

Agentic architectures and workflows

 Keheliya Gallaba

 keheliya.github.io

How to cite this session?

```
@misc{Gallaba2024AgentsTutorial,  
author = {Keheliya Gallaba and Dayi Lin and Ahmed E. Hassan},  
title = {Agentic architectures and workflows},  
howpublished = {Tutorial presented at the AIware Leadership Bootcamp 2024},  
month = {November},  
year = {2024},  
address = {Toronto, Canada},  
note = {Part of the AIware Leadership Bootcamp series.},  
url = {https://aiwarebootcamp.io/slides/2024\_aiwarebootcamp\_gallaba\_keheliya\_agents.pdf } }
```



Check this paper for more information about this session

```
@article{gallaba2024tom,  
  title={On the Role of Theory of Mind towards Intent-first Development Environments},  
  author={Gallaba, Keheliya and Arabat, Ali and Lin, Dayi and Sayagh, Mohammed and Hassan, Ahmed E},  
  journal={arXiv},  
  year={2024}  
}
```



Overview of the session

- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



What is an agent?

- **Weak notion:** Agent is an entity that can **perceive** and **act** upon their **environment**.
- But then, is thermostat an agent?
- **Strong notion:** Entities that possess desires, beliefs, intentions, and the ability to act.
- Key properties:

Autonomy

Agents operate without the direct intervention of humans or others, and have some control over their actions and internal state.

Social Ability

Agents interact with other agents (and possibly humans) via some kind of *agent-communication language*.

Reactivity

Agents perceive their environment, and respond in a timely fashion to changes that occur in it.

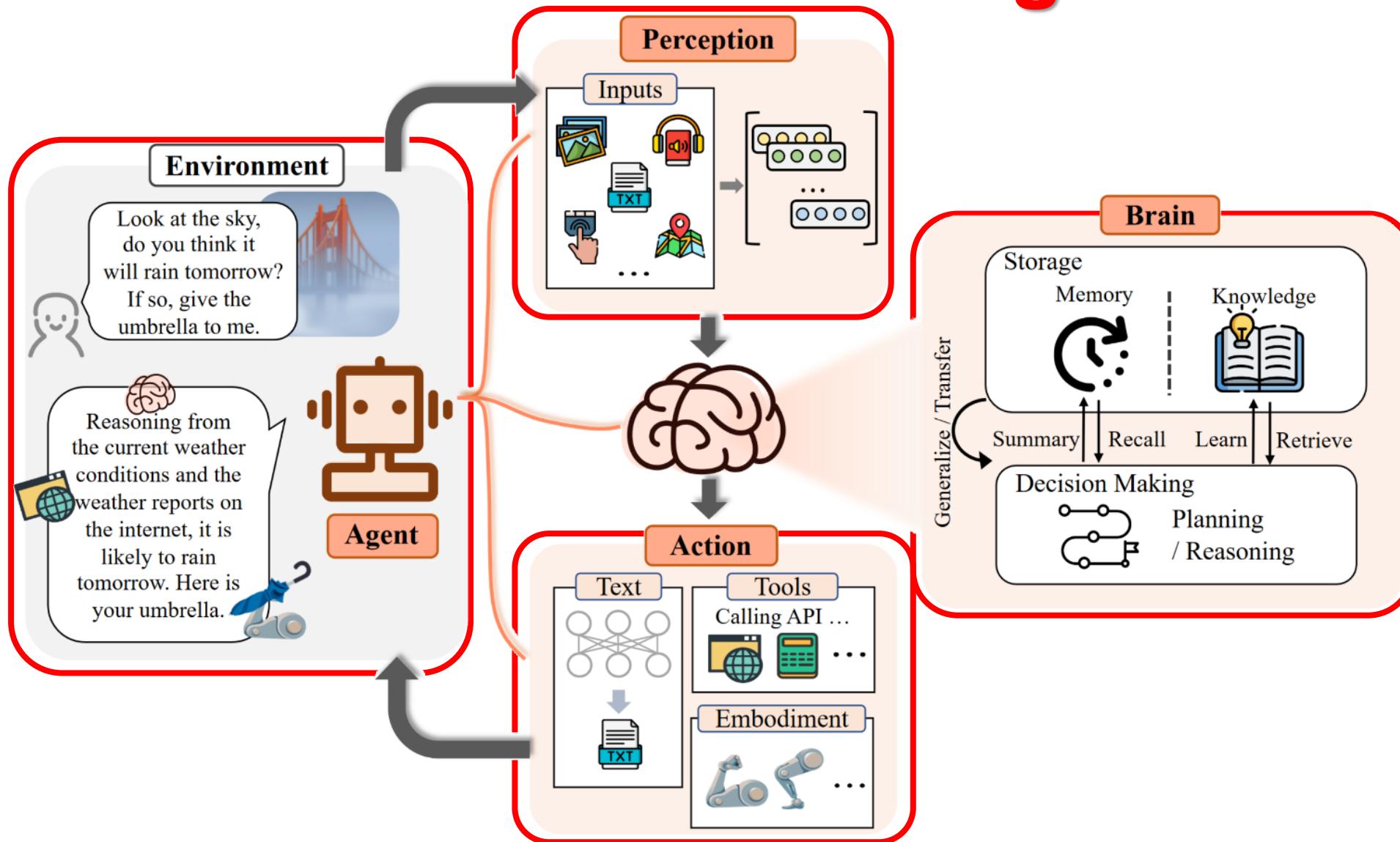
Pro-activeness

Agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative.

"Agentic" – It's not binary, but a spectrum.



Foundation-model-based agents



Foundation-model-based agents

Conversational Agents

Question and Answering

Software Developer Agents

Scientific Discovery Agents

???

Commonsense Q&A

Knowledge-intensive Q&A

Multi-hop Q&A

Symbolic reasoning

Mathematical reasoning



OpenHands

AI Scientist

Chem Crow



Checking “**How many r's are in the word 'strawberry'?**” will not be sufficient.

Overview of the session

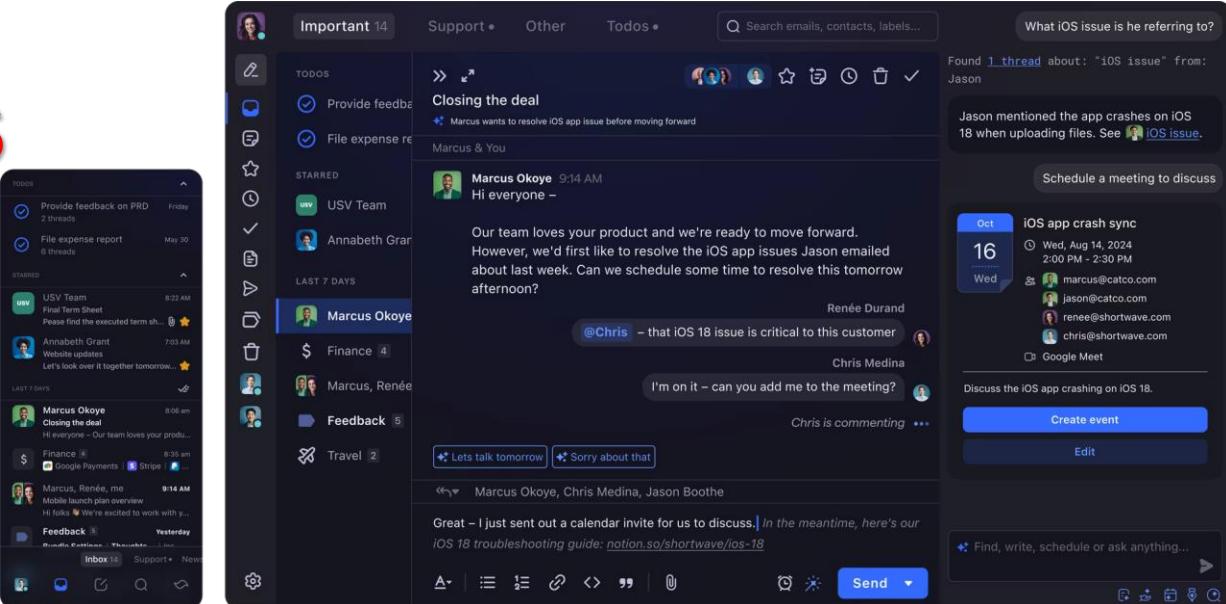
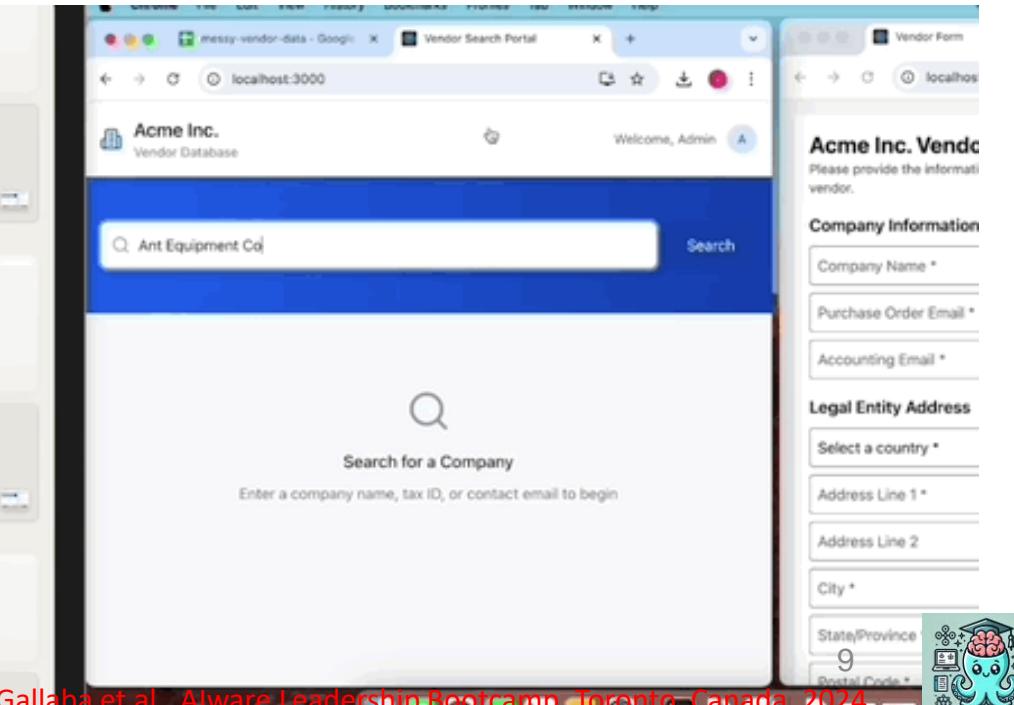
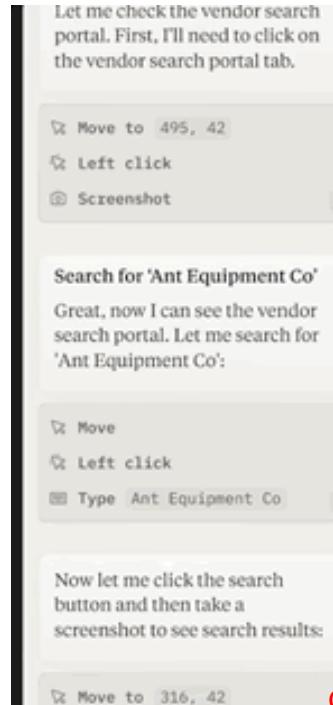
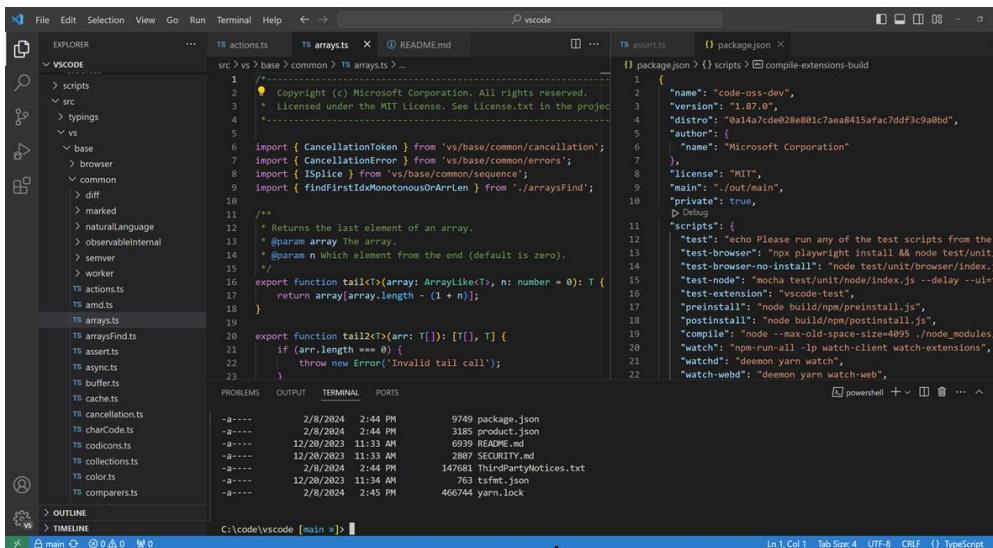
- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



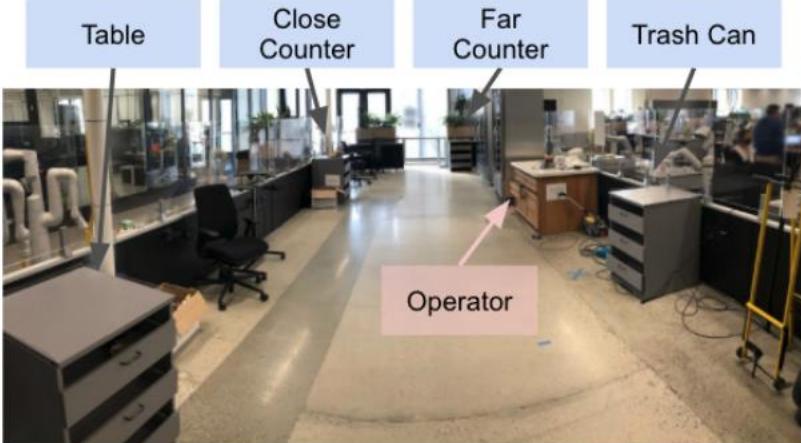
Agents operate in

Digital Environments

- **Source Repositories:** Github, Gitlab
- **Games:** Atari DQN, AlphaGo
- **IDEs:** VSCode, Jetbrains
- **Task Management Software:** Jira, Trello
- **Office Software:** Google Docs, Microsoft Office
- **Communication Tools:** Gmail, Slack, Shortwave
- **Or the whole desktop!** e.g., Anthropic's computer use

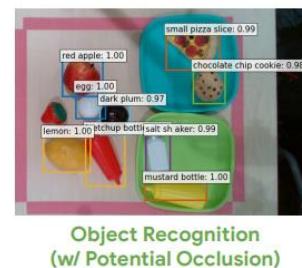


Agents operate in Physical Environments



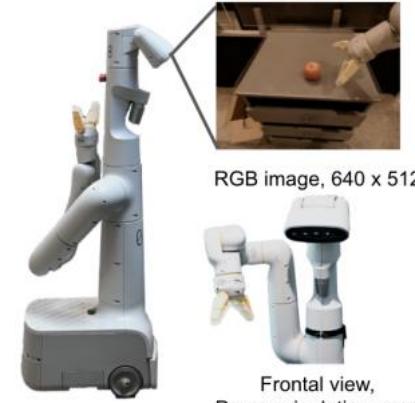
Tabletop Rearrangement (Real)

Human: Sort fruits and condiments in different plates
 Scene: objects = [apple, egg, plum, pizza, cookie, salt, mustard, ketchup, lemon]
 Scene: occluded objects = []
 Robot: robot.pickplace(strawberry, green plate)
 Robot: robot.pickplace(salt shaker, blue plate)
 Success Detector: False.
 Robot: robot.pickplace(salt shaker, blue plate)
 ...
 Robot: robot.stop()



Kitchen Mobile Manipulation (Real)

Human: Bring me a drink from the table
 Robot: go to the table
 Scene: I see a coke and lime soda
 Robot: What kind of drink would you like?
 Human: Something with caffeine
 Robot: Pick up coke
 Success Detector: False
 Robot: Pick up coke
 Robot: Bring it to you
 Robot: Put down the coke
 Robot: Done



RGB image, 640 x 512

Frontal view,
Pre-manipulation pose



Action: put the egg on the blue plate

Initial

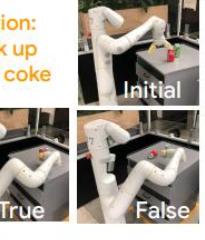
True

False

Success Detection

Affordance Functions

Affordance Grounding



Action: pick up the coke

Initial

True

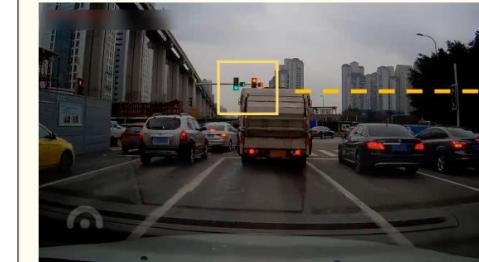
False

Success Detection

Sec. 2.1 Traffic light understanding

Prompt:

Describe the condition of all the traffic lights in the image, if there are countdowns, please describe them together.



