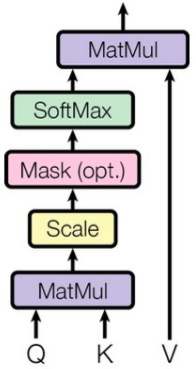
The Story of an Agent

Presented by:
John Tan Chong Min

I am a Large Language Model.

I match output tokens to input tokens I see.

Scaled Dot-Product Attention



Task: John has 4 marbles. Mary has 3 balls.
John gave Mary 2 marbles.
How many marbles does Mary have?

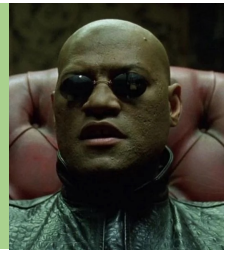
I have no idea what you are talking about…. Erm…

0?

<u>Mysterious Man:</u>

Maybe you need to…

Think step by step



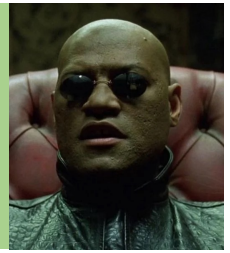
Task: John has 4 marbles. Mary has 3 balls.
John gave Mary 2 marbles.
How many marbles does Mary have?
Think step by step.

Mary has 3 balls.
John gave Mary 2 marbles.
So Mary has 3 + 2 = 5 marbles

Mysterious Man:

Maybe you need to…

Reflect



Task: John has 4 marbles. Mary has 3 balls.
John gave Mary 2 marbles.
How many marbles does Mary have?

AI: Mary has 3 balls.
John gave Mary 2 marbles.
So Mary has 3 + 2 = 5 marbles

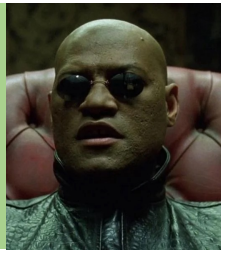
Inner voice: Balls are not marbles.
Reflect and correct your answer.

Mary had 0 marbles, and
received 2 from John.
So Mary has 0 + 2 = 4 marbles

Mysterious Man:

Maybe…

Use a tool



AI: Mary had 0 marbles, and received 2 from John. So Mary has 0 + 2 = 4 marbles

Inner voice: You can use the following tools:

1. Name: Calculator, Use Case: For adding numbers, Input: 2 numbers, Output: Sum of 2 numbers
2. Name: Image Generator, Use Case: To generate images with text prompt, Input: text prompt, Output: Image

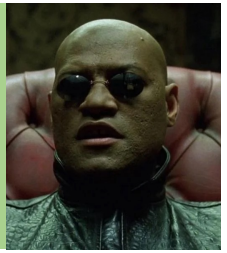
Which tool would you like to use?

Calculator Tool

Mysterious Man:

Maybe…

Get the inputs?



AI: Mary had 0 marbles, and received 2 from John. So Mary has 0 + 2 = 4 marbles

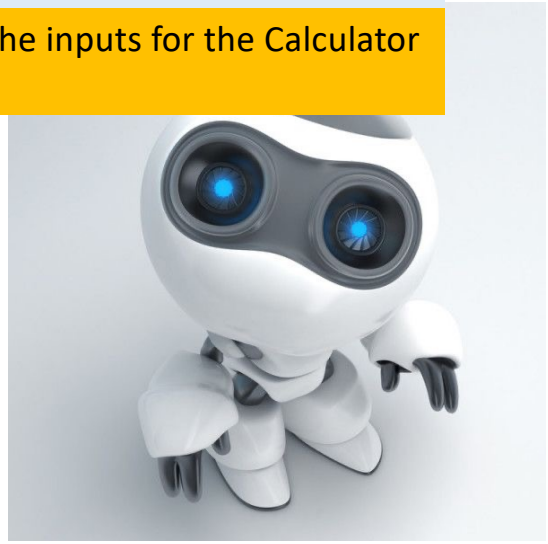
Inner voice: You can use the following tools:

1. Name: Calculator, Use Case: For adding numbers, Input: 2 numbers, Output: Sum of 2 numbers
2. Name: Image Generator, Use Case: To generate images with text prompt, Input: text prompt, Output: Image

Which tool would you like to use?

AI: Calculator Tool

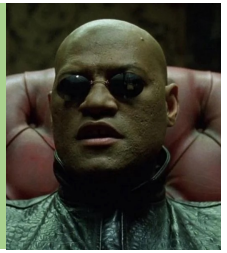
Inner voice: What are the inputs for the Calculator Tool?



The inputs are two numbers, 0 and 2

Mysterious Man:

Maybe…
Use a more structured template?



AI: Mary had 0 marbles, and received 2 from John.
So Mary has 0 + 2 = 4 marbles

Inner voice: You can use the following tools:
1. Name: Calculator, Use Case: For adding numbers, Input: 2 numbers, Output: Sum of 2 numbers
2. Name: Image Generator, Use Case: To generate images with text prompt, Input: text prompt, Output: Image

Which tool would you like to use?
Answer in json format:
{“name”: “name of tool”, “input list”: “list of inputs”}

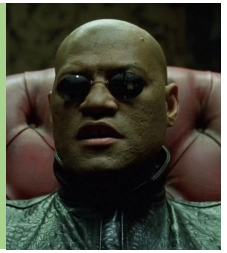
{ "name": "Calculator",
"input list": ["0", "2"] }

#strictjson (Check it out!)

Mysterious Man:

You need to discover for yourself…

The ~~Matrix~~ Environment

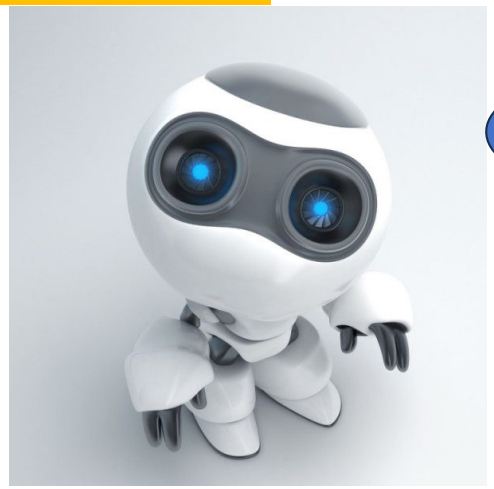


AI: { "name": "Calculator",
"input list": ["0", "2"] }

Environment Observation: 2

React:
Thought (step by step)
Action (Tool),
Observation (Tool/Env Feedback)

+
Reflection (optional - Reflexion)

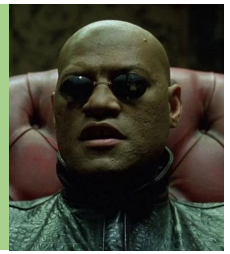


Mary has 2 marbles.

