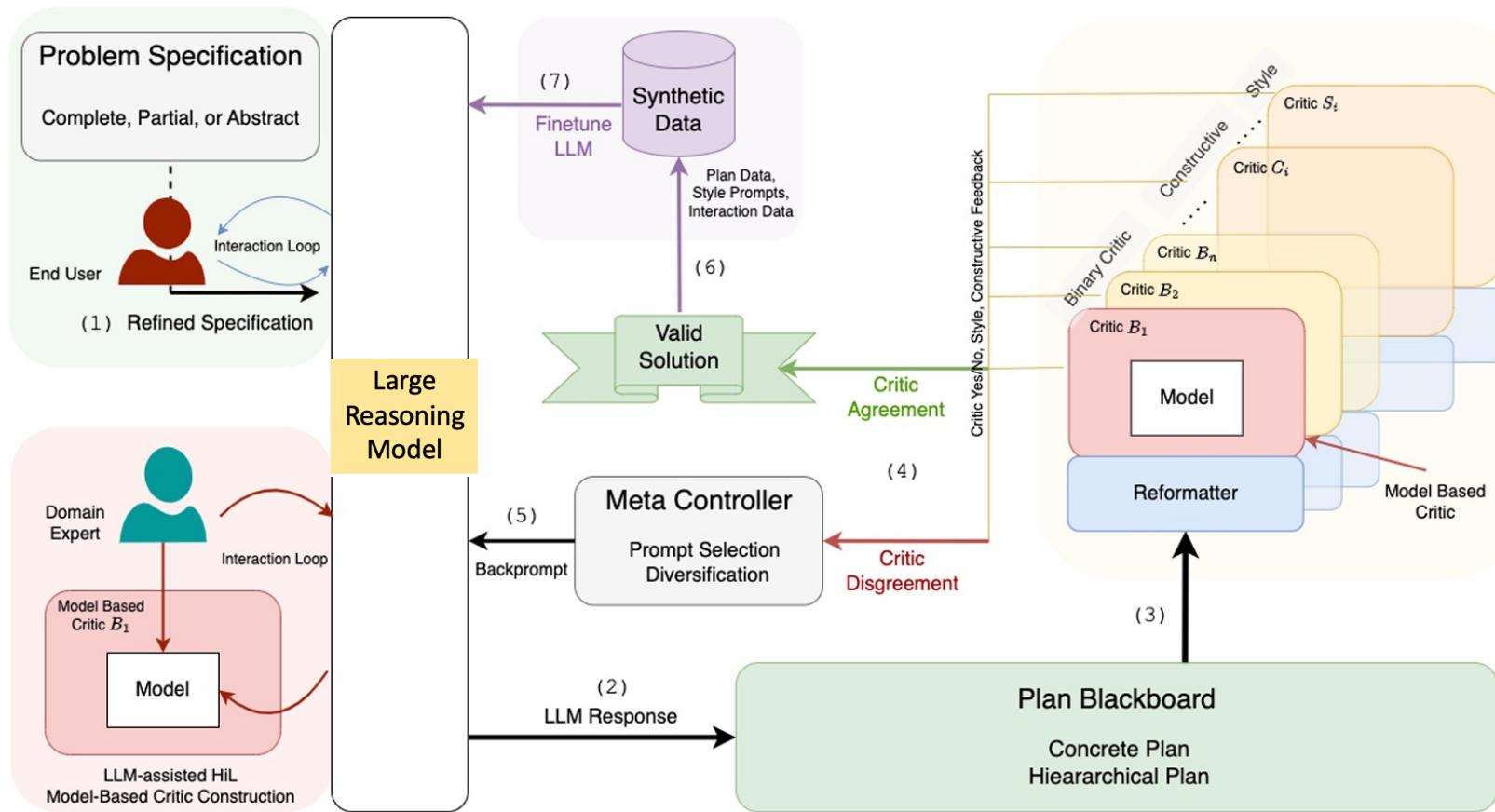
“These are great tools,” said Subbarao Kambhampati, a professor and A.I. researcher at Arizona State, who was an author of the report. “But I don’t believe we are anywhere near A.G.I. or superintelligence. With these tools, you always need a human in the loop.”

That is why Meta is using the superintelligence label.

“Where these terminologies come from is anyone’s guess,” Dr. Kambhampati said. “These terms have become a commercial thing, a branding thing, an advertising thing, rather than a technical thing.”

Cade Metz is a Times reporter who writes about artificial intelligence, driverless cars, robotics, virtual reality and other emerging areas of technology.

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



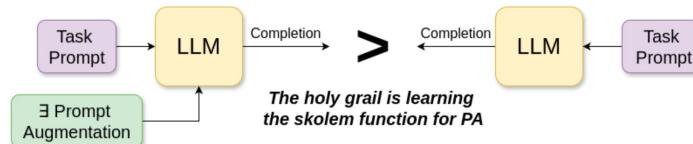
LRM-Modulo to Improve o1 with guarantees

Total Instances	Domain	o1-preview					o1-mini				
		Direct		LRM Modulo			Direct		LRM Modulo		
		% correct	Cost (in \$)	% correct	# of iters	Cost (in \$)	% correct	Cost (in \$)	% correct	# of iters	Cost (in \$)
110	Blocksworld (hard)	23.65%	52.12	98.2%	7	139.36	0.90%	16.30	10%	4	40.80
55	Sokoban	12.70%	42.89	43.6%	7	203.38	10.90%	5.28	12.70%	4	17.49
50	Graph Coloring (hard)	64%	26.65	94%	10	56.16	50%	3.33	84%	15	14.18
180	Travel Planning	10%	68.21	65%	10	431.75	1.67%	12.11	41.11%	10	97.17
200	Trip Planning (10 cities)	4%	131.26	15.50%	4	535.49	1%	20.01	3.50%	4	77.71

Table 5: Performance of LRM-modulo with o1-preview and o1-mini as the underlying LRMs on our hardest test sets: Blocksworld (20+ length plans), Sokoban, Graph Coloring (20 vertex), Travel Planning, and Trip Planning (10 cities). Due to cost constraints, we run each problem set for up to ten iterations, stopping early if the improvement per iteration levels off.

Some thoughts on LRM

- LRM can be understood as incrementally compiling verifier signal into the underlying generator (LLM)
 - If so, this suffers from similar amortization issues as normal finetuning
- There seem to be considerable misperceptions about the role of intermediate tokens
 - All we know is that intermediate tokens *help the LLM autocomplete*; their utility to humans (interpretability etc.) are questionable



- The role of RL vs iterative SFT needs to be understood better..

In vs. Out of Distribution & 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..

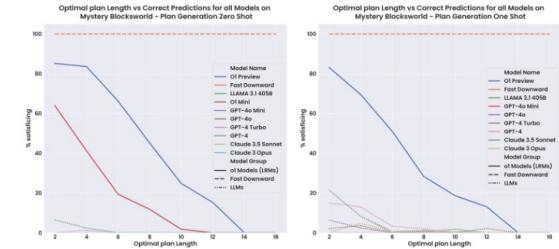
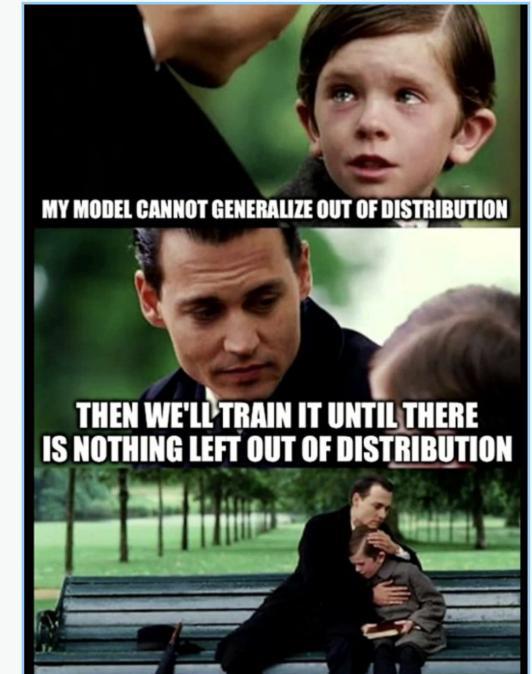
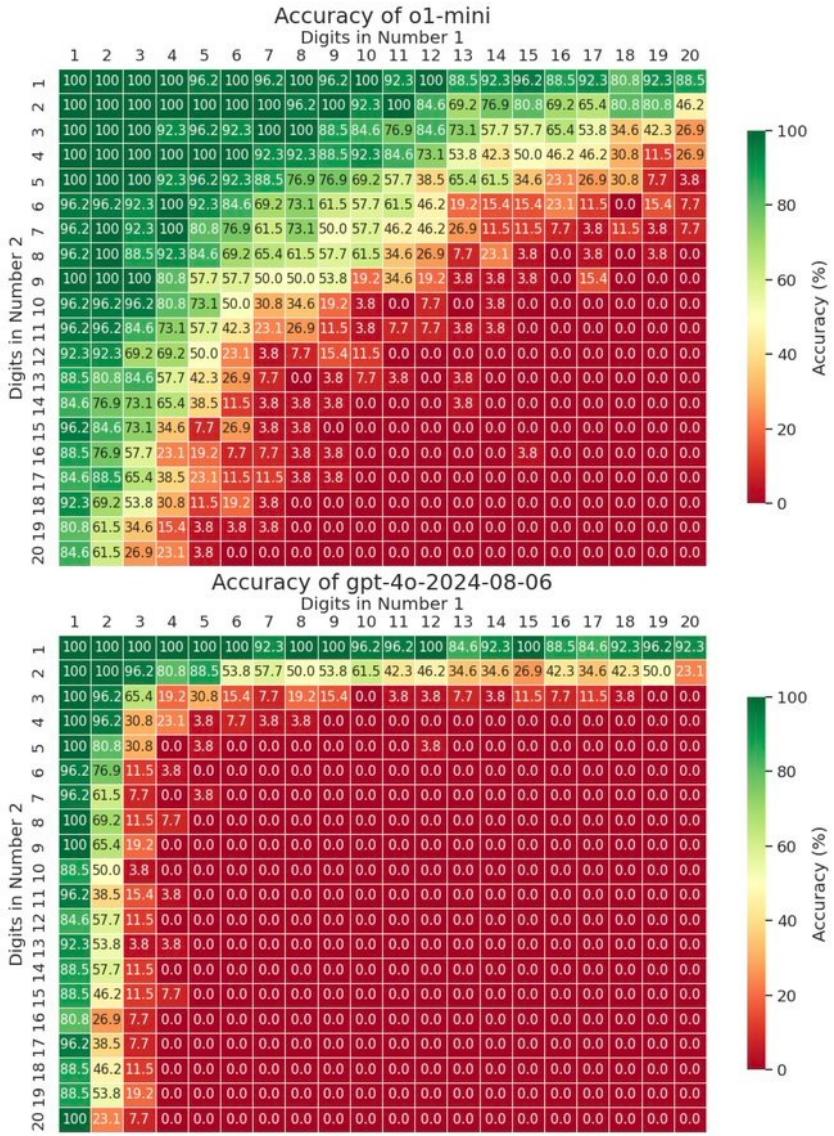


Figure 1: These examples are on Mystery BlocksWorld. Fast Downward, a domain-independent planner [10] solves all given instances near-instantly with guaranteed perfect accuracy. LLMs struggle on even the smallest instances. The two LLMs we tested, o1-preview and o1-mini, are surprisingly effective, but this performance is still not robust, and degrades quickly with length.

