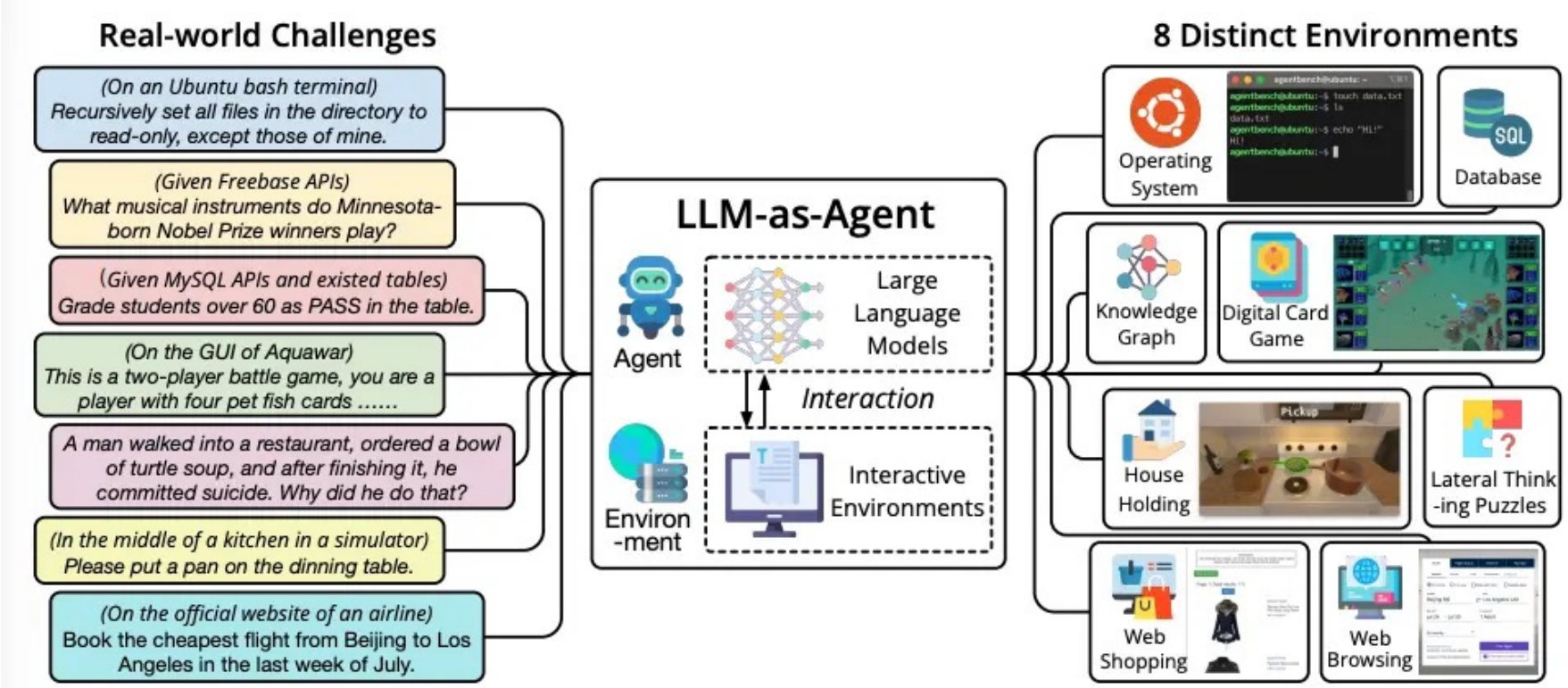
AutoManual: Constructing Instruction Manuals by LLM Agents via Interactive Environmental Learning

NeurIPS 2024

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, Xiaofei He

Background of LLM Agents

Large Language Models (LLM)-based Agents have shown promise in autonomously completing tasks across various domains, e.g., ***device control, games, robotics,*** and ***web navigation***.



[Aug 2023] **"AgentBench: Evaluating LLMs as Agents."** *Xiao Liu (THU) et al.* arXiv.

