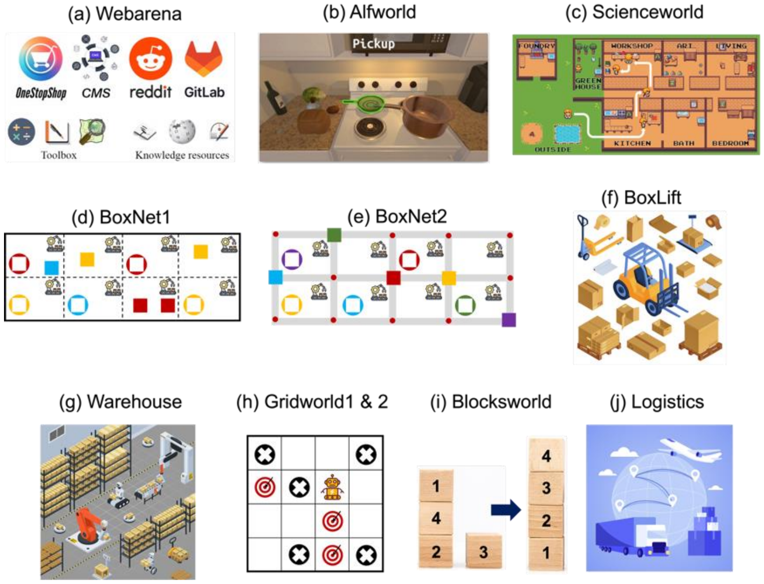
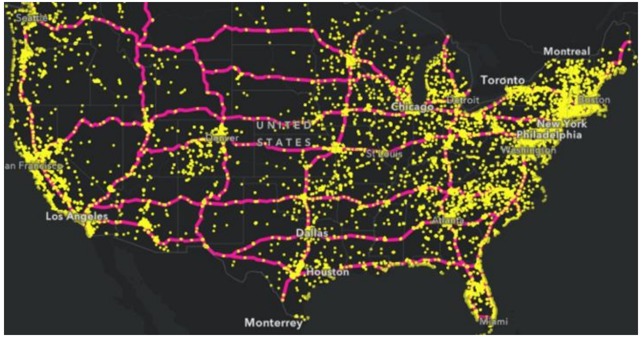
Large Language Models Can Solve Real-World Planning Rigorously

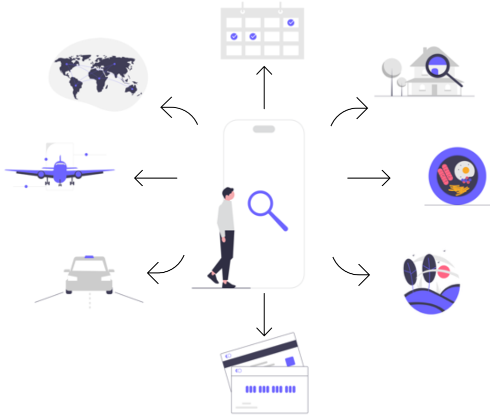
with formal reasoning tools

Chuchu Fan
chuchu@mit.edu
Associate Professor of MIT AeroAstro
Director of REliable Autonomous systems Lab at MIT (REALM)

Planning with constraints: NP-hard problem and domain-specific tools





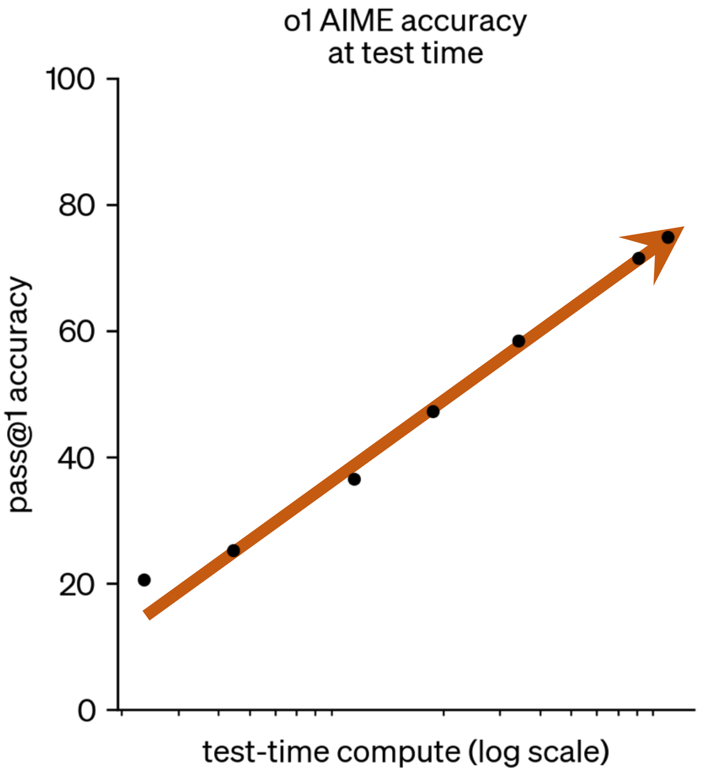


“The Bitter Lesson” from Rich Sutton

One thing that should be learned from the bitter lesson is the great power of ***general purpose methods***... The two methods that seem to scale arbitrarily ***with increased computation*** are ***search*** and ***learning***.

Rich Sutton, The Bitter Lesson, March 13, 2019.

Search by long-chain reasoning: slow thinking



Inference-time textual reasoning to search over answer space.

Long-chain reasoning **cannot search efficiently**

Question: Use the numbers 1, 3, 13, and 13 to make 24.

Okay, let's see. I need to use the numbers 1, 3, 13, and 13 to make 24. Each number has to be used exactly once. Hmm. Let's start by listing out all the possible operations and combinations.

First, the numbers are 1, 3, 13, 13. Wait, there are two 13s. Oh, but the problem says each number has to be used once. But since both 13s are the same, maybe that gives some flexibility. But I need to use both of them, right?

So let's think. Maybe I can combine the 13s in some way. Like 13 divided by 13 is 1. But then I already have a 1. Hmm. So if I do 13/13 =1, then I have 1 and 1 and 3. But then how to get 24 from 1, 1, 3. That might not help. Maybe subtract them? 13 -13 =0, which probably isn't useful here.

Alternatively, adding them: 13 +13 =26. Then I have 26, 3, 1. How to get from 26,3,1 to 24. 26 - (3 -1) =24? Let's check: 26 -2 =24. But that would require 26 - (3-1) =24. So that would be (13+13) - (3 -1) =24. Let me verify: 13+13 is 26. 3-1 is 2. 26-2 is 24. Yes! But wait, have I used all the numbers? Let's check. 13,13,3,1. Yes. Each is used once. So the equation would be (13 +13) - (3 -1) =24. Let me write that with parentheses correctly: ((13 +13) - (3 -1)) =24. Alternatively, it's (13+13)-(3-1) =24. That works. Let me check again: 13+13=26, 3-1=2, 26-2=24. Yep. So that's a possible solution. Let me see if there are other ways.

Alternatively, maybe using multiplication. Let's see: 3 * something. For example, 3*8=24. So if I can make 8 with the remaining numbers 1,13,13. Let's see. 13 - (13/1) =0. Not helpful. 13 - 1=12. 12 +13=25. Not 8. Hmm. Alternatively, 13 - (13 -1) =1. Still not 8.

What about 13 divided by 13 is 1, then 3-1=2. Then 1*2=2. No, that's not enough. Maybe 3*(1 + ...). Let me think. 3*(8) =24. So how to get 8. Let's see. Maybe 13 - (13 - (1 + ...)). Hmm. Not sure. Let's try another approach.

What if I do 13 * (something) to get close to 24. 13*2=26. 26 - 2=24. But where does 2 come from? 3-1=2. So 13*2 -2=24. Wait, but how to structure that. Let's see: 13*(3-1) - (13 - ?). Wait, maybe not. Let's see. If I use 3-1=2, then 13*2=26. Then 26 - (13 - x) where x is the remaining number. But the remaining number is already used? Wait, perhaps not. Let's re-examine. If I use 13*(3-1), that uses 13,3,1. Then subtract the other 13? So 13*(3-1) -13 =26-13=13. Not 24. Hmm.

Alternatively, 13*(something) divided by something. Let me think. 13 * (3) is 39. Then 39 - (13 +1) =25. Not 24. Close. 13*3 -13 -1 =39-14=25. Not enough. Hmm.

Wait, another idea: 13 + 13 =26. Then subtract 3 and add 1. 26-3+1=24. So that's 26-3+1=24. But how to arrange that. The expression would be (13+13) -3 +1. That uses all numbers: 13,13,3,1. Each once. So (13+13) -3 +1 =24. Let's compute: 26-3 is 23, plus 1 is 24. Yes. So that works. So another possible solution is 13+13-3+1=24. But the problem is that sometimes the order of operations might require parentheses. Let's check: 13+13 is 26, minus 3 is 23, plus 1 is 24. So the order is left to right, so no parentheses needed. So 13+13-3+1=24. But does that fit the required format? The previous examples used parentheses, but maybe it's allowed. Let me check the previous answers. For example,...

Long-chain reasoning **cannot search efficiently**

Question: Use the numbers 1, 3, 13, and 13 to make 24.

Okay, let's see. I need to use the numbers 1, 3, 13, and 13 to make 24. Each number has to be used exactly once. Hmm. Let's start by listing out all the possible operations and combinations.

First, the numbers are 1, 3, 13, 13. Wait, there are two 13s. Oh, but the problem says each number has to be used once. But since both 13s are the same, maybe that gives some flexibility. But I need to use both of them, right?

So let's think. Maybe I can combine the 13s in some way. Like 13 divided by 13 is 1 ... I already have a 1. Hmm. So if I do 13/13 =1, then I have 1 and 1 and 3. But then how to get 24 from 1, 1, 3. That might not help. Maybe subtract them? 13 -13 =0, which probab... ful here.

Alternatively, adding them: 13 +13 =26. Then I have 26, 3, 1. How to get from 26,3... 26 - (3-1) =24? Let's check: 26 -2 =24. But that would require 26 - (3-1) =24. So that would be (13+13) - (3 -1) =24. Let me verify: 13+13 is 26. 3-1 is 2. 26-2 is 24. Yes! But wait, have I used all the numbers? Let's check. 13,13,3,1. Yes. Each is used once. So the equation would be (13 +13) - (3 -1) =24. Let me write that with parentheses correctly: ((13 +13) - (3 -1)) =24. Alternatively, it's (13+13)-(3-1) =24. That works. Let me check again: 13+13=26, 3-1=2, 26-2=24. Yep. So that's a possible solution. ...

r1 long-chain reasoning fails and brings high costs

Alternatively, maybe using multiplication. Let's see: 3 * something. For example, 3*8=24. So if I can make 8 with the remaining numbers 1,13,13. Let's see. 13 - (13/1) =0. Not helpful. 13 - 1=12. 12 +13=25. Not 8. Hmm. Alternatively, 13 - (13 -1) =1. Still not 8.

What about 13 divided by 13 is 1, then 3-1=2. Then 1*2=2. No, that's not enough. Maybe 3*(1 + ...). Let me think. 3*(8) =24. So how to get 8. Let's see. Maybe 13 - (13 - (1 + ...)). Hmm. Not sure. Let's try another approach.

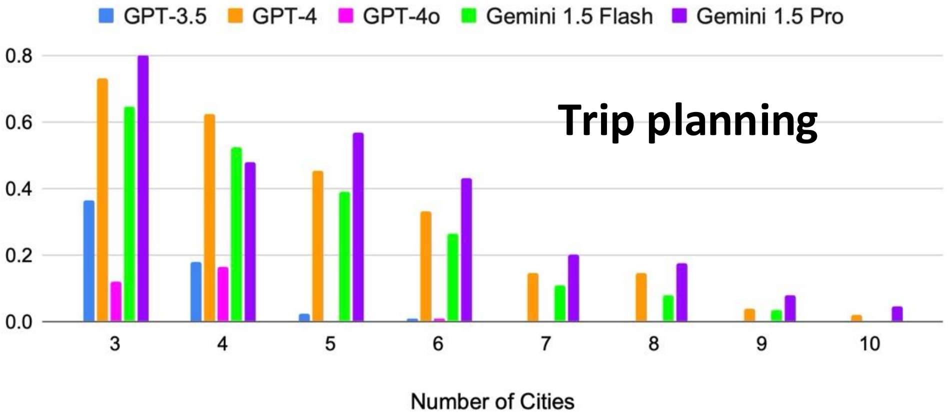
Search requires symbolic computing

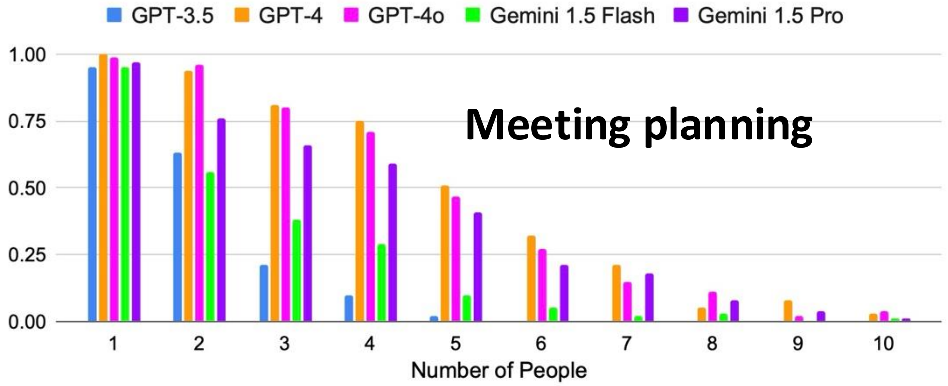
What if I do 13 * (something) to get close to 24. ... ture that. Let's see: 13*(3-1) - (13 - ?). Wait, maybe not. Let's see. If I use 3-1=2, then 13*2=26. Then 26 - (13 - x) where x is the remaining number. But the remaining number is already used? Wait, perhaps not. Let's re-examine. If I use 13*(3-1), that uses 13,3,1. Then subtract the other 13? So 13*(3-1) -13 =26-13=13. Not 24. Hmm.

Alternatively, 13*(something) divided by something. Let me think. 13 * (3) is 39. Then 39 - (13 +1) =25. Not 24. Close. 13*3 -13 -1 =39-14=25. Not enough. Hmm.

Wait, another idea: 13 + 13 =26. Then subtract 3 and add 1. 26-3+1=24. So that's 26-3+1=24. But how to arrange that. The expression would be (13+13) -3 +1. That uses all numbers: 13,13,3,1. Each once. So (13+13) -3 +1 =24. Let's compute: 26-3 is 23, plus 1 is 24. Yes. So that works. So another possible solution is 13+13-3+1=24. But the problem is that sometimes the order of operations might require parentheses. Let's check: 13+13 is 26, minus 3 is 23, plus 1 is 24. So the order is left to right, so no parentheses needed. So 13+13-3+1=24. But does that fit the required format? The previous examples used parentheses, but maybe it's allowed. Let me check the previous answers. For example,...

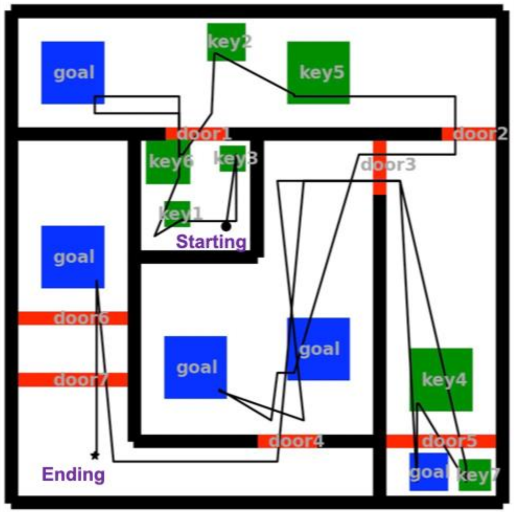
Why is symbolic planning necessary?





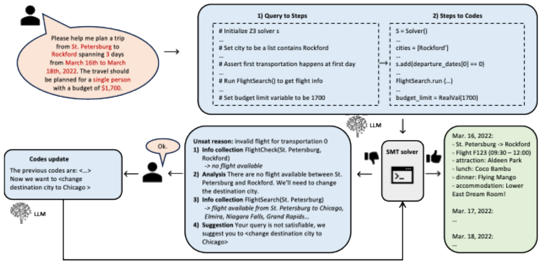
Task complexity can go beyond bounds, but textual scalability is limited.

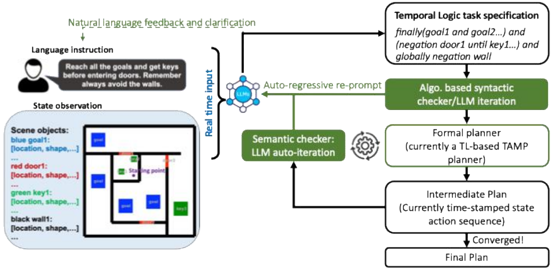
Robot planning (long-horizon)

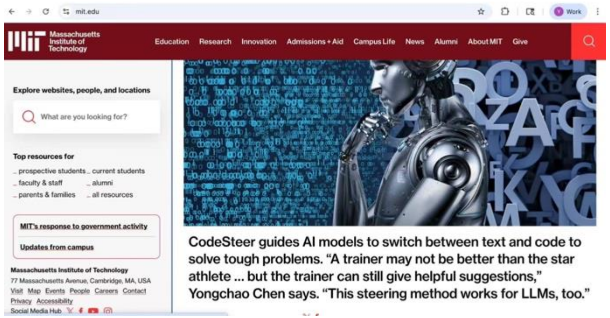


H. Zheng et al. NATURAL PLAN: Benchmarking LLMs on Natural Language Planning, arXiv 2024.

Outline



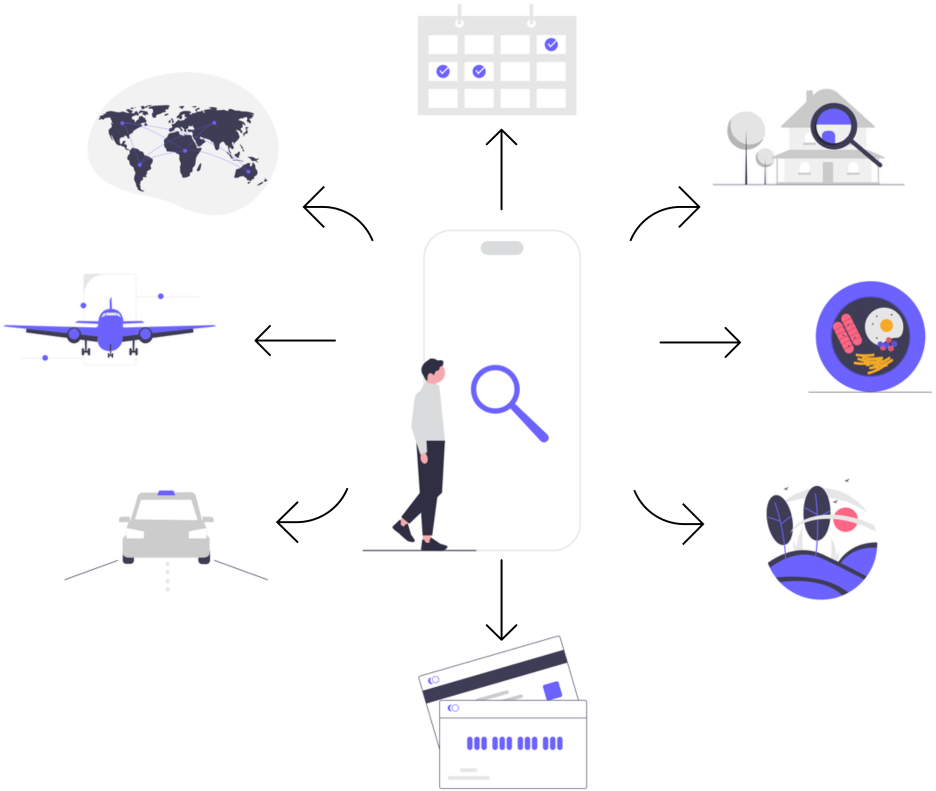




LLM + rigorous solver can make very complex plans [1,2] → LLM + rigorous solver can control dynamic robots [3] → Unifying code, text-based reasoning, and tool use in LLMs [4,5,6]

[1] Y. Hao, Y. Chen, Y. Zhang, and C. Fan. "Large Language Models can solve real-world planning rigorously with formal verification tools." NAACL (2024).
[2] Y. Hao, Y. Zhang, and C. Fan. "Planning anything with rigor: general-purpose zero-shot planning with LLM-based formalized programming." ICLR (2024).
[3] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan. "AutoTAMP: Autoregressive task and motion planning with LLMs as translators and checkers." ICRA (2024).
[4] Y. Chen, Y. Hao, Y. Zhang, and C. Fan. "Code-as-symbolic-planner: Foundation model-based robot planning via symbolic code generation." IROS (2025).
[5] Y. Chen, H. Jhamtani, S. Sharma, C. Fan, C. Wang. "Steering LLMs between Code Execution and Textual Reasoning." ICLR, 2025.
[6] Y. Chen, Y. Hao, Y. Liu, Y. Zhang, and C. Fan. "CodeSteer: Symbolic-Augmented Language Models via Code/Text Guidance." ICML, 2025.

Travel planning can be tedious; does LLM help?



- Complex, multi-constraint planning (e.g., making a travel plan) is complicated, time-consuming, and labor-intensive, even for human
- Can we plan complex travel needs with an LLM agent?