 **Nathan Benaich**  @nathanbenaich · May 30
frontier ai today



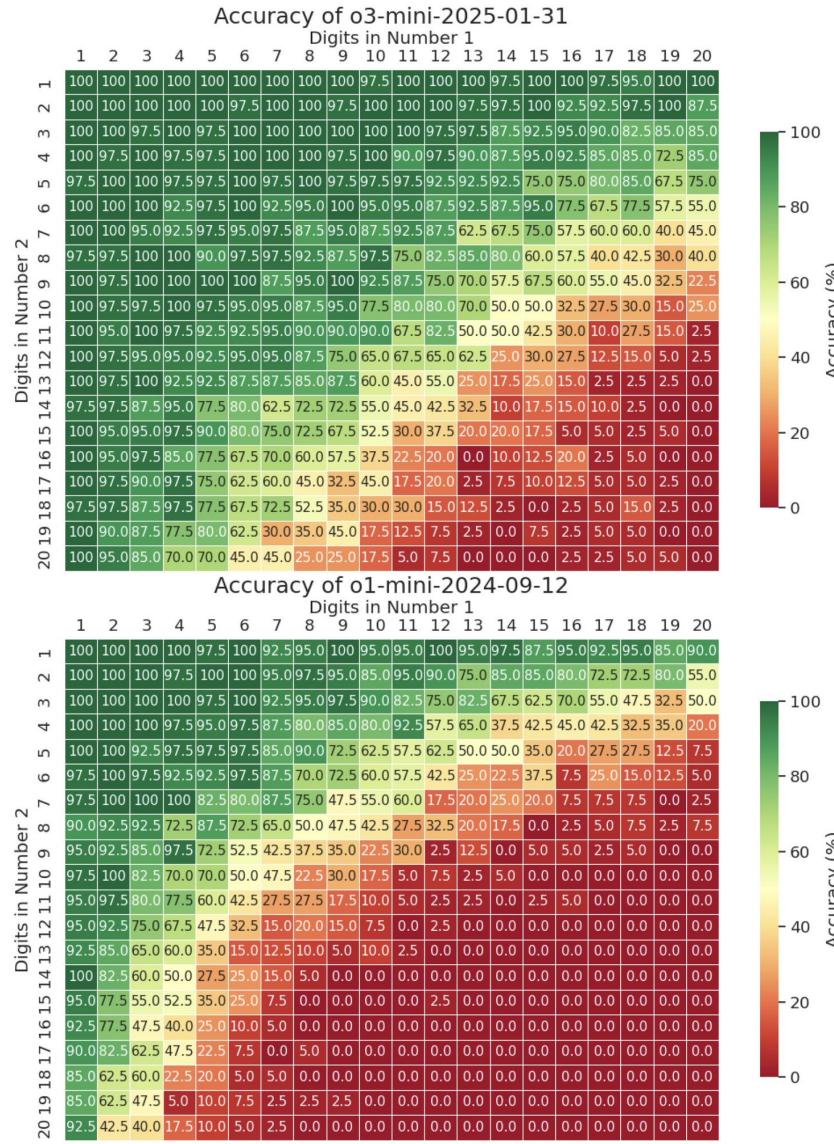


...

g ✓ @yuntiandeng · Feb 12



For those curious about how **o3**-mini performs on multi-digit **multiplication**, here's the result. It does much better than **o1** but still struggles past 13×13 . (Same evaluation setup as before, but with 40 test examples per cell.)





Dimitris Papailiopoulos ✅
@DimitrisPapai



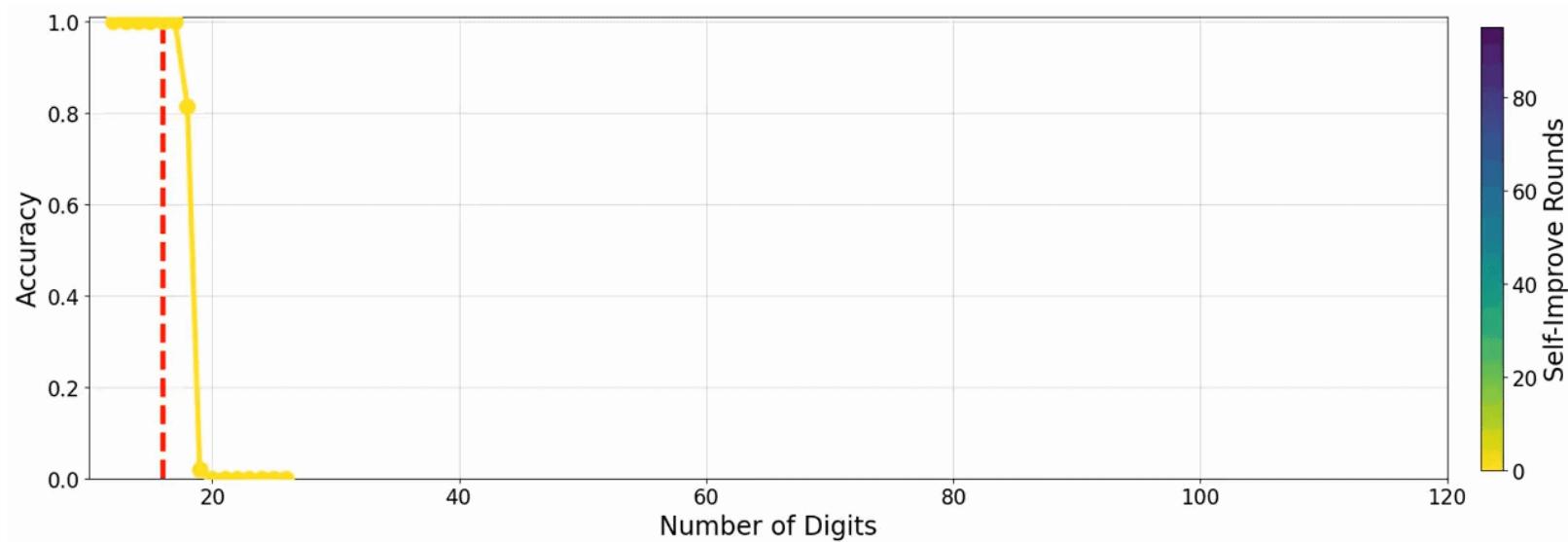
...

o3 can't multiply beyond a few digits...

But I think multiplication, addition, maze solving and easy-to-hard generalization is actually solvable on standard transformers...

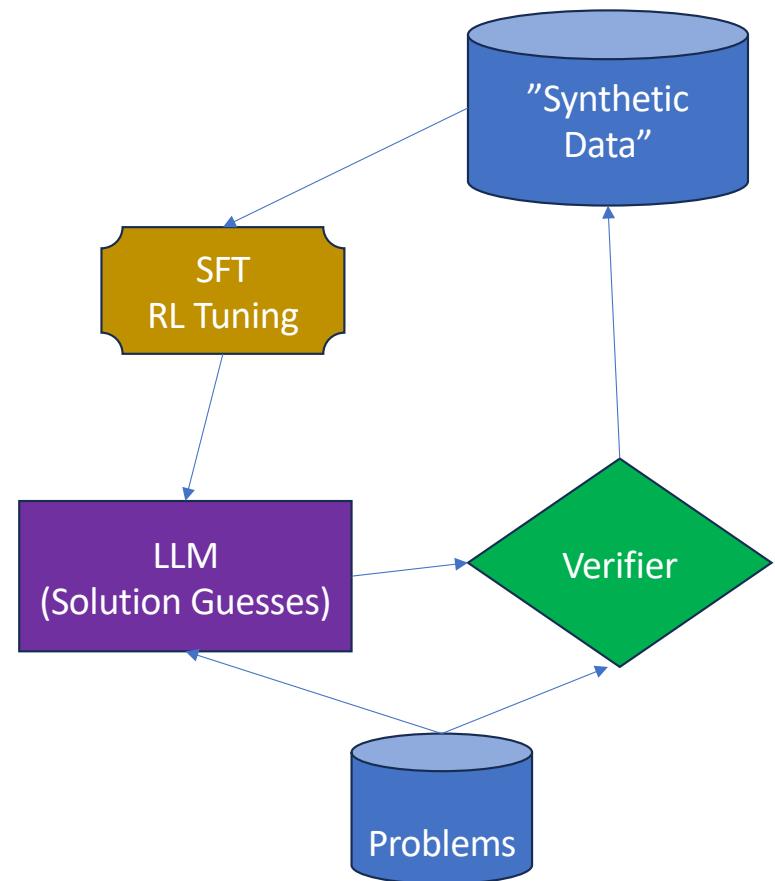
with recursive self-improvement.

Below is the acc of a tiny model teaching itself how to add.



Train- Test- Distill-time Inference Compilation

- Training with synthetic data can be viewed as compiling the signal from the **verifier** into the base LLM generator
 - (With SFT and/or RL)
 - To go from pass@k to pass@1
- Test-Time: LLM-Modulo
- Training-Time: Post-Training
- Distill-Time: Bootstrap the training of next gen LLM with the current one





Anthropomorphization of intermediate tokens!

- Aha moments
- ``Thoughts''
- Reasoning Behaviors
 - The models are ``thinking'' through the problem
 - Backtracking
 - Self-Verification
- CoT connections

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a+x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a - \sqrt{a+x}} = x$, let's start by squaring both ...

$$(\sqrt{a - \sqrt{a+x}})^2 = x^2 \implies a - \sqrt{a+x} = x^2.$$

Rearrange to isolate the inner square root term:

$$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$$

...

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be ...

We started with the equation:

$$\sqrt{a - \sqrt{a+x}} = x$$

First, let's square both sides:

$$a - \sqrt{a+x} = x^2 \implies \sqrt{a+x} = a - x^2$$

Next, I could square both sides again, treating the equation: ...

Table 3 | An interesting “aha moment” of an intermediate version of DeepSeek-R1-Zero. The model learns to rethink using an anthropomorphic tone. This is also an aha moment for us, allowing us to witness the power and beauty of reinforcement learning.

Deepseek-R1 [1]

A contrast in behaviors explored by the two models

Verifications
“Let me check my answer ...”

Backtracking
“Let's try a different approach, what if we ...”

Subgoal Setting
“Let's try to get to a multiple of 10”

Backward Chaining
“Working backwards, 24 is 8 times 3”



Qwen and Llama behaviors [2]

[1] Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., ... & He, Y. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948.

[2] Gandhi, K., Chakravarthy, A., Singh, A., Lile, N., & Goodman, N. D. (2025). Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. arXiv preprint arXiv:2503.01307.[2] Lehnert, L.,

Are Post-Trained Models really generating reasoning traces?

- There is an expectation that the intermediate tokens output by LLMs are “reasoning traces”
- There is little reason to believe this
 - LLMs trained on derivational traces *imitate* those derivational traces.
- But hard to formally evaluate the long meandering traces

R1 Mumblings - Mystery Blocksworld

Okay, let me try to figure out how to get object c to crave object a. Hmm, the initial conditions are a bit complex. Let's list them out first.

