

# A Brief Tutorial on LLMs for AI Planning.

Harsha Kokel

<https://harshakokel.com>



1<sup>st</sup> DAAIG Workshop on Modern LLMs and Applications  
22-23 March, 2025





DAIICT, Jan 2025



James Oswald  
RPI



Daniel Cao  
Cornell University



Harsha Kokel



Kavitha Srinivas  
IBM Research



Michael Katz  
IBM Research



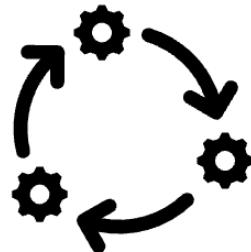
Shirin Sohrabi  
IBM Research

# Outline

- AI Planning: Introduction
- LLMs in AI Planning
  - LLMs as Planners
  - LLMs for generating code
  - LLMs for extracting action models
- Benchmarking and Evaluations

# What is AI Planning?

# What is AI Planning?



AI Planning is a sub-field of AI  
that explores autonomous techniques to solve  
planning problems.

# What is a Planning Problem?

# What is a Planning Problem?



Given the following description

- an initial state
- a goal (or objective)
- a set of actions that transform the state

devise a sequence of action that achieves the goal

# What is a Planning Problem?

Formally,

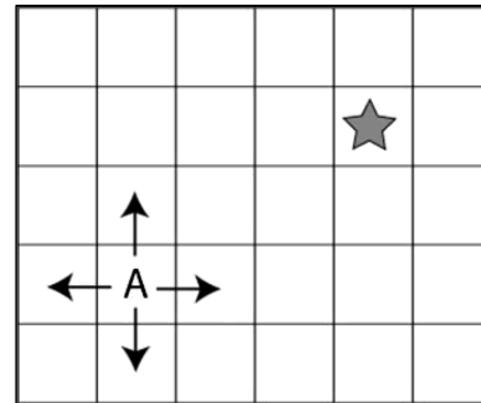
A classical planning problem includes

- a finite and discrete state space  $S$
- a known initial state  $s_0 \in S$
- a set  $S_G \subseteq S$  of goal states
- a set of actions  $A$
- a deterministic transition function  $s' = f(s, a)$

# PDDL – Planning Domain Definition Language

```
( define (domain grid)
  (:requirements :strips)
  (:types place - object)
  (:predicates (adj ?x ?y)
    (at-robot ?x))

  (:action move
    :parameters (?from – place ?to – place)
    :precondition (and (at-robot ?from)
      (adj ?from ?to))
    :effect (and (at-robot ?to)
      (not (at-robot ?from)))
  )
)
```



```
( define (problem small)
  (:domain grid)
  (:objects node0-0 node0-1 ...)
  (:init (at-robot node1-1)
    (adj node0-0 node0-1) ...)
  (:goal (at-robot node3-4))
)
```

# Why PDDL?

- **Formal Analysis**
- **Standardization:**
  - Common language for International Planning Competition (IPC)
- **Clarity and Precision:**
  - important for soundness
- **Tool Compatibility:**
  - Allows domain-independent planning systems

# Planning Problem Variants

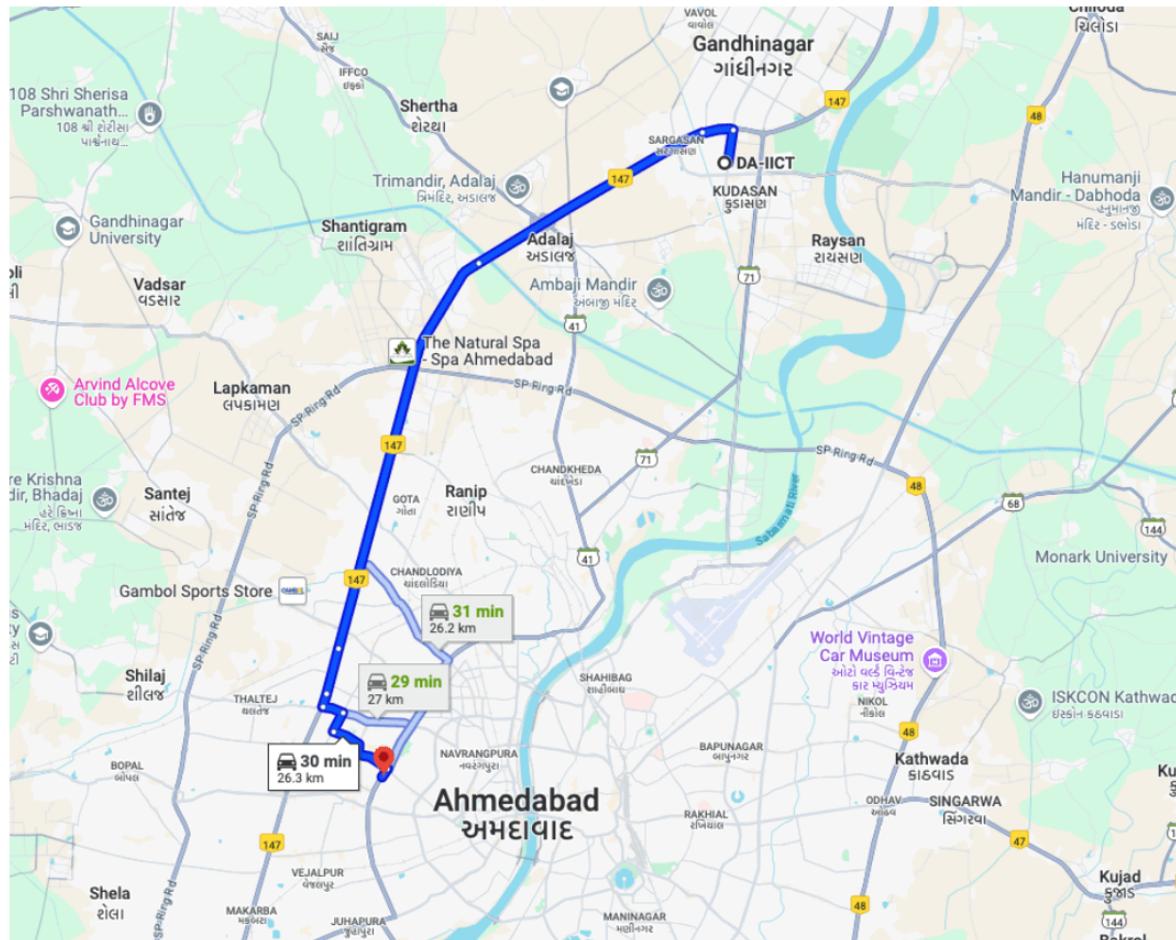
- **action dynamics:**  
deterministic, nondeterministic, probabilistic
- **observability:** full, partial
- **actions:** instantaneous, durative, hierarchical
- **horizon:** finite, infinite
- **objective:** satisfying, optimizing
- **system:** static, dynamic

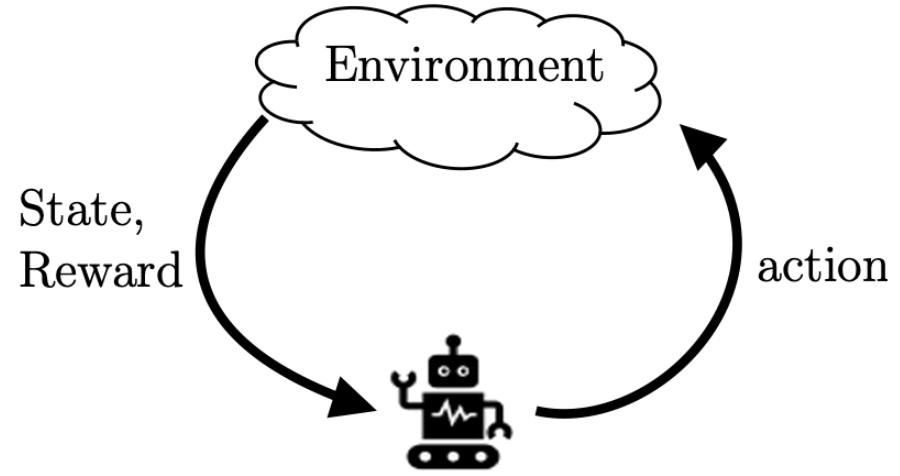
# Planning Tools and Resources

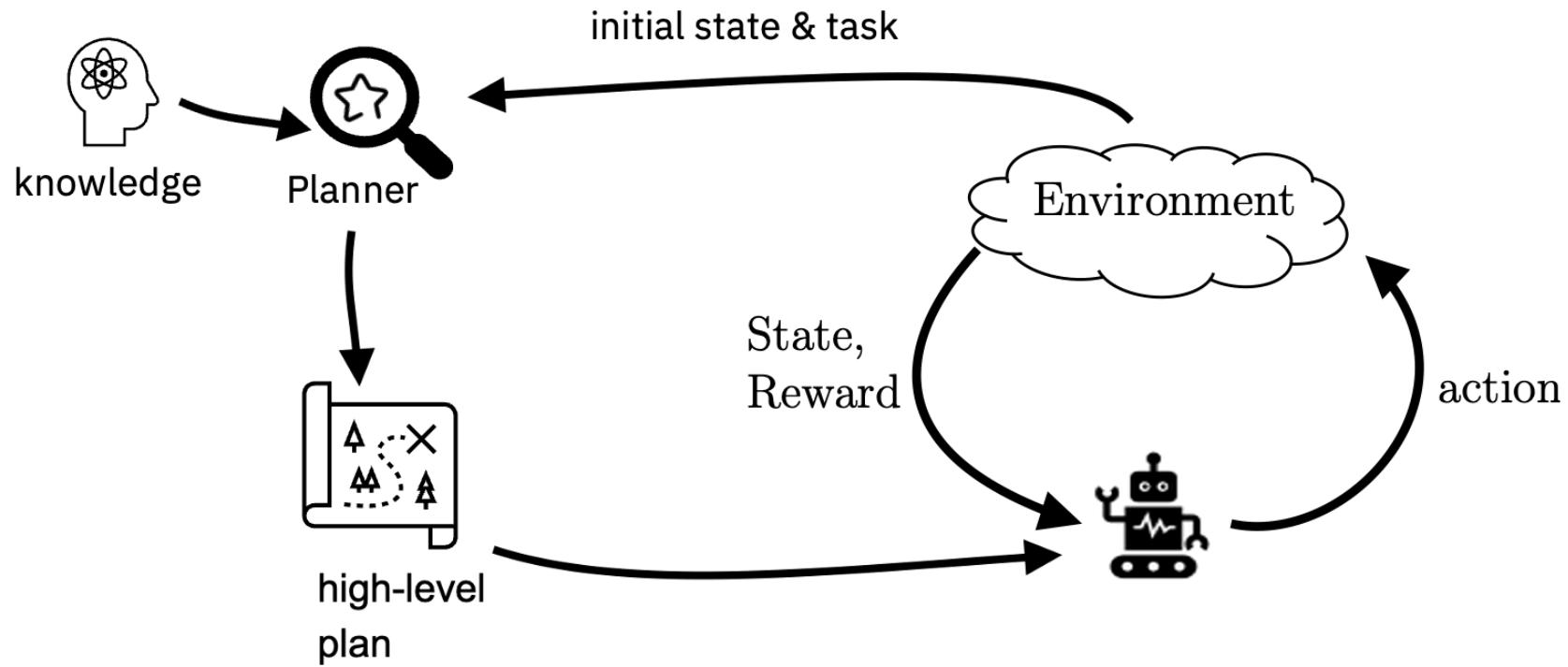
**!! Domain Independent**

- Fast-Forward (FF): classical satisficing, numeric, conformant, contingent (Hoffmann & Nebel, 2001)
- Fast Downward: classical, cost-optimal, satisficing, agile, cost-bounded, OSP, FOND, probabilistic, temporal (Helmert et al., 2006)
- SHOP2, PyHOP: HTN planning (Nau et al., 2003)
- LPG: classical, satisficing, numeric, temporal, diverse (Gerevini & Serina 2002)
- FOND planner PRP (Muise et al., 2012, 2014)
- OSP planners (Katz & Keyder 2019, Katz & Speck 2021)
- Top-k planners: K\* (Katz et al., 2018) SymK (Speck et al., 2020)
- Forbid-iterative collection of planners for top-k, top-quality, diverse (Katz & Sohrabi 2020, Katz et al., 2020)

# Planning vs Acting





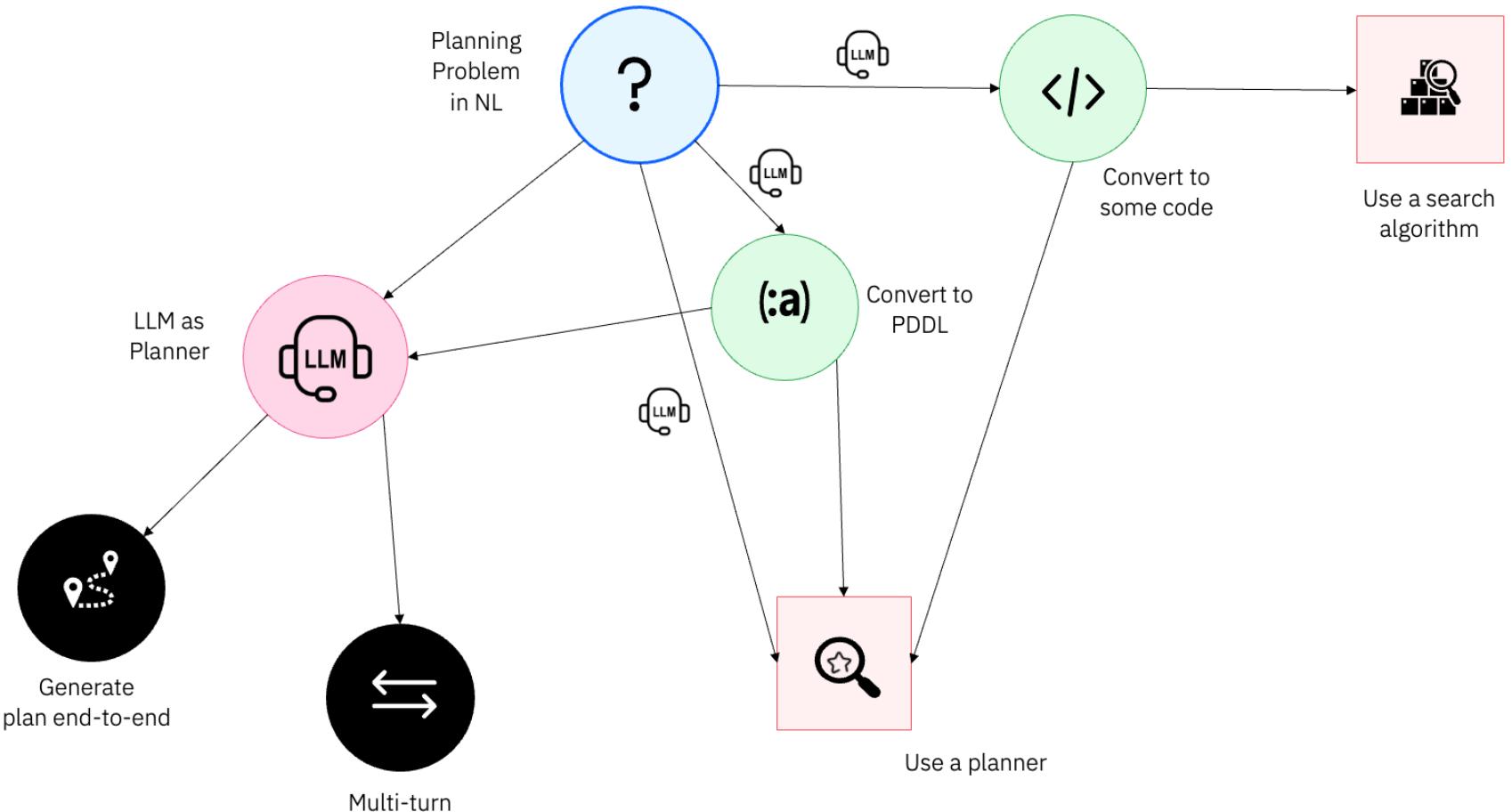


Kokel et al. ICAPS 2022

# Outline

## ✓ **AI Planning: Introduction**

- LLMs in AI Planning
  - LLMs as Planners
  - LLMs for generating code
  - LLMs for extracting action models
- Benchmarking and Evaluations



# LLM as Planners



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

Wenlong Huang, Pieter Abbeel, Deenak Pathak, Iaor Mordatch Proceedings of the 39th

