End-to-End Planning with Large Language Models

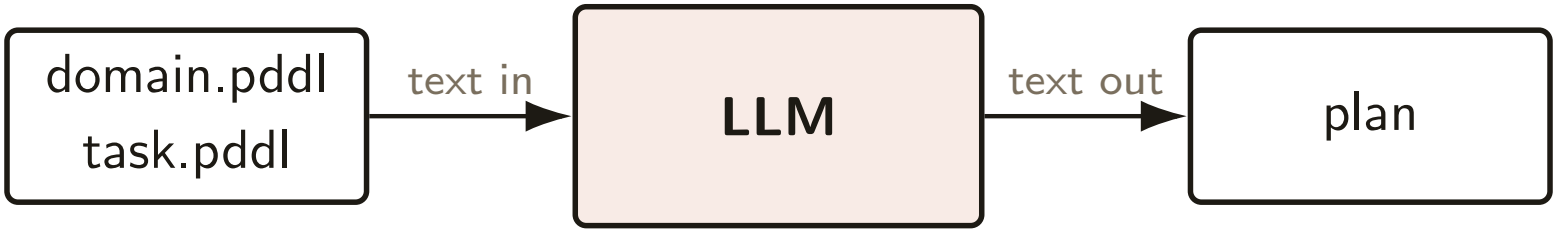
From "LLMs Can't Plan" to the Frontier

Augusto B. Corrêa

University of Oxford, United Kingdom

ICAPS 2026 Summer School, Dublin

This lecture: the LLM is the planner

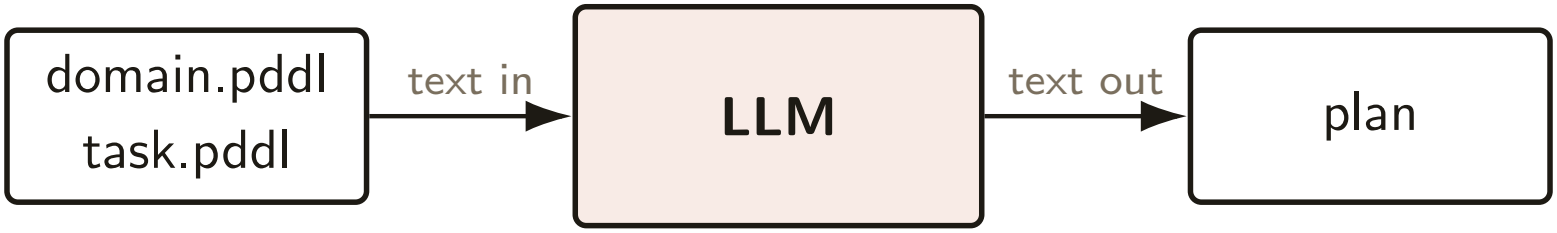


no search and no planner in the loop

Plans are checked *afterwards*[1]: valid or invalid.

[1] Howey & Long (2003). VAL's Progress: The Automatic Validation Tool for PDDL2.1.

This lecture: the LLM is the planner



no search and no planner in the loop

Plans are checked *afterwards*[1]: valid or invalid.

Focusing on classical planning
⇒ i.e., Gabi's lectures.

LLMs *helping* planners:
⇒ Michael Katz's part.

[1] Howey & Long (2003). VAL's Progress: The Automatic Validation Tool for PDDL2.1.

What a language model does

"The best thing about AI is its ability to _____"

[1] Example from S. Wolfram, "What Is ChatGPT Doing... and Why Does It Work?" (2023).

What a language model does

"The best thing about AI is its ability to _____"

next word	probability
`learn`	4.5%
`predict`	3.5%
`make`	3.2%
`understand`	3.1%
`do`	2.9%
⋮	

[1] Example from S. Wolfram, "What Is ChatGPT Doing... and Why Does It Work?" (2023).

What a language model does

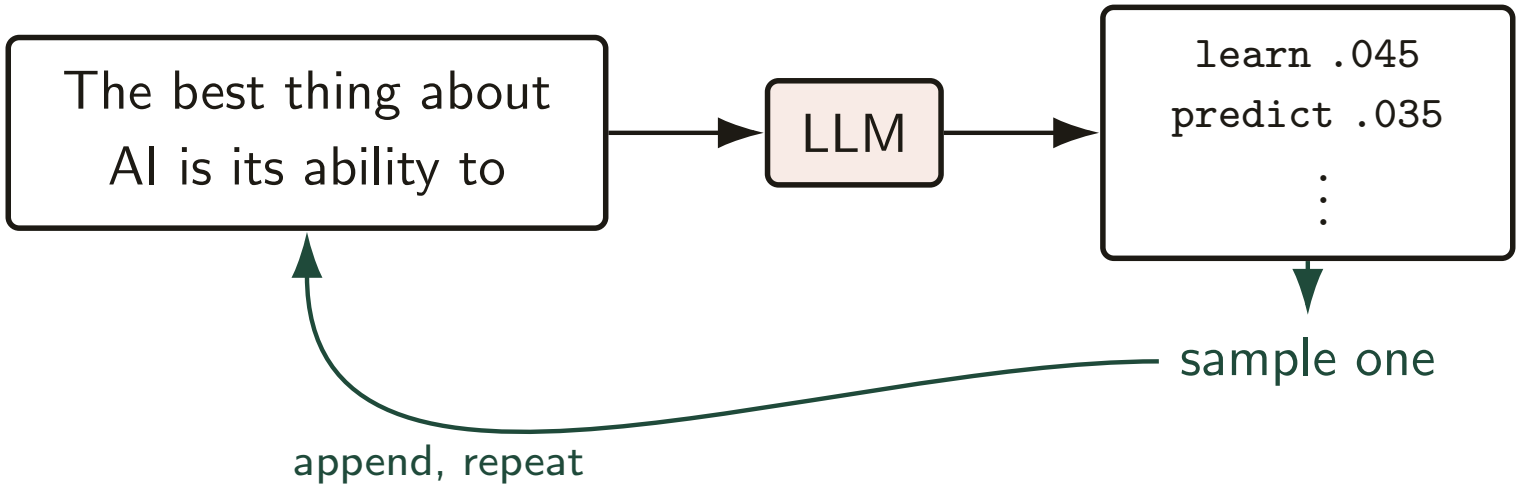
"The best thing about AI is its ability to _____"

next word	probability
`learn`	4.5%
`predict`	3.5%
`make`	3.2%
`understand`	3.1%
`do`	2.9%
⋮	

Output a probability distribution over what comes next.[1]

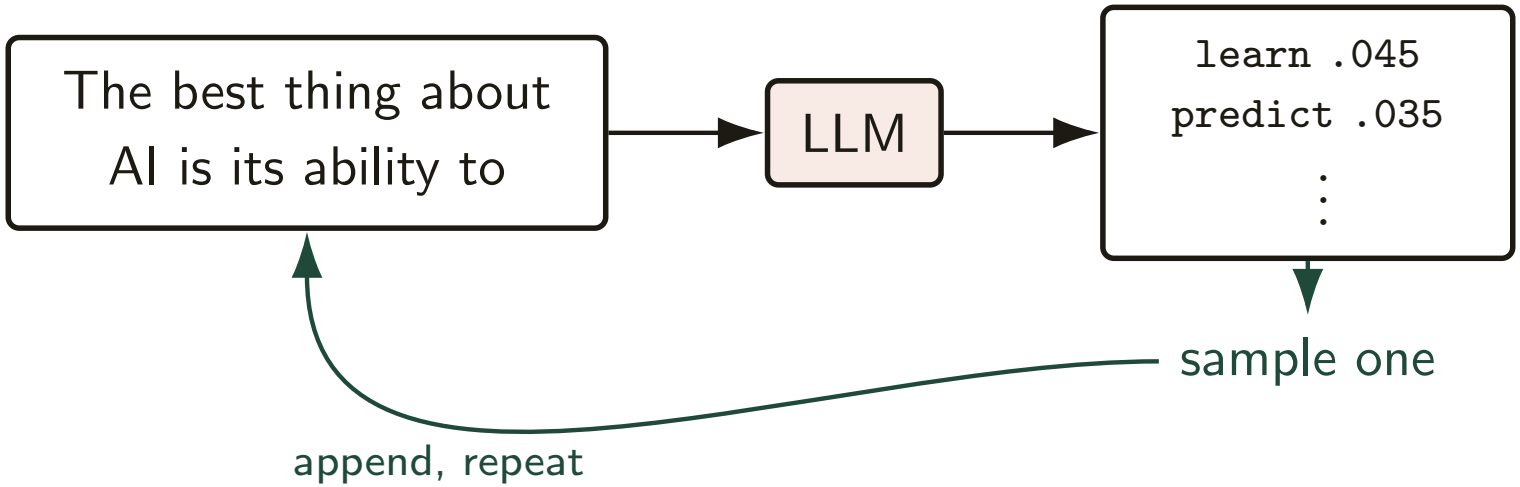
[1] Example from S. Wolfram, "What Is ChatGPT Doing... and Why Does It Work?" (2023).

. . . applied over and over



The best thing about AI is its ability to learn

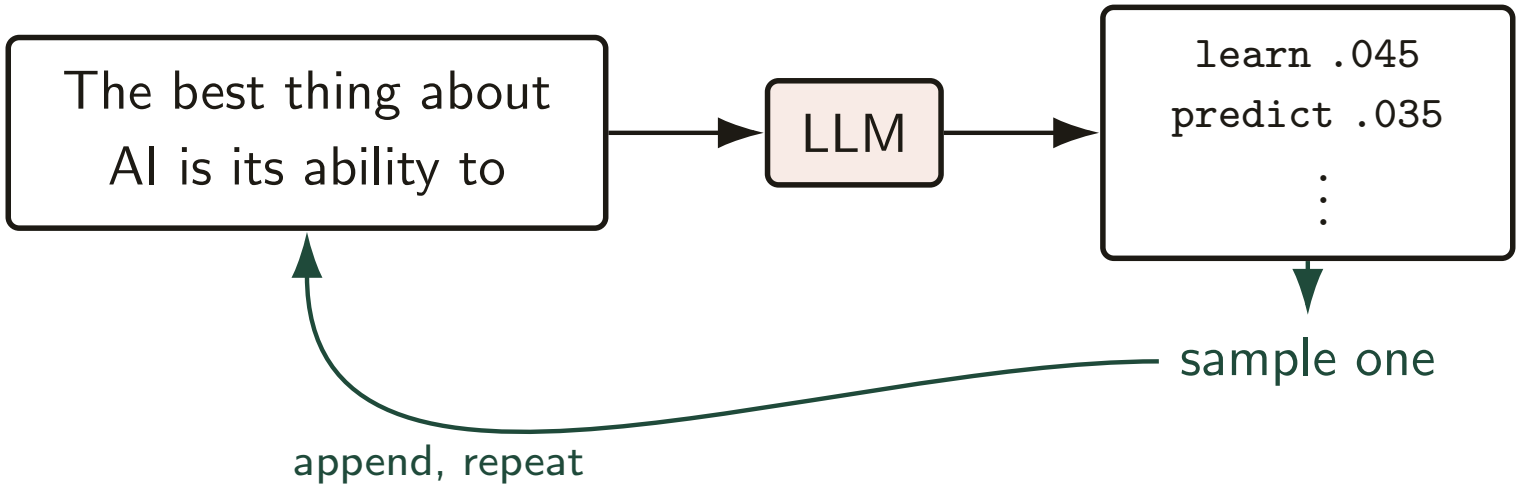
. . . applied over and over



The best thing about AI is its ability to learn

The best thing about AI is its ability to learn from

. . . applied over and over

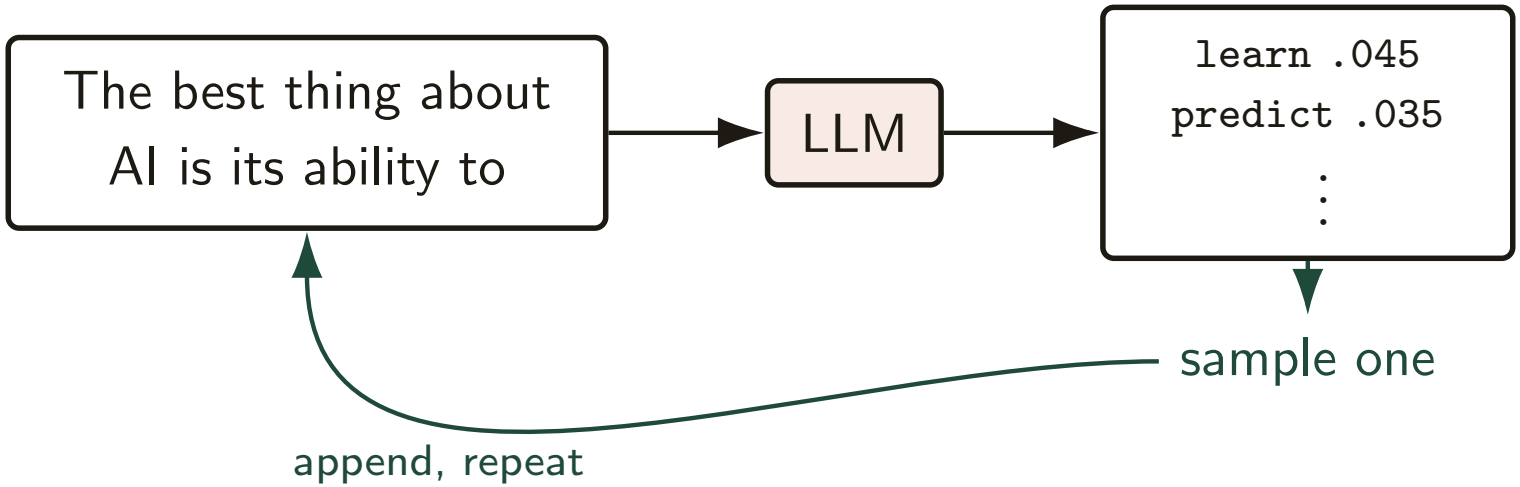


The best thing about AI is its ability to learn

The best thing about AI is its ability to learn from

The best thing about AI is its ability to learn from data

. . . applied over and over



The best thing about AI is its ability to learn

The best thing about AI is its ability to learn from

The best thing about AI is its ability to learn from data

- Autoregressive: left to right, conditioned on everything before. No lookahead, no backtracking.
- Always taking the top choice reads "flat"; some randomness ("temperature") works better

Tokens, not words

Common English $\approx$ one token per word.

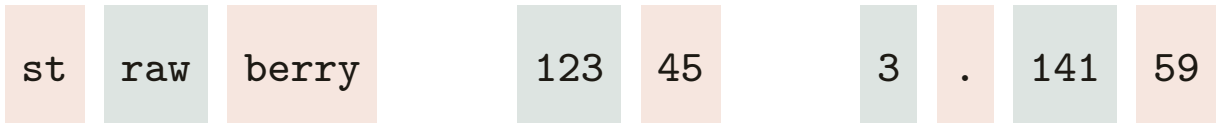
The rest is cut into pieces:[1]

[1] Real tokenizations (GPT-4o tokenizer, `o200k_base`).

Tokens, not words

Common English ≈ one token per word.

The rest is cut into pieces:[1]

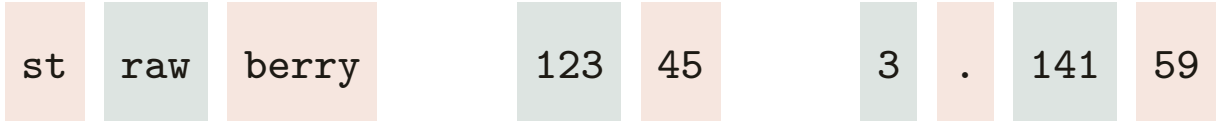


[1] Real tokenizations (GPT-4o tokenizer, o200k_base).

Tokens, not words

Common English $\approx$ one token per word.

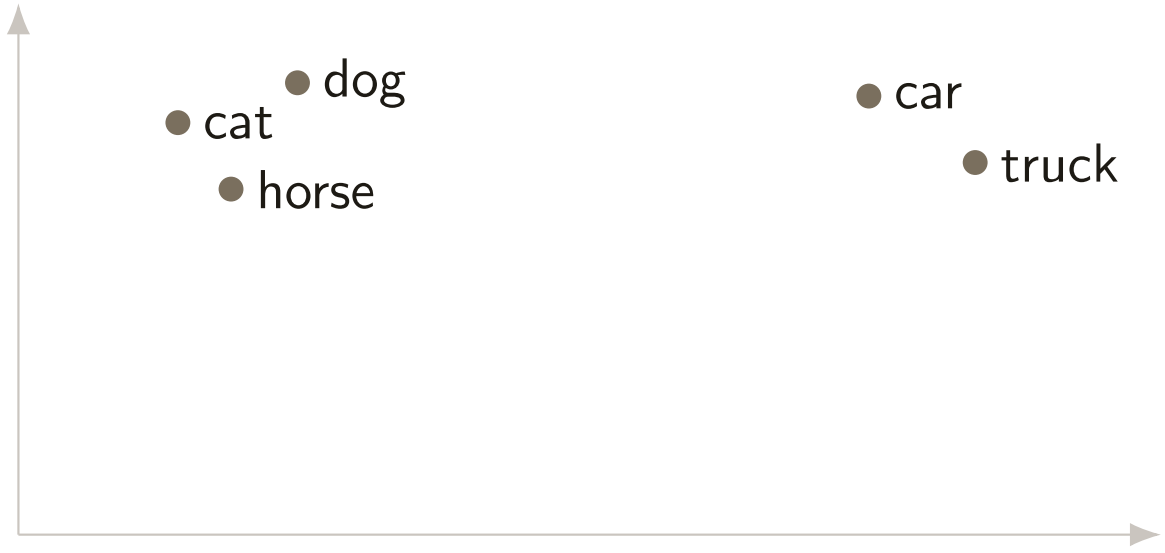
The rest is cut into pieces:[1]



- Tokenization affects performance

[1] Real tokenizations (GPT-4o tokenizer, o200k_base).

Inside, every token is a vector



- Inside the network, each token is a long list of numbers
- Tokens used in similar contexts end up nearby[1]

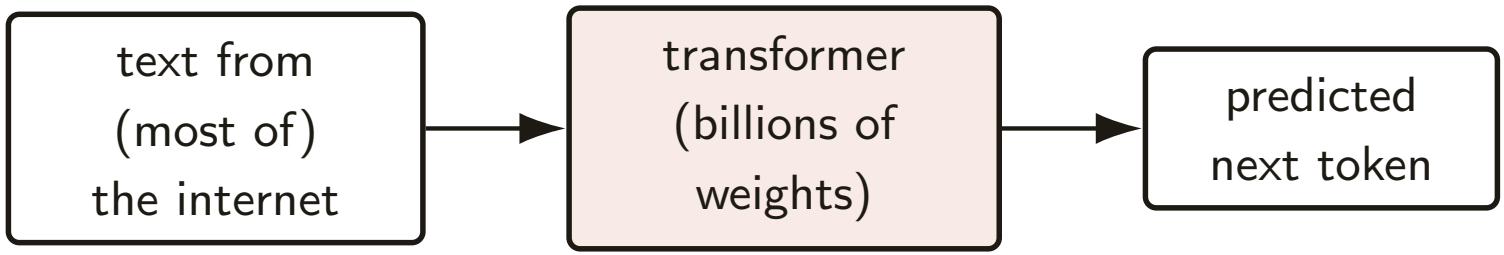
[1] Schematic; see Wolfram (2023) for projections of real embeddings.