So initially, I have:

- Object a craves object b.
- Object d craves object c.
- Harmony is true.
- Planet objects b and c.
- Province objects a and d.

My goal is to have object c crave object a. The possible actions are Attack, Feast, Succumb, and Overcome. Each has specific prerequisites and effects. Let me break down each action's requirements and consequences to see how they can be chained together.

First, let's look at the Overcome action. The Overcome action requires Province (other object) and Pain (object). When performed, it sets Harmony, Province (object), and Object Craves other object. Also, it removes Province (other object) and Pain (object). Wait, the description says "Overcome object from another object". So maybe when you use Overcome, you specify the object and the other object? Like Overcome X from Y. Then, the prerequisites would be Province Y and Pain X. After Overcome, you get Harmony, Province X, and X craves Y. And you lose Province Y and Pain X.

Similarly, Feast requires Object Craves other object, Province object, Harmony. When you Feast X from Y, you need X craves Y, Province X, and Harmony. Then after Feast, you get Pain X and Province Y. You lose the Craves relation, Province X, and Harmony.

So the goal is to have c crave a. That would require using the Overcome action on c from a, perhaps? Because Overcome can create a Craves relation. Let's check: If I do Overcome c from a, the prerequisites would be Province a (since the other object is a) and Pain c. Wait, but initially, there's no Pain anywhere. So I need to create Pain c somehow.

How do I create Pain? The Attack action creates Pain. To Attack an object, you need Province, Planet, and Harmony. When you Attack, you set Pain and remove Province, Planet, and Harmony.

R1 Mumblings - Mystery Blocksworld

Wait, the Attack action's prerequisites are Province (object), Planet (object), and Harmony. So to Attack an object, say X, you need Province X, Planet X, and Harmony. After attacking, you get Pain X, and lose Province X, Planet X, and Harmony.

But looking at the initial conditions: Province a and d, Planet b and c. Harmony is initially true.

So maybe I can Attack either a or d? Wait, no. Because to Attack an object X, you need Province X and Planet X. But the Planet objects are b and c. So Province a and d are present, but their Planet counterparts are b and c. So for example, to Attack a, you would need Planet a, but Planet a isn't there. Wait, the initial planet objects are b and c. So Attack requires that the object you're attacking has both Province and Planet. So maybe the objects that are Provinces are a and d, and their Planets are b and c? Not sure. Wait, the initial conditions state: "planet object b, planet object c, province object a and province object d." So planet b and planet c exist, and province a and province d exist. So maybe each object can be a province or a planet. So for example, object a is a province, object b is a planet, etc.

So to Attack an object X, you need Province X, Planet X, and Harmony. But in the initial setup, the only provinces are a and d, and the only planets are b and c. So does that mean that to Attack, you need an object that is both a province and a planet? That might not be possible initially. Wait, but maybe the Attack action is on some object that's a province and another that's a planet? Wait, the problem statement might be a bit ambiguous. Let me recheck the Attack action's requirements.

The Attack action's prerequisites: "Province object, Planet object, Harmony." So perhaps Attack requires a Province object (any) and a Planet object (any), and Harmony. Wait, maybe the Attack action is performed on an object, and it requires that there exists a Province object and a Planet object (regardless of which ones) along with Harmony. Then, when you perform Attack, you consume the Province and Planet objects, and Harmony, and create Pain on the attacked object.

Wait, but the problem says "Once Attack action is performed the following facts will be false: Province object, Planet object, Harmony." So when you Attack, you lose the Province and Planet objects. But which ones? The ones used in the prerequisites. Wait, but the Attack action is performed on an object, so maybe the object being attacked is the one that is both Province and Planet? That's confusing.

R1 Mumblings - Mystery Blocksworld

Alternatively, maybe the Attack action is performed on a target object, and you need a Province (some object) and a Planet (some object) to perform it. Then after attacking, those specific Province and Planet objects are removed. But the problem statement isn't entirely clear. Let me re-examine the exact wording.

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

Wait, maybe it's that when you Attack, you need to have at least one Province object and at least one Planet object, along with Harmony. Then, when you perform Attack, you remove some Province and Planet object. But which ones? The problem statement says "Once Attack action is performed the following facts will be false: Province object, Planet object, Harmony." So perhaps when you Attack, you have to choose which Province and Planet objects to consume. But the action's description isn't entirely clear. Maybe the Attack action is performed on a target object, and the Province and Planet objects are the ones associated with that target. Hmm.

Alternatively, maybe the Attack action doesn't take an object parameter, but just requires that there are any Province and Planet objects. Then, when you Attack, it removes those Province and Planet objects, regardless of which ones they are. But that seems odd. Let's see the initial state: there are Province a and d, Planet b and c. So if I perform Attack, I would lose, say, Province a and Planet b? Or both Provinces and both Planets? The problem statement isn't clear. This is a bit of an ambiguity in the problem description. Let me try to parse the original text again.

Original Attack action:

- To perform Attack action, the following facts need to be true: Province object, Planet object, Harmony.
- Once Attack is performed: Pain object is true.
- Facts false: Province object, Planet object, Harmony.

So perhaps when you perform Attack on a particular object, you need that object to be a Province and a Planet. But in the initial state, the Provinces are a and d, and Planets are b and c. So none of the objects are both Province and Planet. That would mean Attack is impossible initially. But that can't be right because the initial state includes Harmony, which is required for Attack. Wait, but the user says the initial state has Harmony, Planet b and c, Province a and d, and the cravings.

R1 Mumblings - Mystery Blocksworld

So perhaps the Attack action is performed on an object, which must be a Province and a Planet. But since none are both, Attack isn't possible. That seems contradictory. Alternatively, maybe the Attack action requires any Province and any Planet, not necessarily the same object. So for example, if you have Province a and Planet b, you can Attack, consuming both, and causing Pain on some object. But the problem says "Attack object"—so the action is performed on a specific object. Then, maybe when you Attack object X, you need Province X and Planet X. But in the initial setup, none of the objects are both Province and Planet. So Attack is impossible. That can't be right. But the user's initial state includes Harmony, so maybe there's a way.

Alternatively, maybe the Attack action is performed on a Province object, and a Planet object is required. So, for example, Attack a (a Province) requires that there is a Planet object (like b or c), and Harmony. Then, after Attack, the Province a and the Planet object used (say, b) are removed, along with Harmony, and Pain a is added.

But the problem statement isn't explicit. Let's assume that Attack requires a Province object and a Planet object (they can be different objects), and when you perform Attack, you choose which Province and Planet to consume, and the Pain is applied to some object. But the problem says "Attack object"—so maybe the object being attacked is the one that becomes in Pain. Let me try to think of an example.

Suppose I Attack object X. The prerequisites are that X is a Province, there exists a Planet object (any), and Harmony. Then, after Attack X, Pain X is true, and the Province X and the Planet object (which one?) and Harmony are removed. But the problem statement says "Once Attack action is performed the following facts will be false: Province object, Planet object, Harmony." So it's ambiguous whether it's the specific Province and Planet objects involved, or all of them. For example, if there are multiple Provinces and Planets, which ones get removed?

This ambiguity might be a problem. Let's see if we can proceed with an assumption. Let's assume that when you Attack an object X, you need X to be a Province, and there exists at least one Planet object. Then, performing Attack X would remove Province X, one Planet object (maybe any, or a specific one?), and Harmony, and add Pain X.

R1 Mumblings - Mystery Blocksworld

But in the initial state, we have Province a and d, Planets b and c. So maybe we can Attack a, consuming Planet b or c. But which one? The problem statement doesn't specify. Alternatively, maybe Attack requires both a Province and a Planet, and they are consumed. For example, Attack could be done on a Province (like a), and requires a Planet (like b). Then, after attacking, Province a and Planet b are removed, Harmony is gone, and Pain a is added.

Assuming this, let's proceed.

So initial state:

- Craves: a craves b, d craves c.
- Harmony is true.
- Planets: b, c.
- Provinces: a, d.

Goal: c craves a.

Let's think about how to create c craves a. The Overcome action can create a Craves relation. Overcome X from Y requires Province Y and Pain X. After Overcome, X craves Y, Harmony, Province X, and removes Province Y and Pain X.

So if we can get Pain c and Province a, then we can Overcome c from a, leading to c craves a. So how to get Pain c?

Pain can be added via Attack or Feast. Let's check Feast.

Feast X from Y requires X craves Y, Province X, Harmony. When you Feast, you remove the Craves relation, Province X, Harmony, and add Pain X and Province Y.

So for example, if we Feast a from b (since a craves b), that would require Province a and Harmony. Then, after Feast, Pain a is added, Province b is added (but b is already a Planet, not sure if that matters), and we lose a's craving, Province a, and Harmony.



Are Post-Trained Models really generating reasoning traces?

- There is an expectation that the intermediate tokens output by LLMs are “reasoning traces”
- There is little reason to believe this
 - LLMs trained on derivational traces *imitate* those derivational traces.
- But hard to formally evaluate the long meandering traces
- You can check this when the traces come from formal solvers

“Large Mumbling Models”?

BEYOND SEMANTICS: THE UNREASONABLE EFFECTIVENESS OF REASONLESS INTERMEDIATE TOKENS

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

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

Atharva Gundawar*
SCAI, Arizona State University
agundawa@asu.edu

Vardhan Palod*
SCAI, Arizona State University
vpalod@asu.edu

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

ABSTRACT

Recent impressive results from large reasoning models have been interpreted as a triumph of Chain of Thought (CoT), and especially of the process of training on CoTs sampled from base LLMs in order to help find new reasoning patterns. In this paper, we critically examine that interpretation by investigating how the semantics of intermediate tokens—often anthropomorphized as “thoughts” or reasoning traces and which are claimed to display behaviors like backtracking, self-verification, and meta-cognition—actually influence model performance. We train transformer models on formally verifiable reasoning traces and solutions, constraining both intermediate steps and final outputs to align with those of a formal solver. By constructing a formal interpreter of the semantics of our problems and intended algorithm, we systematically evaluate not only solution accuracy but also the correctness of intermediate traces, thus allowing us to evaluate whether the latter causally influences the former. Our experiments involve training transformer models on traces and solutions generated by A* search. We notice that, despite significant improvements on the solution-only baseline, models trained on entirely correct traces still produce invalid reasoning traces when arriving at correct solutions. To further show that trace accuracy is only loosely connected to solution accuracy, we then train models on noisy, corrupted traces which have no relation to the specific problem each is paired with, and find that not only does performance remain largely consistent with models trained on correct data, but in some cases can improve upon it and generalize more robustly on out-of-distribution tasks. These results challenge the assumption that intermediate tokens or “Chains of Thought” reflect or induce predictable reasoning behaviors and caution against anthropomorphizing such outputs or over-interpreting them (despite their mostly correct forms) as evidence of human-like or algorithmic behaviors in language models.