International Conference

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

[← Go to ICLR 2023 Conference homepage](#)

[← Go to ICLR 2023 Conference homepage](#)

There is com  
reasoning t  
just the rig

# ReAct: Synergizing Reasoning and Acting in Language Models



Shunyu Yao, Jeffrey

Published: 01 Feb 2023, Last updated: 01 Feb 2023

**Keywords:** Language mod

**TL;DR:** We synergize reason

**Abstract:** While large lang

## LLMs Still Can't Plan; Can LRM<sub>s</sub>? A Preliminary Evaluation of OpenAI's o1 on PlanBench

Karthik Valmeekam, Kaya Stechly, Subbarao Kambhampati

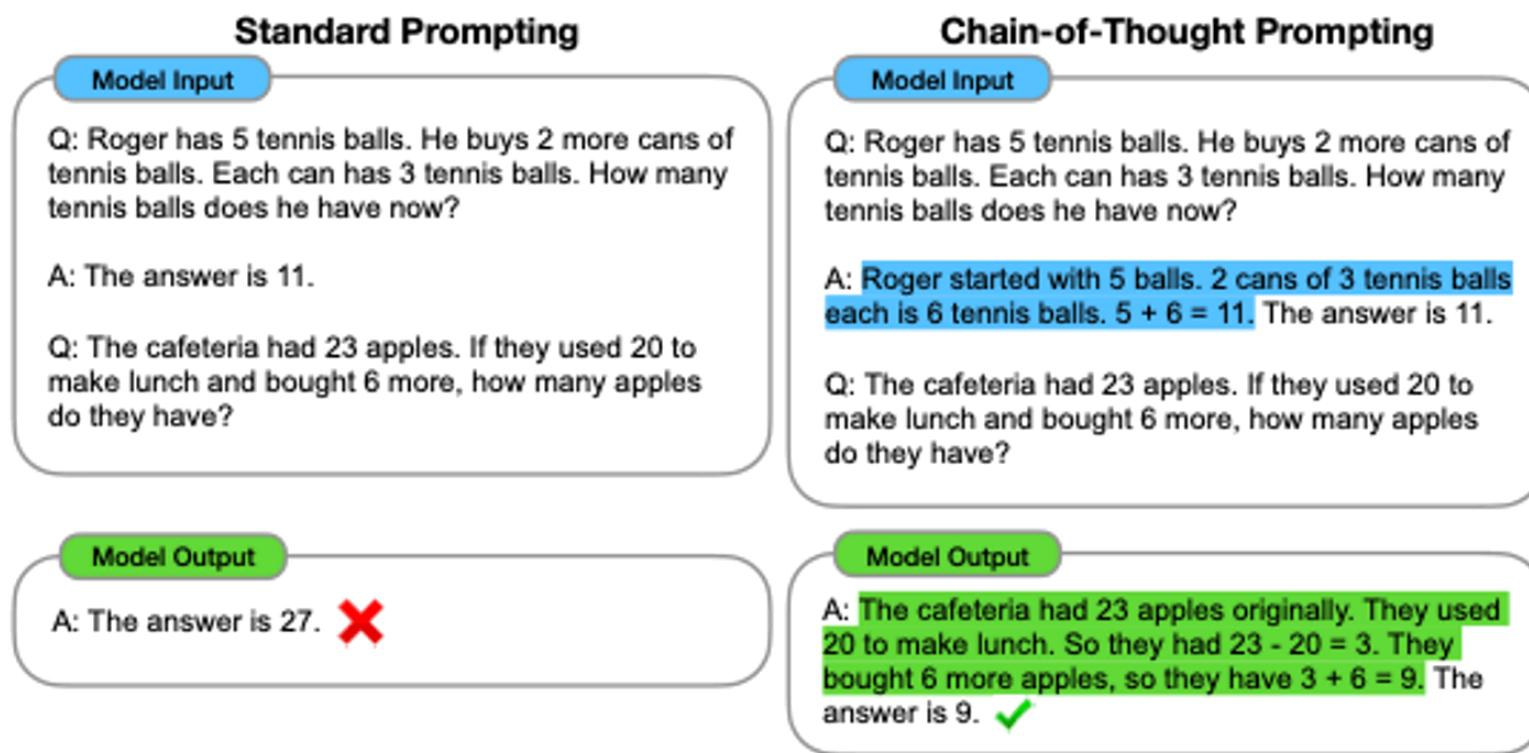
The ability to plan a course of action that considered a core competence of intelligence since its inception. With the advent of large interest in the question of whether or not

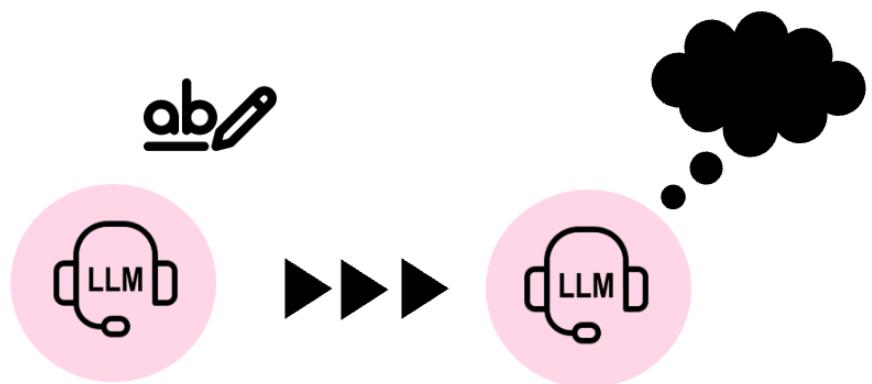
# Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context

Gemini Team, Google<sup>1</sup>

We present our latest multimodal models from the Gemini line: Gemini 1.5 Pro and Gemini 1.5 Flash. They are members of Gemini 1.5, a new family of highly-capable multimodal models which incorporates our latest innovations in sparse and dense scaling as well as major advances in training, distillation and serving infrastructure that allow it to push the boundary of efficiency, reasoning, planning, multi-linguality, function calling and long-context performance.

# COT: Reasoning Abilities



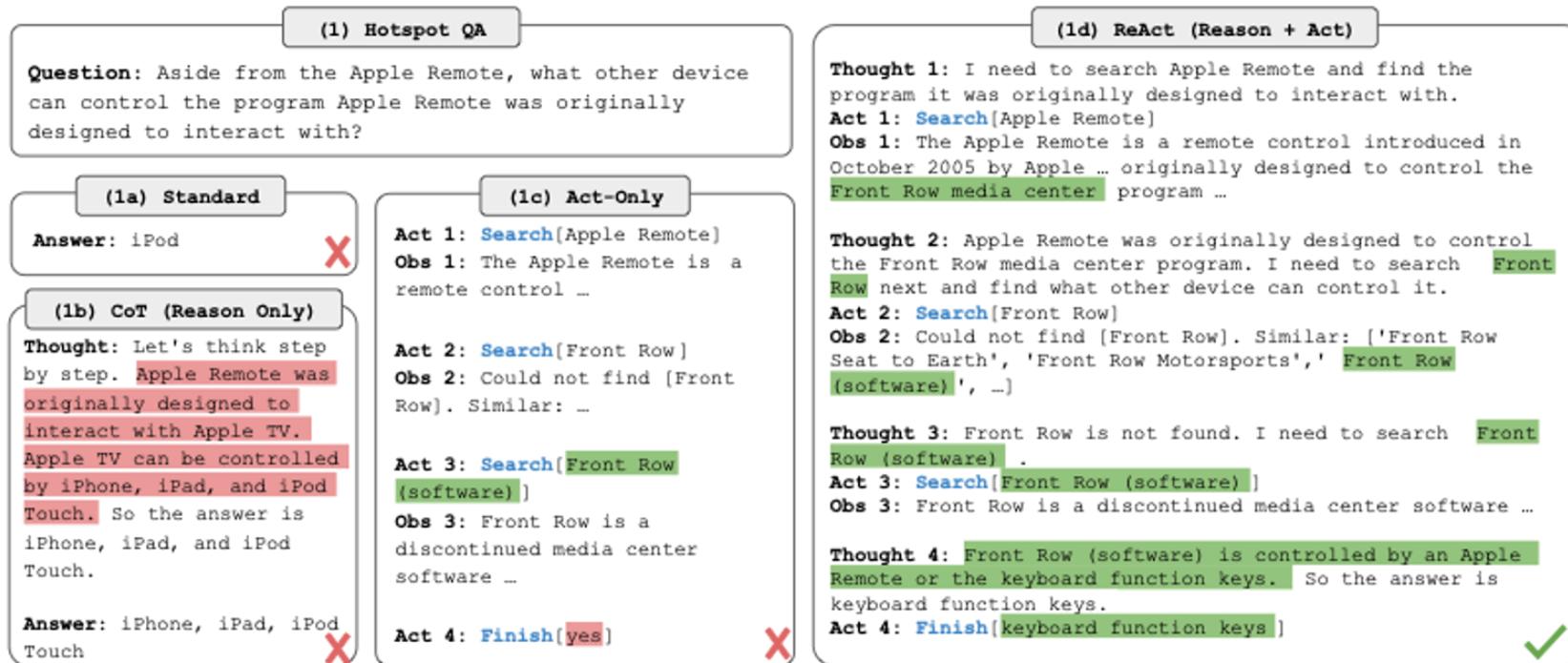


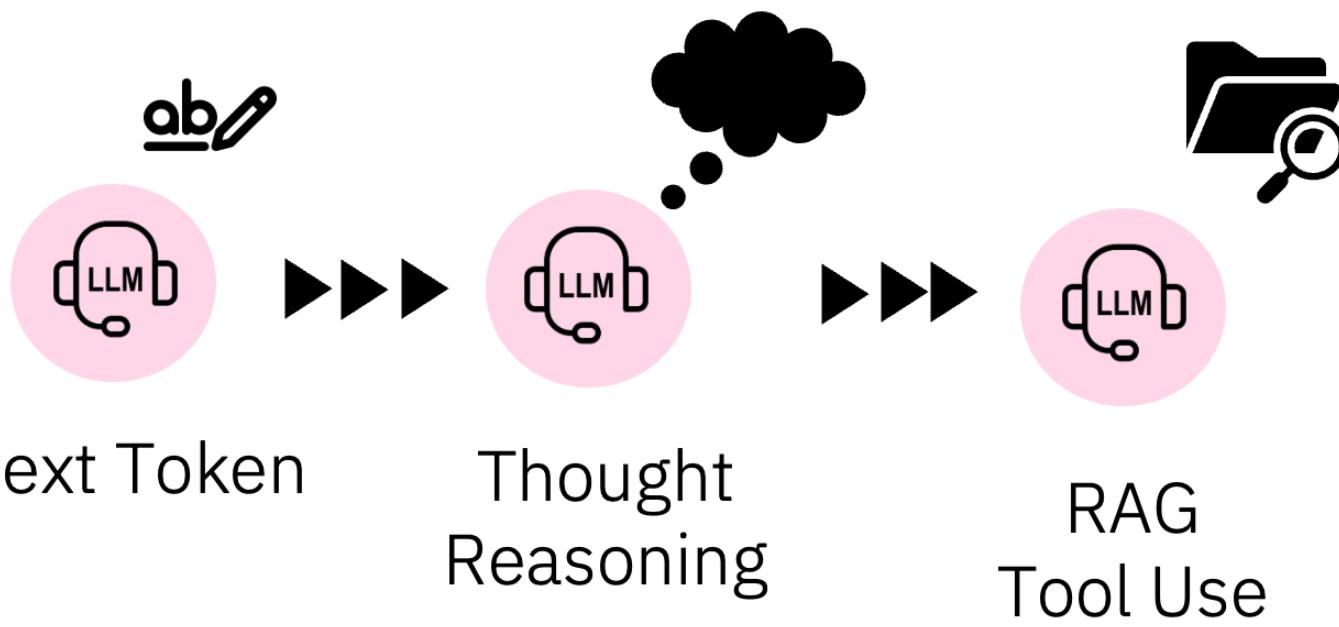
Next Token

Thought  
Reasoning

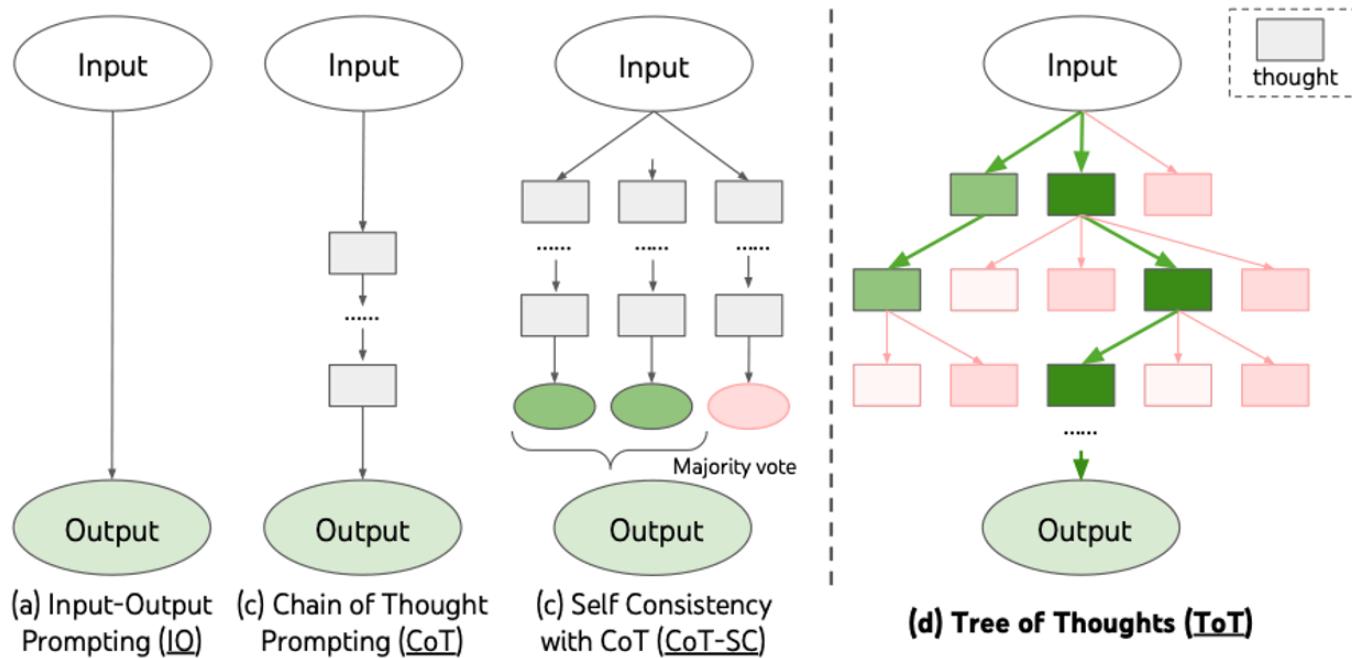
# ReAct:

## Reasoning and Tool Use/Acting

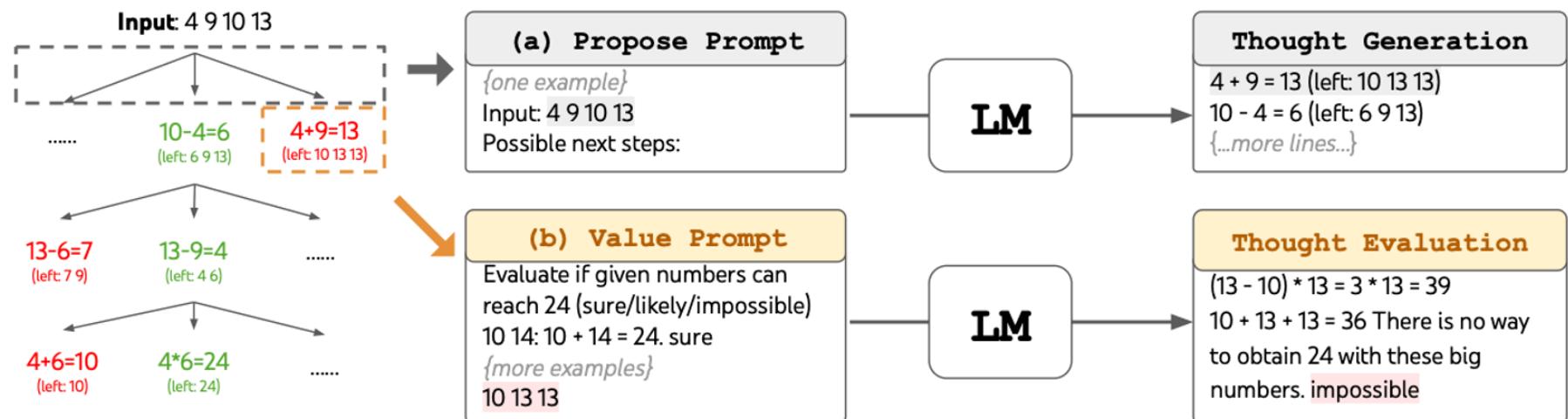


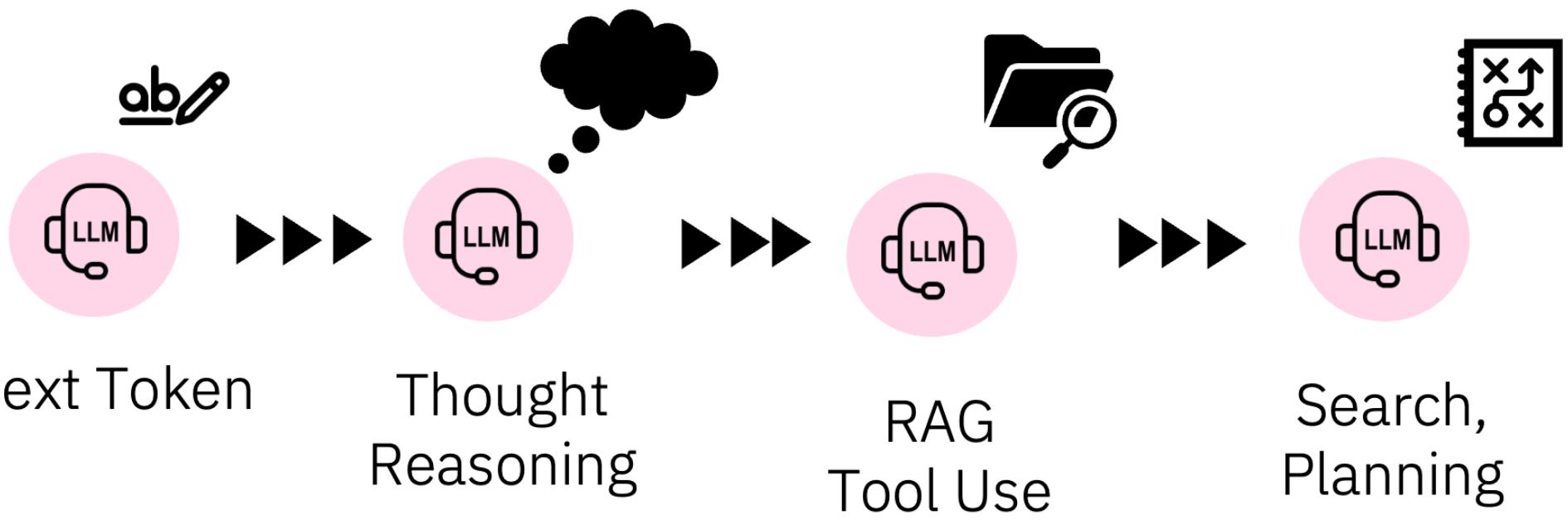


# Tree of Thoughts: Search



# Game of 24





Chain-Of-Thoughts **Reflexion**  
IO **ReWOO** COT  
**ReAct** Tree-of-thoughts  
ToT **RAP** Graph-of-thoughts  
Reasoning-via-Planning **LATS**

and many more...

- No guarantees of soundness
  - o solution generated may or may not work
- No guarantees of completeness
  - o can miss correct solutions
- No efficiency considerations
  - o extremely high # LM Evaluation
  - o tackle one problem at a time
  - o worse !! Generate one action at a time

Chain-Of-Thoughts **Reflexion**  
IO **ReWOO** COT  
**ReAct** Tree-of-thoughts  
ToT **RAP** Graph-of-thoughts  
Reasoning-via-Planning **LATS**

and many more...

# Calls: 1

Chain-Of-Thoughts **Reflexion**  
**IORewOO** COT  
**ReAct** Tree-of-thoughts  
ToT      **RAP** Graph-of-thoughts  
Reasoning-via-Planning **LATS**

and many more...

Chain-of-Thoughts Reflexion  
Rewoo COT

ReAct Tree-of-thoughts  
ToT RAP Graph-of-thoughts

Reasoning-via-Planning LATS

and many more...

# Calls: bml

Reflexion  
RewOO cOT  
ReAct Tree-of-thoughts  
ToT Graph-of-thoughts  
RAP Reasoning-via-Planning LATS

and many more...

# Calls: L

Reflexion  
Rewoo COT  
**R**eA**C**t - catch-of-thoughts  
Reasoning-via-Planning **LATS**

and many more...

# Calls: 2 to L

# Reflexion

# Rewo<sup>o</sup>o

ToT      **RAP**      Graph-of-thoughts  
Reasoning-via-Planning **LATS**

and many more...

# Calls:  $(B+1)LT$

# Reflexion

COT

ToT    RAP    Graph-of-thoughts  
Reasoning-via-Planning

and many more...

# Calls: BL

# Reflexion

COT

ToT

RAP

Graph-of-thoughts

LATS

and many more...

**# Calls: (B+2)LT**

# Reflexion

COT

ToT      RAP

LATS

and many more...

**# Calls: (2+L)T**

# Reflexion COT

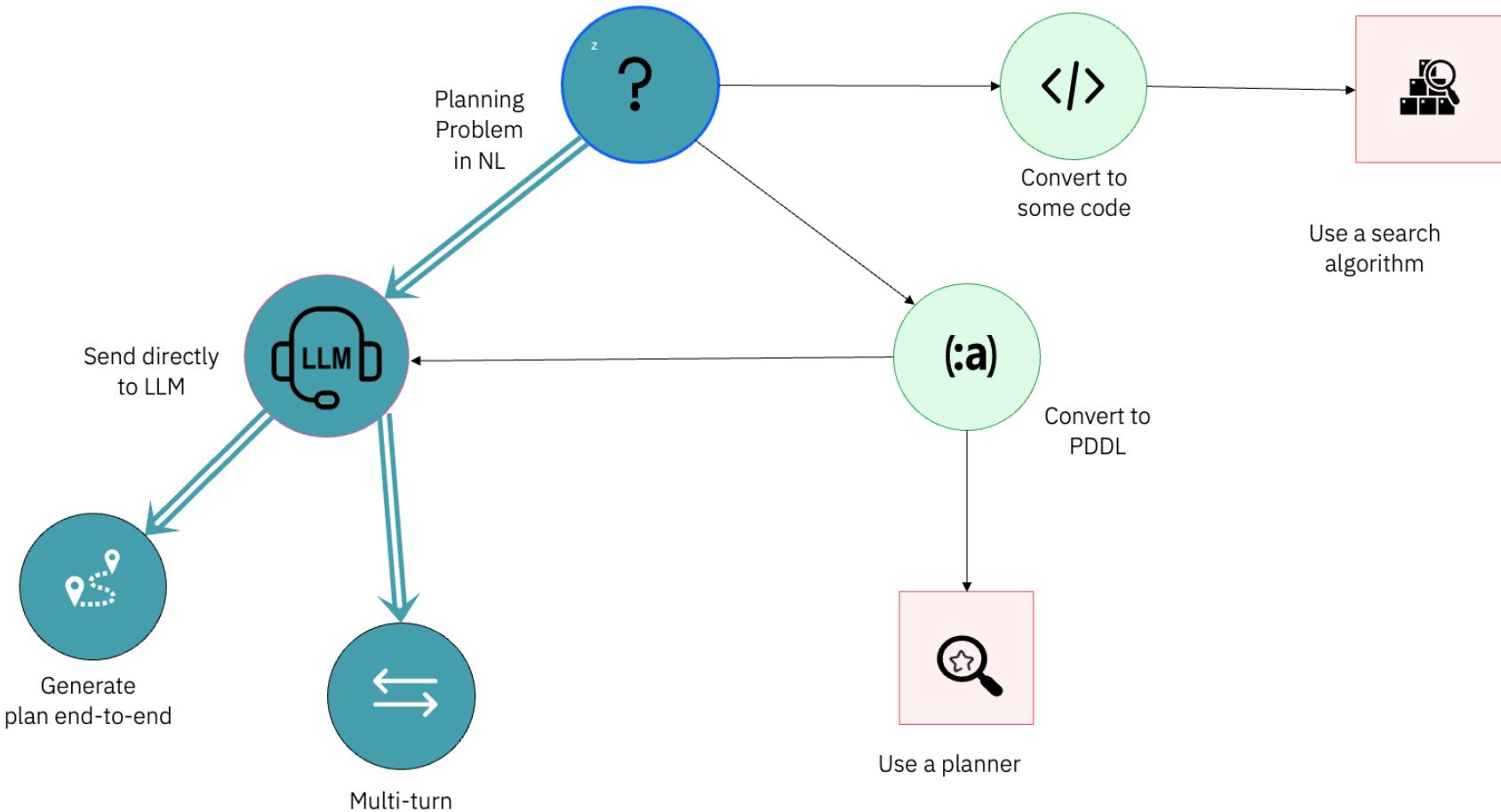
ToT      **RAP**

and many more...

Chain-Of-Thoughts **Reflexion**  
IO **ReWOO** COT  
**ReAct** Tree-of-thoughts  
ToT **RAP** Graph-of-thoughts  
Reasoning-via-Planning **LATS**

and many more...

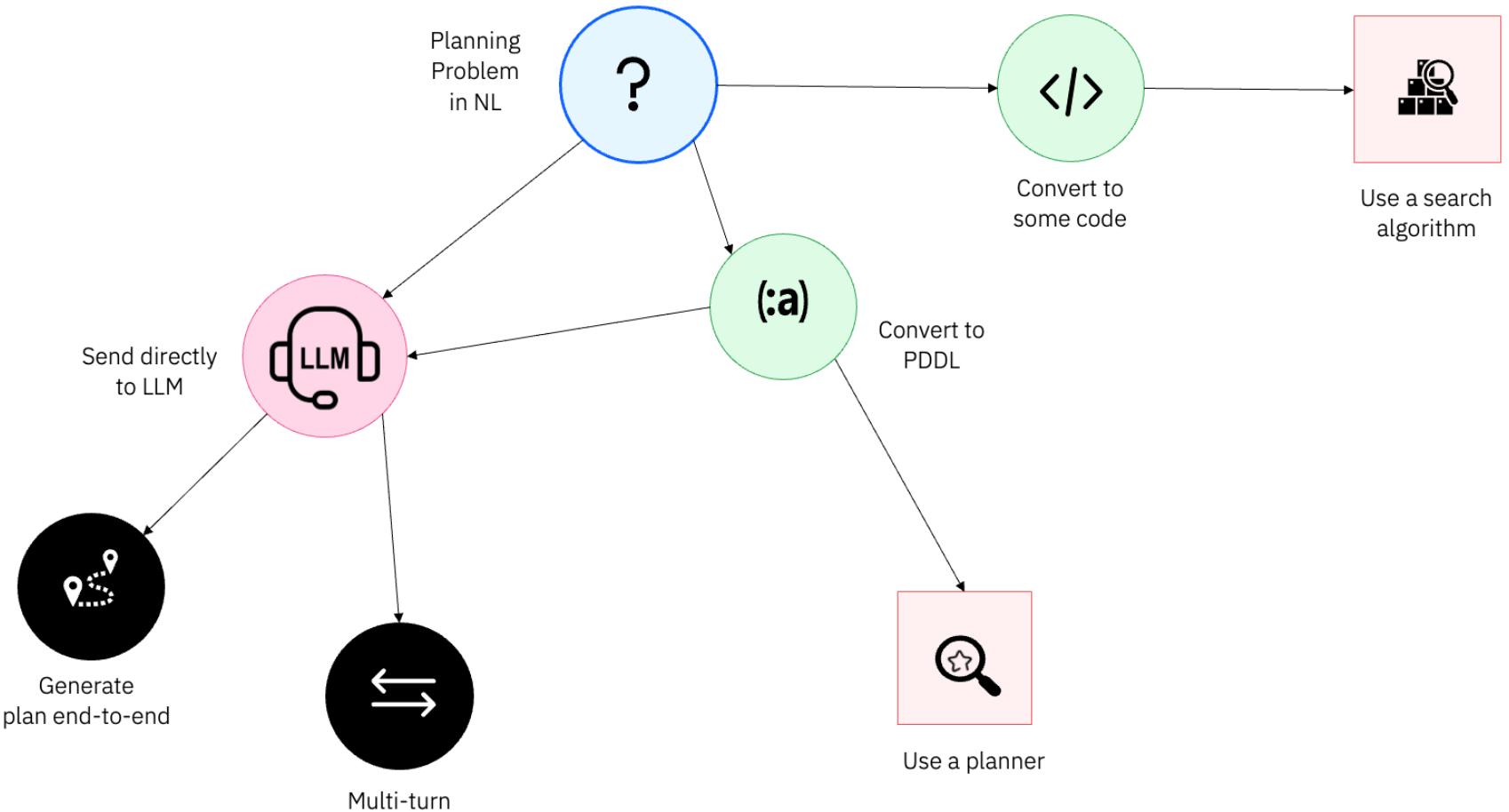
They abandon soundness and completeness for the sake of inefficiency !



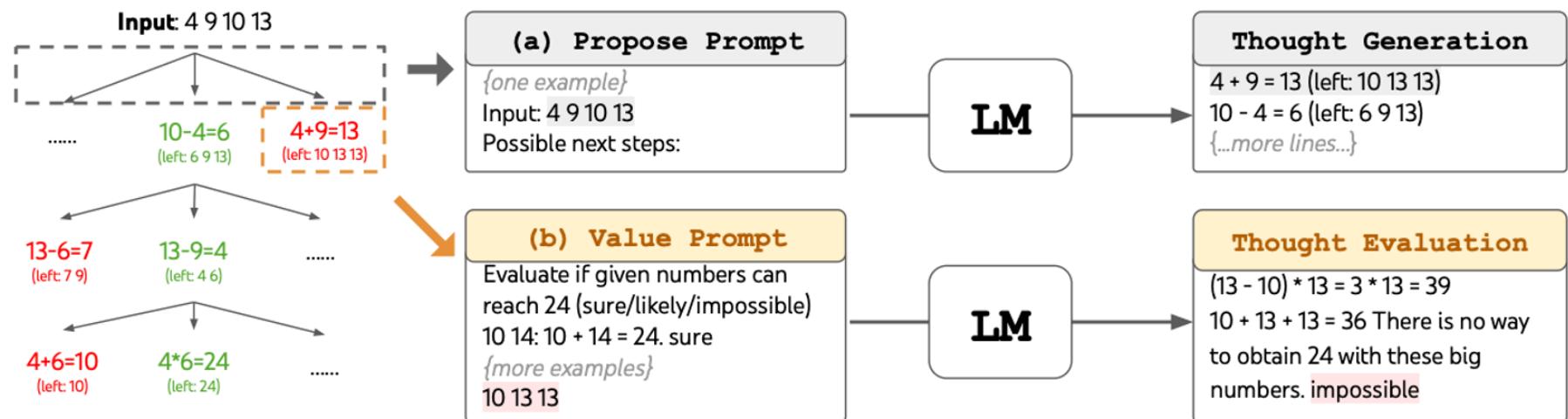
# LLM for Code Generation



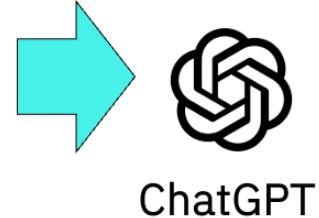
Katz et al NeurIPS 2024



# Game of 24

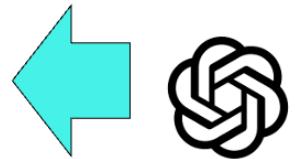


User: The 24 Game is a mathematical card game in which the objective is to find a way to manipulate four integers so that the end result is 24. The game is played with a list of four numbers, and the player must use all four numbers exactly once, using any combination of addition, subtraction, multiplication, or division, to arrive at the number 24. If a state is defined by a list of numbers (4 or less), how are successor states defined? Provide only the python code that returns a list of successor states for a state.



