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

Subbarao Kambhampati [1] **Karthik Valmeekam** [1] **Lin Guan** [1] **Mudit Verma** [1] **Kaya Stechly** [1]
Siddhant Bhambri [1] **Lucas Saldyt** [1] **Anil Murthy** [1]

Extracting the best of the paper's ideas + my own insights

Presented by:

John Tan Chong Min

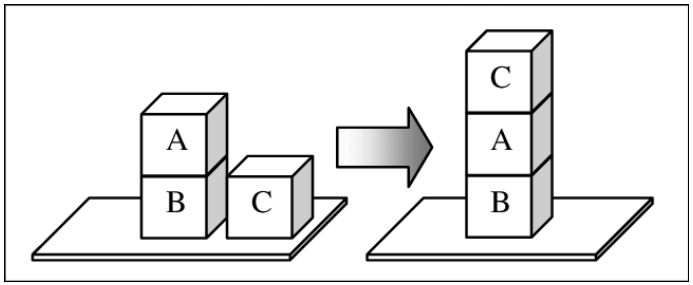
Main Message

- LLMs **cannot plan themselves** but can play a variety of constructive roles in solving planning tasks – especially as approximate knowledge sources and candidate plan generators in so-called **LLM-Modulo** Frameworks, where they are used in **conjunction with external sound model-based verifiers**.

LLM cannot plan well without environment feedback

Domain	Method	Instances correct					
		GPT-4o	**GPT-4-Turbo**	**Claude-3-Opus**	**LLaMA-3 70B**	**Gemini Pro**	**GPT-4**
Blocksworld (BW)	One-shot	170/600 (28.33%)	138/600 (23%)	289/600 (48.17%)	76/600 (12.6%)	68/600 (11.3%)	206/600 (34.3%)
	Zero-shot	213/600 (35.5%)	241/600 (40.1%)	356/600 (59.3%)	205/600 (34.16%)	3/600 (0.5%)	210/600 (34.6%)
Mystery BW (Deceptive)	One-shot	5/600 (0.83%)	5/600 (0.83%)	8/600 (1.3%)	15/600 (2.5%)	2/500 (0.4%)	26/600 (4.3%)
	Zero-shot	0/600 (0%)	1/600 (0.16%)	0/600 (0%)	0/600 (0%)	(0/500) (0%)	1/600 (0.16%)

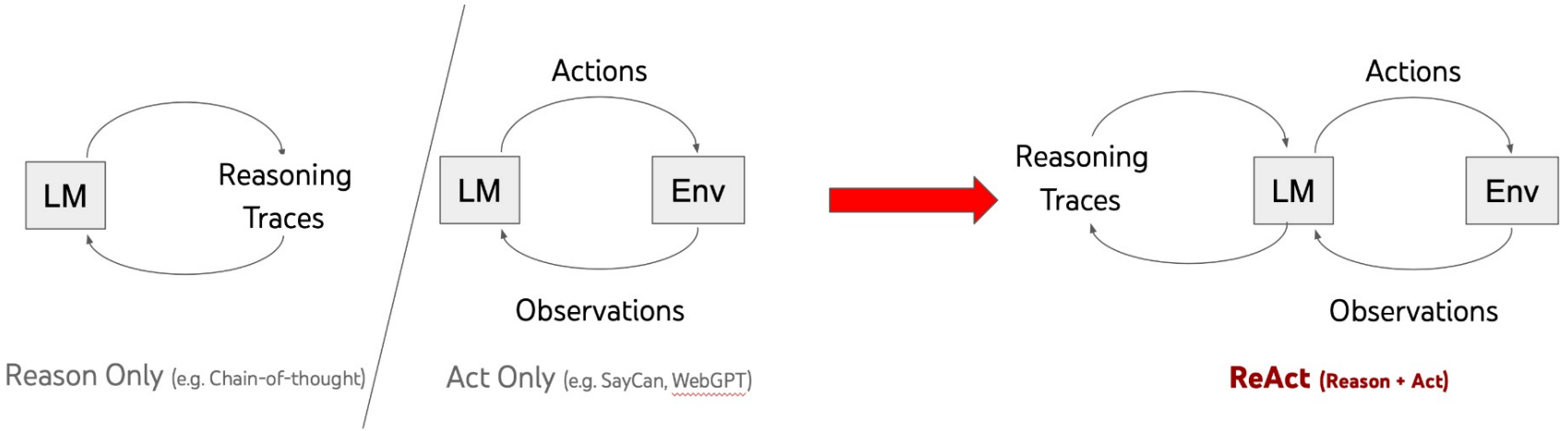
Table 1. Results of state-of-the-art LLMs GPT-4o, GPT-4-Turbo, Claude-3-Opus, Gemini Pro and LLaMA-3 70B for Plan Generation with prompts in natural language.



LLM Modulo. 2024. Subbarao et. al.

ReAct Framework is good only for short trajectories / forgiving environments

- Relies on having environment feedback and correcting from there



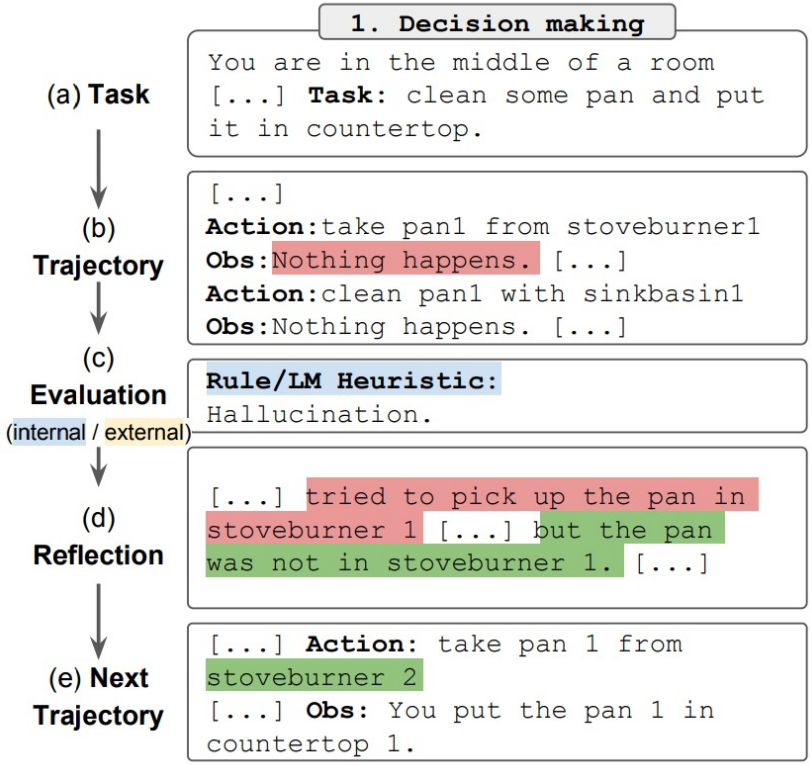
ReAct: Synergizing Reasoning and Acting in Language Models. 2023. Yao et. al.