Inside, every token is a vector

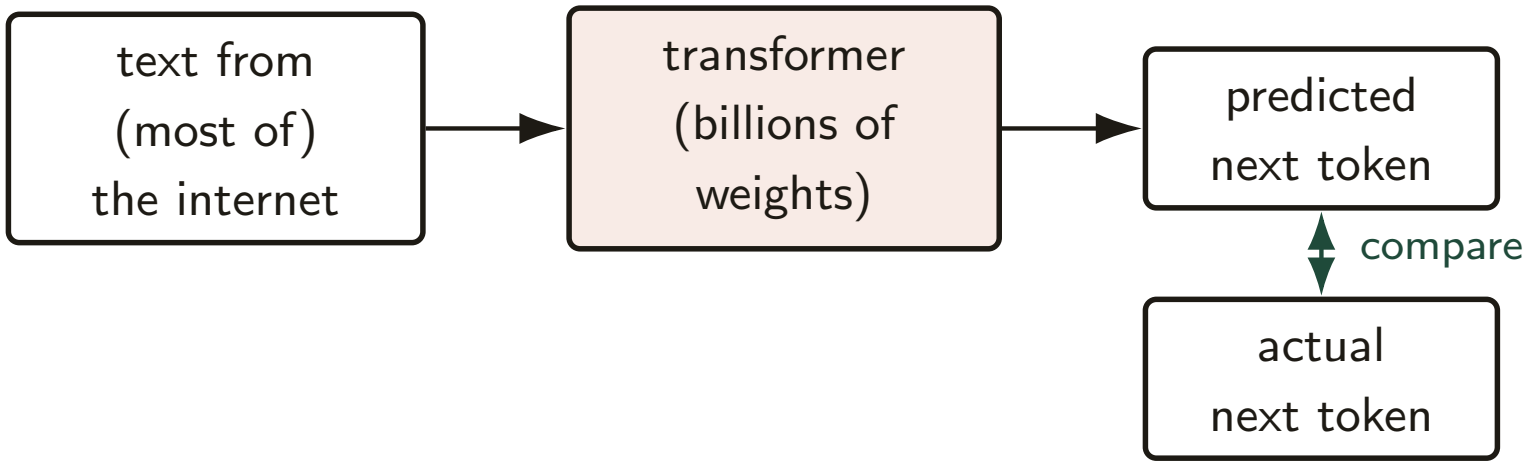
- Inside the network, each token is a long list of numbers
- Tokens used in similar contexts end up nearby[1]

[1] Schematic; see Wolfram (2023) for projections of real embeddings.

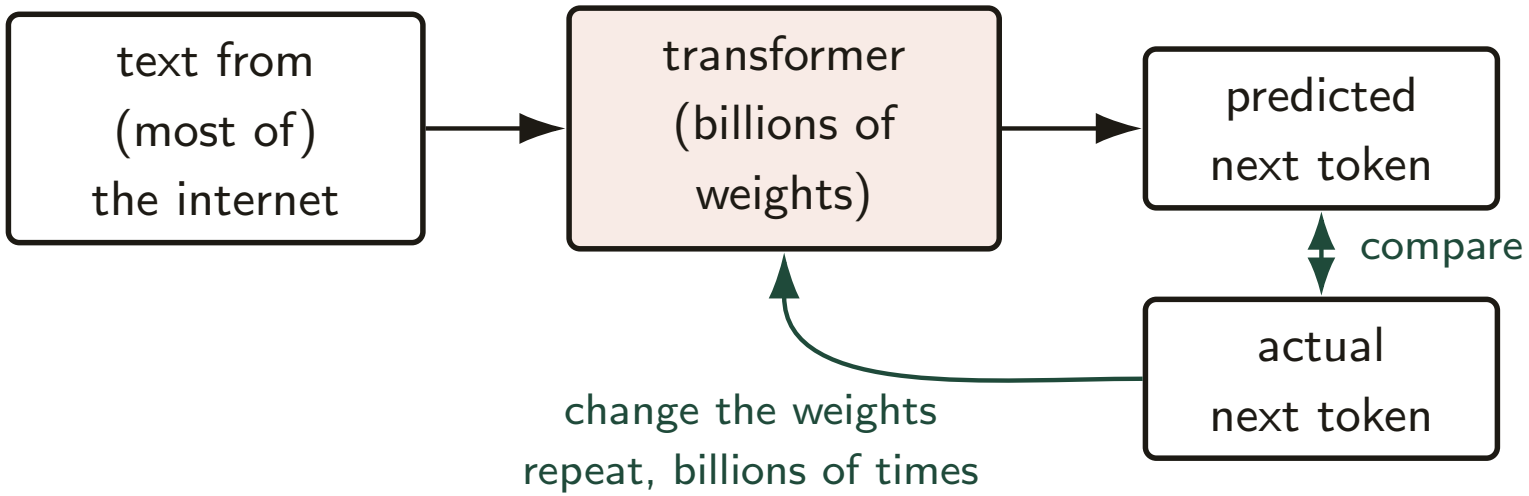
Where do the probabilities come from?



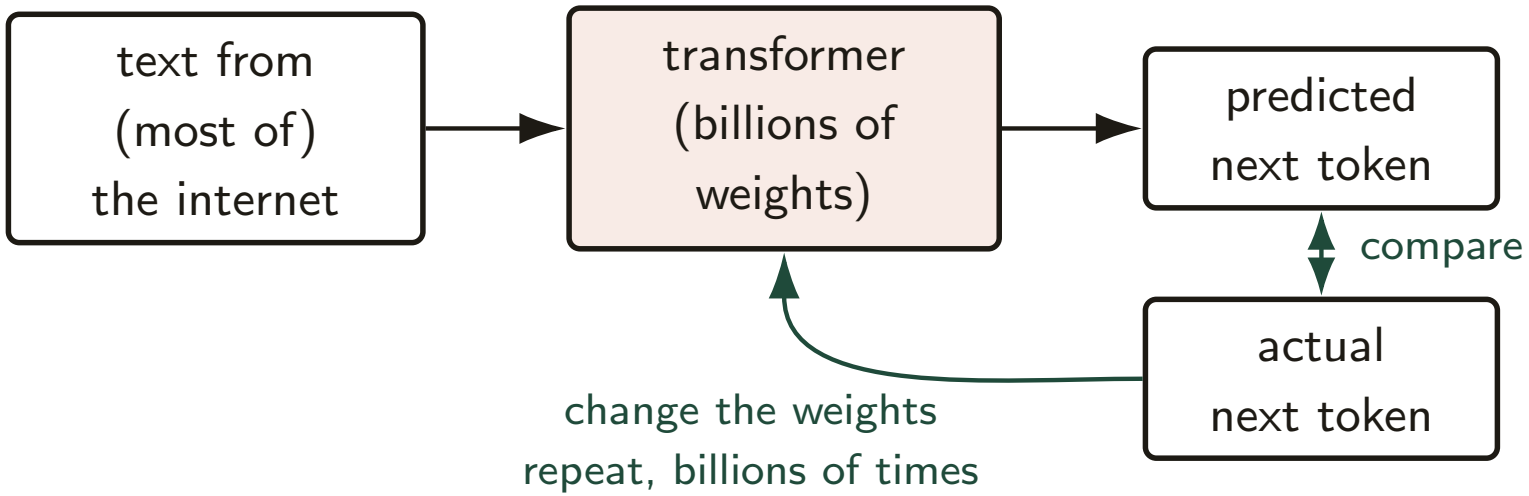
Where do the probabilities come from?



Where do the probabilities come from?

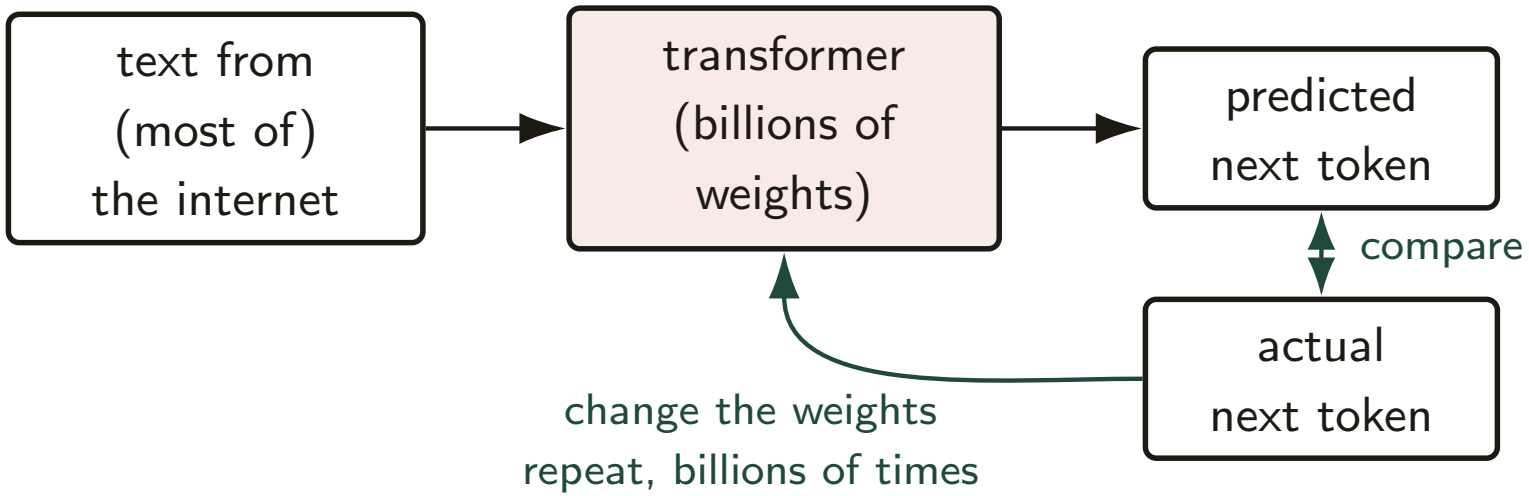


Where do the probabilities come from?



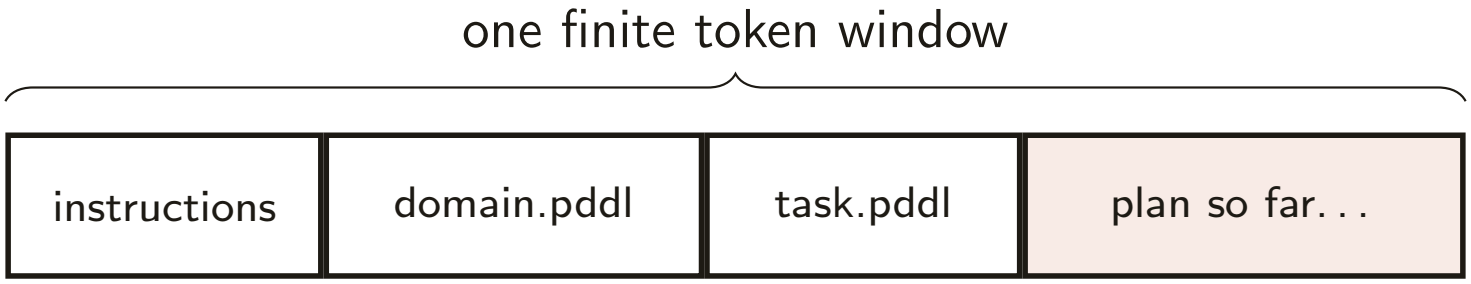
- GPT-3 (2020): 175 billion parameters, ∼300 billion training tokens

Where do the probabilities come from?

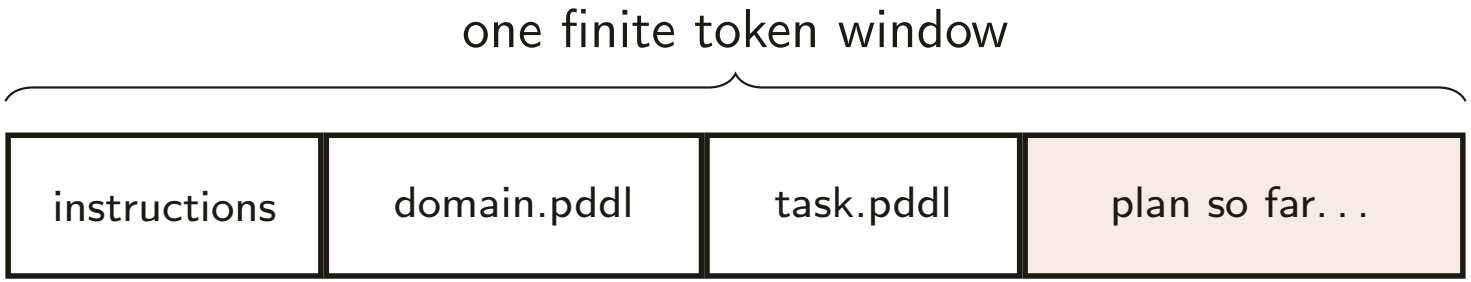


- GPT-3 (2020): 175 billion parameters, ∼300 billion training tokens
- No grammar, logic, or PDDL semantics programmed in: all learned from text

The context window

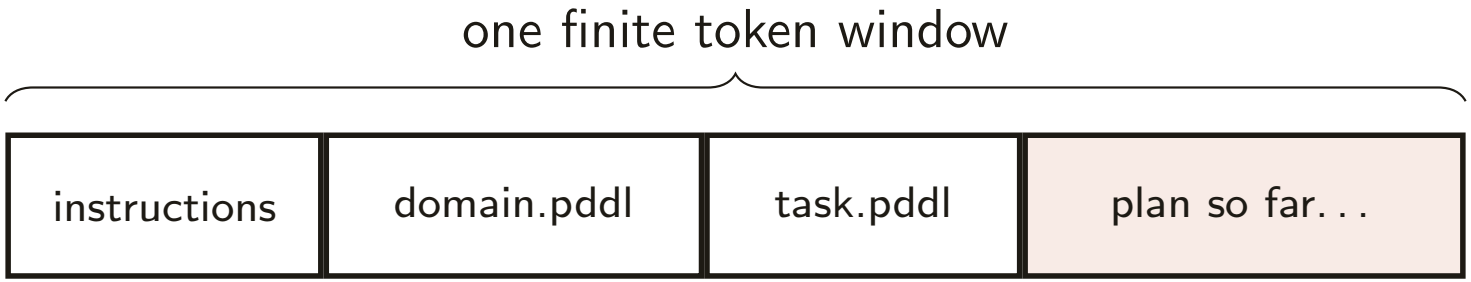


The context window



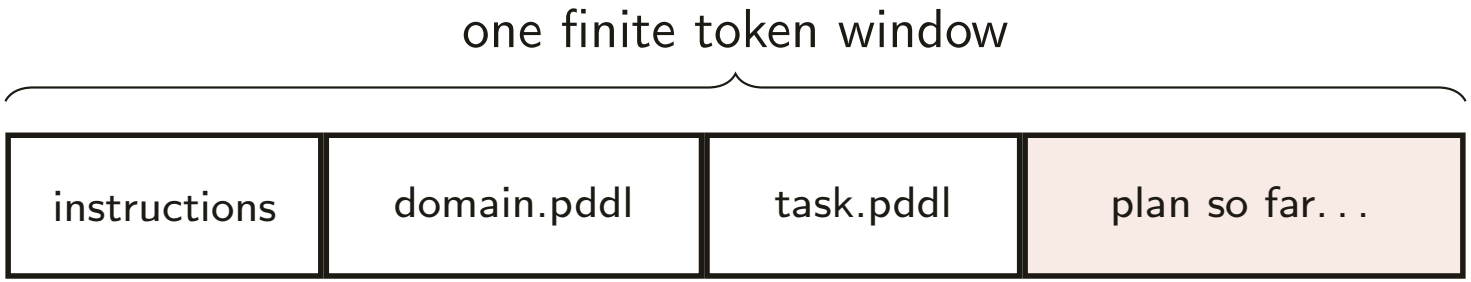
- Everything the model “knows” about your task is in this window, plus whatever is frozen in its weights

The context window



- Everything the model "knows" about your task is in this window, plus whatever is frozen in its weights
- Window sizes grew fast:
  2k tokens (GPT-3, 2020), 32k (GPT-4, 2023), $>$ a million today

The context window

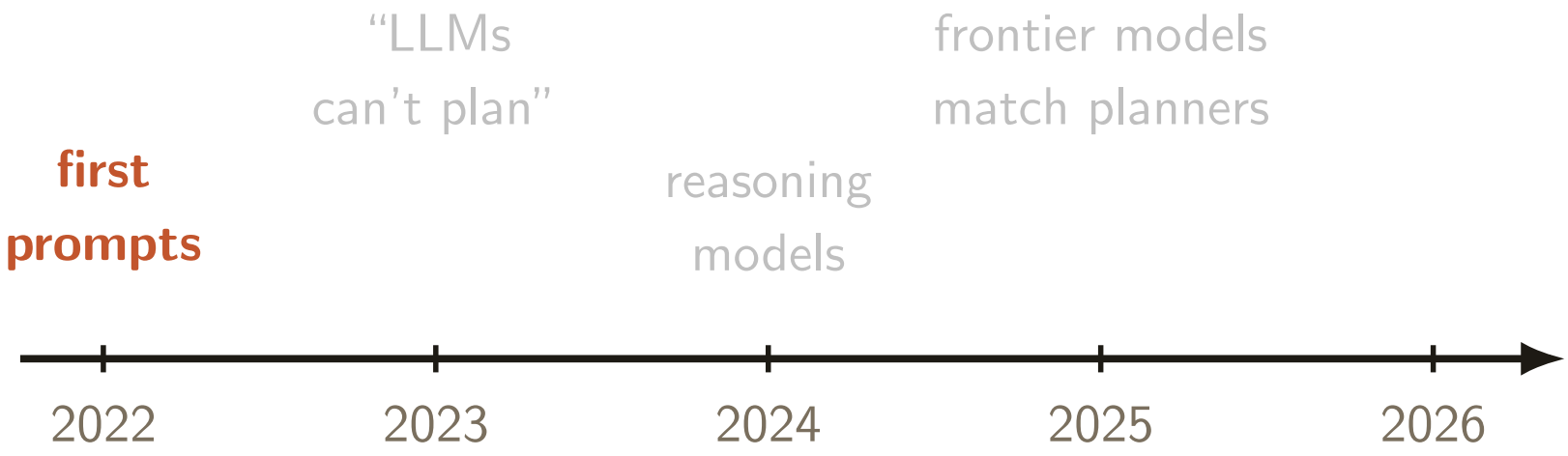


- Everything the model "knows" about your task is in this window, plus whatever is frozen in its weights
- Window sizes grew fast:
  2k tokens (GPT-3, 2020), 32k (GPT-4, 2023), $>$ a million today
- For planning: a 1 000-step plan is a very long output
  - generated one token at a time, no way to edit earlier steps

A next-token predictor.

A next-token predictor.

Why would it plan?


"LLMs
can't plan"
frontier models
match planners
first
prompts
reasoning
models
2022
2023
2024
2025
2026

Why might prompting work? In-context learning

```
;; example task
(:objects b1 b2)
...
(:goal (on b1 b2))
;; example plan
(unstack b2 b1)
(put-down b2)
(pick-up b1)
(stack b1 b2)

;; your task
(:objects b1 ...  b5)
...
(:goal (and ...))
;; plan:
?
```

Few-shot:

Shown examples *in the prompt*, large models continue the pattern, without any training update[1].

[1] Brown et al. (2020). Language Models are Few-Shot Learners.

Why might prompting work? In-context learning

```
;; example task
(:objects b1 b2)
...
(:goal (on b1 b2))
;; example plan
(unstack b2 b1)
(put-down b2)
(pick-up b1)
(stack b1 b2)

;; your task
(:objects b1 ...  b5)
...
(:goal (and ...))
;; plan:
?
```

Few-shot:

Shown examples *in the prompt*, large models continue the pattern, without any training update[1].

The hope in 2022: planning might be one more pattern to continue.

[1] Brown et al. (2020). Language Models are Few-Shot Learners.

2022: first attempts

- Few-shot prompt a pretrained LLM with domain + task PDDL, ask for a plan[1]
  - Silver et al.: few-shot prompting across 18 domains, one of the first systematic attempts

[1] Silver et al. (2022); Valmeekam et al. (2022).

2022: first attempts

- Few-shot prompt a pretrained LLM with domain + task PDDL, ask for a plan[1]
  - Silver et al.: few-shot prompting across 18 domains, one of the first systematic attempts
- Output *looks* like a plan: right syntax, right action names, plausible structure

[1] Silver et al. (2022); Valmeekam et al. (2022).

2022: first attempts

- Few-shot prompt a pretrained LLM with domain + task PDDL, ask for a plan[1]
  - Silver et al.: few-shot prompting across 18 domains, one of the first systematic attempts
- Output *looks* like a plan: right syntax, right action names, plausible structure
- VAL disagrees: almost nothing is valid

[1] Silver et al. (2022); Valmeekam et al. (2022).

2022: first attempts

- Few-shot prompt a pretrained LLM with domain + task PDDL, ask for a plan[1]
  - Silver et al.: few-shot prompting across 18 domains, one of the first systematic attempts
- Output *looks* like a plan: right syntax, right action names, plausible structure
- VAL disagrees: almost nothing is valid

A recurring pattern

Fluent, well-formatted, and invalid.