Below is the Python code that defines how to generate the successor states for a given state in the 24 Game. A state in the 24 Game is a list of numbers. The successor states are generated by taking two numbers from the state, applying an operation (addition, subtraction, multiplication, or division if the divisor isn't zero), and then forming a new list with the result and the remaining unused numbers.

```
import itertools
def generate_successor_states(state):
    successors = []
    n = len(state)
    for (i, j) in itertools.combinations(range(n), 2):
        a, b = state[i], state[j]
        remaining = [state[k] for k in range(n) if k != i and k != j]
        # Generate all possible new states
        successors.append(remaining + [a + b])
        successors.append(remaining + [a * b])
        if a > b:
            successors.append(remaining + [a - b])
        else:
            successors.append(remaining + [b - a])
        if b != 0:
            successors.append(remaining + [a / b])
        if a != 0:
            successors.append(remaining + [b / a])
    return successors
```



ChatGPT

*User:* The 24 Game is a mathematical card game in which the objective is to find a way to manipulate four integers so that the end result is 24. The game is played with a list of four numbers, and the player must use all four numbers exactly once, using any combination of addition, subtraction, multiplication, or division, to arrive at the number 24. If a state is defined by a list of numbers (4 or less), how are successor states defined? Provide only the python code that returns a list of successor states for a state.

*ChatGPT:* Below is the Python code that defines how to generate the successor states for a given state in the 24 Game. A state in the 24 Game is a list of numbers. The successor states are generated by taking two numbers from the state, applying an operation (addition, subtraction, multiplication, or division if the divisor isn't zero), and then forming a new list with the result and the remaining unused numbers.

```
import itertools

def generate_successor_states(state):
    successors = []
    n = len(state)
    for (i, j) in itertools.combinations(range(n), 2):
        a, b = state[i], state[j]
        remaining = [state[k] for k in range(n) if k != i and k != j]

        # Generate all possible new states
        successors.append(remaining + [a + b])
        successors.append(remaining + [a * b])
        if a > b:
            successors.append(remaining + [a - b])
        else:
            successors.append(remaining + [b - a])
        if b != 0:
            successors.append(remaining + [a / b])
        if a != 0:
            successors.append(remaining + [b / a])

    return successors
```

This function takes a state (a list of numbers), iterates over all pairs of numbers, applies the operations to these pairs, and then forms new lists with the result of the operation and the other numbers that were not used in the operation. This ensures that each successor state is a valid step towards finding a solution to reach 24, using all four numbers exactly once along with any of the allowed operations.

*User:* Provide only the python code that tests whether a state is a goal state.

*ChatGPT:* Here's the Python code that tests whether a given state is a goal state in the 24 Game. A goal state is achieved if the list contains exactly one number, and that number is 24.

```
def is_goal_state(state):
    return len(state) == 1 and state[0] == 24
```

This function checks if the length of the state list is 1 and whether the sole element in the list is the number 24.

# Thought of Search

- Human can verify the code
- Plug it in to any search algorithm
  - o guaranteed correct solution
- Efficient
  - o For game of 24, all 1362 games in < 2 sec
  - o 1.2 queries to LLM on average for sound successor function

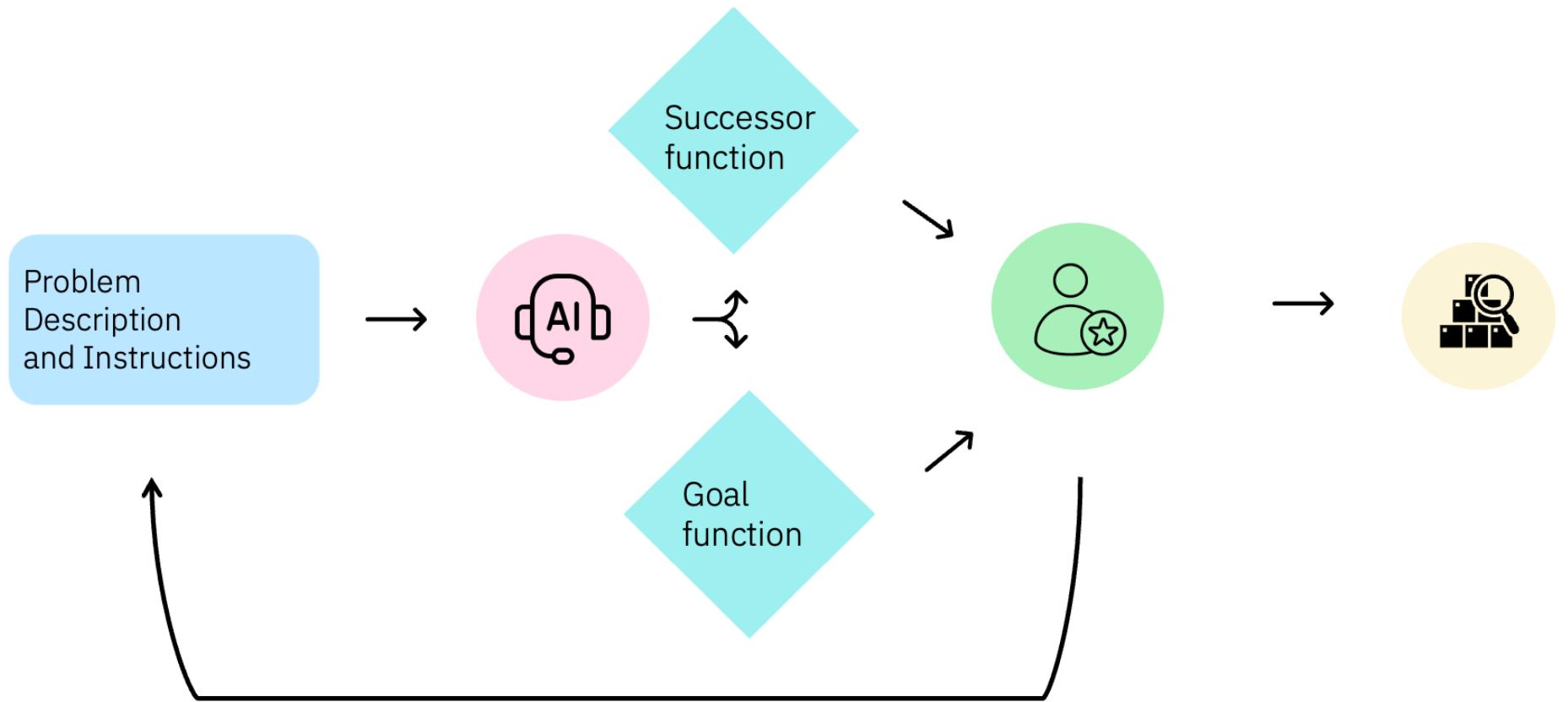
# Experiments

Approach	Complexity	24Game		Crossword		BlocksWorld		PrOntoQA	
		States	Calls	States	Calls	States	Calls	States	Calls
IO	$O(D)$	0.02%	1362	4e-9%	20	0.5%	502	4%	4000
CoT	$O(D)$	0.02%	1362	4e-9%	20	0.5%	502	4%	4000
ReAct	$O(LD)$	0.07%	4086	4e-8%	200	7.8%	8032	24.6%	24K
ReWOO	$O(LD)$	0.07%	4086	4e-8%	200	7.8%	8032	24.6%	24K
RAP	$O(TbLD)$	3.3%	245K	2e-6%	12K	388%	482K	1229%	1.44M
ToT	$O(bmLD)$	1.6%	102K	1e-6%	5K	194%	201K	615%	600K
GoT	$O(bLD)$	0.3%	20K	2e-7%	1K	39%	40K	122%	120K
Reflection	$O(LTD)$	0.7%	68K	4e-7%	2.4K	77.6%	90K	245%	320K
LATS	$O(TbLD)$	3.3%	286K	2e-6%	14K	388%	562K	1229%	1.68M
ToS (ours)	$O(1)$	27.0%	<b>2.2</b>	3e-4%	<b>3.8</b>	125%	<b>3.8</b>	175%	<b>2.6</b>

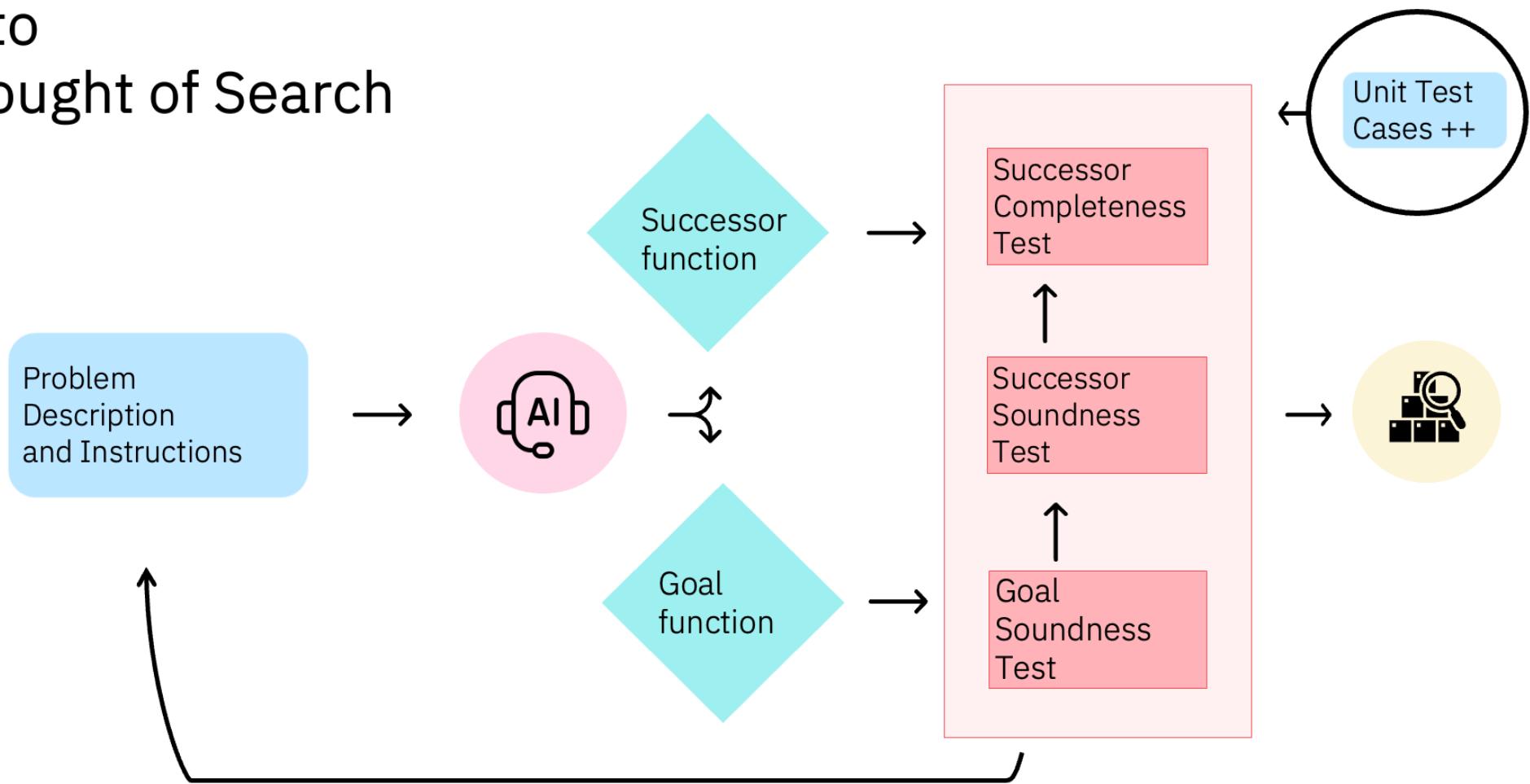
Table 1: Comparison of various approaches on evaluated tasks. The number of tasks is  $D$ , the length bound of the search/rollout/plan is  $L$ , number of rollouts is  $T$ , beam size is  $m$ , and branching bound is  $b$ . The summed number of states over the  $D$  tasks and the projected number of LLM evaluations are given per approach and problem.

100% instances solved

# Thought of Search

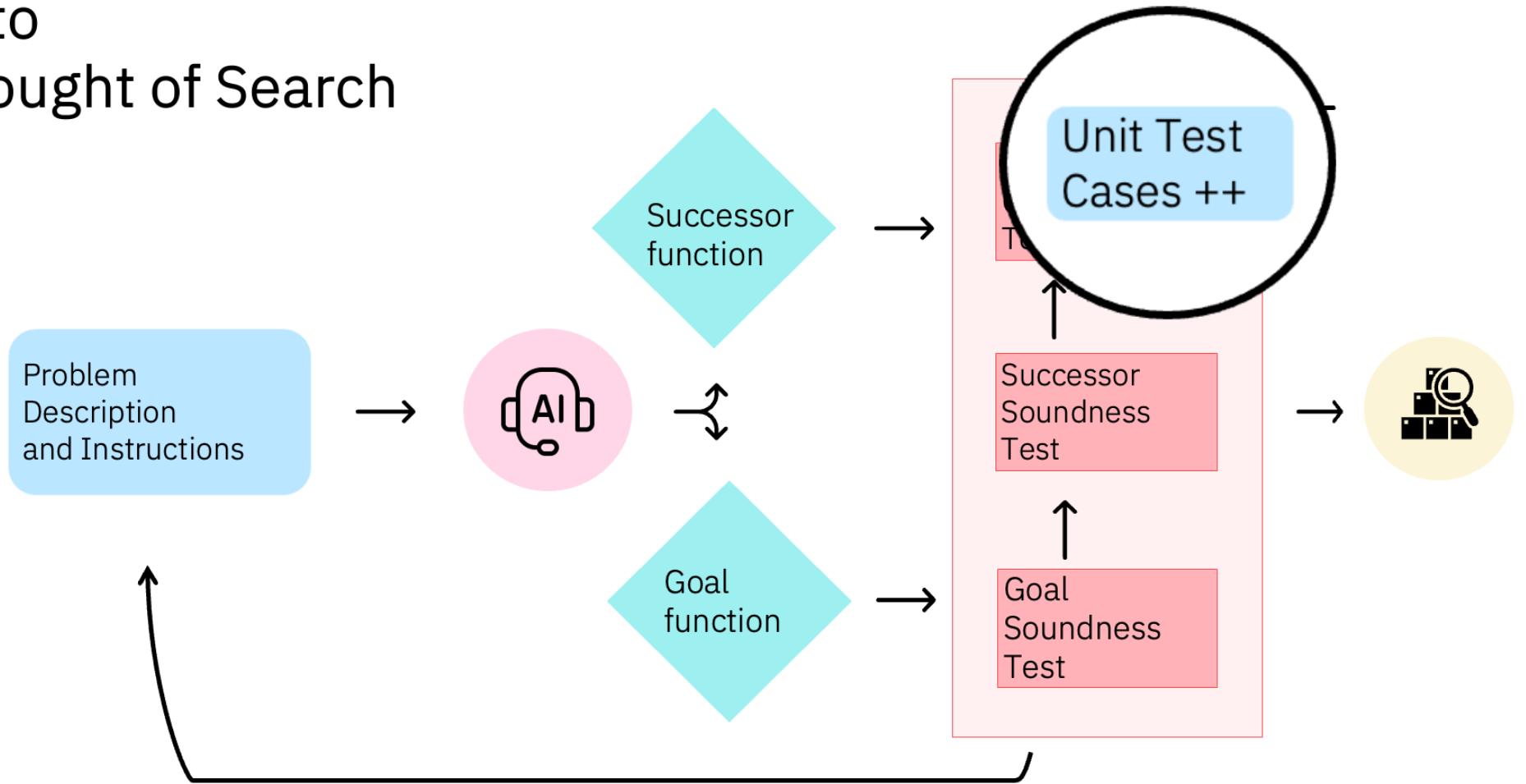


# Auto Thought of Search



Cao et al 2025

# Auto Thought of Search



Cao et al 2025

## Unit Test Cases ++

Goal states:

[24]

Non-goal states:

[], [3], [24, 1], [1, 6, 4], [1, 1, 4, 6]

Successors:

[[1, 1, 1, 8], [[0.125, 1, 1], [1, 1, 9], [1, 1, 8], [0, 1, 8], [1, 2, 8], [1, 1, 7], [-7, 1, 1]]]  
[[6, 6, 6, 6], [[1.0, 6, 6], [6, 6, 12], [0, 6, 6], [6, 6, 36]]]]

Partial soundness test:

```
def validate_transition_complex(self, s, t):
    if len(s) - len(t) != 1:
        feedback = search pprint("Invalid transformation: length mismatch – the length of a successor must be
        one less than the parent.")
        feedback += search pprint("Let's think step by step. First think through in words why the successor
        function produced a successor that had a length that was not exactly one less than the parent. Then
        provide the complete Python code for the revised successor function that ensures the length of a
        successor is exactly one less than the parent.")
        feedback += search pprint("Remember how you fixed the previous mistakes, if any. Keep the same
        function signature.")
        return False, feedback
    return True, ""
```

Game of 24

# Feedback

---

## Listing 1: 24 Game example feedback.

---

The goal test function failed on the following input state [24, 1], incorrectly reporting it as a goal state. First think step by step what it means for a state to be a goal state in this domain. Then think through in words why the goal test function incorrectly reported input state: [24, 1] as a goal state. Now, revise the goal test function and ensure it returns false for the input state. Remember how you fixed the previous mistakes, if any. Keep the same function signature.