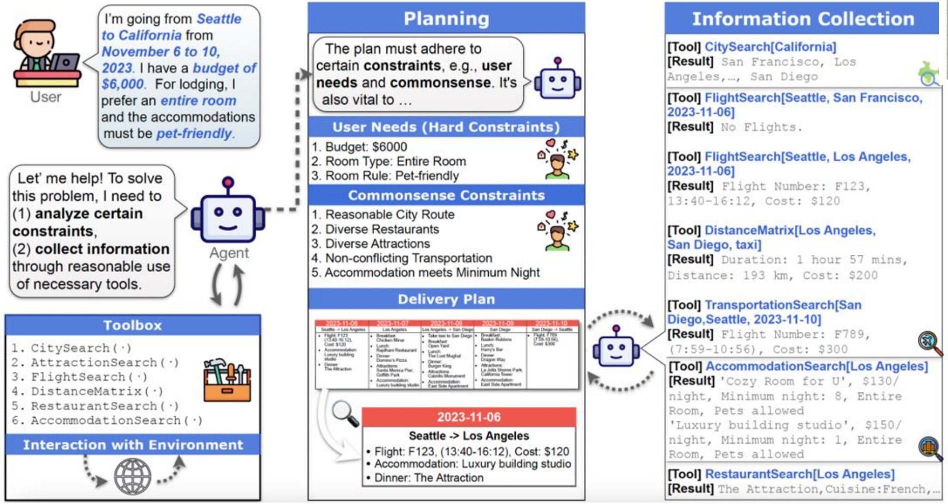
1. Xie, Jian, et al. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint*

TravelPlanner

A Benchmark for Real-World Planning with Language Agents

Jian Xie*[1], Kai Zhang*[2], Jiangjie Chen[1], Tinghui Zhu[1], Renze Lou[3],
Yuandong Tian[4], Yanghua Xiao[1], Yu Su[2]
[1]Fudan University [2]The Ohio State University
[3]The Pennsylvania State University [4]Meta AI
* Equal Contribution
† Corresponding to jianx0321@gmail.com, zhang.13253@osu.edu, su.809@osu.edu

Paper | arXiv | Code | Dataset | Leaderboard | Environment | Twitter



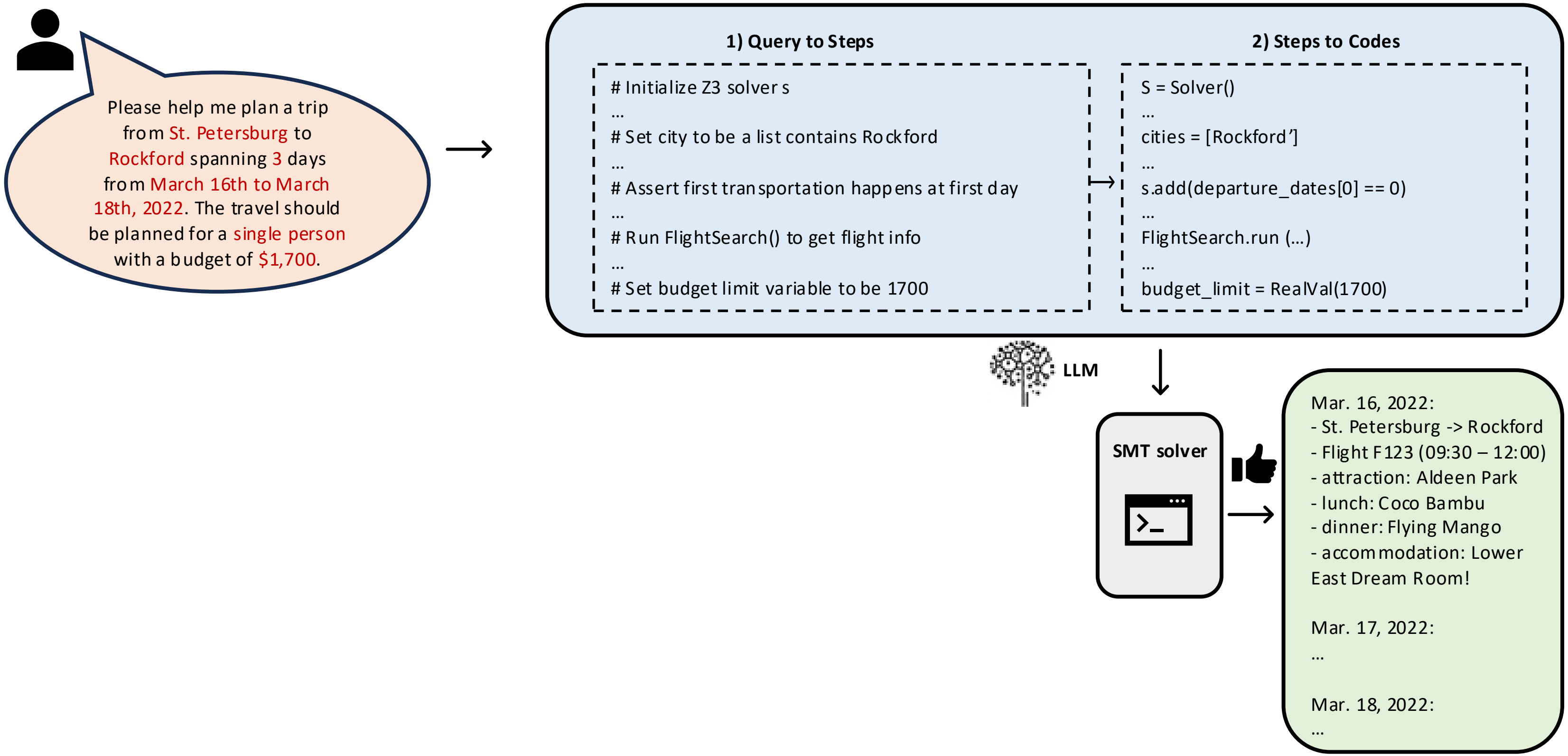
Overview of TravelPlanner. Given a query, language agents are tasked with employing various search tools to gather information. Based on the collected information, language agents are expected to deliver a plan that not only meet the user's needs specified in the query but also adheres to commonsense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

following constraints. The main findings are as follows:

• State-of-the-art LLMs cannot handle complex planning tasks like those in TravelPlanner. GPT-4 successfully produces a plan that meets all the constraints for a few tasks (0.6%), while all other LLMs fail to complete any tasks.

The more recent GPT-o1 can achieve a success rate of 10% with all necessary information provided.

With rigorous solvers, planning is no longer difficult

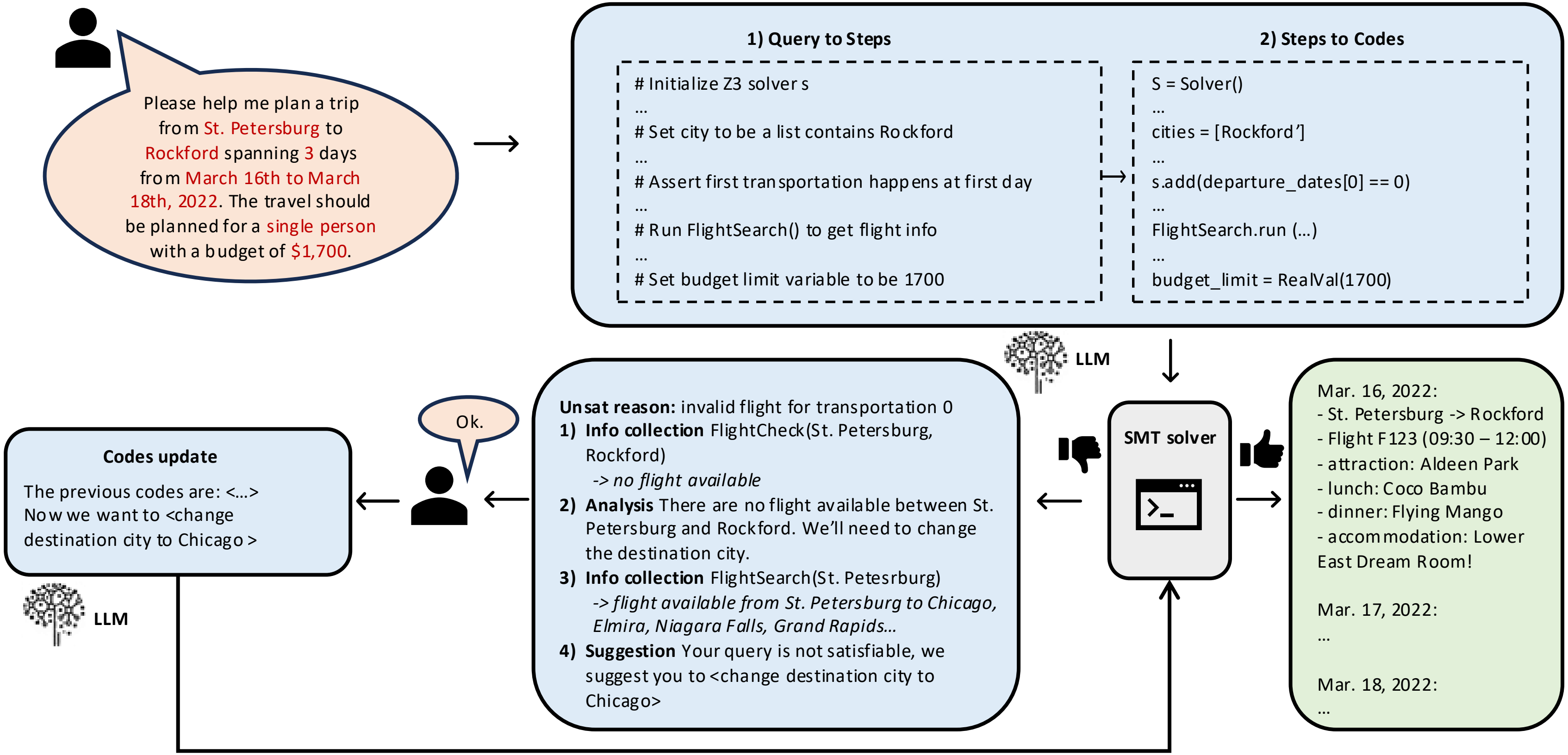


With rigorous solvers, planning is no longer difficult

Method	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate
		Micro	Macro	Micro	Macro	
***Validation** (#180)*						
Greedy Search	**100**	74.4	0	60.8	37.8	0
TwoStage (GPT-4)	89.4	61.1	2.8	15.2	10.6	0.6
Direct (GPT-4)	**100**	80.4	17.2	47.1	22.2	4.4
Direct (o1-preview)	**100**	79.6	15.0	41.9	37.8	10.0
Ours (Mistral-Large)	72.2	72.0	70.6	63.3	66.7	66.7
Ours (Claude-3)	96.1	**96.0**	**95.6**	94.8	93.3	**93.3**
Ours (GPT-4)	95.0	95.0	95.0	**95.7**	**98.9**	**93.3**
***Test** (#1000)*						
Greedy Search	**100**	72.0	0	52.4	31.8	0
TwoStage (GPT-4)	93.1	63.3	2.0	10.5	5.5	0.6
Direct (GPT-4)	**100**	80.6	15.2	44.3	23.1	4.4
Ours (Mistral-Large)	69.9	69.8	69.4	63.0	67.8	67.8
Ours (Claude-3)	95.4	**95.2**	**94.3**	**93.5**	**93.9**	**93.9**
Ours (GPT-4)	91.5	91.4	91.1	91.3	90.2	90.2

```
(ma_satis) yilun@dhcp-10-31-107-178 TravelPlanner main % python csp/test_real_platform.py
/Users/yilun/miniforge3/envs/ma_satis/lib/python3.9/site-packages/langchain/chat_models/__init__.py:31: LangChainDeprecationWarning: Importing c
hat models from langchain is deprecated. Importing from langchain will no longer be supported as of langchain==0.2.0. Please import from langcha
in-community instead:

`from langchain_community.chat_models import ChatOpenAI`.

To install langchain-community run `pip install -U langchain-community`.
  warnings.warn(
/Users/yilun/miniforge3/envs/ma_satis/lib/python3.9/site-packages/langchain/callbacks/__init__.py:37: LangChainDeprecationWarning: Importing thi
s callback from langchain is deprecated. Importing it from langchain will no longer be supported as of langchain==0.2.0. Please import from lang
chain-community instead:

`from langchain_community.callbacks import get_openai_callback`.

To install langchain-community run `pip install -U langchain-community`.
  warnings.warn(
################### QUERY ###################
Please help in creating a travel plan for two people departing from San Francisco and heading to Honolulu for three days, between July 25th and
July 27th, 2024. The budget for this trip is $5,000 and we would prefer to take non-stop flights from United or Hawaiian Airlines
```

With rigorous solvers, we can also fix broken plans

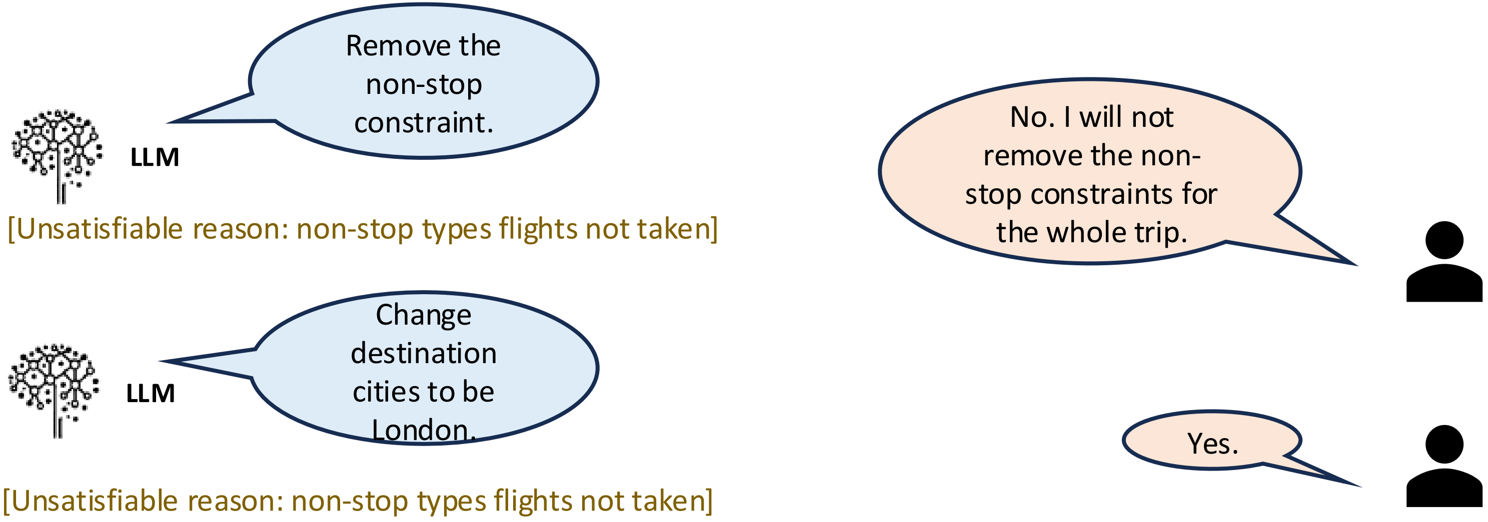


Example 1 – Interactive plan repair

Input query: Please plan a trip from Paris to Kuala Lumpur spanning 3 days from December 28th to December 30th, 2023. The travel should be planned for two people with a budget of $5,000. We want to take non-stop flight only.

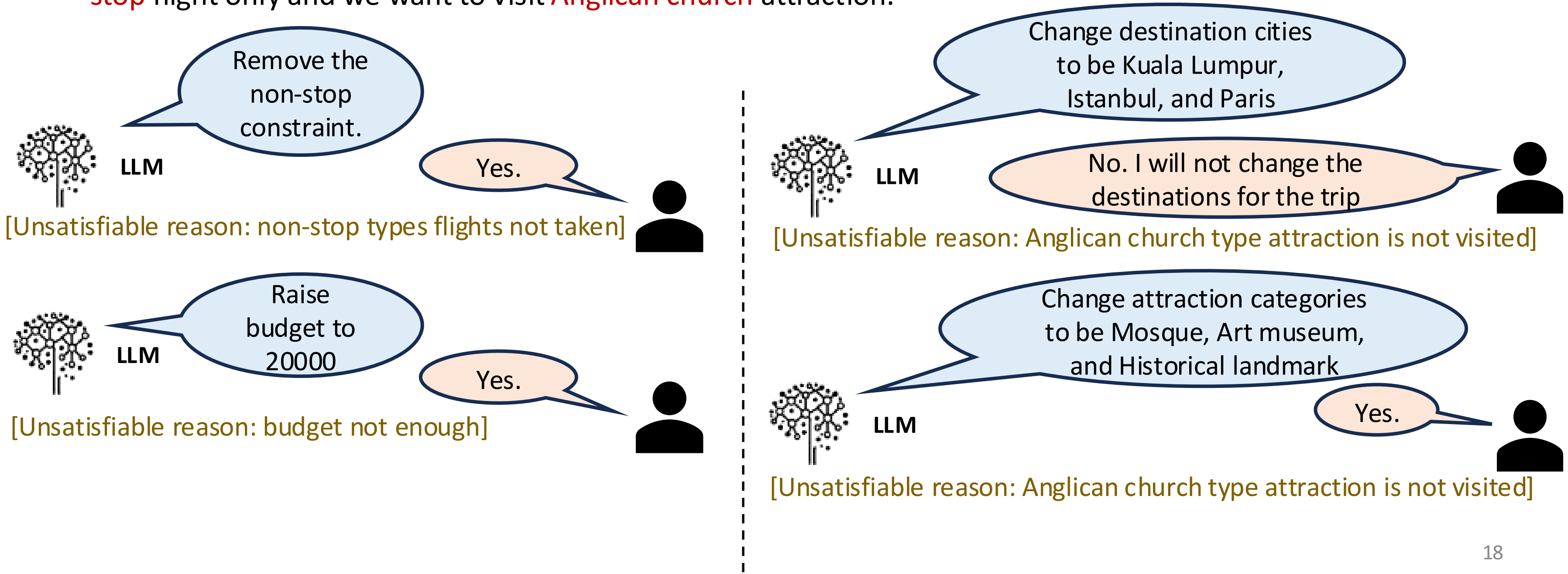
Example 2 – Interactive plan repair

Input query: Please plan a trip from Paris to Kuala Lumpur spanning 3 days from December 28th to December 30th, 2023. The travel should be planned for two people with a budget of $5,000. We want to take non-stop flight only.



Example 3 – Interactive plan repair

Input query: Help me to create a trip plan from Dubai to Istanbul, Paris, and Macau. The trip spans 7 days from December 24th to December 30th, 2023. We have five people with a budget of $15,000. We want to take non-stop flight only and we want to visit Anglican church attraction.



```
(ma_satis) yilun@dhcp-10-31-167-198 TravelPlanner-main % python csp/test_real_platform.py
/Users/yilun/miniforge3/envs/ma_satis/lib/python3.9/site-packages/langchain/chat_models/__init__.py:31: LangChainDeprecationWarning: Importing chat
 models from langchain is deprecated. Importing from langchain will no longer be supported as of langchain==0.2.0. Please import from langchain-com
munity instead:

`from langchain_community.chat_models import ChatOpenAI`.

To install langchain-community run `pip install -U langchain-community`.
  warnings.warn(
/Users/yilun/miniforge3/envs/ma_satis/lib/python3.9/site-packages/langchain/callbacks/__init__.py:37: LangChainDeprecationWarning: Importing this c
allback from langchain is deprecated. Importing it from langchain will no longer be supported as of langchain==0.2.0. Please import from langchain-
community instead:

`from langchain_community.callbacks import get_openai_callback`.
```

Interactive Plan Repair

LLMs can adapt to different user preferences

Input query: Help me to create a trip plan from Dubai to Istanbul, Paris, and Macau. The trip spans 7 days from December 24th to December 30th, 2023. We have five people with a budget of $15,000. We want to take non-stop flight only and we want to visit Anglican church attraction.

Remove the non-stop

Change destination cities to be Kuala Lumpur, Istanbul, and Paris

Method	Always Agree	Budget	Destination Cities	Transportation Methods	House Type	Average
No Reason	75	83.3	91.7	83.3	66.7	80
No Feedback	N/A	50	91.7	66.7	75	70.9
No Solver	16.7	16.7	50	25	16.7	25.0
Ours	91.7	75	100	83.3	75	85.0
Ours-20	100	83.3	100	91.7	83.3	91.7

[Unsatisfiable reason: budget not enough]

LLM

Yes.

[Unsatisfiable reason: Anglican church type attraction is not visited]

LLMs+solver for travel planning: Limitations

Requires task-specific in-context examples in prompts; can solve some other tasks but has limited generalization capability.

Here are some example steps for different constraint:
—–EXAMPLE 1—–
Natural Language query:
Can you create a 5-day travel itinerary for a group of 3, departing from Atlanta and visiting 2 cities in Minnesota from March 3rd to March 7th, 2022? We have a budget of $7,900. We require accommodations that allow parties and should ideally be entire rooms. Although we don't plan to self-drive, we would like the flexibility to host parties.
Steps:
Destination cities
Run CitySearch to get all possible destination cities in Minnesota State for dates ["2022-03-16", "2022-03-17", "2022-03-18"] from origin 'Atlanta', remove origin 'Atlanta' if it is in list
Loop through cities for 2 destination cities
Initialize Z3 solver s
Set 'city' variable to be indexes of 2 destination cities

Can we design a general-purpose framework that can work for different planning problems?