Mysterious Man:

At last…

The truth is out.



AI: { "name": "Calculator",
"input list": ["0", "2"] }

Environment Observation: 2

AI: Mary has 2 marbles.

User: What was the initial question?

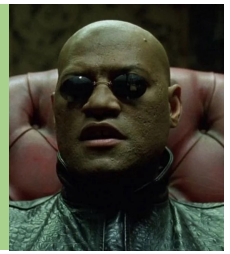
Erm…

I totally forgot…

Mysterious Man:

Maybe you need…

Memory



Task: John has 4 marbles. Mary has 3 balls.
John gave Mary 2 marbles.
How many marbles does Mary have?

AI: Mary has 3 balls.
John gave Mary 2 marbles.
So Mary has 3 + 2 = 5 marbles

Inner voice: Balls are not marbles.
Reflect and correct your answer.

AI: Mary had 0 marbles, and received 2 from John.
So Mary has 0 + 2 = 4 marbles

Inner voice: You can use the following tools:
1. Name: Calculator, Use Case: For adding numbers, Input: 2 numbers, Output: Sum of 2 numbers
2. Name: Image Generator, Use Case: To generate images with text prompt, Input: text prompt, Output: Image

Which tool would you like to use?
Answer in json format:
{“name”: “name of tool”, “input list”: “list of inputs”}

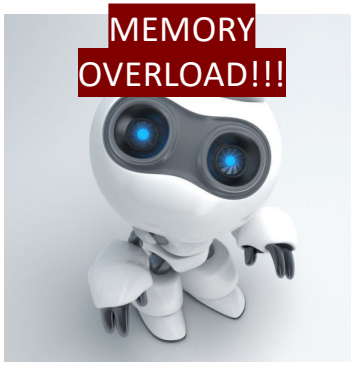
Memory

AI: { "name": "Calculator",
"input list": ["0", "2"] }

Environment Observation: 2

AI: Mary has 2 marbles.

User: What was the initial question?

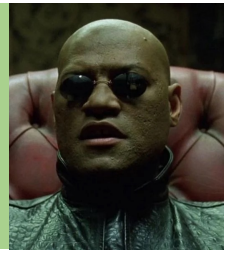


John has 4 marbles. Mary has 3 balls.
John gave Mary 2 marbles.
How many marbles does Mary have?

Mysterious Man:

This cannot scale.

Use ~~The Matrix~~ External Storage



Inner voice: You can use the following tools:

1. Name: Calculator, Use Case: For adding numbers, Input: 2 numbers, Output: Sum of 2 numbers
2. Name: Image Generator, Use Case: To generate images with text prompt, Input: text prompt, Output: Image

Which tool would you like to use?
Answer in json format:
{“name”: “name of tool”, “input list”: “list of inputs”}

AI: { "name": "Calculator",
"input list": ["0", "2"] }

Environment Observation: 2

AI: Mary has 2 marbles.

User: What was the initial question?

Short-term Memory
(Past X conversations)

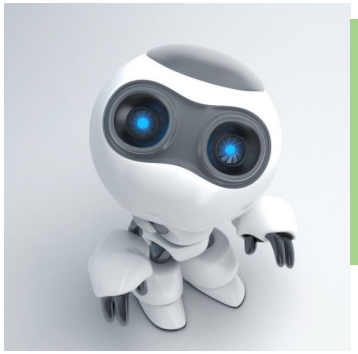
The initial question was about how many marbles Mary has after John gave her 2 marbles, given that John initially had 4 marbles, and Mary had 3 balls.

The conversation clarified that Mary had 0 marbles before receiving the 2 marbles from John, and thus, she now has 2 marbles.

Long-term Memory
(Summarised Conversation)

Task: John has 4 marbles. Mary has 3 balls.
John gave Mary 2 marbles.
How many marbles does Mary have?

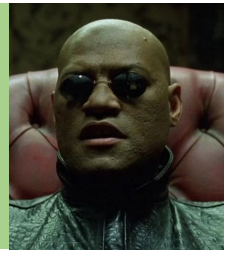
Similar Context
(Retrieved from
Top K Messages based on
Relevance to query)