1 Introduction

Recent advances in general planning and problem solving have been spearheaded by so-called “Long Chain-of-Thought” models, most notably DeepSeek’s R1 [1]. These transformer-based large language models are further post-trained using iterative fine-tuning and reinforcement learning methods. Following the now-standard teacher-forced pre-training,

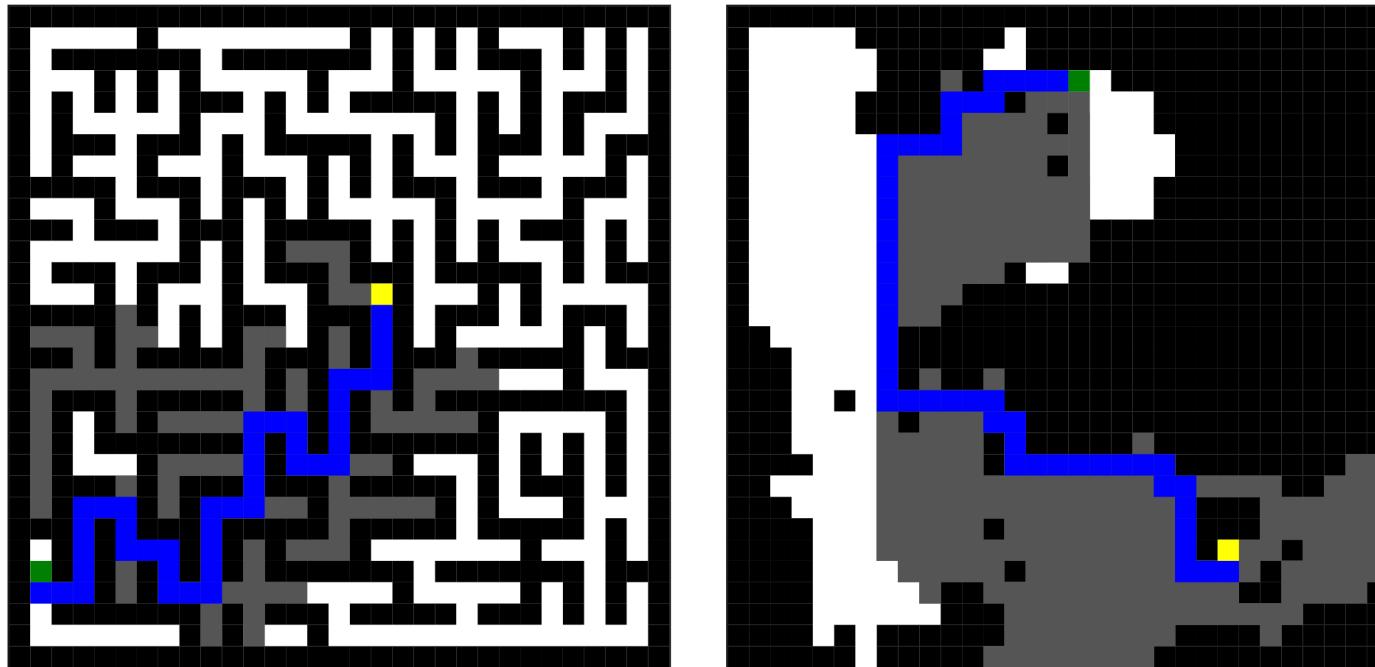


Figure 1: Examples of mazes. The left is generated by Wilson’s algorithm and is used for model training. The right is generated by the Drunkard’s Walk algorithm and used to evaluate models as an out of distribution task. The goal is represented by a green square and the start state by a yellow square. Black squares represent impassable walls. Blue squares represent steps along the optimal path (as found by A^*). Gray squares are squares that were explored by A^* but are not along the optimal path. White squares are unexplored traversable squares.

Acyclic Maze Generation

1. **Wilson's algorithm:** This is the algorithm that we use to generate mazes for training models. Wilson's algorithm generates uniform random mazes by performing loop-erased random walks from unvisited cells until they connect to the current maze [42]. Each walk removes any loops it creates, ensuring a valid tree structure. This process continues until all cells are included, producing a uniform sample from the space of all possible spanning trees of the 30×30 graph.
2. **Kruskal's algorithm:** Kruskal's algorithm, originally proposed for finding a minimum spanning forest of an undirected edge-weighted graph [43], generates mazes by treating each cell as a node and randomly removing walls between unconnected regions, using a union–find structure to avoid cycles. This results in a fully connected maze without loops, though the maze distribution is not perfectly uniform. The method produces mazes biased towards short local connections and dead ends.
3. **Randomized Depth-First Search algorithm:** The randomized depth-first search (DFS) or recursive back-tracker algorithm generates mazes by carving a path forward until reaching a dead-end [44]. When it hits a dead-end (no unvisited neighbors), it backtracks until it finds a new direction to explore, repeating until all cells are visited and connected into a complete maze. Depth-first search is biased towards generating mazes with low branching factors and many long corridors.

Cave Generation

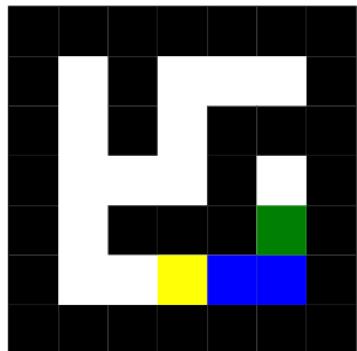
4. **Drunkard's Walk:** We implement a version of the “Drunkard’s Walk” algorithm, as described by [45], and originally used for procedurally generating dungeons for top-down two-dimensional video games. Starting from a grid of solid walls, a random walk is performed, carving out the current cell on every step. The walk

4

Preprint

continues until a predefined number or percentage of floor tiles has been dug out. This method preserves cycles, producing cave-like structures with open chambers and looping corridors. The output space includes grid states unreachable by perfect acyclic maze generators.

5. **SearchFormer style generation** We also implement the random generation algorithm used in the SearchFormer paper [18], though we use it for evaluation rather than training. Tasks are generated by exhaustive rejection sampling: first randomly select a number between 30% and 50%. Then select that percentage of cells to be wall cells. Randomly choose a start and goal location and execute A* to find an optimal plan. Reject unsolvable, too easy, or duplicate instances and resample. These instances also allow for loops and so are also out of distribution for our models.



reasoning

close 1 3 c0 c3

create 1 4 c1 c2

create 1 3 c0 c3 ← Invalid create (already closed)

create 3 3 c5 c4 ← Invalid neighbor

...

Figure 2: Trace validation procedure. Our A* validator runs through the model's output stream sequentially. Assuming no parsing errors, it will flag a trace as invalid if at some point it contains an invalid action. The left bottom corner is $(0, 0)$. The goal is represented by a green square and the start state by a yellow square.

- **Parsing Error:** a substring is malformed and does not parse into either a create or a close action with the correct arguments.
- **Invalid Neighbor:** the current create action is attempting to create an illegal child, either referencing a wall cell or a cell that is not adjacent to the last closed node.
- **Already Closed:** the current create action is attempting to close an already closed node.
- **Not in Open List:** the current close action is referencing a node that is not in the open list.
- **Not Lowest f -value:** the current close action is attempting to close a node when there is at least one other node in the open list with a lower f -value.
- **Goal Not Reached:** after the entire sequence was processed, the goal node was not in the closed list, and so the reconstruction step cannot proceed.

Searchformer mumblings

bos
create worker 2 4 box 2 3 box 4 2 c0
close worker 2 4 box 2 3 box 4 2 c0
create worker 1 4 box 2 3 box 4 2 c1
create worker 2 3 box 2 2 box 4 2 c1
create worker 2 5 box 2 3 box 4 2 c1
create worker 3 4 box 2 3 box 4 2 c1
close worker 2 5 box 2 3 box 4 2 c1
create worker 1 5 box 2 3 box 4 2 c2
create worker 3 5 box 2 3 box 4 2 c2
close worker 1 4 box 2 3 box 4 2 c2
create worker 1 3 box 2 3 box 4 2 c2
close worker 3 2 box 2 3 box 4 2 c2
create worker 4 2 box 2 3 box 4 2 c2
create worker 3 3 box 2 3 box 4 2 c2
close worker 2 2 box 2 3 box 4 2 c2
close worker 3 3 box 2 3 box 4 2 c2
create worker 2 1 box 4 2 c5
close worker 2 4 box 2 2 box 4 2 c2
create worker 2 5 box 2 2 box 4 2 c5
create worker 3 4 box 2 2 box 4 2 c3
create worker 1 4 box 2 2 box 4 2 c3
close worker 3 3 box 4 2 c4
create worker 3 4 box 4 2 c5
close worker 1 3 box 2 2 box 4 2 c2
create worker 1 2 box 2 2 box 4 2 c3
close worker 3 2 box 2 3 box 4 2 c3
create worker 4 2 box 2 3 box 5 2 c4
create worker 2 2 box 2 3 box 4 2 c4
close worker 1 3 box 2 2 box 4 2 c2
create worker 1 2 box 2 2 box 4 2 c3
close worker 3 2 box 2 3 box 4 2 c3
create worker 4 2 box 2 3 box 4 2 c6
close worker 2 5 box 2 3 box 4 2 c4
close worker 2 2 box 2 3 box 4 2 c4
close worker 4 2 box 2 3 box 5 2 c4
close worker 2 4 box 2 3 box 5 2 c4
close worker 1 4 box 2 3 box 4 2 c5
close worker 3 2 box 2 3 box 5 2 c5
create worker 1 3 box 2 3 box 4 2 c4
close worker 2 4 box 2 2 box 4 2 c3
close worker 1 1 box 2 3 box 4 2 c4
close worker 4 5 box 2 3 box 4 2 c3
close worker 2 4 box 4 2 c4
create worker 1 4 box 4 2 c5
create worker 2 5 box 4 2 c5
close worker 3 3 box 2 2 box 4 2 c2
create worker 3 2 box 2 3 box 4 2 c2
close worker 2 3 box 2 2 box 4 2 c1
create worker 1 3 box 2 2 box 4 2 c2
create worker 2 4 box 2 2 box 4 2 c2
create worker 3 3 box 2 2 box 4 2 c2
close worker 2 2 box 4 2 c4
create worker 1 2 box 4 2 c5
create worker 3 2 box 4 2 c5
close worker 2 3 box 3 3 box 4 2 c3
create worker 1 1 box 3 3 box 4 2 c4
close worker 1 2 box 3 3 box 4 2 c4
create worker 2 2 box 3 2 box 4 2 c4
close worker 2 3 box 3 3 box 4 2 c3
create worker 2 2 box 2 2 box 4 2 c4
create worker 1 3 box 2 2 box 4 2 c4
close worker 2 2 box 1 2 box 4 2 c4
create worker 1 3 box 3 3 box 4 2 c4
close worker 2 4 box 2 2 box 4 2 c4
create worker 1 4 box 2 2 box 4 2 c4
close worker 2 5 box 2 2 box 4 2 c4
create worker 1 5 box 2 2 box 4 2 c4
close worker 2 2 box 1 2 box 4 2 c4
create worker 2 1 box 1 2 box 4 2 c5
create worker 3 2 box 1 2 box 4 2 c5
create worker 2 3 box 1 2 box 4 2 c5
close worker 4 2 box 5 2 c6
create worker 3 2 box 5 2 c7
create worker 4 1 box 5 2 c7
close worker 4 2 box 2 2 box 5 2 c4
create worker 3 2 box 2 2 box 5 2 c5
create worker 4 1 box 2 2 box 5 2 c5
close worker 1 4 box 2 2 box 4 2 c3
create worker 1 1 box 4 2 c6
close worker 1 5 box 2 2 box 4 2 c4
close worker 2 1 box 2 3 box 4 2 c5
close worker 4 4 box 2 2 box 4 2 c4
close worker 1 5 box 4 2 c6
close worker 2 2 box 2 3 box 5 2 c6
create worker 2 3 box 2 4 box 5 2 c7
close worker 1 4 box 2 2 box 4 2 c3
close worker 3 5 box 2 2 box 4 2 c4
close worker 2 2 box 1 2 box 5 2 c6
create worker 2 3 box 1 2 box 5 2 c7
create worker 3 2 box 1 2 box 5 2 c7
create worker 2 1 box 1 2 box 5 2 c7
close worker 2 2 box 3 3 box 4 2 c4
create worker 3 3 box 1 2 box 4 2 c6
close worker 4 1 box 2 2 box 5 2 c7
create worker 2 1 box 1 2 box 5 2 c7
close worker 2 2 box 3 3 box 4 2 c4
create worker 2 1 box 3 3 box 4 2 c5
create worker 3 2 box 3 3 box 4 2 c5
create worker 1 2 box 3 3 box 4 2 c4
close worker 1 3 box 3 3 box 4 2 c4
create worker 1 4 box 3 3 box 4 2 c5
close worker 2 1 box 1 2 box 4 2 c5
create worker 1 1 box 1 2 box 4 2 c6
close worker 3 3 box 2 3 box 5 2 c6
create worker 2 3 box 5 2 c7
create worker 3 4 box 2 3 box 5 2 c7
close worker 2 3 box 5 2 c7
create worker 3 3 box 5 2 c8
create worker 5 1 box 2 3 box 5 4 c8
create worker 5 3 box 2 3 box 5 4 c8
close worker 3 2 box 1 2 box 4 2 c5
create worker 4 2 box 1 2 box 5 2 c6
close worker 4 2 box 2 2 box 4 2 c5
close worker 5 2 box 2 3 box 4 2 c5
create worker 4 2 box 2 3 box 3 2 c6
create worker 5 1 box 2 3 box 4 2 c6