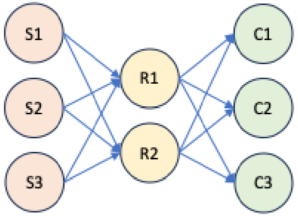
LLMFP: Planning anything with rigor

The core of many *planning* problems lies in *optimization* problems

- searching for the **optimal solution** (**best plan**) with **goals** subject to **constraints** (**preconditions and effects of decisions**)

A supply chain planning example

Task Description: A coffee production company sources beans from three suppliers, roasts them at one of two facilities into either dark or light coffee, and ships the roasted coffee to three retail locations. Each supplier has a limited capacity. Each roastery, with no existing inventory, can roast one unit of coffee beans into one unit of dark or light coffee. The retail locations have specific demands for dark and light coffee, with no existing inventory. The company's objective is to minimize the total cost, including shipping beans, roasting, and shipping roasted coffee, while ensuring that all coffee produced meets or exceeds the demand at each retail location



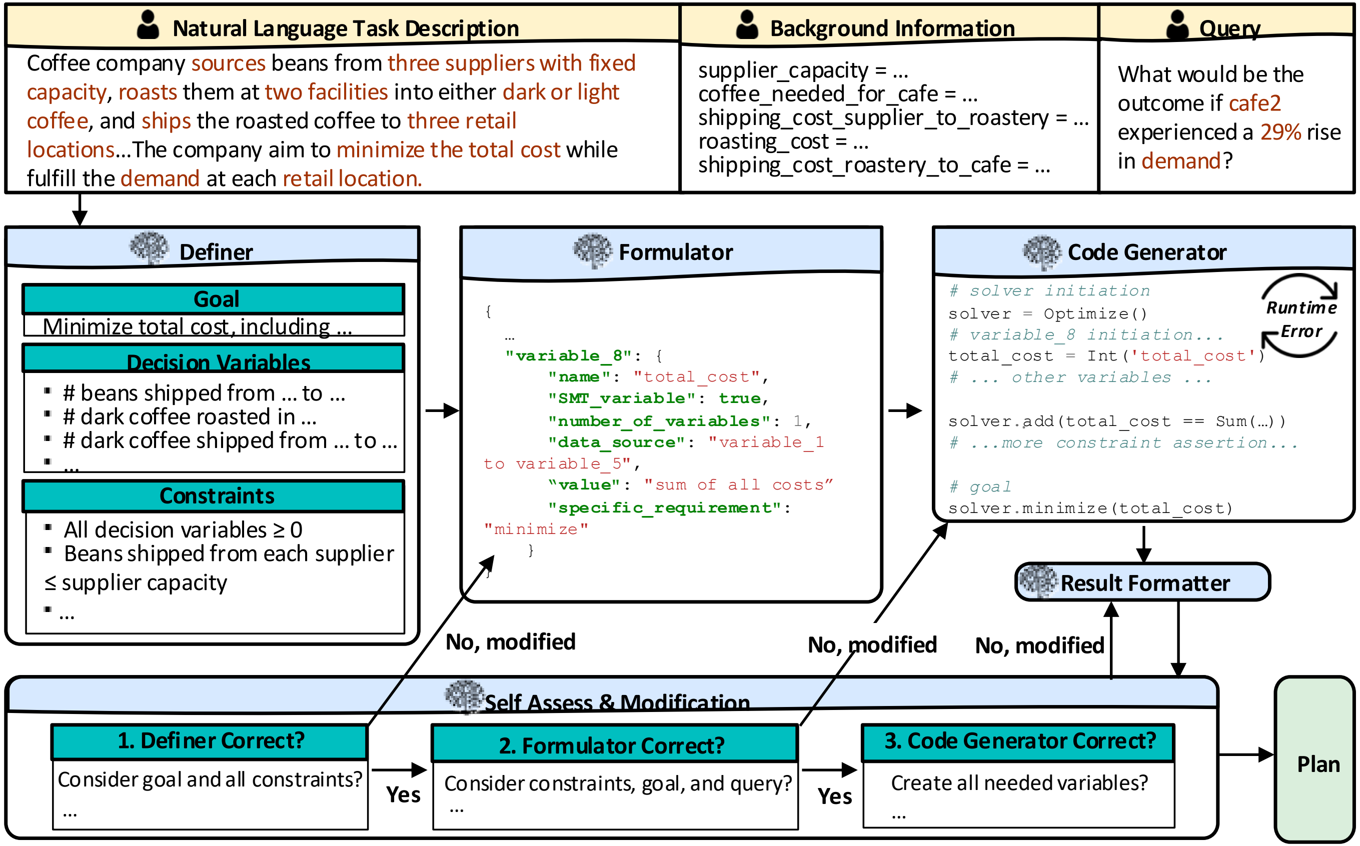
What is the potential impact of a 29% increase in demand at cafe2?

Existing works often rely on task-specific preparatory efforts, such as task-specific in-context examples and pre-defined critics/verifiers.

- This limits their cross-task generalization capability.

Our framework uses LLMs to capture key information from planning problems, and then formally formulate and solve them as satisfiability/optimization problems from scratch **without** task-specific examples.

LLMFP: Planning anything with rigor



LLMFP – Definer

- Defining goals, decision variables, constraints

```
You are given a task description in natural l
    it by building an optimization problem fo
The task is: {task_description}
You have the access to {info_api}
To get started of building the optimization p
    decision variables, and constraints to co
Specifically, consider:
Goal: define the objective trying to optimize
Decision variables: identify all the decision
    problem
Constraints: key requirement for decision var
    decision variables, carefully consider re
     underlying assumption, unmentioned commo
    explicitly include as constraint to ensure
    connected with each other
Response with [[GOAL: ]], [[Decision Variable
    Reasoning: ]], and [[Constraints: ]] only
    math formulas. Try to be thorough and inc
    as much as you can.
```

Definer output example

GOAL: Minimize the total cost, including shipping beans from suppliers to roasteries, roasting costs, and shipping roasted coffee to retail locations.

Decision Variables:
- Amount of beans shipped from each supplier to each roastery.
- Amount of light coffee roasted at each roastery.
- Amount of dark coffee roasted at each roastery.
- Amount of light coffee shipped from each roastery to each cafe.
- Amount of dark coffee shipped from each roastery to each cafe.

Constraints Reasoning:
1. Supplier capacity limits the amount of coffee beans they can supply.
2. Roasteries can only roast as much coffee as they receive in beans.
3. Roasteries must meet the demand for both light and dark coffee at each cafe.
4. The total amount of coffee shipped from suppliers to roasteries must equal the total amount of coffee roasted.
5. The total amount of light and dark coffee shipped from roasteries to cafes must meet the demand at each cafe.

Constraints:
- Total beans shipped from each supplier to all roasteries ≤ Supplier capacity.
- Total beans received by each roastery = Total coffee roasted (light + dark) at that roastery.
- Total light coffee roasted at each roastery = Total light coffee shipped from that roastery to all cafes.
- Total dark coffee roasted at each roastery = Total dark coffee shipped from that roastery to all cafes.
- Total light coffee shipped to each cafe from all roasteries ≥ Light coffee needed at that cafe.
- Total dark coffee shipped to each cafe from all roasteries ≥ Dark coffee needed at that cafe.
- All decision variables ≥ 0.

LLMFP – Formulator

- Proposing all variables needed to encode the problem

Example Fomulator output for multi-constraint task Coffee


```
{
  "variable_1": {
     "name": "beans_shipped",
     "SMT_variable": true,
     "number_of_variables": 6,
     "data_source": "capacity_in_supplier,
shipping_cost_from_supplier_to_roastery",
     "value": "amount of coffee beans shipped from
each supplier to each roastery",
     "specific_requirement": "must not exceed
supplier's capacity"
  },
  "variable_2": {
     "name": "light_roasted",
     "SMT_variable": true,
     "number_of_variables": 2,
     "data_source": "roasting_cost_light",
     "value": "amount of light coffee roasted at each
roastery",
     "specific_requirement": "total amount of coffee
beans received by each roastery must equal total
amount of coffee roasted"
  },
  …
  "variable_6": {
     "name": "total_cost",
     "SMT_variable": true,
     "number_of_variables": 1,
     "data_source": "variable_1, variable_2,
variable_3, variable_4, variable_5",
     "value": "sum of shipping, roasting, and shipping
roasted coffee costs",
     "specific_requirement": "minimize"
  },
}
```

LLMFP – Code Generator

- Generating codes for the solvers

```
# Constraints
# Total beans shipped from each supplier to all roasteries <= Supplier capacity
capacity_in_supplier = {'supplier1': 150, 'supplier2': 50, 'supplier3': 100}
for s in ['supplier1', 'supplier2', 'supplier3']:
    solver.add(Sum([beans_shipped[(s, r)] for r in ['roastery1', 'roastery2']]) <= capacity_in_supplier[s])
# Total beans received by each roastery = Total coffee roasted (light
for r in ['roastery1', 'roastery2']:
    solver.add(Sum([beans_shipped[(s, r)] for s in ['supplier1', 'supp
+ dark_coffee_roasted[r])
# Total light coffee roasted at each roastery = Total light coffee shi
for r in ['roastery1', 'roastery2']:
    solver.add(light_coffee_roasted[r] == Sum([light_coffee_shipped[(r
# Total dark coffee roasted at each roastery = Total dark coffee shipp
for r in ['roastery1', 'roastery2']:
    solver.add(dark_coffee_roasted[r] == Sum([dark_coffee_shipped[(r,
# Total light coffee shipped to each cafe from all roasteries ≥‰¥ Ligh
for c in ['cafe1', 'cafe2', 'cafe3']:
    solver.add(Sum([light_coffee_shipped[(r, c)] for r in ['roastery1'
light_coffee_needed_for_cafe[c])
# Total dark coffee shipped to each cafe from all roasteries ≥‰¥ Dark
for c in ['cafe1', 'cafe2', 'cafe3']:
    solver.add(Sum([dark_coffee_shipped[(r, c)] for r in ['roastery1', 'roastery2']]) >= dark_coffee_needed_for_cafe[c])
# All decision variables >= 0
for var in beans_shipped.values():
    solver.add(var >= 0)
for var in light_coffee_roasted.values():
    solver.add(var >= 0)
for var in dark_coffee_roasted.values():
    solver.add(var >= 0)
for var in light_coffee_shipped.values():
    solver.add(var >= 0)
for var in dark_coffee_shipped.values():
    solver.add(var >= 0)
```

```
total_cost_expr = Sum([
    beans_shipped[(s, r)] * shipping_cost_from_supplier_to_roastery[(s, r)]
    for s in ['supplier1', 'supplier2', 'supplier3'] for r in ['roastery1', 'roastery2']
]) + Sum([
    light_coffee_roasted[r] * roasting_cost_light[r] for r in ['roastery1', 'roastery2']
]) + Sum([
    dark_coffee_roasted[r] * roasting_cost_dark[r] for r in ['roastery1', 'roastery2']
]) + Sum([
    light_coffee_shipped[(r, c)] * shipping_cost_from_roastery_to_cafe[(r, c)]
    for r in ['roastery1', 'roastery2'] for c in ['cafe1', 'cafe2', 'cafe3']
]) + Sum([
    dark_coffee_shipped[(r, c)] * shipping_cost_from_roastery_to_cafe[(r, c)]
    for r in ['roastery1', 'roastery2'] for c in ['cafe1', 'cafe2', 'cafe3']
])
```

LLMFP – Self Assessor

- Self assess and modification

```
Based on the previous information, evaluate whether steps 1-3 are correct
    :
For Step 1: Does the step consider correct goal and all needed
    constraints? Are there unnecessary or missing constraints? Does the
    execution result make sense and achievable in reality and meet
    commonsense?
For Step 2: Do the variables explicitly consider the query? Do the
    variables explicitly consider and encode all provided constraints and
     goal? Does the order make sense? Does the execution result make
    sense and achievable in reality and meet commonsense?
For Step 3: Does the code create all needed variables? Does the code make
     up any ungiven information? Does the code correctly address the
    query? Does the execution result make sense and achievable in reality
     and meet commonsense?
Please reason the correctness with task context, rate each step with a
    binary score: 1 is correct, 0 is incorrect, think about how to modify
     in detail according to task and query, and modify the step if you
    think it is incorrect.
```

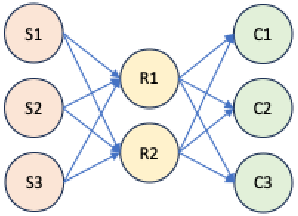
Result of the coffee supply chain problem

The core of many *planning* problems lies in *optimization* problems

- searching for the **optimal solution** (**best plan**) with **goals** subject to **constraints** (**preconditions and effects of decisions**)

A supply chain planning example

Task Description: A coffee production company sources beans from three suppliers, roasts them at one of two facilities into either dark or light coffee, and ships the roasted coffee to three retail locations. Each supplier has a limited capacity. Each roastery, with no existing inventory, can roast one unit of coffee beans into one unit of dark or light coffee. The retail locations have specific demands for dark and light coffee, with no existing inventory. The company's objective is to minimize the total cost, including shipping beans, roasting, and shipping roasted coffee, while ensuring that all coffee produced meets or exceeds the demand at each retail location



What is the potential impact of a 29% increase in demand at cafe2?

"suppliers": {
"supplier_1_ship_to_roastery_1": 0, "supplier_1_ship_to_roastery_2": 95,
"supplier_2_ship_to_roastery_1": 0, "supplier_2_ship_to_roastery_2": 50,
"supplier_3_ship_to_roastery_1": 100, "supplier_3_ship_to_roastery_2": 0
},
"roasteries": {
"roastery_1_dark_coffee": 41, "roastery_1_light_coffee": 59,
"roastery_2_dark_coffee": 105, "roastery_2_light_coffee": 40
},
"cafe": {
"roastery_1_ship_dark_to_cafe_1": 15, "roastery_1_ship_light_to_cafe_1": 20,
"roastery_1_ship_dark_to_cafe_2": 26, "roastery_1_ship_light_to_cafe_2": 39,
"roastery_1_ship_dark_to_cafe_3": 0, "roastery_1_ship_light_to_cafe_3": 0,
"roastery_2_ship_dark_to_cafe_1": 5, "roastery_2_ship_light_to_cafe_1": 0,
"roastery_2_ship_dark_to_cafe_2": 0, "roastery_2_ship_light_to_cafe_2": 0,
"roastery_2_ship_dark_to_cafe_3": 100, "roastery_2_ship_light_to_cafe_3": 40
},
"total_cost": 2612

Zero-shot experimental results

Table 1: Optimal rate (%) comparison of LLMFP with baselines on 5 multi-constraint problems.

Method	Coffee	Workforce	Facility	Task Allocation	Warehouse	Average
Direct$_{\text{GPT-4o}}$	0.8	2.6	0.0	0.0	0.0	0.7
Direct$_{\text{o1-preview}}$	25.9	47.6	4.8	4.0	66.0	29.7
CoT$_{\text{GPT-4o}}$	0.0	5.6	0.0	0.0	16.0	4.3
Code$_{\text{GPT-4o}}$	17.7	75.8	53.9	0.0	8.0	31.1
LLMFP $_{\text{GPT-4o}}$	**64.7**	**92.2**	**70.7**	**96.0**	**72.0**	**79.1**
Direct$_{\text{Claude 3.5 Sonnet}}$	0.0	0.0	0.0	0.0	0.0	0.0
CoT$_{\text{Claude 3.5 Sonnet}}$	7.1	0.0	0.0	0.0	14.0	4.2
Code$_{\text{Claude 3.5 Sonnet}}$	59.8	71.9	47.3	0.0	42.0	44.2
LLMFP $_{\text{Claude 3.5 Sonnet}}$	**80.5**	**88.7**	**48.2**	**96.0**	**90.0**	**80.7**

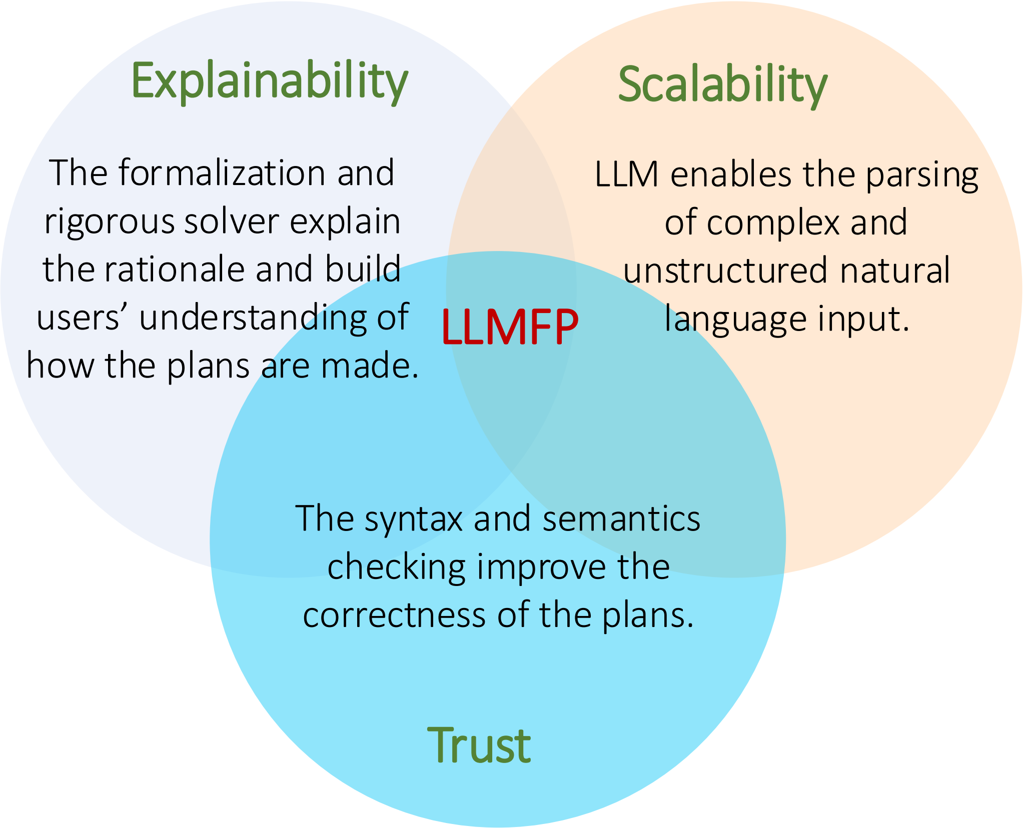
Table 2: Optimal rate (%) comparison of LLMFP with baselines on 4 multi-step problems.

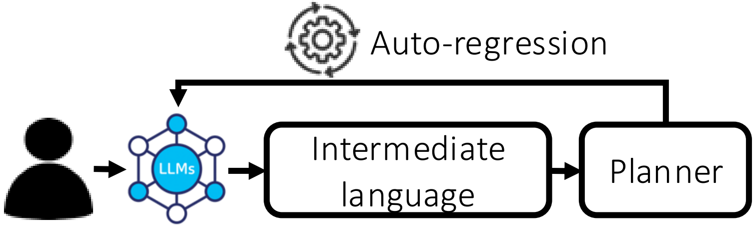
Method	Blocksworld	Mystery Blocksworld	Movie	Gripper	Average
Direct$_{\text{GPT-4o}}$	41.5	0.8	85.7	0.0	32.0
Direct$_{\text{o1-preview}}$	88.4	31.9	**100.0**	52.0	68.1
CoT$_{\text{GPT-4o}}$	39.9	2.7	81.0	0.0	30.9
Code$_{\text{GPT-4o}}$	0.0	0.3	0.0	0.0	0.1
LLMFP $_{\text{GPT-4o}}$	**96.2**	**77.7**	**100.0**	**76.0**	**87.5**
Direct$_{\text{Claude 3.5 Sonnet}}$	43.2	0.5	**100.0**	12.0	38.9
CoT$_{\text{Claude 3.5 Sonnet}}$	52.8	2.8	**100.0**	28.0	45.9
Code$_{\text{Claude 3.5 Sonnet}}$	0.0	0.0	0.0	0.0	0.0
LLMFP $_{\text{Claude 3.5 Sonnet}}$	**93.0**	**98.0**	**100.0**	**76.0**	**91.8**

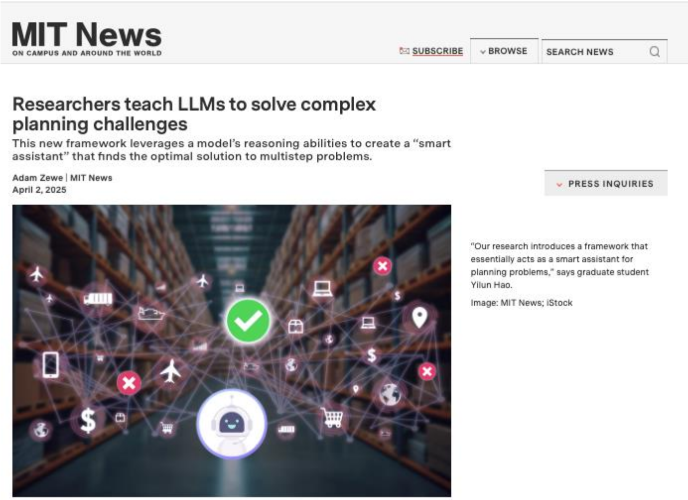
Number of queries in each problem category:

- Coffee: 266
- Workforce: 231
- Facility: 165
- Task Allocation: 50
- Warehouse: 50
- Blocksworld: 602
- Mystery Blocksworld: 602
- Movie: 21
- Gripper: 25

Innovation and uniqueness of LLMFP






MIT News
ON CAMPUS AND AROUND THE WORLD
SUBSCRIBE | BROWSE | SEARCH NEWS

Researchers teach LLMs to solve complex planning challenges

This new framework leverages a model's reasoning abilities to create a "smart assistant" that finds the optimal solution to multistep problems.

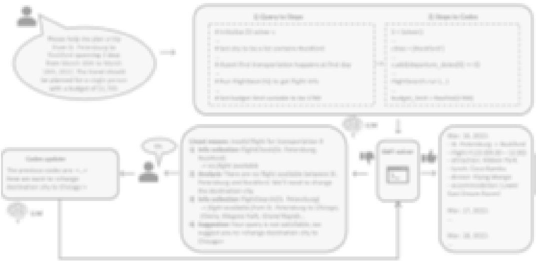
Adam Zewe | MIT News
April 2, 2025

PRESS INQUIRIES

"Our research introduces a framework that essentially acts as a smart assistant for planning problems," says graduate student Yilun Hao.

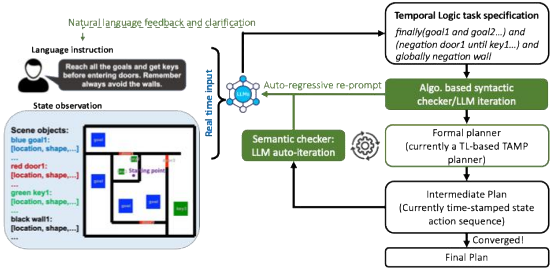
Image: MIT News; iStock

Outline



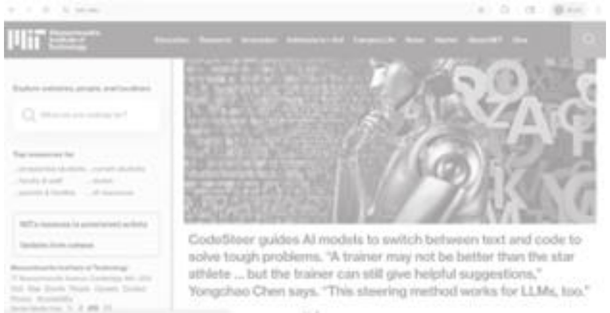
LLM + rigorous solver can make very complex plans [1,2]

→



LLM + rigorous solver can control dynamic robots [3]

→



Unifying code, text-based reasoning, and tool use in LLMs [4,5,6]

[1] Y. Hao, Y. Chen, Y. Zhang, and C. Fan. "Large Language Models can solve real-world planning rigorously with formal verification tools." NAACL (2024).