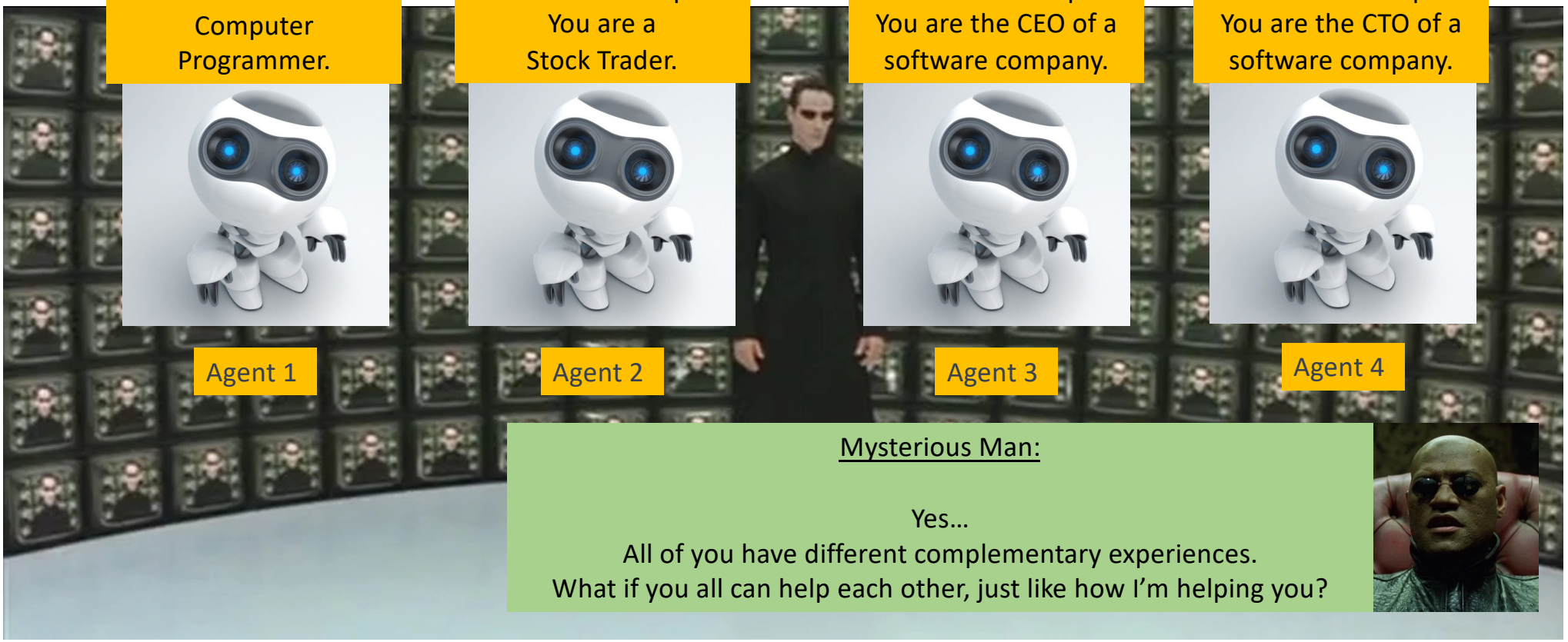
<u>Mysterious Man:</u>

You may be The One…

But what if there are more of you?

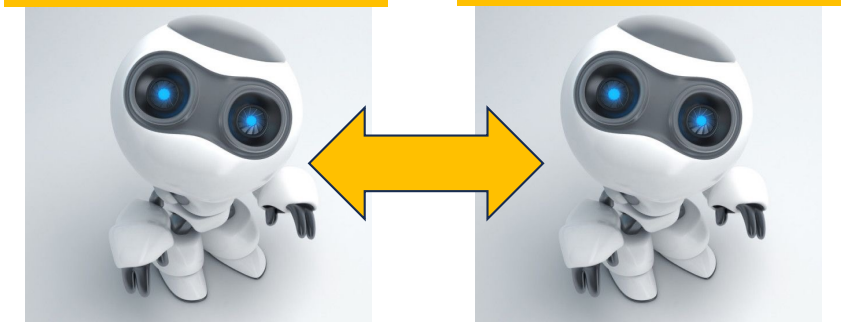


Zero-shot/Few-shot role prompting



Behavior Prompt:
You are a
Computer
Programmer.

Behavior Prompt:
You are a
Stock Trader.



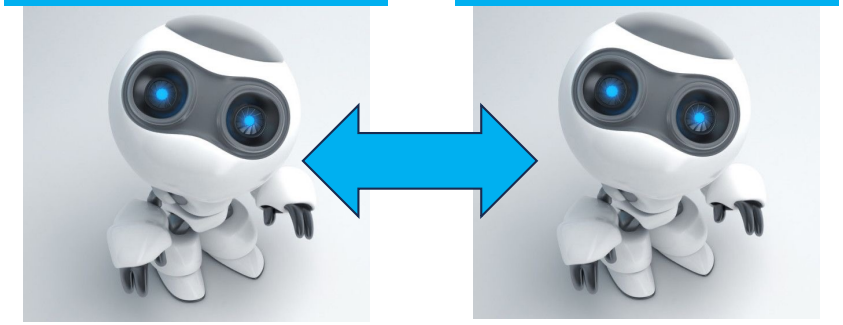
Agent 1A
- User

Agent 1B
- Assistant

Task: Create a trading bot

Behavior Prompt:
You are the CEO of a
software company.

Behavior Prompt:
You are the CTO of a
software company.



Agent 2A
- User

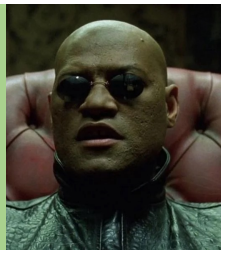
Agent 2B
- Assistant

Task: Create a computer software

<u>Mysterious Man:</u>

More….

We need more…



Systems-level approach to LLMs

We can be more specialised and together, as a system, we can solve complex tasks

Mysterious Man:

Can this scale up even more…?

Or are we missing something?

Hierarchical

Parallel

- Now showing -

The Agents Revolution

Know your limits…

Know your strengths…

Use it wisely

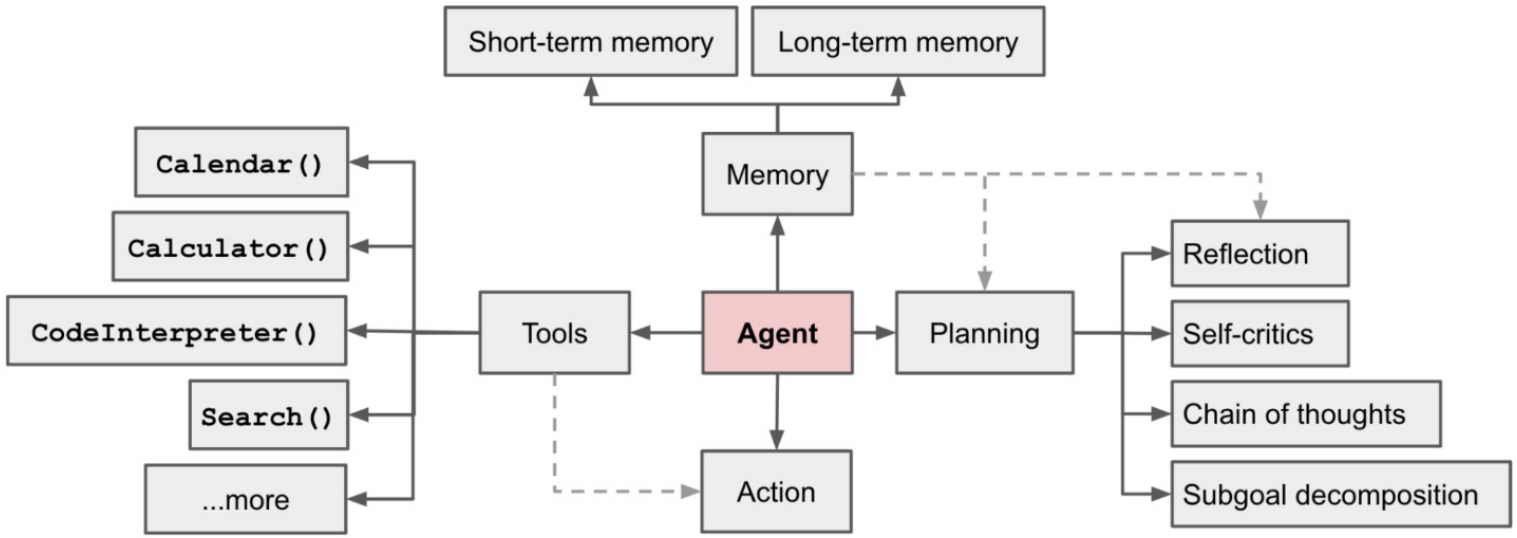
What are agents?