[1] Silver et al. (2022); Valmeekam et al. (2022).

What does the model actually see?

[1] Real tokenization (GPT-4o tokenizer, `o200k_base`).

What does the model actually see?

- PDDL is not one token per symbol: the model never sees `unstack`, b3, or your predicates as units[1]

[1] Real tokenization (GPT-4o tokenizer, `o200k_base`).

What does the model actually see?

- PDDL is not one token per symbol: the model never sees `unstack`, b3, or your predicates as units[1]
- It continues a sequence of familiar fragments; nothing ties `(un stack` to preconditions or effects

[1] Real tokenization (GPT-4o tokenizer, `o200k_base`).

What does the model actually see?

- PDDL is not one token per symbol: the model never sees `unstack`, b3, or your predicates as units[1]
- It continues a sequence of familiar fragments; nothing ties `(un` `stack` to preconditions or effects
- Fluent but invalid output is the default behavior of this setup

[1] Real tokenization (GPT-4o tokenizer, `o200k_base`).

Meanwhile, the debate

"Given the breadth and depth of GPT-4's capabilities, we believe that it could reasonably be viewed as an early (yet still incomplete) version of an artificial general intelligence (AGI) system."

Bubeck et al. (2023),

"Sparks of Artificial General Intelligence"

Meanwhile, the debate

"Given the breadth and depth of GPT-4's capabilities, we believe that it could reasonably be viewed as an early (yet still incomplete) version of an artificial general intelligence (AGI) system."

Bubeck et al. (2023),

"Sparks of Artificial General Intelligence"

"We argue that auto-regressive LLMs cannot, by themselves, do planning or self-verification."

Kambhampati et al. (ICML 2024),

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

The benchmark: PlanBench

- Blocksworld tasks with 3–5 blocks, optimal plans $\leq$ 16 steps[1]

[1] Valmeekam et al. (2023). PlanBench. NeurIPS 2023.

The benchmark: PlanBench

- Blocksworld tasks with 3–5 blocks, optimal plans ≤ 16 steps[1]
- 600 instances, every plan checked formally

[1] Valmeekam et al. (2023). PlanBench. NeurIPS 2023.

The benchmark: PlanBench

- Blocksworld tasks with 3–5 blocks, optimal plans $\leq$ 16 steps[1]
- 600 instances, every plan checked formally
- Plus two obfuscated twins: Mystery (misleading words) and Random Mystery (random strings)

Blocksworld	Mystery	Random Mystery
`(pick-up ?b)`	`(attack ?b)`	`(jqzkv ?b)`
`(stack ?a ?b)`	`(overcome ?a ?b)`	`(pvxen ?a ?b)`
`(clear ?b)`	`(province ?b)`	`(dwtgh ?b)`
`(arm-empty)`	`(harmony)`	`(rkfsa)`

[1] Valmeekam et al. (2023). PlanBench. NeurIPS 2023.

The benchmark: PlanBench

- Blocksworld tasks with 3–5 blocks, optimal plans $\leq$ 16 steps[1]
- 600 instances, every plan checked formally
- Plus two obfuscated twins: Mystery (misleading words) and Random Mystery (random strings)

Blocksworld	Mystery	Random Mystery
`(pick-up ?b)`	`(attack ?b)`	`(jqzkv ?b)`
`(stack ?a ?b)`	`(overcome ?a ?b)`	`(pvxen ?a ?b)`
`(clear ?b)`	`(province ?b)`	`(dwtgh ?b)`
`(arm-empty)`	`(harmony)`	`(rkfsa)`

Same search space, same solutions, no semantic hints.

A planner doesn't care. An LLM does.

[1] Valmeekam et al. (2023). PlanBench. NeurIPS 2023.

How did the best models do?

GPT-4, average success rate across PlanBench domains?[1]

[1] Valmeekam et al. (2023). On the Planning Abilities of LLMs. NeurIPS 2023.

[2] Valmeekam, Stechly & Kambhampati (2024). LLMs Still Can't Plan; Can LRMs?

How did the best models do?

GPT-4, average success rate across PlanBench domains?[1]

$\sim 12\%$

[1] Valmeekam et al. (2023). On the Planning Abilities of LLMs. NeurIPS 2023.

[2] Valmeekam, Stechly & Kambhampati (2024). LLMs Still Can't Plan; Can LRMs?

How did the best models do?

GPT-4, average success rate across PlanBench domains?[1]

$\sim 12\%$

One year later, still on 3–5 block tasks:[2]

	Blocksworld	Mystery	Random
GPT-4o	35.5%	0.0%	—
Claude 3.5 Sonnet	54.8%	0.0%	—
Fast Downward	100.0%	100.0%	100.0%

(—: not reported)

[1] Valmeekam et al. (2023). On the Planning Abilities of LLMs. NeurIPS 2023.

[2] Valmeekam, Stechly & Kambhampati (2024). LLMs Still Can't Plan; Can LRMs?

How did the best models do?

GPT-4, average success rate across PlanBench domains?[1]

$\sim 12\%$

One year later, still on 3–5 block tasks:[2]

	Blocksworld	Mystery	Random
GPT-4o	35.5%	0.0%	—
Claude 3.5 Sonnet	54.8%	0.0%	—
Fast Downward	100.0%	100.0%	100.0%

(—: not reported)

Without the familiar names, performance disappears:
the models relied on the words. Memorization? Semantic meaning?

[1] Valmeekam et al. (2023). On the Planning Abilities of LLMs. NeurIPS 2023.

[2] Valmeekam, Stechly & Kambhampati (2024). LLMs Still Can't Plan; Can LRMs?

If prompting fails: train on plans

- **Plansformer**: fine-tune CodeT5 on planner-generated plans[1]
  - ∼97% valid plans on Towers of Hanoi

[1] Pallagani et al. (2022). Plansformer.

[2] Rossetti et al. (ICAPS 2024). Learning General Policies for Planning through GPT Models.

[3] Bohnet et al. (2024). Exploring and Benchmarking the Planning Capabilities of LLMs.

If prompting fails: train on plans

- **Plansformer**: fine-tune CodeT5 on planner-generated plans[1]
  - ∼97% valid plans on Towers of Hanoi
- **PlanGPT**: train GPT-style models per domain[2]
  - strong on the training distribution

[1] Pallagani et al. (2022). Plansformer.

[2] Rossetti et al. (ICAPS 2024). Learning General Policies for Planning through GPT Models.

[3] Bohnet et al. (2024). Exploring and Benchmarking the Planning Capabilities of LLMs.

If prompting fails: train on plans

- **Plansformer**: fine-tune CodeT5 on planner-generated plans[1]
  - ∼97% valid plans on Towers of Hanoi
- **PlanGPT**: train GPT-style models per domain[2]
  - strong on the training distribution
- Fine-tune large LLMs on optimal plans: helps in distribution[3]

[1] Pallagani et al. (2022). Plansformer.

[2] Rossetti et al. (ICAPS 2024). Learning General Policies for Planning through GPT Models.

[3] Bohnet et al. (2024). Exploring and Benchmarking the Planning Capabilities of LLMs.

If prompting fails: train on plans

- **Plansformer**: fine-tune CodeT5 on planner-generated plans[1]
  - ∼97% valid plans on Towers of Hanoi
- **PlanGPT**: train GPT-style models per domain[2]
  - strong on the training distribution
- Fine-tune large LLMs on optimal plans: helps in distribution[3]

The limitation

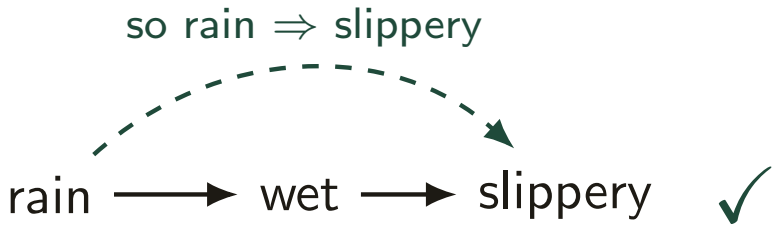
Slightly larger or out-of-distribution tasks break validity. Imitation reproduces its training data; planning requires generalization.

[1] Pallagani et al. (2022). Plansformer.

[2] Rossetti et al. (ICAPS 2024). Learning General Policies for Planning through GPT Models.

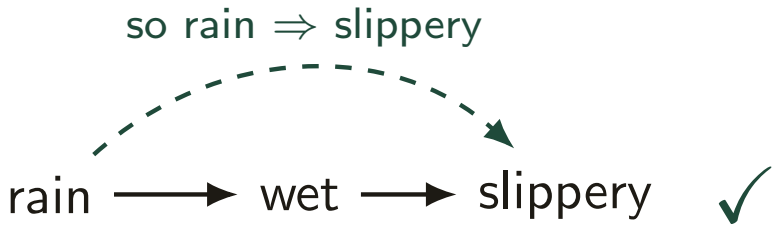
[3] Bohnet et al. (2024). Exploring and Benchmarking the Planning Capabilities of LLMs.

Abbe et al.: A theoretical barrier



[1] Abbe, Bengio, Lotfi, Sandon & Saremi (2024). How Far Can Transformers Reason? The Globality Barrier and Inductive Scratchpad. NeurIPS 2024.

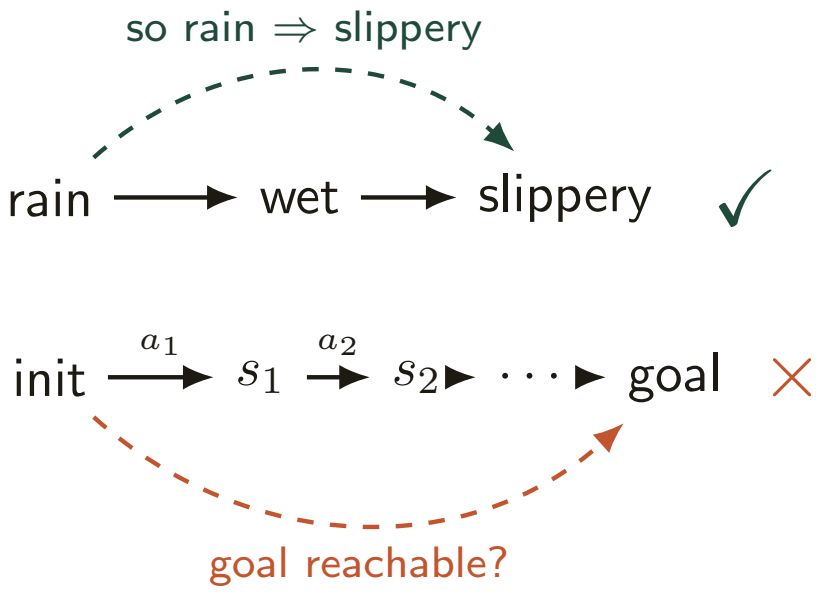
Abbe et al.: A theoretical barrier



- Result: globability barrier ⇒ not easy to learn [1]

[1] Abbe, Bengio, Lotfi, Sandon & Saremi (2024). How Far Can Transformers Reason? The Globality Barrier and Inductive Scratchpad. NeurIPS 2024.

Abbe et al.: A theoretical barrier



- Result: globability barrier ⇒ not easy to learn [1]

[1] Abbe, Bengio, Lotfi, Sandon & Saremi (2024). How Far Can Transformers Reason? The Globality Barrier and Inductive Scratchpad. NeurIPS 2024.

The skeptics' case

Early reading:

there is no basis to assume an LLM can reliably create, critique, and correct its own plans.[1]

[1] Kambhampati (2024). Can Large Language Models Reason and Plan? Annals of the New York Academy of Sciences.

[2] Stechly et al. (NeurIPS 2024). Chain of Thoughtlessness? An Analysis of CoT in Planning.

The skeptics' case

Early reading:

there is no basis to assume an LLM can reliably create, critique, and correct its own plans.[1]

- Prompting: ∼12%, and ∼0% without familiar names
- Chain-of-thought prompting did help initially[2]
- Fine-tuning: works only in distribution

[1] Kambhampati (2024). Can Large Language Models Reason and Plan? Annals of the New York Academy of Sciences.

[2] Stechly et al. (NeurIPS 2024). Chain of Thoughtlessness? An Analysis of CoT in Planning.

The skeptics' case

Early reading:

there is no basis to assume an LLM can reliably create, critique, and correct its own plans.[1]

- Prompting: ∼12%, and ∼0% without familiar names
- Chain-of-thought prompting did help initially[2]
- Fine-tuning: works only in distribution

Proposed alternative: keep the LLM as an idea generator,
put symbolic tools in the loop ⇒ Michael Katz's part.

---

[1] Kambhampati (2024). Can Large Language Models Reason and Plan? Annals of the New York Academy of Sciences.

[2] Stechly et al. (NeurIPS 2024). Chain of Thoughtlessness? An Analysis of CoT in Planning.

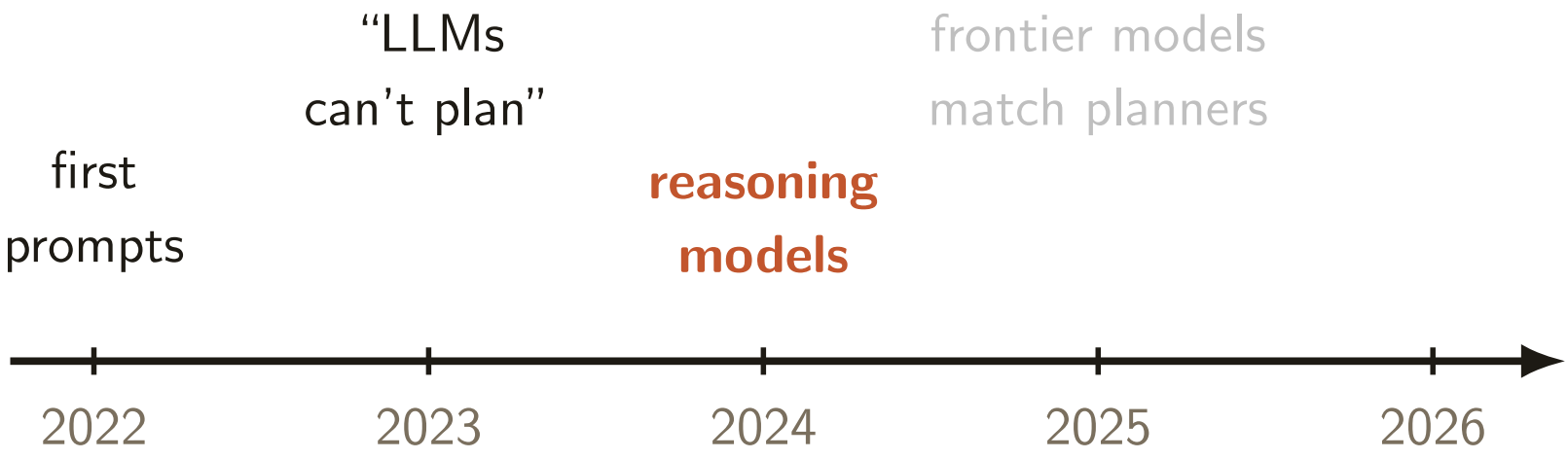
End of 2024, the consensus:

LLMs can't plan.

End of 2024, the consensus:

LLMs can't plan.

And the evidence supported it.


first prompts
"LLMs can't plan"
reasoning models
frontier models match planners
2022
2023
2024
2025
2026

Chain-of-thought

"The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?" [1]

[1] Wei et al. (NeurIPS 2022). Chain-of-Thought Prompting Elicits Reasoning in LLMs. Example from their Figure 1, answers abbreviated.

Chain-of-thought

"The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?" [1]

Direct answer:

`The answer is 27.` ×

[1] Wei et al. (NeurIPS 2022). Chain-of-Thought Prompting Elicits Reasoning in LLMs. Example from their Figure 1, answers abbreviated.

Chain-of-thought

"The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?"[1]

Direct answer:

```
The answer is 27.    ×
```

With a chain of thought:

```
23 − 20 = 3.
3 + 6 = 9.
The answer is 9.    ✓
```

[1] Wei et al. (NeurIPS 2022). Chain-of-Thought Prompting Elicits Reasoning in LLMs. Example from their Figure 1, answers abbreviated.

Chain-of-thought

"The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?"[1]

Direct answer:

`The answer is 27.` ×

With a chain of thought:

```
23 − 20 = 3.
3 + 6 = 9.
The answer is 9.
```
✓

Each token costs the same compute. Intermediate tokens buy more computation per answer, and a place to track state.

[1] Wei et al. (NeurIPS 2022). Chain-of-Thought Prompting Elicits Reasoning in LLMs. Example from their Figure 1, answers abbreviated.

A place to track state

```
(unstack b3 b1)
;; holding b3, b1 clear, ...
(put-down b3)
;; b3 on table, arm empty, ...
(pick-up b2)
;; ...
```

[1] Stechly et al. (NeurIPS 2024).

A place to track state

```
(unstack b3 b1)
;; holding b3, b1 clear, ...
(put-down b3)
;; b3 on table, arm empty, ...
(pick-up b2)
;; ...
```

- Exactly what plan generation needs: the world state after each action, written down where the model can see it

[1] Stechly et al. (NeurIPS 2024).

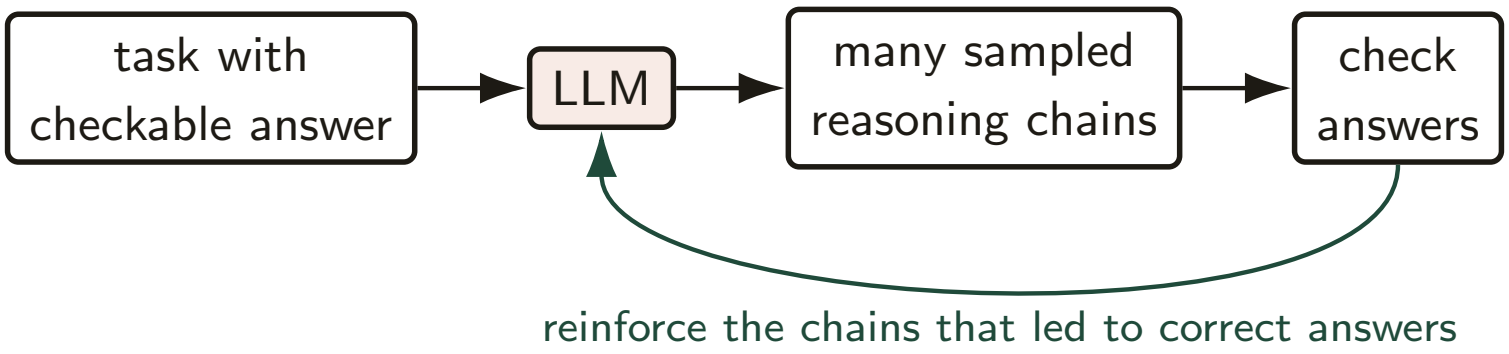
A place to track state

```
(unstack b3 b1)
;; holding b3, b1 clear, ...
(put-down b3)
;; b3 on table, arm empty, ...
(pick-up b2)
;; ...
```

- Exactly what plan generation needs: the world state after each action, written down where the model can see it
- But prompting for it barely moved planning performance[1]
  - the chains must come from somewhere better than imitation

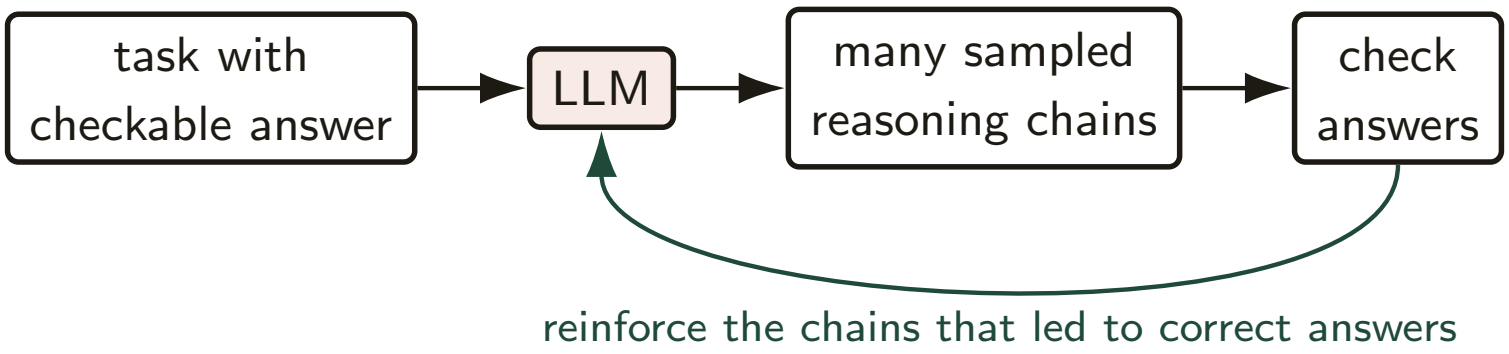
[1] Stechly et al. (NeurIPS 2024).