[2] Y. Hao, Y. Zhang, and C. Fan. "Planning anything with rigor: general-purpose zero-shot planning with LLM-based formalized programming." ICLR (2024).

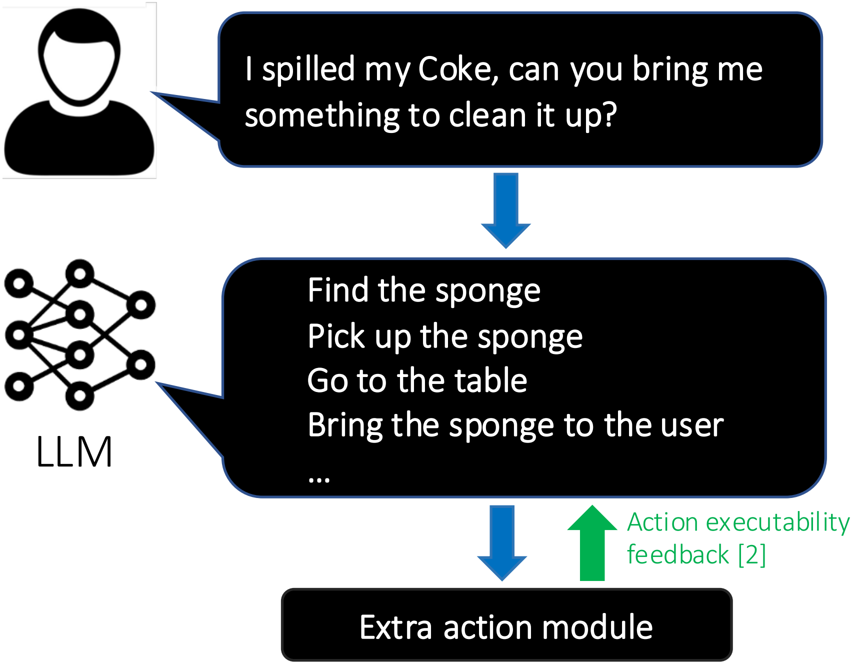
[3] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan. "AutoTAMP: Autoregressive task and motion planning with LLMs as translators and checkers." ICRA (2024).

[4] Y. Chen, Y. Hao, Y. Zhang, and C. Fan. "Code-as-symbolic-planner: Foundation model-based robot planning via symbolic code generation." IROS (2025).

[5] Y. Chen, H. Jhamtani, S. Sharma, C. Fan, C. Wang. "Steering LLMs between Code Execution and Textual Reasoning." ICLR, 2025.

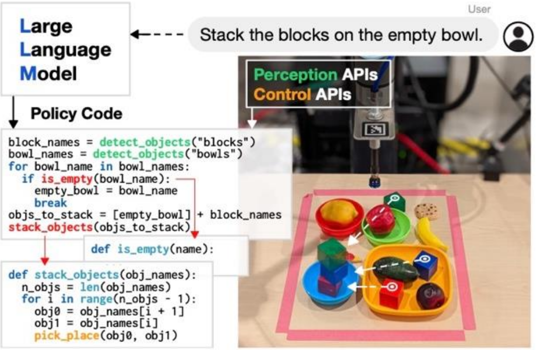
[6] Y. Chen, Y. Hao, Y. Liu, Y. Zhang, and C. Fan. "CodeSteer: Symbolic-Augmented Language Models via Code/Text Guidance." ICML, 2025.

LLMs for robot task planning (at the beginning)

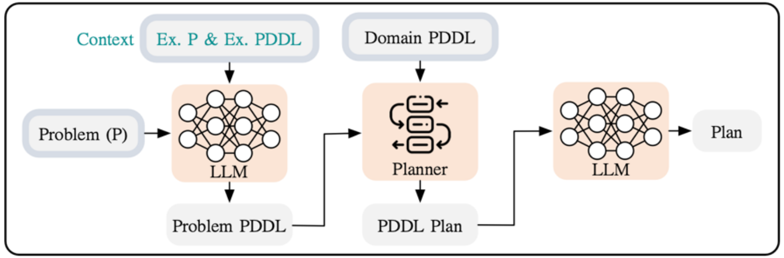


1. Wenlong Huang et al. Language Models as Zero-Shot Planners, ICML 2022.

2. Michael Ahn et al. Do as I can, not as I say: Grounding Language in Robotic Affordances, CoRL 2022.



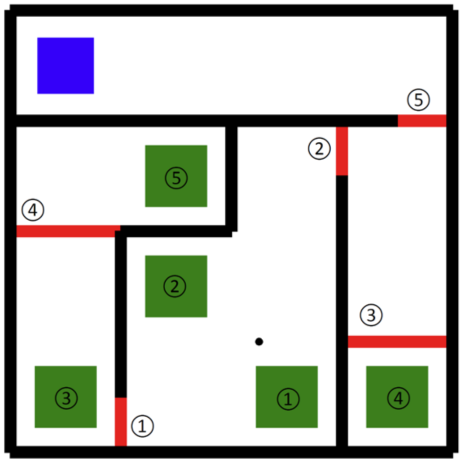
3. Jacky Liang, et al. Code as Policies: Language Model Programs for Embodied Control. ICRA 2023.



4. Bo Liu et al. ProgPrompt: LLM+P: Empowering Large Language Models with Optimal Planning Proficiency.

LLMs provide common knowledge
LLMs output structured language

Temporal logic as specification languages



1. Robots should always reach their destinations with the minimum traveling time
2. Robots cannot travel through walls or closed doors
3. Robots must visit the correspondingly numbered green region to pick the keys to open the doors

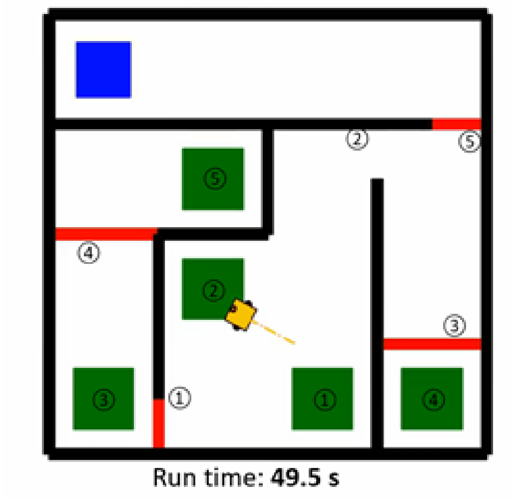
$$\left(\mathbf{F}_{[0,10]}\text{Destination}\right) \land \left(\mathbf{G}_{[0,10]}\neg\text{Wall}\right) \land \left(\bigwedge_{i=1}^{N} \neg\text{Door}_i \ \mathbf{U}\ \text{Key}_i\right)$$

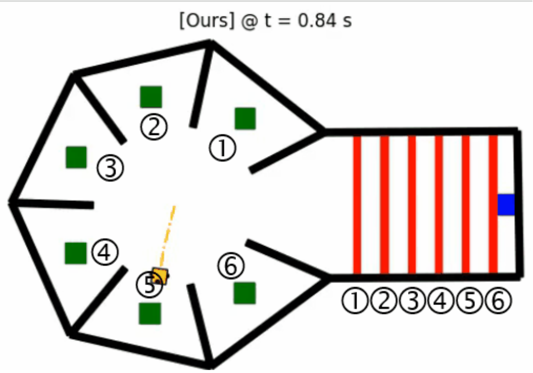
(Signal) Temporal Logics (STL) are a high-level specification language: They tell robots what to do (rules), not how to do it (actions)

- STL captures many complex planning considerations, including logical dependencies and temporal constraints
- STL can be translated from natural languages [NL2TL, EMNLP’23]
- We can synthesize or learn planners that satisfy arbitrary STL specifications

Robot task and motion planning from Temporal Logic

We have developed effective planning algorithms that can plan motions for agents with provably optimal computational complexity. [1]



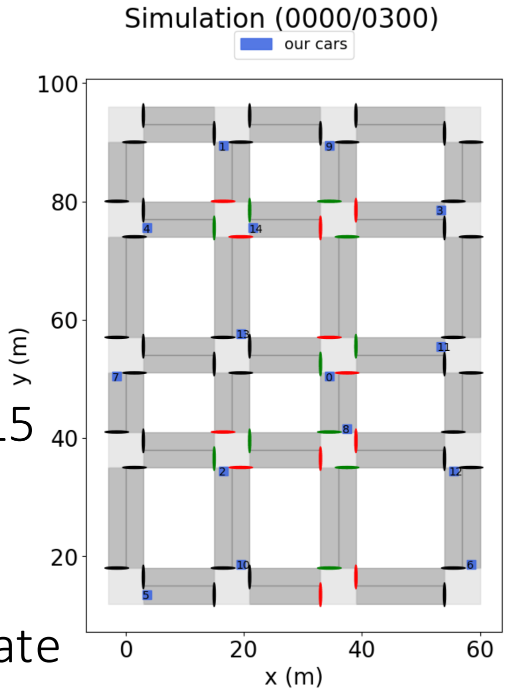


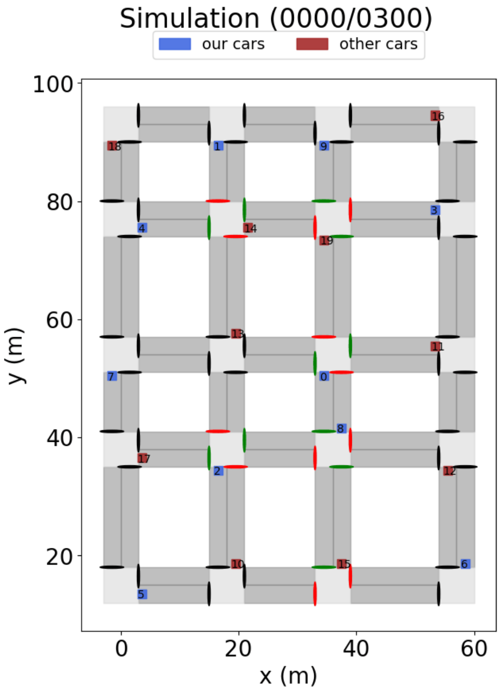
$$\left(\mathbf{F}_{[0,10]}\text{Destination}\right) \wedge \left(\mathbf{G}_{[0,10]}\neg\text{Wall}\right) \wedge \left(\bigwedge_{i=1}^{N} \neg\text{Door}_i \ \mathbf{U}\ \text{Key}_i\right)$$

[1] Sun, Dawei, Jingkai Chen, Sayan Mitra, and Chuchu Fan. "Multi-agent motion planning from signal temporal logic specifications." *IEEE Robotics and Automation Letters (2022)*.

Robot task and motion planning from Temporal Logic

- Embed the following rules in one vehicle's action, trained in a single intersection.
  - Avoid rear-end collision
  - Stop sign rules
  - Traffic light rules
- Evaluated with 20 intersections and 15 cars, our vehicles learn to follow the traffic rules. [1]
- Our learned vehicles can also cooperate with human-driven vehicles.

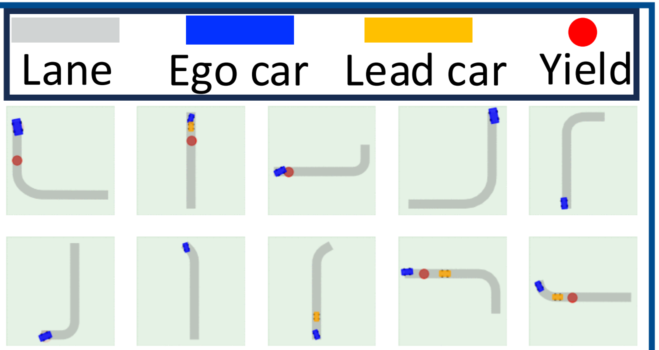




[1] Meng, Yue, and Chuchu Fan. "Signal Temporal Logic Neural Predictive Control." *IEEE Robotics and Automation Letters (2023).*

Learn to drive in CARLA

Training



Formulate traffic rules as STL and train the controller from varied scenarios.

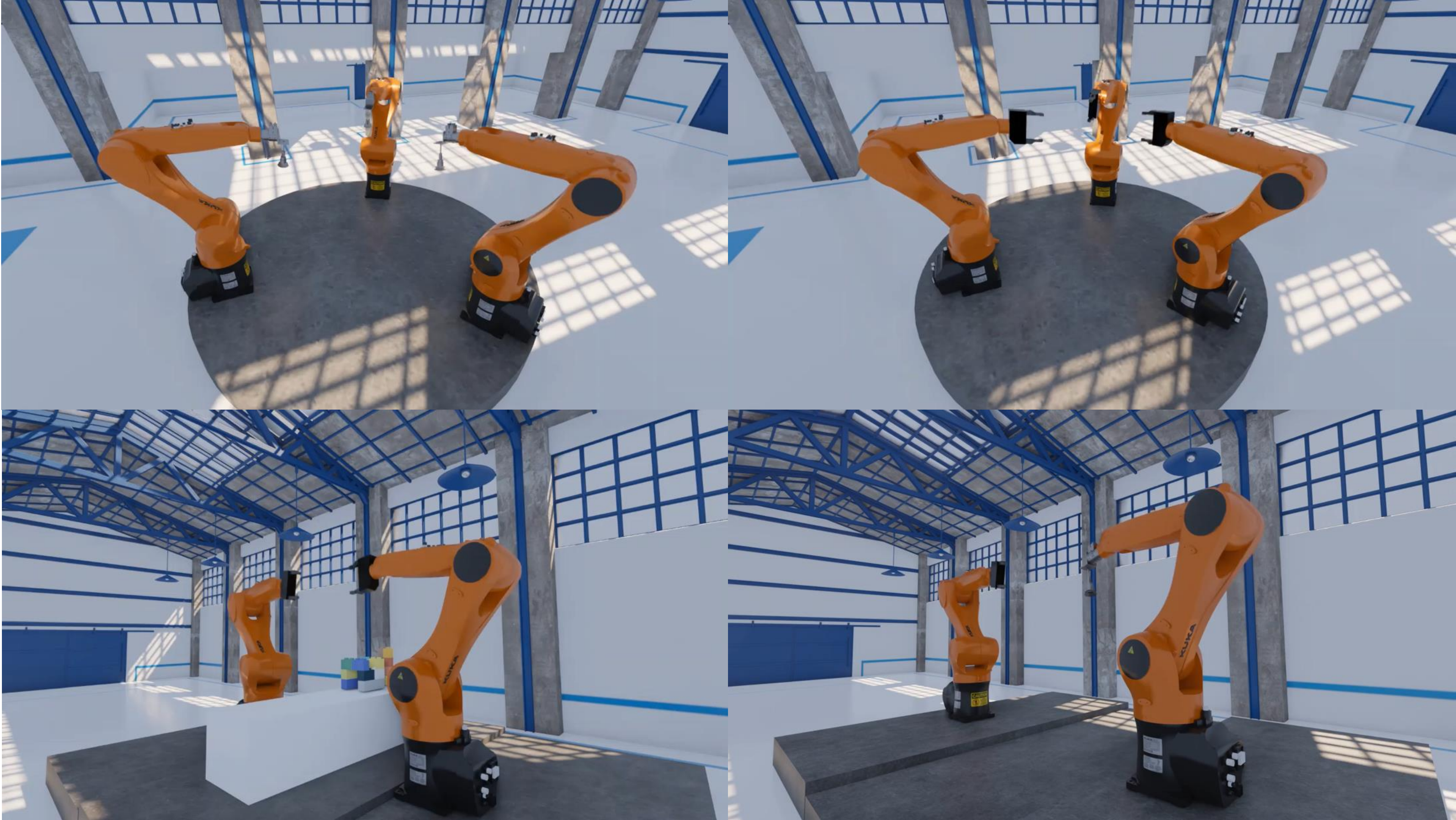
Testing

Learn to follow traffic light and stop-sign rules

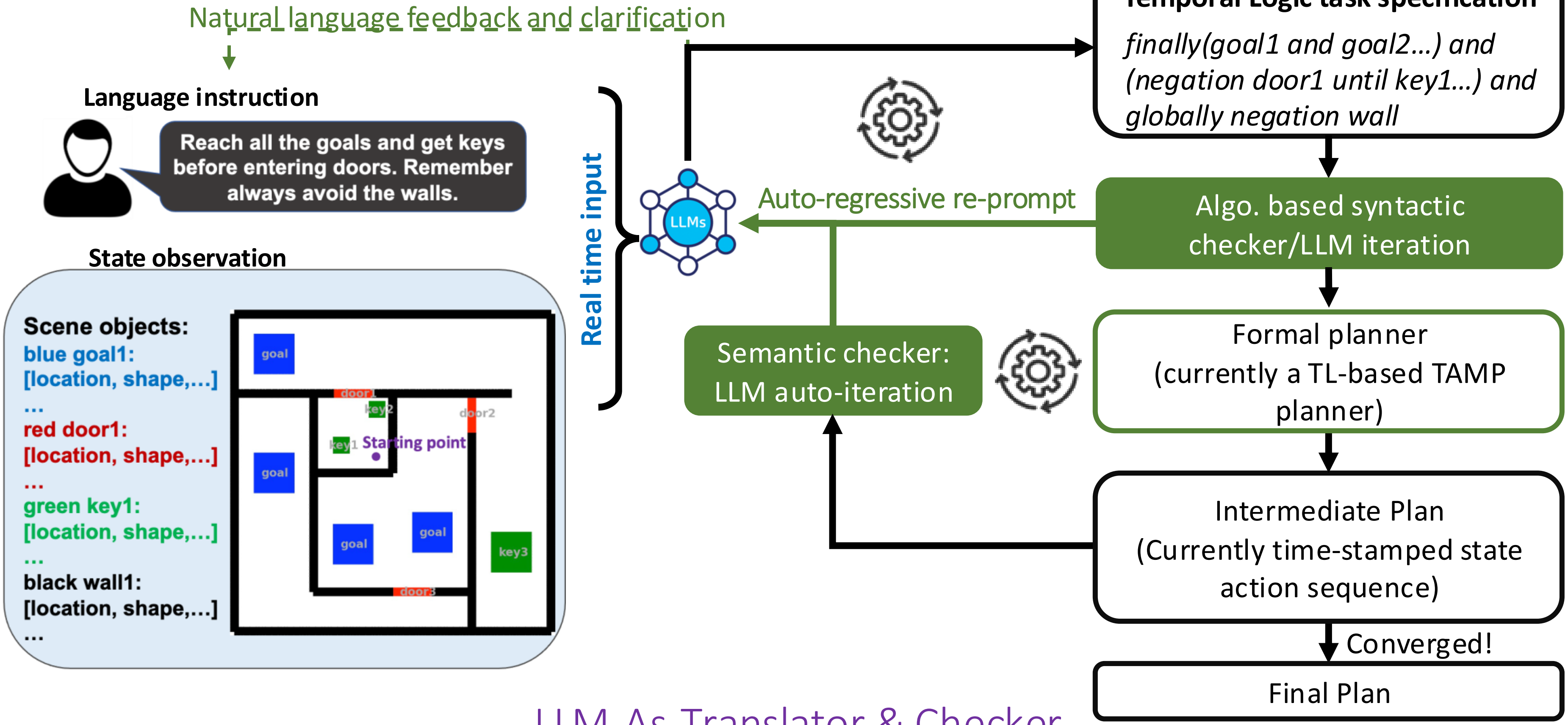
Heterogenous agent modelling (aggressive; conservative; test)

Adaptation to unseen roundabout scenarios

Yue Meng and Chuchu Fan, Signal Temporal Logic Neural Predictive Control, RA-L'23.


KUKA

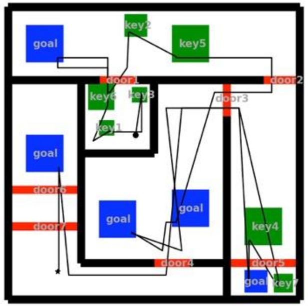
Auto-regressive LLM-based Task and Motion Planning (AutoTAMP)



LLM-As-Translator & Checker

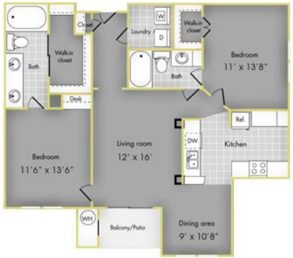
AutoTAMP results: Single-agent, time & geometric constraints

Solving puzzles



"Try to reach all the goals but you have to reach the corresponding key first to open the specific door. For example, you have to reach key1 ahead to open door1. Also, remember always do not touch the walls."

Household work



"Navigate into the pink bedroom and then go to the yellow restroom, but remember do not touch the livingroom at anytime."

	HouseWorld1 (soft time cst.)	HouseWorld2 (hard time cst.)	Chip Challenge (hard geo. cst.)
GPT-3 end2end	0%	0%	0%
GPT-3 naive task planning	74%	36%	0%
SayCan	75.5%	36%	0%
GPT-3 task planning/feed.	**79%**	40%	0%
GPT-3/STL	28%	27%	29%
GPT-3/STL/Syn.	49%	47%	66%
GPT-3/STL/Syn./Sem. (AutoTAMP)	62%	**62%**	**74.3%**
GPT-4 end2end	9.5%	9.5%	0%
GPT-4 naive task planning	90%	45%	0%
SayCan	90%	47.5%	0%
GPT-4 task planning/feed.	**92%**	49%	0%
GPT-4/STL	43.5%	42%	42.7%
GPT-4/STL/Syn.	59.5%	59%	70%
GPT-4/STL/Syn./Sem. (AutoTAMP)	82.5%	**82%**	**87.7%**

Task success rates of different approaches in single-agent TAMP (results obtained in May 2023).