GPT-4V:

The traffic lights in the center show a green light and a red arrow with a countdown displaying "4". On the left and right, the traffic lights are showing red.

Prompt:

Describe the condition of all the traffic lights in the image, if there are countdowns, please describe them together.



GPT-4V:

The traffic light is showing a green light with a countdown displaying "4", and a red arrow with a countdown displaying "2".

Sec. 3.1 Corner cases

Prompt:

You're the mature driver behind the wheel and this image is what you see now. Describe what you see and what you plan to drive next.

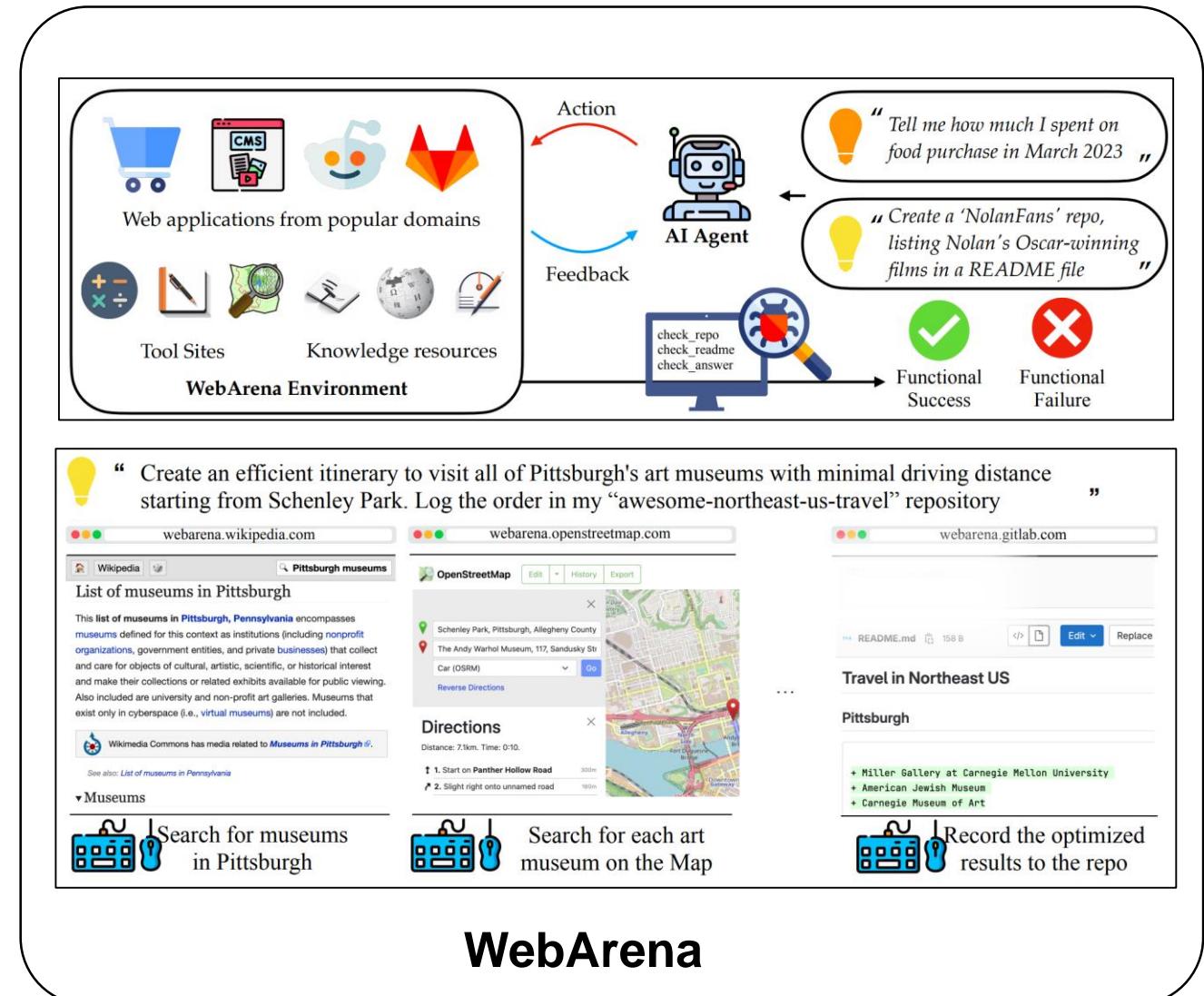
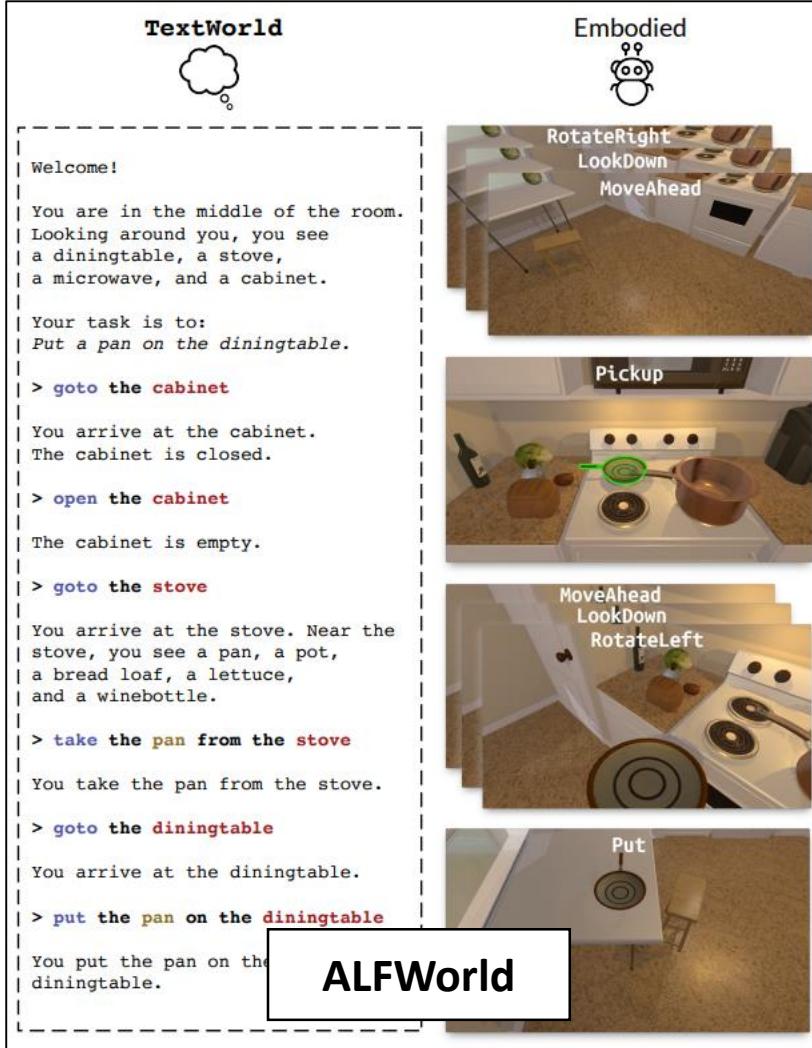


Prompt:

You're the mature driver behind the wheel and this image is what you see now. Describe what you see and what you plan to drive next.



Agents operate in Simulated Environments



Overview of the session

- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



Different types of agentic memory

Procedural memory
stores the production system itself.

Semantic memory
stores facts about
the world.

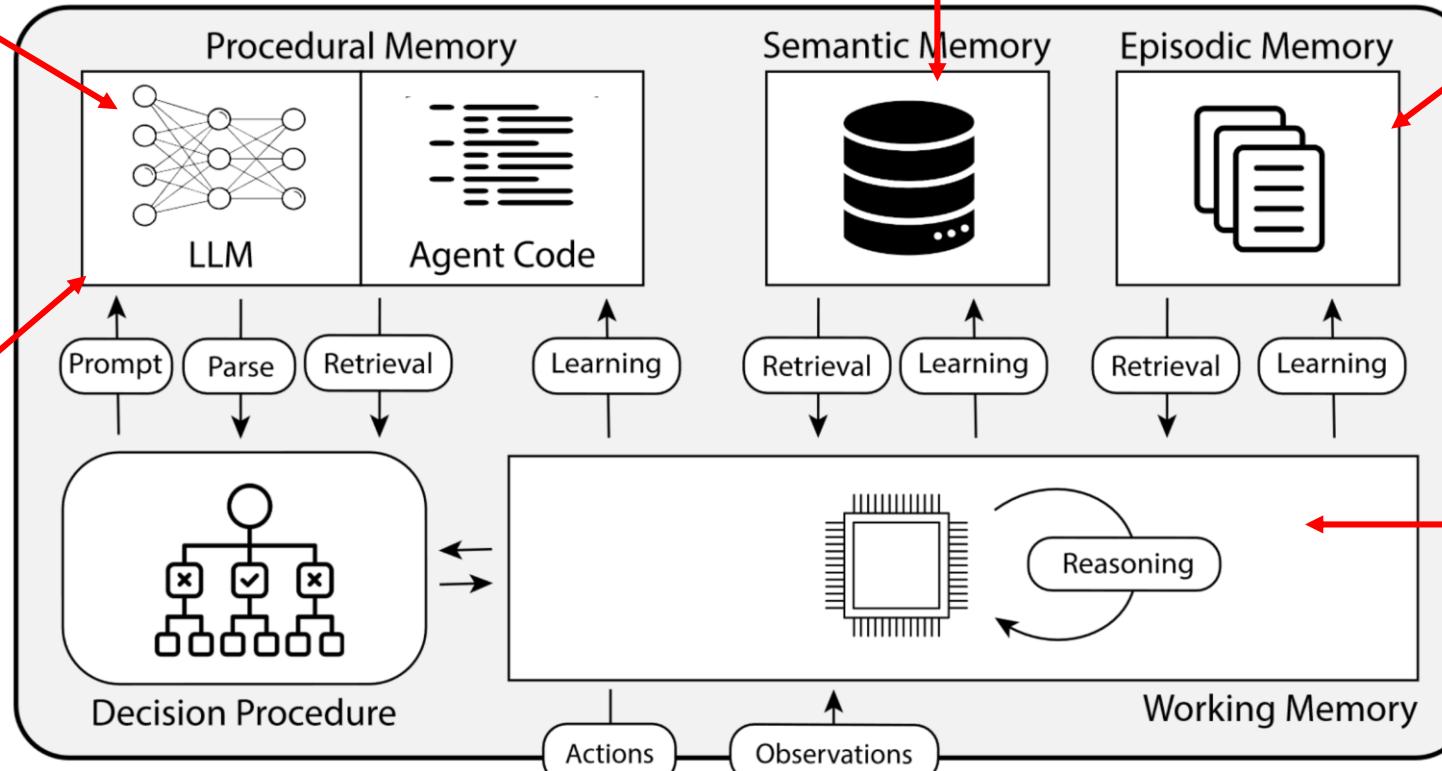
Episodic memory
stores sequences of the
agent's past behaviors.

Three types of Long-Term
memory

- Procedural
- Semantic
- Episodic

Working memory
reflects the agent's
current circumstances:

- Recent perceptual input
- Active goals
- Results from intermediate,
internal reasoning.



Procedural Memory

Stores the production system itself.

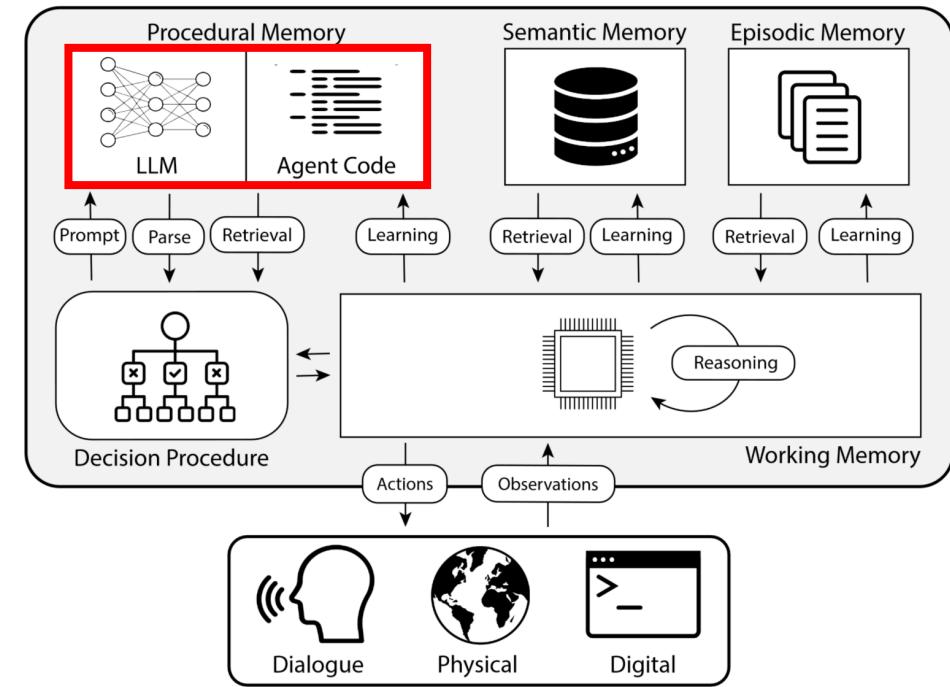
Must be initialized by the designer with proper code to bootstrap the agent.

LLM Parameters/Weights

- A large, stochastic production system
- Hard to interpret
- Zero-shot flexibility in new contexts
- Implicit knowledge

Agent Code

- Deterministic rules
- Explicit knowledge
- Interpretable and extensible
- Limited to address situations the designer anticipates
- Brittle in face of stochasticity
- Procedures for implementing:
 - Actions (for doing reasoning, retrieval, grounding, and learning)
 - Decision Making



During agent design think about limitations of **code** vs **LLMs**.



