



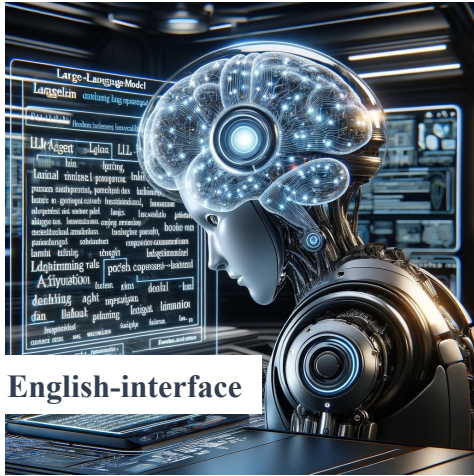
# Self-Improve LLM-based Agent and its Prospects

Yule Wang, PhD

Misson Cloud, MLE (Consulting)

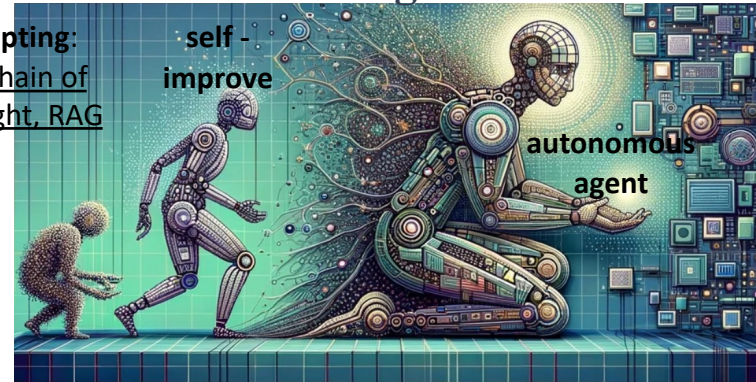
# Outline

## I. What is an LLM-Agent?

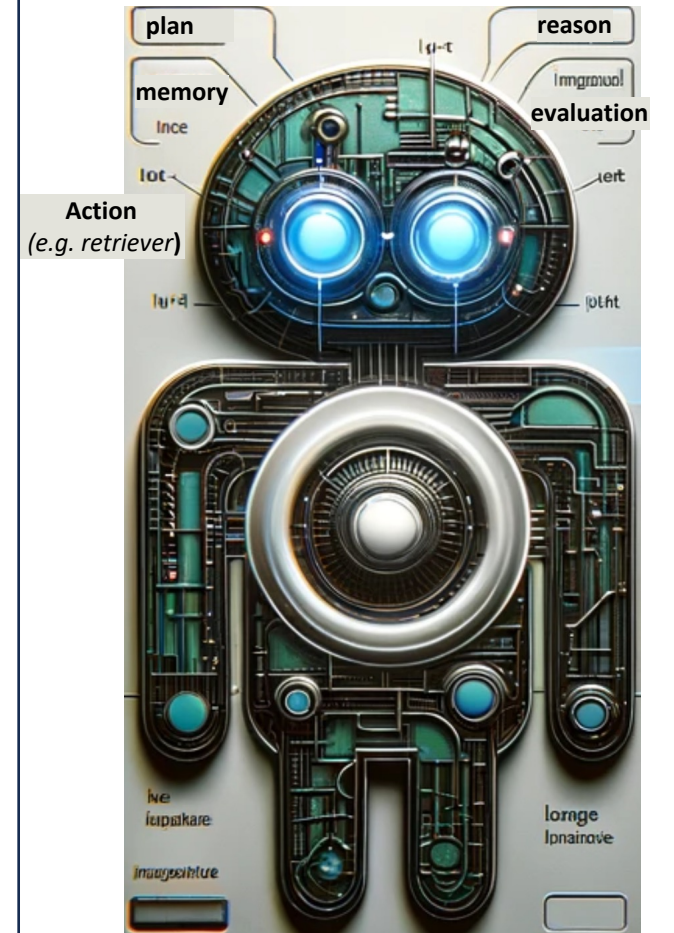


## II. Evolutionary Approach to build an autonomous LLM-agent for Diverse Problem-Solving

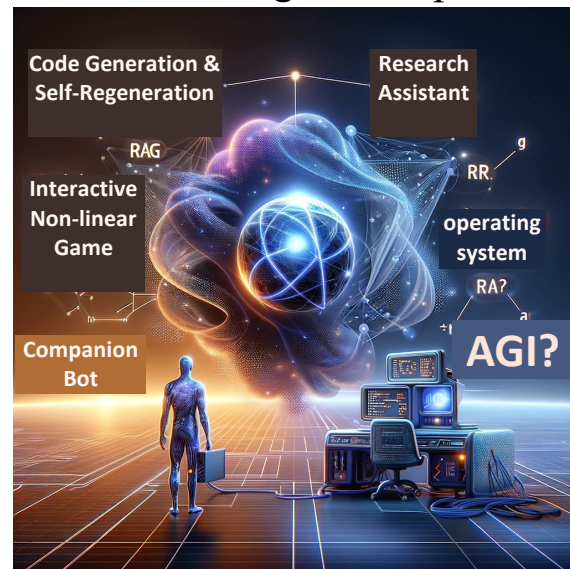
**prompting:**  
e.g. chain of thought, RAG



## III. Create your own independent autonomous LLM-agent



## IV. LLM-agent Prospects



# I. What is an LLM-Agent?



# Strategy on Downstream NLP Tasks

## Fine-tuning vs Prompting & Agent

Wikipedia, books, webpage, etc...

self-supervised



Pretrained LLM

- **linguistic/semantic knowledge**
- **(common-sense) reasoning ability**
- **knowledge & skills in some domains**

# Strategy on Downstream NLP Tasks

## Fine-tuning vs Prompting & Agent

Wikipedia, books, webpage, etc...

self-supervised

Pretrained LLM

- **linguistic/semantic knowledge**
- **(common-sense) reasoning ability**
- **knowledge & skills in some domains**

gap

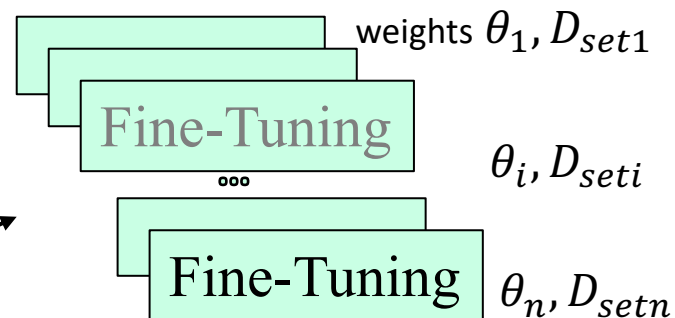
Domain-Specific Task/Question

Missing:

- up-to-date or domain-specific knowledge
- specific expertise to specific-domain

# Strategy on Downstream NLP Tasks

## Fine-tuning vs Prompting & Agent



### Finetuning Issues:

- Needs to finetune on every domain
- May lose or conflicts on pre-trained info
- Hard to enhance reasoning ability
- Bad at some specific tasks without external tools, e.g. arithmetic calculator/specific APIs

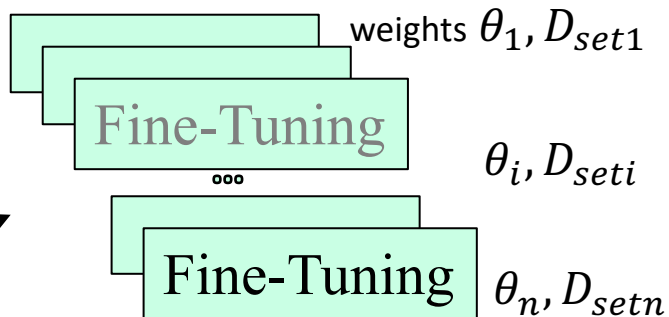
Pretrained LLM



Domain-Specific Task/Question

# Strategy on Downstream NLP Tasks

## Fine-tuning vs Prompting & Agent



### Finetuning Issues:

- Needs to finetune on every domain
- May lose or conflicts on pre-trained info
- Hard to enhance reasoning ability
- Bad at some specific tasks without external tools, e.g. arithmetic calculator/specific APIs

Pretrained LLM



Domain-Specific Task/Question

Prompting → LLM Agent (more complex & auto)

# LLM Agent

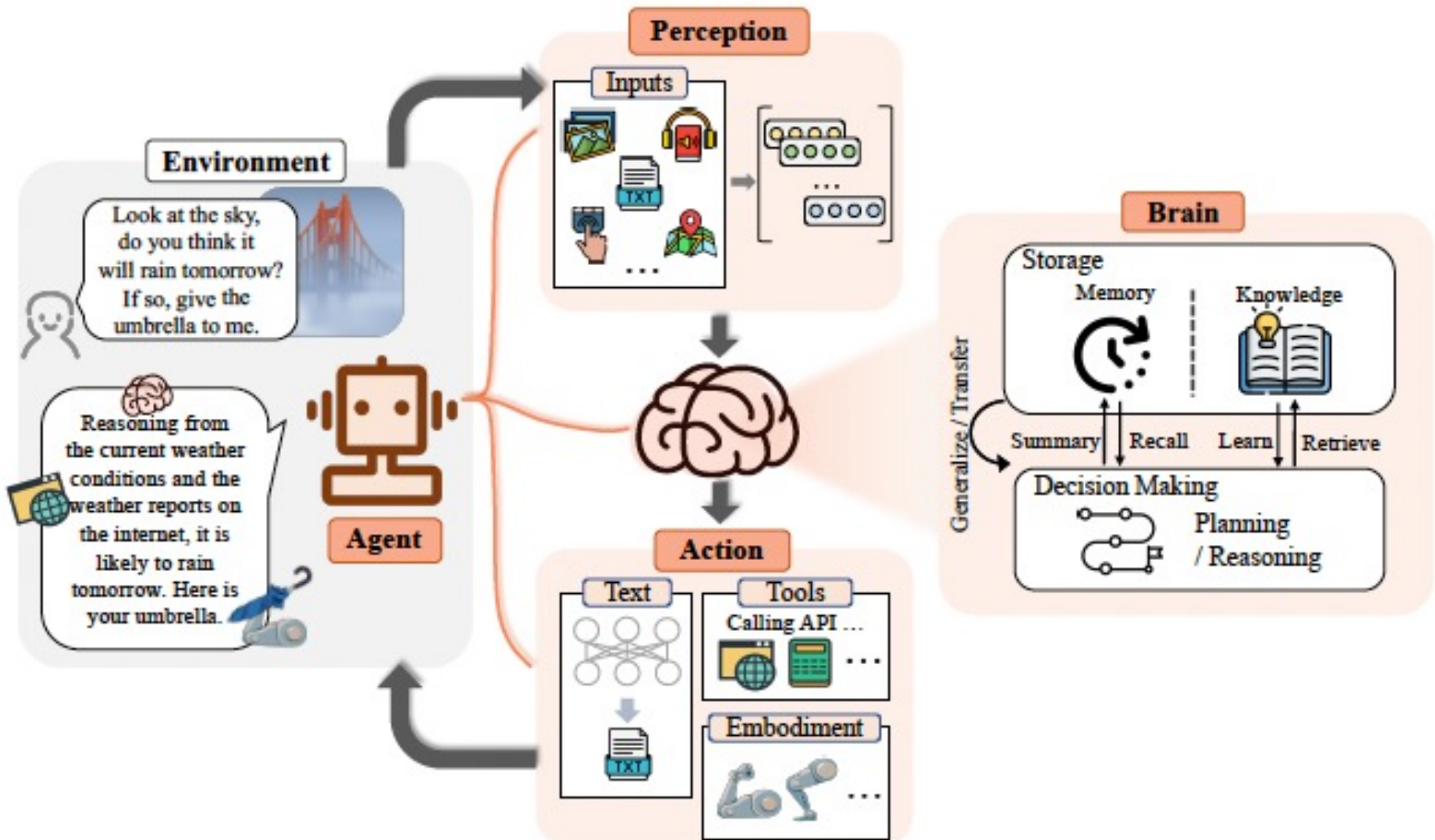
*“If they find a parrot who could answer to everything, I would claim it to be an intelligent being without hesitation.”*