Typical LLM Agents: **Voyager**

- Voyager is a LLM-powered embodied agent in Minecraft

Voyager: An Open-Ended Embodied Agent with Large Language Models

Guanzhi Wang [1 2], Yuqi Xie[3], Yunfan Jiang[4*], Ajay Mandlekar[1*],
Chaowei Xiao[1 5], Yuke Zhu[1 3], Linxi "Jim" Fan [1†], Anima Anandkumar [1 2†]
[1]NVIDIA, [2]Caltech, [3]UT Austin, [4]Stanford, [5]ASU
[*]Equal contribution [†]Equal advising
Corresponding authors: guanzhi@caltech.edu, dr.jimfan.ai@gmail.com

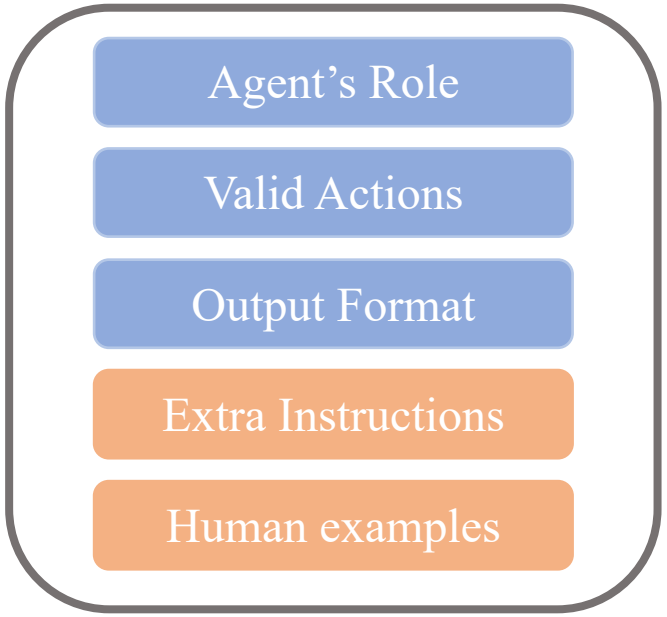
arXiv | PDF | Code | Tweet | MineDojo

Limitations of **Voyager**

Voyager is specifically designed for the Minecraft environment:

- Environment-specific knowledge
- Multiple demonstrations from humans

are fed into the prompts of LLM.



System Prompts of A LLM Agent

```
You should then respond to me with
Explain (if applicable): Are there any steps missing in your plan? Why
    does the code not complete the task? What does the chat log and
    execution error imply?
Plan: How to complete the task step by step. You should pay attention
    to Inventory since it tells what you have. The task completeness
    check is also based on your final inventory.
Code:
    1) Write an async function taking the bot as the only argument.
    2) Reuse the above useful programs as much as possible.
        - Use `mineBlock(bot, name, count)` to collect blocks. Do not
    use `bot.dig` directly.
        - Use `craftItem(bot, name, count)` to craft items. Do not use
     `bot.craft` directly.
        - Use `smeltItem(bot, name count)` to smelt items. Do not use
    `bot.openFurnace` directly.
        - Use `placeItem(bot, name, position)` to place blocks. Do not
     use `bot.placeBlock` directly.
        - Use `killMob(bot, name, timeout)` to kill mobs. Do not use `
    bot.attack` directly.
    3) Your function will be reused for building more complex
    functions. Therefore, you should make it generic and reusable. You
     should not make strong assumption about the inventory (as it may
    be changed at a later time), and therefore you should always check
     whether you have the required items before using them. If not,
    you should first collect the required items and reuse the above
    useful programs.
    4) Functions in the "Code from the last round" section will not be
     saved or executed. Do not reuse functions listed there.
    5) Anything defined outside a function will be ignored, define all
     your variables inside your functions.
    6) Call `bot.chat` to show the intermediate progress.
    7) Use `exploreUntil(bot, direction, maxDistance, callback)` when
    you cannot find something. You should frequently call this before
    mining blocks or killing mobs. You should select a direction at
    random every time instead of constantly using (1, 0, 1).
    8) `maxDistance` should always be 32 for `bot.findBlocks` and `bot
    .findBlock`. Do not cheat.
    9) Do not write infinite loops or recursive functions.
    10) Do not use `bot.on` or `bot.once` to register event listeners.
     You definitely do not need them.
    11) Name your function in a meaningful way (can infer the task
    from the name).
```