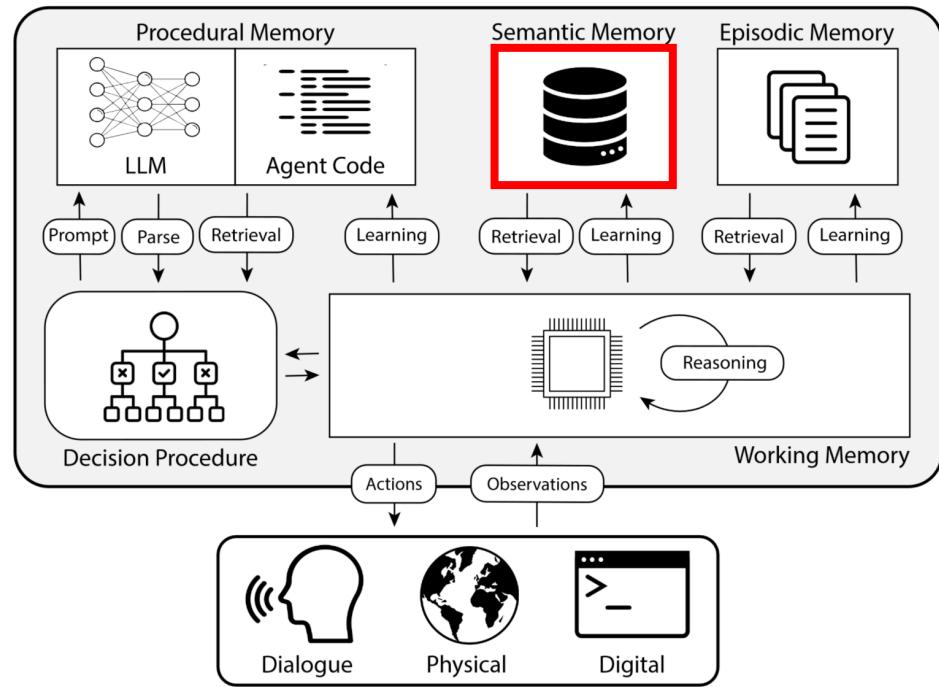
E.g., implementing **tree search** in code to mitigate myopia induced by autoregressive generation in **Tree-of-Thought**.



Types of agentic memory

Semantic Memory

- Stores an agent's knowledge about the world and itself to be used for **reasoning** or **decision-making**.
- Using an external database for knowledge support.
- Retrieving from semantic memory:
 - Vector databases in Retrieval Augmented Generation (**RAG**) used to retrieve unstructured text.
 - Leveraging game manuals and facts to affect policy in **RL**.
- Updating semantic memory:
 - **NLMap**: Use vision-language models (VLMs) to build a semantic map of the environment. Query it later when executing instructions.
 - **Generative Agents**: Reason about raw experiences and store the resulting inferences.
 - **Reflexion**: Reflect on failed episodes and store the results as knowledge to be attached as context when solving later episodes.



During agent design, think about **immutability**.



E.g., employing a fixed, read-only semantic memory Vs. allowing writing new knowledge obtained from LLM reasoning into semantic memory (to incrementally build up world knowledge from experience.)



Types of agentic memory

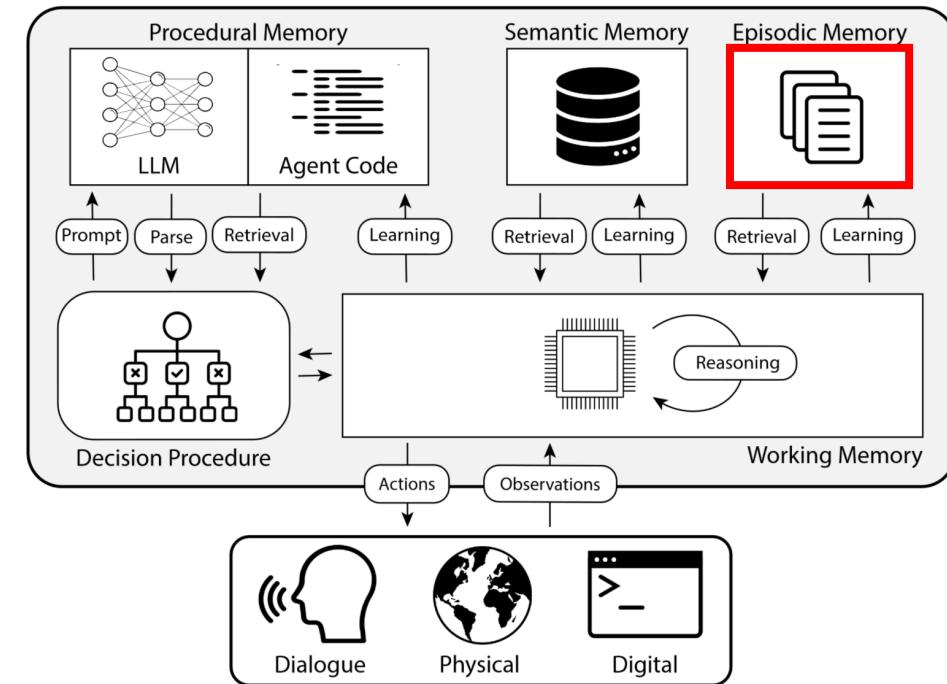
Episodic Memory

Stores sequences (raw experiences) of the agent's past behaviors or earlier decision cycles.

- Past event flows
- Previous game trajectories
- Training input-output pairs

During **planning**, these episodes may be retrieved into **working memory** to support **reasoning**.

Alternatively, periodical fine-tuning the LM on high-scoring trajectories (i.e., from episodic memory to **procedural memory**).



During agent design think about **querying** episodic memories. You may need a combination of approaches:

- **Recency** (rule-based)
- **Importance** (reasoning-based)
- **Relevance** (embedding-based)



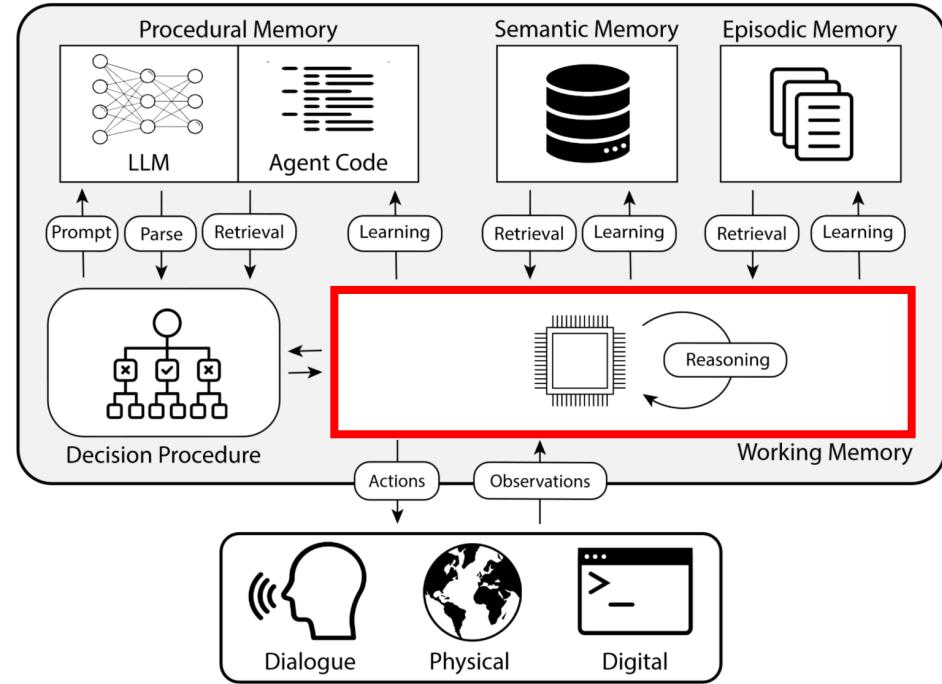
Working Memory

Maintaining active and readily available information for the current decision cycle

A central hub connecting different components and managing agent's current circumstances)

Can be implemented by

- using LLM's own context
 - Chain-of-thought
 - Scratchpads
- a data structure persisting across LLM calls



During agent design think about

- what to track in working memory vs long-term memory
- a structured reasoning procedure to update fields in working memory.

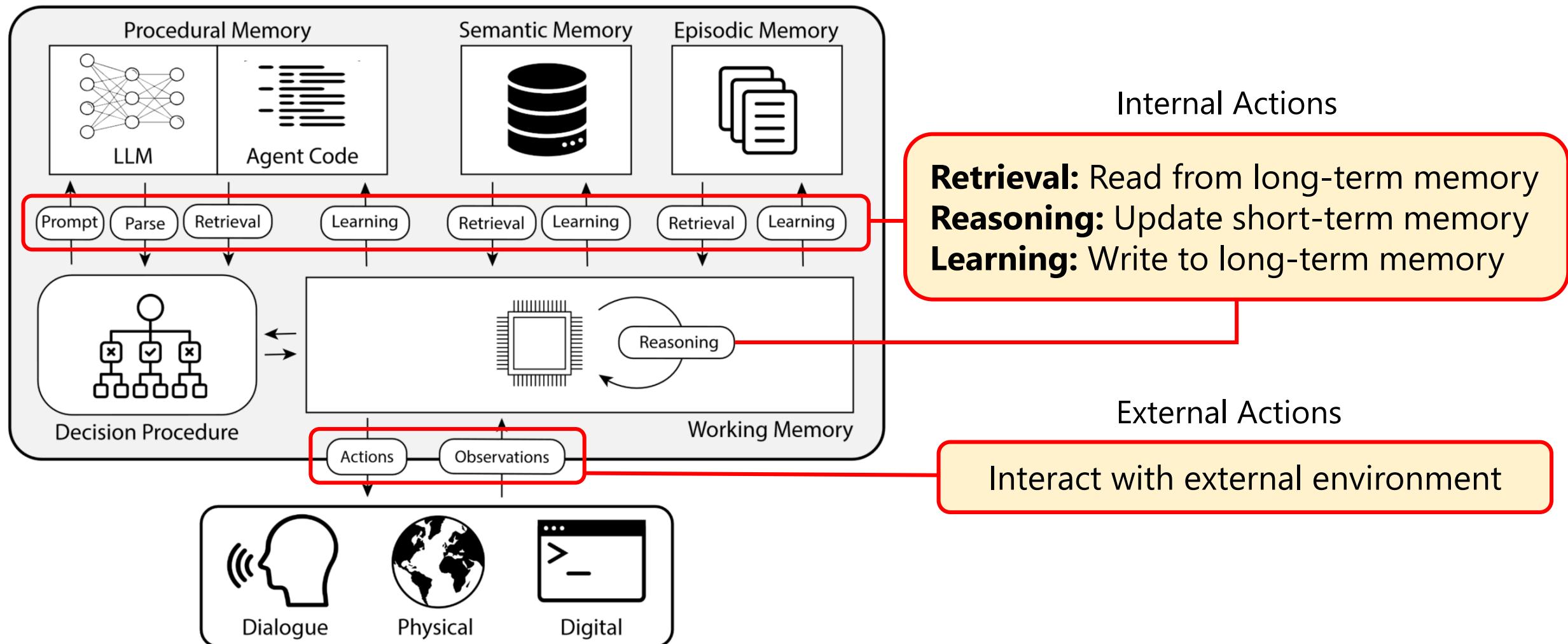


Overview of the session

- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



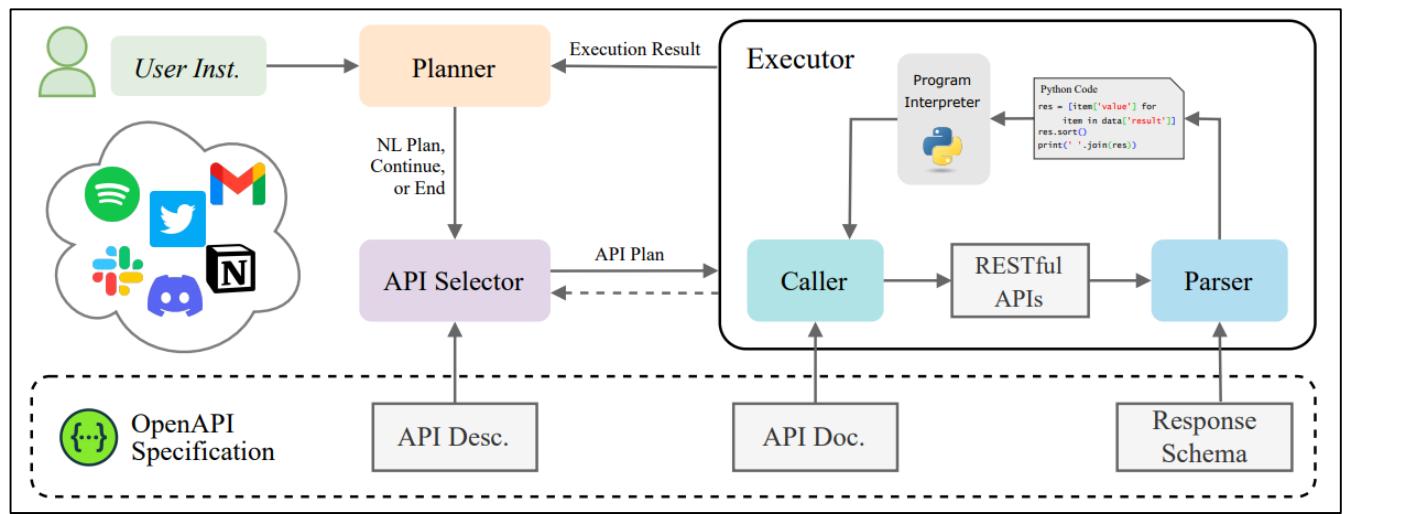
Action space of agents



Action space of agents

External Tools

Service APIs



Knowledge bases



External Models



Symbolic Modules

Python Interpreter

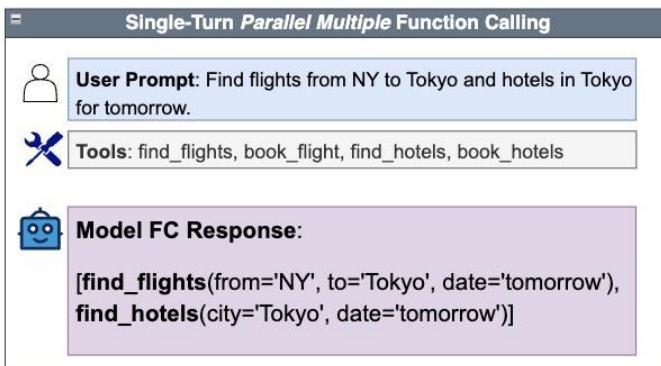
Physics Engine

Constraint Solver

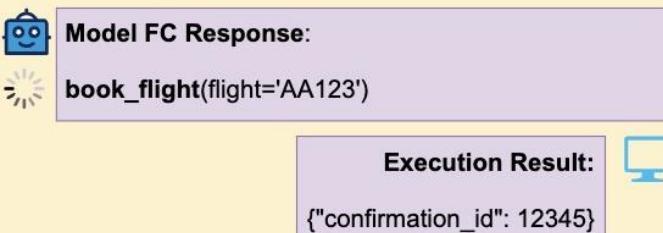


Interaction Patterns with External Tools

Single Turn

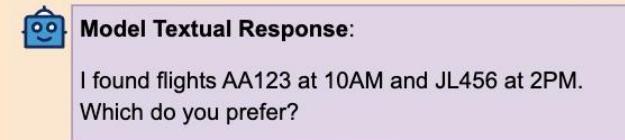


Multi-Step

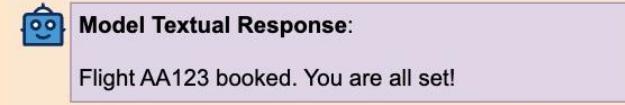
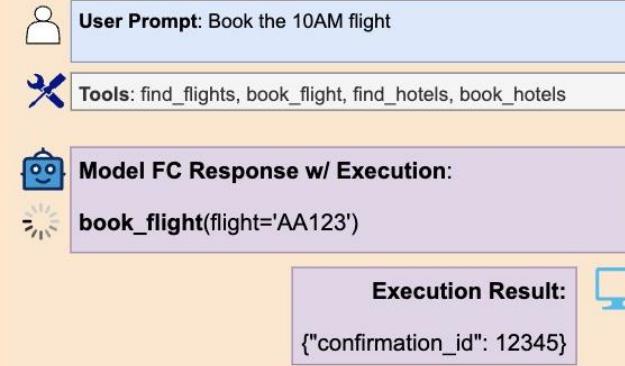


Multi-Turn

Turn #1



Turn #2



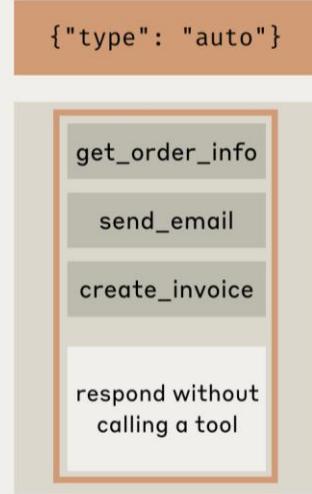
To achieve one **goal**, the agent may need to call **multiple tools** while addressing **multiple rounds** of user requests.