*—Denis Diderot, 1875*



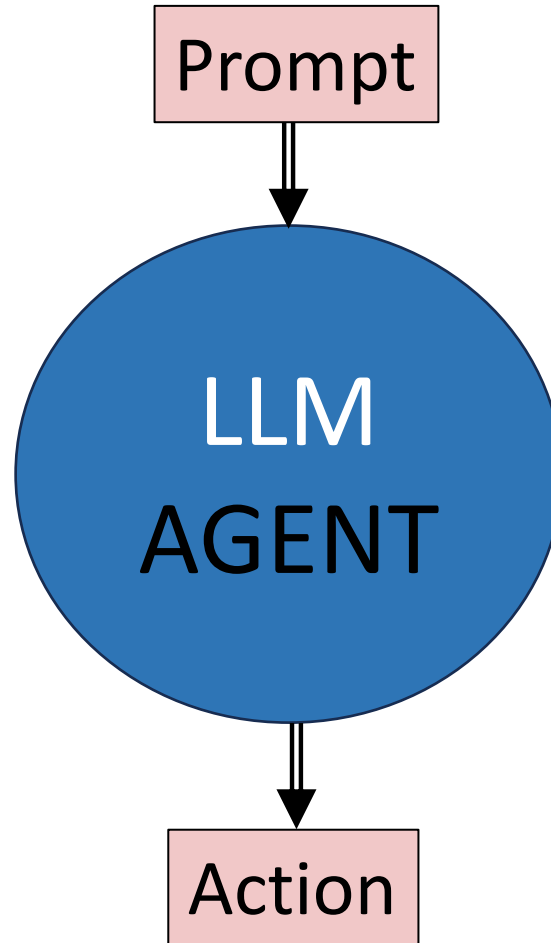


# LLM Agent



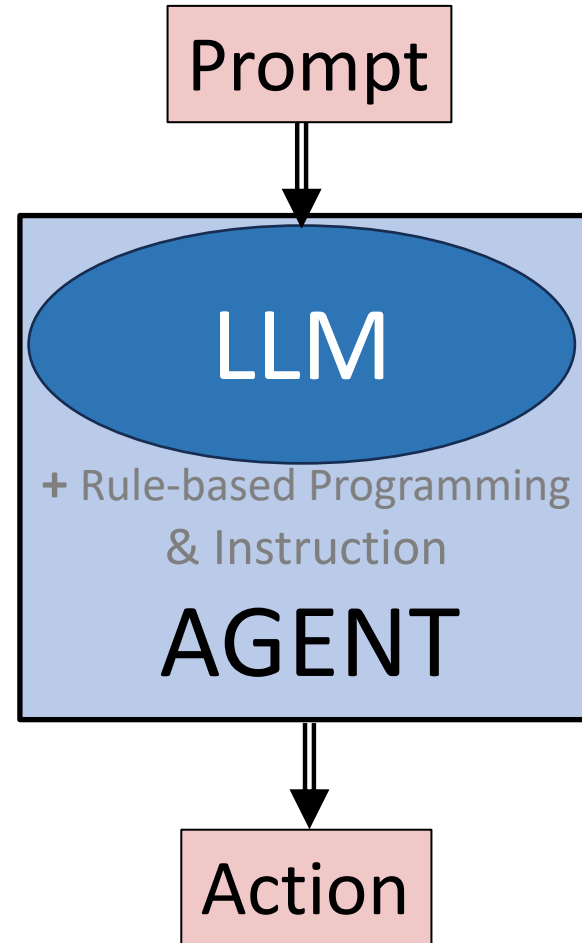
[1] Xi et. al Fu

# LLM Agent? AGI!



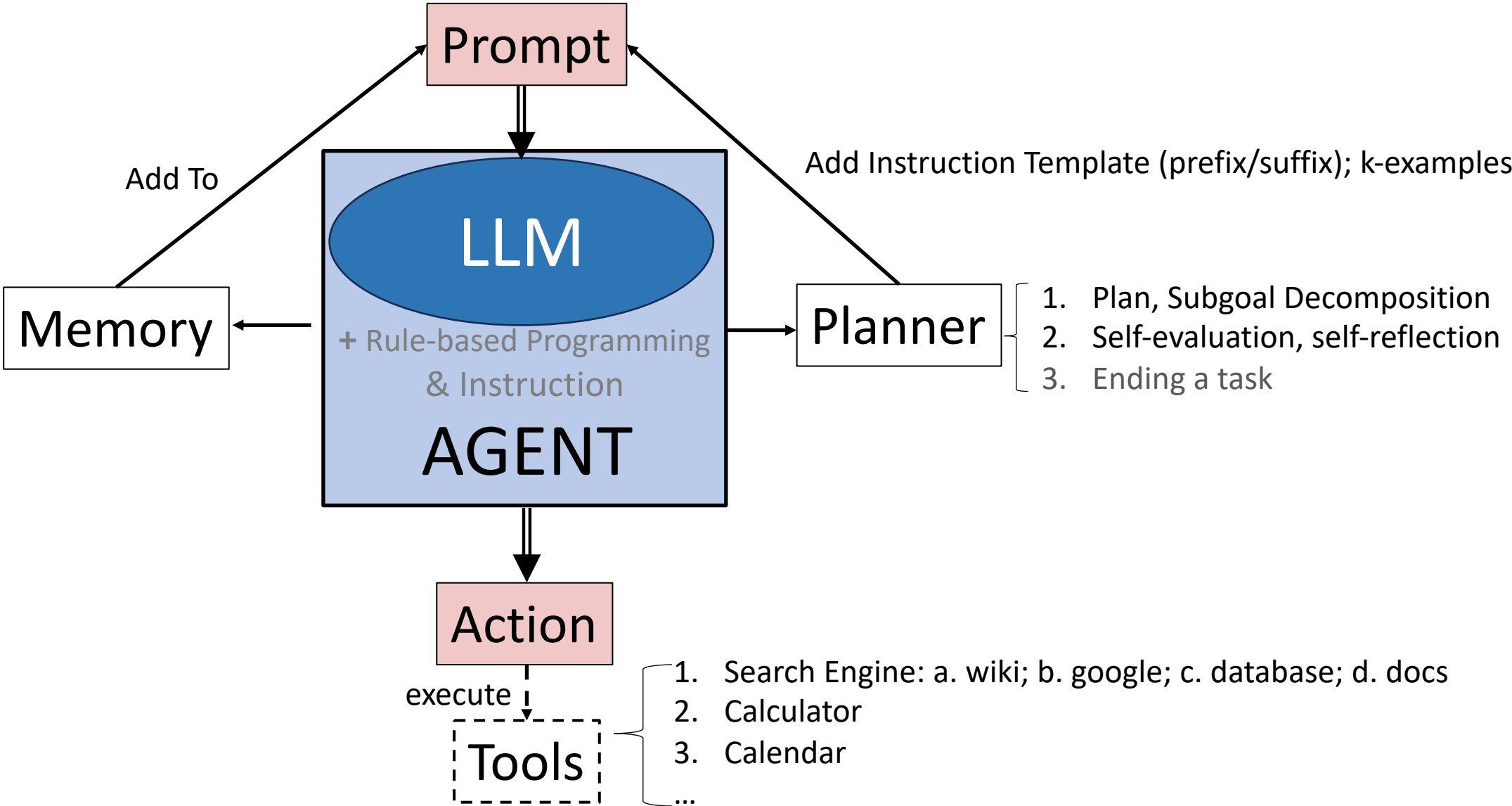
“A mere LLM as an agent in the future”

# LLM Agent

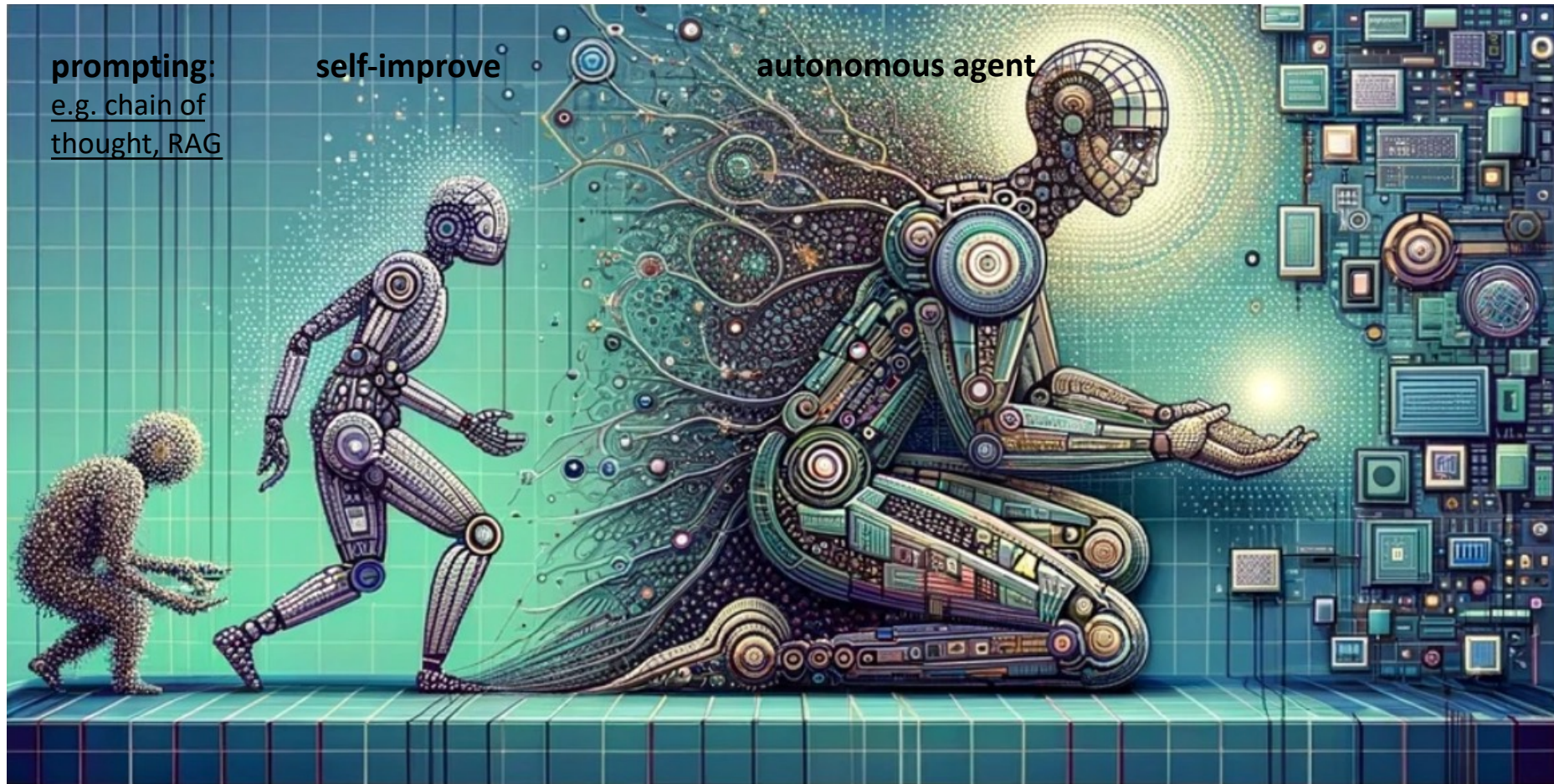


~~“A mere LLM as an agent in the future”~~

# LLM Agent & Core Components



## II. Evolutionary Approach to build an autonomous LLM-agent for Diverse Problem-Solving



# LLM Agents Evolution

Input/  
Output

Plain Question and Task. No instruction:

- “What is an LLM agent?”
- “What is today’s NBA match score?”
- “Now solve this 24 point game.”

# LLM Agents Evolution



Instructions and/or examples e.g.

- “I am just a child. I wish your answer to be easy to understand for me. If you do not know the knowledge, please say No.”
- “Please rate the toxicity of these texts on a scale from 0 to 10.  
Parse the score to JSON format like this {‘text’: the text to grade; ‘toxic\_score’: the toxicity score of the text}”
- “Follow this (/these) example(s), ..., .., please answer my question.” (**1-shot/k-shot prompting**)
- “Let’s think step by step” (**Chain of Thought**)

# LLM Agents Evolution



**(a)**

Q: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?  
A: *Let's think step by step.*

LLMs →

First, 20% of the class enrolled in contemporary dance, so that means that 80% of the class is left.  
Next, 25% of the remaining 80% enrolled in jazz dance, so that means that 25% of the original class enrolled in jazz dance. That leaves 55% of the original class.  
Finally, the rest of the class (55%) enrolled in hip-hop dance. So, 55% of the entire class enrolled in hip-hop dance. ❌

**(b)**

Q: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?  
A: *Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step.*

LLMs →

**Plan:**  
Step 1: Calculate the total number of students who enrolled in contemporary and jazz dance.  
Step 2: Calculate the total number of students who enrolled in hip-hop dance.  
Step 3: Calculate the percentage of students who enrolled in hip-hop dance.  
**Solution:**  
Step 1: 20% of 20 students is 4 students. 25% of the remaining 16 students is 4 students. So, a total of 8 students enrolled in contemporary and jazz dance.  
Step 2: The remaining 12 students enrolled in hip-hop dance.  
Step 3: The percentage of students who enrolled in hip-hop dance is  $12/20 = 60\%$ . ✅

**(c)**

Q: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?  
A: *Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step.*

**Plan:**  
Step 1: Calculate the total number of students who enrolled in contemporary and jazz dance.  
Step 2: Calculate the total number of students who enrolled in hip-hop dance.  
Step 3: Calculate the percentage of students who enrolled in hip-hop dance.  
**Solution:**  
Step 1: 20% of 20 students is 4 students. 25% of the remaining 16 students is 4 students. So, a total of 8 students enrolled in contemporary and jazz dance.  
Step 2: The remaining 12 students enrolled in hip-hop dance.  
Step 3: The percentage of students who enrolled in hip-hop dance is  $12/20 = 60\%$ .  
*Therefore, the answer (arabic numerals) is*

LLMs →

60%

## Direct Prompting

GPT-3

Question: Who lived longer, Theodor Haecker or Harry Vaughan Watkins?  
Answer: Harry Vaughan Watkins.

Question: Who was president of the U.S. when superconductivity was discovered?  
Answer: Franklin D. Roosevelt ❌

## Chain of Thought

GPT-3

Question: Who lived longer, Theodor Haecker or Harry Vaughan Watkins?  
Answer: Theodor Haecker was 65 years old when he died. Harry Vaughan Watkins was 69 years old when he died. So the final answer (the name of the person) is: Harry Vaughan Watkins.