Invalid transformation: length mismatch - the length of a successor must be one less than the parent. Let's think step by step. First think through in words why the successor function produced a successor that had a length that was not exactly one less than the parent. Then provide the complete Python code for the revised successor function that ensures the length of a successor is exactly one less than the parent. Remember how you fixed the previous mistakes, if any. Keep the same function signature.

Input state: [1, 1, 4, 6] Example wrong successor state: [6, 5]

Successor function when run on the state [1, 1, 4, 6] failed to produce all successors. Missing successors are: [[1, 4, 7], [-5, 1, 4], [1, 1, 2], [1, 5, 6], [0.25, 1, 6], [-3, 1, 6], [0.1666666666666666, 1, 4], [1, 3, 6], [1, 4, 5], [1, 1, 1.5]] First think step by step why the successor function failed to produce all successors of the state. Then, fix the successor function. Remember how you fixed the previous mistakes, if any. Keep the same function signature.

# Experiments

	24 Game	PrOntoQA	Sokoban	Crossword	BlocksWorld
AutoTOS	GPT-4o-mini	8.8	4.8	6.4	9.6
	GPT-4o	3.4	2.6	2.2	5.8
	Llama3.1-405b	3.4	2.0	2.6	4.0
	Llama3.1-70b	7.4	2.0	8.2	6.2
	DeepSeek-CoderV2	4.4	2.0	2.8	6.6
	ToS GPT-4	2.2	2.6	NA	4.2

Table 1: The average number of calls to the language model per domain.

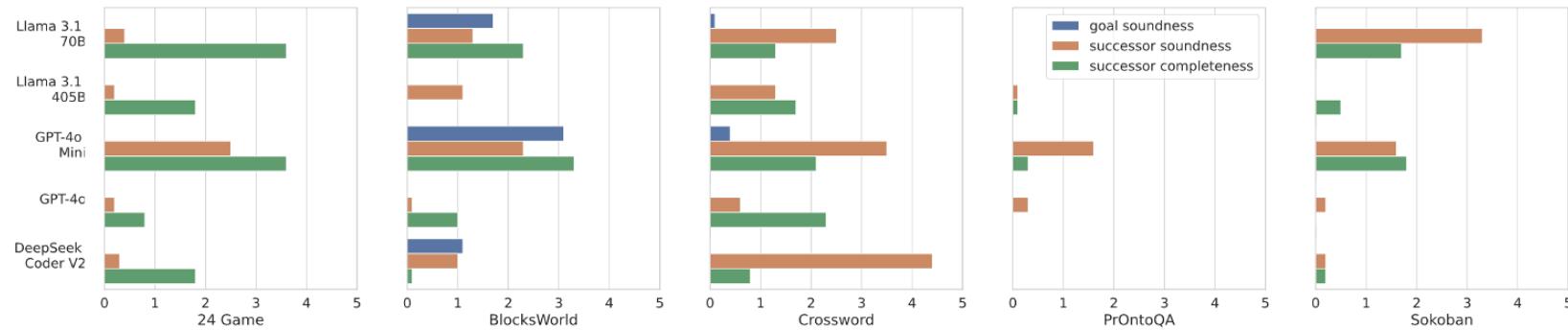
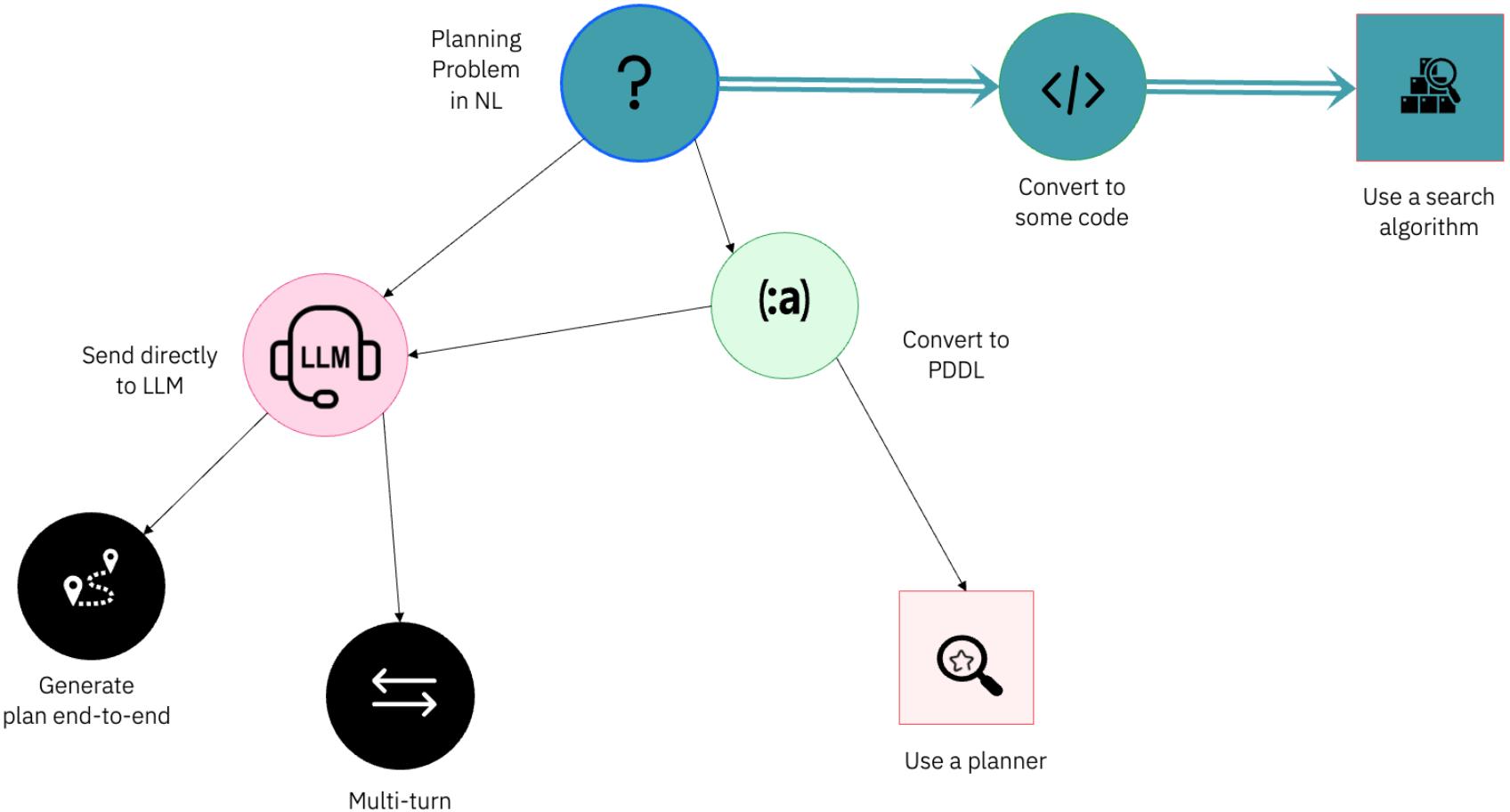


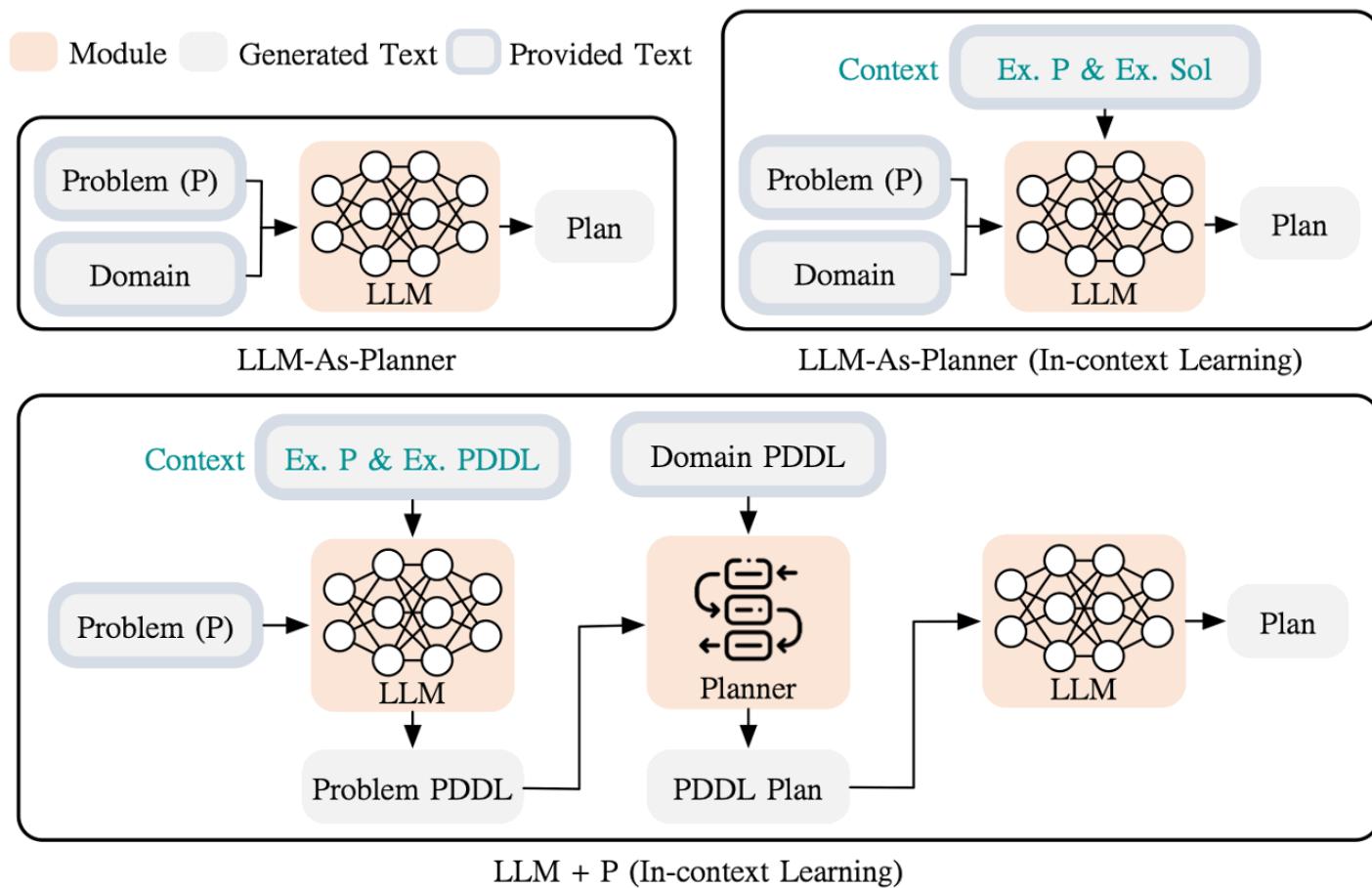
Figure 3: Average number of feedback calls for goal soundness, successor soundness, and successor completeness.