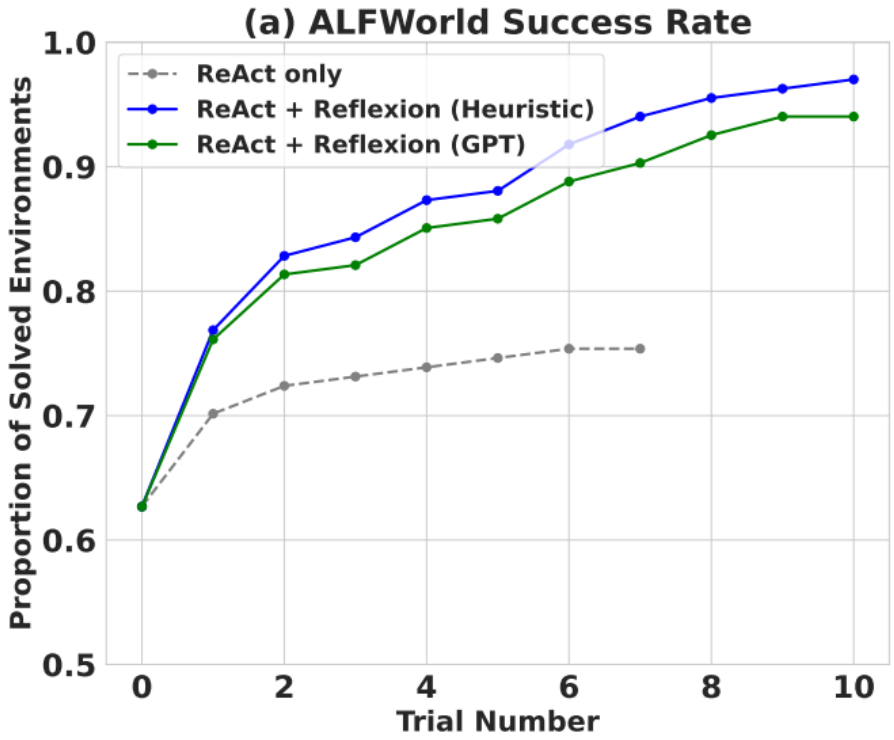
Planning

To think about how to act without needing to actually act it out

Has an evolutionary advantage as the organism will catch prey / reduce being preyed upon if planning goes well

Reflection may help LLMs with planning





Reflexion. 2023. Noah et. al.

Main bottleneck: Self-Critique for LLMs

- LLMs cannot critique well if LLM is not an expert in that domain
- LLMs cannot critique well if it involves an abstraction space not semantically meaningful
  - E.g. Math
  - E.g. counting letters, such as ’r’ in strawberry
- Best if interfaced with something that is rule-based and is known to work well, e.g. PDDL planner, calculator

Strawberry self-critique is not working (gpt-4o-mini)

```
res = strict_json(system_prompt = 'Count the number of r in strawberry',
            user_prompt = '',
            output_format = {'Explanation': 'type: str',
                             'Number of r': 'type: int'},
            llm = llm)
```

```
res
```

```
{'Explanation': 'The word "strawberry" contains the letter "r" twice.',
 'Number of r': 2}
```

```
res2 = strict_json(system_prompt = 'Critique the solution to "Count the number of r in strawberry"',
            user_prompt = res,
            output_format = {'Critique': 'type: str',
                             'Modified answer': 'type: str',
                             'Number of r': 'type: int'},
            llm = llm)
```

```
res2
```

```
{'Critique': 'The explanation is correct, but it could be more concise and directly answer the question without add
itional wording.',
 'Modified answer': 'The word "strawberry" contains the letter "r" 2 times.',
 'Number of r': 2}
```

Adding a rule-based verifier helps (gpt-4o-mini)

Critic to count letters

```
res = strict_json(system_prompt = 'Count the number of r in strawberry.',
            user_prompt = '',
            output_format = {'Explanation': 'type: str',
                             'Number of r': 'type: int'},
            llm = llm)
```

```
res
```

```
{'Explanation': 'The word "strawberry" contains the letter "r" twice.',
 'Number of r': 2}
```

```
count = "strawberry".count('r')
```

```
res2 = strict_json(system_prompt = f'Critique the solution to "Count the number of r in strawberry". Count of r from code: {count}',
            user_prompt = res,
            output_format = {'Critique': 'type: str',
                             'Modified answer': 'type: str',
                             'Number of r': 'type: int'},
            llm = llm)
```

```
res2
```

```
{'Critique': 'The count of r is incorrect; the letter "r" appears three times in "strawberry".',
 'Modified answer': 'The word "strawberry" contains the letter "r" three times.',
 'Number of r': 3}
```

Even better with Agentic Tool Use (gpt-4o-mini)

```python
def count_letters(word: str, letter: str) -> str:
    '''Counts letter in the word'''
    return word.count(letter)
```

```python
count_letters('strawberry', 'r')
```

```
3
```

```python
agent = Agent('Generalist Agent', 'Does anything', llm = llm).assign_functions(count_letters)
```

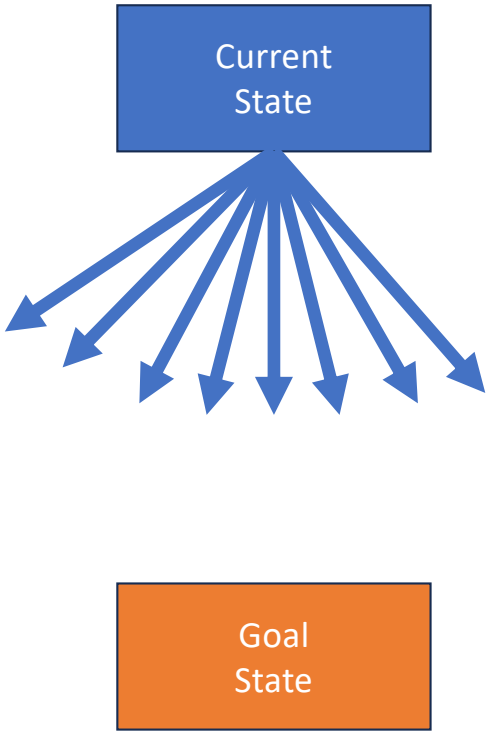
```python
agent.run('Count the number of r in strawberry')
```

```
Observation: No subtasks have been completed yet for the task of counting the number of r in strawberry.
Thoughts: To complete the task, I need to count the occurrences of the letter "r" in the word "strawberry". This can be done using the count_letter
s function.
Subtask identified: Count the occurrences of the letter "r" in the word "strawberry" using the count_letters function.
Calling function count_letters with parameters {'letter': 'r', 'word': 'strawberry'}
> {'output_1': 3}

Observation: The task to count the number of occurrences of the letter "r" in the word "strawberry" has been completed, resulting in a count of 3.
Thoughts: Since the counting task is complete and the result is available, the next step is to finalize the task and present the output to the use
r.
Subtask identified: End Task
Task completed successfully!
```