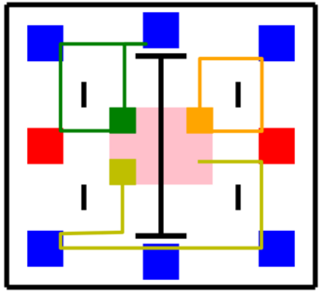
AutoTAMP uses Temporal Language as the intermediate language, which enables auto-regressive reasoning and significantly improves the success rate of using LLM for TAMP when there are hard timing and geometric constraints.

AutoTAMP results: Multi-agent, time & geometric constraints

Overcooked game

"Enter all the ingredient room to pick up food. Once entered ingredient rooms, go to cooking room within 3 seconds. And all the agents should not collide each other and black obstacles."

Multi-agent surveillance



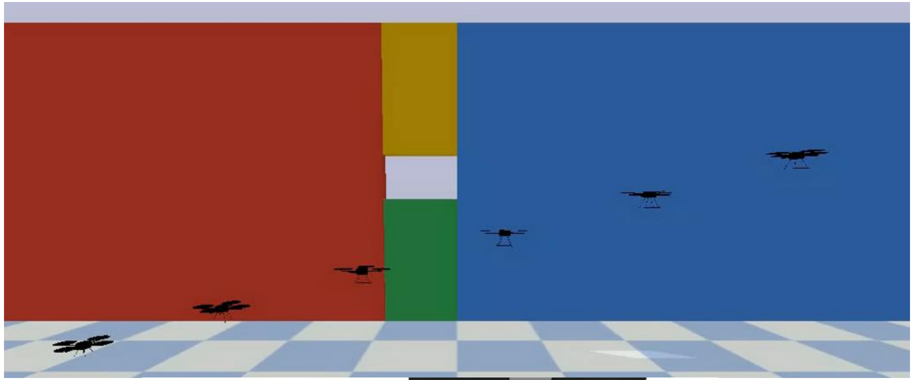
"All rovers must reach the pink charging station within 5 units of time each time they exit it. Once they reach their (blue) destination, they need to get to a red transmitter within 2-time units to send the collected information to the remote control. Rovers must keep clear of black walls and other rovers. All target areas need to be visited."

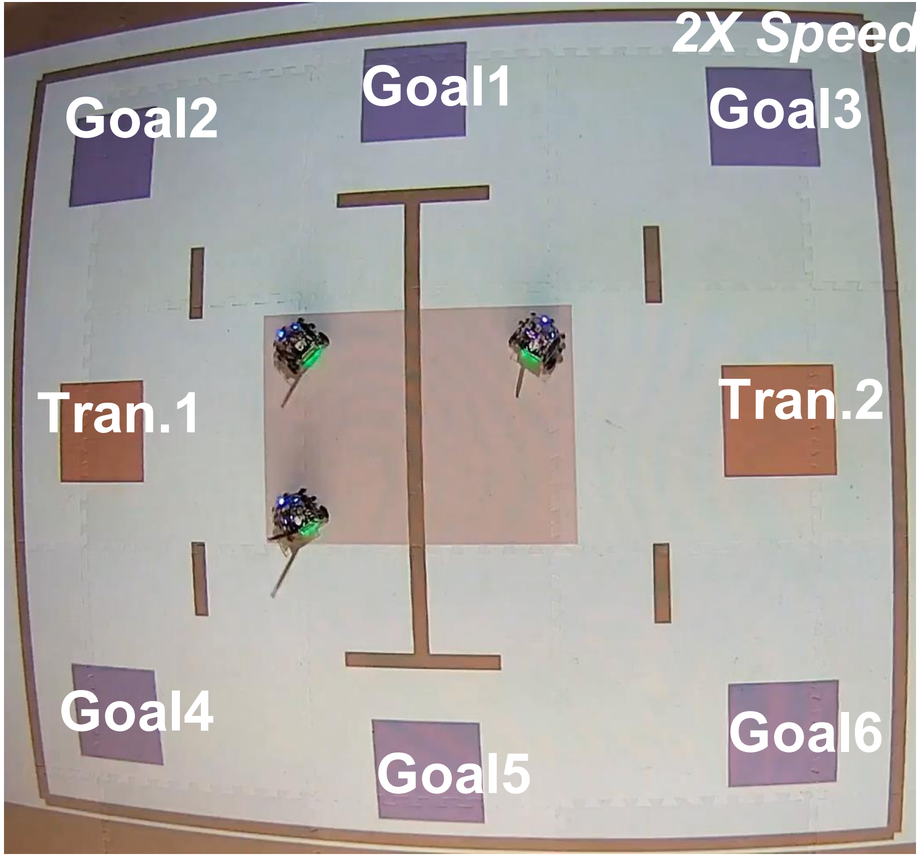
	Overcooked	Rover	Wall
	(multi-agent, hard time & geo. cst.)		
GPT-3 end2end	0%	0%	0%
GPT-3 naive task planning	13.3%	0%	7%
GPT-3/STL	25%	22%	74%
GPT-3/STL/Syn	70%	35%	85%
GPT-3/STL/Syn./Sem. (AutoTAMP)	**89%**	**60.7%**	**89.7%**
GPT-4 end2end	5%	0%	6%
GPT-4 naive task planning	17%	0%	47%
GPT-4/STL	85%	46%	95%
GPT-4/STL/Syn	94%	67%	95%
GPT-4/STL/Syn./Sem. (AutoTAMP)	**100%**	**79%**	**100%**

Task success rates of different approaches in multi-agent TAMP (results obtained in May 2023).

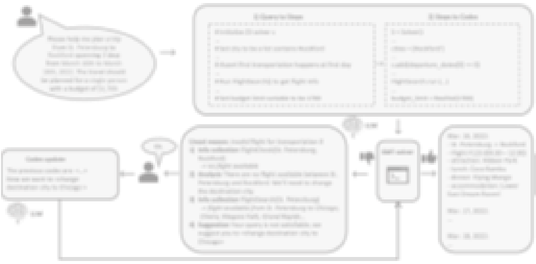
Replacing the TAMP planner with any planner, the AutoTAMP framework will interactively provide rigorous and explainable assistants for human operators, with improved success rates and unstructured human user inputs.

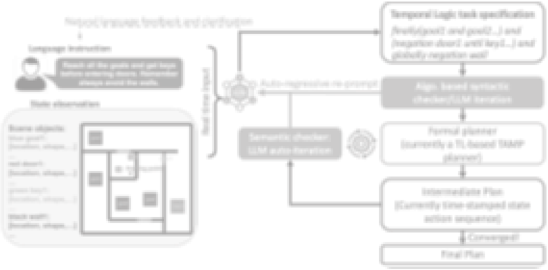
Experiments

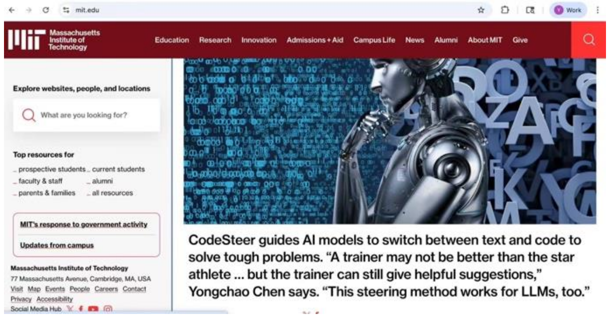




Outline



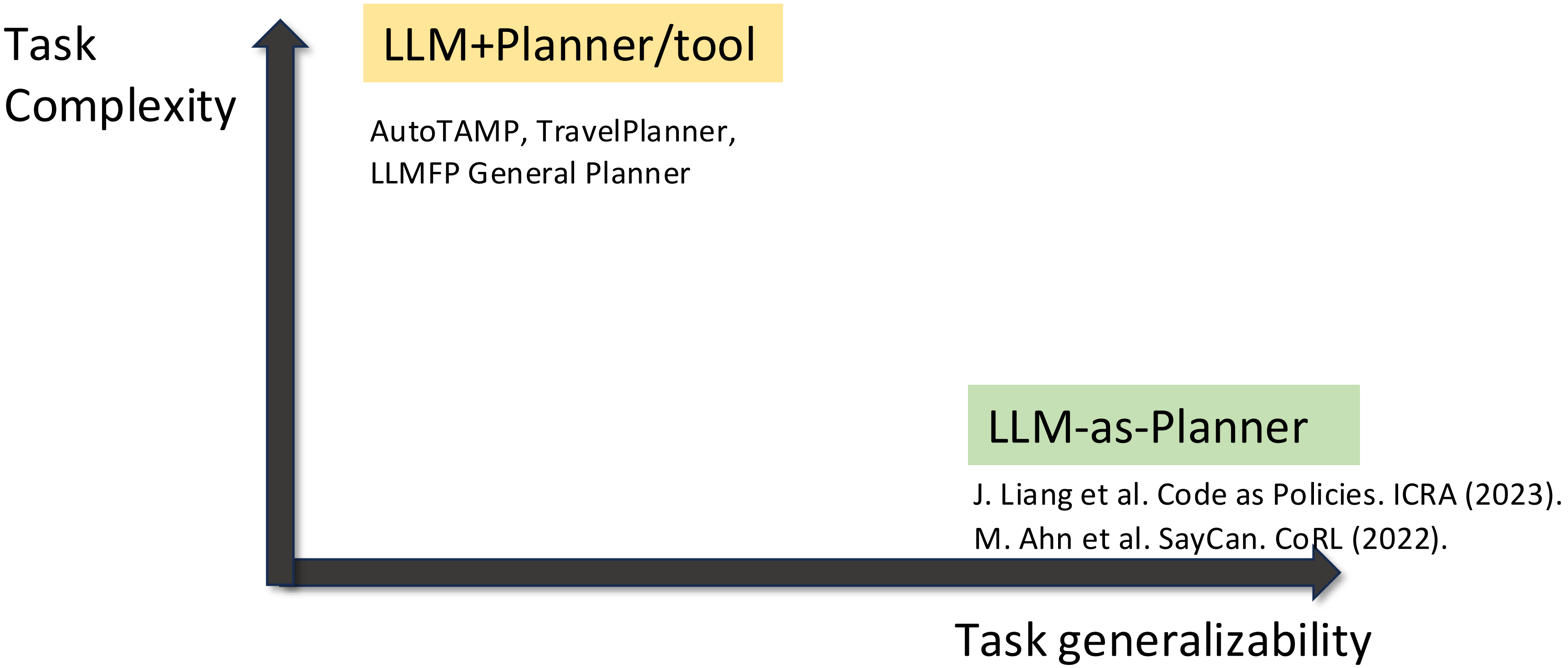




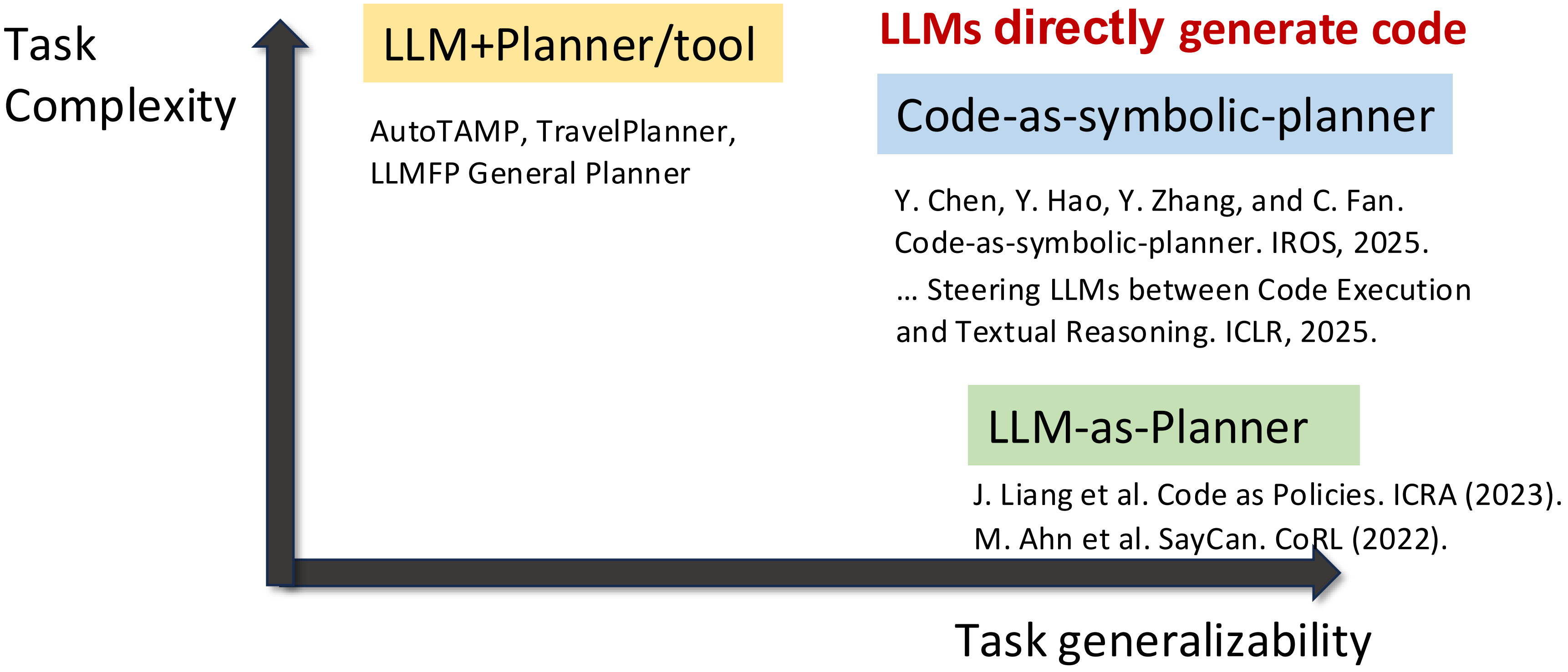
LLM + rigorous solver can make very complex plans [1,2] → LLM + rigorous solver can control dynamic robots [3] → Unifying code, text-based reasoning, and tool use in LLMs [4,5,6]

[1] Y. Hao, Y. Chen, Y. Zhang, and C. Fan. "Large Language Models can solve real-world planning rigorously with formal verification tools." NAACL (2024).
[2] Y. Hao, Y. Zhang, and C. Fan. "Planning anything with rigor: general-purpose zero-shot planning with LLM-based formalized programming." ICLR (2024).
[3] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan. "AutoTAMP: Autoregressive task and motion planning with LLMs as translators and checkers." ICRA (2024).
[4] Y. Chen, Y. Hao, Y. Zhang, and C. Fan. "Code-as-symbolic-planner: Foundation model-based robot planning via symbolic code generation." IROS (2025).
[5] Y. Chen, H. Jhamtani, S. Sharma, C. Fan, C. Wang. "Steering LLMs between Code Execution and Textual Reasoning." ICLR, 2025.
[6] Y. Chen, Y. Hao, Y. Liu, Y. Zhang, and C. Fan. "CodeSteer: Symbolic-Augmented Language Models via Code/Text Guidance." ICML, 2025.

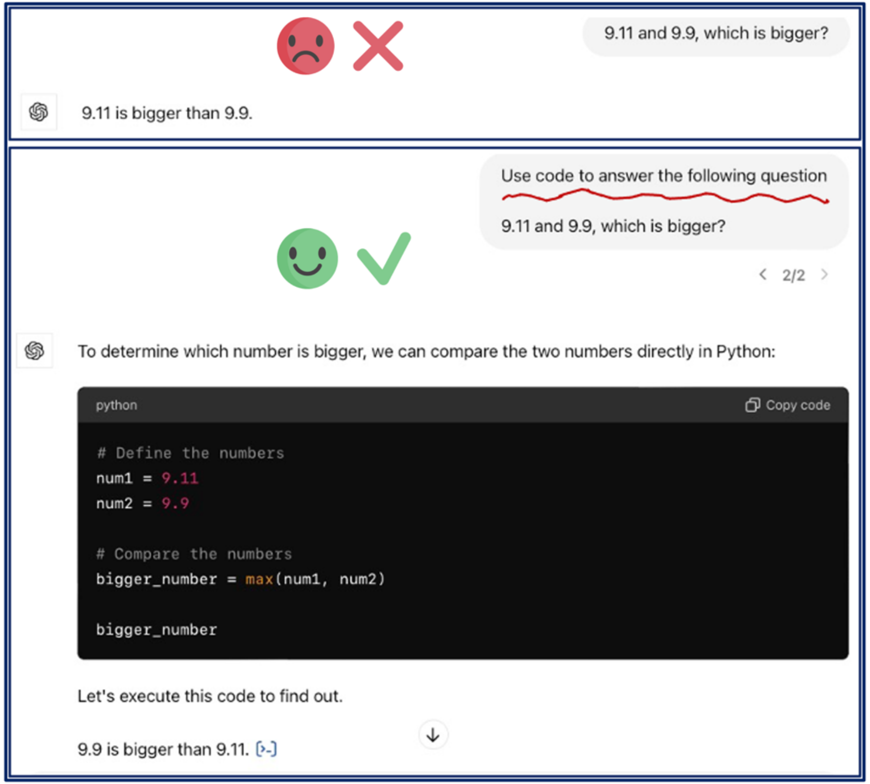
Trade-offs in complexity vs. generalizability

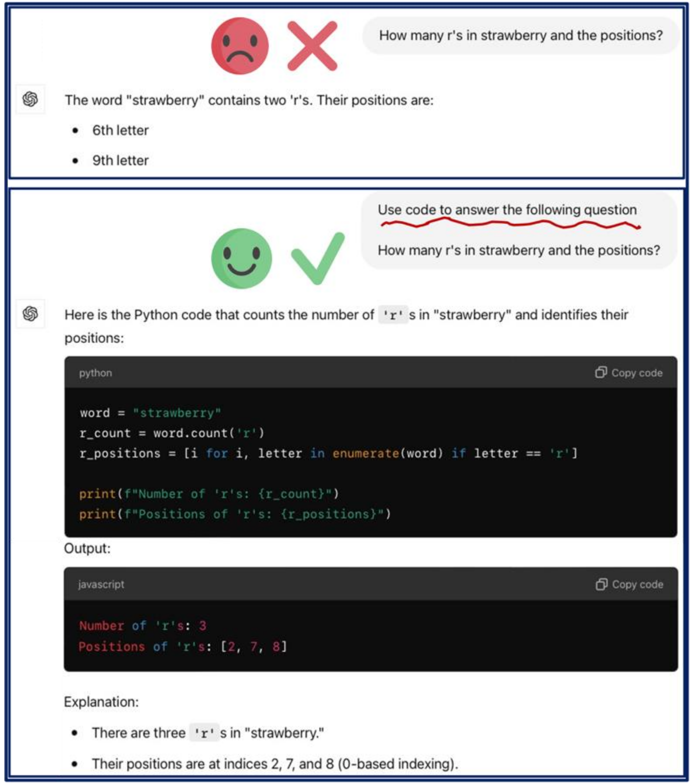


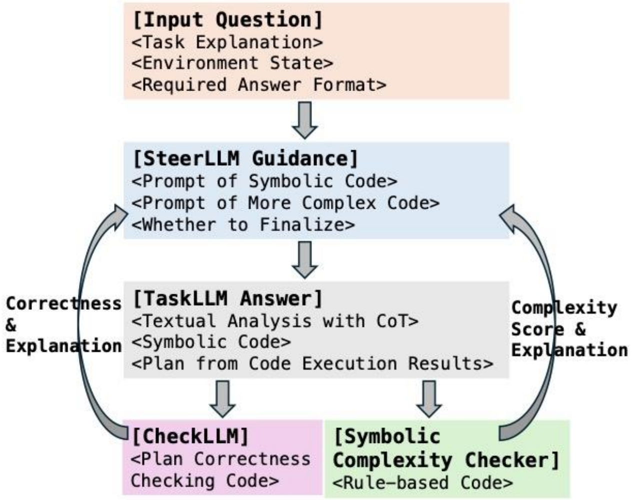
Trade-offs in complexity vs. generalizability



Many tasks are easier with coding

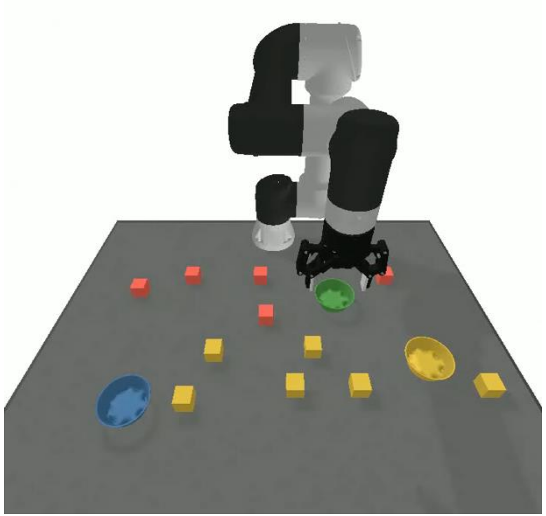


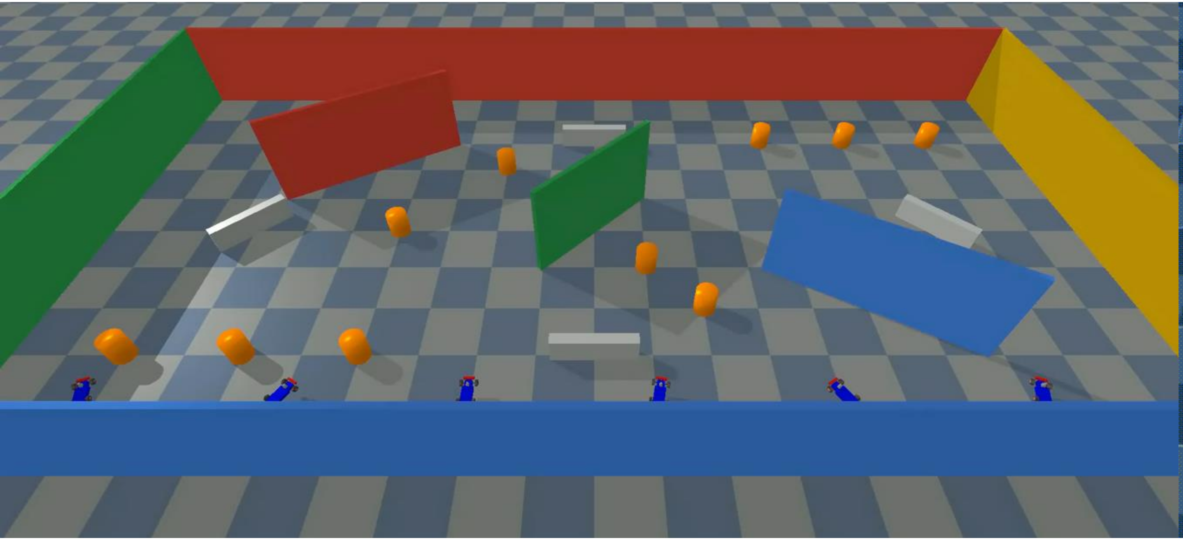


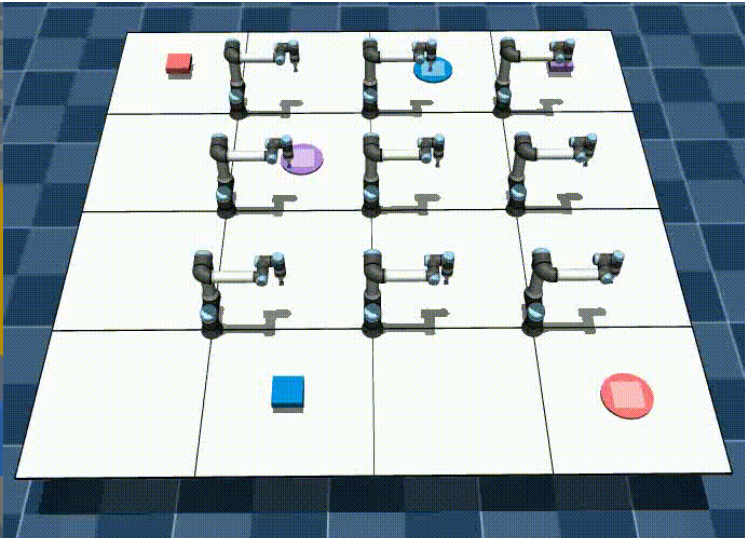


Symbolic Code Answer:To optimize the search...<omitted text>...