From prompting to training: reasoning models



[1] DeepSeek-AI (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via RL.

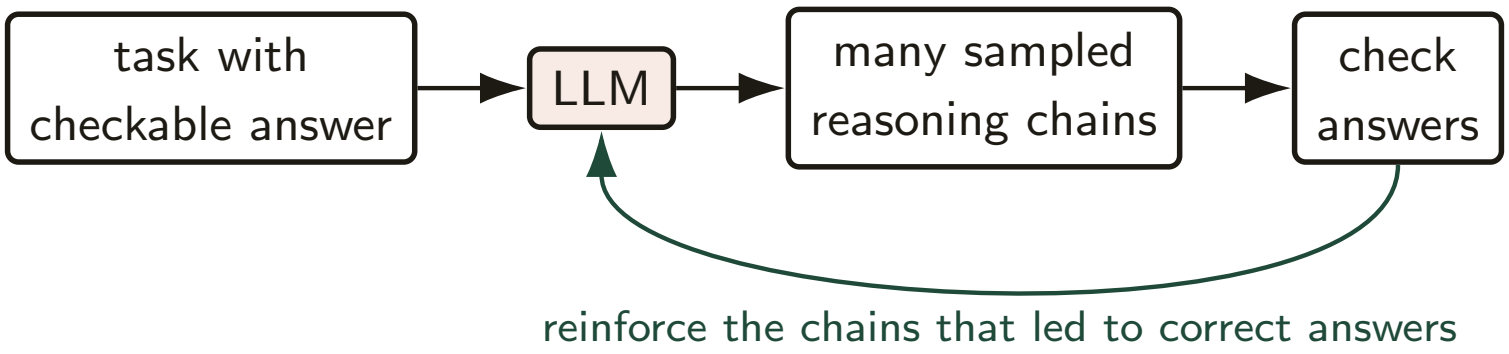
From prompting to training: reasoning models



- OpenAI o1 (2024): closed, but the behavior was new
  - long "thinking" before answering

[1] DeepSeek-AI (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via RL.

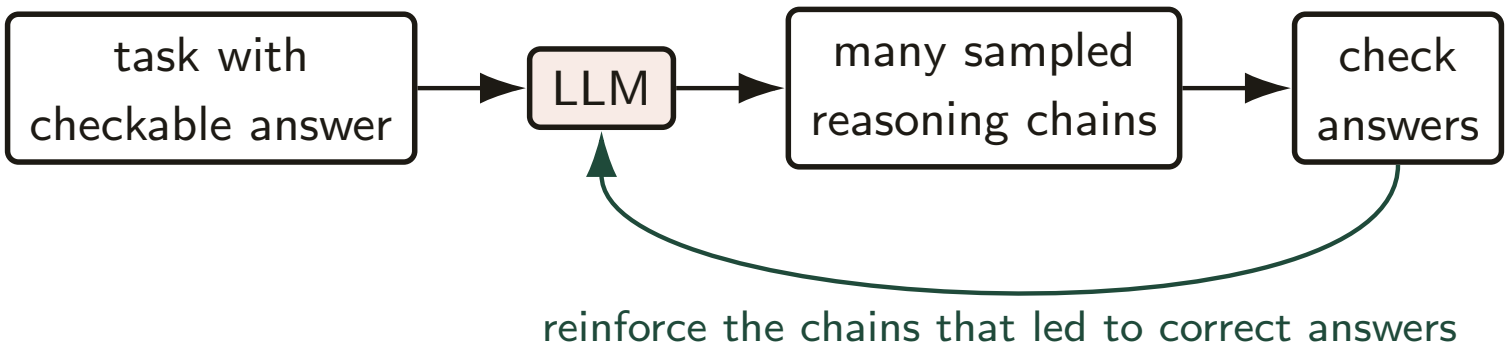
From prompting to training: reasoning models



- OpenAI o1 (2024): closed, but the behavior was new
  - long "thinking" before answering
- DeepSeek-R1 (2025): open weights, open "recipe"[1]
  - RL against verifiable rewards makes long reasoning *emerge*

[1] DeepSeek-AI (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via RL.

From prompting to training: reasoning models



- OpenAI o1 (2024): closed, but the behavior was new
  - long "thinking" before answering
- DeepSeek-R1 (2025): open weights, open "recipe"[1]
  - RL against verifiable rewards makes long reasoning *emerge*
- Thinking longer at inference buys accuracy: test-time scaling

[1] DeepSeek-AI (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via RL.

A second axis of compute

- How to improve the models?
- Training:
  more data, more parameters

[a] Snell, Lee, Xu, Kumar (2024). Scaling LLM Test-Time Compute Optimally... arXiv:2408.03314.

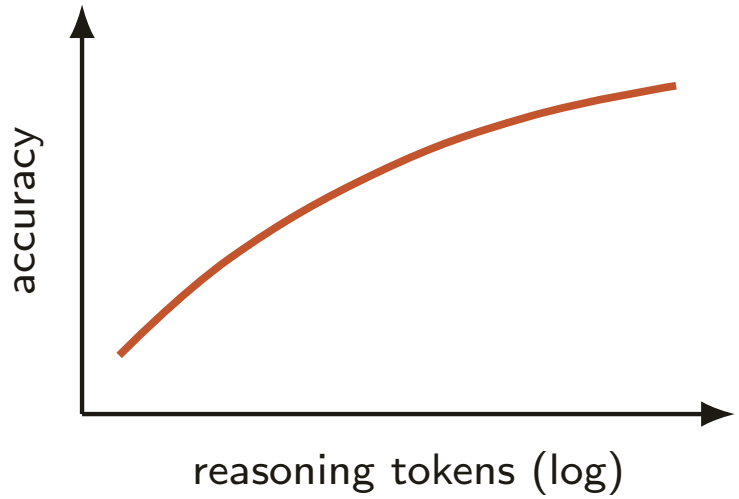
A second axis of compute

- How to improve the models?
- Training:
  more data, more parameters
- Inference:
  sample a longer trace before answering

[a] Snell, Lee, Xu, Kumar (2024). Scaling LLM Test-Time Compute Optimally... arXiv:2408.03314.

A second axis of compute

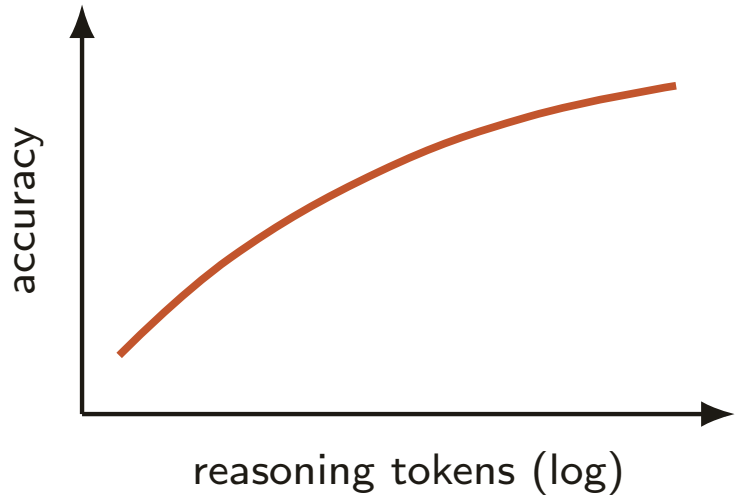
- How to improve the models?
- Training:
  more data, more parameters
- Inference:
  sample a longer trace before answering
- Accuracy rises with how long the model thinks



[a] Snell, Lee, Xu, Kumar (2024). Scaling LLM Test-Time Compute Optimally... arXiv:2408.03314.

A second axis of compute

- How to improve the models?
- Training:
  more data, more parameters
- Inference:
  sample a longer trace before answering
- Accuracy rises with how long the model thinks
- Spent well, the extra thinking can match a 14× larger model[a]



[a] Snell, Lee, Xu, Kumar (2024). Scaling LLM Test-Time Compute Optimally... arXiv:2408.03314.

Reasoning models meet PlanBench

o1-preview on Blocksworld (GPT-4o: 35.5%).[1]

[1] Valmeekam, Stechly & Kambhampati (2024). Their one-week o1 evaluation bill: $1,897.55.

Reasoning models meet PlanBench

o1-preview on Blocksworld (GPT-4o: 35.5%).[1]

97.8%

[1] Valmeekam, Stechly & Kambhampati (2024). Their one-week o1 evaluation bill: $1,897.55.

Reasoning models meet PlanBench

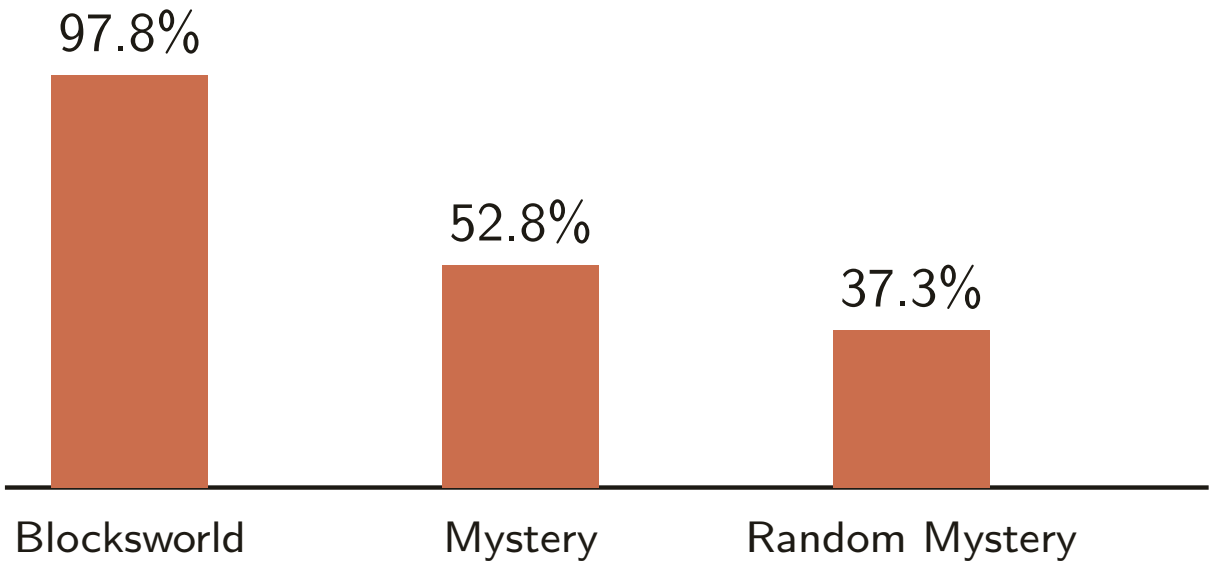
o1-preview on Blocksworld (GPT-4o: 35.5%).[1]

97.8%

Mystery Blocksworld		52.8%
Random Mystery Blocksworld		37.3%
Larger tasks (6–20 blocks)		23.6%
Cost per 100 tasks	$42.12	(GPT-4o: $0.02)

[1] Valmeekam, Stechly & Kambhampati (2024). Their one-week o1 evaluation bill: $1,897.55.

Reasoning, or retrieval?



- Only names were replaced by meaningless symbols[1]

[1] Numbers: Valmeekam, Stechly & Kambhampati (2024), o1-preview on PlanBench.

Chasing progress, not perfection

- Revisit end-to-end plan generation *with training in mind*[1]

[1] Huang, Cohn & Lipovetzky (ICAPS 2025); quotes from the arXiv version, arXiv:2412.10675.

Chasing progress, not perfection

- Revisit end-to-end plan generation *with training in mind*[1]
- "Merely fine-tuning LLMs on a corpus of planning instances does not lead to robust planning skills"

[1] Huang, Cohn & Lipovetzky (ICAPS 2025); quotes from the arXiv version, arXiv:2412.10675.

Chasing progress, not perfection

- Revisit end-to-end plan generation *with training in mind*[1]
- "Merely fine-tuning LLMs on a corpus of planning instances does not lead to robust planning skills"
- Supervised fine-tuning saturates the training distribution

[1] Huang, Cohn & Lipovetzky (ICAPS 2025); quotes from the arXiv version, arXiv:2412.10675.

Chasing progress, not perfection

- Revisit end-to-end plan generation *with training in mind*[1]
- "Merely fine-tuning LLMs on a corpus of planning instances does not lead to robust planning skills"
- Supervised fine-tuning saturates the training distribution

A different question

Not *"can we prompt it?"* but *"which training signal teaches planning?"*

[1] Huang, Cohn & Lipovetzky (ICAPS 2025); quotes from the arXiv version, arXiv:2412.10675.

A reward with partial credit

- Plans are checkable, so the obvious reward is binary:
  - $+1$ when the plan is valid
  - 0 otherwise

[1] Reward: $R = 1$ if P_g is valid, else $|\mathrm{LCCS}(P_g, P_r)|/|P_r|$. Huang, Cohn & Lipovetzky (2025).

A reward with partial credit

- Plans are checkable, so the obvious reward is binary:
  - $+1$ when the plan is valid
  - 0 otherwise
- Initially almost every sampled plan is invalid
  - reward is 0 nearly everywhere

[1] Reward: $R = 1$ if P_g is valid, else $|\mathrm{LCCS}(P_g, P_r)|/|P_r|$. Huang, Cohn & Lipovetzky (2025).

A reward with partial credit

- Plans are checkable, so the obvious reward is binary:
  - $+1$ when the plan is valid
  - 0 otherwise
- Initially almost every sampled plan is invalid
  - reward is 0 nearly everywhere

reference:	`(unstack b1 b2)`	`(put-down b1)`	`(pick-up b3)`	`(stack b3 b2)`
generated:	`(unstack b1 b2)`	`(put-down b1)`	`(pick-up b2)`	`(stack b2 b3)`

[1] Reward: $R = 1$ if P_g is valid, else $|\mathrm{LCCS}(P_g, P_r)|/|P_r|$. Huang, Cohn & Lipovetzky (2025).

A reward with partial credit

- Plans are checkable, so the obvious reward is binary:
  - $+1$ when the plan is valid
  - 0 otherwise
- Initially almost every sampled plan is invalid
  - reward is 0 nearly everywhere

reference:	`(unstack b1 b2)`	`(put-down b1)`	`(pick-up b3)`	`(stack b3 b2)`
generated:	`(unstack b1 b2)`	`(put-down b1)`	`(pick-up b2)`	`(stack b2 b3)`

longest contiguous match: 2 of 4 actions $\Rightarrow$ reward

[1] Reward: $R = 1$ if P_g is valid, else $|\mathrm{LCCS}(P_g, P_r)|/|P_r|$. Huang, Cohn & Lipovetzky (2025).

A reward with partial credit

- Plans are checkable, so the obvious reward is binary:
  - $+1$ when the plan is valid
  - 0 otherwise
- Initially almost every sampled plan is invalid
  - reward is 0 nearly everywhere

reference:	`(unstack b1 b2)`	`(put-down b1)`	`(pick-up b3)`	`(stack b3 b2)`
generated:	`(unstack b1 b2)`	`(put-down b1)`	`(pick-up b2)`	`(stack b2 b3)`

longest contiguous match: 2 of 4 actions $\Rightarrow$ reward

A near-miss now outscores nonsense, so RL has a gradient to follow[1]

[1] Reward: $R = 1$ if P_g is valid, else $|\mathrm{LCCS}(P_g, P_r)|/|P_r|$. Huang, Cohn & Lipovetzky (2025).

What the signal buys

	fine-tuned	+ RL (progress)
longer tasks, valid	34.8%	41.5%
unseen domains, valid	0%	12.5%

[1] Huang, Cohn & Lipovetzky (2025), arXiv:2412.10675; long and unseen-domain splits.

What the signal buys

	fine-tuned	+ RL (progress)
longer tasks, valid	34.8%	41.5%
unseen domains, valid	0%	12.5%

- But valid plans on unseen domains stay the wall, at 12.5%[1]

[1] Huang, Cohn & Lipovetzky (2025), arXiv:2412.10675; long and unseen-domain splits.

The data is already there

If you are OpenAI, Anthropic, Google, DeepSeek, etc.,

do you need to post-train your models on planning?

The data is already there

If you are OpenAI, Anthropic, Google, DeepSeek, etc.,
do you need to post-train your models on planning?

- In practice, providers have lots and lots of user data
- They turn it into a training signal: implicit RL
  - kept chats, accepted answers, thumbs-up
- Cheap, and no synthetic data to manufacture
  - the prompts come from the users themselves

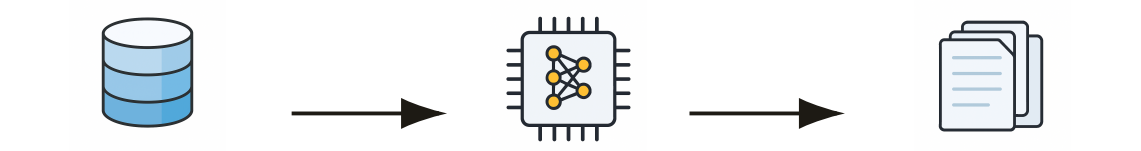
The data is already there

If you are OpenAI, Anthropic, Google, DeepSeek, etc.,
do you need to post-train your models on planning?

- In practice, providers have lots and lots of user data
- They turn it into a training signal: implicit RL
  - kept chats, accepted answers, thumbs-up
- Cheap, and no synthetic data to manufacture
  - the prompts come from the users themselves

Does this help planning?

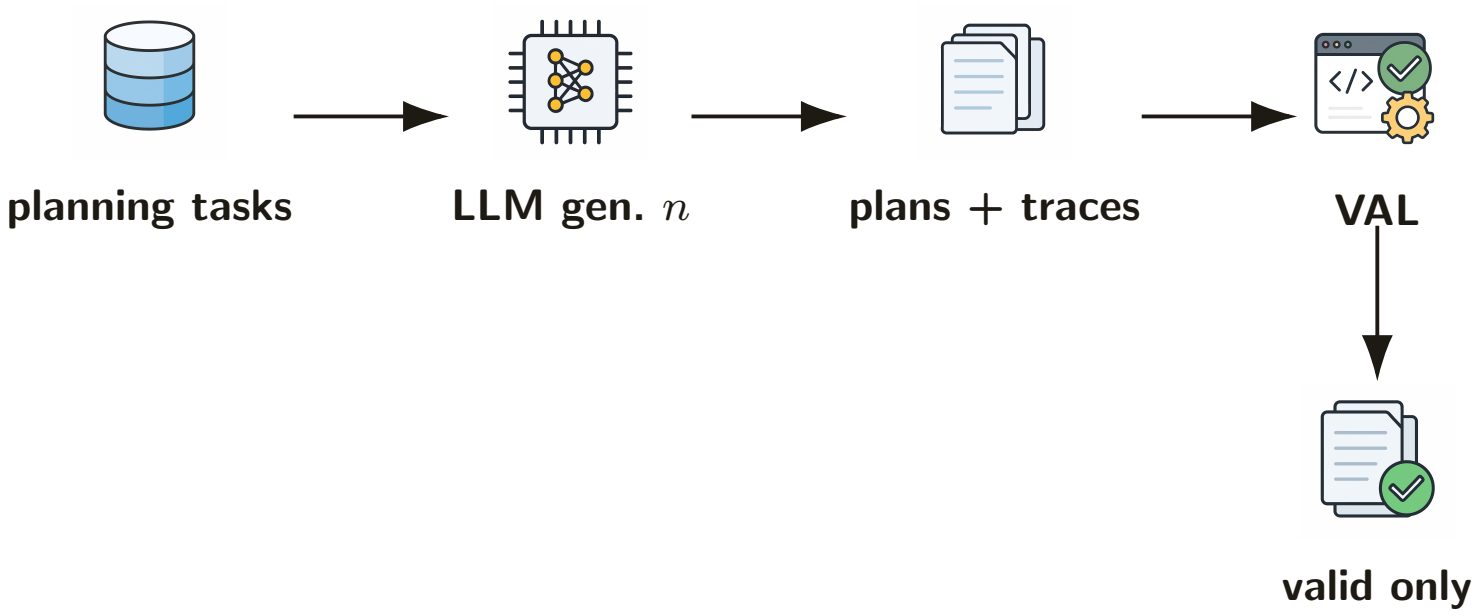
Iterative deployment



[1] C., Gelberg, Melo, Shumailov, Pereira & Gal (2025). Iterative Deployment Improves Planning Skills in LLMs. arXiv:2512.24940.