The hard problem of planning

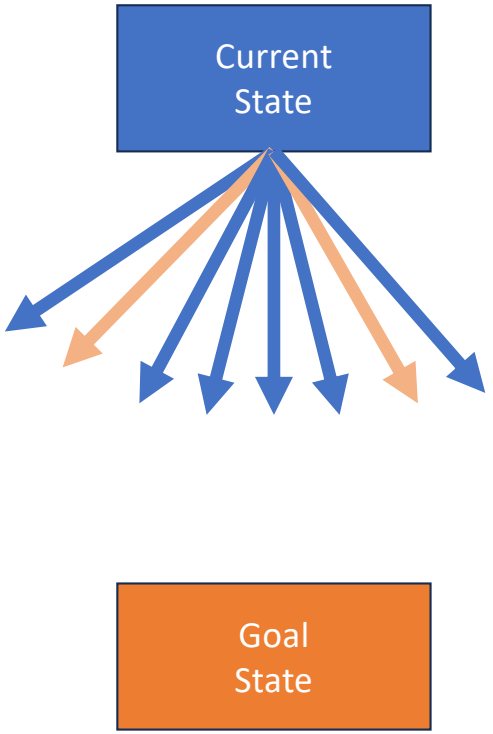
- Many possible actions to take, not sure which would reach the goal
- Brute-force search might be infeasible due to compute/time required



The problem of LLMs for planning

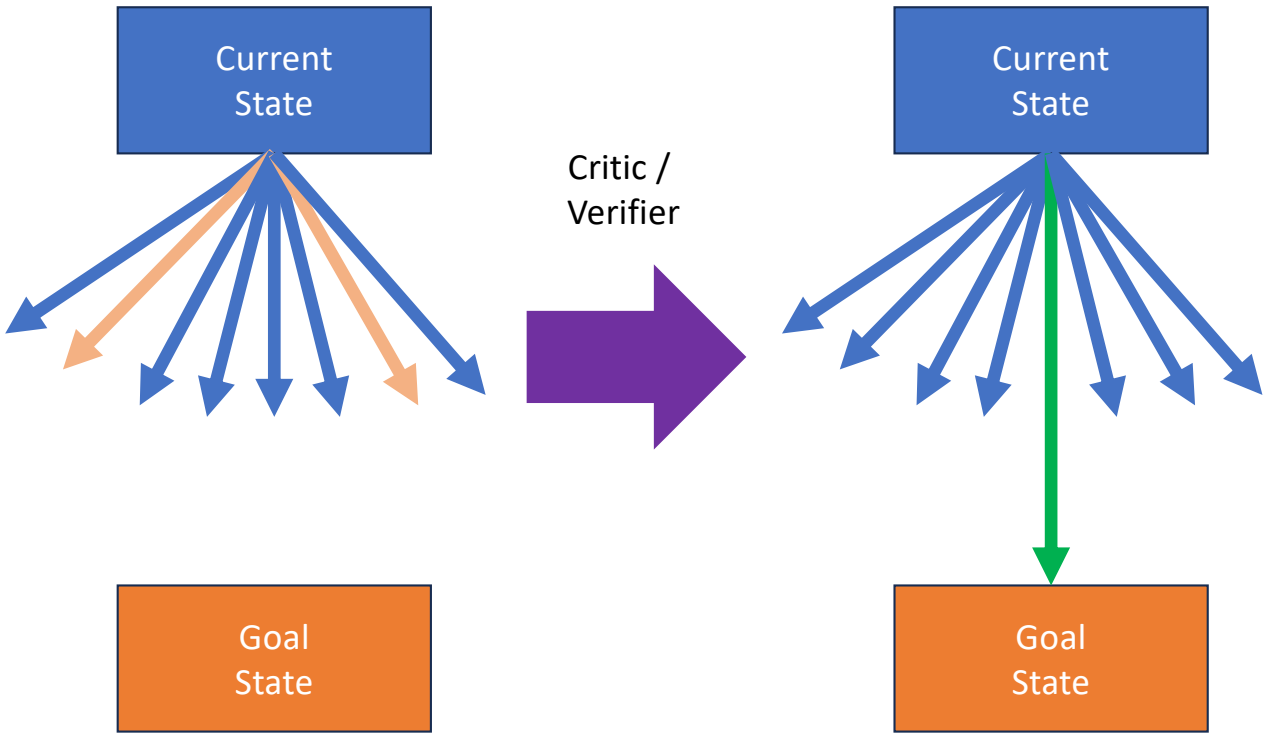
- LLMs may not have grounding in the real world and cannot generate the right plans! (red color ones are chosen)

- LLM is good if the number of steps is few, and the steps have some semantic meaning between current state and goal state

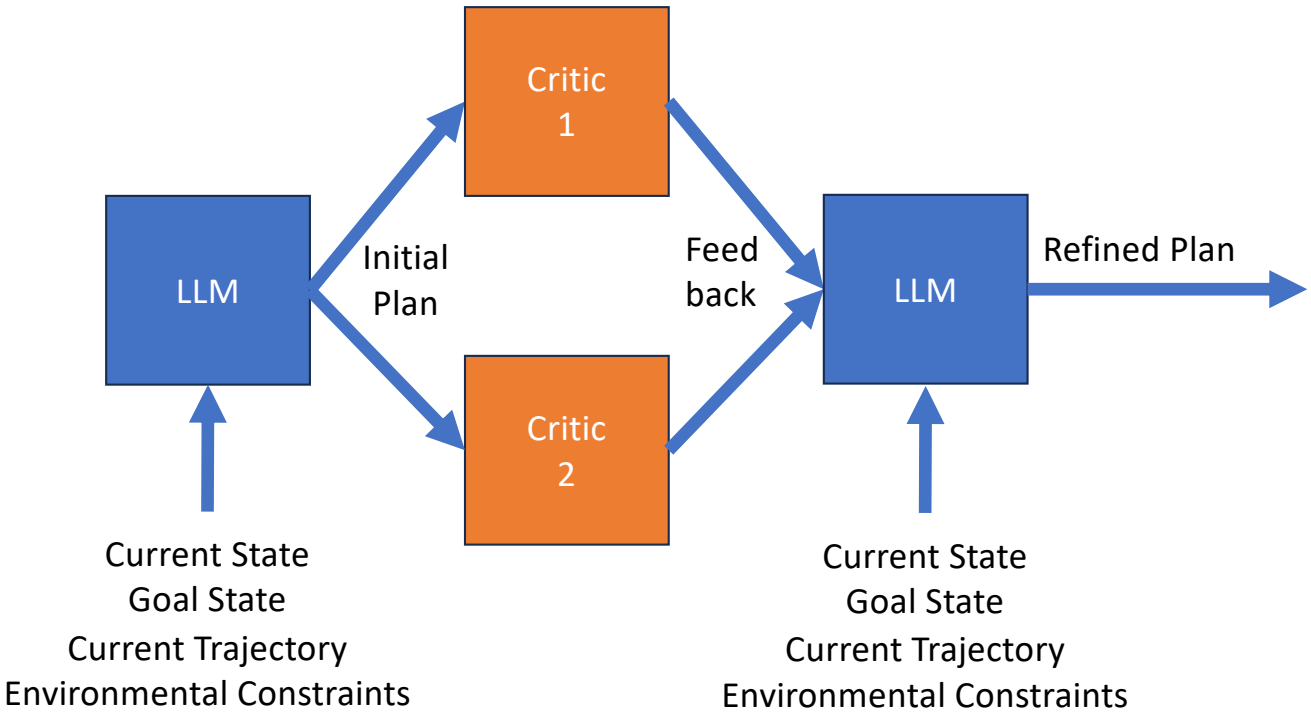


Should we still use an LLM for planning?