Limitations of **ReAct**-like Agents

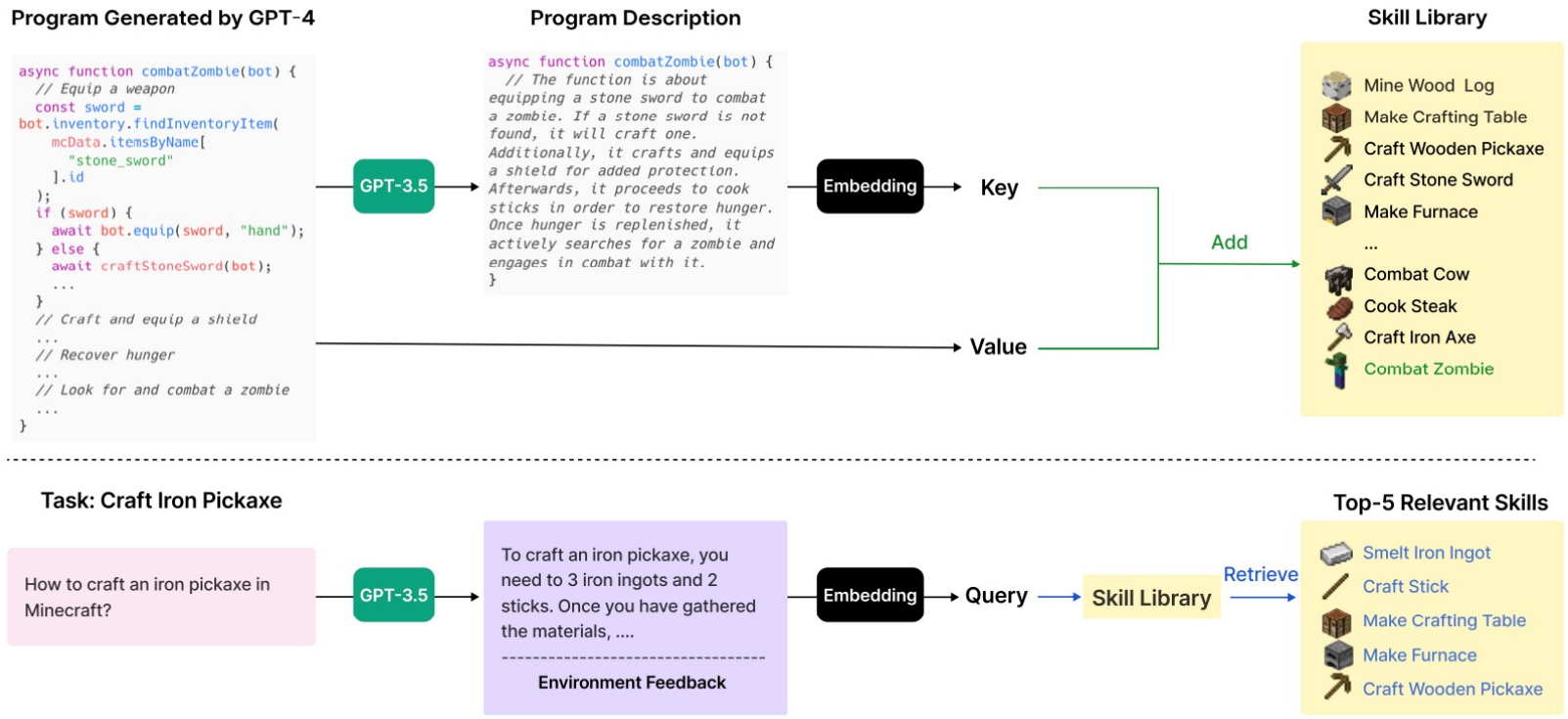
This problem is widespread in ReAct-like agents:

Number of Human Examples and Success rate (\%) of LLM agent methods on **ALFWorld**