Lilian Weng
@lilianweng

Agent = LLM + memory + planning skills + tool use

This is probably just a start of a new era :)

Agent Overview



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

Chain of Thought and Abstraction Spaces

- Hard to get from input to output directly, especially if the task is difficult

- Use **Chain of Thought** to prompt the LLM to think in a sequential manner
  - “Let’s think step by step”
  - Or provide the steps manually by asking it to generate responses from broad categories to specific categories

- Some problems are better solved at **different levels of granularity**
  - Can have different Chain of Thoughts to different abstraction spaces for different agents
  - ARC Challenge Multiple Agents with Different Abstraction Spaces (mine)

Incorporating World Feedback

The problem with LLMs

LLMs typically do not experience the world

Would need some feedback for them to understand what is going on externally

How to incorporate World Feedback

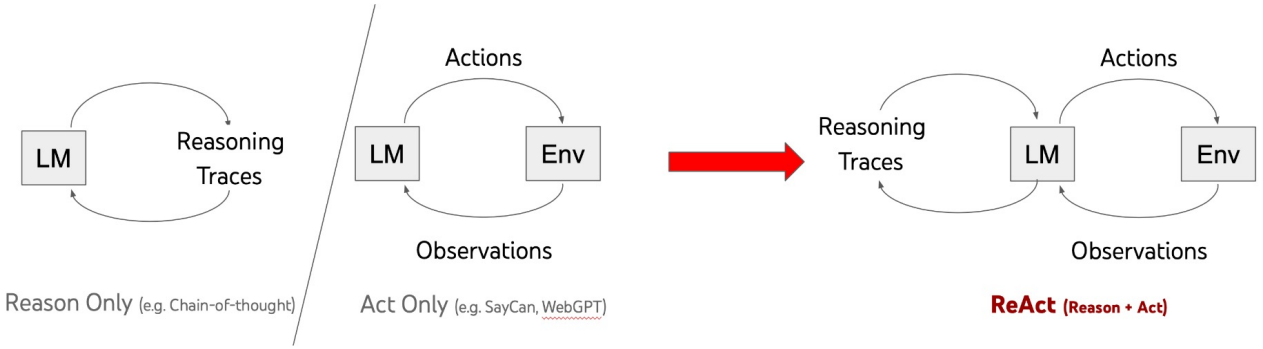


Image from: https://react-lm.github.io/

- ReAct: Chain of thought prompting plus environment grounding
  - Thought, Action, Observation

- Reflexion: ReAct plus “Reflect on your answer”

Voyager – Iterative Prompting with World Feedback

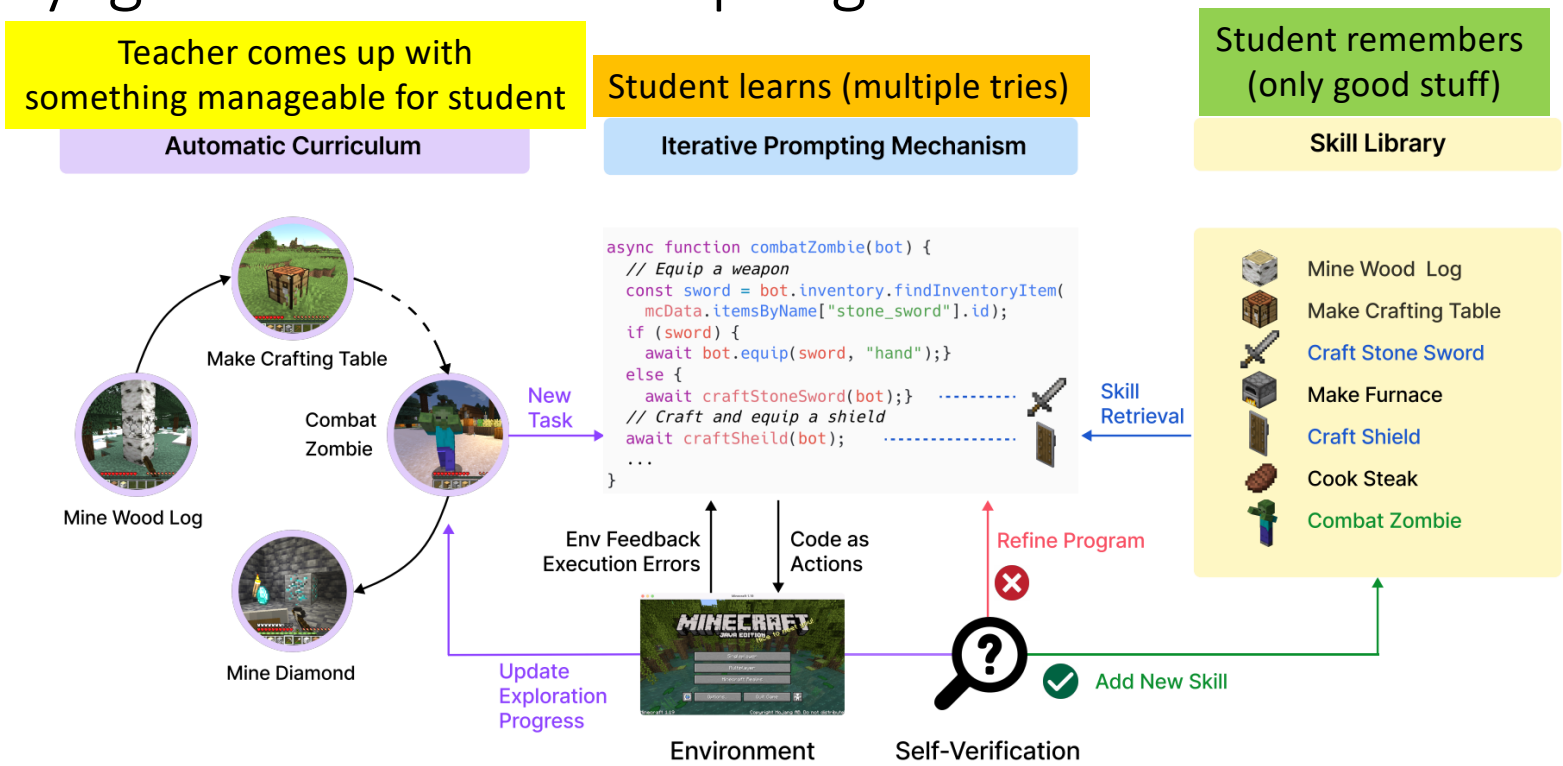


Figure 2: VOYAGER consists of three key components: an automatic curriculum for open-ended exploration, a skill library for increasingly complex behaviors, and an iterative prompting mechanism that uses code as action space.