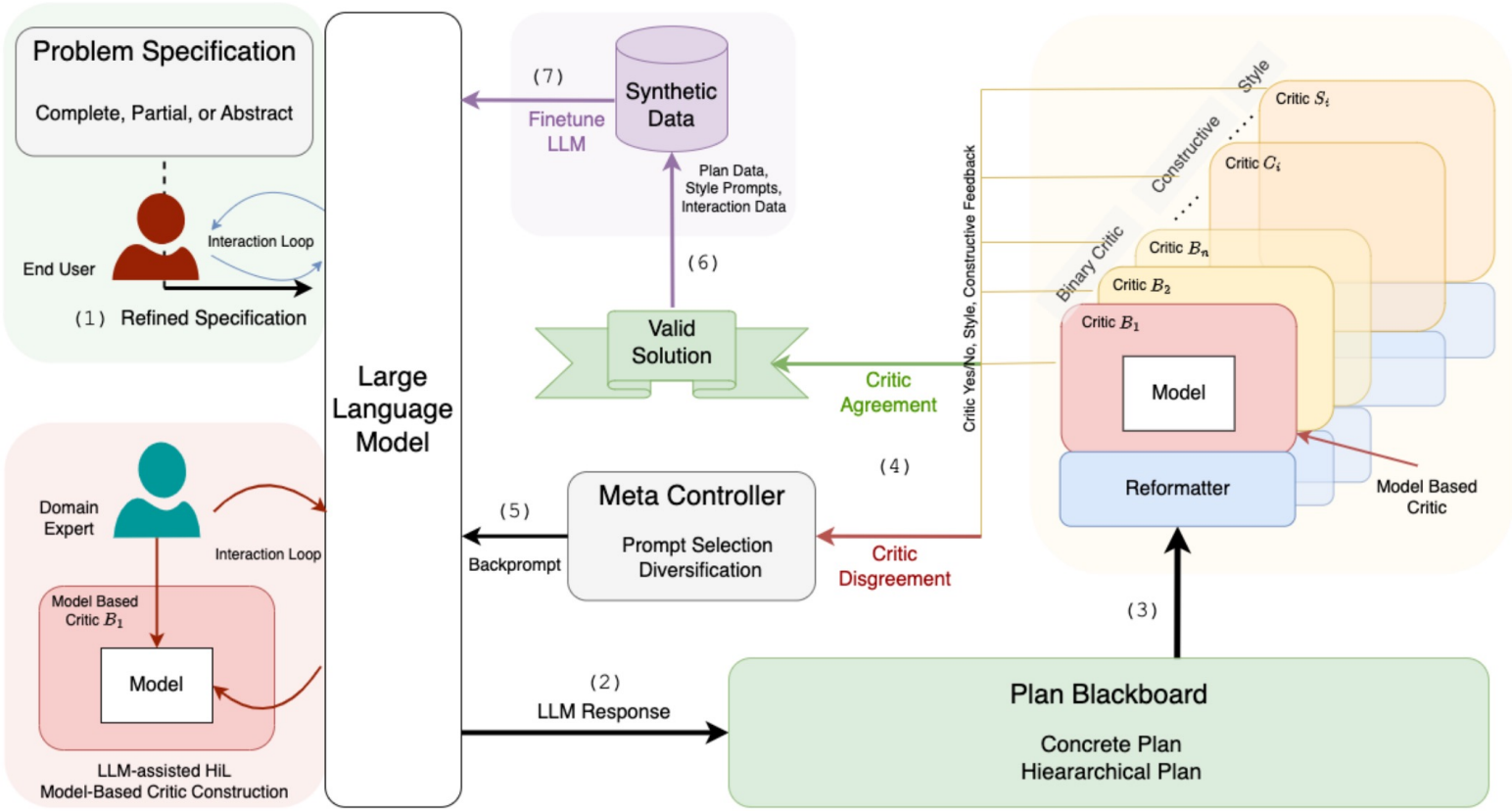
- Most practical problems cannot be solved easily via rule-based planning
- LLMs can still help to truncate the search space
- LLMs with verifiers in an iterative cycle can be beneficial for searching



Simple LLM Planning Architecture

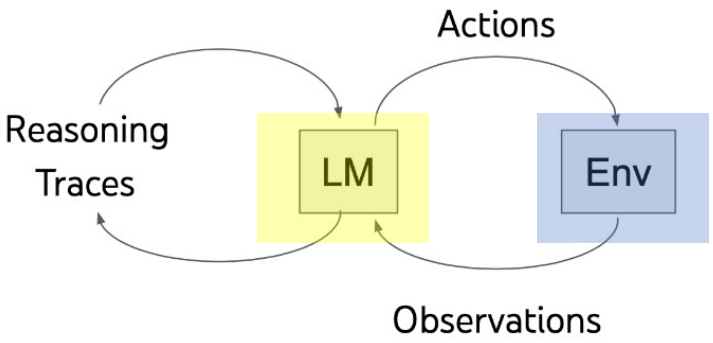


LLM-Modulo Architecture

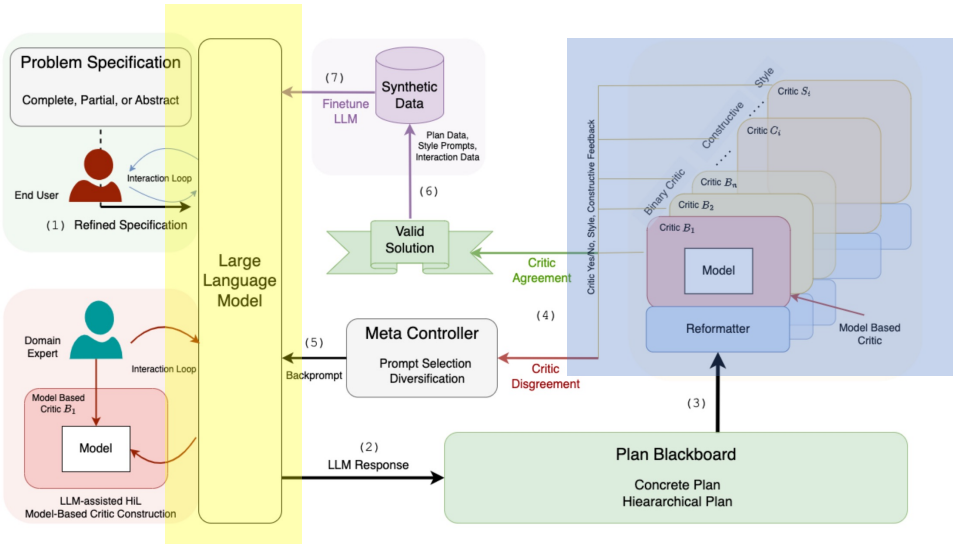


LLM Modulo. 2024. Subbarao et. al.