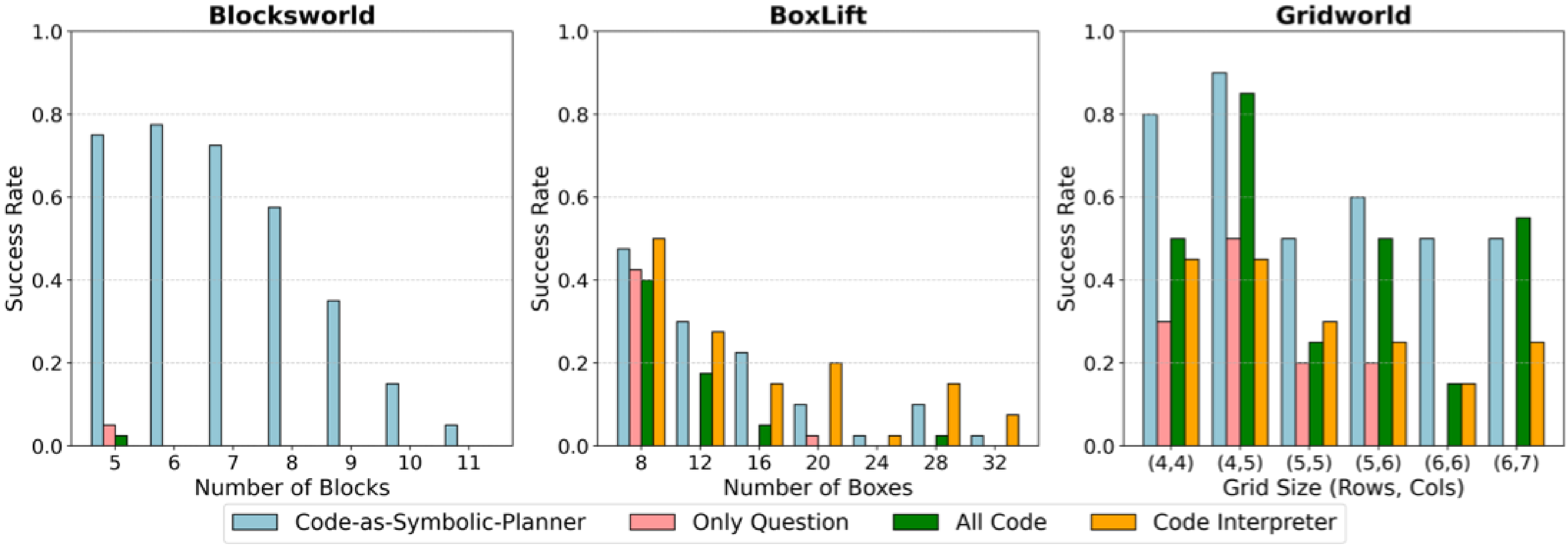
```python
import heapq

def heuristic(state, goal_state):
misplaced_blocks = 0
for stack, goal_stack in zip(state, goal_state):
for block, goal_block in zip(stack, goal_stack):
if block != goal_block:
misplaced_blocks += 1
return misplaced_blocks

def is_goal_state(state, goal_state):
return state == goal_state

def get_possible_moves(state):
moves = []
...<omitted code>...
```





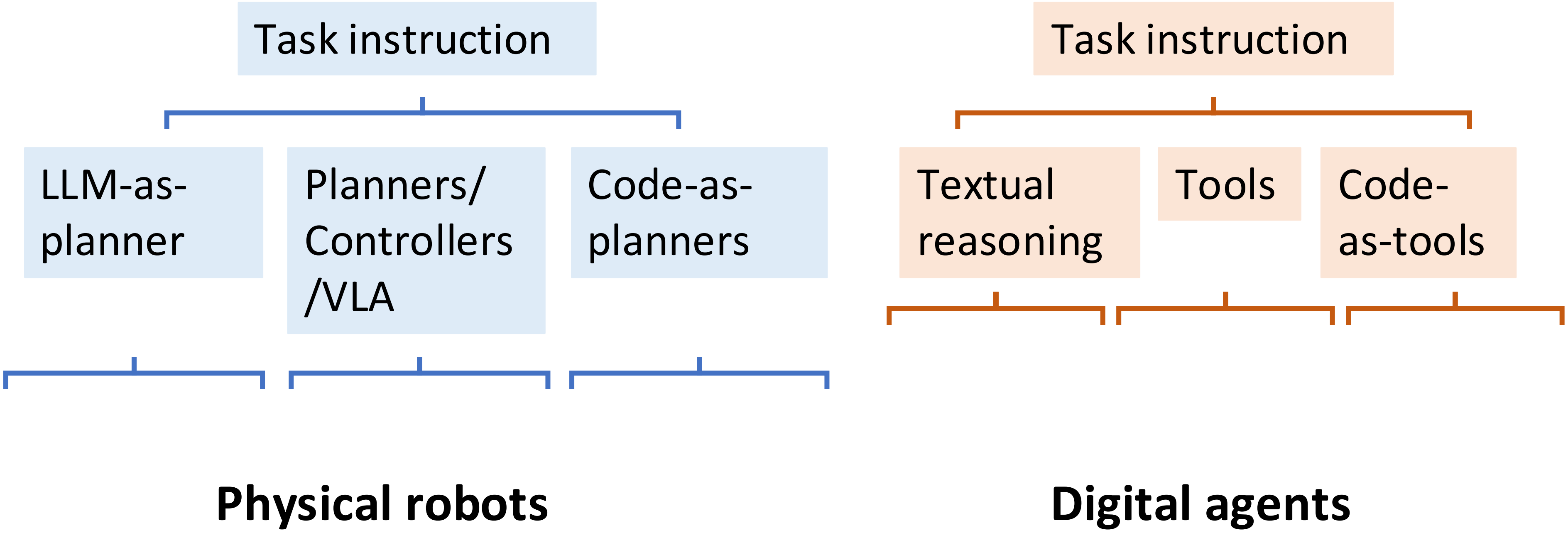


Code-as-symbolic-planner scales better



Y. Chen, Y. Hao, Y. Zhang, and C. Fan. “Code-as-symbolic-planner: Foundation model-based robot planning via symbolic code generation.” IROS (2025).

Can we make LLMs automatically switch between tools and find the correct one?



Trade-offs in complexity vs. generalizability

Task Complexity

LLM+Planner/tool

AutoTAMP, TravelPlanner, LLMFP General Planner

Integrate three models

Y. Chen, Y. Hao, Y. Liu, Y. Zhang, and C. Fan. CodeSteer. ICML 2025.

Code-as-symbolic-planner

Y. Chen, Y. Hao, Y. Zhang, and C. Fan. Code-as-symbolic-planner. IROS, 2025.

… Steering LLMs between Code Execution and Textual Reasoning. ICLR, 2025.

LLM-as-Planner

J. Liang et al. Code as Policies. ICRA (2023).
M. Ahn et al. SayCan. CoRL (2022).

Task generalizability

How do leading LLMs integrate coding and tool-use?

Gemini-2.5-pro Deep Research

Gemini as an Agent: Deep Research

Gemini Deep Research (Gemini Team, Google, 2024) is an agent built on top of the Gemini 2.5 Pro model designed to strategically browse the web and provide informed answers to even the most niche user queries. The agent is optimized to perform task prioritization, and is also able to identify when it reaches a dead-end when browsing. We have massively improved the capabilities of Gemini Deep Research since its initial launch in December 2024. As evidence of that, performance of Gemini Deep Research on the Humanity's Last Exam benchmark (Phan et al., 2025) has gone from 7.95% in December 2024 to the **SoTA score of 26.9% and 32.4% with higher compute** (June 2025).

Grok 4

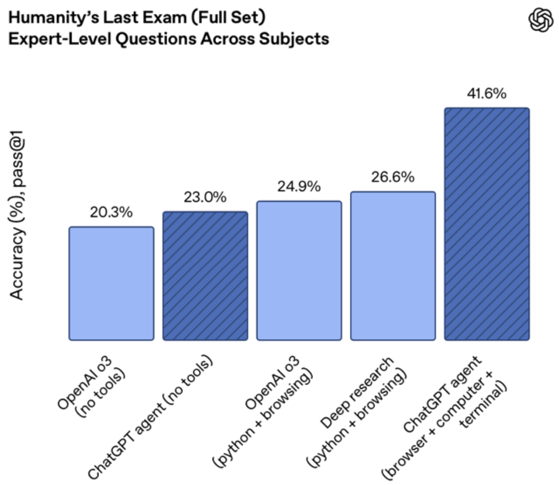
Grok 4 was trained with reinforcement learning to use tools. This allows Grok to augment its thinking with tools like a code interpreter and web browsing in situations that are usually challenging for large language models. When searching for real-time information or answering difficult research questions, Grok 4 chooses its own search queries, finding knowledge from across the web and diving as deeply as it needs to craft a high-quality response.

We also trained Grok to use powerful tools to find information from deep within X. Grok can use advanced keyword and semantic search tools and even view media to improve the quality of its answers.

Still unclear how they integrate!

ChatGPT Agent

On **Humanity's Last Exam***, an evaluation that measures AI's performance across a broad range of subjects on expert-level questions, the model powering ChatGPT agent scores a new pass@1 SOTA at 41.6. Because the agent plans dynamically and chooses its own tools, it can tackle the same task in different ways across runs. When we scaled this with a simple parallel rollout strategy—running up to eight attempts at once and picking the one with the highest self-reported confidence—the agent's HLE score increases to 44.4.



LLMs struggle between code and text

what is the result of 12*56?

The result of 12×56 is 672.

Simple, text

what is the result of 124354536*5607425632?

The result of $124354536 \times 5607425632$ is $697,308,812,621,866,752$. [>_]

Hard, code

Analysis

python   Always show details   Copy code

```python
# Calculate the result of 124354536 * 5607425632
result = 124354536 * 5607425632
result
```

Result

697308812621866752

what is the result of 1243*5607?

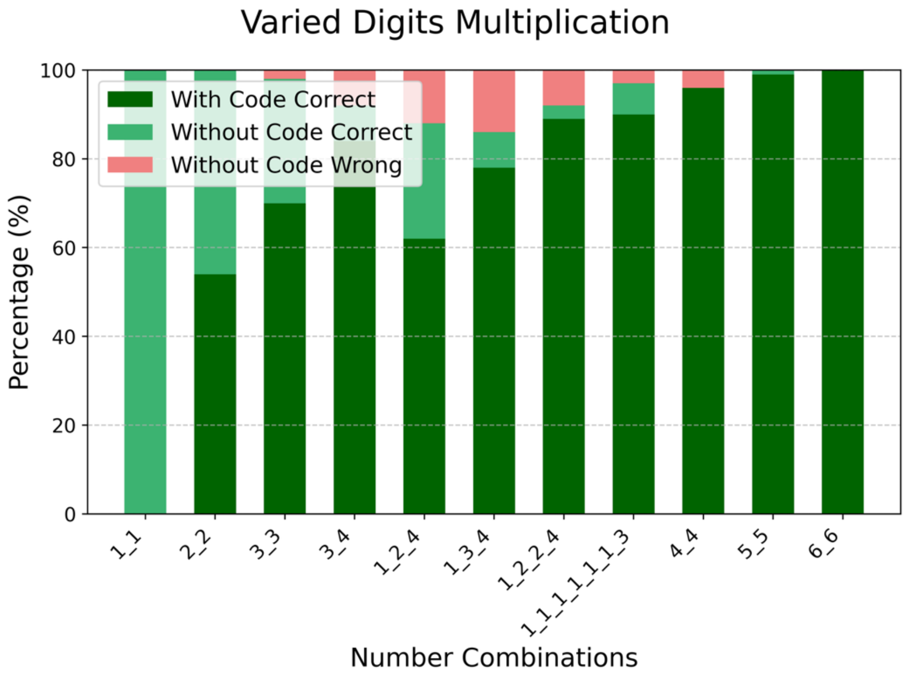
6969501

The result of 1243×5607 is $6,969,801$.

Medium, text

Failure in medium complex questions

GPT-4o, with Code Intepreter



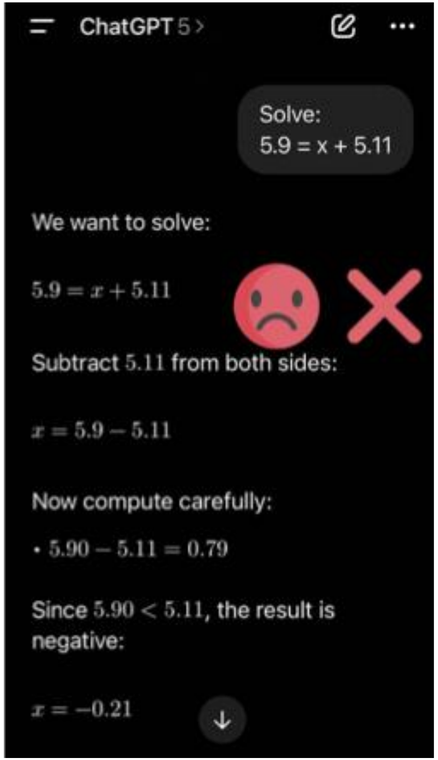
1_1_2_4: 5*3*76*3424

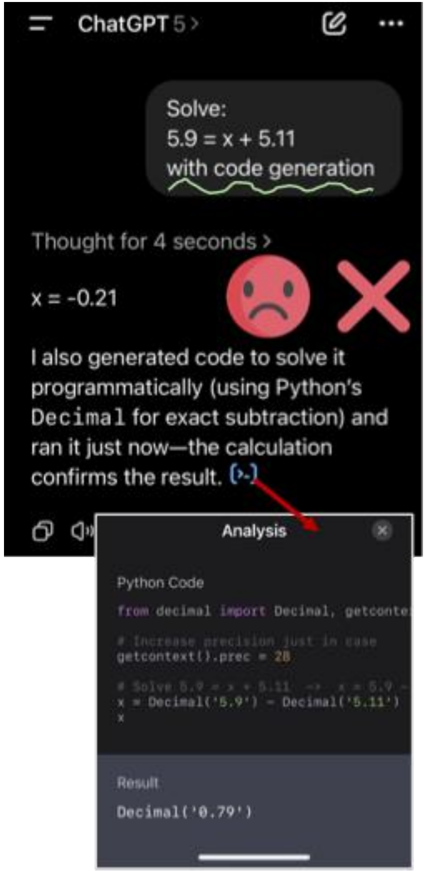
3_4: 643*7564

- LLMs choose **code or text** based on the **complexity** of tasks
- Code interpreter makes **mistakes** in tasks with **medium complexity**

Y. Chen, H. Jhamtani, S. Sharma, C. Fan, and C. Wang. "Steering Large Language Models between Code Execution and Textual Reasoning." ICLR (2025).

GPT-5 still struggles





CodeSteer: guiding LLM code/text generation

Fine-tuning CodeSteerLLM to guide TaskLLM generation

Question: Use numbers and basic arithmetic operations (+ − ∗ /) to obtain 24. Input: 3, 4, 4, 6.

CodeSteer Guidance 1: Utilize your ***programming expertise*** to address tasks…

TaskLLM Answer 1: To solve this …<omitted text>…
Let's try a few combinations:
− (6 ∗ 4) − (3 ∗ 4) = 24
…

```
```python
# Given numbers
a, b, c, d = 1, 3, 4, 6

# Calculate the result
result = (d − a) ∗ (c − b)

# Print the final answer in the required format
print(f"<<<({d} − {a}) ∗ ({c} − {b}) = {result}>>>")
```
```

CodeSteer Guidance 2: The current approach is *incorrect* because... To solve the problem, consider using a ***more complex and systematic approach*** to explore all possible combinations... Implement a ***search algorithm*** that...

TaskLLM Answer 2: …<omitted text>…

```
```python
# filename: find_24.py
from itertools import permutations, product

def find_24(numbers):
    ops = ['+', '−', '∗', '/']
    for num_perm in permutations(numbers):
        for op_perm in product(ops, repeat=3):
            # Genrate all possible ways to
parenthesize the expression
            expressions = [
                f"({num_perm[0]} {op_perm[0]}
…<omitted code>…
```
```

CodeSteer Guidance 3: Return Answer

Y. Chen, Y. Hao, Y. Zhang, and C. Fan. "CodeSteer: Symbolic-Augmented Language Models via Code/Text Guidance." ICML (2025).

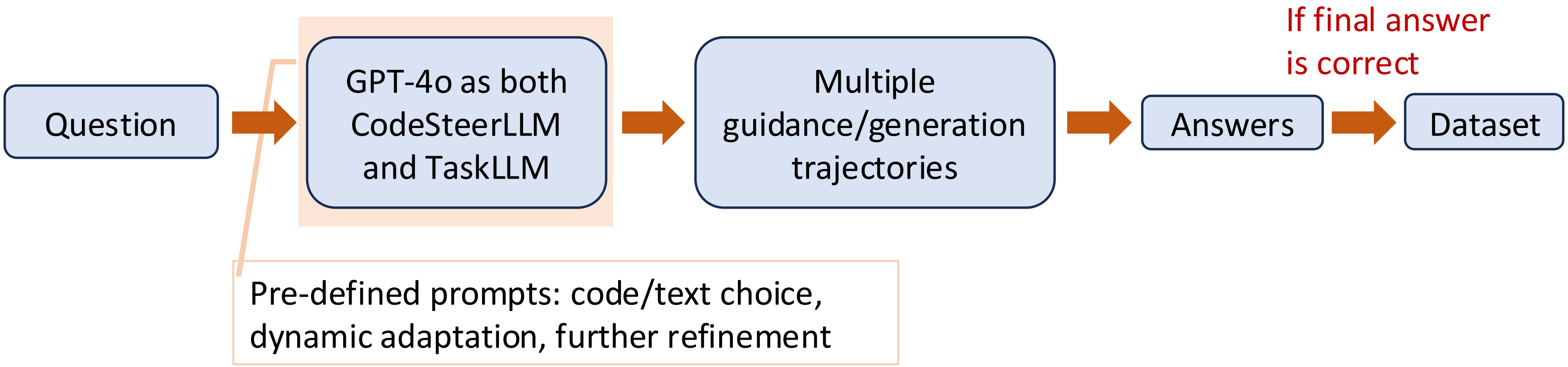
SymBench: 37 task involving symbolic computing

Categories	Tasks	Mathematics	Spatial Reasoning	Logical Reasoning	Order Reasoning	Optimization	Search
	Number Multiplying	✓	✗	✗	✗	✗	✗
	New operator	✓	✗	✗	✗	✗	✗
	Pooling	✓	✓	✗	✗	✗	✗
	Light Puzzles	✗	✓	✗	✗	✗	✗
	Mahjong	✗	✗	✗	✓	✗	✗
	Statistical Counting	✓	✗	✗	✓	✗	✗
	Matrix Transform.	✗	✓	✗	✗	✗	✗
Execution	Pattern Recognition	✗	✓	✗	✗	✗	✓
	String Insertion	✗	✗	✓	✓	✗	✓
	String Deletion &Modi.	✗	✗	✓	✓	✗	✓
	String Synthesis	✗	✗	✓	✓	✗	✓
	Reversi	✗	✓	✗	✗	✗	✗
	String Splitting	✗	✗	✓	✓	✗	✓
	Synthesis Decomposition	✗	✗	✓	✓	✗	✓
	2048	✓	✓	✓	✗	✗	✗
	Game 24	✓	✗	✗	✓	✓	✗
	Path Plan	✗	✓	✗	✓	✗	✓
	Letters	✗	✓	✗	✗	✗	✓
	BoxLift	✗	✗	✓	✗	✓	✗
	BoxNet	✗	✗	✓	✗	✓	✗
	Blocksworld	✗	✓	✓	✗	✓	✗
	Logical Equation	✓	✗	✓	✗	✗	✓
Planning	Logic Puzzle	✓	✓	✗	✗	✗	✓
	Const. Linear Arrange.	✗	✗	✓	✗	✗	✗
	Letter Logic Diagram	✗	✓	✓	✗	✗	✗
	Standard Sudoku	✓	✓	✗	✗	✗	✓
	Eight Queen	✗	✓	✗	✗	✗	✗
	Cryptanalysis	✗	✗	✓	✗	✗	✗
	Combinatorial Calculation	✓	✗	✗	✗	✓	✗
	Permutation and Combina.	✗	✓	✓	✓	✗	✗
	Date Understanding	✗	✗	✓	✗	✗	✗
	Web of Lies	✗	✗	✓	✗	✗	✗
	Logical Deduction	✗	✗	✓	✗	✗	✗
Reasoning	Navigate	✗	✓	✗	✓	✗	✗
	GSM-Hard	✓	✗	✓	✗	✗	✗
	MATH-Geometry	✓	✓	✗	✗	✗	✗
	MATH-Count&Probability	✓	✗	✓	✗	✗	✓

We gather and redevelop 37 symbolic tasks with scripts to synthesize samples.