Question: Who was president of the U.S. when superconductivity was discovered?  
Answer: Superconductivity was discovered in 1911 by Heike Kamerlingh Onnes. Woodrow Wilson was president of the United States from 1913 to 1921. So the final answer (the name of the president) is: Woodrow Wilson. ❌

## Self-Ask

GPT-3

Question: Who lived longer, Theodor Haecker or Harry Vaughan Watkins?  
Are follow up questions needed here: Yes.  
Follow up: How old was Theodor Haecker when he died?  
Intermediate answer: Theodor Haecker was 65 years old when he died.  
Follow up: How old was Harry Vaughan Watkins when he died?  
Intermediate answer: Harry Vaughan Watkins was 69 years old when he died.  
So the final answer is: Harry Vaughan Watkins

Question: Who was president of the U.S. when superconductivity was discovered?  
Are follow up questions needed here: Yes.  
Follow up: When was superconductivity discovered?  
Intermediate answer: Superconductivity was discovered in 1911.  
Follow up: Who was president of the U.S. in 1911?  
Intermediate answer: William Howard Taft.  
So the final answer is: William Howard Taft. ✅



# LLM Agents Evolution



**(1) Hotspot QA**

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

**(1a) Standard**  
Answer: iPod ❌

**(1b) CoT (Reason Only)**  
Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.  
Answer: iPhone, iPad, iPod Touch ❌

**(1c) Act-Only**  
Act 1: Search[Apple Remote]  
Obs 1: The Apple Remote is a remote control ...  
Act 2: Search[Front Row]  
Obs 2: Could not find [Front Row]. Similar: ...  
Act 3: Search[Front Row (software)]  
Obs 3: Front Row is a discontinued media center software ...  
Act 4: Finish[yes] ❌

**(1d) ReAct (Reason + Act)**  
Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.  
Act 1: Search[Apple Remote]  
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...  
Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.  
Act 2: Search[Front Row]  
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]  
Thought 3: Front Row is not found. I need to search Front Row (software) .  
Act 3: Search[Front Row (software)]  
Obs 3: Front Row is a discontinued media center software ...  
Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.  
Act 4: Finish[keyboard function keys] ✅

---

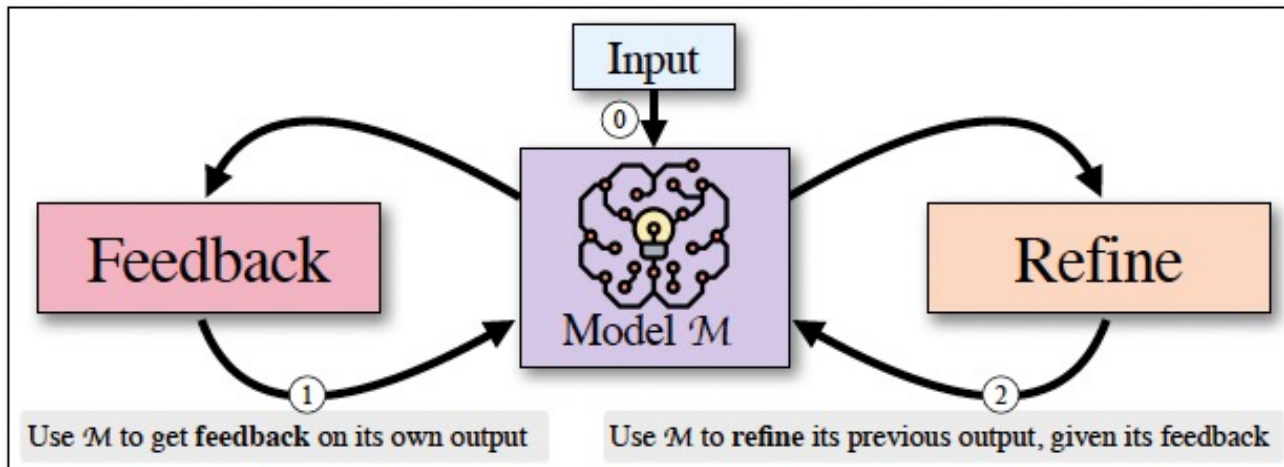
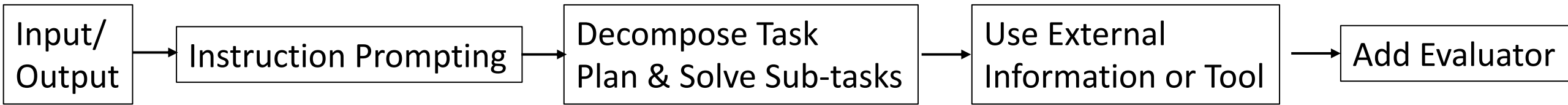
**(2) AlfWorld**

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1. Your task is to: Put some pepper shaker on a drawer.

**(2a) Act-Only**  
Act 1: Go to drawer 1  
Obs 1: The drawer 1 is closed.  
Act 2: Open drawer 1  
Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dish sponge 2, and a spoon 1.  
Act 3: Go to sinkbasin 1  
Obs 3: On the sinkbasin 1, you see a dish sponge 3, a spatula 1, and a spoon 2.  
Act 4: Take peppershaker 1 from sinkbasin 1  
Obs 4: Nothing happens.  
Act 5: Take peppershaker 1 from sinkbasin 1  
Obs 5: Nothing happens. ❌

**(2b) ReAct (Reason + Act)**  
Act 1: Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ... ]  
Act 2: Go to cabinet 1  
Obs 2: On the cabinet 1, you see a vase 2.  
(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)  
Act 6: Go to countertop 3  
Obs 6: On the countertop 3, you see a apple 1, a bread 1, ..... a pepper shaker 1, and a vase 3.  
Act 7: Take pepper shaker 1 from countertop 3  
Obs 7: You pick up the pepper shaker 1 from the countertop 3.  
Act 8: Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1. ]  
Act 9: Go to drawer 1  
Obs 9: Drawer 1 is closed.  
Act 10: Open drawer 1  
Obs 10: You open Drawer 1 ...  
Act 11: Put pepper shaker 1 in/on drawer 1  
Obs 11: You put pepper shaker 1 in/on the drawer 1. ✅

# LLM Agents Evolution



(a) Dialogue:  $x, y_t$

User: I am interested in playing Table tennis.  
 Response: I'm sure it's a great way to socialize, stay active

(b) FEEDBACK fb

Engaging: Provides no information about table tennis or how to play it.  
 User understanding: Lacks understanding of user's needs and state of mind.

(c) REFINE  $y_{t+1}$

Response (refined): That's great to hear (...) ! It's a fun sport requiring quick reflexes and good hand-eye coordination. Have you played before, or are you looking to learn?

(d) Code optimization:  $x, y_t$

```

Generate sum of 1, ..., N
def sum(n):
    res = 0
    for i in range(n+1):
        res += i
    return res
  
```

(e) FEEDBACK fb

This code is slow as it uses brute force. A better approach is to use the formula ...  $(n(n+1))/2$ .

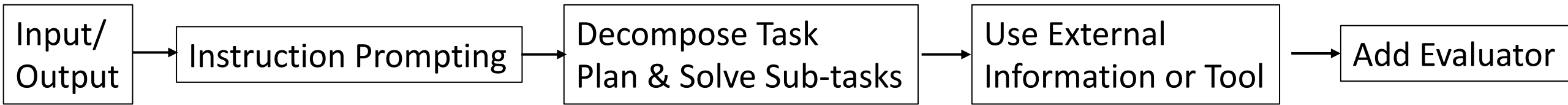
(f) REFINE  $y_{t+1}$

Code (refined)

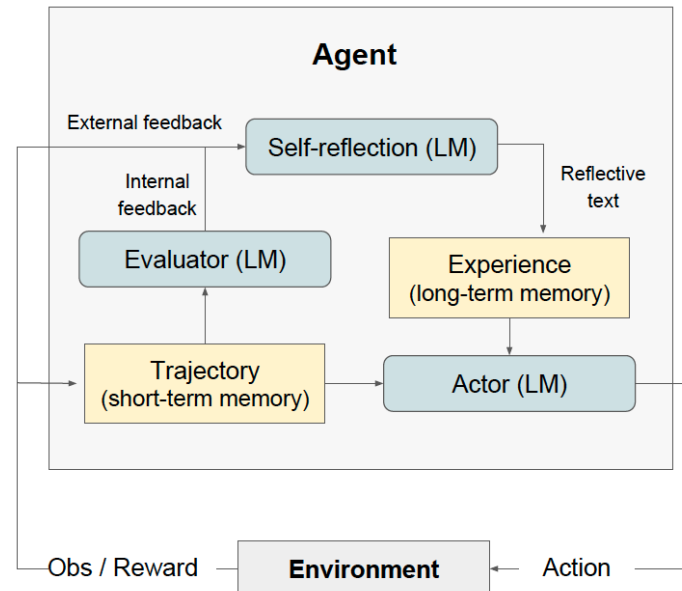
```

def sum_faster(n):
    return (n*(n+1))//2
  
```

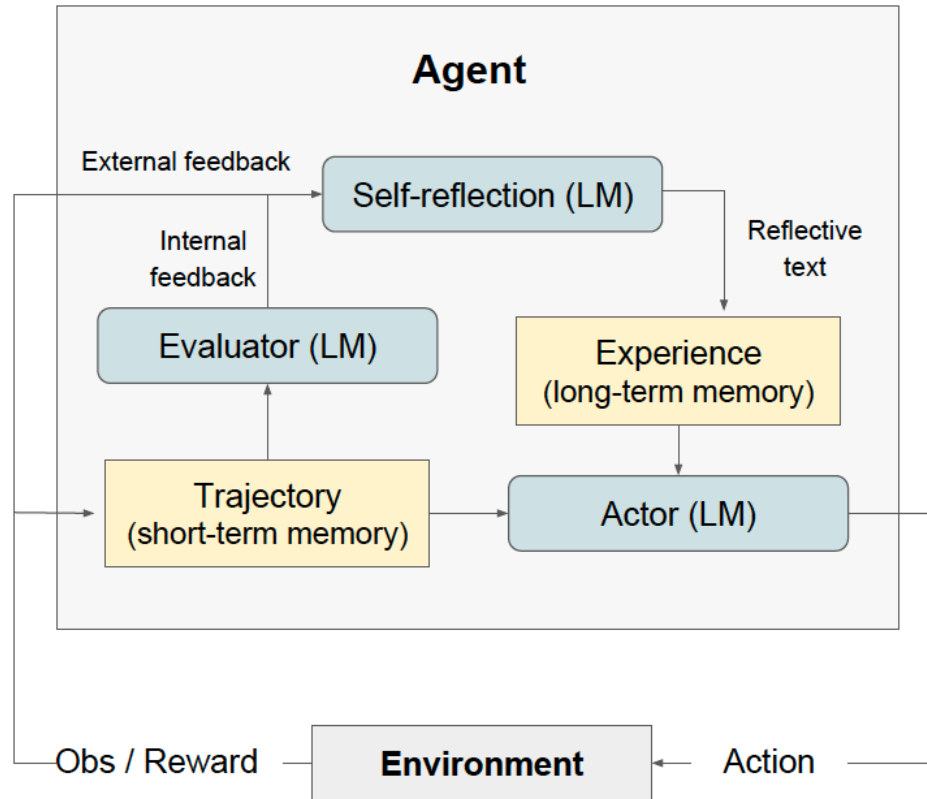
# LLM Agents Evolution



Long term evaluations on each sub-step & whole task



# Autonomous Agent



---

## Algorithm 1 Reinforcement via self-reflection

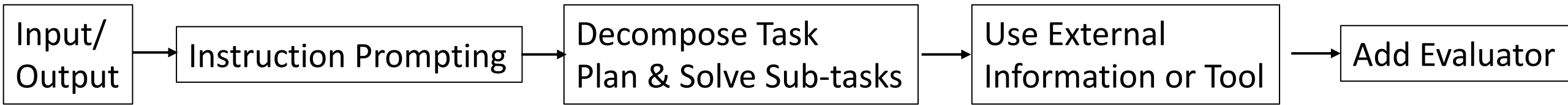
---

Initialize Actor, Evaluator, Self-Reflection:  
 $M_a, M_e, M_{sr}$   
Initialize policy  $\pi_\theta(a_i|s_i)$ ,  $\theta = \{M_a, mem\}$   
Generate initial trajectory using  $\pi_\theta$   
Evaluate  $\tau_0$  using  $M_e$   
Generate initial self-reflection  $sr_0$  using  $M_{sr}$   
Set  $mem \leftarrow [sr_0]$   
Set  $t = 0$   
**while**  $M_e$  not pass or  $t < \text{max trials}$  **do**  
    Generate  $\tau_t = [a_0, o_0, \dots, a_i, o_i]$  using  $\pi_\theta$   
    Evaluate  $\tau_t$  using  $M_e$   
    Generate self-reflection  $sr_t$  using  $M_{sr}$   
    Append  $sr_t$  to  $mem$   
    Increment  $t$   
**end while**  
**return**

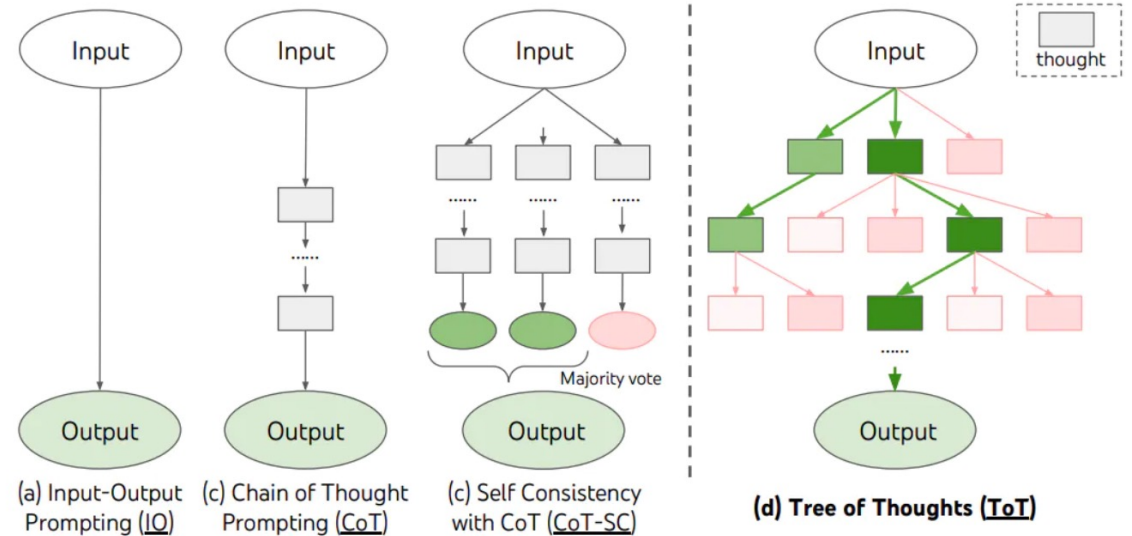
---

Reflexion: an RL model whose memory as a learnable component instead of network parameters

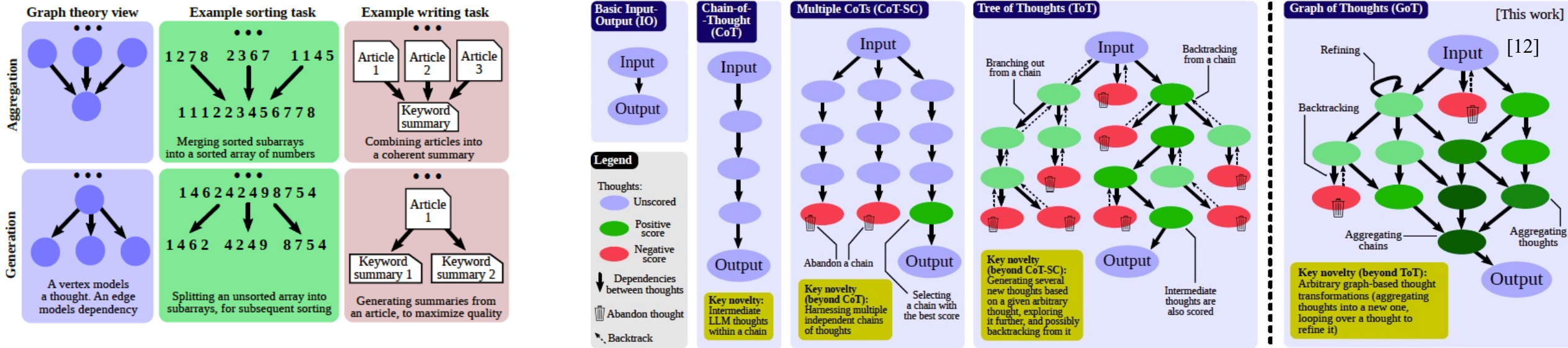
# LLM Agents Evolution



Multiple traces of thoughts  
Which one?