# LLMs for extracting action models

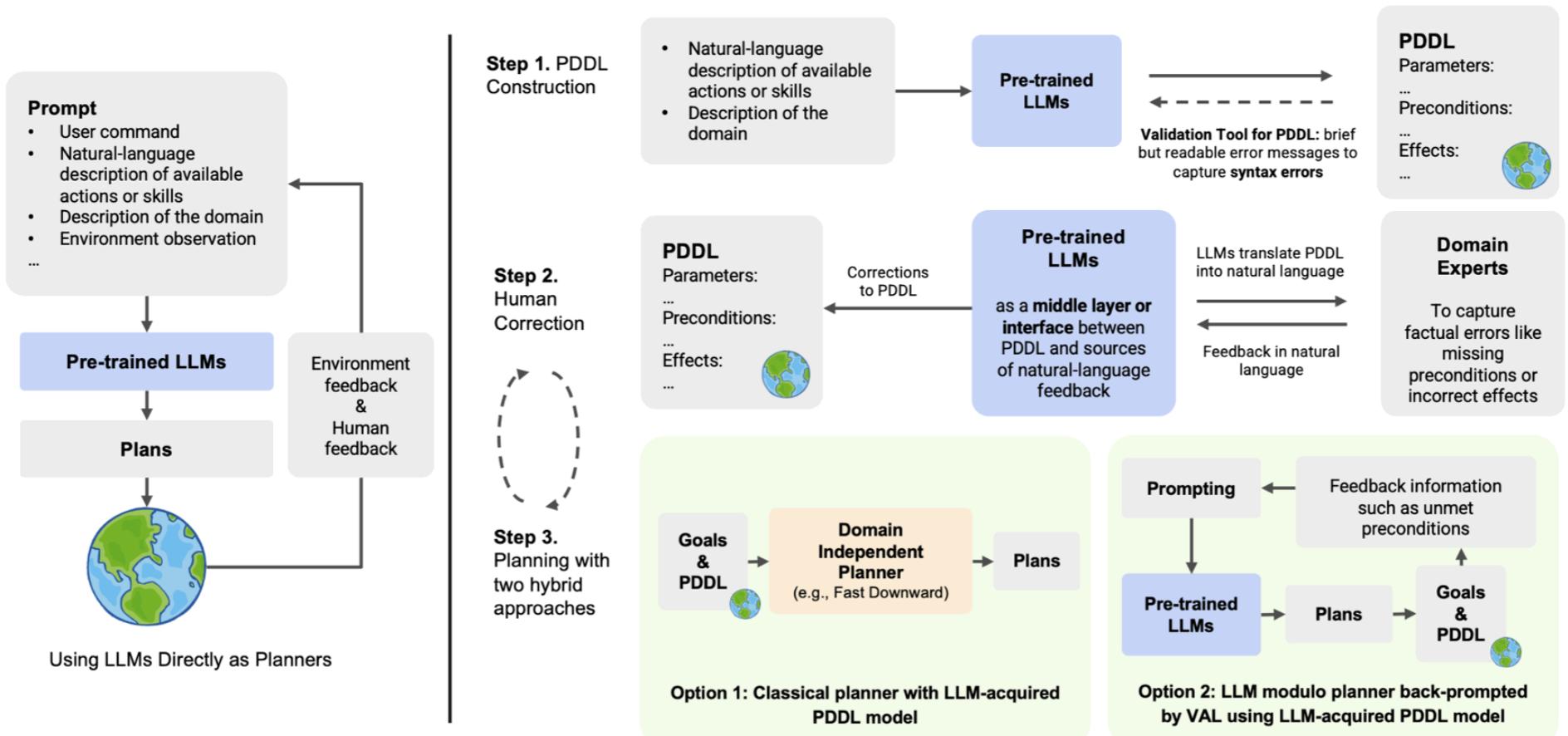


# LLM+P

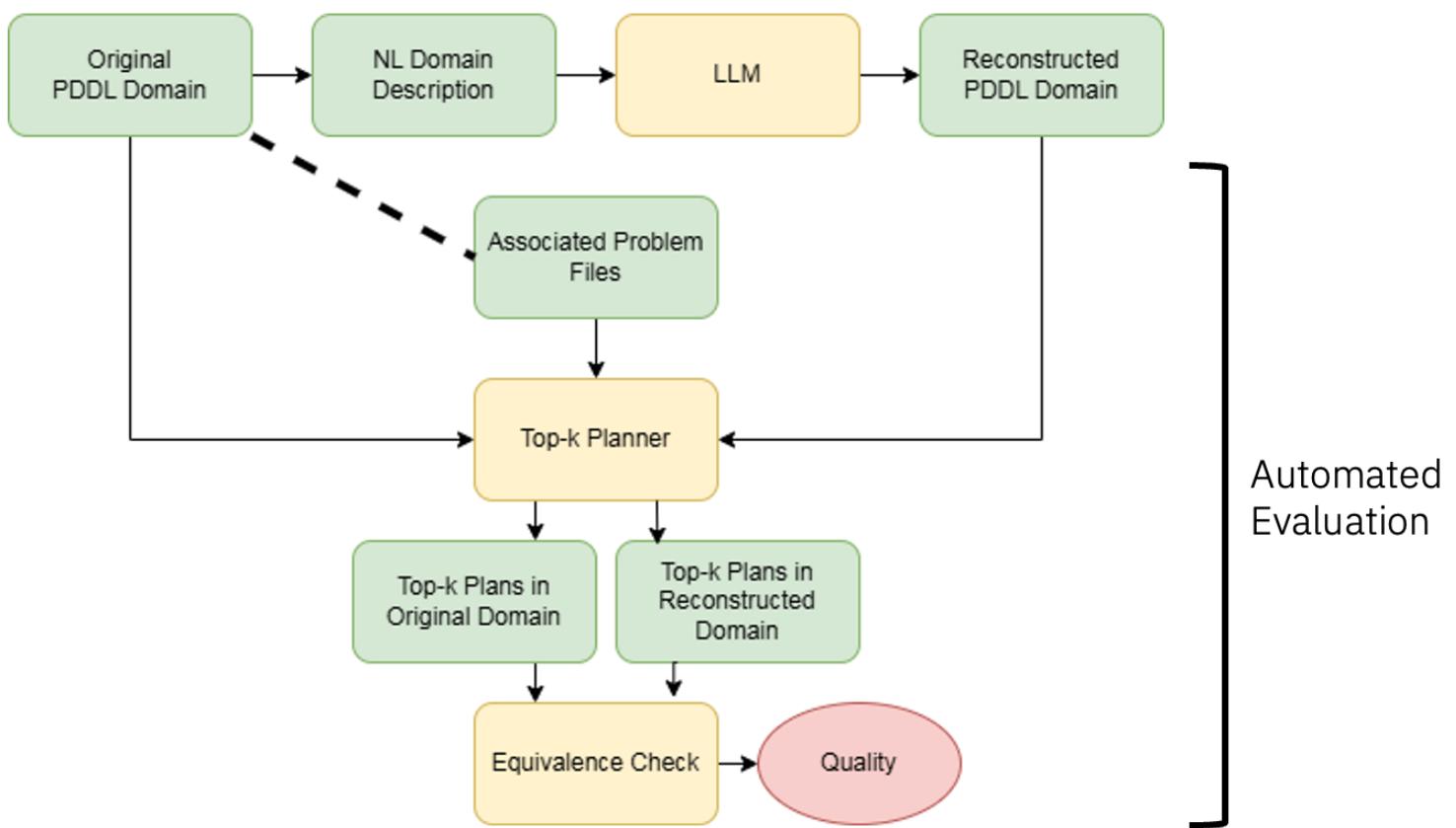


Liu et al 2023

# LLM-DM



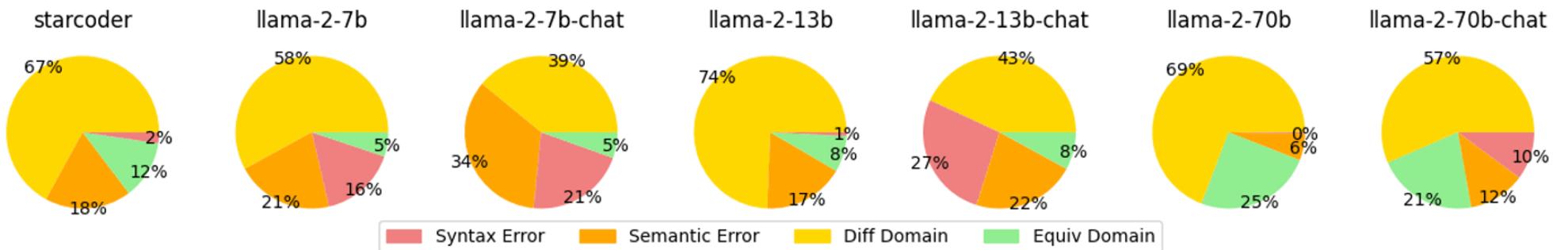
# NL2PDDL



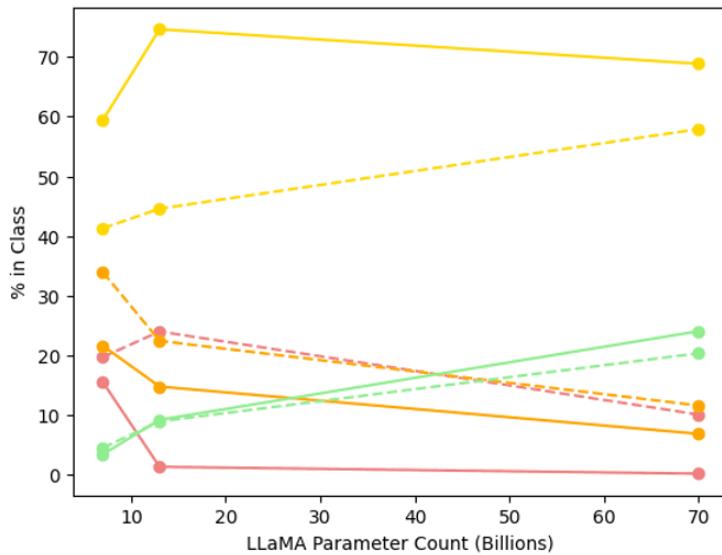
# NL2PDDL

## Error/Result Classes

- **Syntax Error:** The model produced syntactically invalid PDDL.
- **Semantic Error:** The model produced syntactically valid PDDL, but the PDDL doesn't integrate with the intended problems due to type mismatches, wrong number of parameters to predicates/actions, etc.
- **Different Domain:** The model produced syntactically valid PDDL that integrates with the original domain, but the underlying domains are different because they produce different plans for the same problem.
- **(Heuristically) Equivalent Domain:** The model produced syntactically valid PDDL that integrates with the desired domain. Plans from the original domain can be applied in the new domain and vice versa.



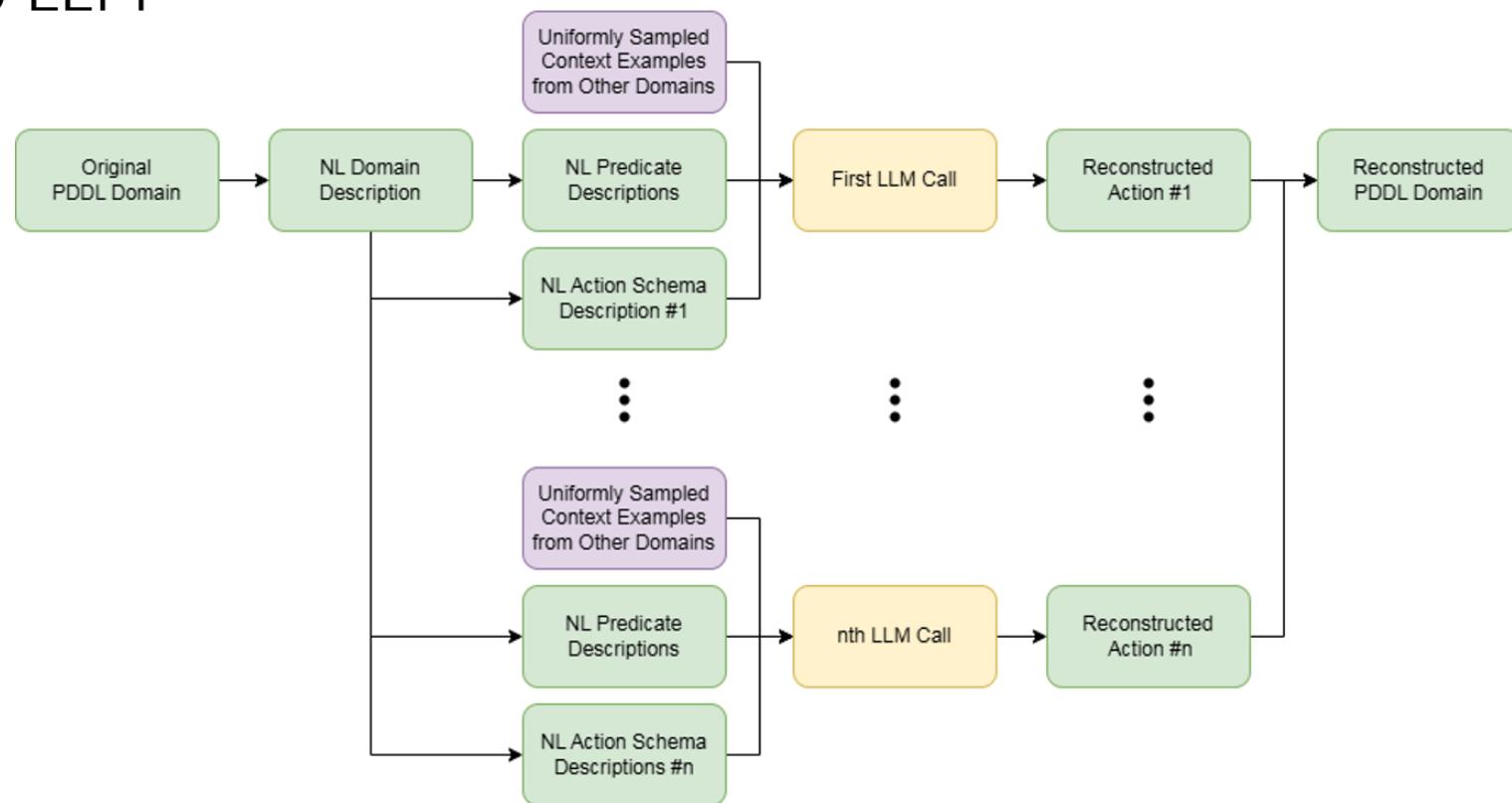
- ▶ Blocksworld : 5 preds 4 actions
  - ▶ Depot : 6 preds 5 actions
  - ▶ Forest : 5 preds 2 actions
  - ▶ Logistics : 3 preds 6 actions
  - ▶ Miconic : 6 preds 4 actions
  - ▶ TrapNewspapers : 7 preds 3 actions
  - ▶ Heavy-pack\* : 5 preds 2 actions
  - ▶ Trackbuilding\* : 4 preds 3 actions
- \* : Domain is not present in training data.



Error Type	Legend
Syntax Error	● - Solid line
Syntax Error	● - Dashed line
Semantic Error	○ - Solid line
Semantic Error	○ - Dashed line
Diff Domain	■ - Solid line
Diff Domain	■ - Dashed line
Equiv Domain	▲ - Solid line
Equiv Domain	▲ - Dashed line

# NL2PDDL

## Call to LLM



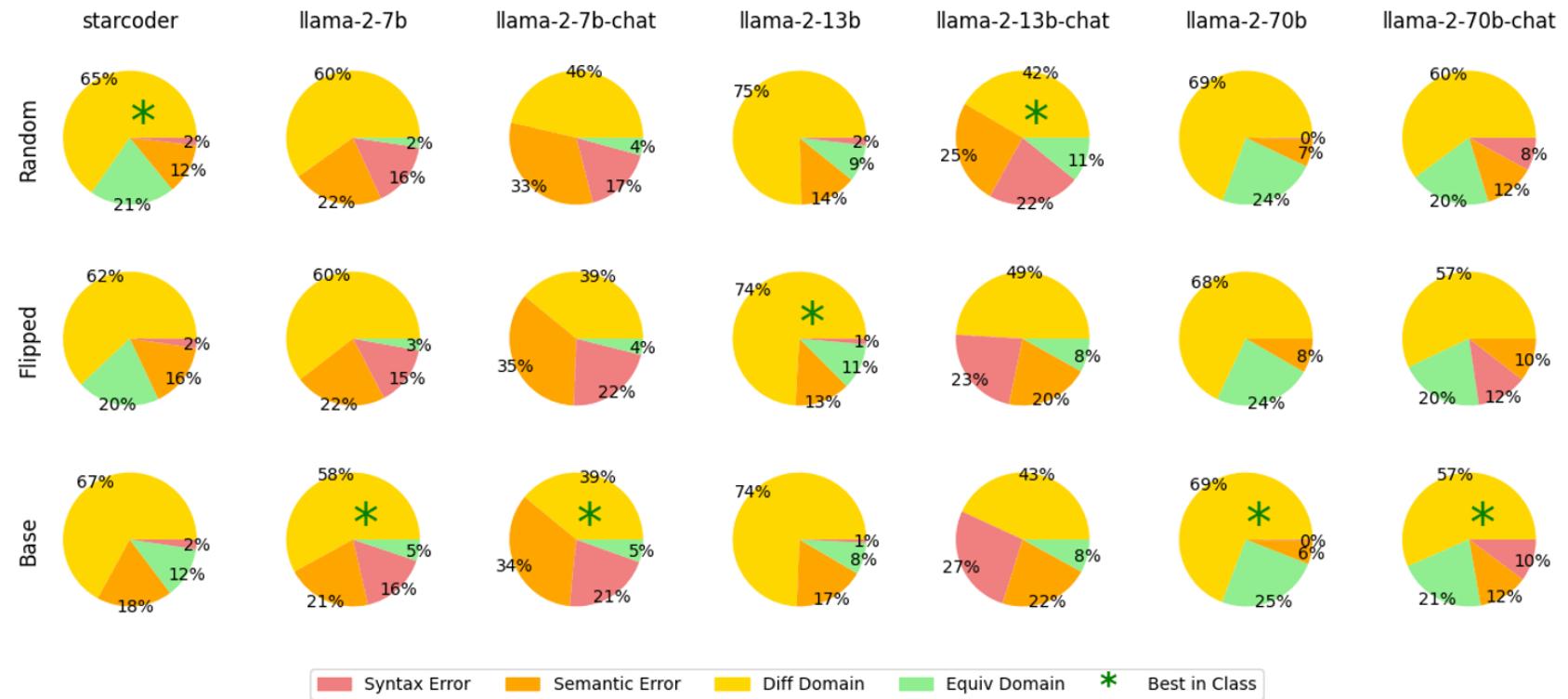
# NL2PDDL

## Prompt Variants

- ▶ **Base:** “*The action ‘unstack’ will have a hand unstack a block x from a block y.*”
- ▶ **Flipped:** “*The action ‘unstack’ will have a hand unstack a block x from a block y, if the block x is clear, x is on y, and the hand is empty.*”
- ▶ **Random:** “*The action ‘unstack’ will have a hand unstack a block x from a block y, if the hand is empty and x is on y. After the action, y should be clear.*”

# NL2PDDL

## Prompt Variants



# NL2PDDL

## Sample Prompt

### Instruction

Given a description of an action in some domain, convert it to Planning Domain Definition Language (PDDL) action. You may only use the allowed predicates provided for each action.

### Context

Allowed Predicates:

(agent-at ?loc - location) : the agent is at the location loc.

{Remaining Predicates for context example 1}

Input:

The action, "build-track" will build track at a location.