[2] Shumailov, Shumaylov, Zhao, Papernot, Anderson & Gal (2024). AI Models Collapse When Trained on Recursively Generated Data. Nature.

Iterative deployment

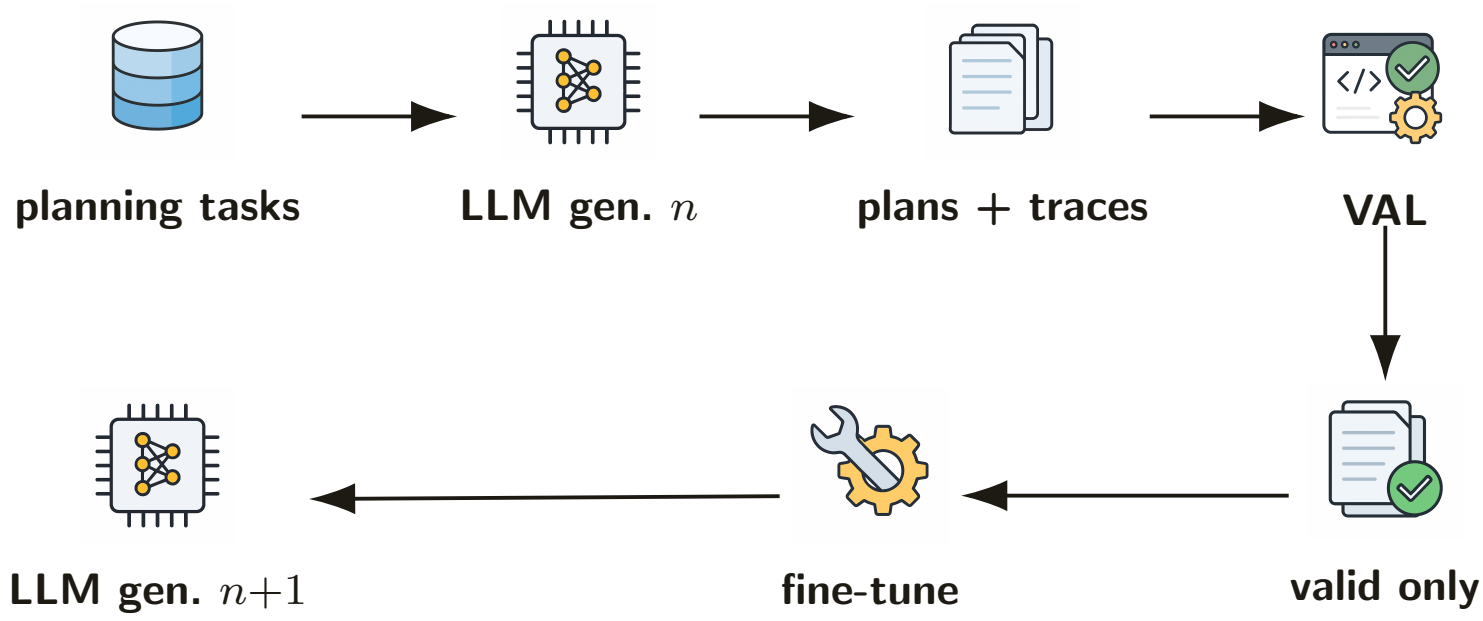


- Keep only traces whose plan is valid[1]
  - this filter is what prevents collapse[2]

[1] C., Gelberg, Melo, Shumailov, Pereira & Gal (2025). Iterative Deployment Improves Planning Skills in LLMs. arXiv:2512.24940.

[2] Shumailov, Shumaylov, Zhao, Papernot, Anderson & Gal (2024). AI Models Collapse When Trained on Recursively Generated Data. Nature.

Iterative deployment



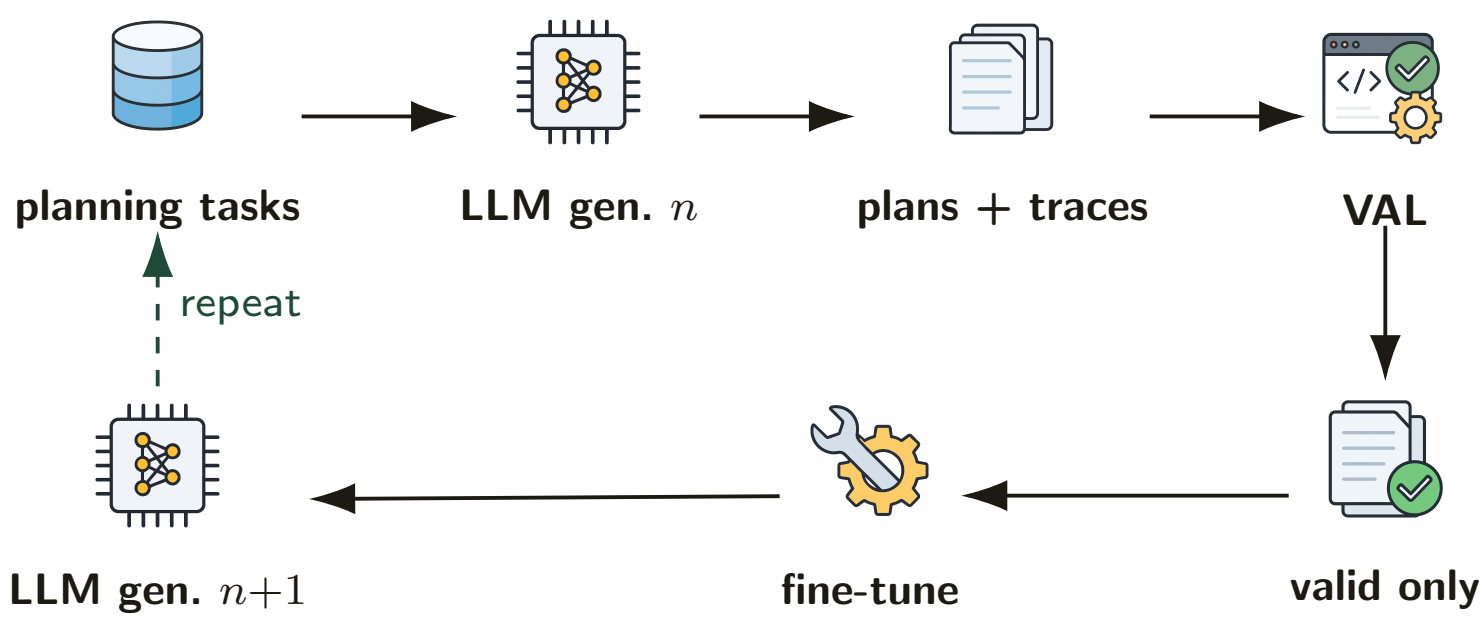
- Keep only traces whose plan is valid[1]
  - this filter is what prevents collapse[2]
- Fine-tune and deploy again

---

[1] C., Gelberg, Melo, Shumailov, Pereira & Gal (2025). Iterative Deployment Improves Planning Skills in LLMs. arXiv:2512.24940.

[2] Shumailov, Shumaylov, Zhao, Papernot, Anderson & Gal (2024). AI Models Collapse When Trained on Recursively Generated Data. Nature.

Iterative deployment



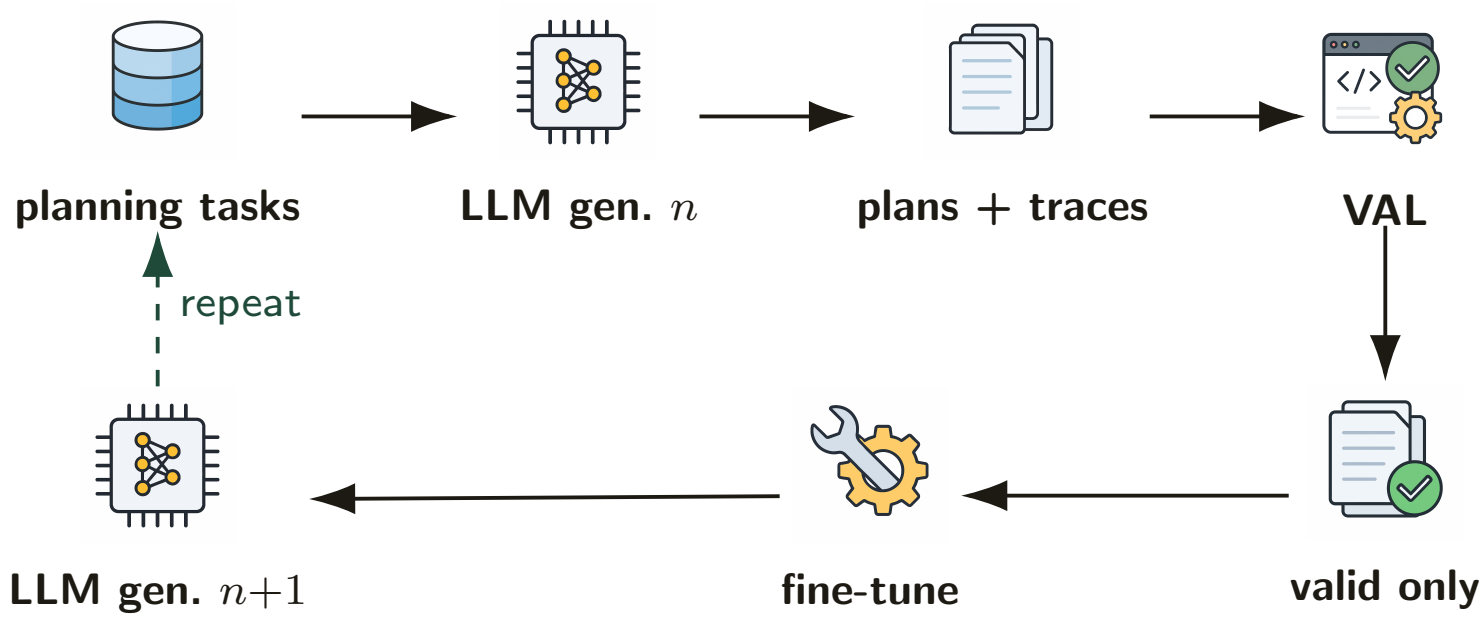
- Keep only traces whose plan is valid[1]
  - this filter is what prevents collapse[2]
- Fine-tune and deploy again

---

[1] C., Gelberg, Melo, Shumailov, Pereira & Gal (2025). Iterative Deployment Improves Planning Skills in LLMs. arXiv:2512.24940.

[2] Shumailov, Shumaylov, Zhao, Papernot, Anderson & Gal (2024). AI Models Collapse When Trained on Recursively Generated Data. Nature.

Iterative deployment



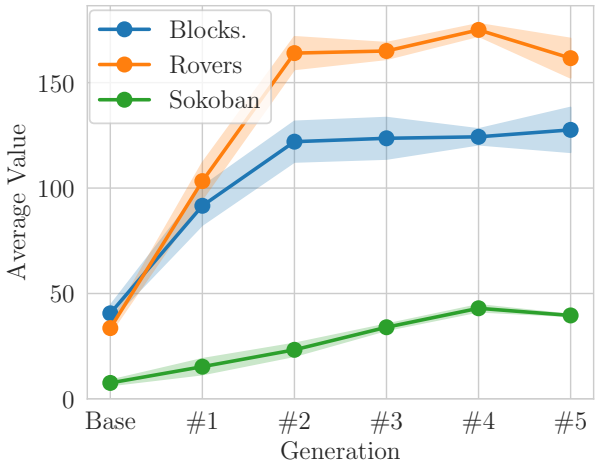
- Keep only traces whose plan is valid[1]
  - this filter is what prevents collapse[2]
- Fine-tune and deploy again
- No planner as a teacher. No human data. No supervision.

[1] C., Gelberg, Melo, Shumailov, Pereira & Gal (2025). Iterative Deployment Improves Planning Skills in LLMs. arXiv:2512.24940.

[2] Shumailov, Shumaylov, Zhao, Papernot, Anderson & Gal (2024). AI Models Collapse When Trained on Recursively Generated Data. Nature.

Iterative deployment improves planning

Number of solved tasks per generation.

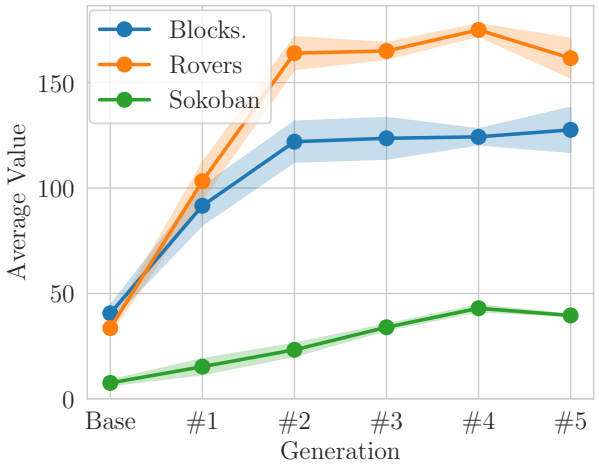


- Qwen3 4B, a model 100× smaller than DeepSeek R1
  - evaluated on 1 000 unseen, harder tasks per domain

[1] cf. STaR, Zelikman et al. (2022).

Iterative deployment improves planning

Number of solved tasks per generation.

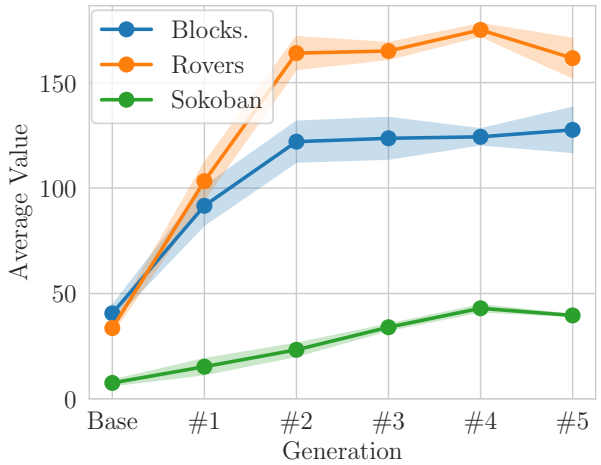


- Qwen3 4B, a model 100× smaller than DeepSeek R1
  - evaluated on 1 000 unseen, harder tasks per domain
- The model solves harder tasks requiring *longer* plans
  - it does not simply solve the same tasks more often

[1] cf. STaR, Zelikman et al. (2022).

Iterative deployment improves planning

Number of solved tasks per generation.



- Qwen3 4B, a model 100× smaller than DeepSeek R1
  - evaluated on 1 000 unseen, harder tasks per domain
- The model solves harder tasks requiring *longer* plans
  - it does not simply solve the same tasks more often
- Fine-tuning on validated successes is equivalent to REINFORCE with a binary reward[1]

[1] cf. STaR, Zelikman et al. (2022).

Back to the frontier

- The controlled loop improves planning, with no planner and no human data

Back to the frontier

- The controlled loop improves planning, with no planner and no human data
- Frontier models run the same loop at far larger scale
  - but with a noisier oracle: human approval, not VAL

Back to the frontier

- The controlled loop improves planning, with no planner and no human data
- Frontier models run the same loop at far larger scale
  - but with a noisier oracle: human approval, not VAL

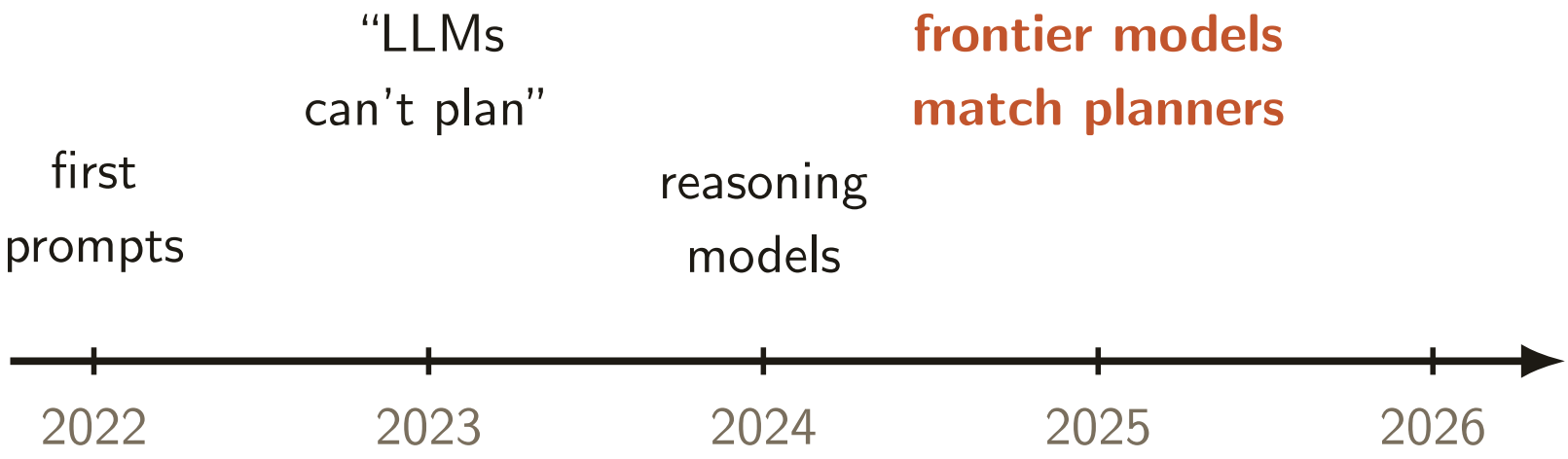
So planning might improve without planning-specific training data.

Back to the frontier

- The controlled loop improves planning, with no planner and no human data
- Frontier models run the same loop at far larger scale
  - but with a noisier oracle: human approval, not VAL

So planning might improve without planning-specific training data.

Each generation should plan better than the last. Does it?


first prompts
"LLMs can't plan"
reasoning models
frontier models match planners
2022
2023
2024
2025
2026

A harder benchmark

- 8 domains from the IPC 2023 Learning Track[1],
  newly generated tasks[2]
  - fresh instances, no contamination

[1] Taitler et al. (2024). The 2023 International Planning Competition. AI Magazine.

[2] C., Pereira & Seipp (2025). Frontier Large Language Models Rival State-of-the-Art Planners. arXiv:2511.09378.

A harder benchmark

- 8 domains from the IPC 2023 Learning Track[1], *newly generated* tasks[2]
  - fresh instances, no contamination
- Baseline planners from previous IPCs.

[1] Taitler et al. (2024). The 2023 International Planning Competition. AI Magazine.

[2] C., Pereira & Seipp (2025). Frontier Large Language Models Rival State-of-the-Art Planners. arXiv:2511.09378.

A harder benchmark

- 8 domains from the IPC 2023 Learning Track[1], *newly generated* tasks[2]
  - fresh instances, no contamination
- Baseline planners from previous IPCs.
- Every plan checked with VAL; planners get one CPU core, 8 GiB

[1] Taitler et al. (2024). The 2023 International Planning Competition. AI Magazine.

[2] C., Pereira & Seipp (2025). Frontier Large Language Models Rival State-of-the-Art Planners. arXiv:2511.09378.