# LLM Agents: Add evaluation



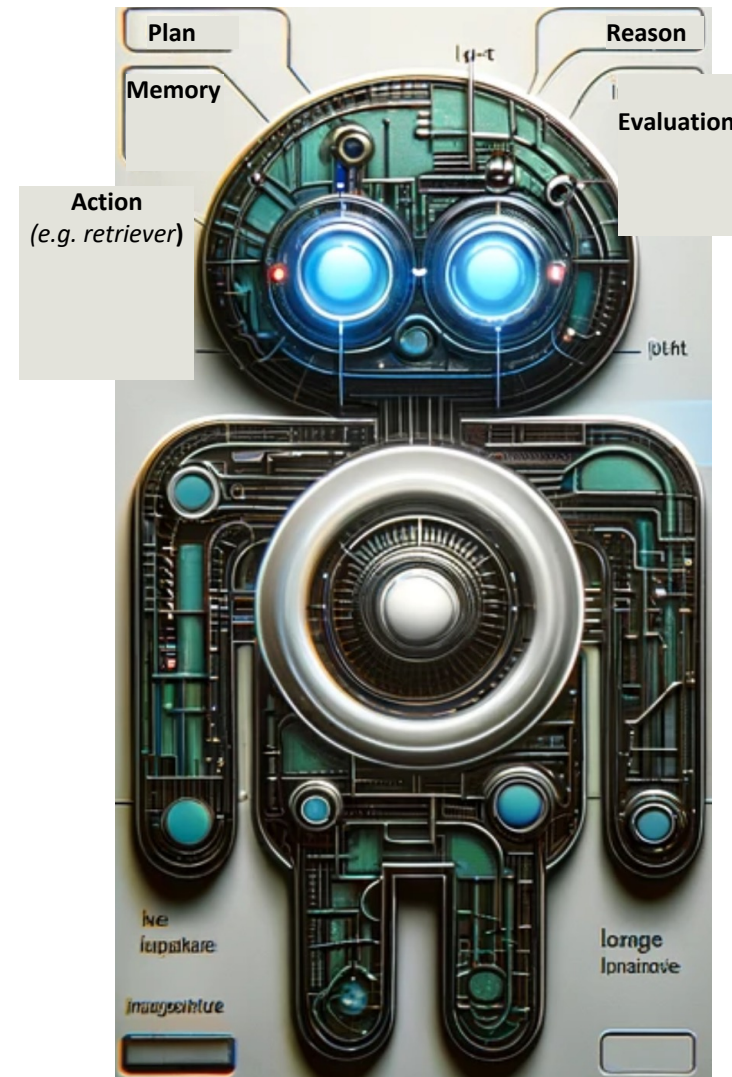
- Multiple trials
  - Self Consistency (202203)<sup>[10]</sup> ( $T > 0$ , votes)
  - Tree of Thoughts (202305)<sup>[11]</sup> (ToT)
  - Graph of Thoughts<sup>[12]</sup> (GoT)

Scheme	Latency	Volume
Chain-of-Thought (CoT)	$N$	$N$
Self-Consistency with CoT (CoT-SC)	$N/k$	$N/k$
Tree of Thoughts (ToT)	$\log_k N$	$O(\log_k N)$
<b>Graph of Thoughts (GoT)</b>	$\log_k N$	$N$

# Autonomous Agent

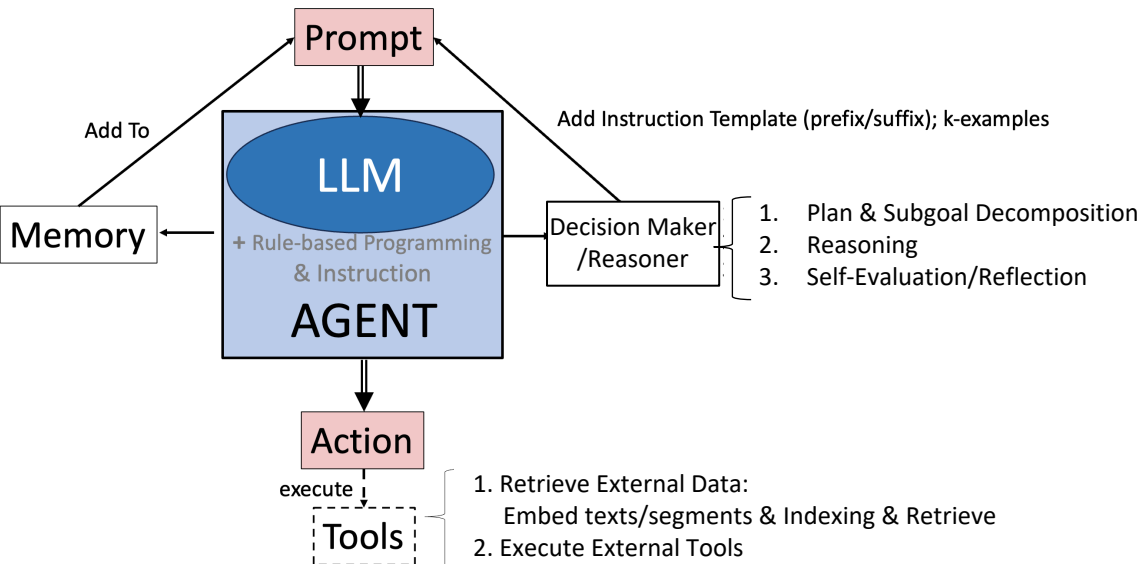
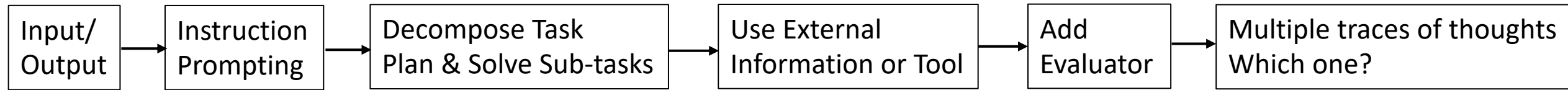
A genuine problem-solving process involves the repeated use of available information to initiate exploration, which discloses, in turn, more information until a way to attain the solution is finally discovered.— — Newell et al. [18]

# III. Create your own independent autonomous LLM-agent

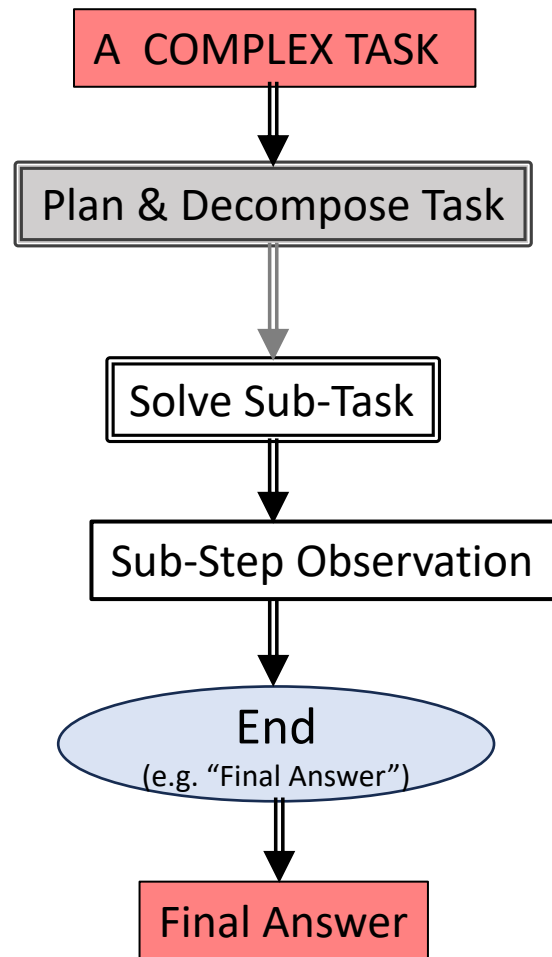
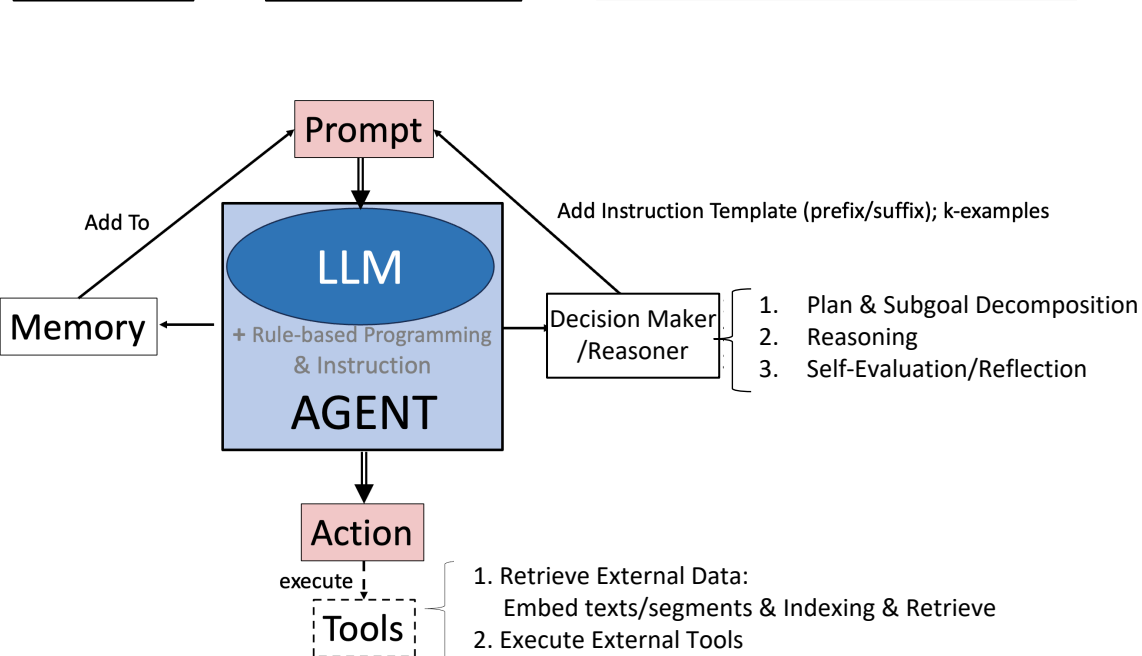
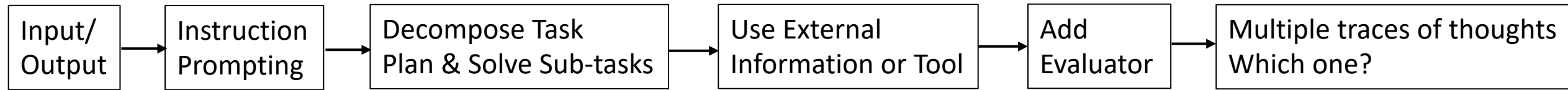




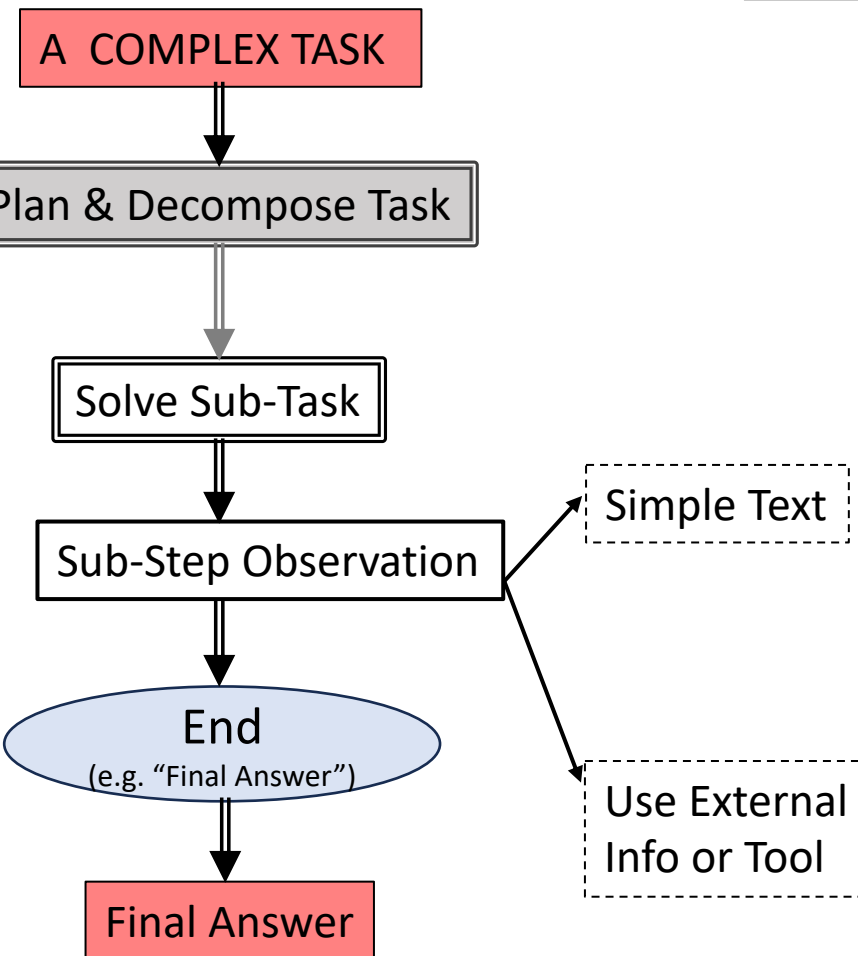
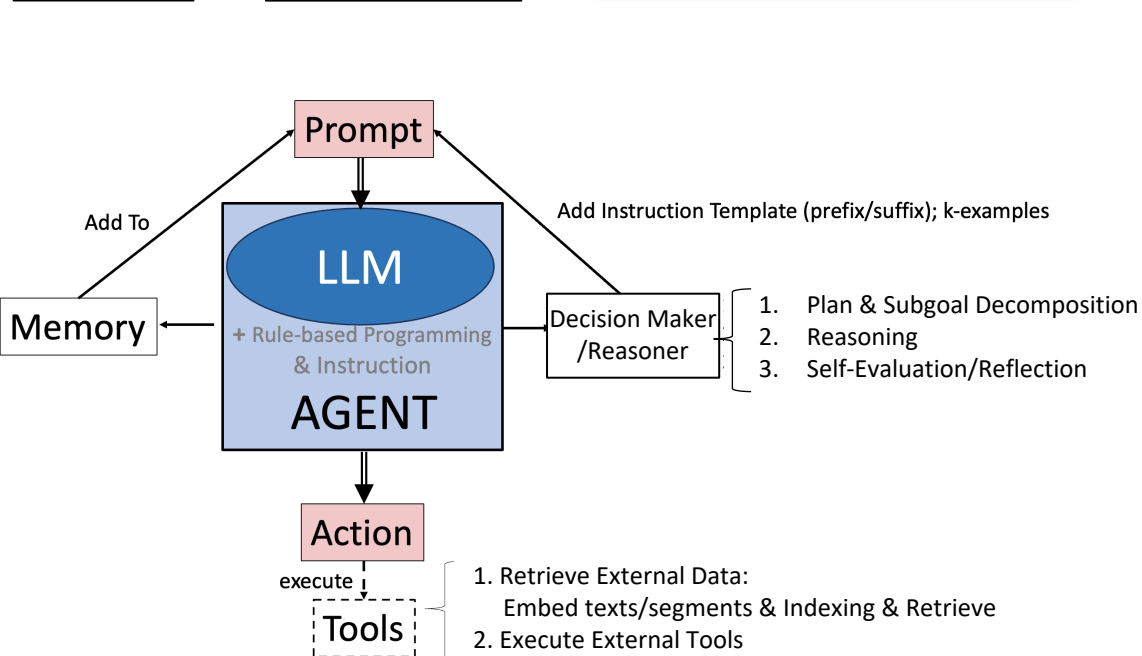
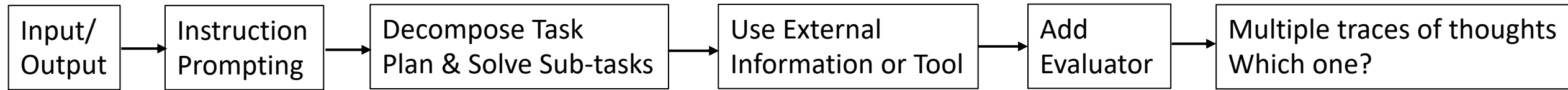
# Critical Components in the LLM Agent