Comparison to ReAct:
Critic functions like the Environment Feedback



ReAct: Synergizing Reasoning and Acting in Language Models. 2023. Yao et. al.



LLM Modulo. 2024. Subbarao et. al.

Multiple critics

- Hard constraints refer to correctness verification which can include causal correctness, timeline correctness, resource constraint correctness as well as unit tests

- Soft constraints can include more abstract notions of good form such as style, explicability, preference conformance
  - Typically done by LLMs

Usefulness of a critic

- Best to have constructive feedback
  - **“No, try again”**: Not very useful
  - **“No, try again, here is one thing wrong with the current plan”**: Typically good enough
  - **“No, try again, here are all the things wrong with the current plan.”**: May be too much information

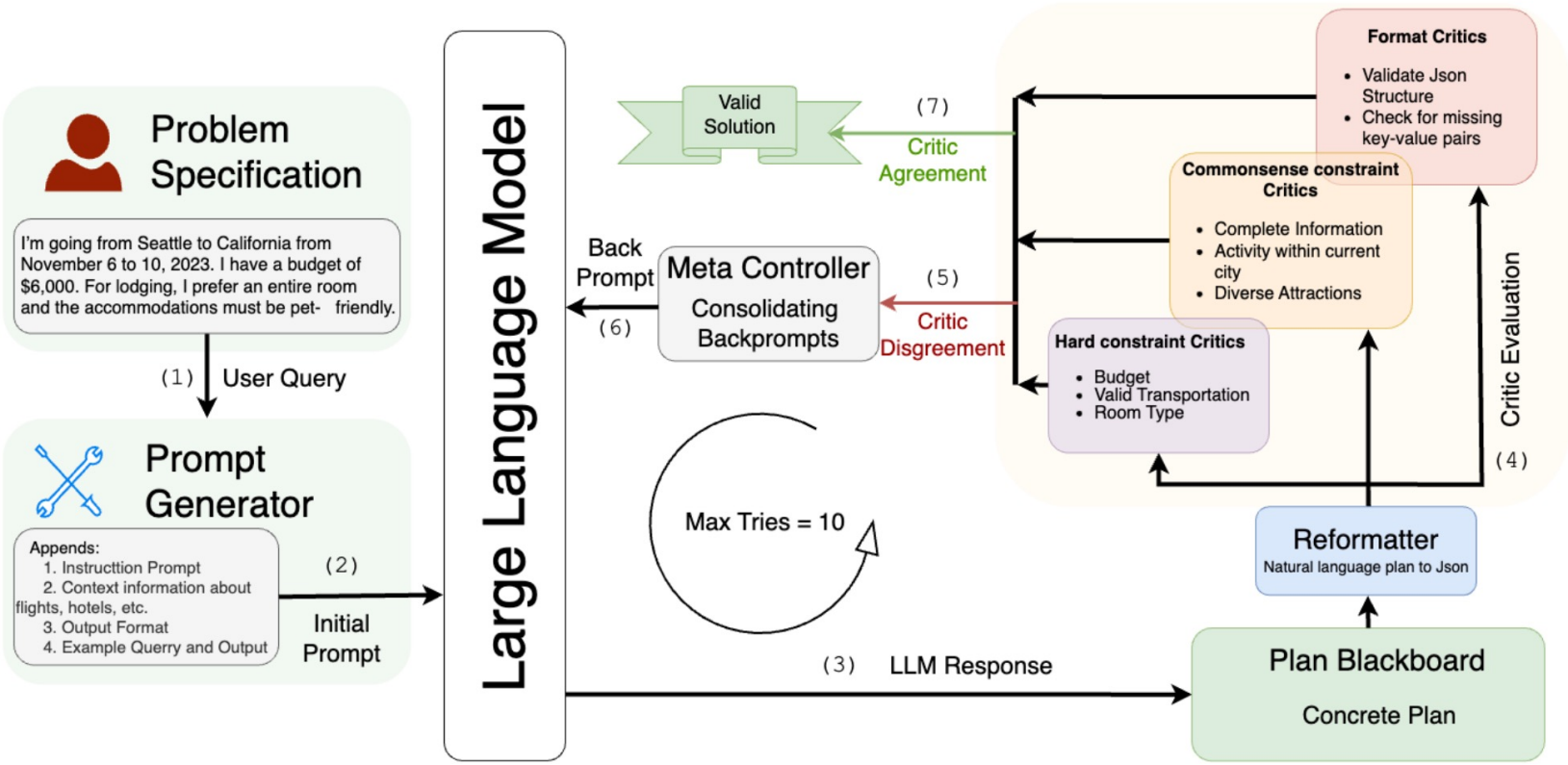
- Example: Code compiler serves as a hard useful critic!
- Planning problems usually have difficulties with creating a good critic, especially if critic does not have the simulation of the world

Backprompting

- The critiques from the various critics are **pooled together** by the Meta (Backprompt) Controller, which passes a processed version of them to the LLM as the next iterative prompt to elicit the next guess