Tool / Function / API / Skill Use

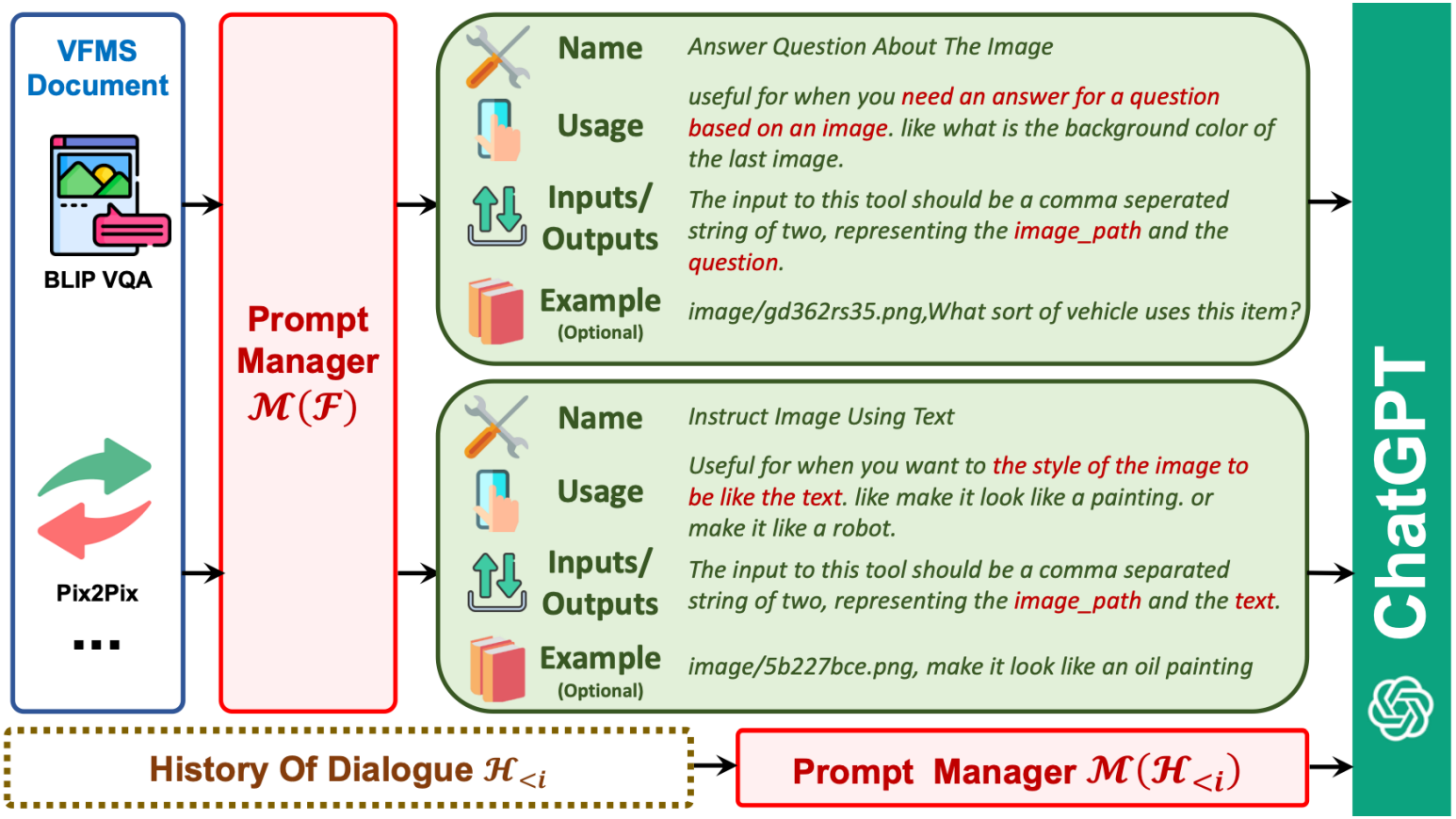
How to decide which tool to use

- Can go through each tool one by one to ask LLM if it is relevant to the query
  - Can ask it to rate on a scale of 0 – 3:
    - 0 – Not Useful
    - 1 – Useful to a limited extent, may provide background information to help solve the query
    - 2 – Moderate Usefulness, helps greatly but is not sufficient to solve fully
    - 3 – High Usefulness, can solve fully