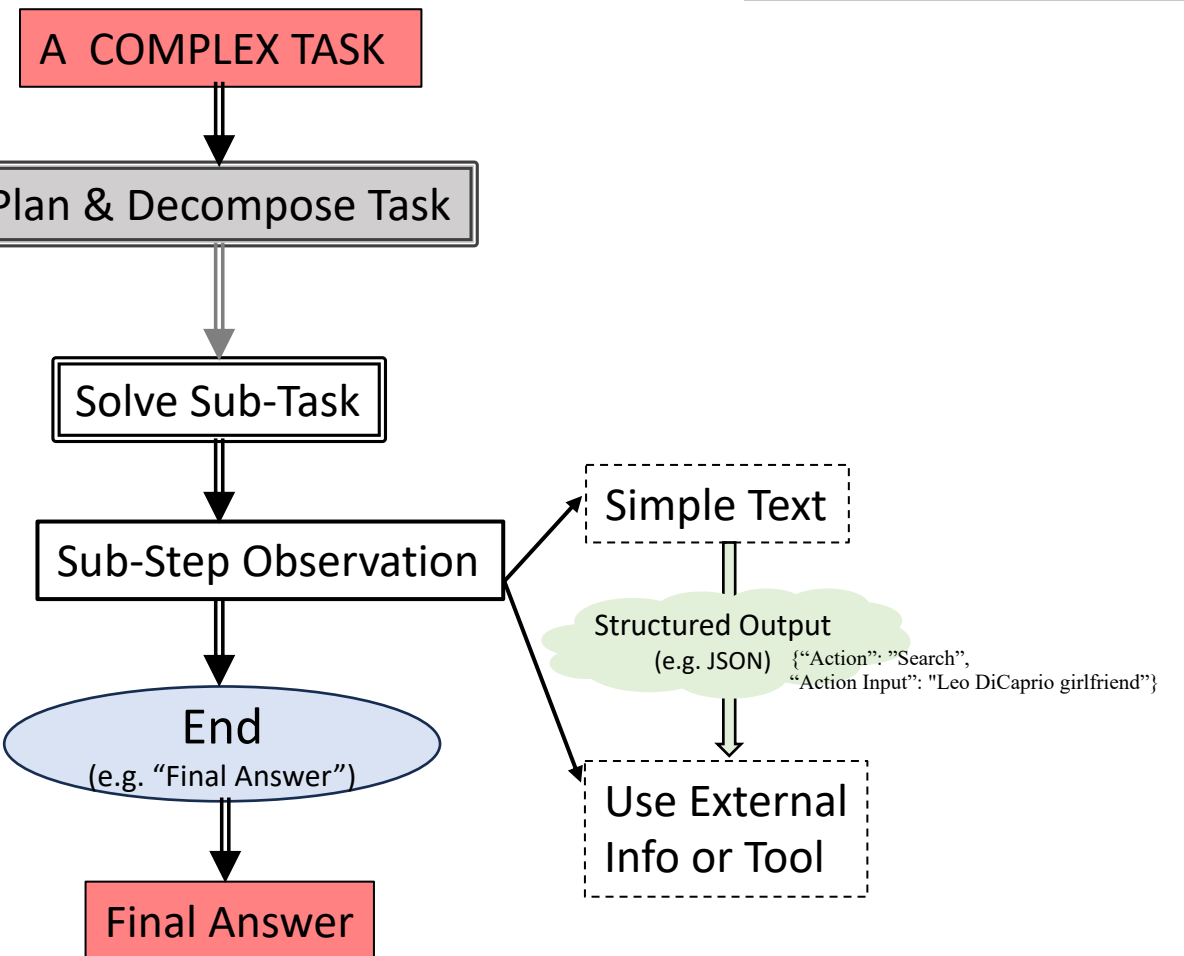
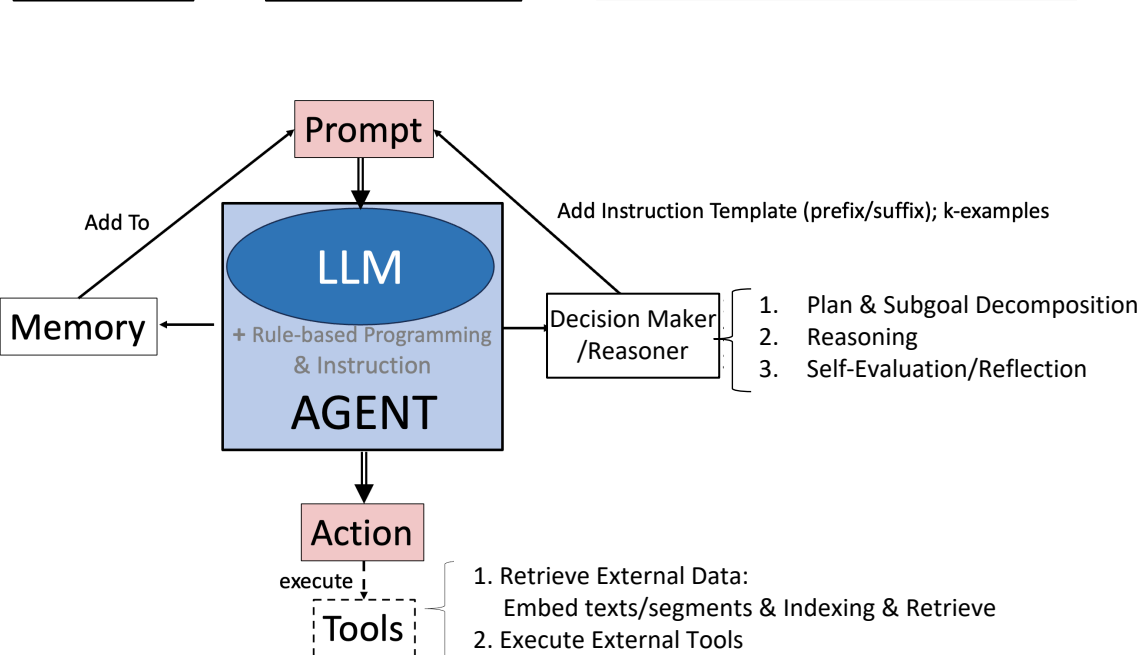
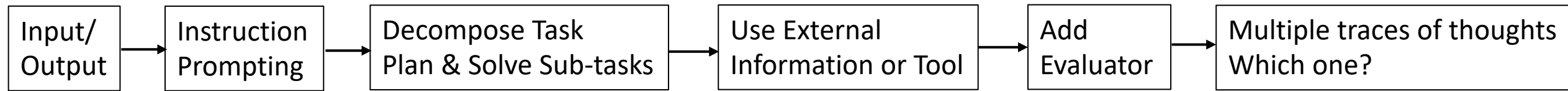
# Automation of Agent: Single Thought Chain



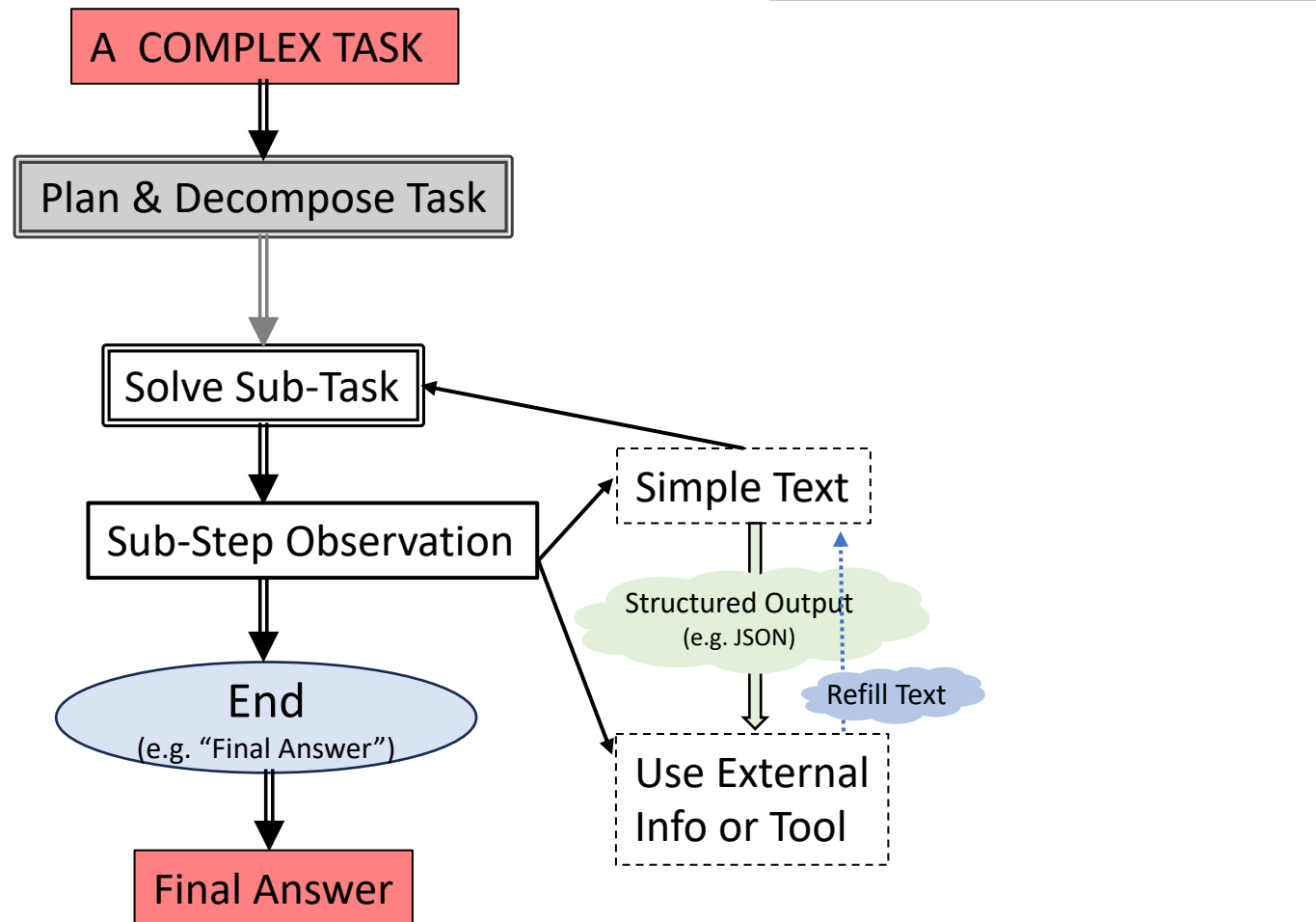
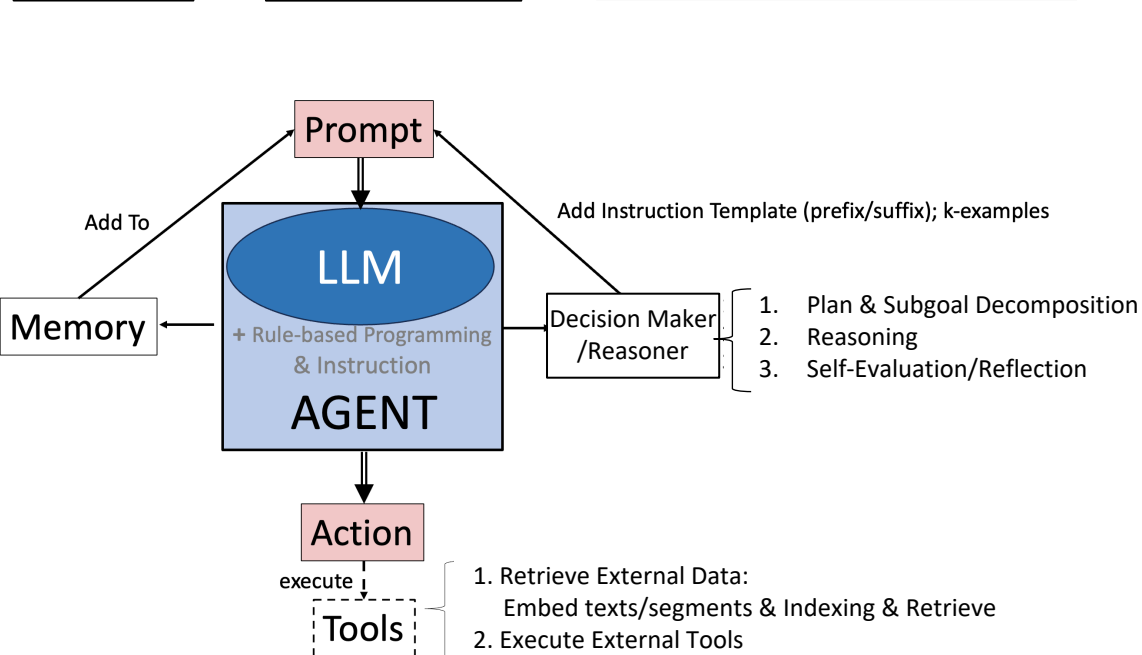
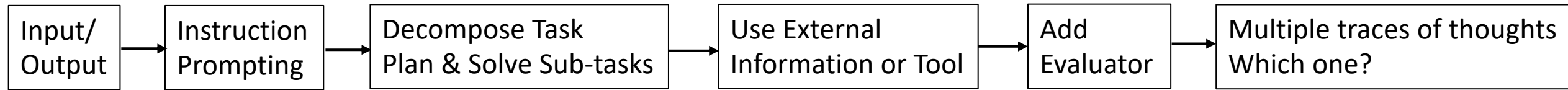
# Automation of Agent: Single Thought Chain