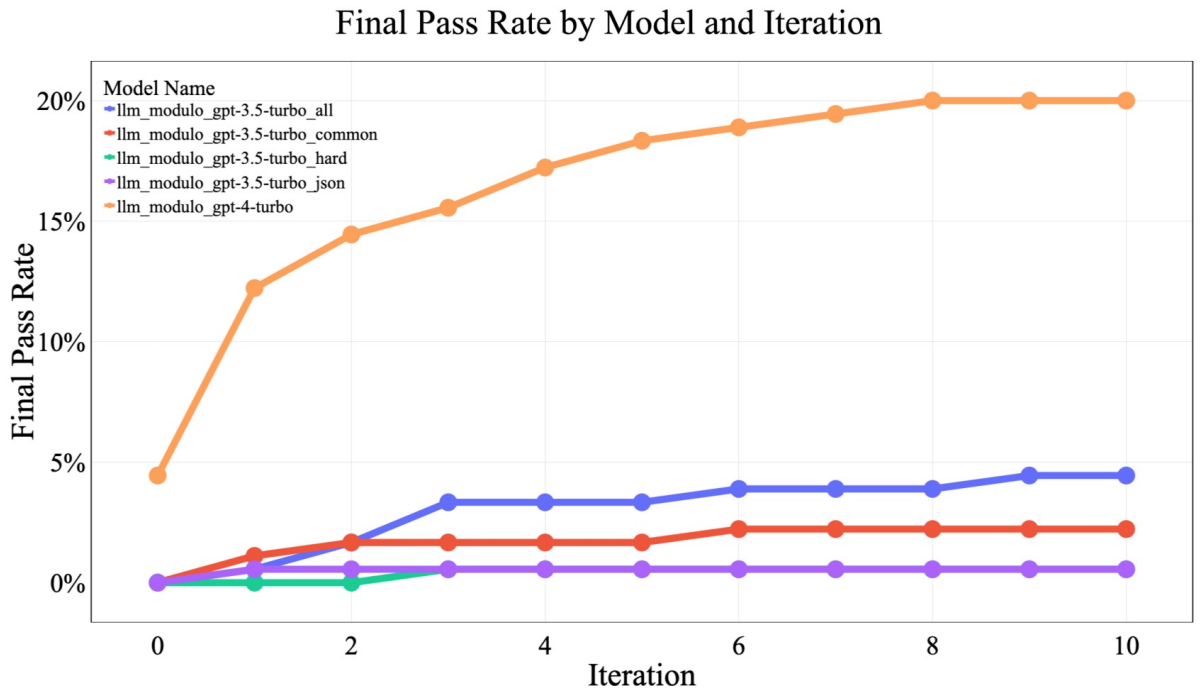
- The processing in the controller be:
  - Simple round-robin selection of prompts
  - Summarised prompt
  - Prompt diversification strategy to elicit the next candidate from a different part of the implicit search space

LLM Modulo Architecture for Travel Planning



LLM Modulo. 2024. Subbarao et. al.

LLM Modulo helps (limited) for improvement in Travel Planning



LLM Modulo. 2024. Subbarao et. al.