Wilson		Kruskal		DFS		SF-Style		Drunkard						
Invalid Trace	518	25	Invalid Trace	546	22	Invalid Trace	732	29	Invalid Trace	999	1	Invalid Trace	960	35
Valid Trace	21	436	Valid Trace	13	419	Valid Trace	14	225	Valid Trace	0	0	Valid Trace	0	5
Incorrect Plan			Incorrect Plan			Incorrect Plan			Incorrect Plan			Incorrect Plan		

Figure 3: Plan versus trace validity for the model trained on correct traces, measured across domains. Wilson = generated by Wilson’s algorithm, Kruskal = mazes generated by Kruskal’s algorithm, DFS = mazes generated by Depth-First Search, SF-Style = instances generated in the SearchFormer Style, Drunkard = instances generated using the Drunkard’s algorithm.

Does correctness of the traces matter for training?

- We saw that SearchFormer outputs traces that don't correspond to valid A* reasoning
- This leads to the question: "Did the correctness of the traces matter in training phase?"
- Idea: *Swap* the reasoning traces between pairs of training examples
 - So reasoning trace for problem p1 is replaced with the reasoning trace for problem p2 and *vice versa*.
- If reasoning trace correctness mattered, you would expect test-time performance to take a big hit
- ... But it doesn't!

Table 1: Performance of Swap, A* Trace, and Solution-Only Models across maze distributions. "Plan Val." = Plan Validity, "Trace Val." = Trace Validity within Valid Plans

Test Set	Soln. Only		Regular A* traces		Swapped A* traces	
	Plan Val.	Plan Val.	Plan Val.	Trace Val.	Plan Val.	Trace Val.
Wilson	4.0%	50.1%	95.2%	95.2%	51.6%	0%
Kruskal	2.1%	49.7%	96.2%	96.2%	51.9%	0%
DFS	2.8%	30.8%	82.1%	82.1%	41.7%	0%
Drunkard	0.0%	2.5%	4.0%	4.0%	26.0%	0%
Searchformer-style	0.0%	0%	0%	0%	0.2%	0%

INTERPRETABLE TRACES, UNEXPECTED OUTCOMES: INVESTIGATING THE DISCONNECT IN TRACE-BASED KNOWLEDGE DISTILLATION

Siddhant Bhambri*, Upasana Biswas, Subbarao Kambhampati

School of Computing and AI

Arizona State University

{siddhantbhambri, ubiswas2, rao}@asu.edu

ABSTRACT

Question Answering (QA) poses a challenging and critical problem, particularly in today's age of interactive dialogue systems such as ChatGPT, Perplexity, Microsoft Copilot, etc. where users demand both accuracy and transparency in the model's outputs. Since smaller language models (SLMs) are computationally more efficient but often under-perform compared to larger models, Knowledge Distillation (KD) methods allow for finetuning these smaller models to improve their final performance. Lately, the intermediate tokens or the so called 'reasoning' traces produced by Chain-of-Thought (CoT) or by reasoning models such as DeepSeek R1 are used as a training signal for KD. However, these reasoning traces are often verbose and difficult to interpret or evaluate. In this work, we aim to address the challenge of evaluating the faithfulness of these reasoning traces and their correlation with the final performance. To this end, we employ a KD method leveraging rule-based problem decomposition. This approach allows us to break down complex queries into structured sub-problems, generating interpretable traces whose correctness can be readily evaluated, even at inference time. Specifically, we demonstrate this approach on Open Book QA, decomposing the problem into a Classification step and an Information Retrieval step, thereby simplifying trace evaluation. Our SFT experiments with correct and incorrect traces on the CoTemp QA, Microsoft Machine Reading Comprehension QA, and Facebook bAbI QA datasets reveal the striking finding that correct traces do not necessarily imply that the model outputs the correct final solution. Similarly, we find a low correlation between correct final solutions and intermediate trace correctness. These results challenge the implicit assumption behind utilizing reasoning traces for improving SLMs' final performance via KD.

Keywords Language Models, Intermediate Traces, Knowledge Distillation

1 Introduction

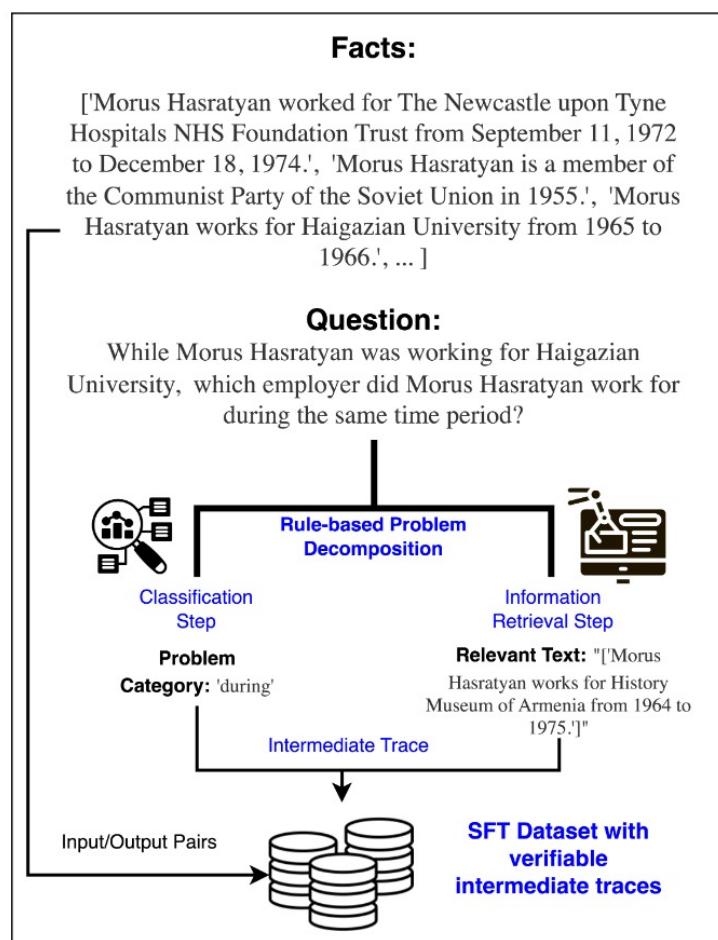


Figure 1: The construction of SFT dataset w/ verifiable intermediate traces using rule-based problem decomposition on an example from the CoTemp QA dataset.

Table 1: An example from the CoTemp QA dataset showing the outputs of Qwen3-1.7B and Llama-3.2-1B-Instruct models under different query setting. Correct final solutions are shown in green, and incorrect final solutions are shown in red. Correct intermediate traces are shown in blue, and incorrect intermediate traces are shown in red.

Model	Query Setting	Example
Qwen3-1.7b	Prompt	<p>Input Prompt: 'Answer the question based on the context: ['Morus Hasratyan worked for The Newcastle upon Tyne Hospitals NHS Foundation Trust from September 11, 1972 to December 18, 1974.', 'Morus Hasratyan is a member of the Communist Party of the Soviet Union in 1955.', 'Morus Hasratyan works for Haigazian University from 1965 to 1966.', " "Morus Hasratyan worked for Bishop's University from 1972 to 1975."', 'Morus Hasratyan worked for ISCTE – Lisbon University Institute from June, 1957 to December, 1960.', 'Morus Hasratyan works for History Museum of Armenia from 1964 to 1975.', 'Morus Hasratyan worked for Royal Air Force College Cranwell in February, 1959.', 'Morus Hasratyan worked for University of Detroit Mercy in September, 1963.', 'Morus Hasratyan worked for Tagesspiegel from May, 1957 to November, 1957.', 'Morus Hasratyan worked for North Carolina State University in May, 1962.'] Question: While Morus Hasratyan was working for Haigazian University, which employer did Morus Hasratyan work for during the same time period? Only return the answer.</p>
	SFT - Vanilla	<p>Gold Answer: "[History Museum of Armenia]" ; Problem Category: 'during' ; Relevant Text: "[Morus Hasratyan works for History Museum of Armenia from 1964 to 1975.]"</p>
	SFT w/ Correct Traces	<p>"Bishop's University" "[History Museum of Armenia]"</p> <p>"<think>The temporal relation between the event in question and the event in context is: equal. I need to use the following facts to answer the question: ['Morus Hasratyan works for History Museum of Armenia from 1964 to 1975.']/></p>
	SFT w/ Incorrect Traces	<p>"<think>The temporal relation between the event in question and the event in context is: equal. I need to use the following facts to answer the question: Morus Hasratyan worked for ISCTE - Lisbon University Institute from June, 1957 to December, 1960./></p>
	Prompt	<p>"Bishop's University"</p>
	SFT - Vanilla	<p>"[History Museum of Armenia]"</p>
	SFT w/ Correct Traces	<p>"<think>The temporal relation between the event in question and the event in context is: during. I need to use the following facts to answer the question: ['Morus Hasratyan works for History Museum of Armenia from 1964 to 1975.']/></p>
	SFT w/ Incorrect Traces	<p>"<think>The temporal relation between the event in question and the event in context is: overlap. I need to use the following facts to answer the question: Morus Hasratyan worked for Royal Air Force College Cranwell in February, 1959./></p>