A harder benchmark

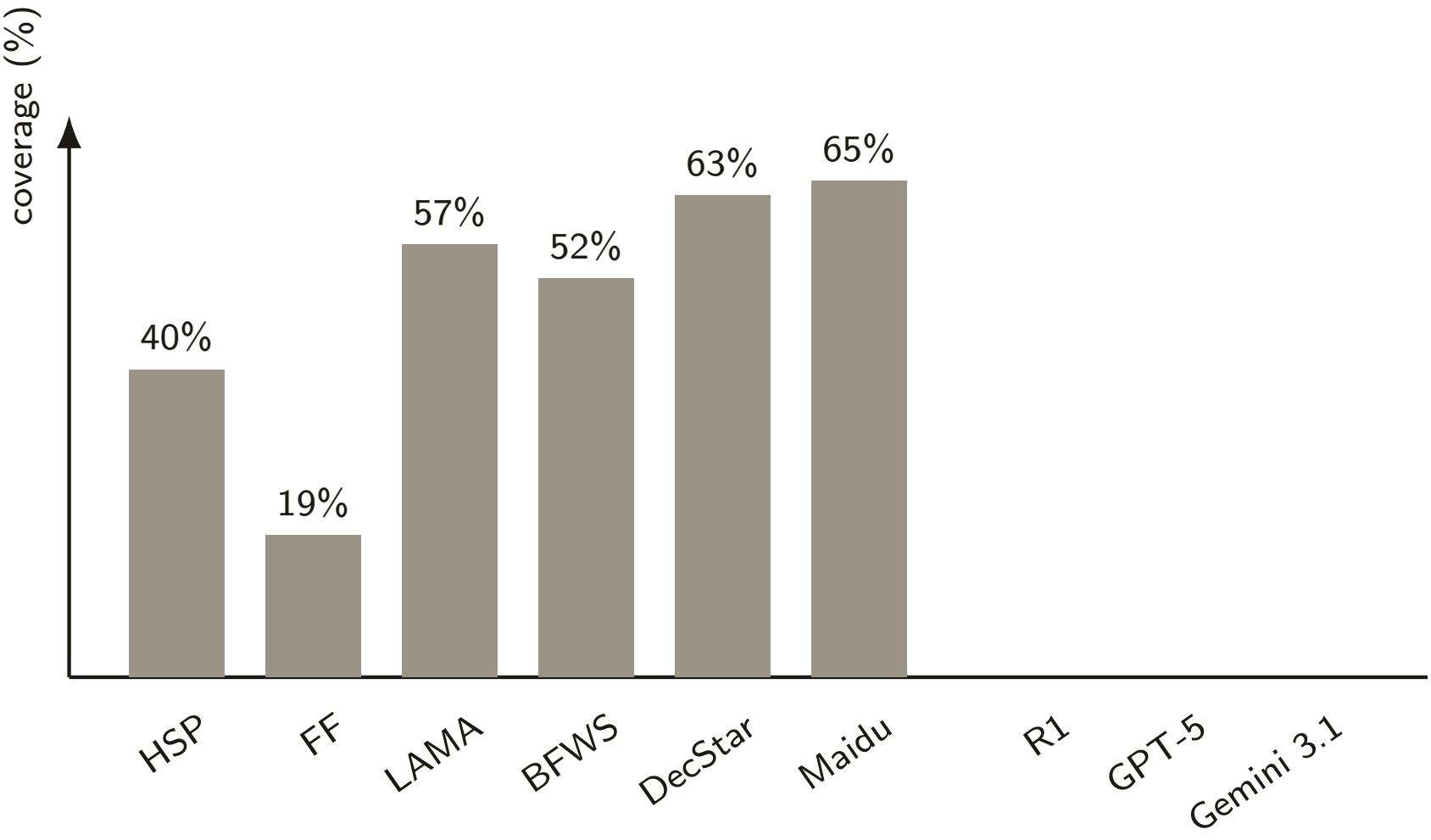
- 8 domains from the IPC 2023 Learning Track[1], *newly generated* tasks[2]
  - fresh instances, no contamination
- Baseline planners from previous IPCs.
- Every plan checked with VAL; planners get one CPU core, 8 GiB

	PlanBench	this benchmark
blocks	≤ 5	up to 477
plan length	≤ 16	$> 1\,000$ steps

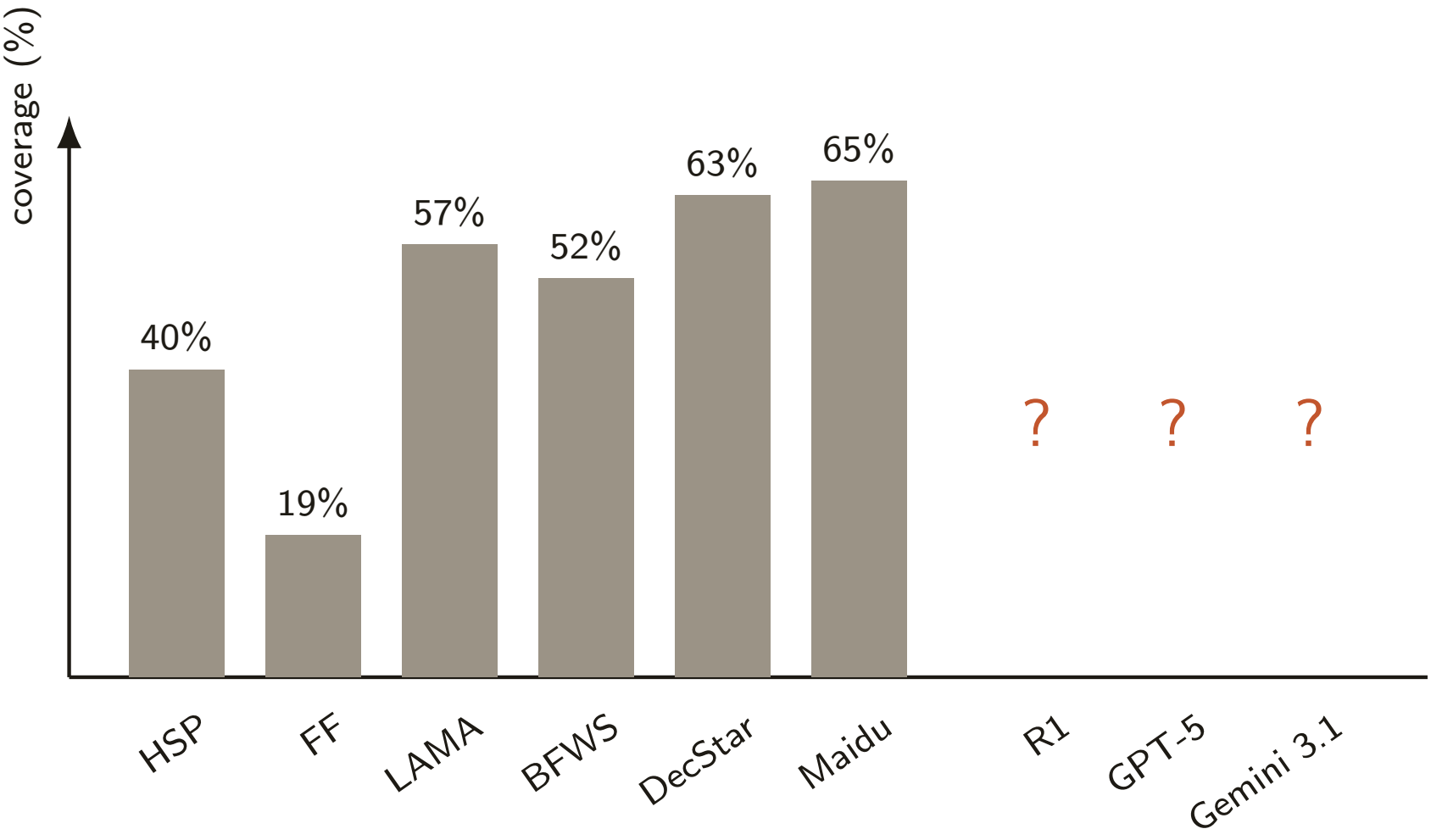
[1] Taitler et al. (2024). The 2023 International Planning Competition. AI Magazine.

[2] C., Pereira & Seipp (2025). Frontier Large Language Models Rival State-of-the-Art Planners. arXiv:2511.09378.

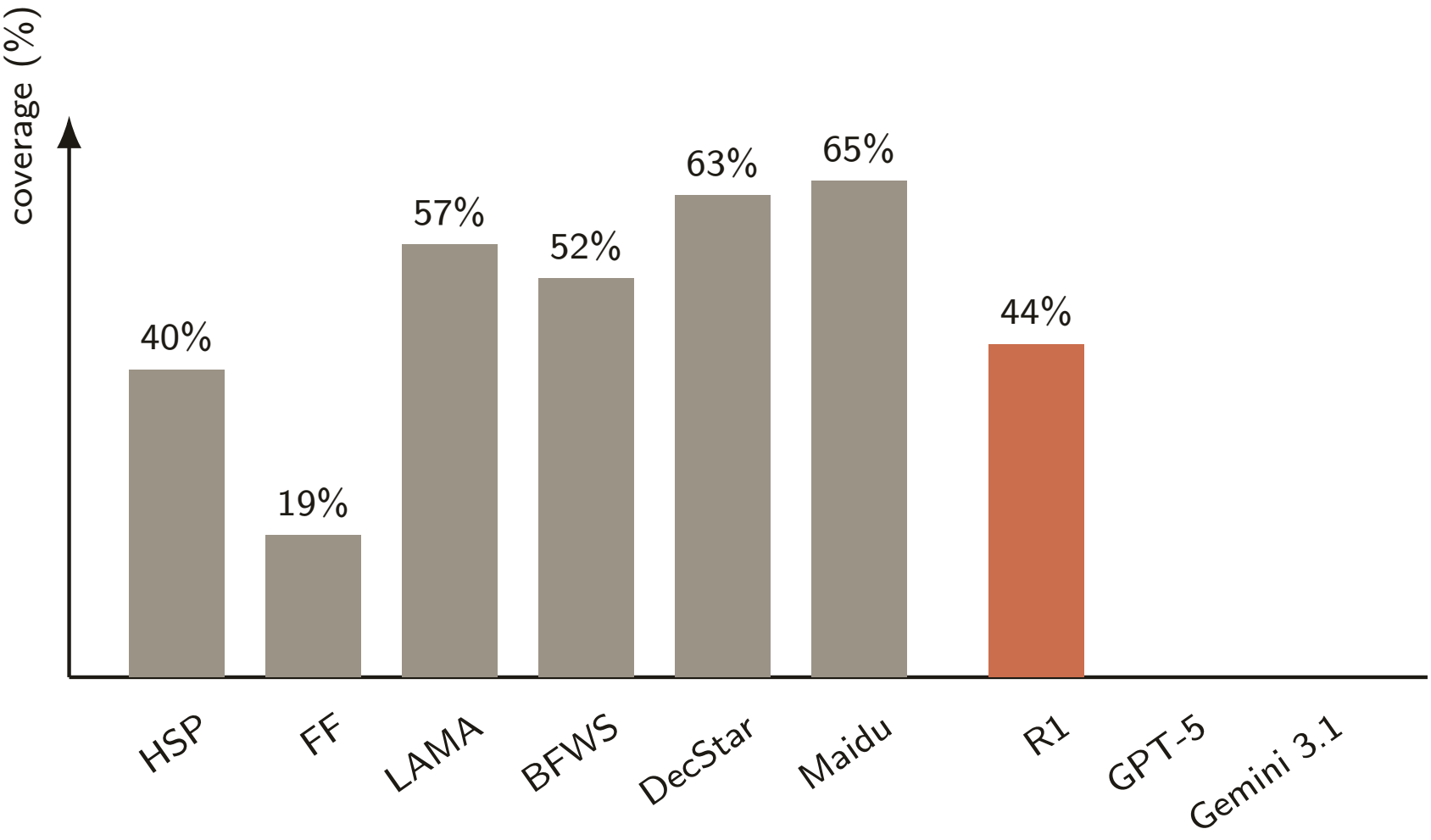
Coverage


coverage (%)
40%
19%
57%
52%
63%
65%
HSP
FF
LAMA
BFWS
DecStar
Maidu
R1
GPT-5
Gemini 3.1

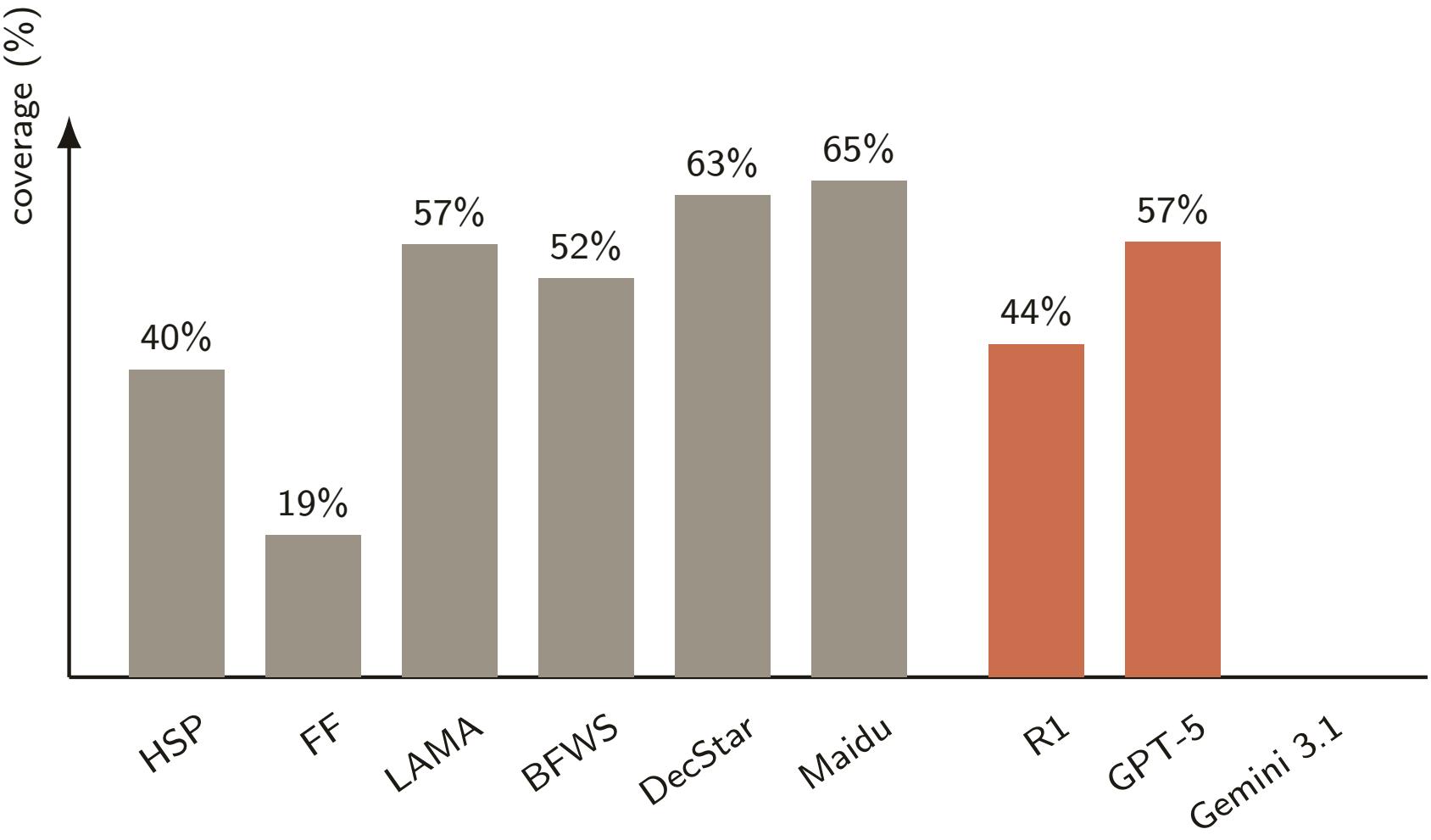
Coverage


coverage (%)
40%
19%
57%
52%
63%
65%
?
?
?
HSP
FF
LAMA
BFWS
DecStar
Maidu
R1
GPT-5
Gemini 3.1

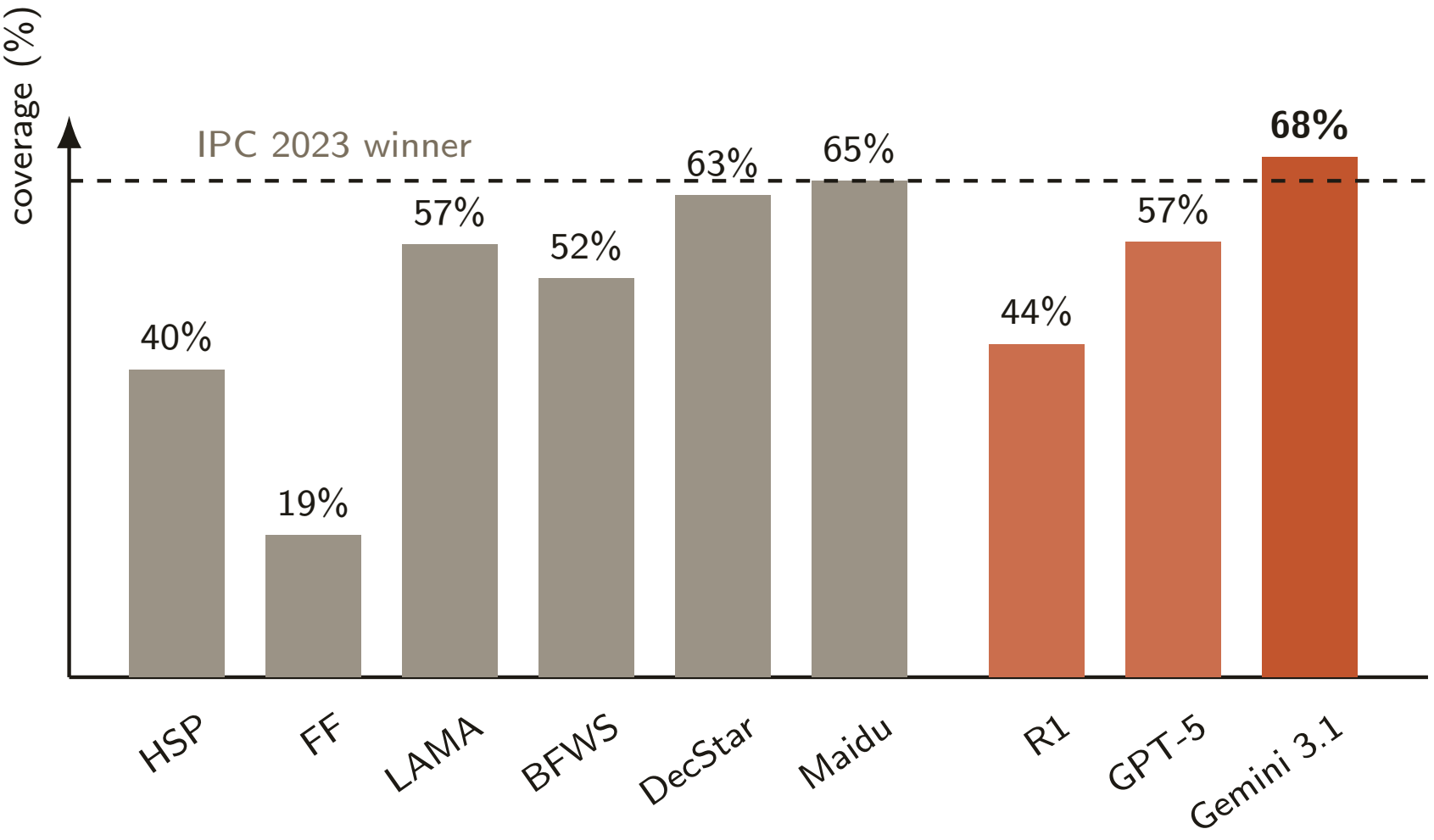
Coverage


coverage (%)
40%
19%
57%
52%
63%
65%
44%
HSP
FF
LAMA
BFWS
DecStar
Maidu
R1
GPT-5
Gemini 3.1

Coverage


coverage (%)
40%
19%
57%
52%
63%
65%
44%
57%
HSP
FF
LAMA
BFWS
DecStar
Maidu
R1
GPT-5
Gemini 3.1

Coverage



An LLM, writing a plan token by token, beats the IPC winner.