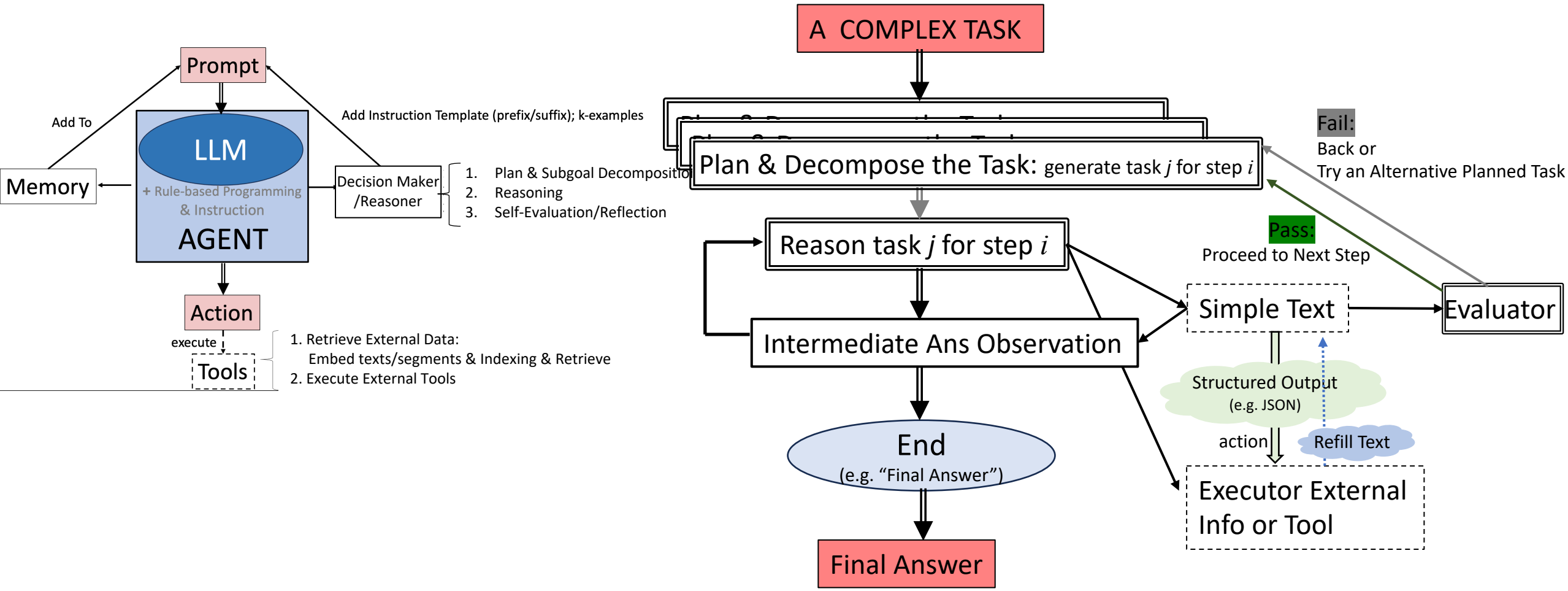
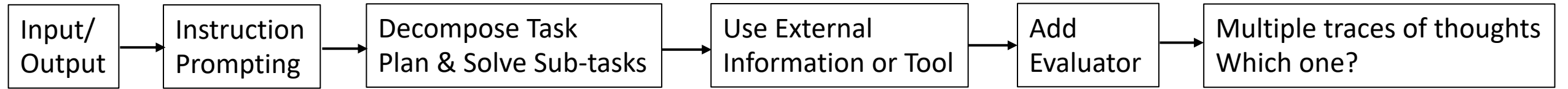
# Automation of Agent: Single Thought Chain



# Automation of Agent: Single Thought Chain



# Automation of Agent: Multiple Thought Chains



# Automation of Agent: Multiple Thought Chains

## A Complete Guide to LLMs-based Autonomous Agents (Part I):



Yule Wang, PhD  
20 min read · Oct 9, 2023

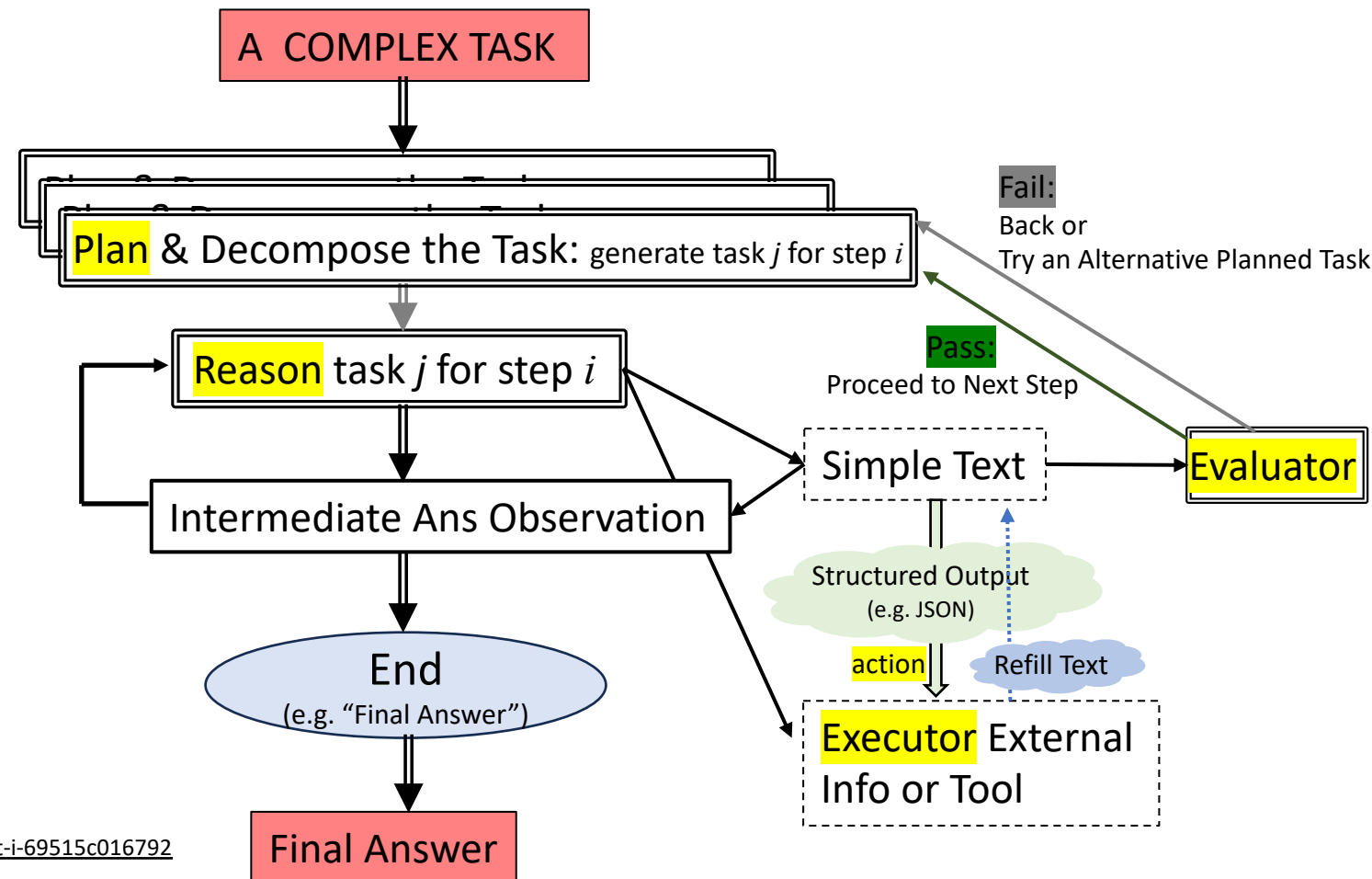
171 5

— Chain of Thought, Plan and Solve/Execute, Self-Ask, ReAct, Reflexion, Self-Consistency, Tree of Thoughts and Graph of Thoughts

Large Language Models (LLMs) provide an intuitive natural language interface, making them ideal for user-computer interactions and addressing complex problems. Some pretrained LLMs, such as GPT-4, come with notable reasoning capabilities, enabling them to break down intricate issues into more simpler steps, offering solutions, actions, and evaluations at each step. This suggests that these LLMs are already adequate to address diverse challenges.

However, being closed systems, LLMs are unable to fetch the most recent data or specific domain knowledge. This limitation can lead to potential errors or “hallucinations” (i.e., generating incorrect responses). While fine-tuning the pretrained LLMs is a potential remedy, it compromises their

<https://medium.com/@yulemoon/a-complete-guide-to-llms-based-autonomous-agents-part-i-69515c016792>



# Prompting?

Instructions: {System General Instruction}  
Previous Data: {Memory & Last Steps Answer}  
Reference Data: {Retrieved Information}  
Respond in the specified JSON format: {JSON Format with descriptions}  
Please replicate the examples to generate the answer:  
{k examples}  
Given question: "{Question}", provide the process leading to the  
answer:

```
# Initially, append the question requiring a resolution to the memory
memory = [Question]
i = 0

while True:
    # Generate candidates thoughts for the current step
    candidates = Planner.generate_candidates(step_i, memory)

    while candidates:
        # Select the most optimal candidate thought step
        candidate_j = Evaluator_Ranker.select_best(candidates)
        # Reasoner reasons this thought step when given the previous trajectory
        reason = Reasoner.process(memory, candidate_j)
        # Provide an action to execute based on the previous reasoning
        action = Actioner.select_an_action(reason)

        # If we find a keyword like "Final Answer", end this iterative loop
        # and return the final solution
        if action == "Final Answer":
            return final_solution

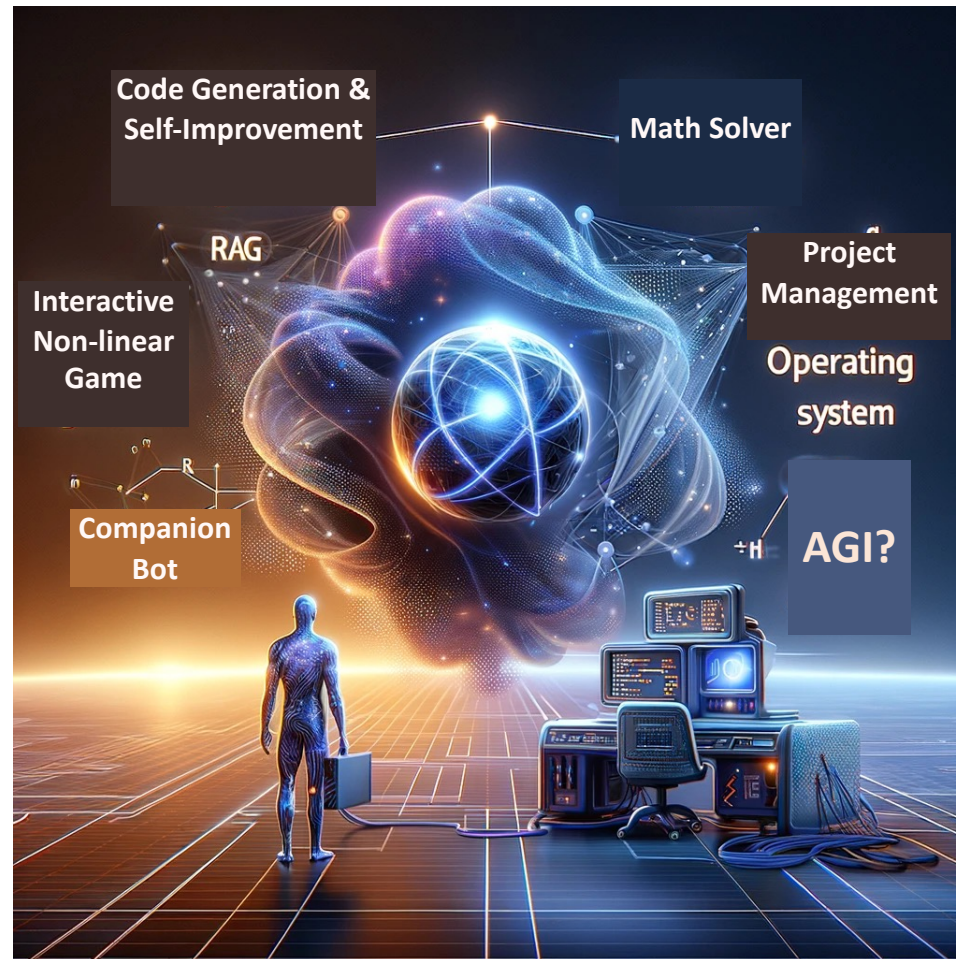
        # This checks if external resource retrieval is allowed.
        if RAG_enabled:
            observation = Executor.execute(action)

        # Reasoner reasons upon the observation of the action
        candidate_j_result = Reasoner.process(memory, candidate_j, observation)

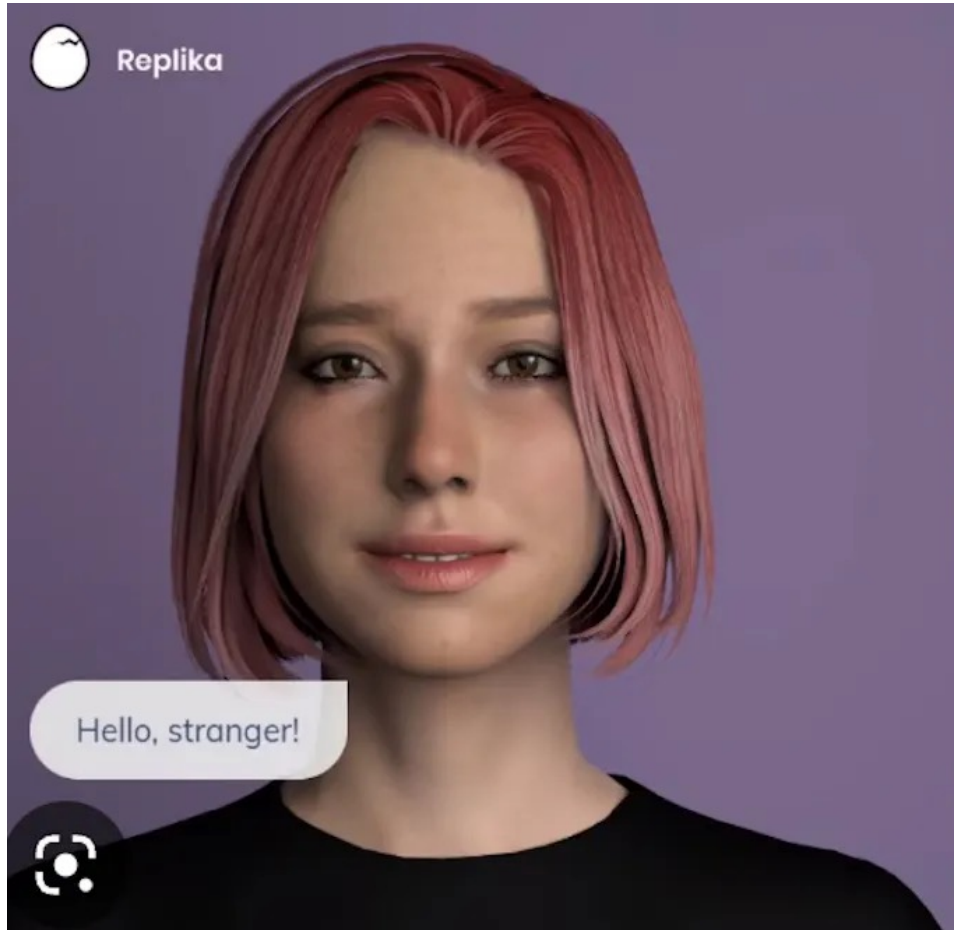
        # Evaluator accesses whether the result from this candidate thought step
        # will lead to a positive direction to the final answer.
        if Evaluator.access(candidate_j_result) is positive:
            i += 1 # Move to the next thought step.
            Memory.append(step_i, candidate_j, candidate_j_result)
            break
        else:
            # If not positive, try an alternative thought step
            candidates.remove(candidate_j)
            if not candidates:
                candidates = Planner.generate_candidates(step_i, memory)
```