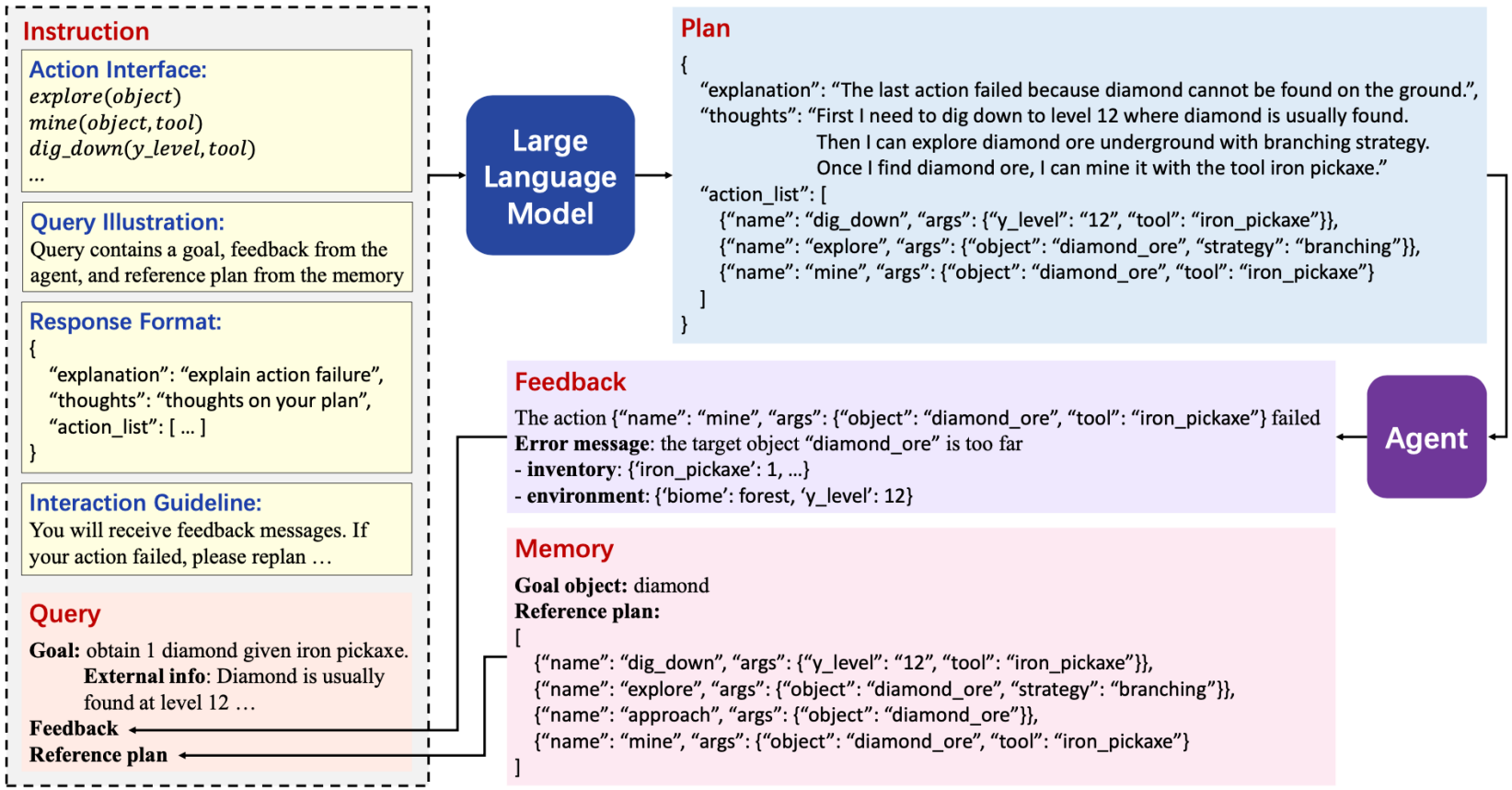
- Can also do as a group by description of tool and use case of tool to ask LLM to select

- Can also group tools into categories to save inference costs. Do a two-step process:
  - Ask LLM to classify which category of tools will be most helpful
  - Then ask LLM to choose a list of tools that will be helpful within that category

Visual ChatGPT - Tool Usage Format



Ghost in the MineCraft – Tool Usage Format



Format for Tool Calling

Can use the Tool Name + Parameters format

Example:

Visual ChatGPT

Ghost in the MineCraft

Can also perhaps do it by Python Code to ask LLM to generate a desired code for tool use (not advisable)

Example:

Voyager

Tool Learning

Perhaps we could create new tools to use by compositing the earlier tools into a meta one

Zoom tool can be composed with a cropping tool followed by a reshape tool

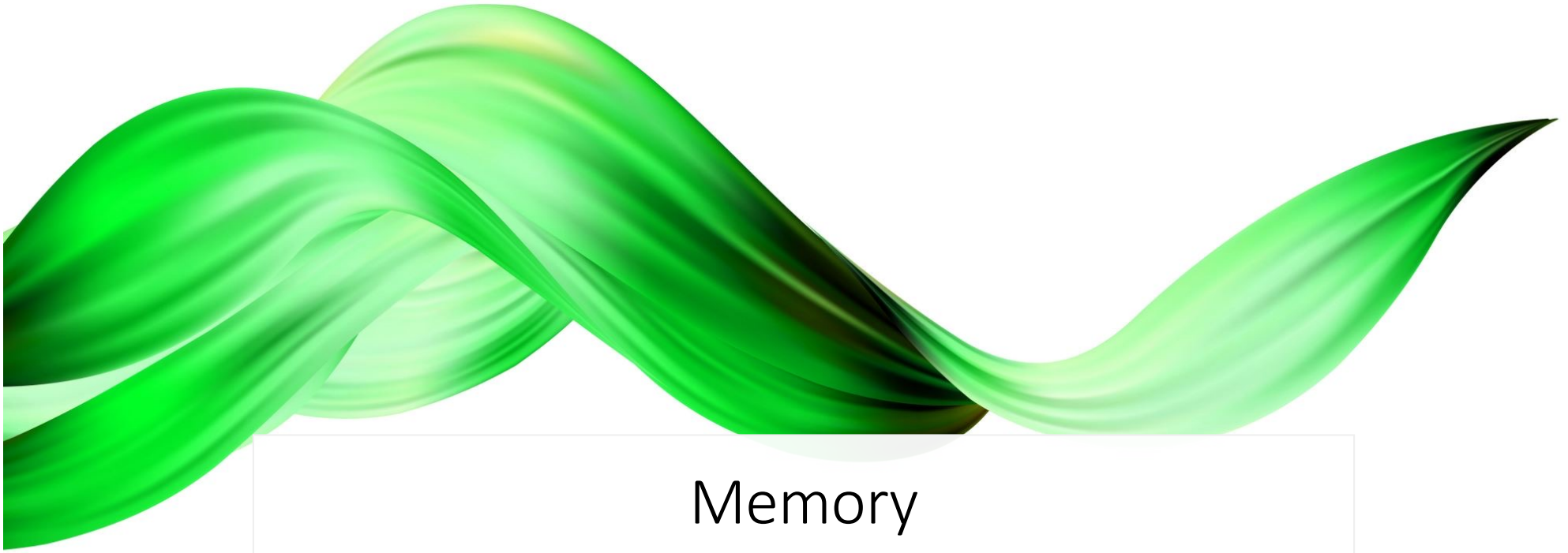
Perhaps new tools can be created by grouping together most commonly used tools

An agent could be asked to analyse past trends to propose new tools

Note: Need not be LLM, can be rule-based

Initial tools may be learned by taking successful workflows and identifying common patterns