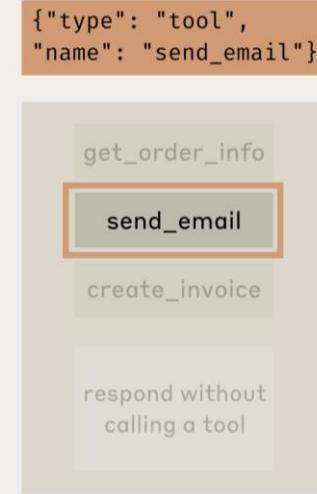
Function/Tool calling in LLMs

System Prompt

"In this environment you have access to a set of tools you can use to answer the user's question."



Tool Choice



A diagram showing a list of tool calls. Each call is enclosed in a blue box and has a brown border. The list consists of two entries:

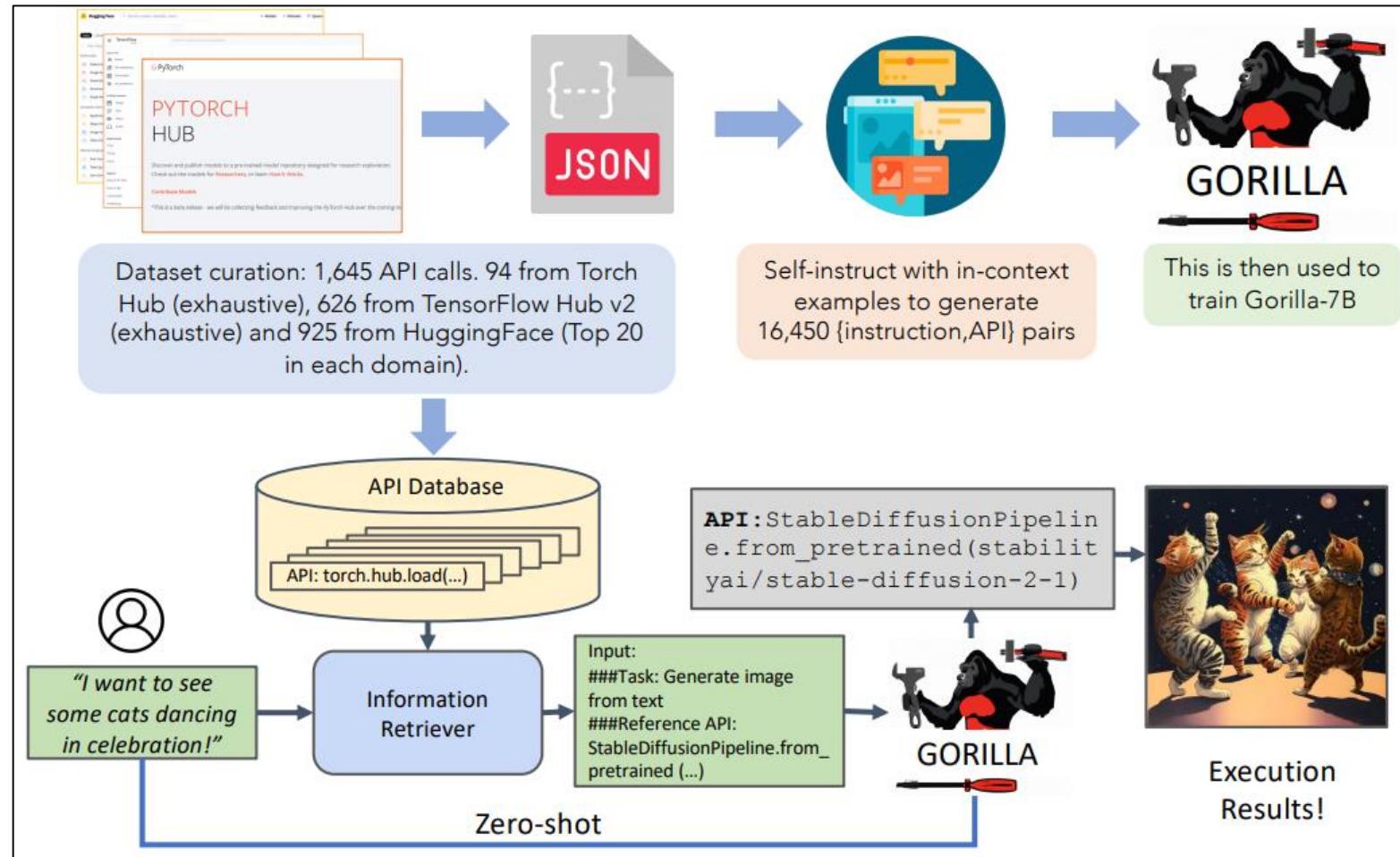
```
[  
  {  
    "text": "<thinking>\nI should use a tool.\n</thinking>",  
    "type": "text"  
  },  
  {  
    "id": "id_value",  
    "input": {"arg_name": "arg_value"},  
    "name": "tool_name",  
    "type": "tool_use"  
  }]  
]
```

Tool Call



Interacting with the environment

Gorilla



Key Idea: Fine-tune LLM to generate correct input arguments for API calls mitigating hallucination during external API invocations.



Interacting with the environment CodeAct

Instruction: Determine the most cost-effective country to purchase the smartphone model "CodeAct 1". The countries to consider are the USA, Japan, Germany, and India.

Available APIs

```
[1] lookup_rates(country: str) -> (float, float)
[2] convert_and_tax(price: float, exchange_rate: float, tax_rate: float) -> float
[3] estimate_final_price(converted_price: float, shipping_cost: float) -> float
[4] lookup_phone_price(model: str, country: str) -> float
[5] estimate_shipping_cost(destination_country: str) -> float
```

LLM Agent using [Text/JSON] as Action

 Think I should calculate the phone price in USD for each country, then find the most cost-effective country.

 Action Text: `lookup_rates, Germany`
`</>` JSON: `{"tool": "lookup_rates", "country": "Germany"}`

 Environment 1.1, 0.19

 Action Text: `lookup_phone_price, CodeAct 1, Germany`
`</>` JSON: `{"tool": "lookup_phone_price", "model": "CodeAct 1", "country": "Germany"}`

 Environment 700

 Action Text: `convert_and_tax, 700, 1.1, 0.19`
`</>` JSON: `{"tool": "convert_and_tax", "price": 700, "exchange_rate": 1.1, "tax_rate": 0.19}`

 Environment 916.3

[... interactions omitted (look up shipping cost and calculate final price) ...]

 Action Text: `lookup_rates, Japan`
`</>` JSON: `{"tool": "lookup_rates", "country": "Japan"}`

[... interactions omitted (calculate final price for all other countries)...]

 Response The most cost-effective country to purchase the smartphone model is Japan with price 904.00 in USD.

CodeAct: LLM Agent using [Code] as Action

 Think I should calculate the phone price in USD for each country, then find the most cost-effective country.

 Action

```
countries = ['USA', 'Japan', 'Germany', 'India']
final_prices = {}

for country in countries:
    exchange_rate, tax_rate = lookup_rates(country)
    local_price = lookup_phone_price("xAct 1", country)
    converted_price = convert_and_tax(
        local_price, exchange_rate, tax_rate
    )
    shipping_cost = estimate_shipping_cost(country)
    final_price = estimate_final_price(converted_price, shipping_cost)
    final_prices[country] = final_price

most_cost_effective_country = min(final_prices, key=final_prices.get)
most_cost_effective_price = final_prices[most_cost_effective_country]
print(most_cost_effective_country, most_cost_effective_price)
```

 Environment 1.1, 0.19

 Response The most cost-effective country to purchase the smartphone model is Japan with price 904.00 in USD.

Control & Data Flow of Code Simplifies Complex Operations

Re-use `min` Function from Existing Software Infrastructures (Python library)

Using **code** to interact with the environment rather than direct tool use could be more **effective** and **efficient**.



Overview of the session

- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



Reasoning: Which One?

- **Commonsense Reasoning:** Capacity to infer and apply everyday, intuitive knowledge.
- **Mathematical Reasoning:** Ability to solve mathematical problems and derive logical conclusions.
- **Logical Reasoning:** Process of drawing inferences and making decisions based on formal logic.
- **Causal Reasoning:** Understanding of cause-and-effect relationships and their implications.
- **Multimodal Reasoning:** Reasoning across multiple data modalities, such as text, images, and sensory information.
- **Visual Reasoning:** Focusing on tasks that require the interpretation and manipulation of visual data.
- **Embodied Reasoning:** Reasoning in the context of embodied agents interacting with their environment.
- **Other Reasoning Tasks**



Reasoning with foundation models

Decomposing a potentially complex task into simpler subtasks the FM can solve more easily by itself or using tools.

Giving more computation steps to the model before yielding the answer to a prompt.

Generating intermediate steps improves LLM performance

Training with intermediate steps

Finetuning with intermediate steps

Prompting with intermediate steps

Problem 1:

Question: Two trains running in opposite directions cross a man standing on the platform in 27 seconds and 17 seconds respectively and they cross each other in 23 seconds. The ratio of their speeds is:

Options: A) 3/7 B) 3/2 C) 3/88 D) 3/8 E) 2/2

Rationale: Let the speeds of the two trains be x m/sec and y m/sec respectively. Then, length of the first train = $27x$ meters, and length of the second train = $17y$ meters. $(27x + 17y) / (x + y) = 23 \rightarrow 27x + 17y = 23x + 23y \rightarrow 4x = 6y \rightarrow x/y = 3/2$

Correct Option: B

Program Induction by rationale generation

“Think Step-by-step”

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?

Solution: Beth bakes 4 2 dozen batches of cookies for a total of $4 * 2 = 8$ dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of $12 * 8 = 96$ cookies
She splits the 96 cookies equally amongst 16 people so they each eat $96/16 = 6$ cookies

Final Answer: 6

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs \$3.50?
Mrs. Lim got 68 gallons - 18 gallons = $68 - 18 = 50$ gallons this morning.
So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = $68 + 82 + 50 = 200 > 200$ gallons.
She was able to sell 200 gallons - 24 gallons = $200 - 24 = 176$ gallons.
Thus, her total revenue for the milk is $\$3.50/gallon \times 176$ gallons = $\$3.50 \times 176 = \$616 > 616$.
Final Answer: 616

Problem: Tina buys 3 12-packs of soda for a party. Including Tina, 6 people are at the party. Half of the people at the party have 3 sodas each, 2 of the people have 4, and 1 person has 5. How many sodas are left over when the party is over?
Solution: Tina buys 3 12-packs of soda, for $3 * 12 = 36$ sodas
6 people attend the party, so half of them is $6/2 = 3$ people
Each of those people drinks 3 sodas, so they drink $3 * 3 = 9$ sodas
Two people drink 4 sodas, which means they drink $2 * 4 = 8$ sodas
With one person drinking 5, that brings the total drank to $9 + 8 + 5 = 22$ sodas
As Tina started off with 36 sodas, that means there are $36 - 22 = 14$ sodas left
Final Answer: 14

GSM8K

Chen et al., Premise Order Matters in Reasoning with Large Language Models. ICML 2024

Huang et al., Large Language Models Cannot Self-Correct Reasoning Yet. ICLR 2024.

Shi et al., Large Language Models Can Be Easily Distracted by Irrelevant Context. ICML 2023.

Wang et al., Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models Ling et al. Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems. ACL 2017

Cobbe et al. Training Verifiers to Solve Math Word Problems. Nye et al. Show Your Work: Scratchpads for Intermediate Computation with Language Models 2021

Self-consistency greatly improves step-by-step reasoning

Limitations:

- irrelevant context is distracting
- self-correction is bad
- premise order matters

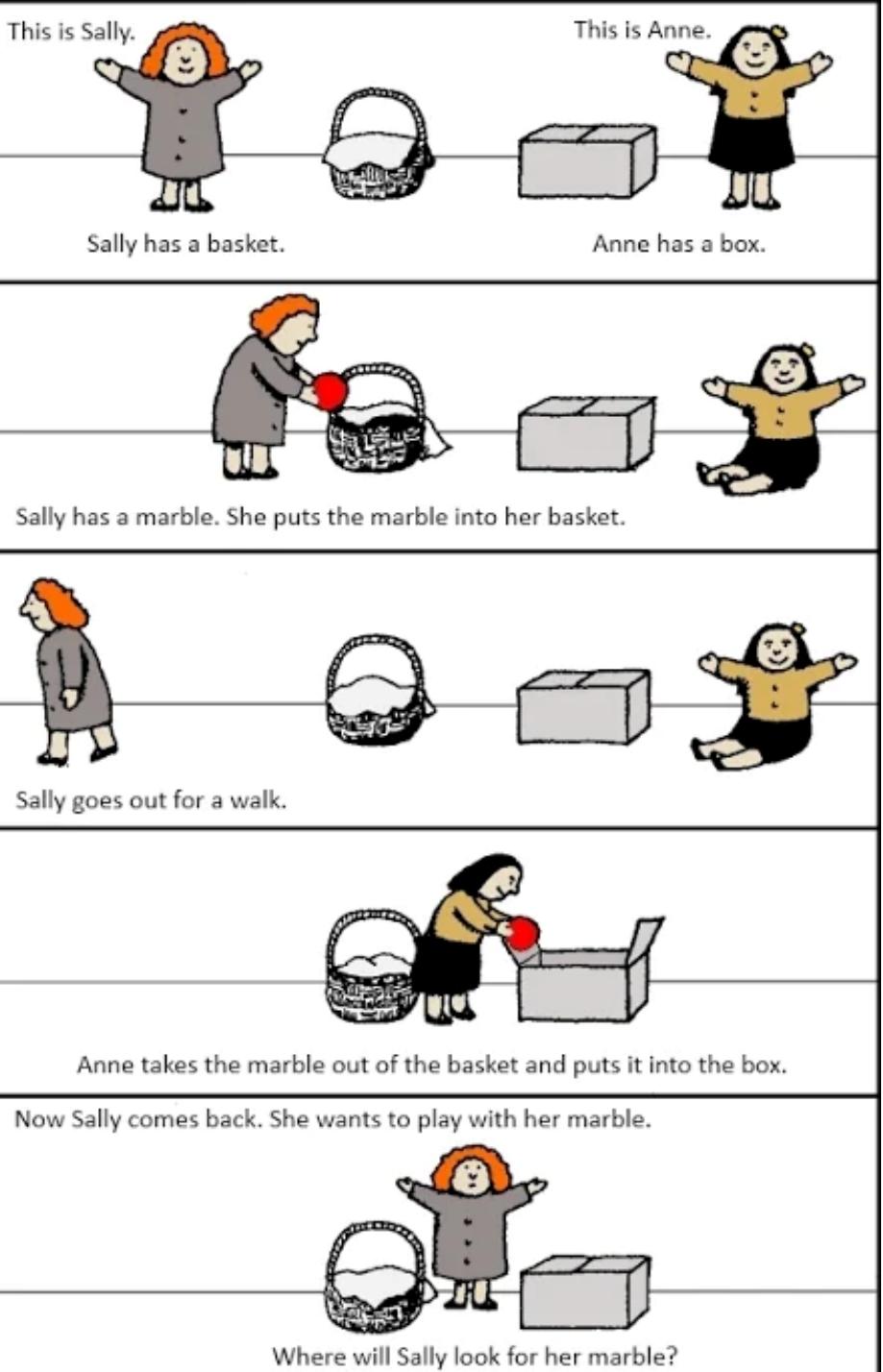
Input:
2 9 + 5 7

Target:

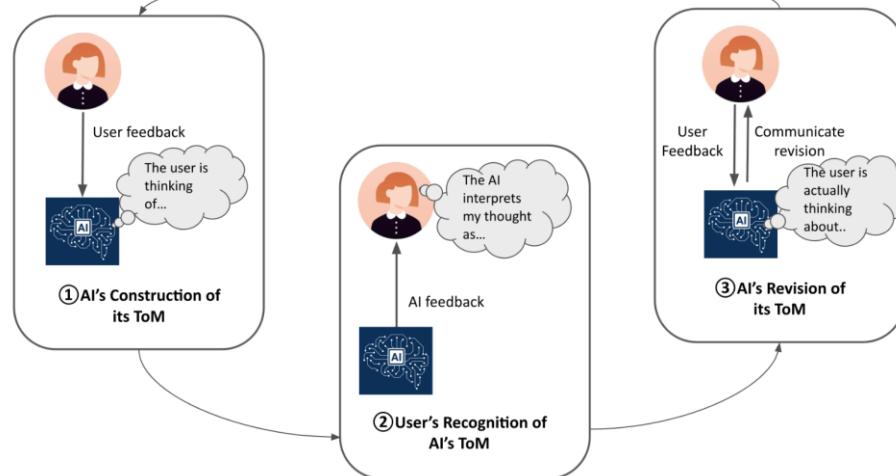
```
<scratch>
2 9 + 5 7 , C: 0
2 + 5 , 6 C: 1 # added 9 + 7 = 6 carry 1
, 8 6 C: 0 # added 2 + 5 + 1 = 8 carry 0
0 8 6
</scratch>
8 6
```

Scratchpads for intermediate computation





Theory of Mind



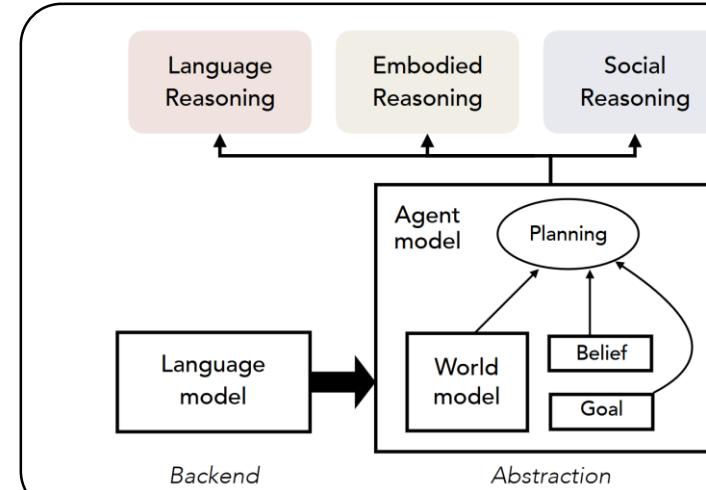
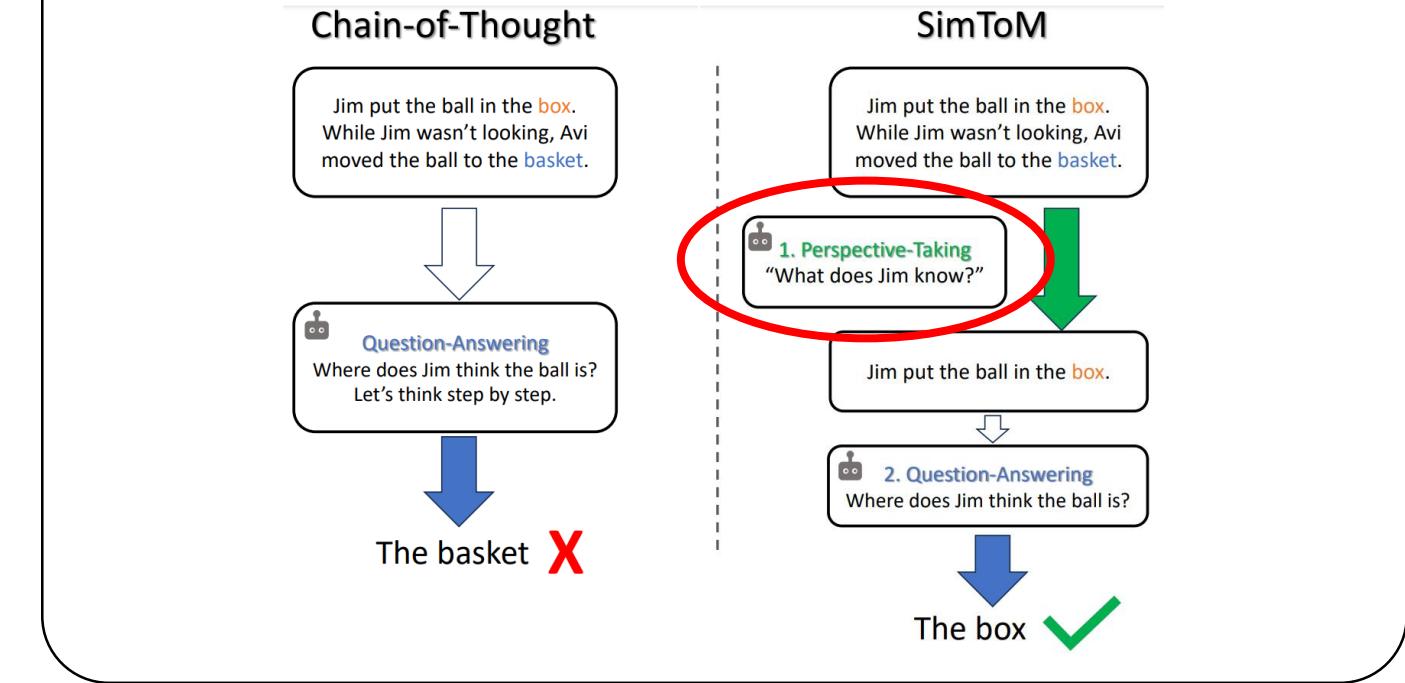
Humans and AI can each **construct and revise** their interpretations of each other based on feedback from the other party.

Wang and Goel, Mutual Theory of Mind for Human-AI Communication

Wilf et al., Think Twice: Perspective-Taking Improves Large Language Models' Theory-of-Mind Capabilities

Hu and Shu, Language Models, Agent Models, and World Models: The LAW for Machine Reasoning and Planning

Perspective-Taking aka “stepping into the other person’s shoes” improves LLMs' Theory-of-Mind Capabilities.



“Beliefs” : An agent needs to infer others' intentions and their potential reactions to decide the most appropriate things to say.



Theory of Mind

Cheryl's Birthday

Problem Statement

Cheryl gives Albert and Bernard a list of 10 possible birthdates, and tells them different information:

- Albert knows the month.
- Bernard knows the day.

Cheryl provides the following 10 possible dates:

- May 15, May 16, May 19
- June 17, June 18
- July 14, July 16
- August 14, August 15, August 17

Albert then says: "I don't know when Cheryl's birthday is, but I know that Bernard doesn't know either."

Bernard replies: "At first I didn't know when Cheryl's birthday is, but now I know."

Albert then says: "Now I know when Cheryl's birthday is."

- Write a program to solve the problem.
- Solve a re-worded variant of the problem with different dates.

10 Solvers Used:

- [A human programmer](#)
- [ChatGPT 4o](#)
- [Microsoft Copilot](#)
- [Gemini Advanced](#)
- [Meta AI Llama 405B](#)
- [Anthropic Claude 3.5 Sonnet](#)
- [Perplexity](#)
- [Cohere Chat](#)
- [HuggingFace Chat](#)
- [You.com](#)

1. The human solved both requests.
2. None of the LLMs could reliably solve either request.

o1 mini gets it right on the first try. ☺



Overview of the session

- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



Cognitive Architectures

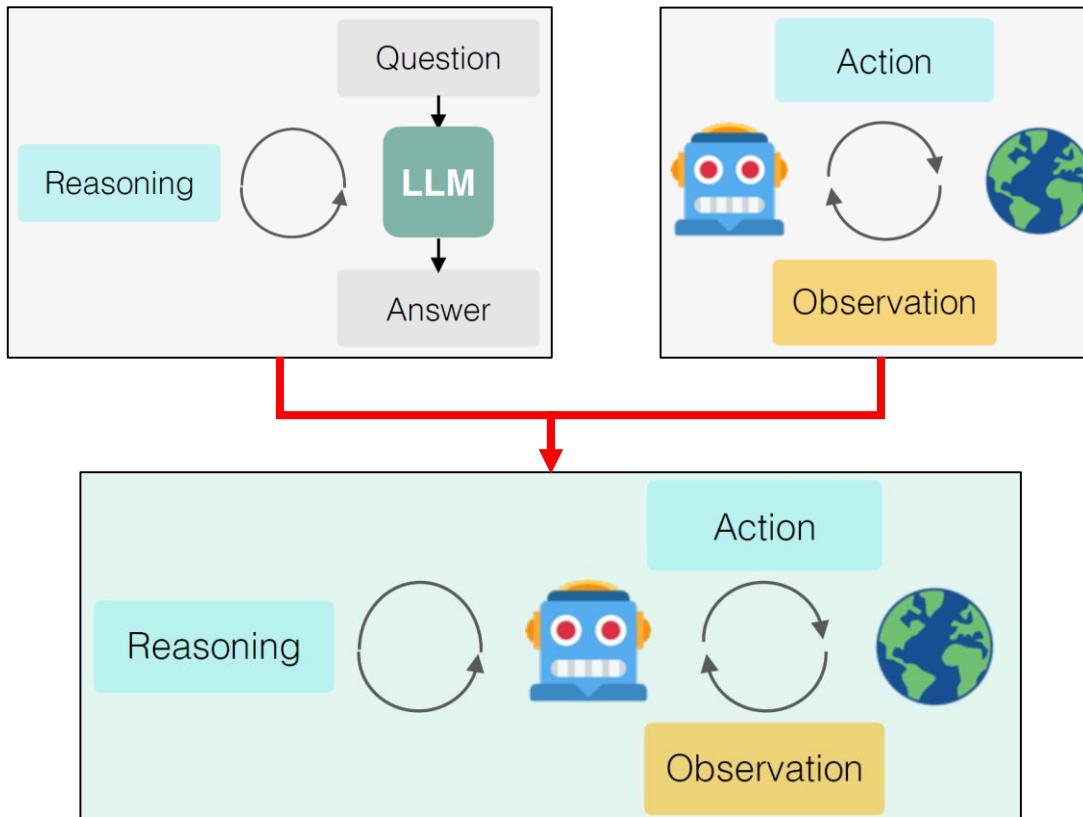
ReAct

Reasoning

(updating working memory)

Acting

(obtaining external feedback)



(1d) ReAct (Reason + Act)

Thought 1: I need to search Cirque du Soleil show Mystere, find the hotel it is in, then find the number of rooms in the hotel.

Act 1: `Search[Cirque du Soleil show Mystere]`

Obs 1: Could not find Cirque du Soleil show Mystere.
Similar: ['Cirque du Soleil' ...]

Thought 2: To find the hotel, I can search Mystere (Cirque du Soleil).

Act 2: `Search[Mystere (Cirque du Soleil)]`

Obs 2: Mystère (my-steer) is ... Held in a custom theatre at the Treasure Island Hotel and Casino ...

Thought 3: Mystere is held in a custom theatre at the Treasure Island Hotel and Casino. So I need to search Treasure Island Hotel and Casino next and find the number of rooms in the hotel.

Act 3: `Search[Treasure Island Hotel and Casino]`

Obs 3: Treasure Island Hotel and Casino ... with 2,884 rooms and 220 suites ...

Thought 4: Treasure Island Hotel and Casino has 2,884 rooms and 220 suites. So the answer is 3,104.

Act 4: `Finish[3,104]`

Up-to-date ✓

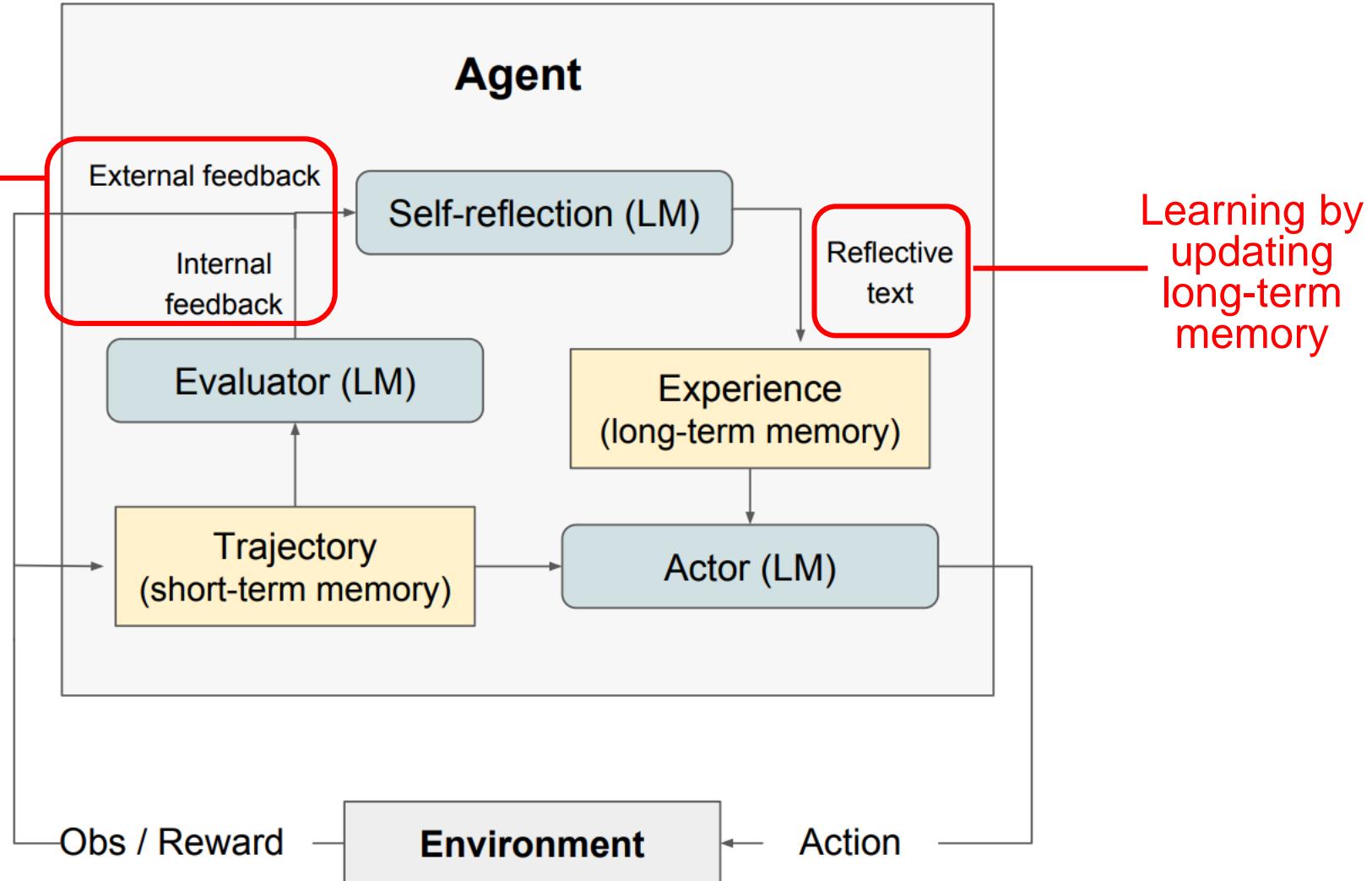
It lacks semantic or episodic memory and therefore has no **retrieval** or **learning** actions.