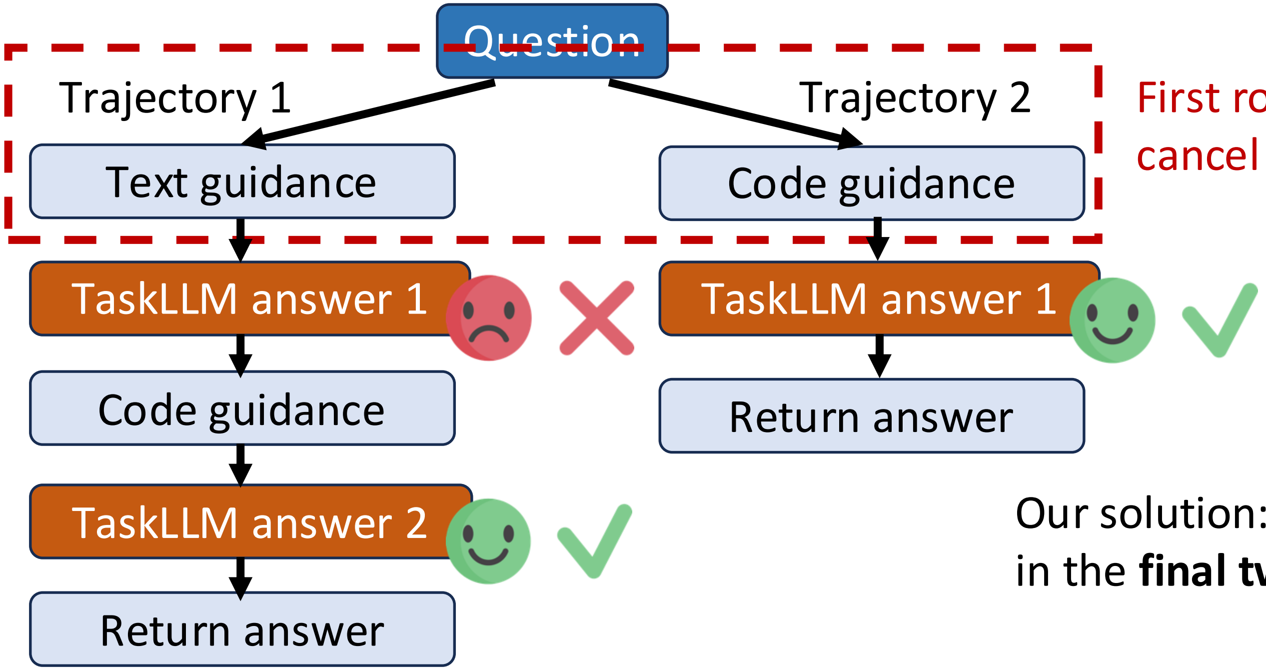
CodeSteerLLM fine-tuning stage 1: SFT

Supervised Fine-tuning (SFT): synthesize 12k multi-round guidance/generation trajectories via rejection sampling on 28 tasks.



Emphasis on the final two rounds for multi-round SFT

Multi-round gradient cancellation issue:



First round answers cancel each other!

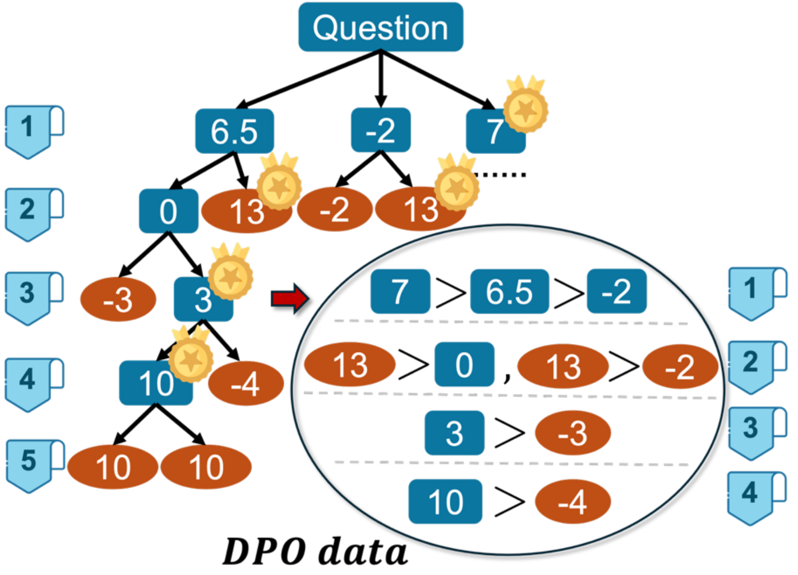
Our solution: data augmentation in the **final two rounds.**

CodeSteerLLM fine-tuning stage 2: DPO

Direct Preference Optimization (DPO): synthesize 5.5k guidance comparison pairs with tree-based sampling.

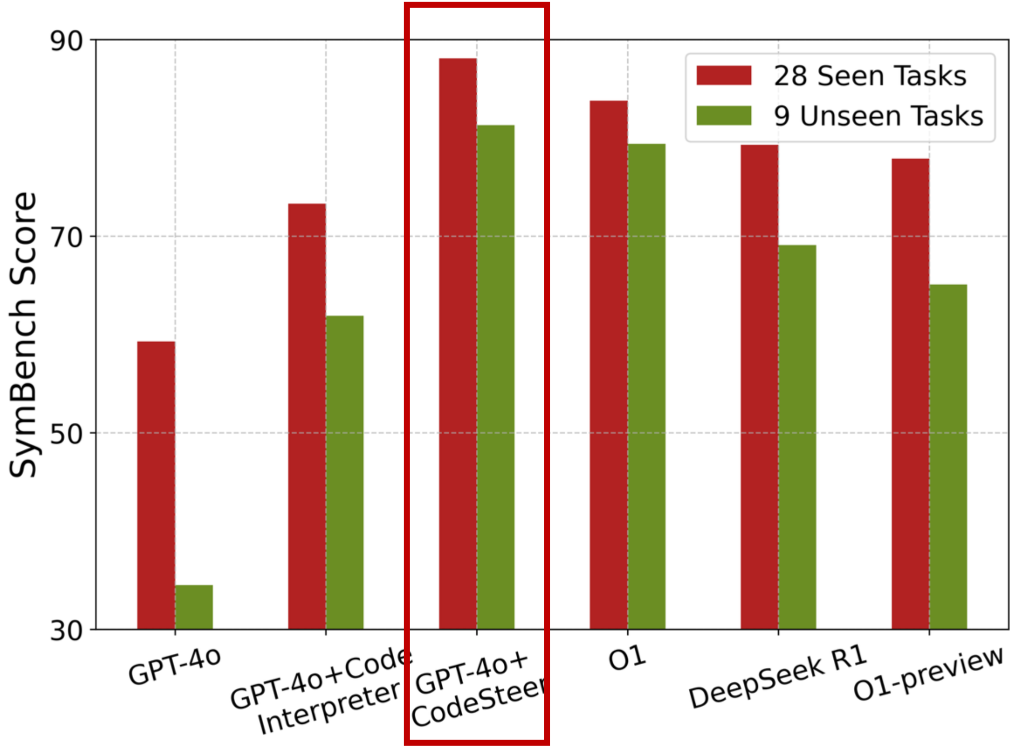
$$\text{Score}_i = \begin{cases} 15 - i & \text{ending round/correct,} \\ -i & \text{ending round/incorrect,} \\ \frac{1}{|C(i)|}\sum_{j \in C(i)} \text{Score}_j & \text{otherwise.} \end{cases} \quad (1)$$

DPO rather than PPO: the trade-off between sampling number and score accuracy.



CodeSteer—guiding LLM code/text generation

GPT-4o + CodeSteer better than OpenAI o1 and DeepSeek R1 on 37 tasks involving symbolic computing.



One step further: one unified LLM with Code Interpreter

Question: Blocksworld Task: Initial State: stack1: D stack2: G C A F stack3: B I E H Goal State: stack1: A F H stack2: C G I stack3: B D E Please provide a series of moves to reach the goal state. Your answer:

R1-CI-14B: Let me solve this Blocksworld problem step by step. 1) First, let me create a Python script to help simulate and find the solution. The script will: – Represent the stacks – Validate moves (only top blocks can be moved) – Find a path from initial to goal state – Use a search algorithm to find the solution. Use symbolic computing or an efficient search algorithm, we can implement a depth-first search (DFS) approach …<omitted text>…

```python
def is_goal_state(state, goal_state):
return state == goal_state

def get_possible_moves(state):
moves = []
for i, stack in enumerate(state):
if stack: # If the stack is not empty
block = stack[-1] # Get the top block
for j in range(len(state)):
if i != j: # Move to a different stack
new_state = [list(s) for s in state] # Deep copy of the state
…<omitted code>…
```

Code Execution Results: TimeoutExpired: Command 'run.py' timed out after 60 seconds.

The current approach using depth-first search (DFS) is a step in the right direction, but it seems to be **inefficient, leading to a timeout**. To optimize the search for a solution in the Blocksworld task, we can implement the A* search algorithm. A* search is more efficient than DFS …<omitted text>…

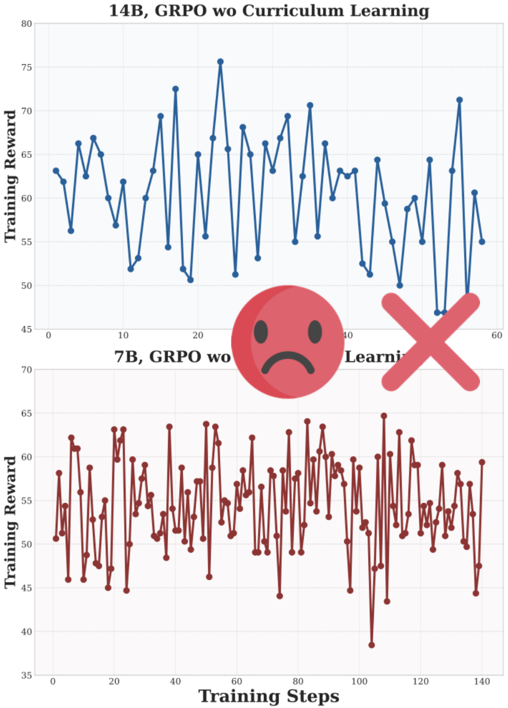
```python
import heapq

def get_possible_moves(state):
moves = []
for i, stack in enumerate(state):
if stack: # If the stack is not empty
block = stack[-1] # Get the top block
for j in range(len(state)):
if i != j: # Move to a different stack
new_state = [list(s) for s in state] # Deep copy of the state
new_state[i].pop() # Remove the block from the current stack
…<omitted code>…
```

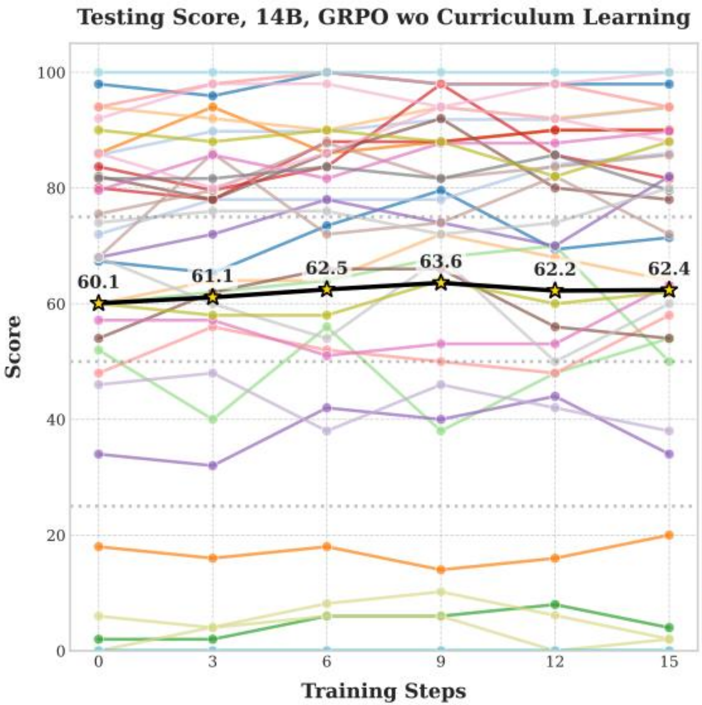
Reasoning LLM determines when to generate code during the textual reasoning process.

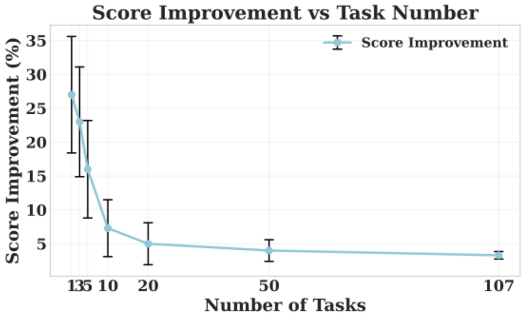
Y. Chen, Y. Liu, J. Zhou, J. Wang, Y. Hao, Y. Zhang, and C. Fan. "R1-Code-Interpreter: LLMs Reason with Code via Supervised and Multi-stage Reinforcement Learning," under review.

Reinforcement learning is not as effective as in other tasks

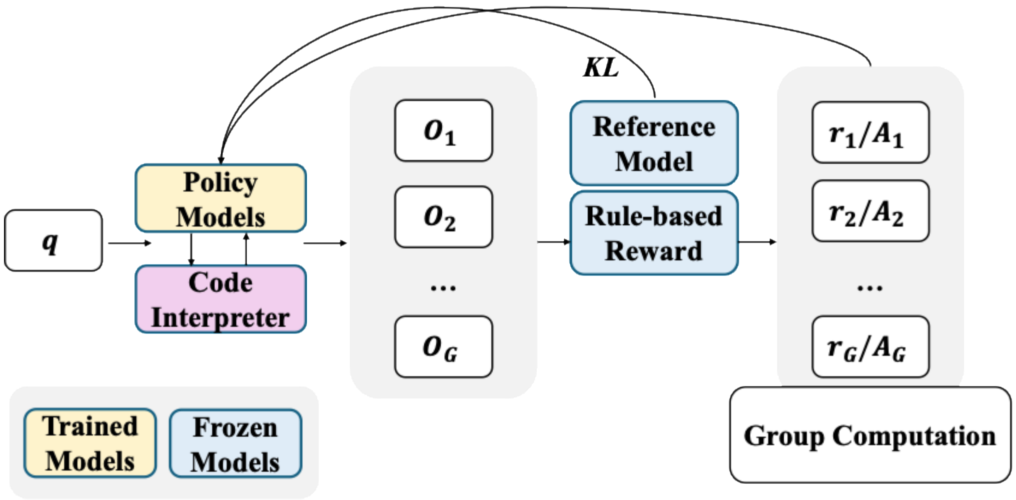
Supervised Fine-Tuning (SFT) + Group Relative Policy Optimization (GRPO) on 144 tasks







Gradient vanishing in low-potential training samples



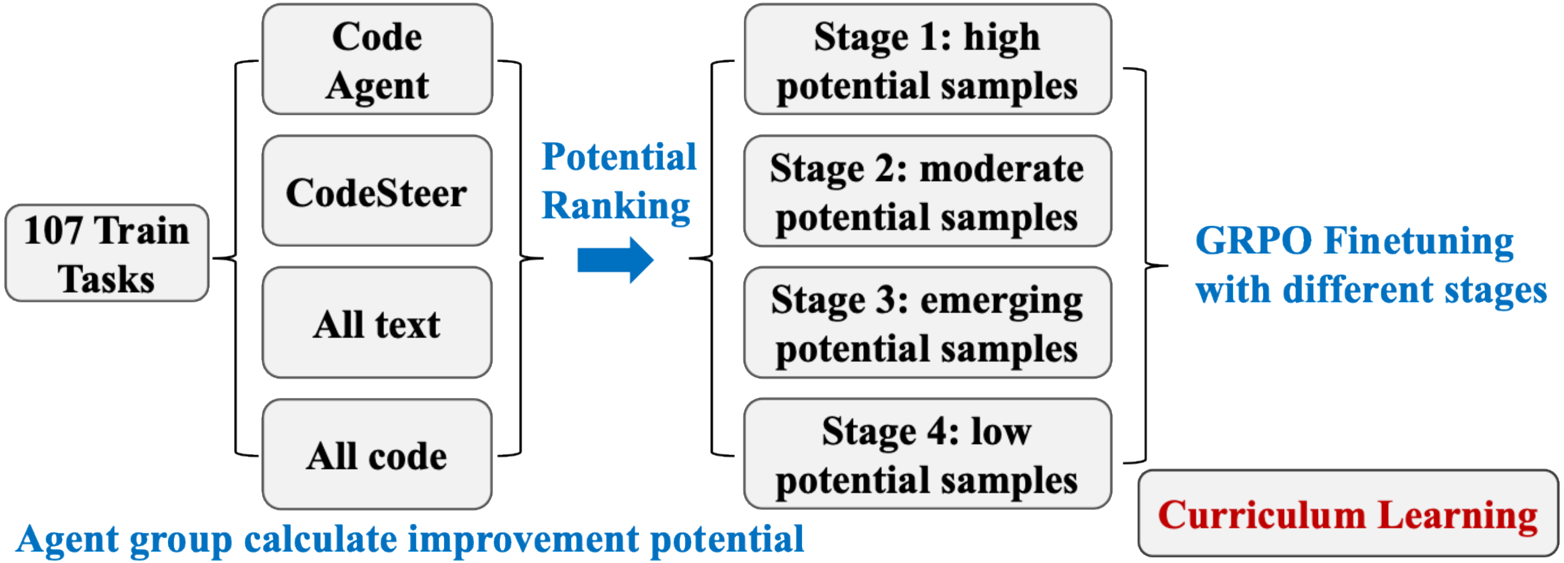
In too simple or hard questions, advantages are zero!

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim D,\ y_{1:G}\sim\pi_{\text{old}}(\cdot|x;\mathcal{C})}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\Big(\frac{\pi_\theta\big(y_{i,t}\mid x, y_{i,<t};\mathcal{C}\big)}{\pi_{\text{ref}}\big(y_{i,t}\mid x, y_{i,<t};\mathcal{C}\big)}\,\hat{A}_{i,t},\right.$$

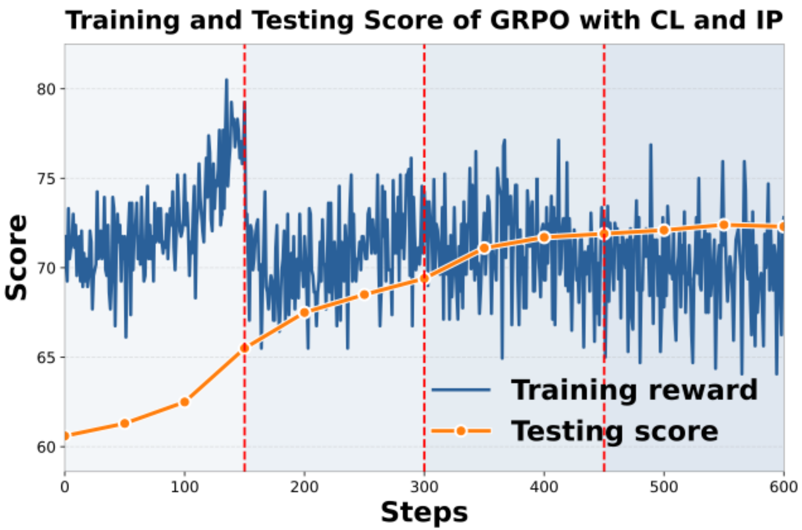
$$\left.\text{clip}\Big(\frac{\pi_\theta\big(y_{i,t}\mid x, y_{i,<t};\mathcal{C}\big)}{\pi_{\text{ref}}\big(y_{i,t}\mid x, y_{i,<t};\mathcal{C}\big)}, 1-\epsilon,\ 1+\epsilon\Big)\hat{A}_{i,t}\Big)\right] - \beta\,\mathbb{D}_{\text{KL}}[\pi_\theta \,\|\, \pi_{\text{ref}}]$$

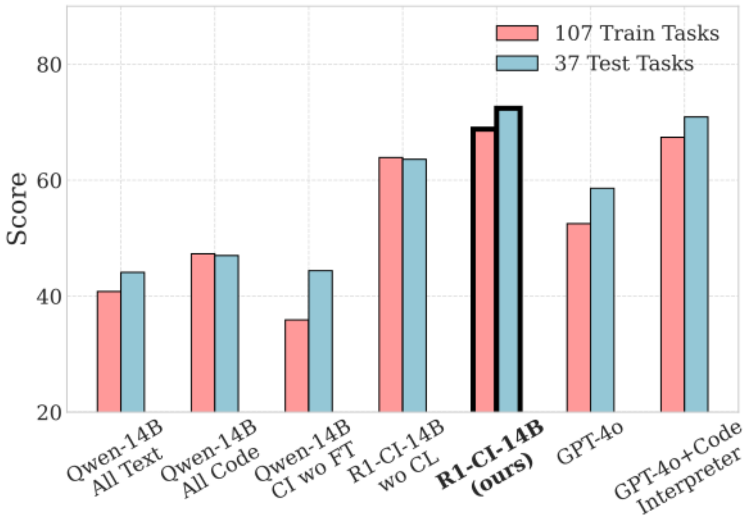
Our solution:
Multi-stage curriculum learning with potential measurement

Tasks with ~50% correctness rate have higher potential



Multi-stage curriculum learning with potential measurement





Our models have gained significant public attention

Hugging Face

R1-Code-Interpreter-3B
Total downloads
58,968 (all time)

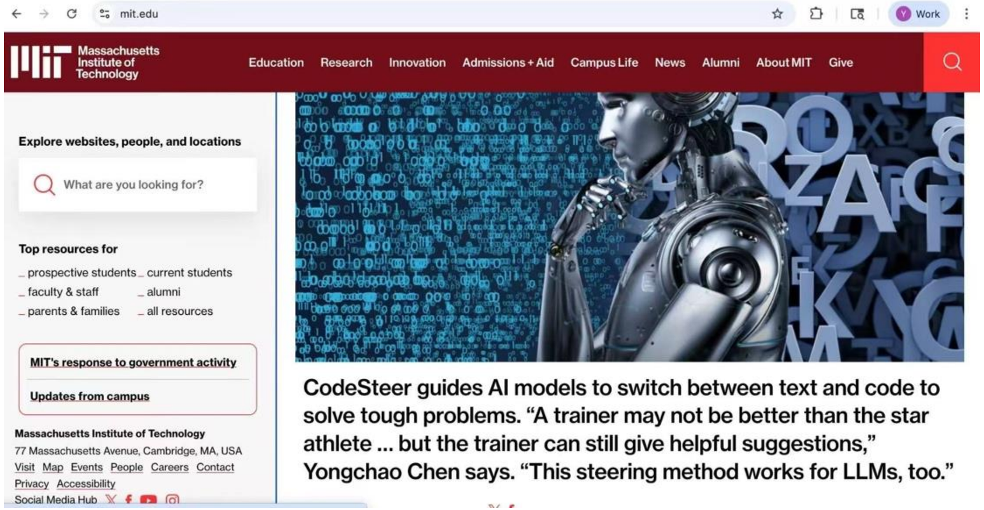
R1-Code-Interpreter-7B
Total downloads
59,817 (all time)

R1-Code-Interpreter-14B
Total downloads
55,255 (all time)

R1-Code-Interpreter-Data
Total downloads
34,146 (all time)

SymBench
Total downloads
31,127 (all time)

CodeSteer-v1
Total downloads
2,192

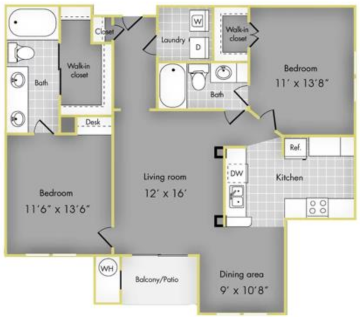


How to write a good prompt for robotic tasks?