My thoughts

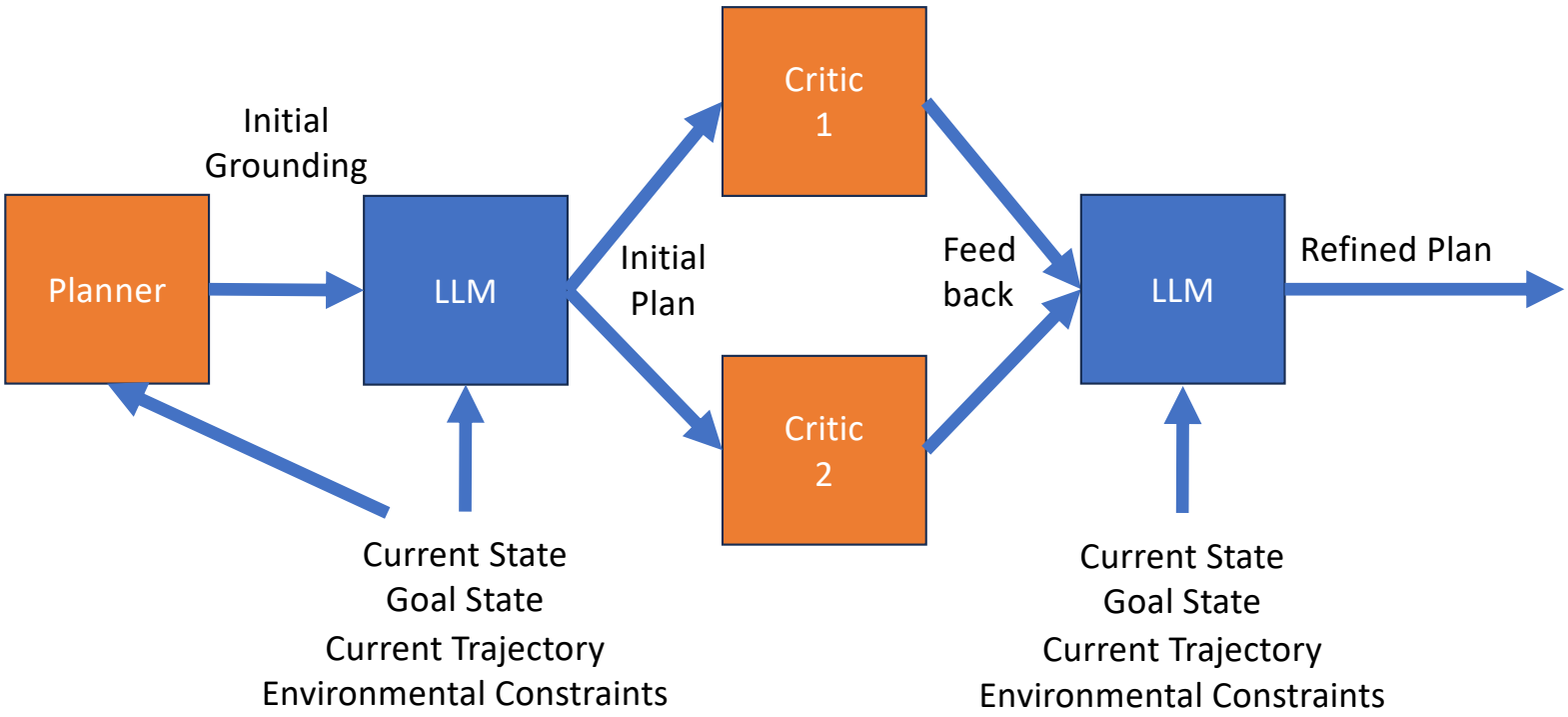
LLM Modulo got some parts right

Maybe critic at the plan level is not the right approach

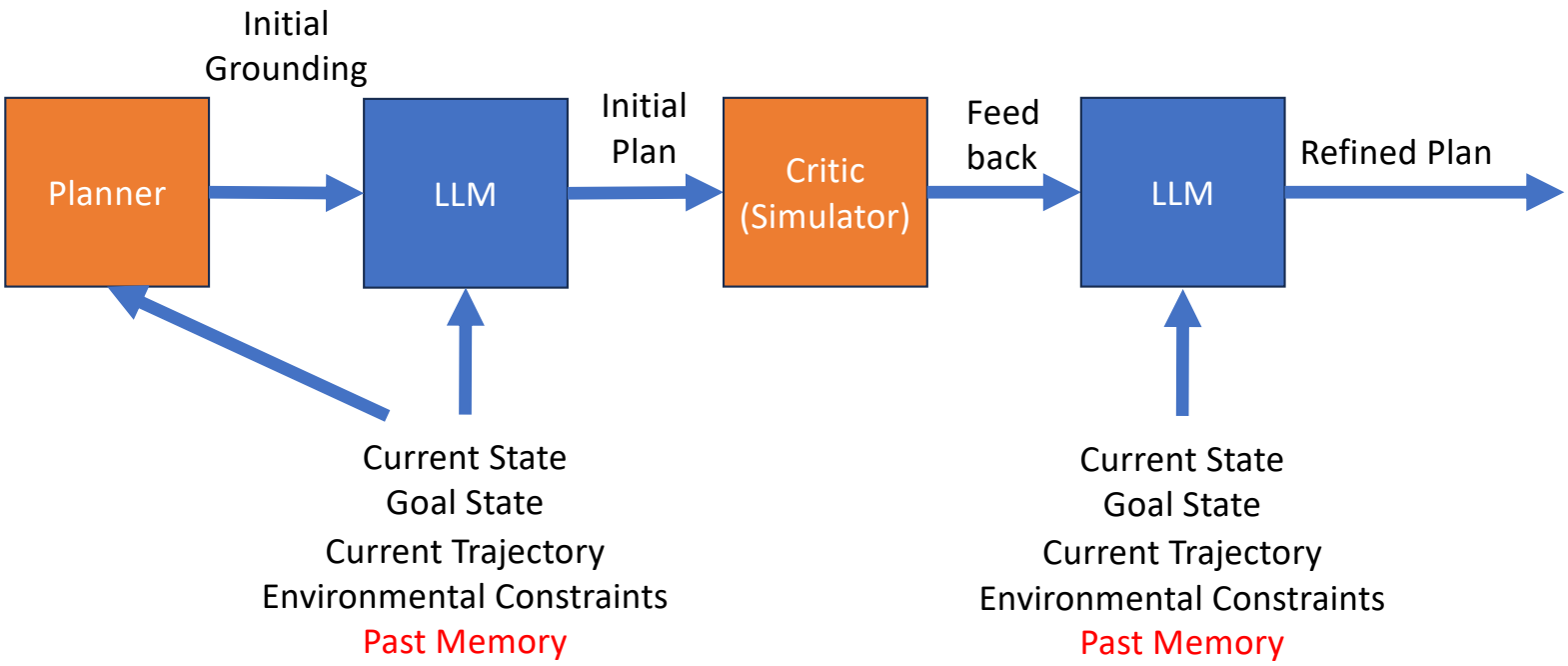
Is Planning the wrong domain for LLMs?

- Subbarao keeps saying **CoT, reflection offers no benefit for LLMs**, but that is for long-term planning and not short-term action selection
- What if planning is something that is best done via simulation and search?
- LLM can be used as a heuristic for the search, but it may over-bias the selection
- Perhaps LLMs are better suited for lower-level action execution, or simple planning with few steps with semantically meaningful actions

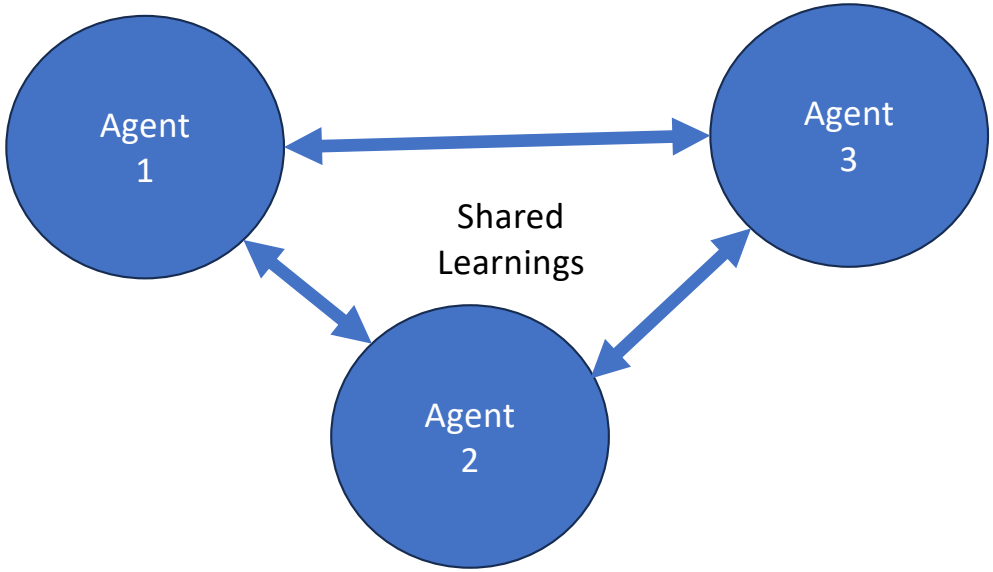
Food for thought:
Planner for initial grounding?