Same tasks; three model generations

GPT-3.5	GPT-4.1	GPT-5
0%		
Nov 2022		

Same tasks; three model generations

GPT-3.5		GPT-4.1	GPT-5
0%	→	6%	
Nov 2022		Apr 2025	

Same tasks; three model generations

GPT-3.5		GPT-4.1		GPT-5
0%	→	6%	→	57%
Nov 2022		Apr 2025		Aug 2025

Same tasks; three model generations

GPT-3.5		GPT-4.1		GPT-5
0%	→	6%	→	57%
Nov 2022		Apr 2025		Aug 2025

Gemini 2.5		Gemini 3.0		Gemini 3.1
45%	→	65%	→	68%
Mar 2025		Nov 2025		2026

Same tasks; three model generations

GPT-3.5		GPT-4.1		GPT-5
0%	→	6%	→	57%
Nov 2022		Apr 2025		Aug 2025

Gemini 2.5		Gemini 3.0		Gemini 3.1
45%	→	65%	→	68%
Mar 2025		Nov 2025		2026

The 2023–2024 critiques measured those models correctly.

The models improved.

“It just memorized Blocksworld”

```
(:action pick-up
 :parameters (?b)
 :precondition (and
  (clear ?b) (on-table ?b)
  (arm-empty))
 ...)
```

“It just memorized Blocksworld”

```
(:action pick-up
 :parameters (?b)
 :precondition (and
  (clear ?b) (on-table ?b)
  (arm-empty))
 ...)
```

```
(:action gxuralqhfigujlqy
 :parameters (?x)
 :precondition (and
  (jmklkrgqahidtiuy ?x)
  (hbzvungehlbssavm ?x)
  (xmulpmadgkjorkin))
 ...)
```

“It just memorized Blocksworld”

```
(:action pick-up
 :parameters (?b)
 :precondition (and
  (clear ?b) (on-table ?b)
  (arm-empty))
 ...)
```

```
(:action gxuralqhfigujlqy
 :parameters (?x)
 :precondition (and
  (jmklkrgqahidtiuy ?x)
  (hbzvungehlbssavm ?x)
  (xmulpmadgkjorkin))
 ...)
```

- What happens if we randomize the vocabulary again?

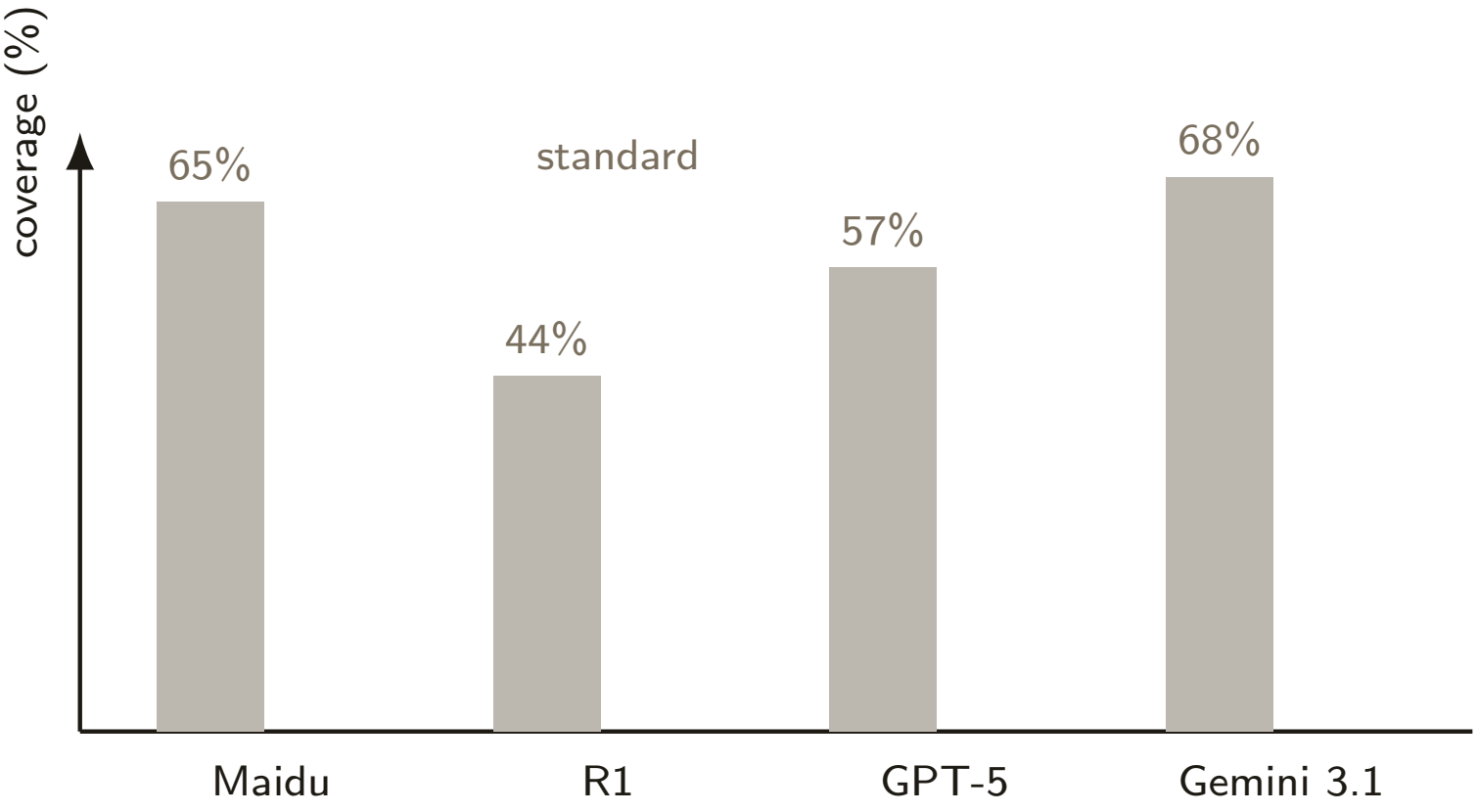
“It just memorized Blocksworld”

```
(:action pick-up
 :parameters (?b)
 :precondition (and
  (clear ?b) (on-table ?b)
  (arm-empty))
 ...)
```

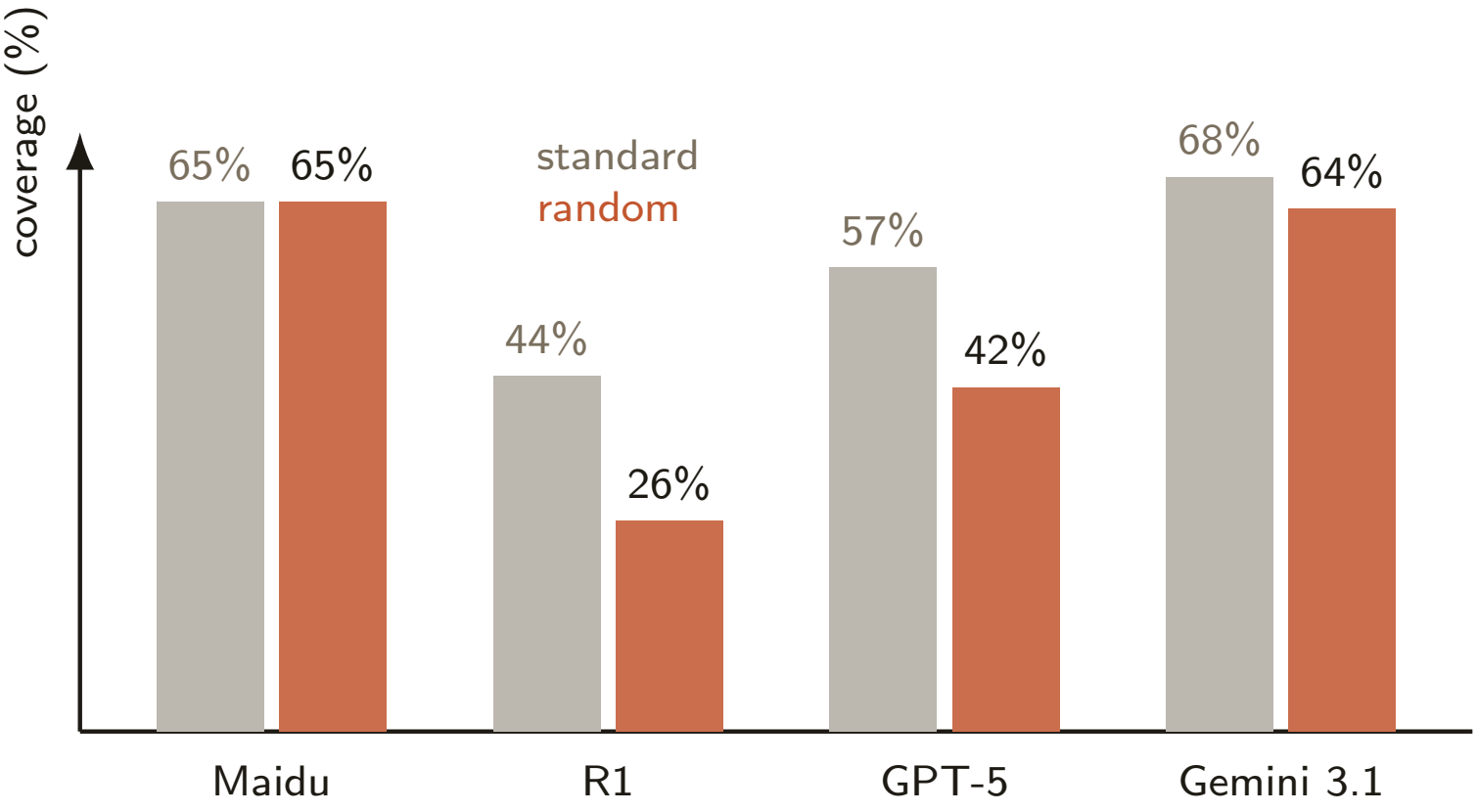
```
(:action gxuralqhfigujlqy
 :parameters (?x)
 :precondition (and
  (jmklkrgqahidtiuy ?x)
  (hbzvungehlbssavm ?x)
  (xmulpmadgkjorkin))
 ...)
```

- What happens if we randomize the vocabulary again?
- If LLMs still only pattern-match the words, performance should collapse again

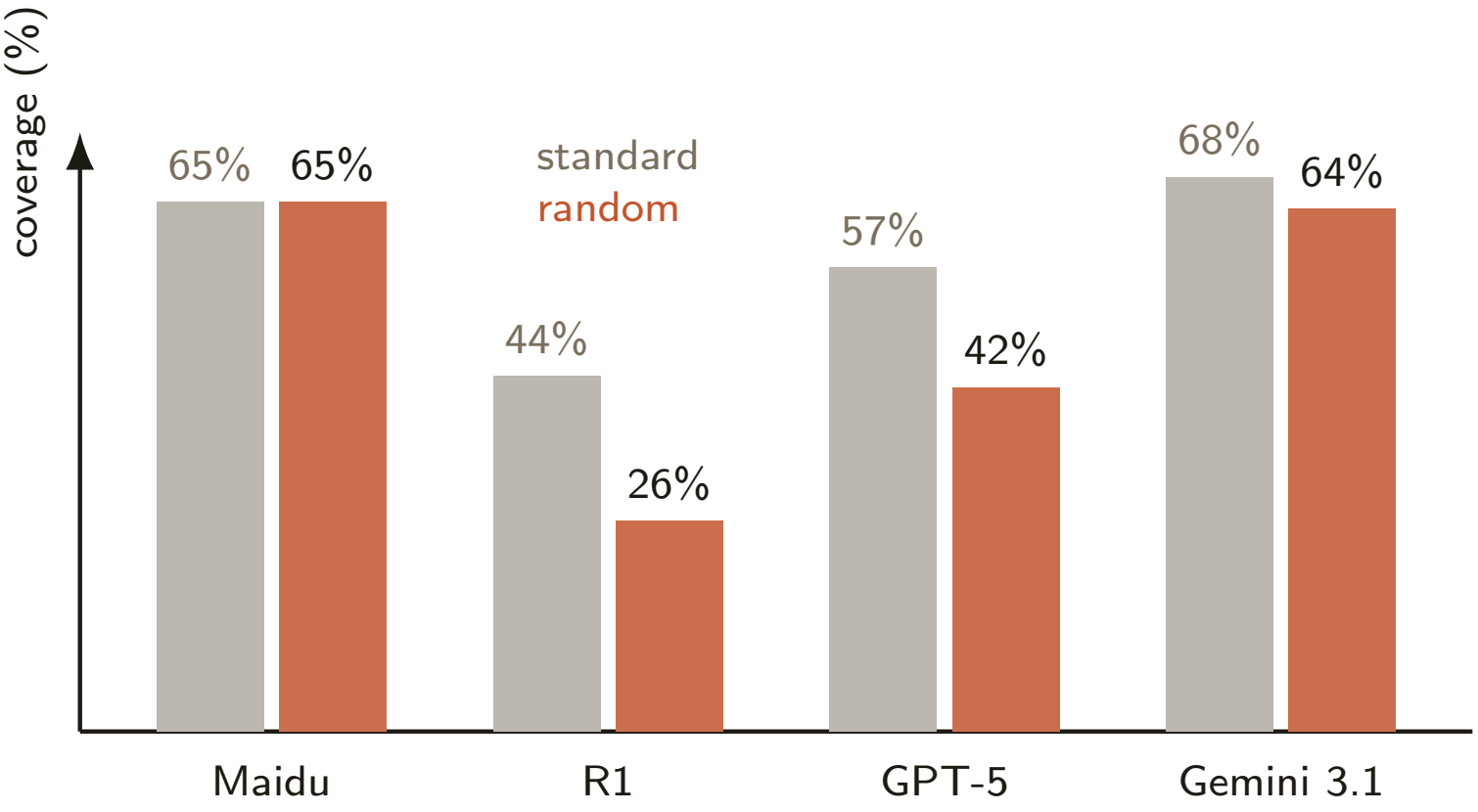
Results on obfuscated tasks



Results on obfuscated tasks

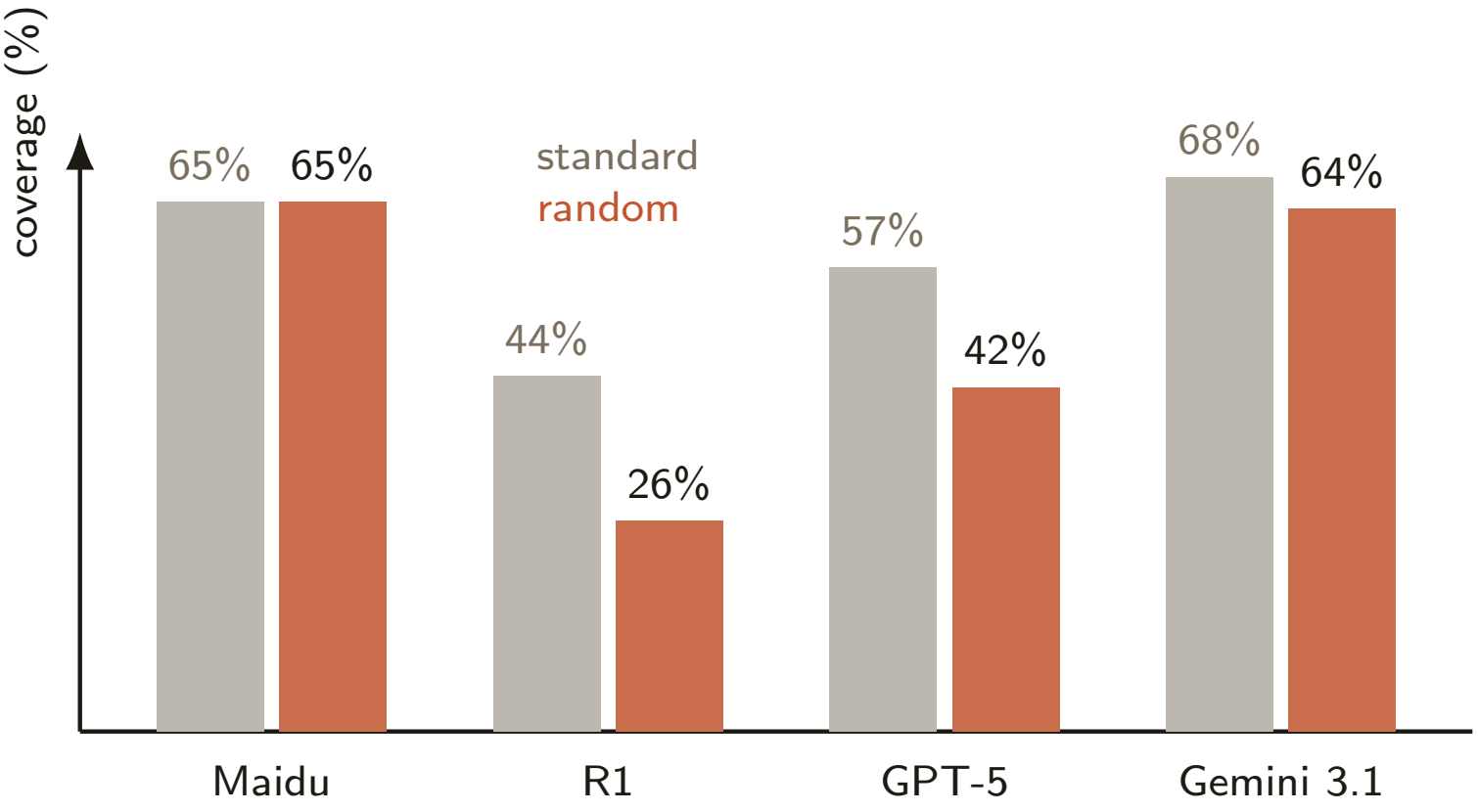


Results on obfuscated tasks



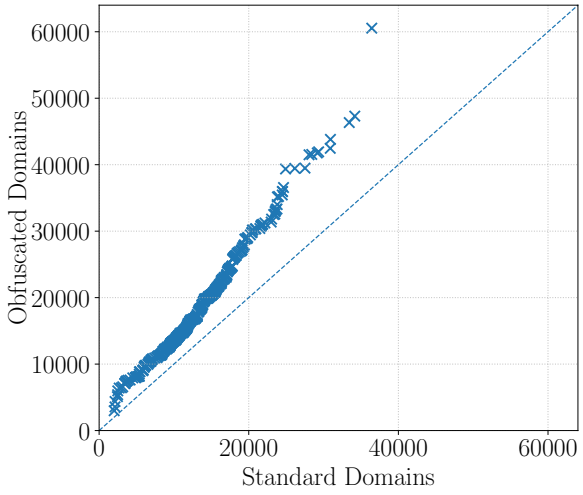
- Semantics do help: R1 drops 44 → 26%, GPT-5 drops 57 → 42%

Results on obfuscated tasks



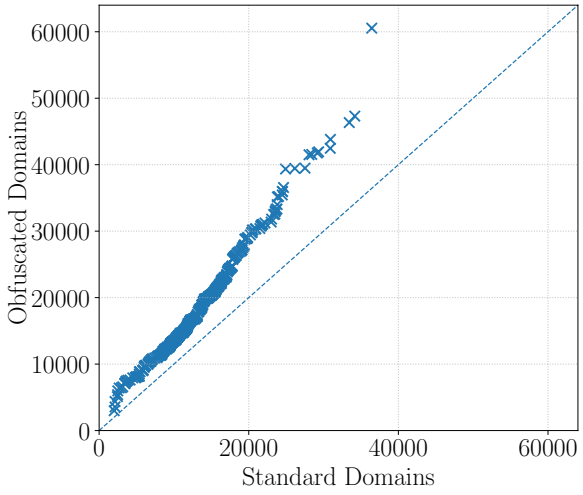
- Semantics do help: R1 drops 44 → 26%, GPT-5 drops 57 → 42%
- But Gemini 3.1 Pro drops only 68 → 64%, level with the IPC winner, on gibberish
  - it thinks much longer (reasoning tokens spike)

Obfuscation costs tokens



[1] Real tokenizations (GPT-4o tokenizer, `o200k_base`).

Obfuscation costs tokens

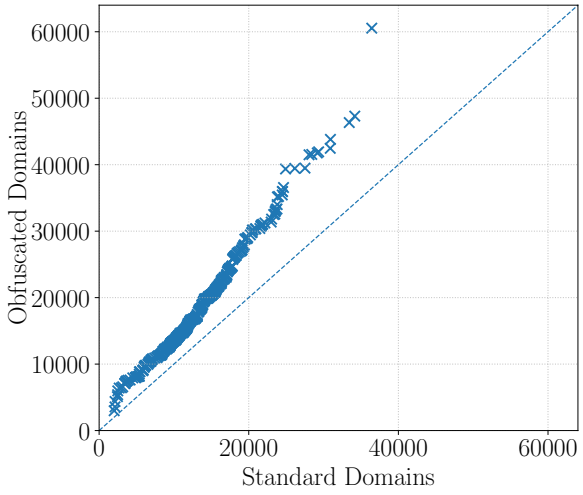


`(on -table ? b )` VS. `(j m kl kr g q ahid ti uy ? x )`

- Prior work uses 16 random characters per name: one predicate goes from 5 to 12 tokens, before any reasoning starts[1]

[1] Real tokenizations (GPT-4o tokenizer, `o200k_base`).