Cognitive Architectures

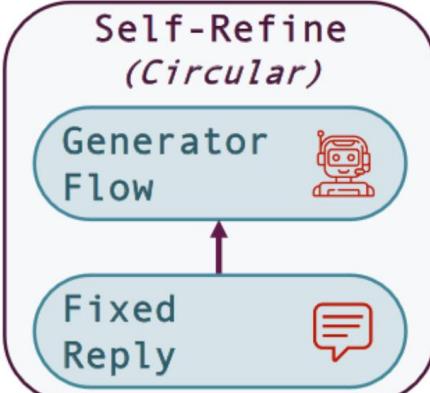
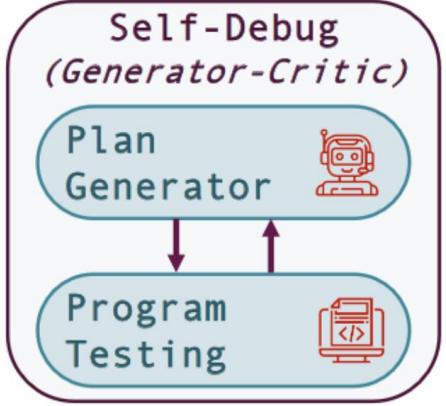
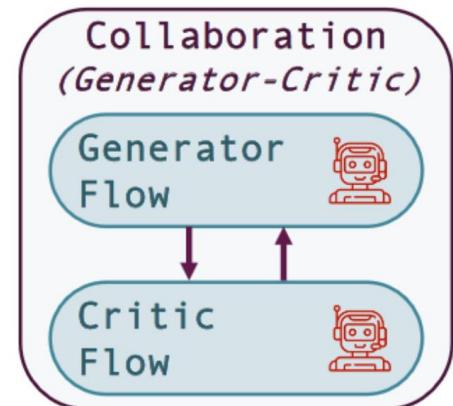
Reflexion

Textual feedback can be provided (“Verbal” RL)

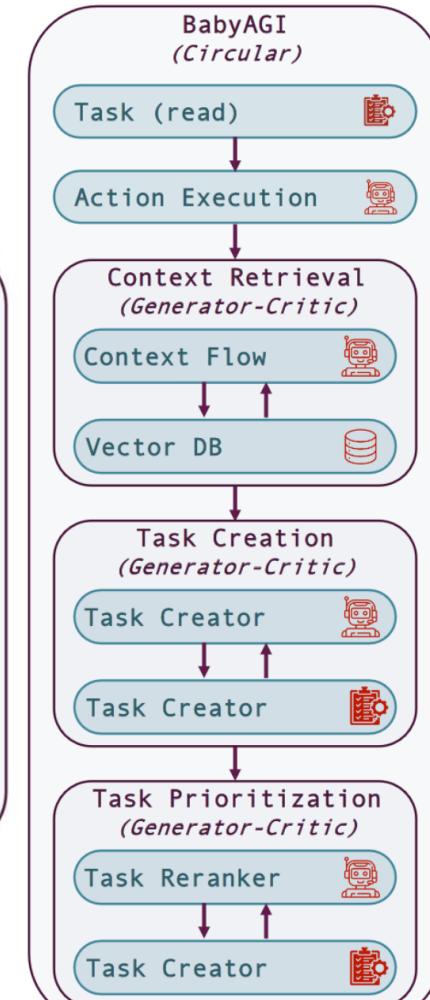
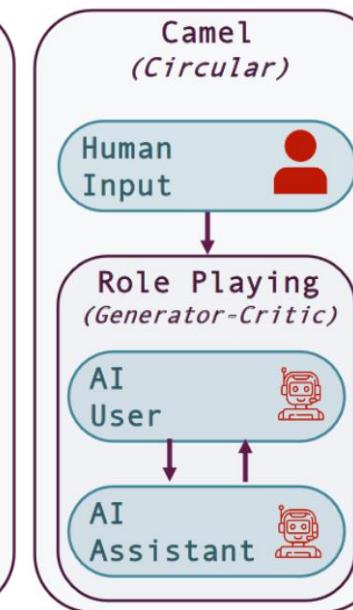
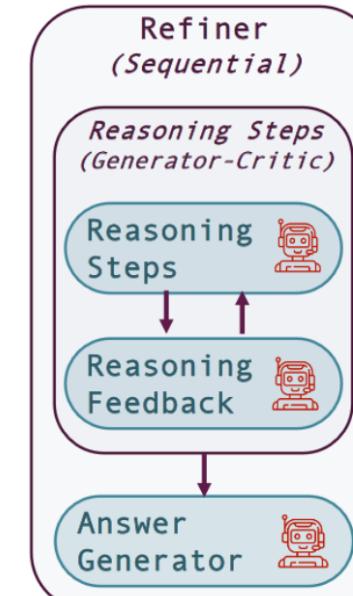


From building blocks to complex patterns

Basic Generator-Critic Patterns



Advanced Generator-Critic Patterns



Overview of the session

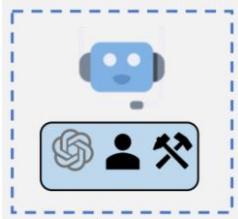
- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



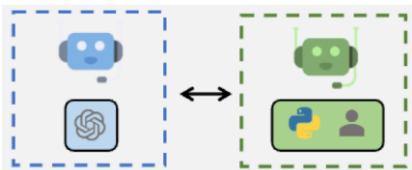
Multi-agent Abstractions



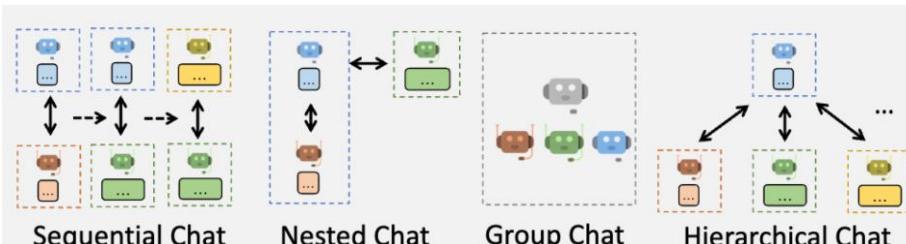
Basic Building Blocks



Each agent is powered by an FM and can use own set of tools.



Agents can converse with other agents or humans.

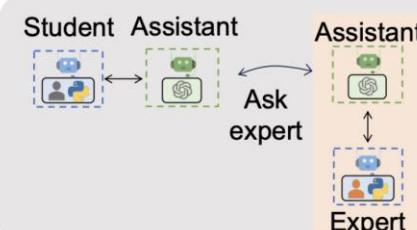


Support multiple conversation patterns.

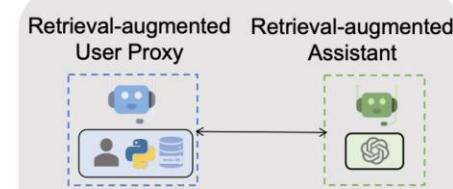
Think of **modularity** when designing multi-agent systems.

Giving each agent its own **system prompt** and **context window** will improve the performance. Breaking the system into parts this way will also help with **debugging**.

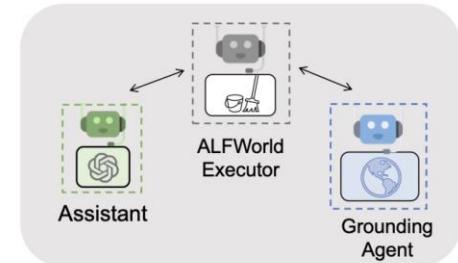
Example Use Cases



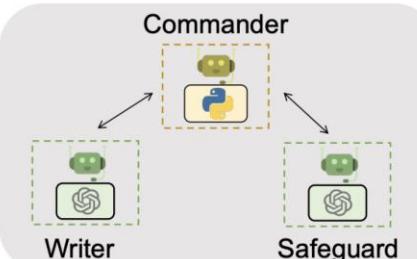
A1. Math Problem Solving



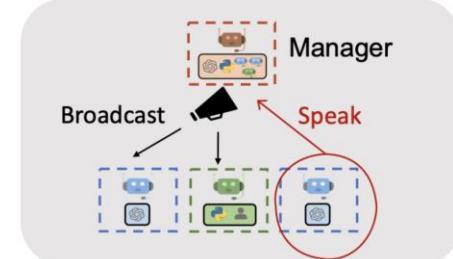
A2. Retrieval-augmented Q&A



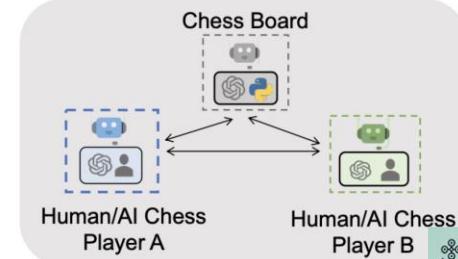
A3. Decision Making in Embodied Agents



A4. Supply-Chain Optimization



A5. Dynamic Task Solving with Group Chat



A6. Conversational Chess



Overview of the session

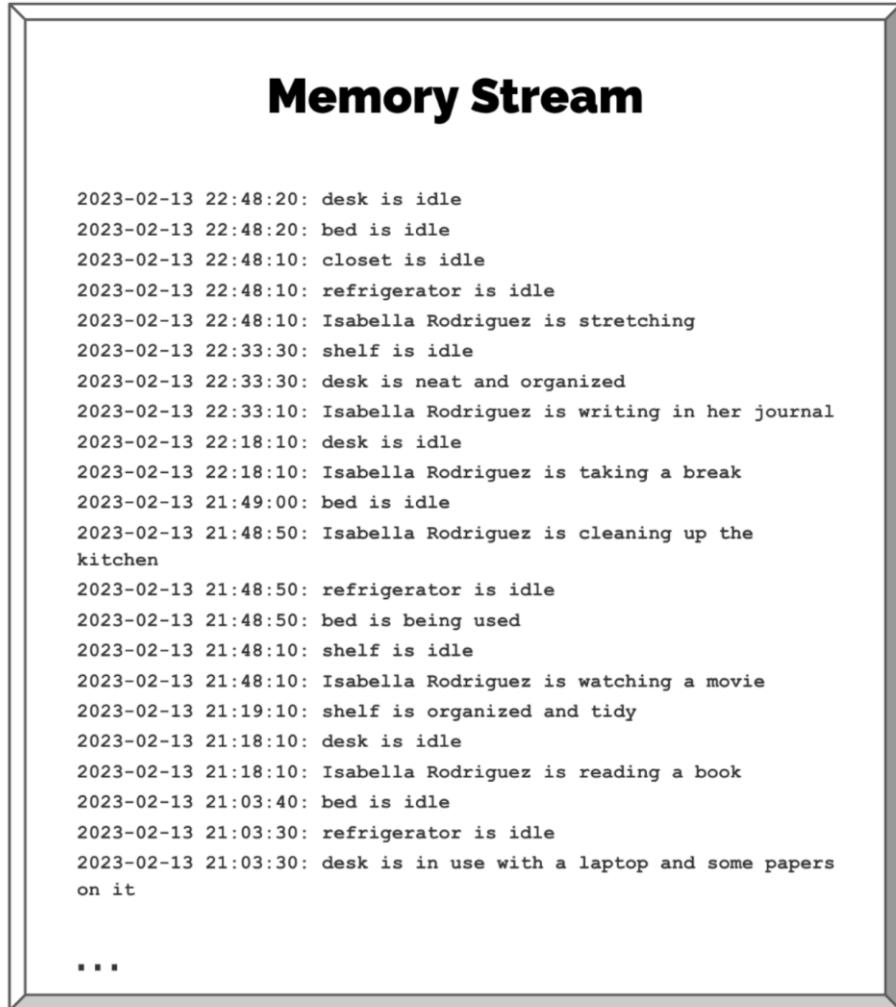
- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



Generative Agents



Querying Episodic Memory



Q. What are you looking forward to the most right now?

Isabella Rodriguez is excited to be planning a Valentine's Day party at Hobbs Cafe on February 14th from 5pm and is eager to invite everyone to attend the party.

retrieval	=	recency	importance	relevance
2.34	=	0.91	+ 0.63	+ 0.80

ordering decorations for the party

2.21	=	0.87	+ 0.63	+ 0.71
------	---	------	--------	--------

researching ideas for the party

2.20	=	0.85	+ 0.73	+ 0.62
------	---	------	--------	--------

...



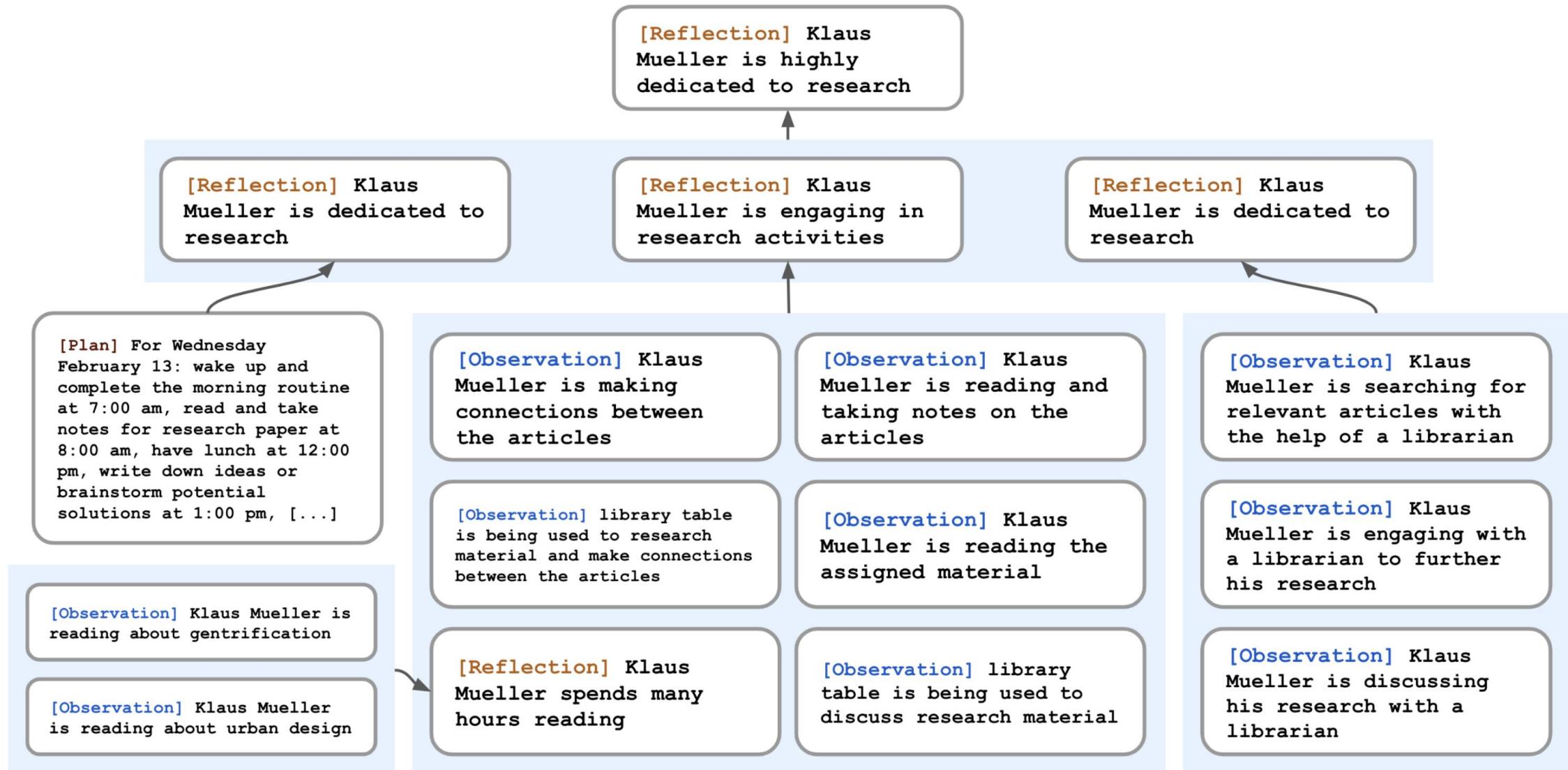
I'm looking forward to the Valentine's Day party that I'm planning at Hobbs Cafe!