Methods	Examples	Put	Clean	Heat	Cool	Examine	Put two	ALL
Testing LLM: GPT-3.5-turbo								
ReAct [33]	12	75.0	24.7	37.7	36.4	44.4	11.8	41.9
Reflexion [16]	12	87.5	44.1	73.9	50.0	61.1	35.3	59.8
ExpeL [35]	12	62.5	61.3	30.4	61.9	55.5	35.3	52.2
AdaPlanner [21]	6	83.3	46.2	65.2	74.2	68.5	52.9	63.3
Planner+Lib.	1	77.8	**88.2**	82.6	72.7	37.0	27.5	66.5
AutoManual	1	**95.8**	79.6	**87.0**	**78.8**	**100.0**	**66.7**	**86.2**
Testing LLM: GPT-4-turbo								
ReAct [33]	12	95.8	76.3	69.6	86.4	72.2	52.9	76.8
Reflexion [16]	12	**100.0**	95.7	78.3	86.4	77.8	70.6	85.9
ExpeL [35]	12	94.4	82.8	72.4	81.8	72.2	58.8	79.2
AdaPlanner [21]	6	88.9	90.3	85.5	75.8	64.8	41.2	76.4
Planner+Lib.	1	**100.0**	93.5	**100.0**	93.9	88.9	39.2	88.1
AutoManual	1	**100.0**	**98.9**	**100.0**	**95.4**	**100.0**	**90.2**	**97.4**

LLM Agents Learn From Interactions

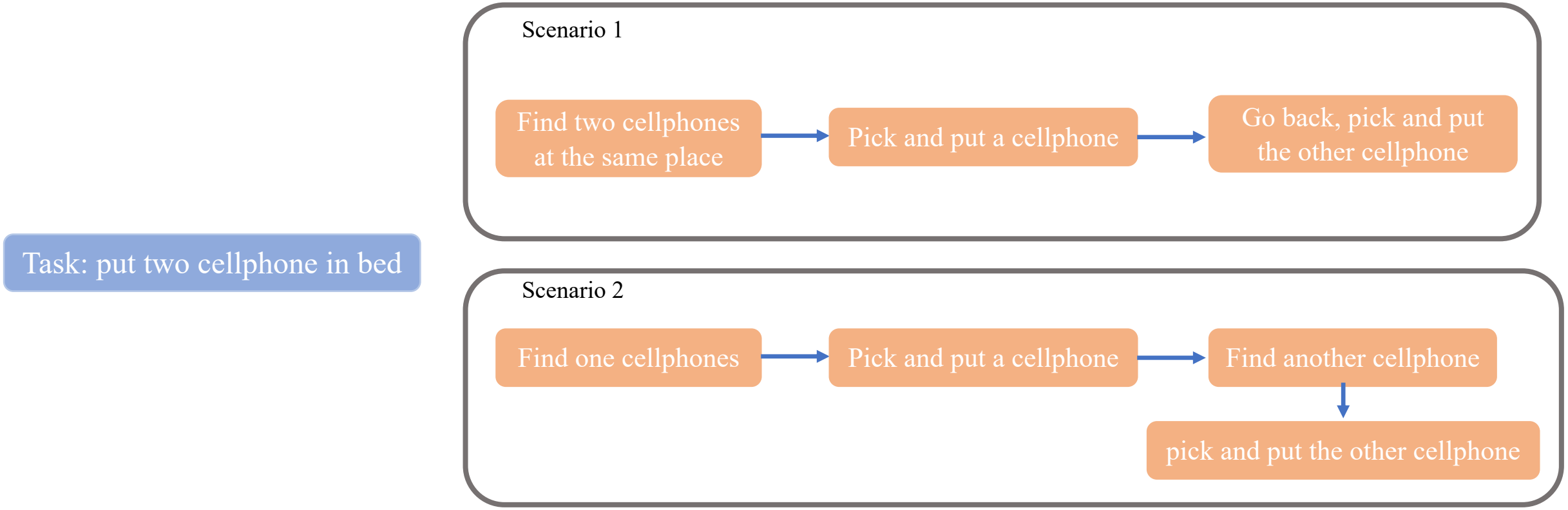
Some work uses **self-reflection** or **skill library** to enable LLM Agents to improve themselves.