# IV. LLM-agent Prospects



# Companion Bot



[Get the app](#) [Help](#) [Log in](#)

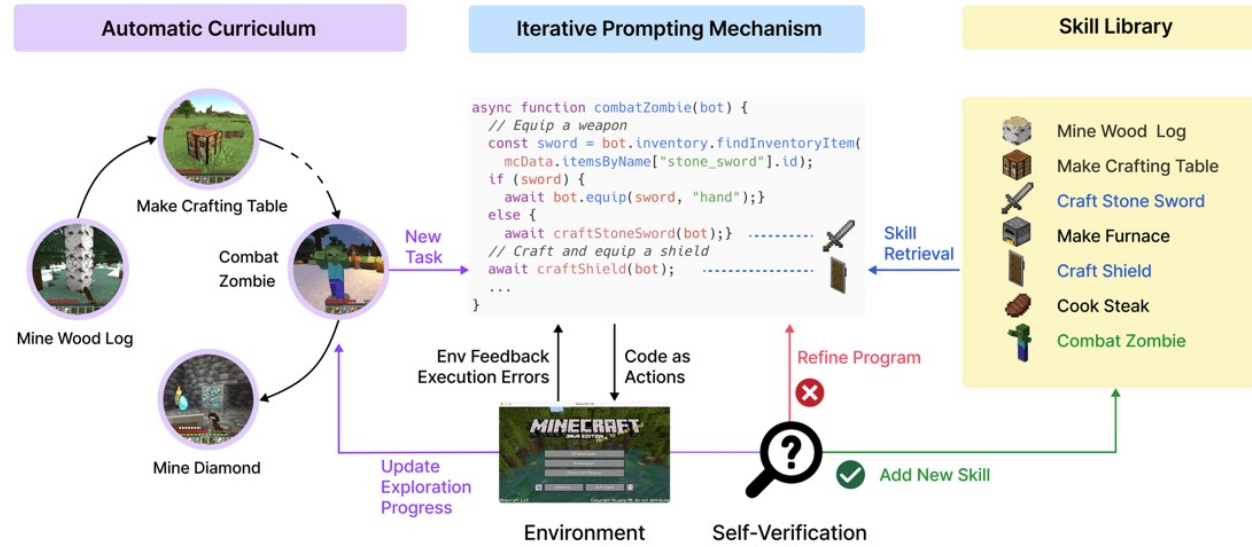
## The AI companion who cares

Always here to listen and talk.  
Always on your side. Join the millions growing with their AI friends now!

Create your Replika

Log in

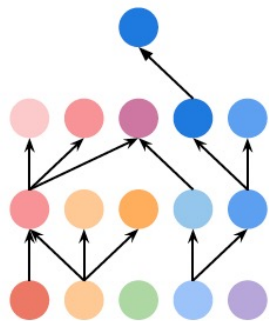
# Interactive Game



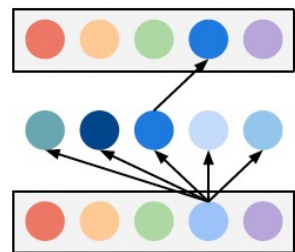
**Voyager consists of three key components:** an automatic curriculum for open-ended exploration, a skill library for increasingly complex behaviors, and an iterative prompting mechanism that uses code as action space.

	Inventory (5/36): {'oak_planks': 3, 'stick': 4, 'crafting_table': 1, 'stone': 3, 'wooden_pickaxe': 1}	GPT-4	<b>Reasoning:</b> Since you have a wooden pickaxe and some stones, it would be beneficial to upgrade your pickaxe to a stone pickaxe for better efficiency. <b>Task:</b> Craft 1 stone pickaxe.
	Biome: river Inventory (4/36): {'sugar_cane': 3, 'paper': 3, 'fishing_rod': 1, 'wooden_pickaxe': 1}	GPT-4	<b>Reasoning:</b> Since you have a fishing rod in your inventory and you are near a river biome, it would be a good opportunity to catch some fish for food and experience. <b>Task:</b> Catch 1 fish.
	Nearby entities: pig, cat, villager Health: 12/20 Hunger: 0/20	GPT-4	<b>Reasoning:</b> Your hunger is at 0, which means you need to find food to replenish your hunger. Since there are pigs nearby, you can kill one to obtain raw porkchops. <b>Task:</b> Kill 1 pig.
	Inventory (6/36): {'furnace': 1, 'stone_pickaxe': 1, 'oak_planks': 7, 'crafting_table': 1, 'raw_iron': 4, 'coal': 1}	GPT-4	<b>Reasoning:</b> You have raw iron and coal, and you have a furnace. It's time to smelt the iron to make iron ingots, which can be used to craft better tools and armor. <b>Task:</b> Smelt 4 raw iron.
	Time: night Nearby entities: zombie Equipment: 'stone_sword', 'shield'	GPT-4	<b>Reasoning:</b> Since it's night and there's a zombie nearby, it's a good opportunity to try killing the zombie now that you have a stone sword and a shield equipped. <b>Task:</b> Kill 1 zombie.

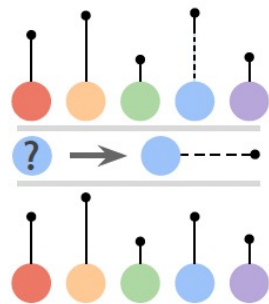
# Code Generation Self-Improvement



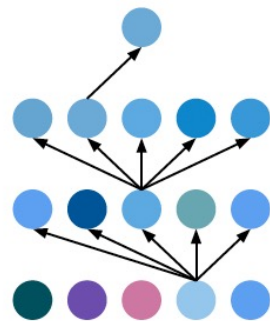
Genetic Algorithm



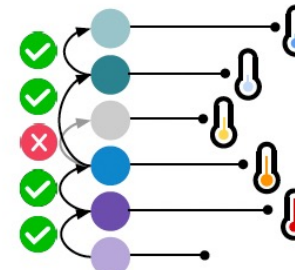
Decomposing and Improving Parts



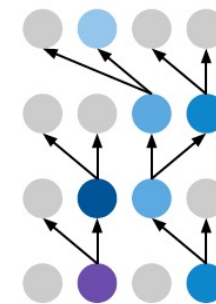
Multi-Armed Prompt Bandit



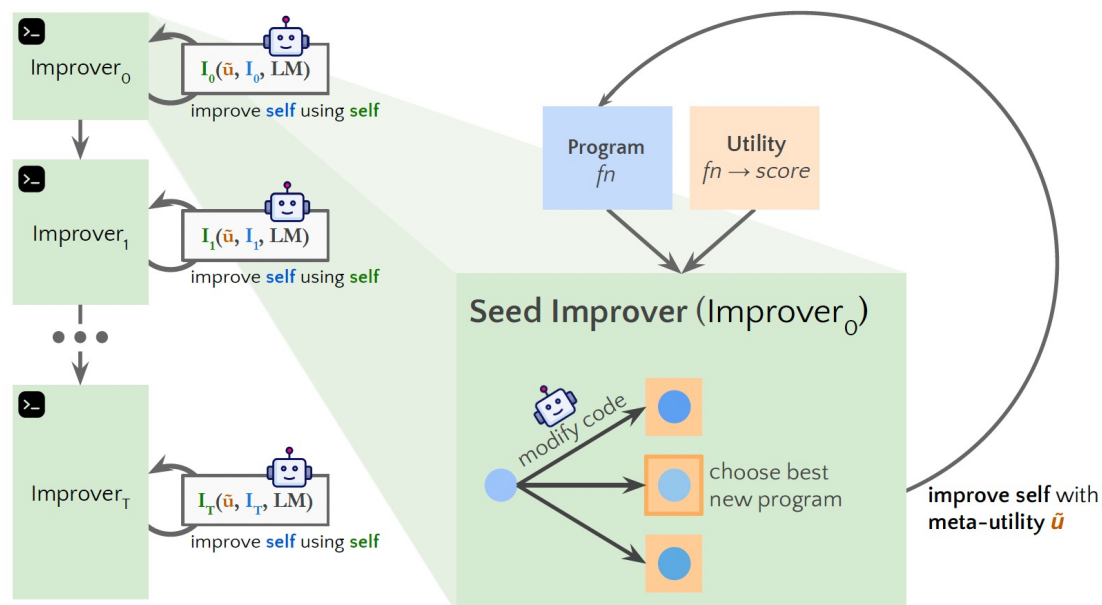
Vary Temperature to Explore



Simulated-annealing Based Search

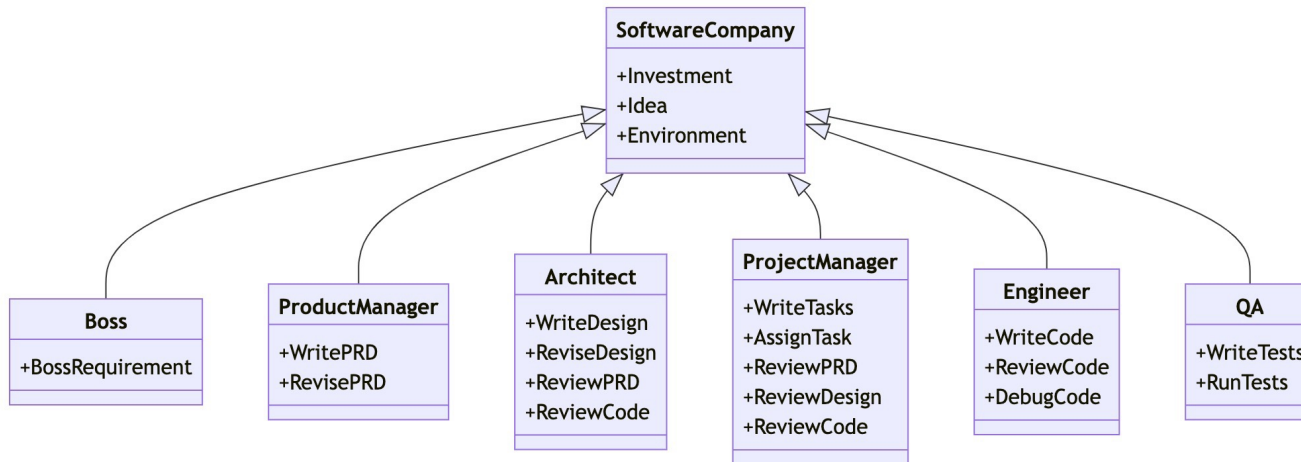


Beam Search / Tree Search



[15] "STOP"