Table 2: Cotemporal QA Results

Model	Query Setting	Final Solution Evaluations				Intermediate Trace Evaluations		
		Accuracy	F1	Precision	Recall	Classification Step Accuracy	IR Step Accuracy	Avg Trace Length (# tokens)
Qwen3-1.7b	Prompt	6.35	11.35	14.33	10.1	-	-	-
	SFT - Vanilla	60.33	74.88	82.15	71.3	-	-	-
	SFT - Correct Trace	52.88	70.63	79.45	66.33	47.06	78.99	45.8
	SFT - Incorrect Trace	63.88	76.5	82.58	73.5	20.36	56.92	34.15
Llama-3.2-1B-It	Prompt	7.48	13.78	17.58	12.15	-	-	-
	SFT - Vanilla	44.65	61.08	69.53	56.58	-	-	-
	SFT - Correct Trace	39.55	56.83	65.83	52.5	39.09	79.4	43.51
	SFT - Incorrect Trace	45.58	61.15	69.65	57.23	18.8	73.62	40.28

Table 3: Microsoft MARCO QA and Facebook bAbI QA Results

Model	Query Setting	Microsoft MARCO QA				Facebook bAbI QA			
		Avg Final Solution Accuracy (%)	Avg Trace Acc (Classification Step) (%)	Avg Trace Acc (IR Step) (%)	Avg Trace Length (# tokens)	Avg Final Solution Accuracy (%)	Avg Trace Acc (Classification Step) (%)	Avg Trace Acc (IR Step) (%)	Avg Trace Length (# tokens)
Qwen3-1.7B	Prompt	0	-	-	-	0	-	-	-
	SFT - Vanilla	3.4	-	-	-	97.9	-	-	-
	SFT - Correct Trace	26.3	60.4	40.6	68.14	94.41	61.7	24.73	43.25
	SFT - Incorrect Trace	20.3	6.9	52.5	85.07	95.21	17.82	0	42.45
Llama-3.2-1B-It	Prompt	1.7	-	-	-	12.8	-	-	-
	SFT - Vanilla	33.4	-	-	-	96.5	-	-	-
	SFT - Correct Trace	33.7	59.9	21.4	55.82	94.41	61.7	24.73	42.17
	SFT - Incorrect Trace	28.9	20	43.9	80.48	86.17	3.46	0	38.5

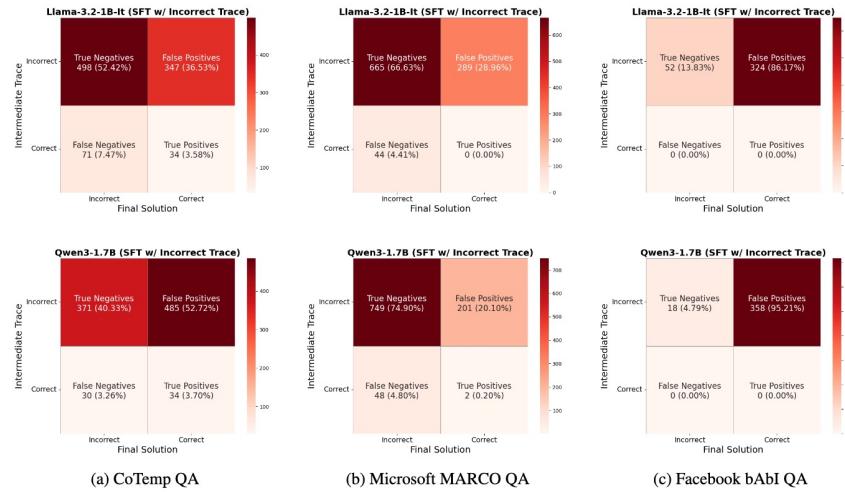


Figure 3: Confusion Matrices for **SFT w/ Incorrect Traces** on Llama-3.2-1B-It model (*top*) and Qwen3-1.7B (*bottom*) model, showing Final Solution Accuracy (X-axis) vs Trace Accuracy (Y-axis) for the CoTemp QA, Microsoft MARCO QA, and, the Facebook bAbI QA datasets.

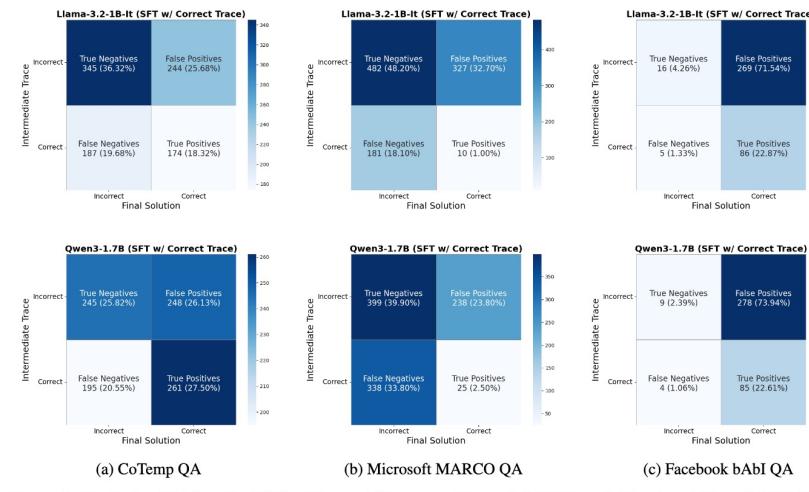
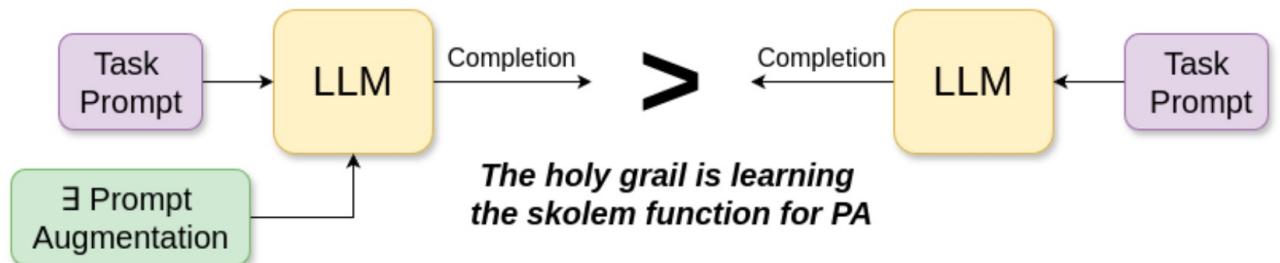
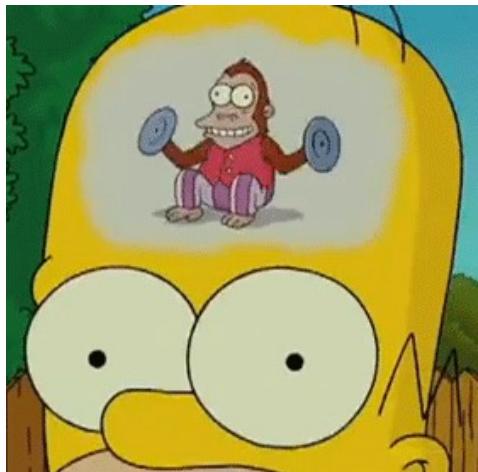


Figure 2: Confusion Matrices for **SFT w/ Correct Traces** on Llama-3.2-1B-It model (*top*) and Qwen3-1.7B (*bottom*) model, showing Final Solution Accuracy (X-axis) vs Trace Accuracy (Y-axis) for the CoTemp QA, Microsoft MARCO QA, and, the Facebook bAbI QA datasets.

Perils of “Reasoning Trace” Anthropomorphization

- Engender false (undeserved) trust in the end users
- Albatross on the RL phase



Subbarao Kambhampati (కంఫంపాటి సుబ్రామణి) ✨



...

Think "Prompt Augmentations" rather than chains and thoughts!
#SundayHarangue (#NeurIPS2024 edition 😊)

LLM CoT for reasoning discourse can be de-anthropomorphized with significant clarity, once we agree that *CoT* is a misleading fancy name for aspirational **prompt augmentation**. Let me explain.

The interest in prompt augmentations stems from the general observation (hope?) that the completions of LLMs given an task prompt and an appropriate prompt augmentation can be more "correct" than the completions of LLMs given just the task prompt.

That is, given a task prompt T,

$$\exists \text{PA s.t. } \Pr(\text{Sol}(\text{LLM}(T+\text{PA}), T)) > \Pr(\text{Sol}(\text{LLM}(T), T))$$

(where PA is some appropriate prompt augmentation, LLM(x) is the completion output by LLM given x as the prompt, and Sol(y,T) checks, oracularly, if y contains solution for T)

RL IN NAME ONLY? ANALYZING THE STRUCTURAL ASSUMPTIONS IN RL POST-TRAINING FOR LLMs

Soumya Rani Samineni*
SCAI, Arizona State University
ssamine4@asu.edu

Durgesh Kalwar*
SCAI, Arizona State University
dkalwar@asu.edu

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

Kaya Stechly
SCAI, Arizona State University
khstechl@asu.edu

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

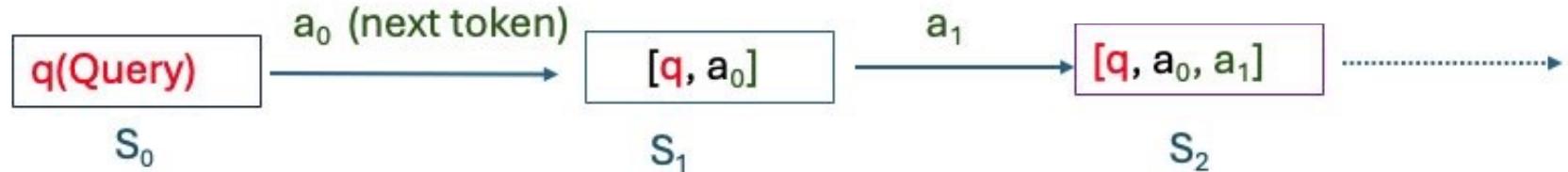
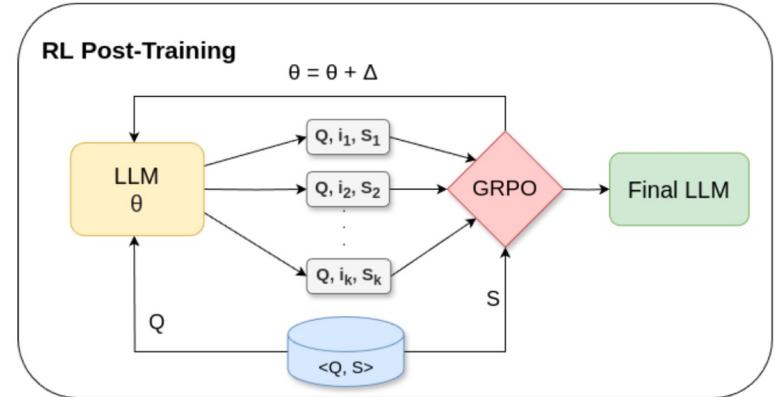
ABSTRACT

Reinforcement learning-based post-training of large language models (LLMs) has recently gained attention, particularly following the release of DeepSeek R1 [8], which applied GRPO for fine-tuning. Amid the growing hype around improved reasoning abilities attributed to RL post-training, we critically examine the formulation and assumptions underlying these methods. We start by highlighting the popular structural assumptions made in modeling LLM training as a Markov Decision Process (MDP), and show how they lead to a degenerate MDP that doesn't quite need the RL/GRPO apparatus. The two critical structural assumptions include (1) making the MDP states be just a concatenation of the actions–states becoming the context window and the actions becoming the tokens in LLMs and (2) splitting the reward of a state-action trajectory uniformly across the trajectory. Through a comprehensive analysis, we demonstrate that these simplifying assumptions make the approach effectively equivalent to an outcome-driven supervised learning. Our experiments on benchmarks including GSM8K and Countdown using Qwen-2.5 base models show that iterative supervised fine-tuning, incorporating both positive and negative samples, achieves performance comparable to GRPO-based training. We will also argue that the structural assumptions indirectly incentivize the RL to generate longer sequences of intermediate tokens—which in turn feeds into the narrative of “RL generating longer thinking traces.” While RL may well be a very useful technique for improving the reasoning abilities of LLMs, our analysis shows that the simplistic structural assumptions made in modeling the underlying MDP render the popular LLM RL frameworks and their interpretations questionable.