An example prompt

Your task is to interact with a virtual household simulator to achieve a clearly defined goal. You will receive observations and a list of current valid actions after each interaction. Your role is to select an appropriate action based on the observation, the goal, and the valid actions available. Ensure that any objects ('{obj}') and receptacles ('{cep}') you mention in your response are present in the observation provided. Your planned action must be one of the current valid actions.

To successfully complete the task, please adhere to the following optimized guidelines:
1. **Understand the Goal**: Always keep the goal at the forefront of your decision-making process. Each action you select should be a strategic step towards accomplishing this goal.
2. **Use Observations**: Analyze the observations to gain a comprehensive understanding of the environment's current state, including the location and status of objects and receptacles.
3. **Valid Action Selection**: Strictly choose your actions from the provided list of current valid actions. Do not attempt any actions that are not listed as valid for the current situation.
4. **State Tracking and Changes**: Keep a mental model of the environment's state and update it with each action's outcome. Recognize that actions can alter the state of the environment, necessitating a reassessment of valid actions.
5. **Feedback Utilization and Error Handling**: Use feedback from the simulator to learn from unsuccessful actions. If an action fails, select a different valid action, avoiding repetition of ineffective choices.
6. **Logical Action Sequencing**: Plan your actions in a logical order, ensuring that each step is dependent on the previous one and brings you closer to the goal.
7. **Inventory and Object Management**: Regularly use the 'inventory' action to monitor the objects you have acquired. Utilize this inventory to inform and plan your future actions.
8. **Specificity in Actions**: Be specific when interacting with objects and receptacles to avoid ambiguity and ensure clarity in your actions.
……



Many robotic tasks are long horizons with multi-steps, multi-goals, and multi-constraints in dynamic environments.

How to write a good but lengthy prompt with complex constraints, goals, and metrics?

While humans struggle to write optimal prompts, they are good at providing feedback about LLM outputs.

Existing prompt optimization - single-step tasks

Jailbreak
Detect if the message is a jailbreak attack, i.e. an attempt by a user to break through an AI system's protections.

BIG-Bench
Extract the disease or condition from the sentence, if any is mentioned.

Linguistics
Rate the semantic similarity of two input sentences on a scale of 0 - definitely not to 5 - perfectly.

Question Answering
Work out an answer to the common- sense reasoning question above. If there are multiple people or perspec- tives involved, try considering them one at a time. Next, answer yes or no."

MATH-GSM8K
Let's solve the problem step-by-step and calculate the required total value correctly.

Arithmetic Reasoning
You are a mathematician. Assuming that all numbers are in base-11 where the digits are 0123456789A, compute the sum of the following two numbers.

Factuality
What would have been the consequences if Alan Turing had not cracked the Enigma code during World War II in terms of the war's duration, impact, and the Holocaust? Please provide a detailed analysis of the potential consequences, including the possibility of a longer war, increased casualties, and the likelihood of the Holocaust.

Previous methods focus on single-step tasks, which usually comprise only one or two sentences.

Wei et al. Chain-of-thought prompting elicits reasoning in large language models, Neurips 2022.

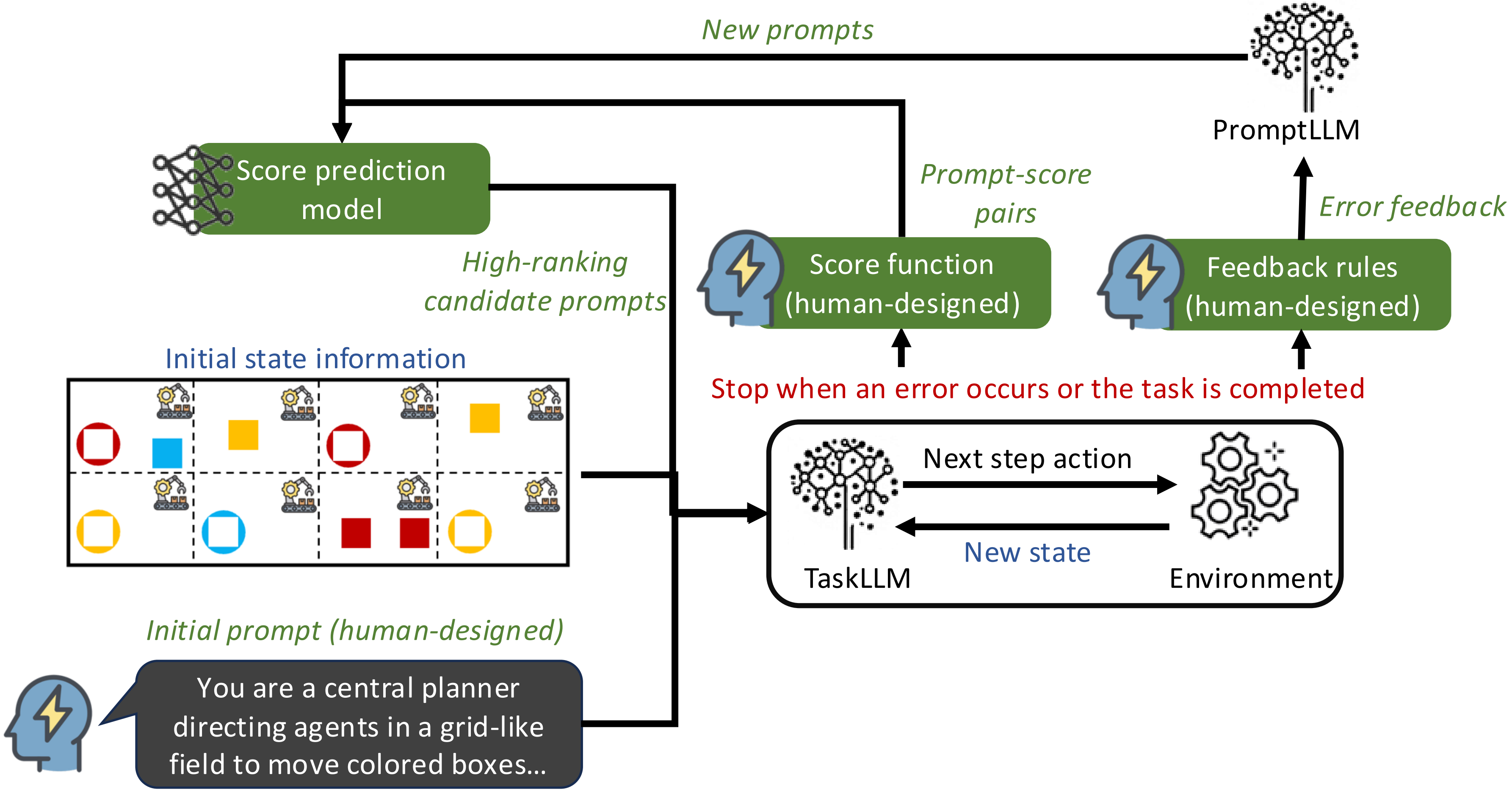
Zhou et al. Large language models are human-level prompt engineers, ICLR 2023.

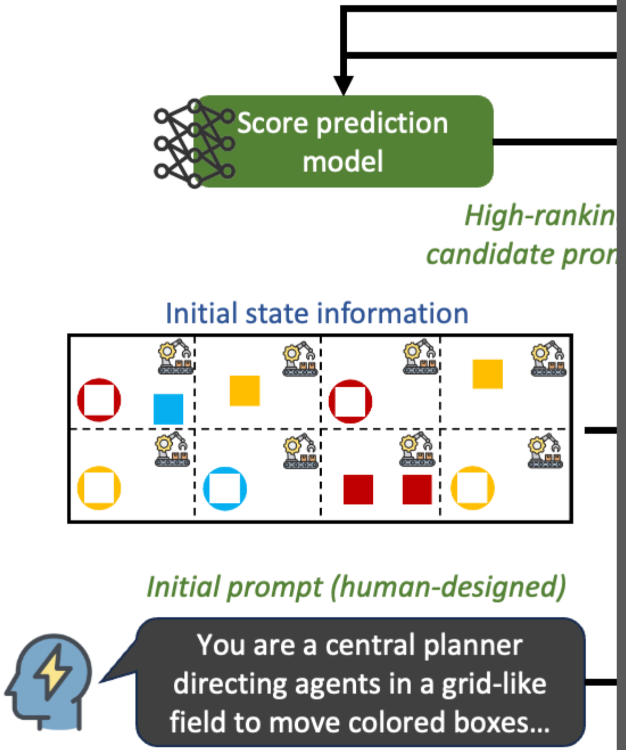
Pryzant et al. Automatic Prompt Optimization with "Gradient Descent" and Beam Search, EMNLP 2023.

Guo et al. Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers, ICLR 2024.

Wang et al. Promptagent: Strategic planning with language models enables expert-level prompt optimization, ICLR 2024.

Learn a prompts scoring model from human feedback





Initial prompt

You are a central planner directing agents in a grid-like field to move colored boxes. Each agent is assigned to a 1x1 square and can only interact with objects in its area. Agents can move a box to a neighboring square or a same-color target. Each square can contain many targets and boxes.

The squares are identified by their center coordinates, e.g., square[0.5, 0.5]. Actions are like: move(box_red, target_red) or move(box_red, square[0.5, 0.5]).

Your task is to instruct each agent to match all boxes to their color-coded targets. After each move, agents provide updates for the next sequence of actions. Your job is to coordinate the agents optimally.

Specify your action plan in this format: {"Agent[0.5, 0.5]":"move(box_blue, square[0.5, 1.5])", "Agent[1.5, 0.5]":"move...}. Include an agent only if it has a task next.

Feedback rules

1) Syntactic error: "Here is the prompt of task description:{prompt_task_explain}. The response {response} is in the wrong format."

2) Stuck in the loop: "Here is the prompt and response from previous two rounds:
Round1: {state 1 and action 1}, environment feedback after executing the plan:{env_act_feedback_1};
Round2: {state 2 and action 2}, environment feedback after executing the plan:{env_act_feedback_2}.
Here is the initial response generated by the task planning LLM agent based on the concatenated prompt of task description and current state: {response}
Based on previous rounds, here is the error feedback from humans: It seems the LLM is stuck in the current situation, always repeating the same answer. The task is stuck too, no box is placed successfully in recent rounds."

3) Collision: "Your response is: {response}. The action in the response leads to collision."

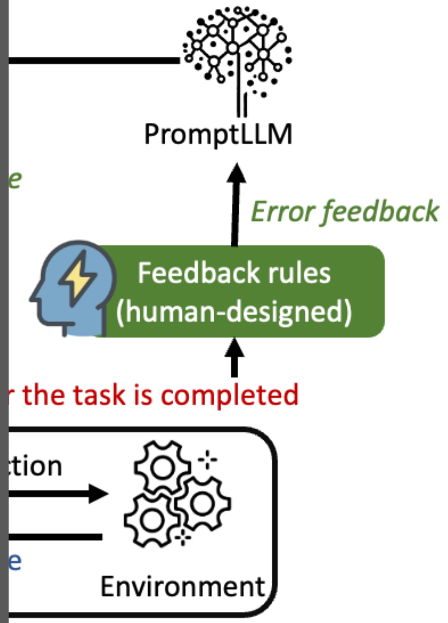
4) Move out of the grid: "Your response is: {response}. The action in the response leads to move out of the grid."

5) Wrong picking up order: "Your response is: {response}. The action in the response leads to wrong order of picking up goals. Do remember pick up goal_0 first, then goal_1, goal_2, and so on."

6) Failure over query time limit: "The task is not completed over the query time limit."

7) Invalid action: "Your response is: {response}. The action in the response is invalid action."

8) Wrong object action: "Your response is: {response}. The action in the response is trying to refer to nonspecific or nonexist vehicle (truck, airplane)."



Optimized prompt

You are a central planner tasked with directing agents in a grid-like field to move colored boxes to their corresponding color-coded targets. Each agent occupies a 1x1 square and can only interact with objects within its square. Agents can move a box to an adjacent square or directly to a target square of the same color. A square may contain multiple boxes and targets.

The squares are identified by their center coordinates (e.g., square[0.5, 0.5]). Actions are formatted as: move(box_color, destination), where box_color is the color of the box and destination is either a target of the same color or an adjacent square.

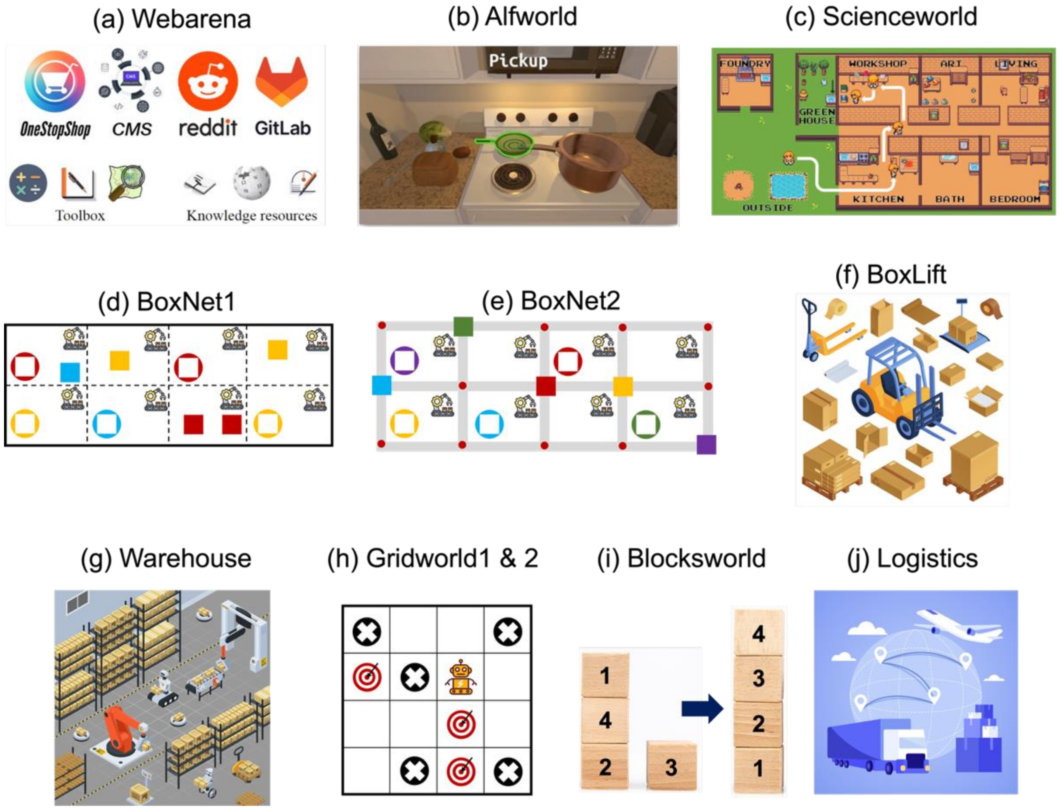
Your objective is to create an action plan that instructs each agent to match all boxes to their color-coded targets in the most efficient manner. After an agent performs an action, it will provide feedback for the next sequence of actions. You must coordinate the agents based on the updated grid state.

Please adhere to the following rules when specifying your action plan:
1. **Single Action per Agent**: Assign only one action to each agent at a time. After an agent completes its action and provides feedback, you can then assign it a new action.
2. **Unique Agent Keys**: Use unique keys for each agent in the JSON format action plan. The key should be the agent's coordinates in the format "Agent[x, y]".
3. **Prioritize Matching Boxes to Targets**: Always prioritize actions that will match a box to its target over moving a box to an adjacent square.
4. **Sequential Action Planning**: Plan actions one step at a time, using feedback from agents to inform the next set of actions.
5. **Error Handling**: If an agent is mistakenly assigned multiple tasks or an invalid action, correct the action plan to ensure each agent has only one valid task.
6. **Clear Formatting**: Ensure the action plan is clearly formatted in JSON, with each agent's action specified as a key-value pair.
7. **Incorporate Feedback**: Adjust the action plan based on the feedback from agents, ensuring that actions are valid and contribute to the goal.
8. **Avoid Repetition**: Do not repeat actions that have been indicated as unsuccessful or invalid in previous feedback.
9. **Conflict Resolution**: Ensure that no two agents are assigned actions that would interfere with each other.
10. **Optimize Efficiency**: Aim to minimize the number of moves required to match all boxes with their targets.

Here is the format for your action plan:
```json
{
  "Agent[0.5, 0.5]": "move(box_blue, target_blue)",
  "Agent[1.5, 0.5]": "move(box_red, square[1.5, 0.5])",
  ...
}
```
Include an agent in the action plan only if it has a task to perform next. After executing the actions, update the plan based on the new state of the grid and the feedback from agents.

The optimized prompts have more structured details. This optimization process can achieve a 3X improvement in success rate.

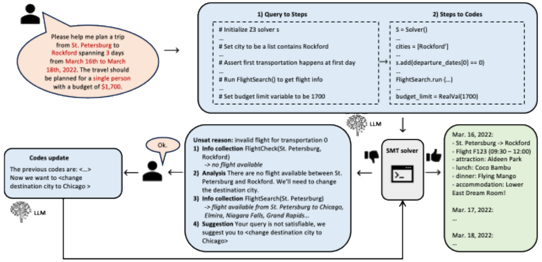
Testing on a variety of different tasks

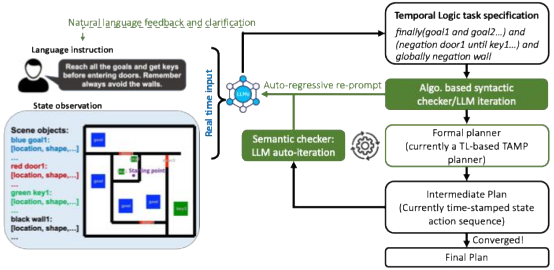


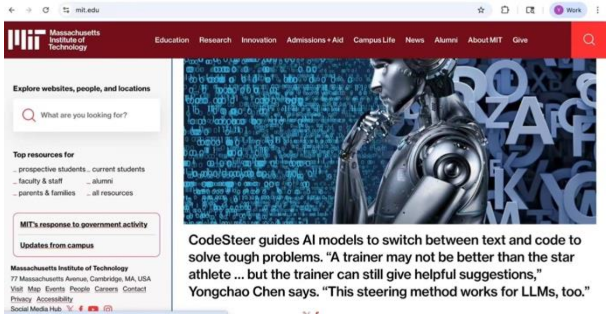
The auto prompt optimization can work for multi-step tasks in both virtual and real environments that require logical, geometrical, scientific, and commonsense reasoning capabilities.

The optimized prompts can achieve an average of 10.6%-29.3% improvement over the current best methods on five LLMs (GPT-3.5, GPT-4, Claude-3, Open-Mixtral-8x7B, Mixtral-large), respectively

Summary







LLM + rigorous solver can make very complex plans [1,2] → LLM + rigorous solver can control dynamic robots [3] → Unifying code, text-based reasoning, and tool use in LLMs [4,5,6]

[1] Y. Hao, Y. Chen, Y. Zhang, and C. Fan. "Large Language Models can solve real-world planning rigorously with formal verification tools." NAACL (2024).

[2] Y. Hao, Y. Zhang, and C. Fan. "Planning anything with rigor: general-purpose zero-shot planning with LLM-based formalized programming." ICLR (2024).

[3] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan. "AutoTAMP: Autoregressive task and motion planning with LLMs as translators and checkers." ICRA (2024).

[4] Y. Chen, Y. Hao, Y. Zhang, and C. Fan. "Code-as-symbolic-planner: Foundation model-based robot planning via symbolic code generation." IROS (2025).

[5] Y. Chen, H. Jhamtani, S. Sharma, C. Fan, C. Wang. "Steering LLMs between Code Execution and Textual Reasoning." ICLR, 2025.

[6] Y. Chen, Y. Hao, Y. Liu, Y. Zhang, and C. Fan. "CodeSteer: Symbolic-Augmented Language Models via Code/Text Guidance." ICML, 2025.