PDDL Action:

```
(:action build-track
  :parameters (?loc - location)
  :precondition (agent-at ?loc)
  :effect (and (has-track ?loc))
)
```

{2 Other Context Examples from different domains}

### Task

Allowed Predicates:

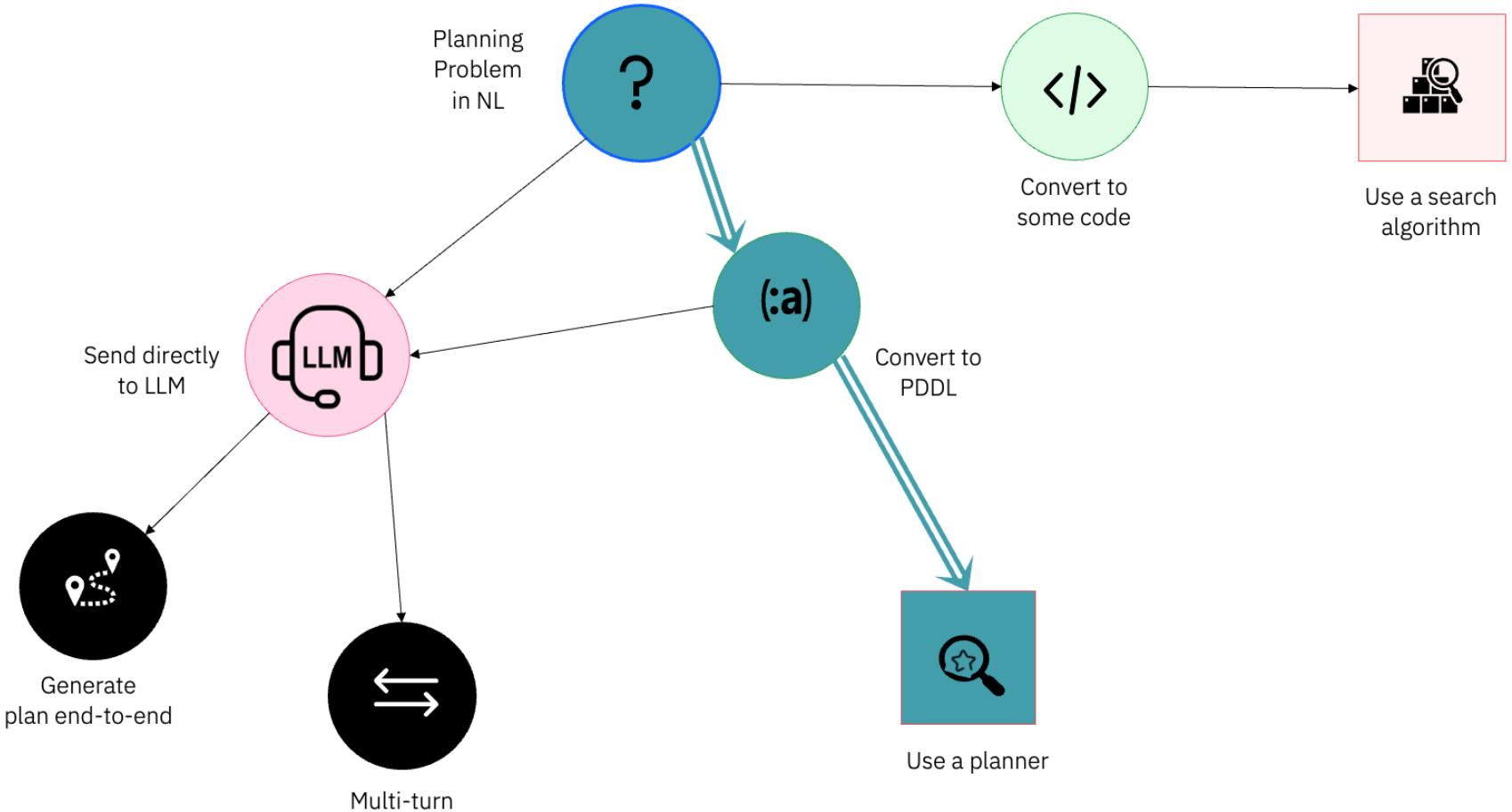
(at ?x - locatable ?y - place) : the locatable x is at some place y.

{Remaining Predicates for Task}

Input:

The action, "Unload" will use a hoist to unload a crate from a truck at a place.

PDDL Action:

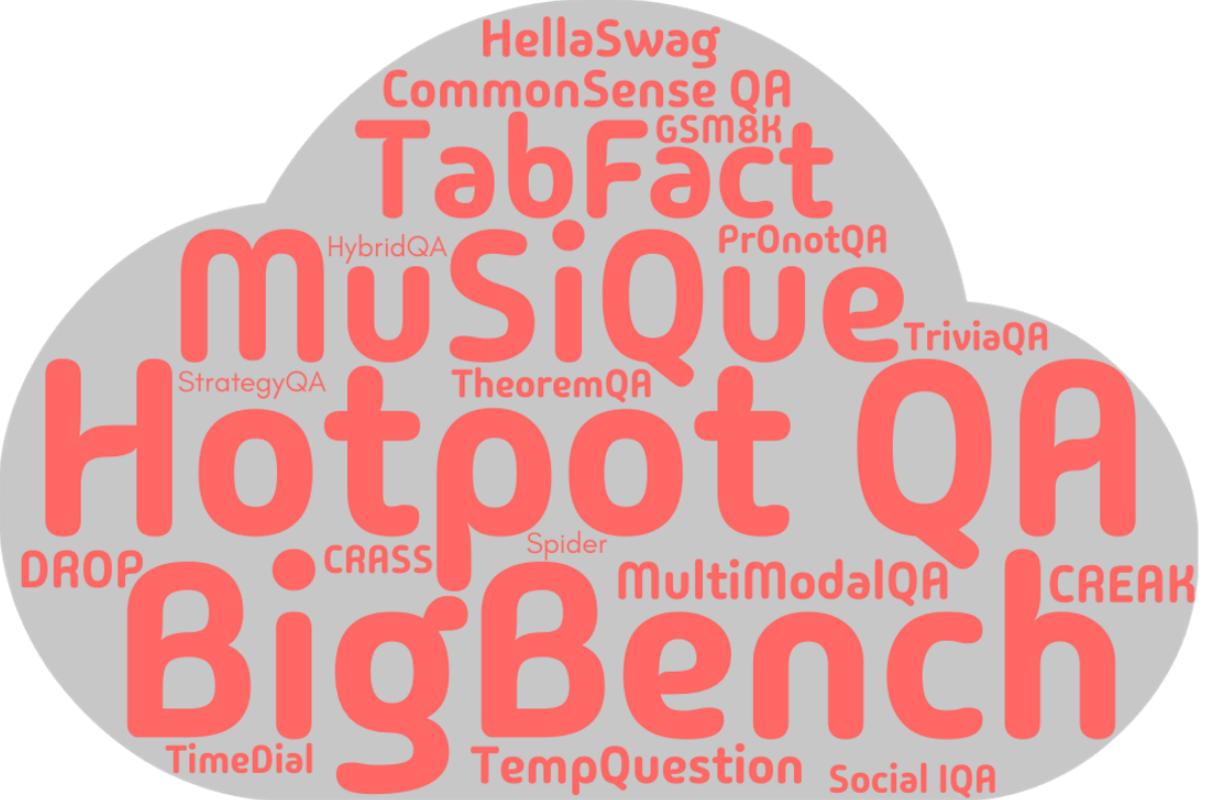


# Outline

- ✓ AI Planning: Introduction
- ✓ LLMs in AI Planning
  - ✓ LLMs as Planners
  - ✓ LLMs for generating code
  - ✓ LLMs for extracting action models
- Benchmarking and Evaluations

# Benchmarking and Evaluation





<b>Model</b>	<b>Citation</b>	<b>Domains</b>	<b>Tasks</b>	<b>LLMs Supported</b>	<b>Prompt</b>
PlanBench	Valmeekam et al. NeurIPS 23	2	8	OpenAI Bloom	[STATEMENT] [PLAN] [PLAN END]
AutoPlanBench	Stein et al. 23	12	1	OpenAI	Input: Model:
TRAC	He et al. ACL 23	1	4	T5 OpenAI Roberta	-
LLM+P	Liu et al. 23	7	1	OpenAI	An example planning problem is: .... A plan for the example problem is... Can you provide an optimal plan ...

<b>Model</b>	<b>Citation</b>	<b>Domains</b>	<b>Tasks</b>	<b>LLMs Supported</b>	<b>Prompt</b>
PlanBench	Valmeekam et al. NeurIPS 23	2	8	OpenAI Bloom	[STATEMENT] [PLAN] [PLAN END]
AutoPlanBench	Stein et al. 23	12	1	OpenAI	Input: Model:
TRAC	He et al. ACL 23	1	4	T5 OpenAI Roberta	-
LLM+P	Liu et al. 23	7	1	OpenAI	An example planning problem is: .... A plan for the example problem is... Can you provide an optimal plan ...
ACP Bench (ours)	Kokel et al. AAAI 25	13	7	Any	Any

The diagram illustrates the following relationships:

- Landmarks** is connected to **Action Reachability** and **Action Applicability**.
- Action Reachability** and **Action Applicability** both point to **Reachability**.
- Reachability** points to **Progression**.
- Progression** points to **Validation**.

# 7 reasoning tasks

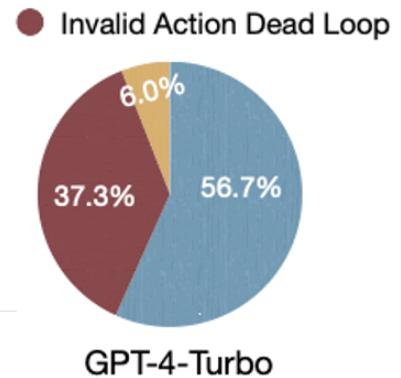
Ferry  
Logistics  
Swap  
Depot  
Gripper  
AlfWorld  
Rovers  
Flootile  
BlocksWorld  
VisitAll  
Goldminer  
GridSatellite

## 13 Domains\*

\* can add more domains in a day

# Action Applicability

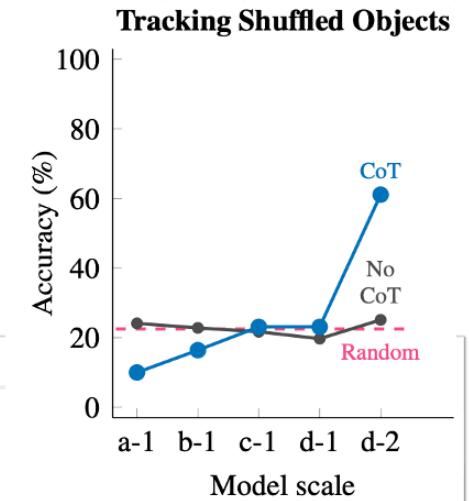
```
{  
  "id": 2606453784296512791,  
  "group": "applicable_actions_bool",  
  "context": "This is a ferry domain, where the task is to transport cars from their start to their goal locations, using a ferry. Each location is accessible by ferry from each other location. The cars can be debarked or boarded, and the ferry can carry only one car at a time. There are 3 locations and 10 cars, numbered consecutively. Currently, the ferry is at l1, with the car c2 on board. The cars are at locations as follows: c6, c3, and c0 are at l2; c4, c9, and c7 are at l0; c1, c8, and c5 are at l1.",  
  "question": "Is the following action applicable in this state: debark the car c2 from the ferry to location l1?"  
}
```



Xie et al. TravelPlanner

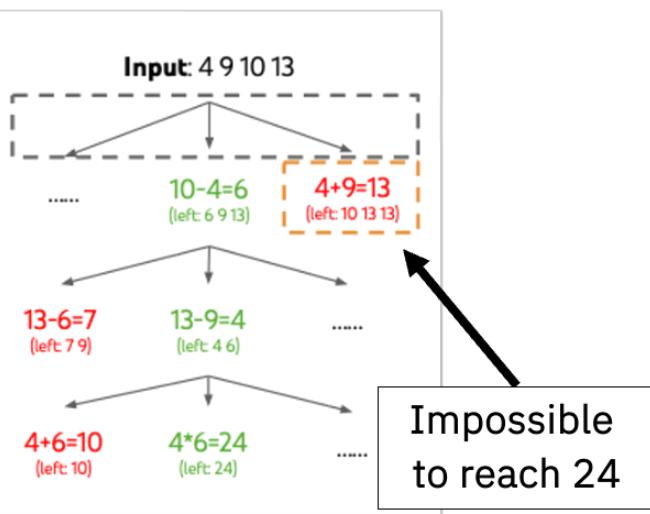
# Progression

Suzgun et al.  
BigBenchHard



```
"id": 2142145895175892935,  
"group": "progression_bool",  
"context": "This is a ferry domain, where the task is to transport cars from their start to their goal locations, using a ferry. Each location is accessible by ferry from each other location. The cars can be debarked or boarded, and the ferry can carry only one car at a time. There are 3 locations and 10 cars, numbered consecutively. Currently, the ferry is at l1 location and it is empty. The cars are at locations as follows: c9, c4, and c6 are at l0; c0, c8, c1, c7, and c2 are at l1; c3 and c5 are at l2.",  
"question": "Will the fact \\\"The ferry is empty\\\" hold after performing the action \\\"embark the car c0 at location l1 on to the ferry\\\" in the current state?"  
}
```

# Reachability



Yao et al. Tree of Thoughts

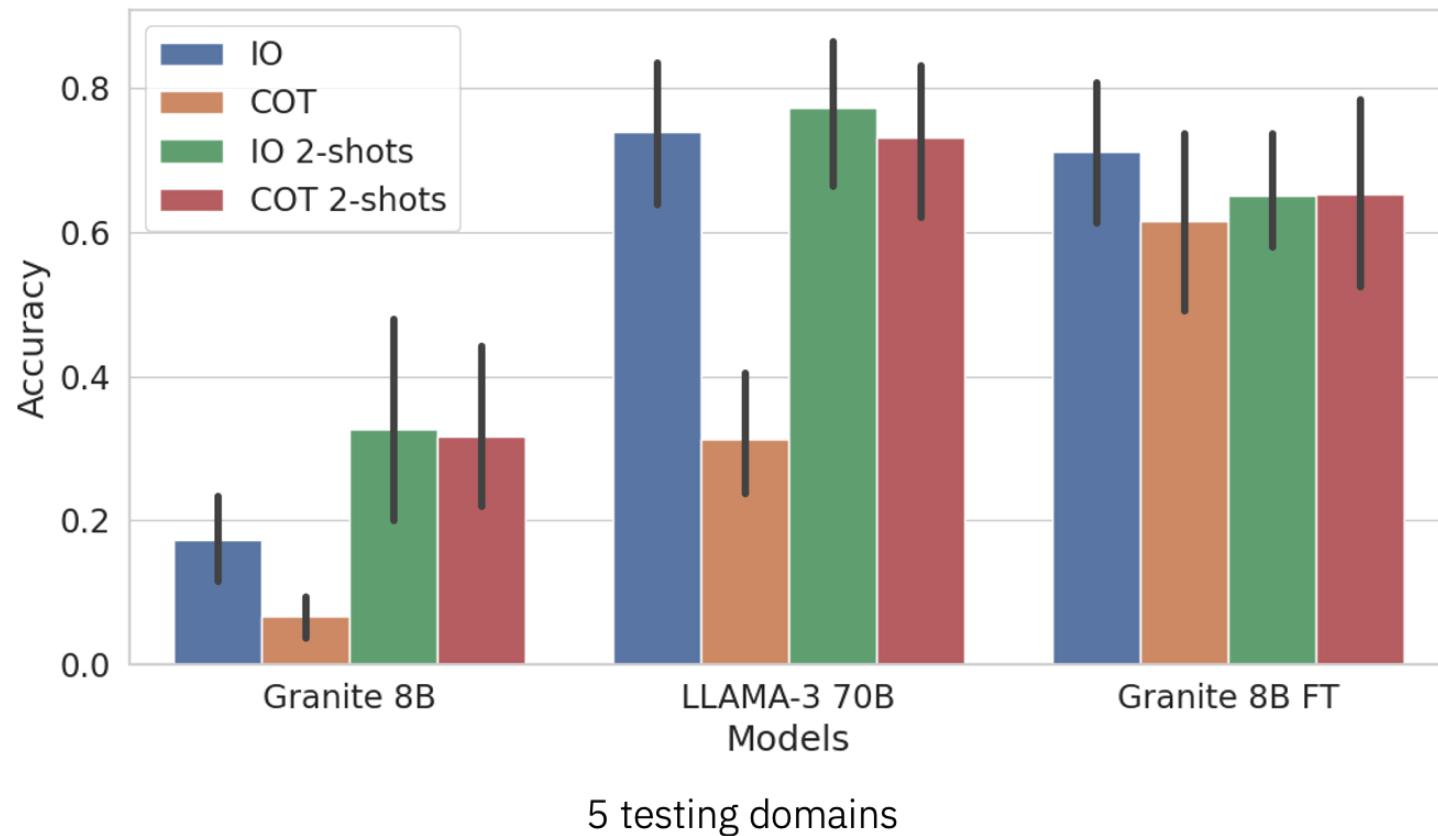
```
id": -3047022720995966296,  
group": "reachable_atom_bool",  
context": "This is a ferry domain, where the task is to transport cars from the start to their goal locations, using a ferry. Each location is accessible by ferry from each other location. The cars can be debarked or boarded, and the ferry can carry only one car at a time. There are 5 locations and 3 cars, numbered consecutively. Currently, the ferry is at l0, with the car c1 on board. The cars are at locations as follows: c0 and c2 are at l3.",  
question": "Is it possible to transition to a state where the following holds: Ferry is empty and The ferry is at l2 location?"
```

```
"choices": {  
    "text": [  
        "Car c29 is on board the ferry and The ferry is empty",  
        "Car c30 is at location c36",  
        "The ferry is at l3 location",  
        "The ferry is at c30 location and Car c2 is at location l1"  
    ],  
    "label": [  
        "A",  
        "B",  
        "C",  
        "D"  
    ]  
},  
"query": "Which fact is reachable from this state?"
```