Example: Ghost in the MineCraft



Memory

Better ground responses using past knowledge

Retrieval Augmented Generation

- <Retrieved Context 1>
- <Retrieved Context 2>
- <Retrieved Context 3>

- <Query>

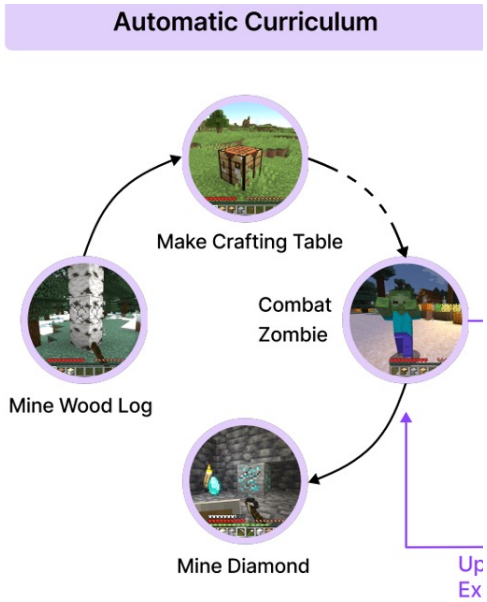
- **Question: Do we only want to retrieve from memory, or modify to fit current context as well?**

Short-term and Long-term Memory

- Short-term: Conversation history of last X turns
- Long-term: Summarised conversation history
- Relevance-based Memory: Retrieves relevant memories based on similarity to query
- So far more geared towards memory of input domain (text)
  - Could be better if we store memory at multiple abstraction layers

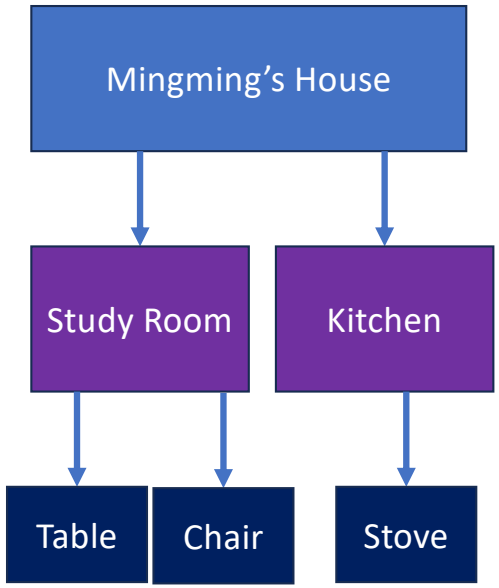
Past Experiences

- Can add in some examples of what has worked, and what has not worked
  - Ground LLM to choose things which work
  - Example: Voyager Automatic Curriculum, DeepMind's prompt-based optimiser
- Can potentially lead to improvement of the quality of responses, provided the LLM is already capable of generating it
- Example context:
  - What has worked: X1, X2, X3
  - What has not worked: A1, A2, A3



Knowledge Graphs and LLMs

- Knowledge Graphs can help select relevant context for the agent by selecting the appropriate level of the graph relevant to the query
- Different levels of the knowledge graph (macro to micro) can be provided as context
- Knowledge Graphs can also serve as a way to form the basis upon which we can build more knowledge from the Agent's experience



Multi-agent systems

A system of both rule-based and LLM-based agents

Benefits and Drawbacks of LLMs

Benefits

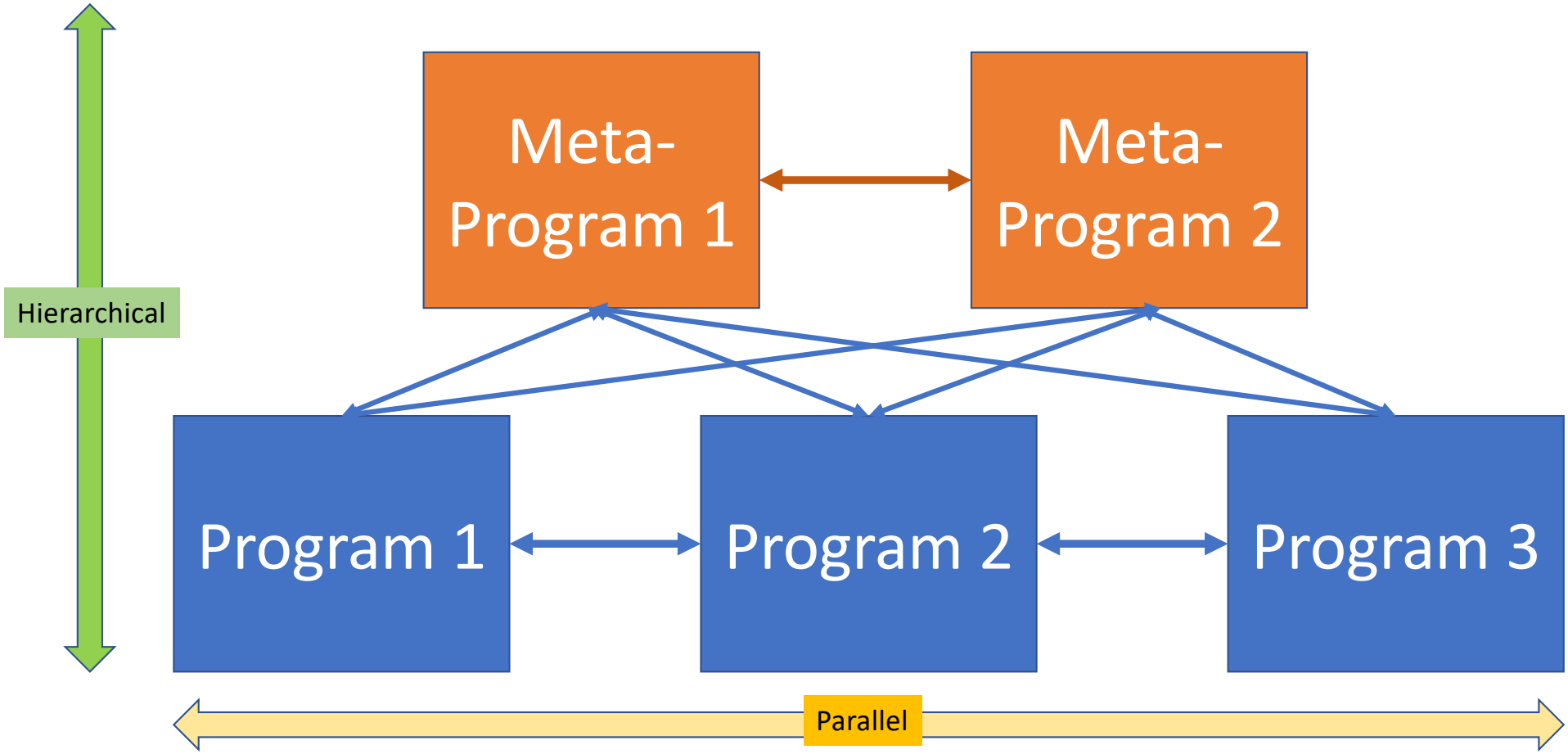
- Flexible input-output
- Able to pattern match semantics and serve as an interface for tools
- Can retrieve information in specified formats well
- Customisable to new tasks easily with zero-shot or few-shot prompting