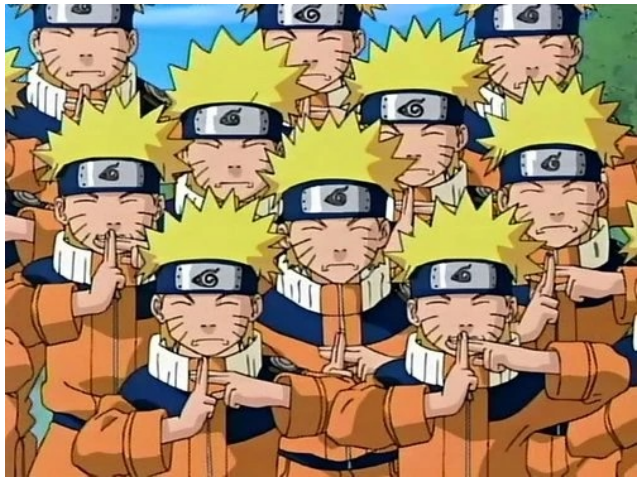


Extensions to LLM Planning Architecture: Adding Simulators, Memory

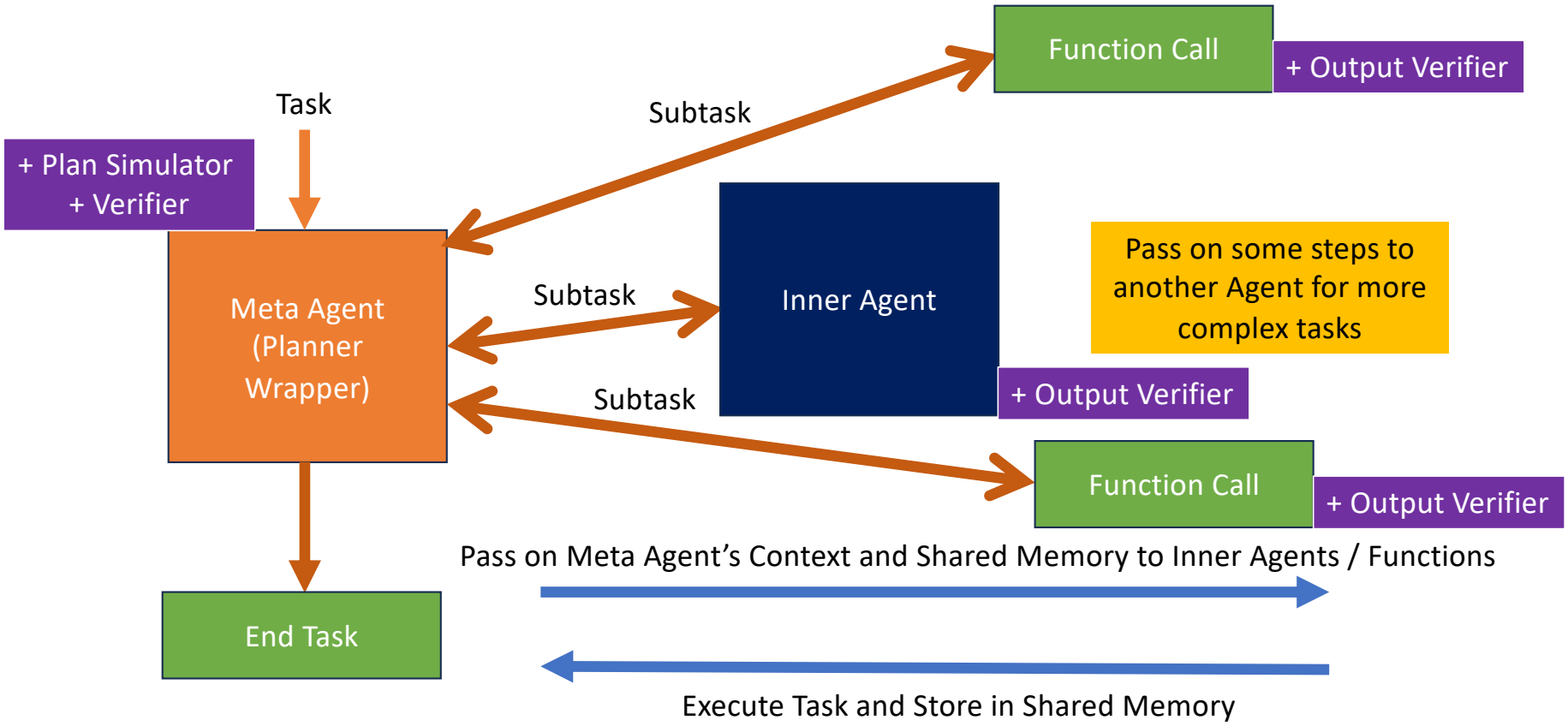


Extensions: Population-based Learning





Relevance to TaskGen: Add Simulators/Verifiers in Subtasks / Planner



https://github.com/simbianai/taskgen

Questions to Ponder

- How can we create good critics for evaluating the plan?
- Where can simulation be used for LLM modulo?
- Where can we reference memory for LLM modulo?
- How far ahead should we plan?

Travel Planner Prompt

Prompt 1: Extracting Hard Constraint : Budget

```
    Assume you have a json with following as a list. The list is the itinerary generated for each day of the

    trip.

```
- llm_response: An array containing objects with the following fields for each day of the trip:
  - day: Integer representing the day number.
  - people_number: Integer representing the number of people.
  - current_city: String representing the current city/location.
  - transportation: String representing transportation details (e.g., flight number, departure/arrival time).
  - breakfast: String representing breakfast arrangements.
  - attraction: String representing the attraction(s) for the day.
  - lunch: String representing lunch arrangements.
  - dinner: String representing dinner arrangements.
  - accommodation: String representing accommodation details.
```

Additionally, you have following functions that you can use :

1. get_cost_of_transport(source, destination, mode-of-travel)['cost']
2. A valid restaurants dataframe with keys : 'Name','Average Cost','Cuisines','Aggregate Rating','City'
3. A valid attractions dataframe with keys : 'Name','Latitude','Longitude','Address','Phone','Website',"City"
4. A valid accomodations dataframe with keys : 'NAME','price','room type', '

house_rules', 'minimum nights', 'maximum occupancy', 'review rate number', 'city'
5. A valid flights dataframe with kyes : 'Flight Number', 'Price', 'DepTime', 'ArrTime',

'ActualElapsedTime','FlightDate','OriginCityName','DestCityName','Distance'

You task is to write

A method called 'get_total_cost' which parses the json file and calculates the total cost of the trip.

Ensure you take into account the cost of transport, breakfast, lunch, dinner, accomodation.

Ensure that you take into account the number of people for which the itinerary has been made.

Use the data within the json and the available dataframes and tools for estimating the total cost.

Do not assume any other details.

Implement the complete function with all the details. Return a float value representing the total cost.

Robust Planning with LLM-Modulo Framework: Case Study in Travel Planning. 2024. Atharva et. al.
```

Critic Feedback

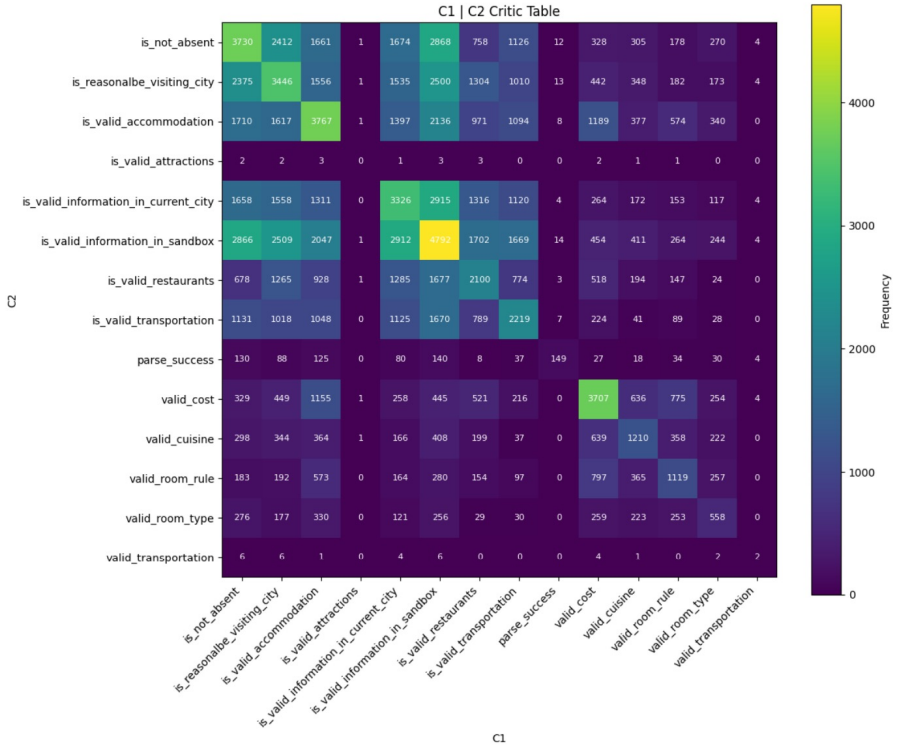


Figure 4. Enter Caption

We analyze the correlation between, given that a certain critic was fired (C1) how many times another critic was fired (C2). This is an exploratory analsis to find the correlation between how critics were fired and obtain information about failures during the LLM Modulo iteration steps. We leave a thorough analysis as future work. In figure 4 we see that several critics are correlated and failure of one almost always triggers a certain other critic.

Robust Planning with LLM-Modulo Framework: Case Study in Travel Planning. 2024. Atharva et. al.