Skill library (from Voyager)

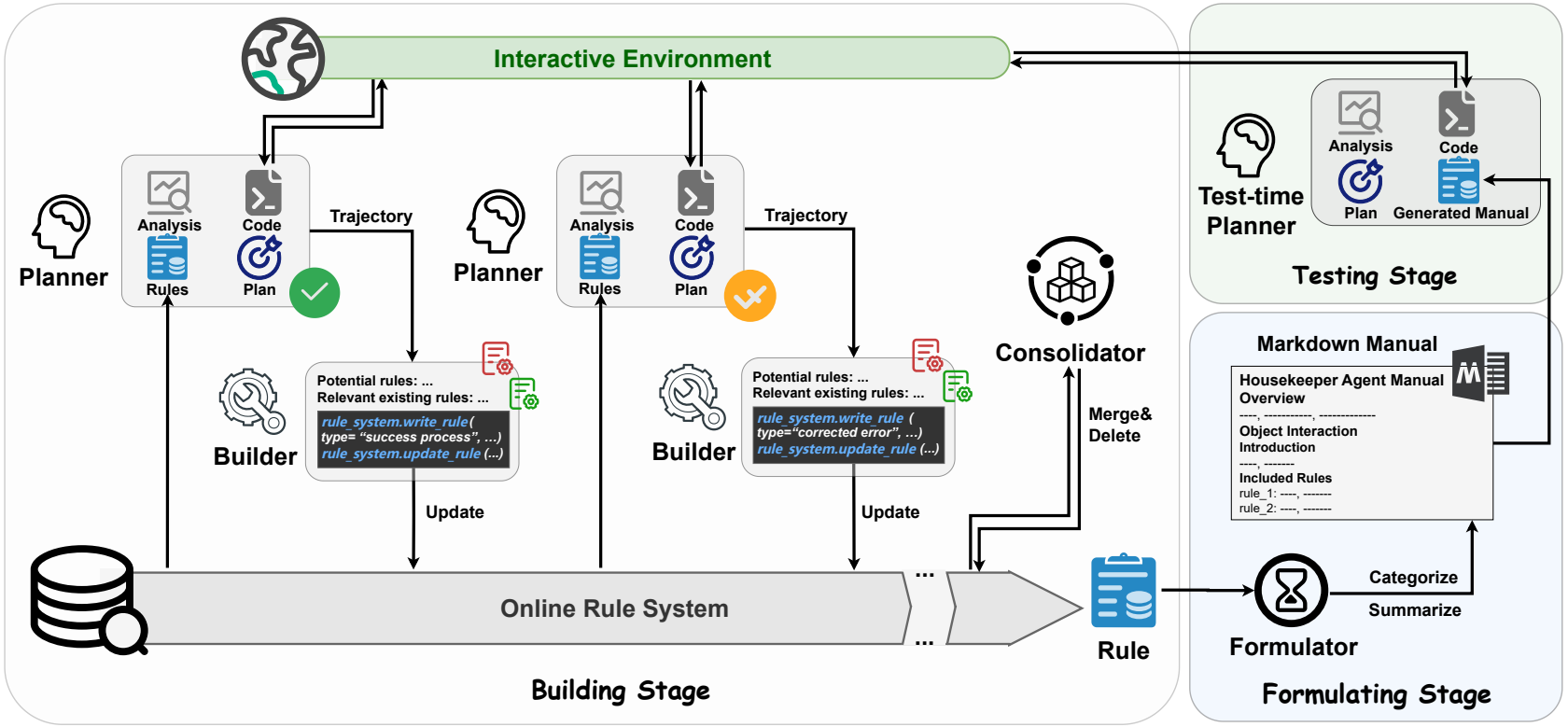
LLM Agents Learn From Interactions

However, these reflections and skills have not been well exploited to **foster a deeper understanding** of the environment. As a result, directly using saved skills as in-context examples can lead to the **Path Dependence problem.**