Drawbacks

- May be too flexible (output may not be of the right type for use case)
- Not consistent
- Not able to follow rules as well as rule-based systems
- Costly and takes a while to generate tokens

A good system will need to have both **LLM's flexibility** and **rule-based solution's consistency**!

Systems-level approach to LLMs

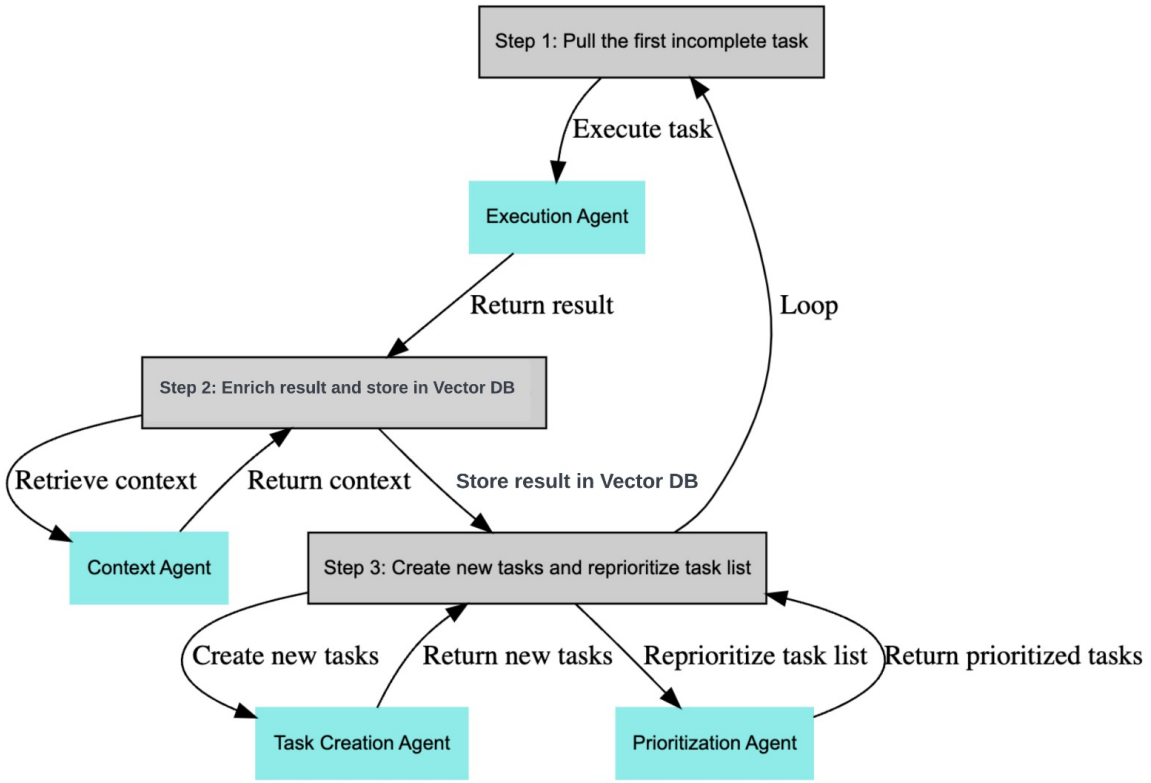


Breaking down the problem

- A difficult problem can be **broken down into sub-parts**
- Each sub-part may be solved more easily
- Can have the solver for each sub-part be an LLM and/or a Rule-based System
  - It could use LLM and/or rule-based approaches to solve
  - It could use rule-based approaches to check if solution is correct and iteratively prompt the solver again with relevant error messages

BabyAGI – Breaking problem into sub-parts

- Uses GPT4 to in a larger ecosystem to:
  - Create Tasks
  - Execute Tasks
  - Prioritize Tasks
- Uses memory to store and retrieve task/result pairs
  - Stored by vector embeddings



https://github.com/yoheinakajima/babyagi

A Community of Agents

Have a group of agents with **different initial contexts** interact with each other to achieve a goal

- Generative Agents Simulacra (The Sims-like simulation)
- ChatDev (Software Company Simulation)

Each agent can help to **bring in particular expertise** to solve a problem

- Each agent can even have their own unique memories and own unique tools

Collective intelligence is greater than the individual

- When there are some tasks which an agent cannot solve, it can be offloaded to others
- Can pass the problems up and down the hierarchy until a suitable agent can solve it

Challenges

Challenges of Implementing Agents (Part 1/2)

- Agents may have **finite context length**
  - Use fine-tuning for tool usage to avoid specifying?
  - Split prompts up into smaller chunks with more agents

- Problems in **long-term planning**
  - LLMs can pattern match in a suitable abstraction space, but may not be able to plan long-term
  - Maybe plan at different abstraction spaces to reduce planning steps? (Hierarchical Navigable Small Worlds)
  - Perhaps use parallel memory retrieval paths to plan (Learning, Fast & Slow)

Challenges of Implementing Agents (Part 2/2)

- Agents may **keep calling tools** when not required
  - Have a tool that does nothing?

- **Formatting errors**
  - Asking it to output code will often result in compile error for complicated codes
  - Have a new consistent format for code?

Questions to Ponder

- How can other modalities be incorporated into an agent?
- How can we better use memory apart from just Retrieval Augmented Generation?
- How are the initial tools / skills learned, or are they provided right at the beginning?
- How do we determine when to generate new tools / skills?
- When should rule-based systems be used, and when should LLMs be used?