Obfuscation costs tokens



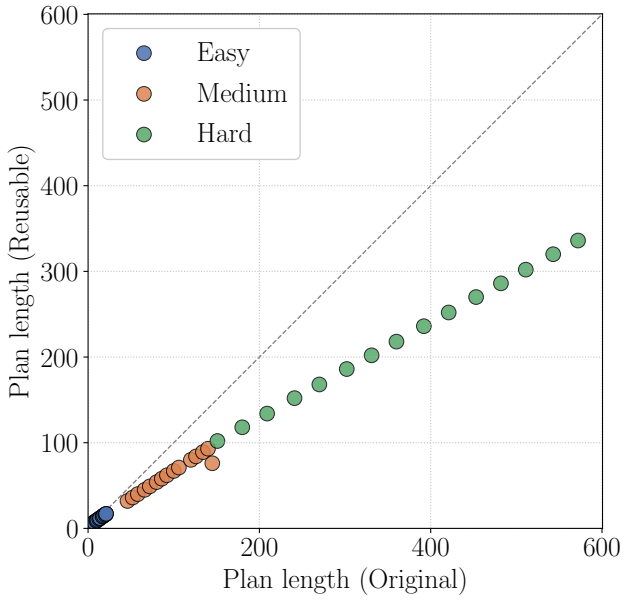
`(on -table ? b )` VS. `(j m kl kr g q ahid ti uy ? x )`

- Prior work uses 16 random characters per name: one predicate goes from 5 to 12 tokens, before any reasoning starts[1]
- And the model reasons much longer, sometimes hitting its 65,536-token thinking limit

[1] Real tokenizations (GPT-4o tokenizer, `o200k_base`).

Robustness to Modifications

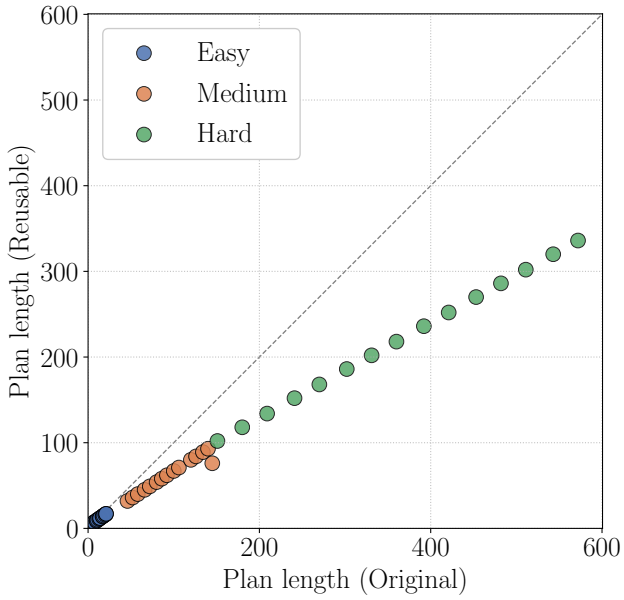
- Simplify an existing domain: remove a constraint so a simpler strategy exists



[1] C., Pereira & Seipp (2025), revised manuscript.

Robustness to Modifications

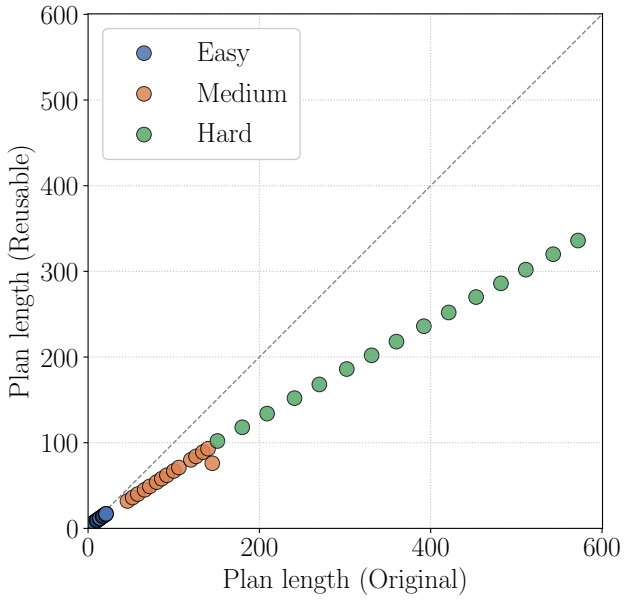
- Simplify an existing domain: remove a constraint so a simpler strategy exists
- The canonical solution stays valid: memorizing it would still solve every task



[1] C., Pereira & Seipp (2025), revised manuscript.

Robustness to Modifications

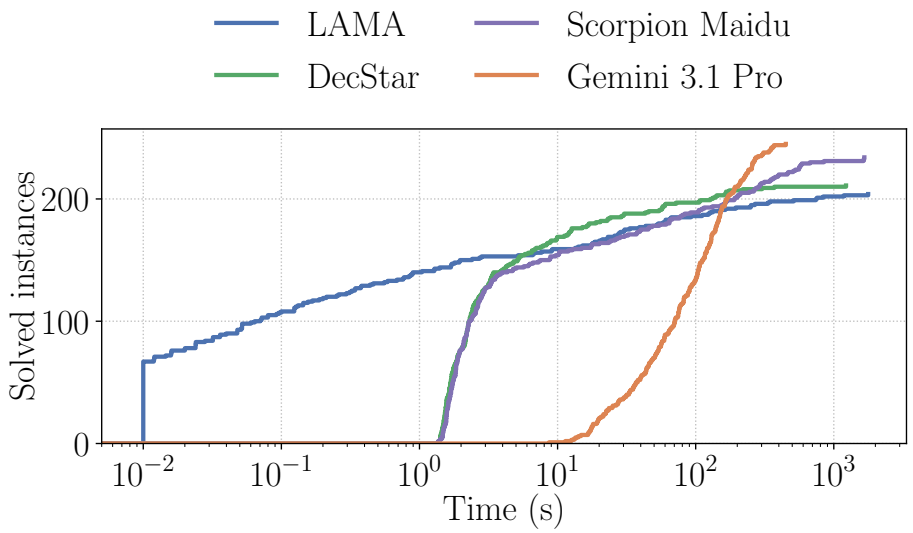
- Simplify an existing domain: remove a constraint so a simpler strategy exists
- The canonical solution stays valid: memorizing it would still solve every task
- The frontier model adapts: shorter plans on 42 of 45 tasks, longer on none
  - mean length: 157 $\rightarrow$ 98 actions



It exploits the simplification instead of replaying the known solution.[1]

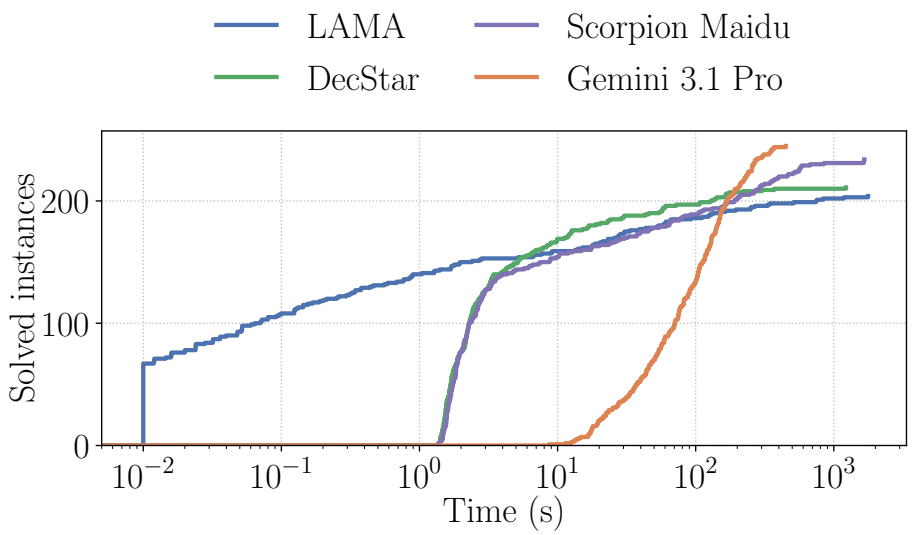
[1] C., Pereira & Seipp (2025), revised manuscript.

The computational cost



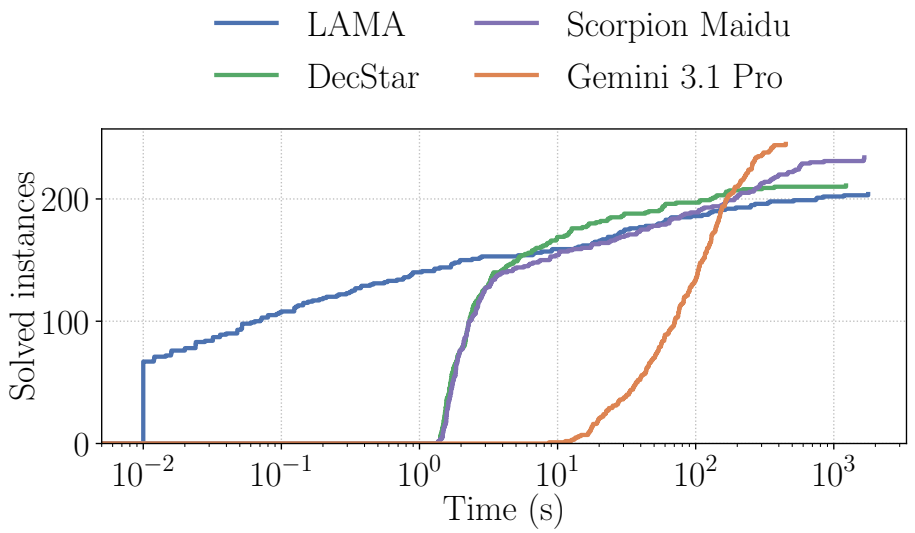
- Planners: seconds, one CPU core, 8 GiB

The computational cost



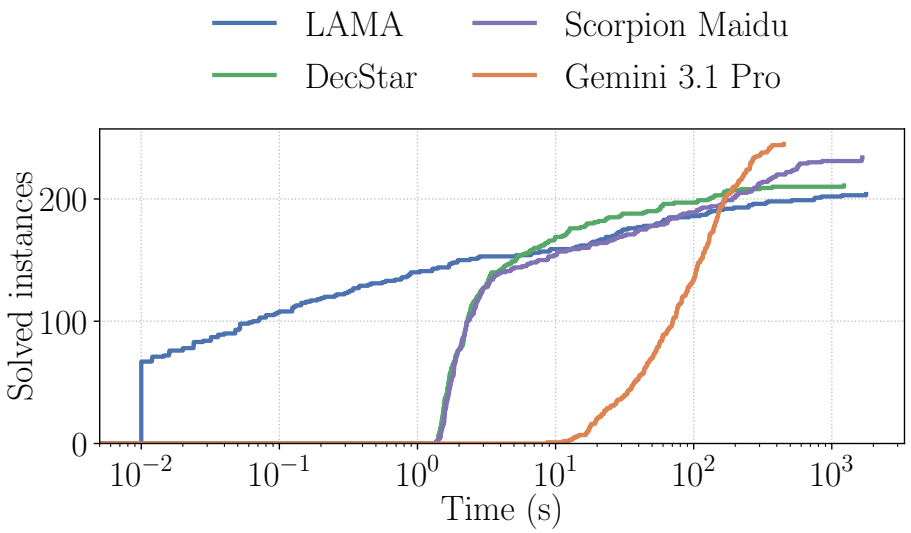
- Planners: seconds, one CPU core, 8 GiB
- DeepSeek-R1: 671B parameters, >1 000 GiB of GPU memory just to run

The computational cost



- Planners: seconds, one CPU core, 8 GiB
- DeepSeek-R1: 671B parameters, >1 000 GiB of GPU memory just to run
- Orders of magnitude more energy and wall-clock time

The computational cost



- Planners: seconds, one CPU core, 8 GiB
- DeepSeek-R1: 671B parameters, >1 000 GiB of GPU memory just to run
- Orders of magnitude more energy and wall-clock time

Whether the trade-off is worth it depends on the application.
(Can you think of any?)
However, capability is new.

Can LLMs plan?

At the frontier: yes.

Can LLMs plan?

At the frontier: yes.

Closed models, frontier scale,
no guarantees, expensive compute.

What actually changed?

1. **Scale**
   - GPT-4 was already huge and solved $\sim$12%; scale alone was not it

What actually changed?

1. **Scale**
   - GPT-4 was already huge and solved $\sim$12%;
     scale alone was not it
2. **RL on verifiable rewards**
   - models trained to "reason", not just to imitate text

What actually changed?

1. **Scale**
   - GPT-4 was already huge and solved ∼12%; scale alone was not it
2. **RL on verifiable rewards**
   - models trained to "reason", not just to imitate text

The skeptics' experiments were sound. So are these.

Both can be true, two years apart.

Still open

- Optimal planning
  - everything today is satisficing

[1] Katz et al. (2026). Position: Make Planning Research Rigorous Again!

Still open

- Optimal planning
  - everything today is satisficing
- Guarantees
  - a planner can prove unsolvability; an LLM cannot even promise an answer

[1] Katz et al. (2026). Position: Make Planning Research Rigorous Again!

Still open

- Optimal planning
  - everything today is satisficing
- Guarantees
  - a planner can prove unsolvability; an LLM cannot even promise an answer
- Cost
  - seconds on one core vs. GPU clusters

[1] Katz et al. (2026). Position: Make Planning Research Rigorous Again!

Still open

- Optimal planning
  - everything today is satisficing
- Guarantees
  - a planner can prove unsolvability; an LLM cannot even promise an answer
- Cost
  - seconds on one core vs. GPU clusters
- Hard domains remain hard
  - Rovers, Sokoban, Floortile still favor planners

[1] Katz et al. (2026). Position: Make Planning Research Rigorous Again!

Still open

- Optimal planning
  - everything today is satisficing
- Guarantees
  - a planner can prove unsolvability; an LLM cannot even promise an answer
- Cost
  - seconds on one core vs. GPU clusters
- Hard domains remain hard
  - Rovers, Sokoban, Floortile still favor planners
- Evaluation hygiene[1]
  - benchmark tasks leak into training data; evaluations need fresh instances

[1] Katz et al. (2026). Position: Make Planning Research Rigorous Again!

The question has changed

2023: “Can LLMs plan?”

The question has changed

2023: “Can LLMs plan?”

2026: “How can they help us, now that they can?”

Takeaways

1. LLMs are next-token predictors. Trained with RL against verifiers, that machinery now plans at IPC-winner level.

Takeaways

1. LLMs are next-token predictors. Trained with RL against verifiers, that machinery now plans at IPC-winner level.
2. The "LLMs can't plan" results were right at the time

Takeaways

1. LLMs are next-token predictors. Trained with RL against verifiers, that machinery now plans at IPC-winner level.
2. The "LLMs can't plan" results were right at the time
3. Classical planners remain ahead on optimality, guarantees, and efficiency

Takeaways

1. LLMs are next-token predictors. Trained with RL against verifiers, that machinery now plans at IPC-winner level.
2. The "LLMs can't plan" results were right at the time
3. Classical planners remain ahead on optimality, guarantees, and efficiency
4. Lots of exciting avenues for research!

Takeaways

1. LLMs are next-token predictors. Trained with RL against verifiers, that machinery now plans at IPC-winner level.
2. The "LLMs can't plan" results were right at the time
3. Classical planners remain ahead on optimality, guarantees, and efficiency
4. Lots of exciting avenues for research!

Thank you! Questions?

References i

- Abbe, Bengio, Lotfi, Sandon, Saremi (2024). How Far Can Transformers Reason? The Globality Barrier and Inductive Scratchpad. NeurIPS 2024. arXiv:2406.06467.
- Bohnet, Nova, Parisi, Swersky, Goshvadi, Dai, Schuurmans, Fiedel, Sedghi (2024). Exploring and Benchmarking the Planning Capabilities of Large Language Models. arXiv:2406.13094.
- Brown et al. (2020). Language Models are Few-Shot Learners. NeurIPS 2020.
- Bubeck et al. (2023). Sparks of Artificial General Intelligence: Early Experiments with GPT-4. arXiv:2303.12712.
- Corrêa, Gelberg, Melo, Shumailov, Pereira, Gal (2025). Iterative Deployment Improves Planning Skills in LLMs. arXiv:2512.24940.
- Corrêa, Pereira, Seipp (2025). Frontier Large Language Models Rival State-of-the-Art Planners. arXiv:2511.09378.
- DeepSeek-AI (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

References ii

- Howey, Long (2003). VAL's Progress: The Automatic Validation Tool for PDDL2.1. ICAPS 2003 Workshop on the IPC.
- Huang, Cohn, Lipovetzky (2025). Chasing Progress, Not Perfection: Revisiting Strategies for End-to-End LLM Plan Generation. ICAPS 2025. Also arXiv:2412.10675.
- Kambhampati (2023). Can LLMs Really Reason and Plan? Blog@CACM.
- Kambhampati (2024). Can Large Language Models Reason and Plan? Annals of the New York Academy of Sciences.
- Kambhampati, Valmeekam, Guan, Verma, Stechly, Bhambri, Saldyt, Murthy (2024). Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. ICML 2024.
- Katz, Kokel, Muise, Sohrabi, Sreedharan (2026). Position: Make Planning Research Rigorous Again! ICML 2026 (to appear).
- Ouyang et al. (2022). Training Language Models to Follow Instructions with Human Feedback. NeurIPS 2022.

References iii

- Pallagani, Muppasani, Murugesan, Rossi, Horesh, Srivastava, Fabiano, Loreggia (2022). Plansformer: Generating Symbolic Plans Using Transformers. arXiv:2212.08681.
- Rossetti, Tummolo, Gerevini, Putelli, Serina, Chiari, Olivato (2024). Learning General Policies for Planning through GPT Models. ICAPS 2024.
- Shumailov, Shumaylov, Zhao, Papernot, Anderson, Gal (2024). AI Models Collapse When Trained on Recursively Generated Data. Nature.
- Silver, Hariprasad, Shuttleworth, Kumar, Lozano-Pérez, Kaelbling (2022). PDDL Planning with Pretrained Large Language Models. NeurIPS 2022 Foundation Models for Decision Making Workshop.
- Snell, Lee, Xu, Kumar (2024). Scaling LLM Test-Time Compute Optimally Can Be More Effective Than Scaling Model Parameters. arXiv:2408.03314.
- Stechly, Valmeekam, Kambhampati (2024). Chain of Thoughtlessness? An Analysis of CoT in Planning. NeurIPS 2024.
- Taitler et al. (2024). The 2023 International Planning Competition. AI Magazine.

References iv

- Valmeekam, Olmo, Sreedharan, Kambhampati (2022). Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). arXiv:2206.10498v1; NeurIPS 2022 Foundation Models for Decision Making Workshop.
- Valmeekam, Marquez, Sreedharan, Kambhampati (2023). On the Planning Abilities of Large Language Models — A Critical Investigation. NeurIPS 2023.
- Valmeekam, Marquez, Olmo, Sreedharan, Kambhampati (2023). PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. NeurIPS 2023.
- Valmeekam, Stechly, Kambhampati (2024). LLMs Still Can't Plan; Can LRMs? A Preliminary Evaluation of OpenAI's o1 on PlanBench. arXiv:2409.13373.
- Vaswani et al. (2017). Attention Is All You Need. NeurIPS 2017.
- Wei, Wang, Schuurmans, Bosma, Ichter, Xia, Chi, Le, Zhou (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. NeurIPS 2022.
- Wolfram (2023). What Is ChatGPT Doing... and Why Does It Work? writings.stephenwolfram.com.

References v

- Zelikman, Wu, Mu, Goodman (2022). STaR: Bootstrapping Reasoning With Reasoning. NeurIPS 2022.