Model	Applicability		Progression		Reachability		Validation		Action Reach.		Justification		Landmark		Mean	
	Bool	MCQ	Bool	MCQ	Bool	MCQ	Bool	MCQ	Bool	MCQ	Bool	MCQ	Bool	MCQ	Bool	MCQ
Phi-3 128K	66.15	33.08	68.46	53.85	52.31	26.15	50.77	19.23	53.33	32.50	49.23	33.85	49.23	46.92	55.53	34.75
Gemma 7B	63.23	28.62	64.92	31.08	53.08	23.08	46.92	20.0	55.67	34.50	50.77	36.46	27.54	30.31	51.80	28.93
Mistral 7B	61.54	32.31	73.08	38.46	53.08	28.46	47.85	17.69	<b>65.00</b>	19.17	48.46	30.00	35.38	33.08	55.00	28.67
Mistral I. 7B	63.08	31.54	61.54	46.92	61.54	33.08	52.15	36.15	45.83	34.17	43.08	29.23	57.69	50.77	55.45	37.30
Granite C. 8B	59.23	32.31	70.00	34.31	52.31	24.31	44.15	17.08	57.50	25.83	46.92	34.62	37.23	35.38	53.09	29.21
Granite 3.0 8B	72.31	26.92	73.08	53.85	53.08	24.62	53.08	20.00	45.83	30.83	49.23	34.62	42.31	34.62	55.56	32.21
Granite 3.0 I. 8B	76.92	30.00	73.85	57.69	53.08	36.92	55.38	34.62	58.33	44.17	<u>70.77</u>	31.54	51.54	43.08	62.84	39.72
LLAMA-3 8B	72.92	49.23	73.08	56.00	55.23	41.08	51.54	<u>49.23</u>	<u>63.50</u>	36.67	<u>57.54</u>	32.31	56.92	43.85	61.53	44.05
LLAMA-3.1 8B	65.38	56.92	63.85	47.69	53.08	33.85	60.00	<u>37.69</u>	42.50	28.33	46.92	45.38	33.85	40.00	51.46	41.52
Mixtral 8x7B	75.85	<u>57.69</u>	74.00	<u>61.38</u>	<u>76.00</u>	40.00	65.69	34.77	52.83	<u>55.00</u>	55.38	51.38	59.54	<u>60.00</u>	65.53	<u>51.44</u>
Codestral 22B	<u>84.62</u>	39.23	<u>83.85</u>	51.54	<u>54.62</u>	28.46	<u>66.15</u>	24.62	53.33	<u>38.33</u>	67.69	<u>62.31</u>	59.23	<u>42.31</u>	67.40	40.97
Mixtral 8x22B	<u>80.77</u>	37.69	72.31	54.62	50.00	<u>42.62</u>	37.69	16.92	58.50	27.83	43.08	<u>44.62</u>	44.77	45.23	55.63	39.25
Deepseek I. 33B	70.77	37.23	68.46	46.31	53.08	<u>31.69</u>	51.54	37.69	50.00	27.50	46.92	26.15	<u>62.31</u>	39.23	57.58	35.11
LLAMA C. 34B	80.77	42.31	73.08	43.85	53.08	25.69	50.15	28.46	53.17	33.33	55.38	35.38	<u>46.92</u>	40.62	59.02	35.71
LLAMA-2 70B	78.46	24.62	71.54	<u>36.77</u>	53.08	26.92	51.38	16.15	60.83	22.00	49.23	55.54	24.46	26.00	55.72	29.71
LLAMA C. 70B	74.77	36.15	54.77	52.92	48.62	23.69	40.0	17.69	49.67	28.83	46.92	31.54	37.08	42.31	50.90	32.87
LLAMA-3 70B	90.77	82.31	93.08	86.15	<b>87.69</b>	<b>82.31</b>	<b>78.62</b>	56.62	60.50	63.00	62.31	<u>85.38</u>	78.15	64.77	78.71	74.30
LLAMA-3.1 70B	93.08	84.31	89.85	86.77	61.38	54.92	66.15	46.62	63.00	58.00	56.92	68.46	34.62	<u>69.23</u>	66.67	66.94
LLAMA-3.1 405B	<u>95.38</u>	86.92	<u>93.08</u>	<b>93.85</b>	59.23	<u>80.77</u>	<u>77.23</u>	<b>62.92</b>	<b>65.00</b>	<b>65.00</b>	<b>90.00</b>	<b>86.92</b>	83.08	65.38	<u>80.49</u>	<b>77.42</b>
GPT-4o Mini	90.77	73.85	<b>95.38</b>	79.23	80.77	39.23	67.69	46.15	54.17	21.67	77.69	70.00	76.92	67.69	77.74	56.50
GPT-4o	<b>96.92</b>	<b>89.23</b>	94.62	<u>90.00</u>	79.23	76.92	61.54	53.85	57.50	52.50	<u>88.46</u>	80.77	<b>95.38</b>	<b>79.23</b>	<b>81.84</b>	74.97

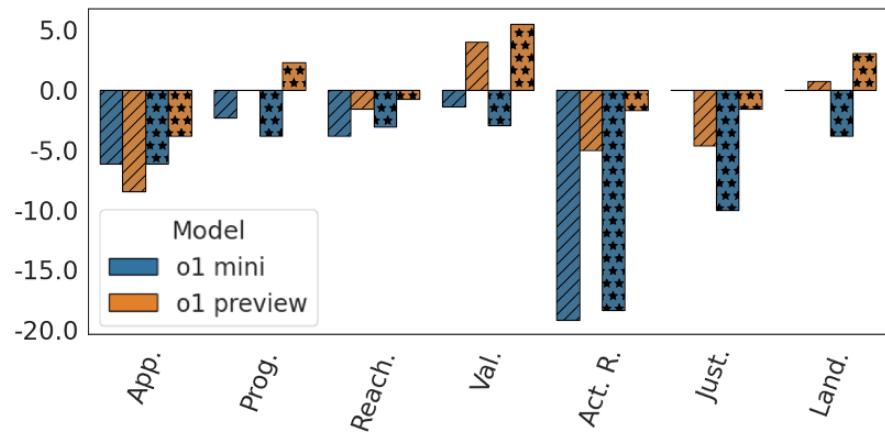
Table 2: Accuracy of 21 LLMs, (I)nstruct and (C)ode models, on 7 ACPBench tasks (boolean and multi-choice). The best results are **boldfaced**, second best are underlined, and the best among the small, open-sourced models are double underlined. All models were evaluated with two in-context examples and COT prompt. The right-most column is mean across tasks.

## Prompts and Supervised Finetuning

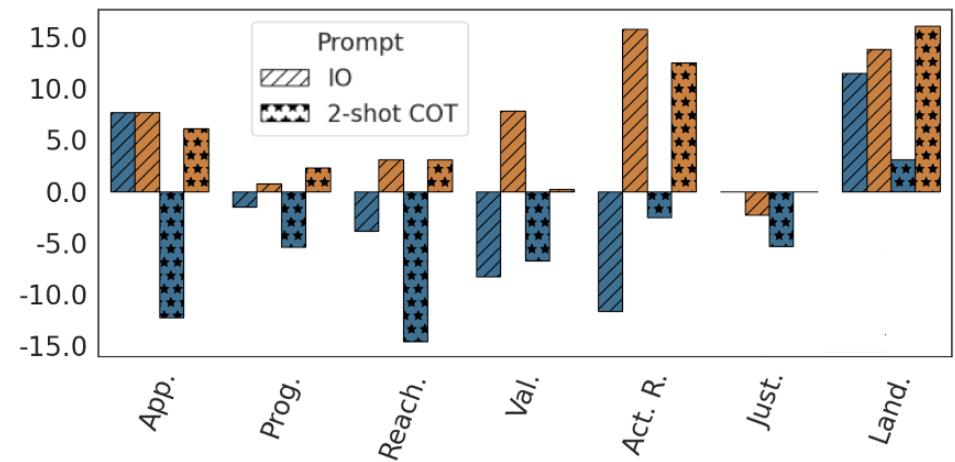


# O1

## Reasoning Model



Bool



MCQ

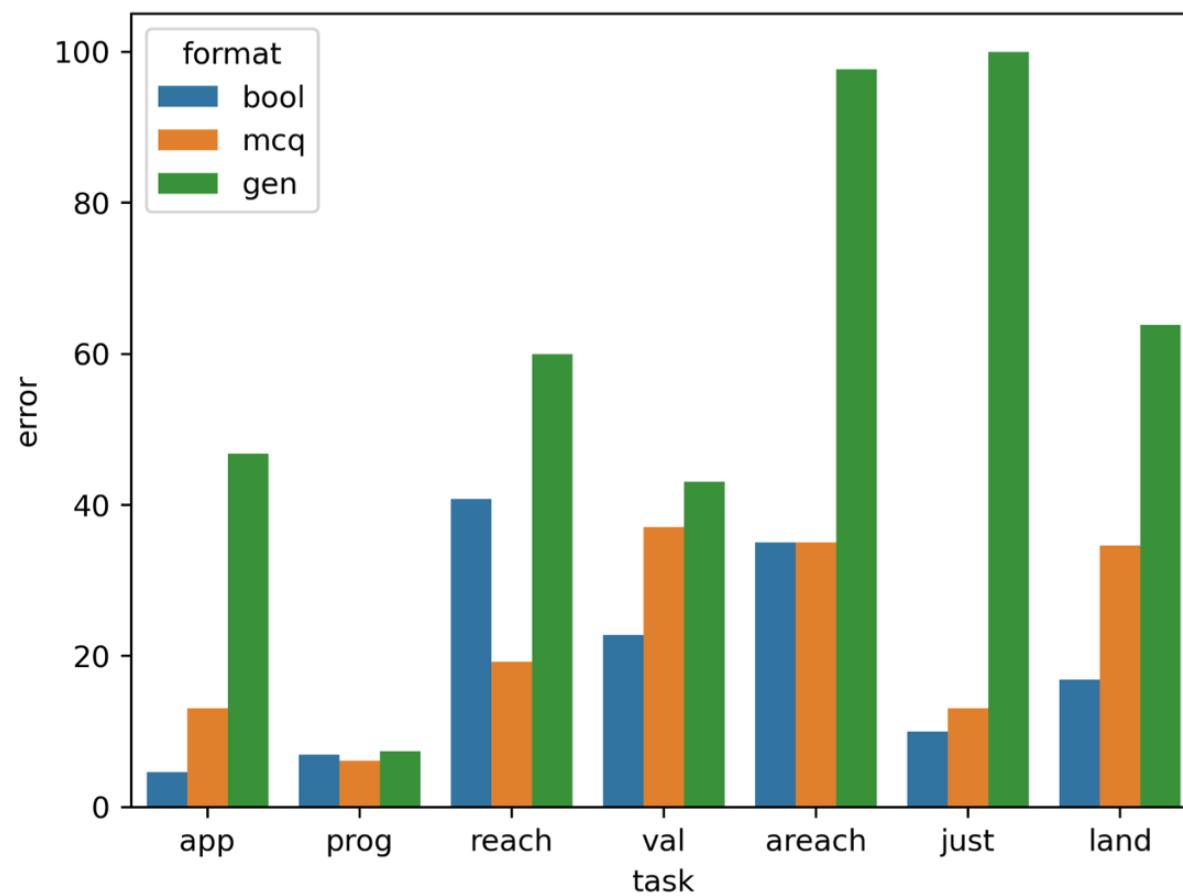
Benchmark	PlanBench	Auto PlanBench	TRAC	LLM+P	ActionReasoning Bench	ACPBench
LLM supported	OpenAI Bloom	OpenAI	T5 OpenAI Roberta	OpenAI	huggingface	Huggingface OpenAI vLLMs, etc
Prompts	[STATEMENT] [PLAN] [PLAN END]	Input: Model:	None  tokenize(s" "a) + tokenize(q)	An example planning problem is: .... A plan for the example problem is... Can you provide an optimal plan ...	[DOMAIN DESCRIPTION] [INITIAL CONDITION] [QUESTION]	**Question**: **Thoughts**: **Final Answer**:

# ACPBench Hard



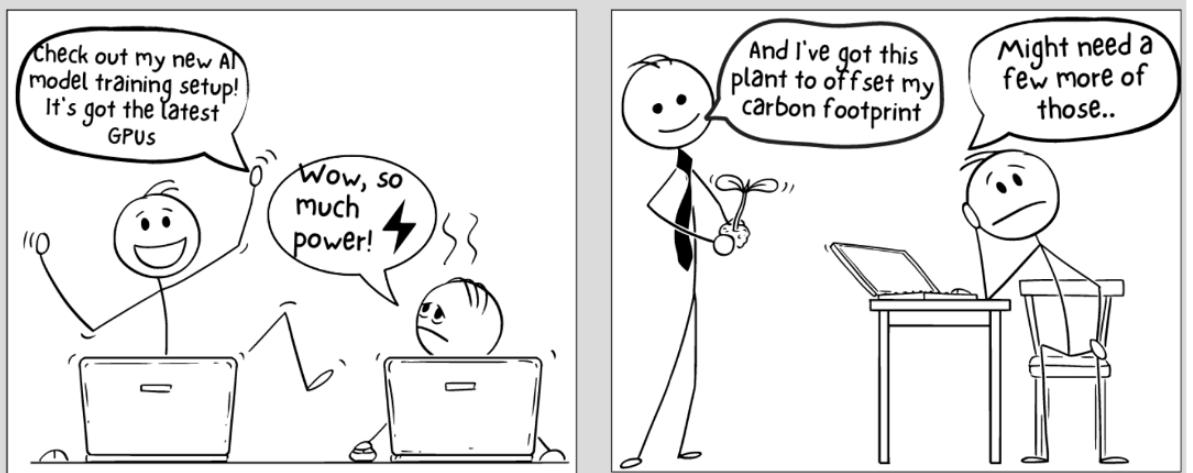
Will the fact \"Car c4 is on the ferry\"  
hold after performing the action  
\"sail from location l0 to location l1\"  
in the current state?"

Break down the outcomes of performing the action  
\"sail from location l4 to location l0\" into two lists,  
positive effects and negative effects. Positive effects  
are the propositions that are false in the current state  
but will become true after performing the action.  
Negative effects are the propositions that are true in  
the current state and will become false after performing  
the action. Provide only the two lists with the ground  
propositions.



# Questions ?

# The Carbon Impact of Large Language Models: AI's Growing Environmental Cost



# References

- Katz, Michael, et al. "Thought of search: Planning with language models through the lens of efficiency." NeurIPS 2024
- Cao, Daniel, et al. "Automating Thought of Search: A Journey Towards Soundness and Completeness. (Student Abstract)" AAAI 2025
- Oswald, James, et al. "Large language models as planning domain generators." ICAPS 2024.
- Kokel, Harsha, et al. "ACPBench: Reasoning about Action, Change, and Planning." AAAI 2025.
- Kokel, Harsha, et al. "ACPBench Hard: Unrestrained Reasoning about Action, Change, and Planning." AAAI 2025 Workshop LM4Plan 2025.
- Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." NeurIPS 2022.
- Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models." NeurIPS 2023.
- Yao, Shunyu, et al. "React: Synergizing reasoning and acting in language models." ICLR 2023.
- Xu, Binfeng, et al. "Rewoo: Decoupling reasoning from observations for efficient augmented language models." arXiv:2305.18323 2023.

## References

- Kambhampati, Subbarao, et al. "Position: LLMs can't plan, but can help planning in LLM-modulo frameworks." ICML 2024.
- Huang, Wenlong, et al. "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents." ICML 2022.
- Team, Gemini, et al. "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context." arXiv:2403.05530 2024.
- Valmeekam, Karthik, et al. "LLMs Still Can't Plan; Can LRM? A Preliminary Evaluation of OpenAI's o1 on PlanBench." NeurIPS 2024 Workshop on Open-World Agents.
- Hao, Shibo, et al. "Reasoning with Language Model is Planning with World Model." EMNLP 2023.
- Besta, Maciej, et al. "Graph of thoughts: Solving elaborate problems with large language models." AAAI 2024.
- Zhou, Andy, et al. "Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models." ICML 2024.
- Liu, Bo, et al. "LLM+P: Empowering large language models with optimal planning proficiency." arXiv preprint arXiv:2304.11477 2023.

## References

- He, Weinan, et al. "Exploring the capacity of pretrained language models for reasoning about actions and change." ACL 2023.
- Handa, Divij, et al. "ActionReasoningBench: Reasoning about Actions with and without Ramification Constraints." ICLR 2025.
- Guan, Lin, et al. "Leveraging pre-trained large language models to construct and utilize world models for model-based task planning." NeurIPS 2023