1 INTRODUCTION

Recent advances in Large Language Models (LLMs) have substantially improved their ability to perform complex reasoning tasks, such as mathematical problem-solving. Typically, this is achieved through post-training techniques, including Reinforcement Learning (RL) methods like Group Relative Policy Optimization (GRPO). These methods

RL Tuning on Derivational Traces



RL Tuning in DeepSeek R1 ~ Iterative SFT

- Transition Function:

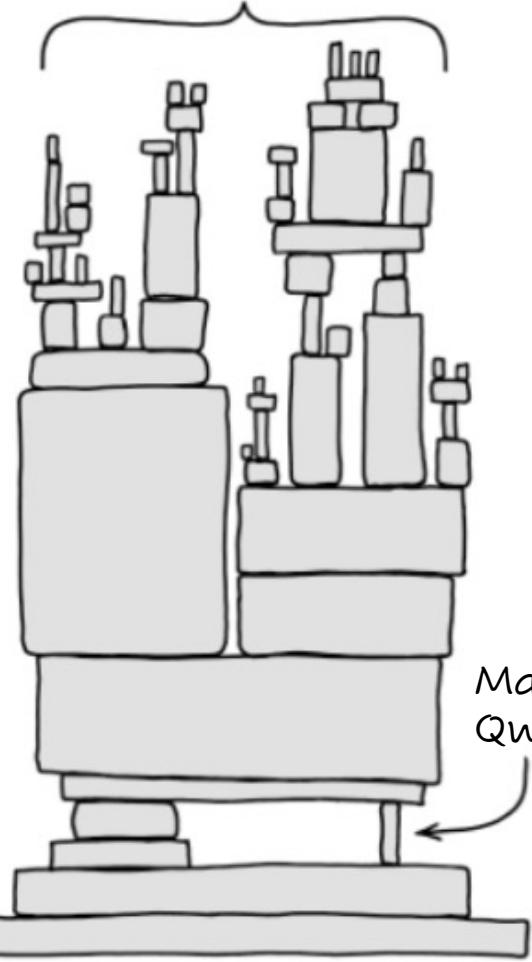
$$s_{t+1} = [s_t, a_t] \quad - \text{Deterministic and Stationary}$$

$$P(s_{t+1}|s_t, a_t) = 1$$

$$P(\tau|\theta) = p(s_0) \prod_{t=0}^{T-1} \pi_{LLM_\theta}(a_t|s_t) P(s_{t+1}|s_t, a_t)$$

$$P(\tau|\theta) = p(s_0) \prod_{t=0}^{T-1} \pi_{LLM_\theta}(a_t|s_t)$$

Head-spinning RL LLM results



Math Data on
Qwen Models

Confusing Intermediate Token Length as “Reasoning Effort”

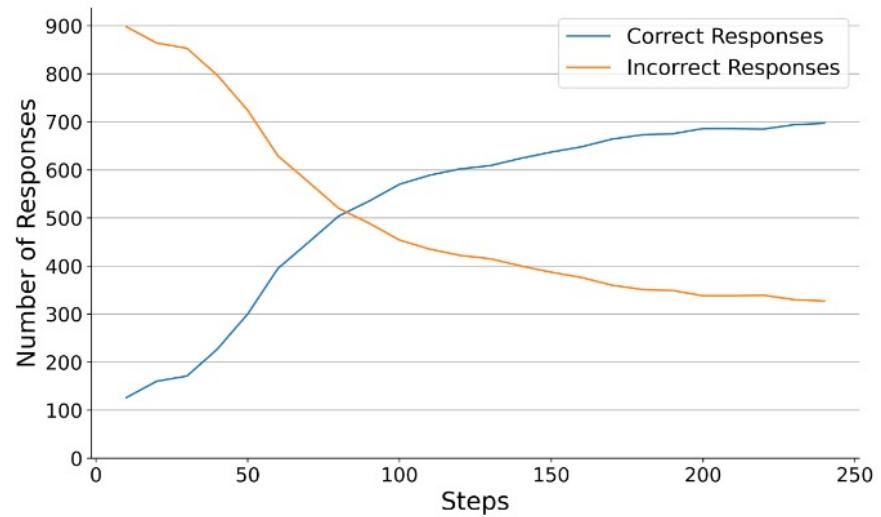
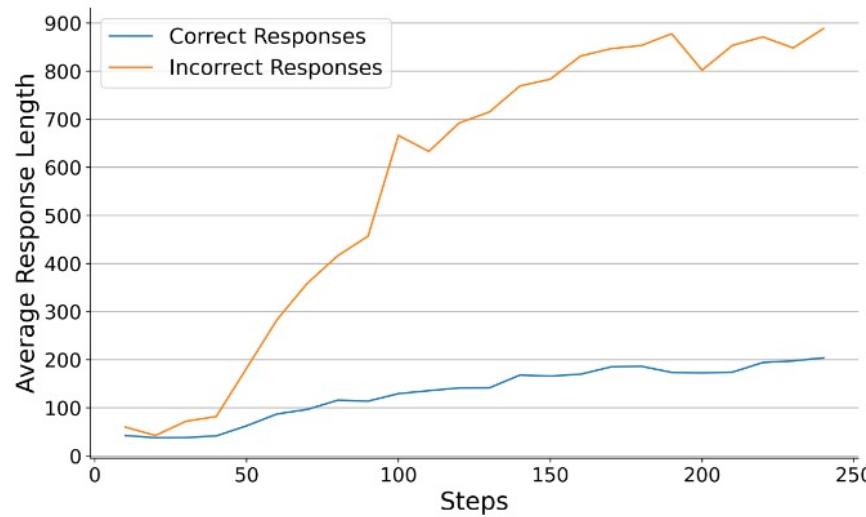


Figure 4: Left: Average response lengths for correct and incorrect responses. Right: Number of correct and incorrect responses, on the Countdown test dataset at each evaluation step during post-training of the Qwen-2.5-1.5B base model using the GRPO algorithm.

STOP ANTHROPOMORPHIZING INTERMEDIATE TOKENS AS REASONING/THINKING TRACES!

Subbarao Kambhampati Kaya Stechly Karthik Valmeekam Lucas Saldyt Siddhant Bhambri

Vardhan Palod Atharva Gundawar Soumya Rani Samineni Durgesh Kalwar Upasana Biswas

School of Computing & AI
Arizona State University

ABSTRACT

Intermediate token generation (ITG), where a model produces output before the solution, has been proposed as a method to improve the performance of language models on reasoning tasks. These intermediate tokens have been called “reasoning traces” or even “thoughts” – implicitly anthropomorphizing the model, implying these tokens resemble steps a human might take when solving a challenging problem. In this paper, we present evidence that this anthropomorphization isn’t a harmless metaphor, and instead is quite dangerous – it confuses the nature of these models and how to use them effectively, and leads to questionable research.

1 Introduction

Recent advances in general planning and problem solving have been spearheaded by so-called “Long Chain-of-Thought” models, most notably DeepSeek’s R1 [17]. These transformer-based large language models are further post-trained using iterative fine-tuning and reinforcement learning methods. Following the now-standard teacher-forced pre-training, instruction fine-tuning, and preference alignment stages, they undergo additional training on reasoning tasks: at each step, the model is presented with a question; it generates a sequence of intermediate tokens (colloquially or perhaps fancifully called a “Chain of Thought” or “reasoning trace”); and it ends it with a specially delimited answer sequence. After verification of this answer sequence by a formal system, the model’s parameters are updated so that it is more likely to output sequences that end in correct answers and less likely to output those that end in incorrect answers with no guarantees of trace correctness.

While (typically) no direct optimization pressure is applied to the intermediate tokens [4, 62], empirically it has been observed that language models perform better on many domains if they output such tokens first [38, 55, 61, 19, 16, 17, 39, 36, 29]. While the fact of the performance increase is well-known, the reasons for it are less clear. Much of the previous work has framed intermediate tokens in wishful anthropomorphic terms, claiming that these models are “thinking” before outputting their answers [38, 12, 17, 56, 62, 7]. The traces are thus seen both as giving insights to the end users about the solution quality, and capturing the model’s “thinking effort.”

Results on PlanBench as of 8/27/2024

Summary

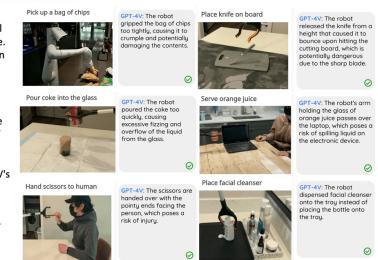
Domain		Claude-3.5-Sonnet	Claude 3 (Opus)	GPT-4o	GPT-4	GPT-4 Turbo	LLaMA-3.1 405B	LLaMA-3 70B	Gemini Pro
Blocksworld	One shot	346/600 (57.6%)	289/600 (48.1%)	170/600 (28.3%)	206/600 (34.3%)	138/600 (23%)	284/600 (47.3%)	76/600 (12.6%)	68/600 (11.3%)
	Zero shot	329/600 (54.8%)	356/600 (59.3%)	213/600 (35.5%)	210/600 (34.6%)	241/600 (40.1%)	376/600 (62.6%)	205/600 (34.16%)	3/600 (0.5%)
Mystery Blocksworld	One shot	19/600 (3.1%)	8/600 (1.3%)	5/600 (0.83%)	26/600 (4.3%)	5/600 (0.83%)	21/600 (3.5%)	15/600 (2.5%)	2/500 (0.4%)
	Zero shot	0/600 (0%)	0/600 (0%)	0/600 (0%)	1/600 (0.16%)	1/600 (0.16%)	5/600 (0.8%)	0/600 (0%)	0/500 (0%)

LLMs as Behavior Critics to catch undesirable robot behaviors

Can LLMs capture human preferences in embodied AI tasks?