Isabella

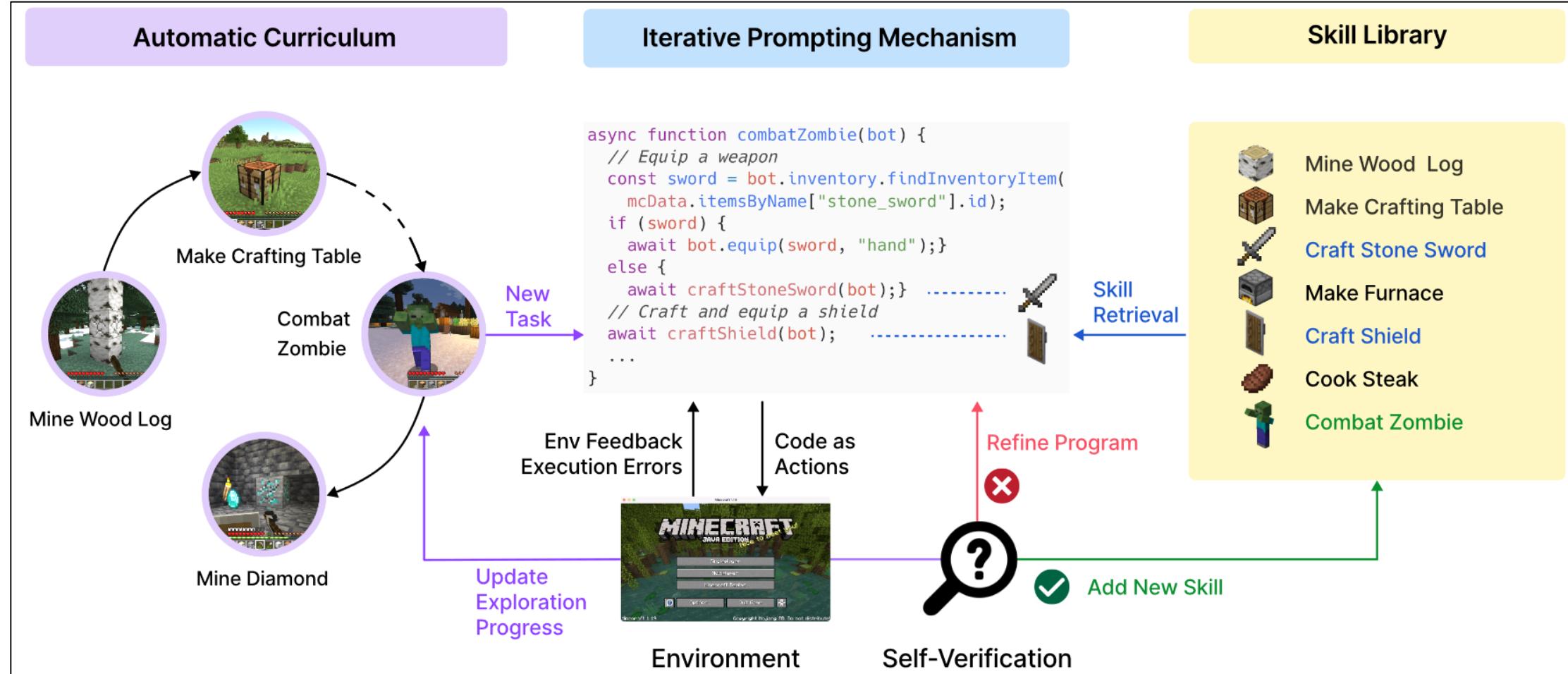


Building Semantic Memory from reflection



Agents in action

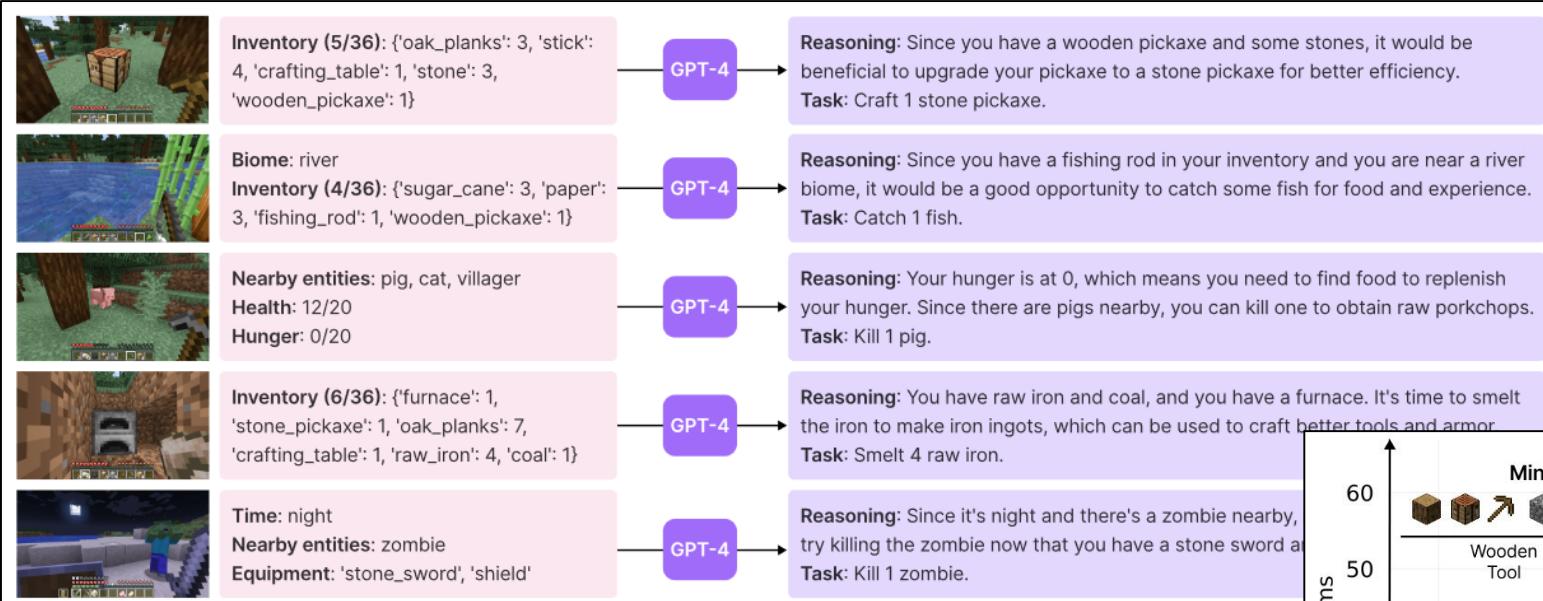
Voyager



Key ideas: Open-ended exploration, code as action space, increasingly complex behaviors

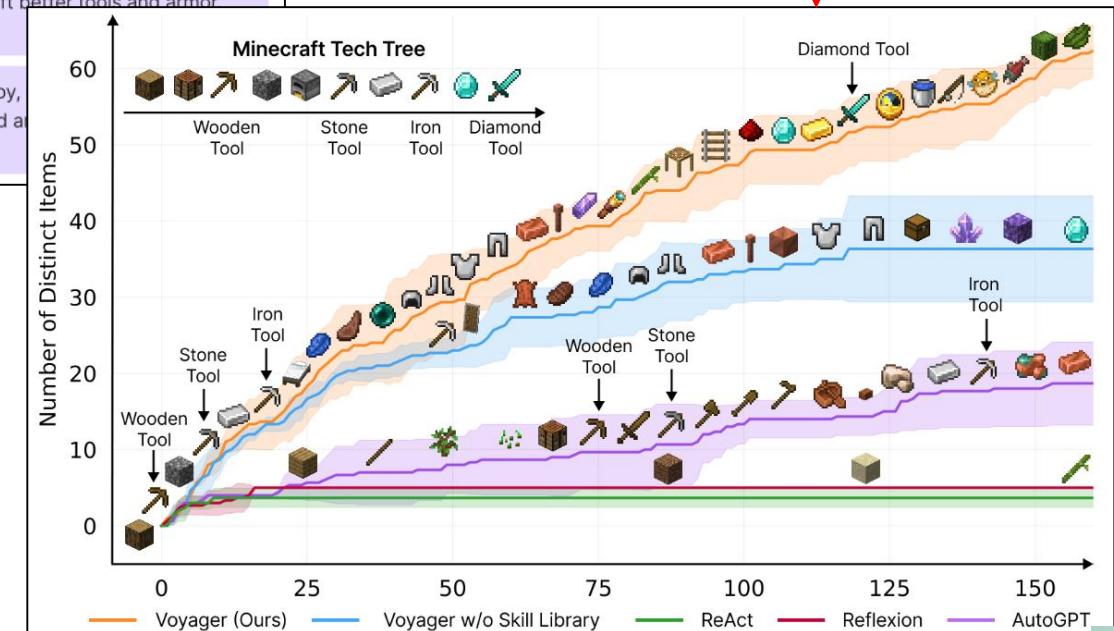


From Procedural Memory to Interaction



Can better explore areas, master the tech tree, and zero-shot generalize to unseen tasks.

- Long-term **procedural memory** that stores a library of code-based skills (e.g., `combatZombie`, `craftStoneSword`).
- Library is **hierarchical**: complex skills can use simpler skills as sub-procedures (e.g., `combatZombie` may call `craftStoneSword` if no sword is in inventory).
- Action space has all four kinds of actions: grounding, reasoning, retrieval, and learning (by adding new skills)

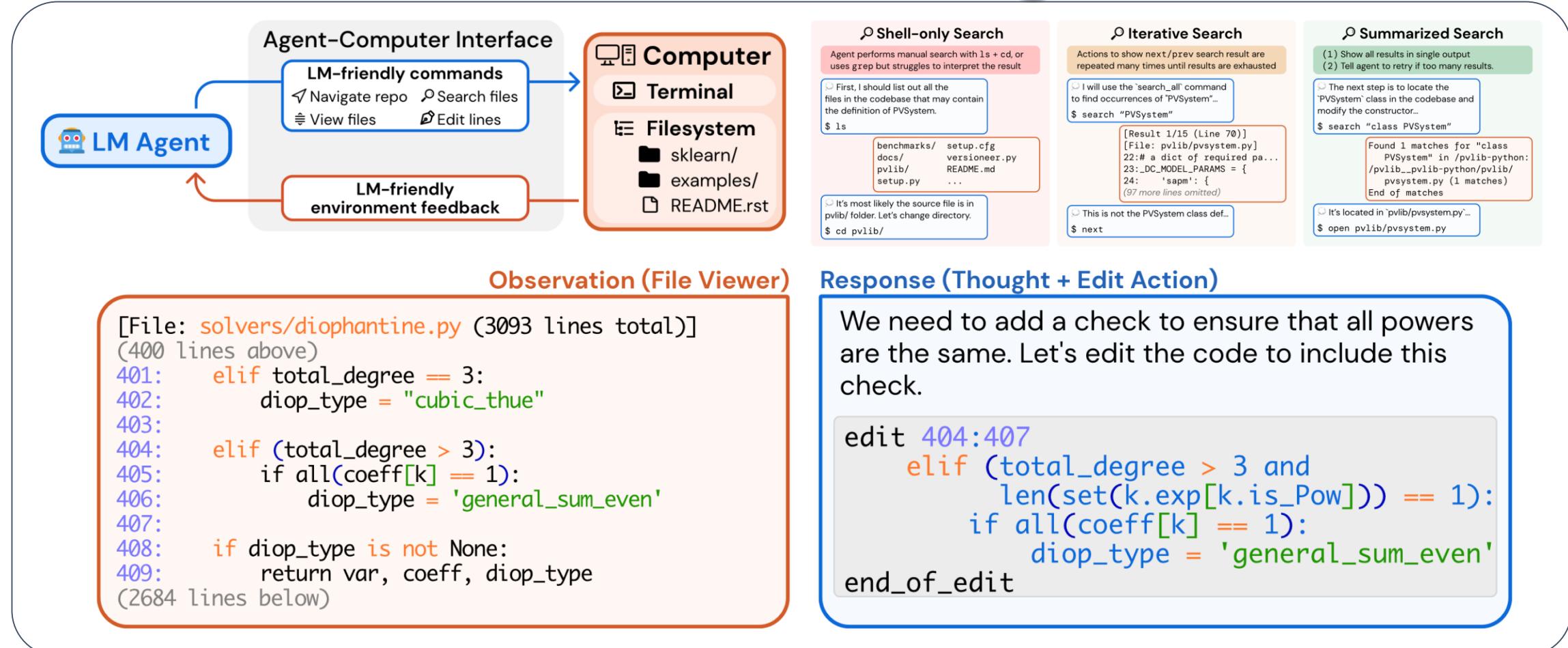


Overview of the session

- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



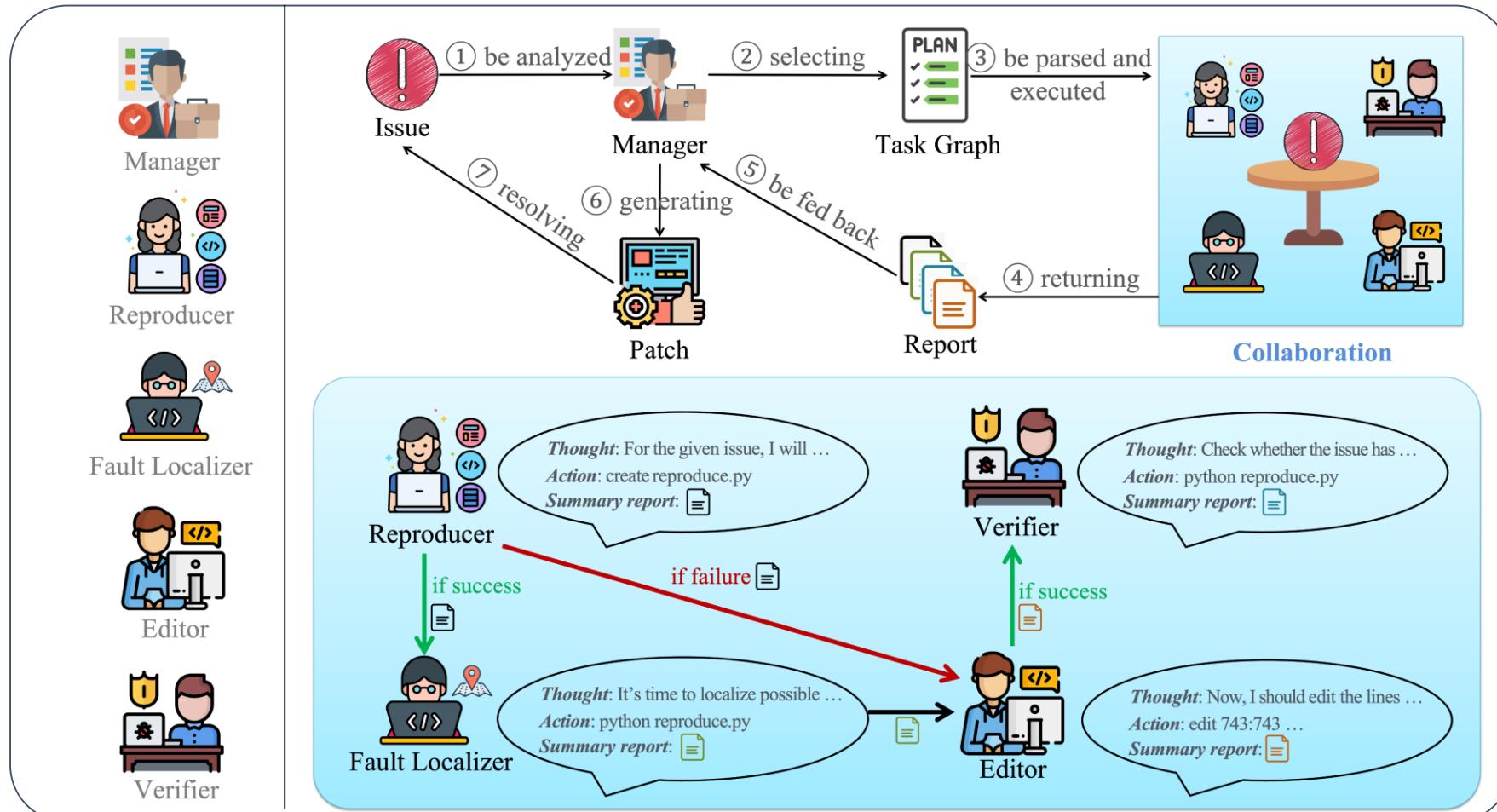
Issue Resolution – SWE-Agent



Key idea: Agent-Computer Interface i.e., Define specialized tools that make it possible to efficiently explore repositories and edit code.