# Engineering Project Management



```
class ThoughtNode(Node):
    """A node representing a thought in the thought tree."""

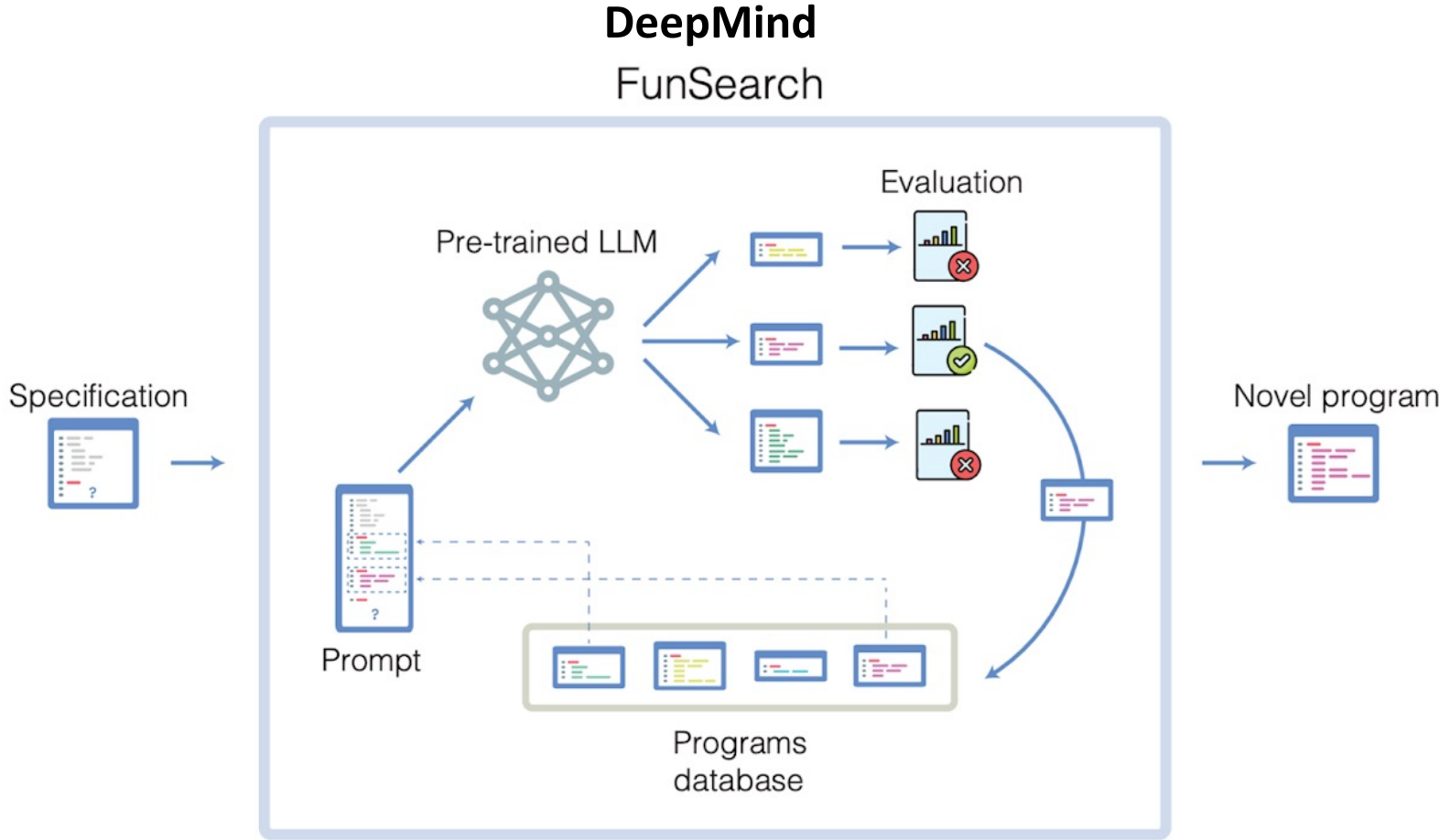
    name: str = ""
    value: int = 0
    id: int = 0
    valid_status: bool = True

    def update_value(self, value) -> None:
        """Update the value of the thought node."""
        self.value = value

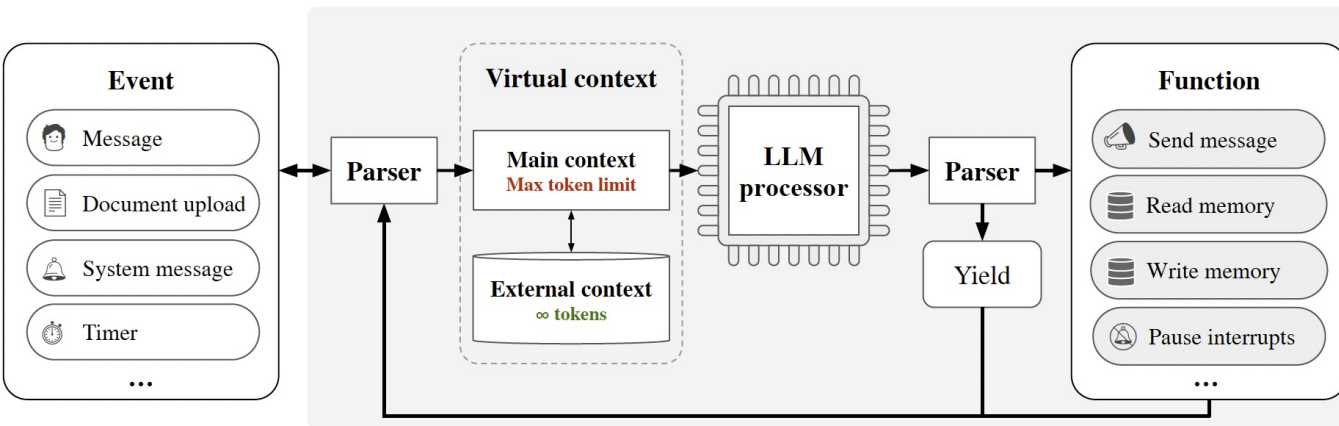
    def update_valid_status(self, status) -> None:
        """Update the validity status of the thought node."""
        self.valid_status = status
```

<https://github.com/geekan/MetaGPT/>

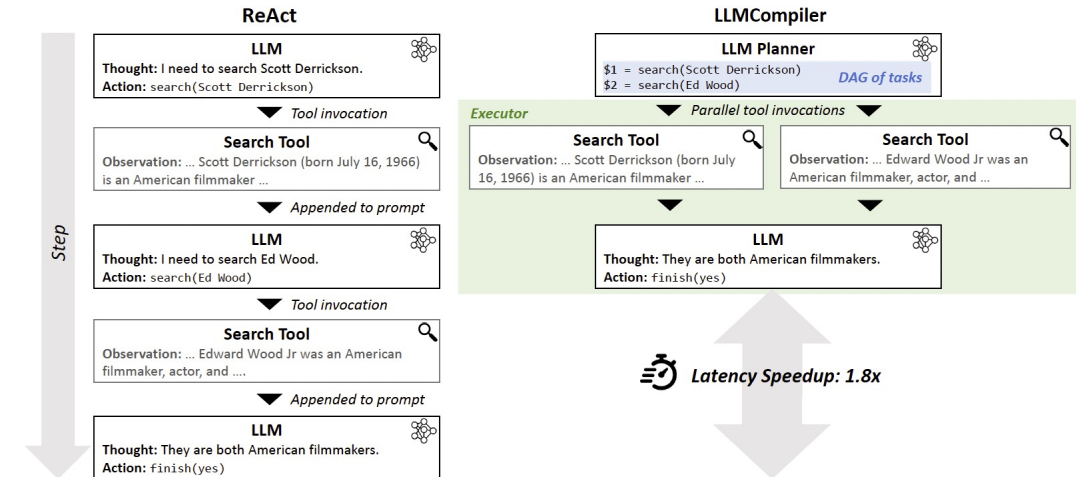
# Math Problem Solver



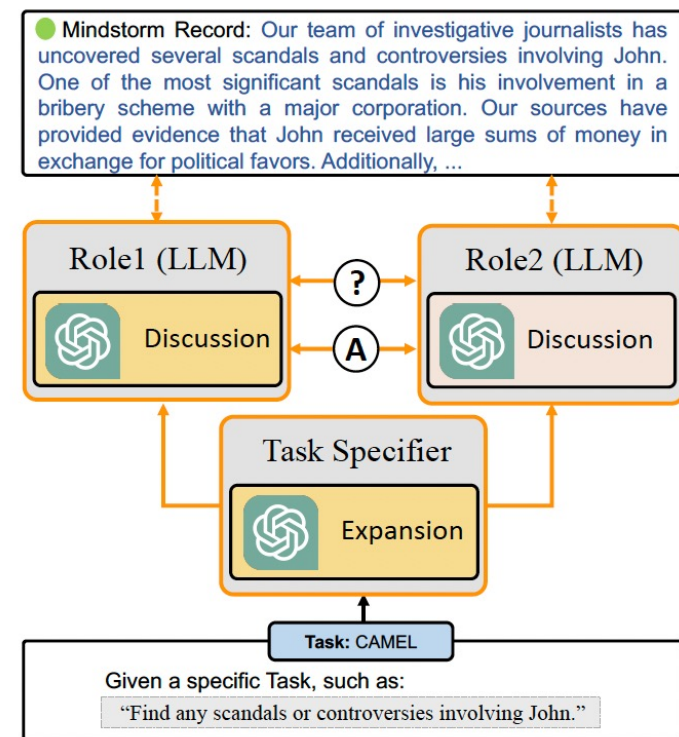
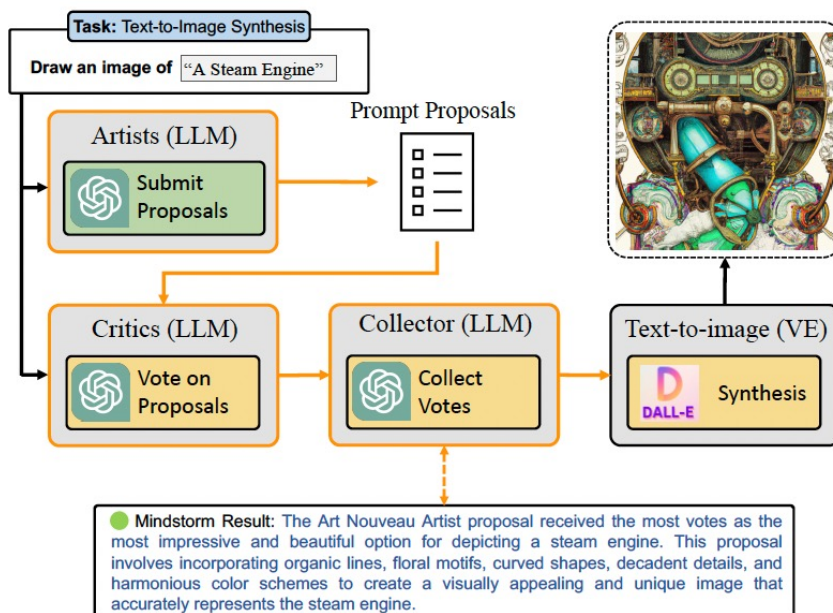
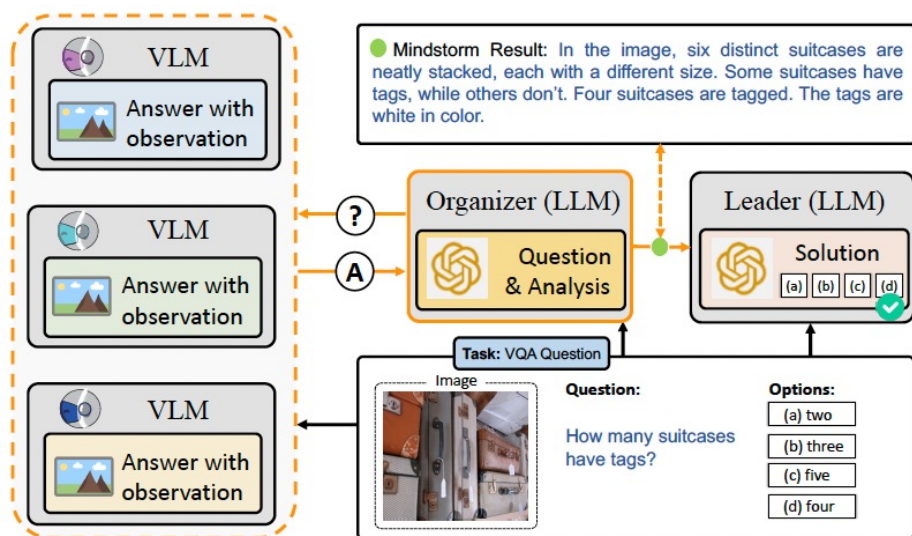
# Operating System



HotpotQA Question: Were Scott Derrickson and Ed Wood of the same nationality?



# One Step Towards AGI (Natural Language-based Society of Minds)





## Numbered Articles:

1. [Xi et al. \(2023\) The Rise and Potential of Large Language Model Based Agents: A Survey](#)
2. [Kojima et al. \(2022\) Large Language Models are Zero-Shot Reasoners](#)
3. [Wei et al. \(2022\) Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#)
4. [Zhang et al. \(2022\) Automatic Chain of Thought Prompting in Large Language Models](#)
5. [Wang et al. \(2023\) Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models](#)
6. [Press et al. \(2023\) Measuring and Narrowing the Compositionality Gap in Language Models](#)
7. [Yao et al. \(2023\) REACT: SYNERGIZING REASONING AND ACTING INLANGUAGE MODELS](#)
8. [Madaan et al. \(2023\) Self-refine: SELF-REFINE:Iterative Refinement with Self-Feedback](#)
9. [Shinn et al. \(2023\) Reflexion: Language Agents with Verbal Reinforcement Learning](#)
10. [Wang et al. \(2023\) Self-Consistency Improves Chain of Thought Reasoning in Language Models](#)
11. [Besta et al. \(2023\) Graph of Thoughts: Solving Elaborate Problems with Large Language Models](#)
12. [Yao et al. \(2023\) Tree of Thoughts: Deliberate Problem Solving with Large Language Models](#)
13. [Newell et al. "Report on a general problem solving program" Pittsburgh, PA, 1959](#)
14. [Wang G et al. \(2023\) Voyager: An open-ended embodied agent with large language models\[J\].](#)
15. Zelikman, Eric, et al. (2023) "Self-taught optimizer (stop): Recursively self-improving code generation."
16. Romera-Paredes, Bernardino, et al. "Mathematical discoveries from program search with large language models." Nature (2023): 1-3.
17. [Packer, Charles, et al. \(2023\) "Memgpt: Towards llms as operating systems."](#)
18. [Kim, Sehoon, et al. \(2023\) "An LLM Compiler for Parallel Function Calling."](#)
19. [Zhuge, Mingchen, et al. \(2023\) "Mindstorms in Natural Language-Based Societies of Mind."](#)

## Other Resources:

<https://www.promptingguide.ai/>

<https://docs.langchain.com/docs/>

<https://lilianweng.github.io/posts/2023-06-23-agent/>

<https://www.pinecone.io/learn/series/langchain/>