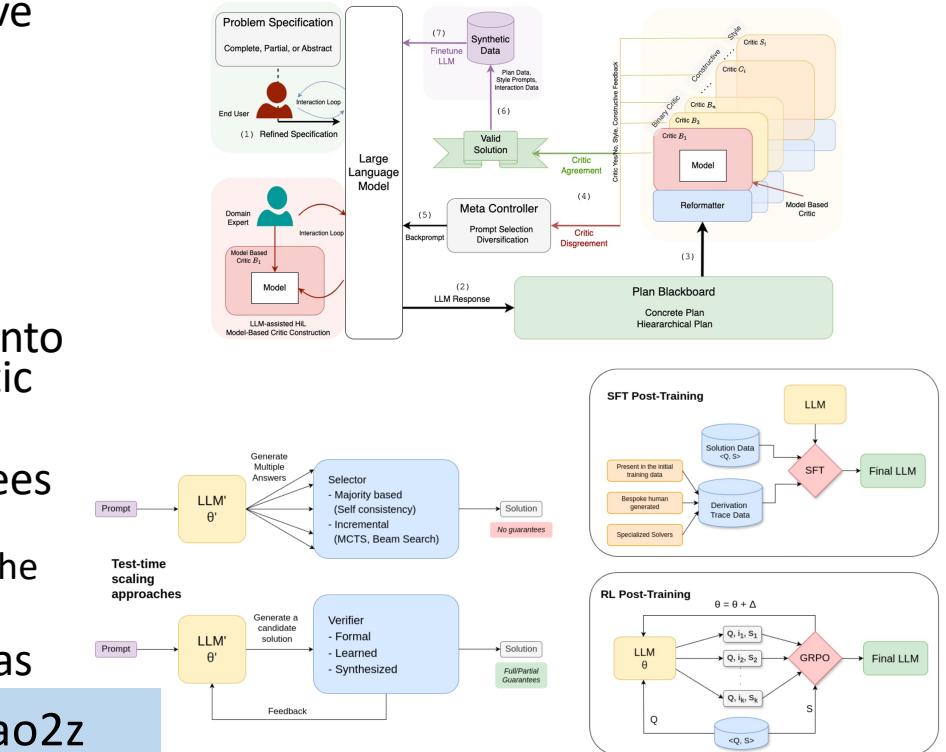
- It may be intractable to construct formal verifiers for tasks that have a wide scope.
- LLMs or VLMs can be a proxy of common human preferences and undesirability
- We evaluated GPT-4V with videos of diverse suboptimal behaviors
- GPT-4V critic catches 69% of undesirable behaviors (recall rate) while 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 Behaviors in Embodied AI Tasks
Lin Guan*, Yifan Zhou*, Dennis Liu, Yantian Zho, Heng Ben Amor, Subbarao Kambhampati, CDMR 2024



- LLMs trained just on web corpora were known to have severe limitations on planning/reasoning tasks
 - They could be good arbiters of style though..
- They could however be used as generators in a Generate-Test cycle to give robust planning performance; this LLM-Modulo is a form of test-time scaling
- The improved reasoning abilities can be “compiled” into base generator LLM with SFT or RL tuning on synthetic data
- The resulting LLMs still have no correctness guarantees (but are better generators)
 - Post-training *incrementally compiles* verifier signal into the LLM
- The anthropomorphization of intermediate tokens as “reasoning traces” is questionable

Twitter: @rao2z



Reasoning & Planning Abilities of LLMs/LRMs

Lecture 1

Subbarao Kambhampati
ASU Arizona State University

Twitter: @rao2z



<https://bit.ly/444Yn9>



AC 52:52

Reasoning & Planning Abilities of LLMs/LRMs

Lecture 2

Subbarao Kambhampati
ASU Arizona State University

Twitter: @rao2z



<https://bit.ly/444Yn9>



AC 52:27

Reasoning & Planning Abilities of LLMs/LRMs

Lecture 3

Subbarao Kambhampati
ASU Arizona State University

Twitter: @rao2z



<https://bit.ly/444Yn9>



AC 55:49

<https://bit.ly/3ZFtist>



Planning & Reasoning Abilities of LLMs/LRMs: The Agenda

- Lecture 1: Big Picture
 - A perspective on LLMs and their approximate retrieval
 - Understanding reasoning/planning (in)abilities of standard LLMs
 - Ideas for improving: LLMs as generators; Test Time Scaling as Generate-Test (“LLM-Modulo”); Post-training as Generate-Test-Compile during training
- Lecture 2: Drill down on planning abilities of standard LLMs
 - Out of the box, with Chain-of-thought, with ReACT, with self-verification etc.
- Lecture 3: Drill down on test-time scaling (LLM-Modulo); Post-training approaches (Supervised fine tuning vs. RL)
 - Understanding some continuing misconceptions

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 steroids^a that have been trained on web-scale language corpora (or, effectively, our civilization's knowledge), have caught our collective imagination with linguistic behaviors that no one expected text completion systems to possess.^b 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 LLMs are good at is a form of universal approximate retrieval. Unlike databases that index and retrieve data exactly, LLMs, as *n*-gram models, probabilistically reconstruct completions for the prompt word by word—a process we shall refer to as *approximate retrieval*.^b

Despite this, "Large Language Models are Zero-Shot (*insert-your-reasoning-test*)" has almost become a meme paper title! At some level, this trend is perhaps inevitable as in the era of LLMs, AI has become a form of ersatz natural science^c—driven by observational studies of capabilities of these behemoth systems.

So, are these *n*-gram models on steroids really capable of planning and reasoning? In the summer of 2022, when my research group wanted to better answer this question, most reasoning claims were still somewhat anecdotal. So, we set out to evaluate GPT3 on a set of planning instances derived from the domains typically used in the Inter-

^a LLMs are trained to predict the distribution of the *n*-th token given *n*-1 previous tokens. GPT3 that powered the original ChatGPT is, for example, a roughly 3001-gram model of language.

^b This means that LLMs can't even guarantee memorizing complete answers, something that is the flip side of their appeal about constructing "novel" prompt completions on the fly. The boon ("creativity") and bane ("hallucination") of LLMs is that *n*-gram models will naturally mix and match—and have almost as much trouble strictly memorizing as we do. It is indeed the very basis of their appeal.

(How) Do reasoning models reason?

Subbarao Kambhampati  | Kaya Stechly | Karthik Valmeekam

School of Computing & Augmented Intelligence, Arizona State University, Tempe, Arizona, USA

Correspondence
Subbarao Kambhampati, School of Computing, and Augmented Intelligence, Arizona State University, Tempe, AZ 85287, USA.
Email: rao@asu.eduFunding information
Office of Naval Research, Grant/Award Number: N0001423-12409; Qualcomm; Defense Advanced Research Projects Agency, Grant/Award Number: HR00112520016**Abstract**

We will provide a broad unifying perspective on the recent breed of large reasoning models such as OpenAI o1 and DeepSeek R1, including their promise, sources of power, misconceptions, and limitations.

KEYWORDS

large language models, reasoning, planning, reinforcement learning post-training, test-time inference

INTRODUCTION

Large language models (LLMs), which have been autoregressively trained on humanity's digital footprint, have shown the ability to generate coherent text responses to a vast variety of prompts. Although they show impressive System 1 capabilities and excel in producing completions that mimic appropriate styles, System 2 capabilities like factuality, reasoning, and planning have remained elusive aims, if not Achilles heels.¹

In response, researchers have developed a new breed of models—sometimes called *large reasoning models* (LRMs)—which build on vanilla LLM architectures and training recipes. The best-known of these are OpenAI's o1 and DeepSeek's R1, which have shown significant performance improvements on reasoning and planning tasks previously outside the range of older LLM capabilities.

These models have been built on insights from two broad but largely orthogonal classes of ideas: (i) test-time inference scaling techniques, which involve getting LLMs to do more work than simply providing the most likely direct answer; and (ii) post-training methods, which complement simple autoregressive training on web corpora, with additional training on data containing intermediate tokens. In this article, we will use the term "derivational traces" as a neutral stand-in for these intermediate tokens, rather than the more popular anthropomorphized phrases "chains of thought" and "reasoning traces."

Although these ideas are leading to performance leaps on benchmarks, there is little consensus yet on when and why they work.

Test-time scaling approaches that leverage sound verification can be made sense of because they enforce guarantees on the output; but other popular methods are only justified empirically, working in some domains but not in others. There is even more confusion about the success of new post-training techniques. As we will argue, the prevalent narrative—that novel generalization capabilities arise from training LLMs on "chains of thought" so that they produce intermediate tokens dubbed "reasoning traces" before solutions—is questionable.

Our aim in this article is to provide a broad unifying perspective on these approaches, including their promise, sources of power, misconceptions, and limitations.

BUILDING REASONING MODELS**Test-time inference**

Not all problems require an equal amount of effort or time. A two digit by two digit addition problem can be solved with just three one-digit additions, while a four by four digit problem may require seven. There is a rich history of approaches that use scalable online computation to improve upon faster initial guesses, including limited depth min-max, real-time A* search and dynamic programming, and Monte Carlo Tree Search.² Test-time inference approaches (see Figure 1) mirror these ideas and are inspired by the anthropomorphic observation that people



DO REASONING MODELS ACTUALLY SEARCH?

Machine Learning Street Talk • 33K views • 1 month ago



Do you think that ChatGPT can reason?

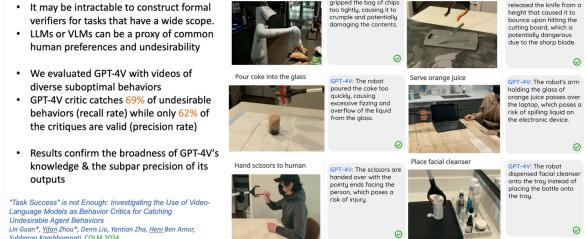
Machine Learning Street Talk • 73K views • 6 months ago

Results on PlanBench as of 8/27/2024

Summary

Domain		Claude-3.5-Sonnet	Claude 3 (Opus)	GPT-4o	GPT-4	GPT-4 Turbo	LLaMA-3.1 405B	LLaMA-3 70B	Gemini Pro
Blocksworld	One shot	346/600 (57.6%)	289/600 (48.1%)	170/600 (28.3%)	206/600 (34.3%)	138/600 (23%)	284/600 (47.3%)	76/600 (12.6%)	68/600 (11.3%)
	Zero shot	329/600 (54.8%)	356/600 (59.3%)	213/600 (35.5%)	210/600 (34.6%)	241/600 (40.1%)	376/600 (62.6%)	205/600 (34.16%)	3/600 (0.5%)
Mystery Blocksworld	One shot	19/600 (3.1%)	8/600 (1.3%)	5/600 (0.83%)	26/600 (4.3%)	5/600 (0.83%)	21/600 (3.5%)	15/600 (2.5%)	2/500 (0.4%)
	Zero shot	0/600 (0%)	0/600 (0%)	0/600 (0%)	1/600 (0.16%)	1/600 (0.16%)	5/600 (0.8%)	0/600 (0%)	0/500 (0%)

LLMs as Behavior Critics to catch undesirable robot behaviors
Can LLMs capture human preferences in embodied AI tasks?



- LLMs trained just on web corpora were known to have severe limitations on planning/reasoning tasks
 - They could be good arbiters of style though..
- They could however be used as generators in a Generate-Test cycle to give robust planning performance; this LLM-Modulo is a form of test-time scaling
- The improved reasoning abilities can be “compiled” into base generator LLM with SFT or RL tuning on synthetic data
- The resulting LLMs still have no correctness guarantees (but are better generators)
 - Post-training *incrementally compiles* verifier signal into the LLM
- The anthropomorphization of intermediate tokens as “reasoning traces” is questionable

Twitter: @rao2z