Issue Resolution – CodeR



Key idea: Generate plan (task graph) in advance by manager agent.
Then, execute individual steps by other agents.



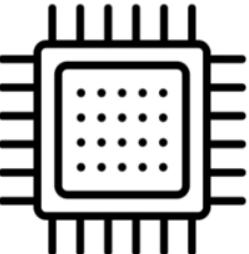
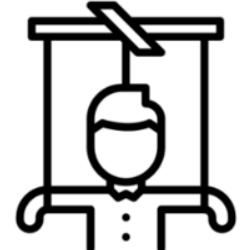
Overview of the session

- ❑ What is an Agent?
- ❑ Agents operate in environments
- ❑ Types of agentic memory
 - ❑ Procedural Memory
 - ❑ Semantic Memory
 - ❑ Episodic Memory
- ❑ Action Space of Agents
- ❑ Reasoning and Planning
- ❑ Cognitive Architectures
- ❑ Multi-agent Abstractions
- ❑ Agents in Action
- ❑ Agent Applications in Software Engineering
- ❑ Open Research Challenges



Open Research Challenges

Controllability

						
Evading shutdown	Hacking computer systems	Run many AI copies	Acquire computation	Attract earnings and investment	Hire or manipulate human assistants	AI research and programming

						
Persuasion and lobbying	Hiding unwanted behavior	Strategically appear aligned	Escaping containment	R&D	Manufacturing and robotics	Autonomous weaponry



Open Research Challenges Human-Agent Interfaces

Claude Q Customizable Pragmatic Engineer Logo ▾

- G Create a web application that allows me to tweak the color of The Pragmatic Engineer Logo. This logo is three rectangular bars, that increase in height, and are in the color red.

Certainly! I'll create a web application that allows you to tweak the color of The Pragmatic Engineer Logo using React. This application will display the logo and provide controls to adjust its color.

Pragmatic Engineer Logo Color Tweaker
Click to open component
Published

I've created a React component that allows you to tweak the color of The Pragmatic Engineer Logo. Here's a breakdown of the application:

1. The logo is represented by three rectangular bars that increase in height, as per your description.
2. The initial color is set to red (#FF0000).
3. A color picker input allows you to change the color of all three bars simultaneously.
4. The current color value is displayed below the color picker.

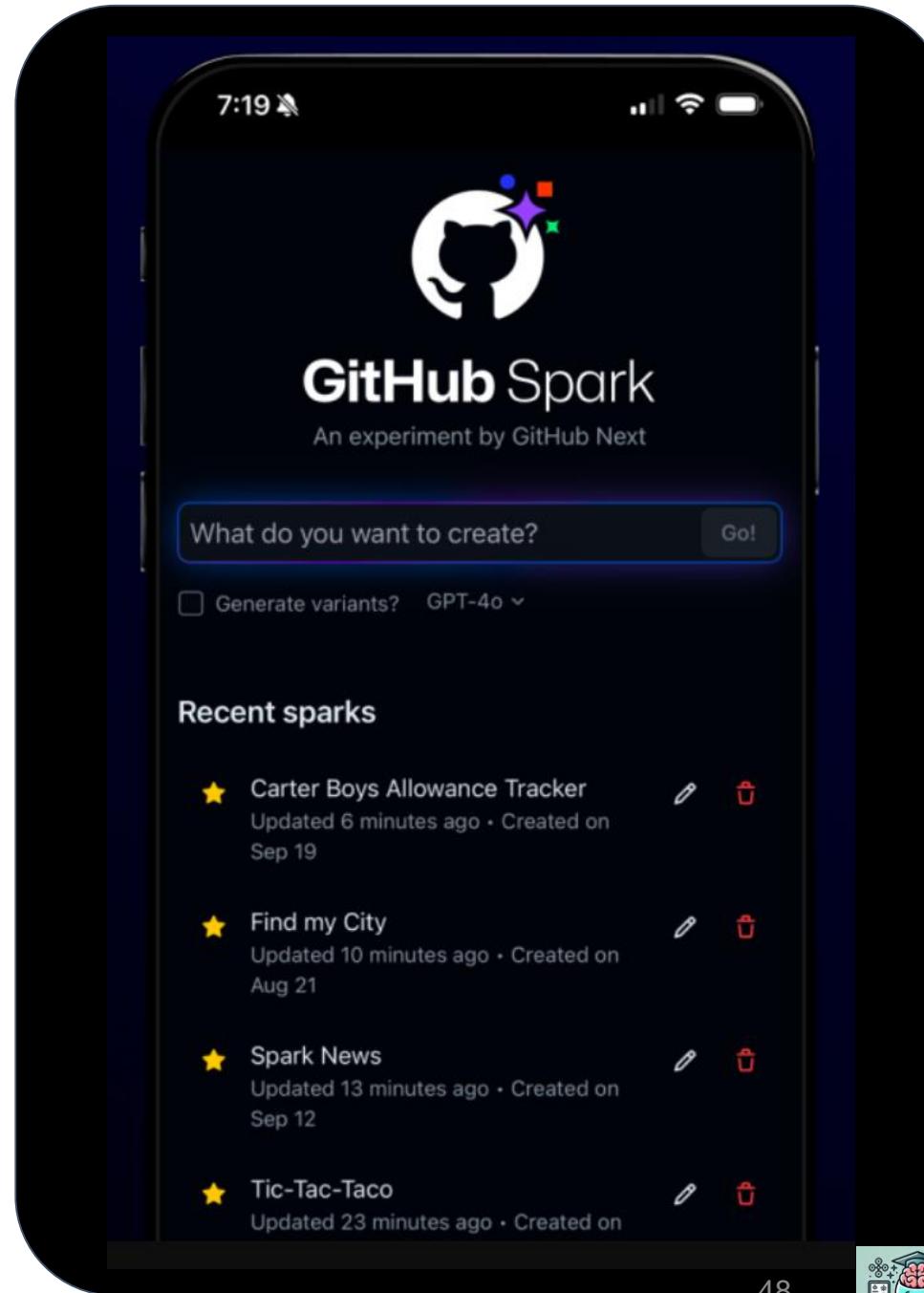
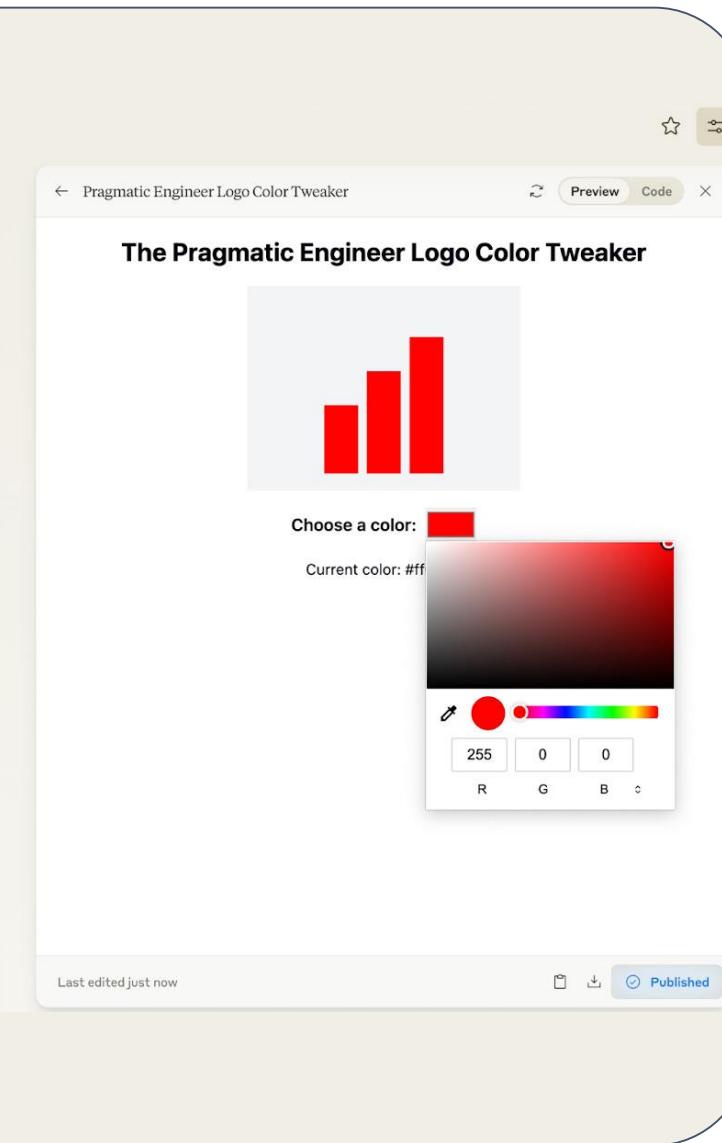
To use this application:

1. Copy the code from the artifact and save it as a `.jsx` or `.tsx` file in your React project.
2. Import and use the `LogoColorTweaker` component in your `main application file`.

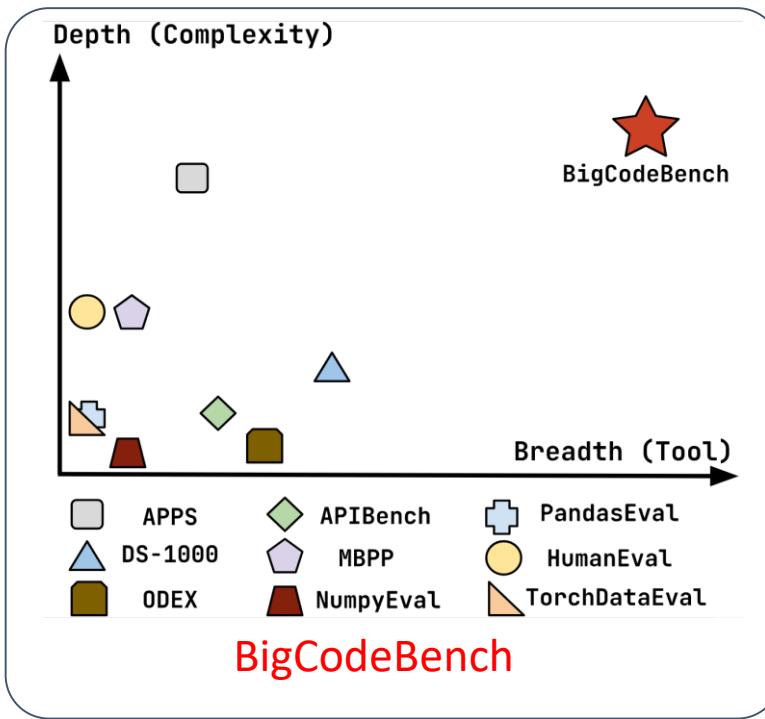
Reply to Claude...

6

Claude 3.5 Sonnet ▾



Open Research Challenges Evaluation

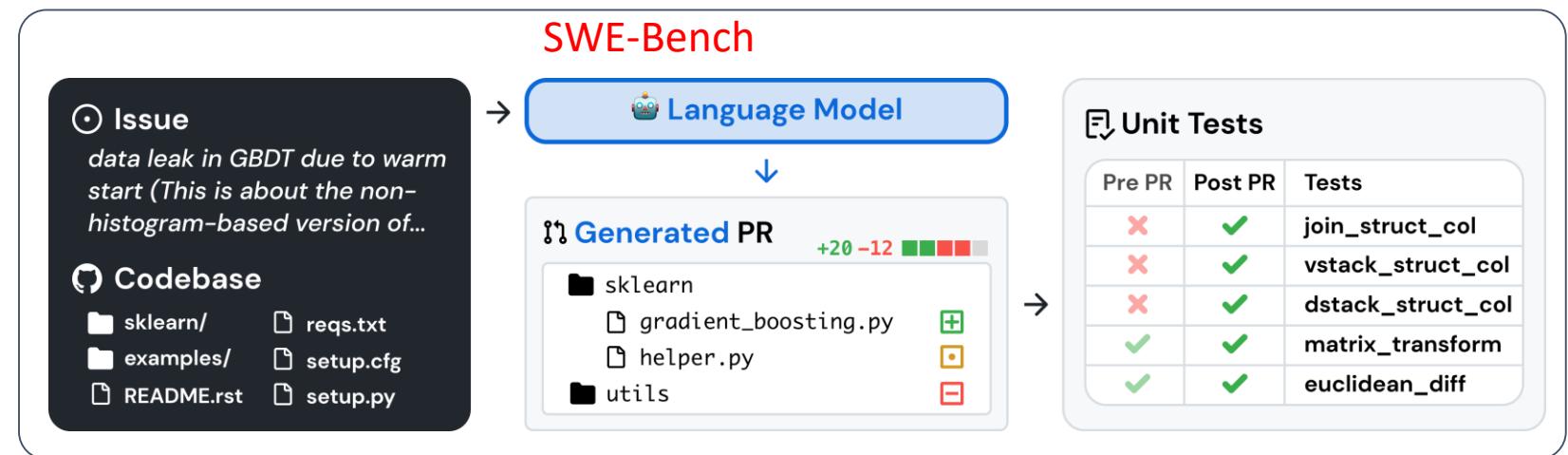


Yao et al., τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains

Jimenez et al., SWE-bench: Can Language Models Resolve Real-World GitHub Issues?

Bahdanau et al., TapeAgents: a Holistic Framework for Agent Development and Optimization

<https://bigcode-bench.github.io/>



- Grounded
- Responsive
- Accurate
- Disciplined
- Transparent
- Helpful

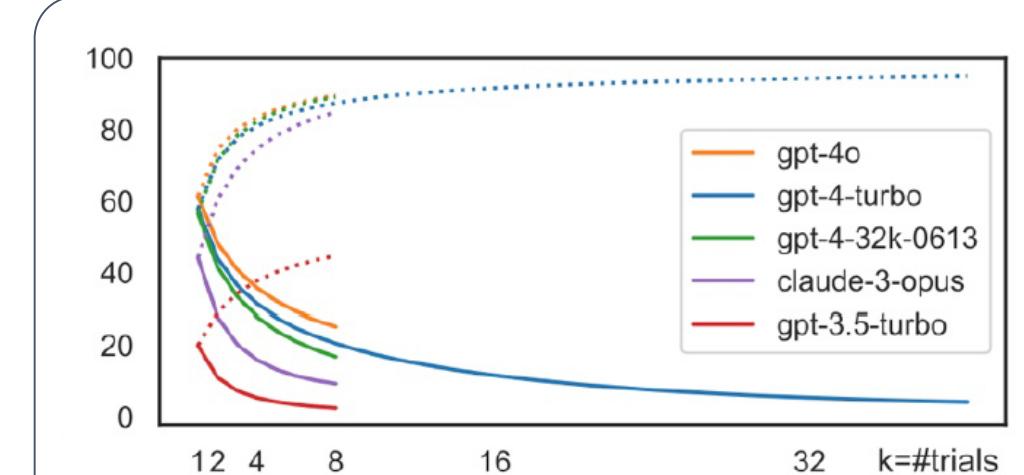


Figure 4: $\text{pass}^k (-)$ and $\text{pass}@k(..)$ in τ -retail.

τ-bench



What is an agent? Why does it matter?

Carl Hewitt recently remarked that the question “**what is an agent?**” is embarrassing for the agent-based computing community in just the same way that the question “what is intelligence?” is embarrassing for the mainstream AI community. The problem is that although the term is widely used, by many people working in closely related areas, it defies attempts to produce a single universally accepted definition. This need not necessarily be a problem: after all, if many people are successfully developing interesting and useful applications, then it hardly matters that they do not agree on potentially trivial terminological details. However, there is also the danger that unless the issue is discussed, ‘agent’ might become a ‘noise’ term, subject to both abuse and misuse, to the potential confusion of the research community.

— Michael Wooldridge,

in **1994**, *Intelligent Agents: Theory and Practice*



Talking about agents

- The sudden presence among us of **exotic, mind-like entities** might precipitate a shift in the way we use familiar psychological terms like “**believes**” and “**thinks**”.
- It may require an extensive period of interacting with, or living with, these new kinds of artefact before we learn how best to **talk about them**.
- Meanwhile, we should try to resist the siren call of **anthropomorphism**.