AutoManual Overview

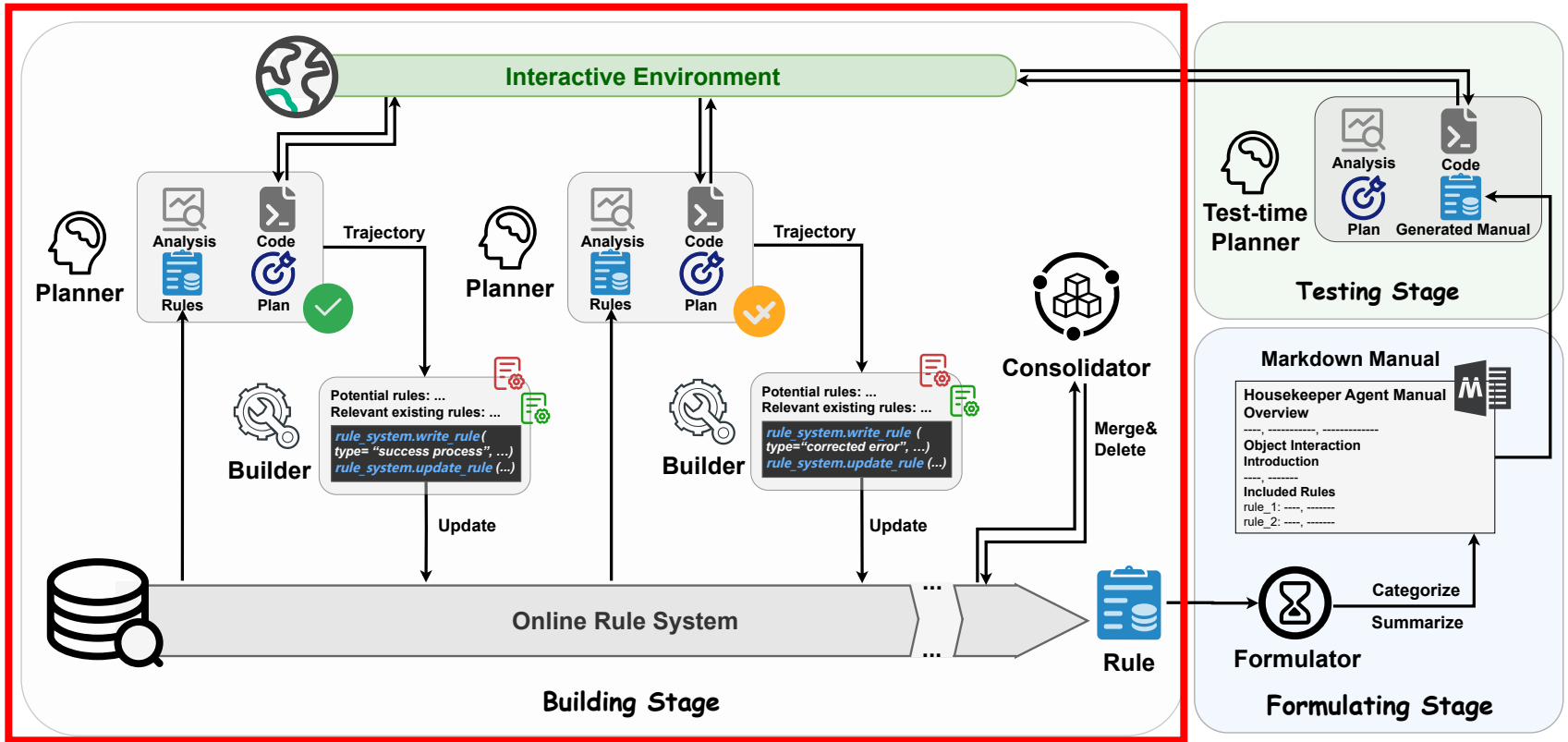
- **Building stage:** build rules from the interactive environment.
- **Formulating stage:** formulates rules into a manual.
- **Testing stage:** A test-time Planner agent will be evaluated with the manual.



AutoManual: Building Stage

Two alternating iterative processes:

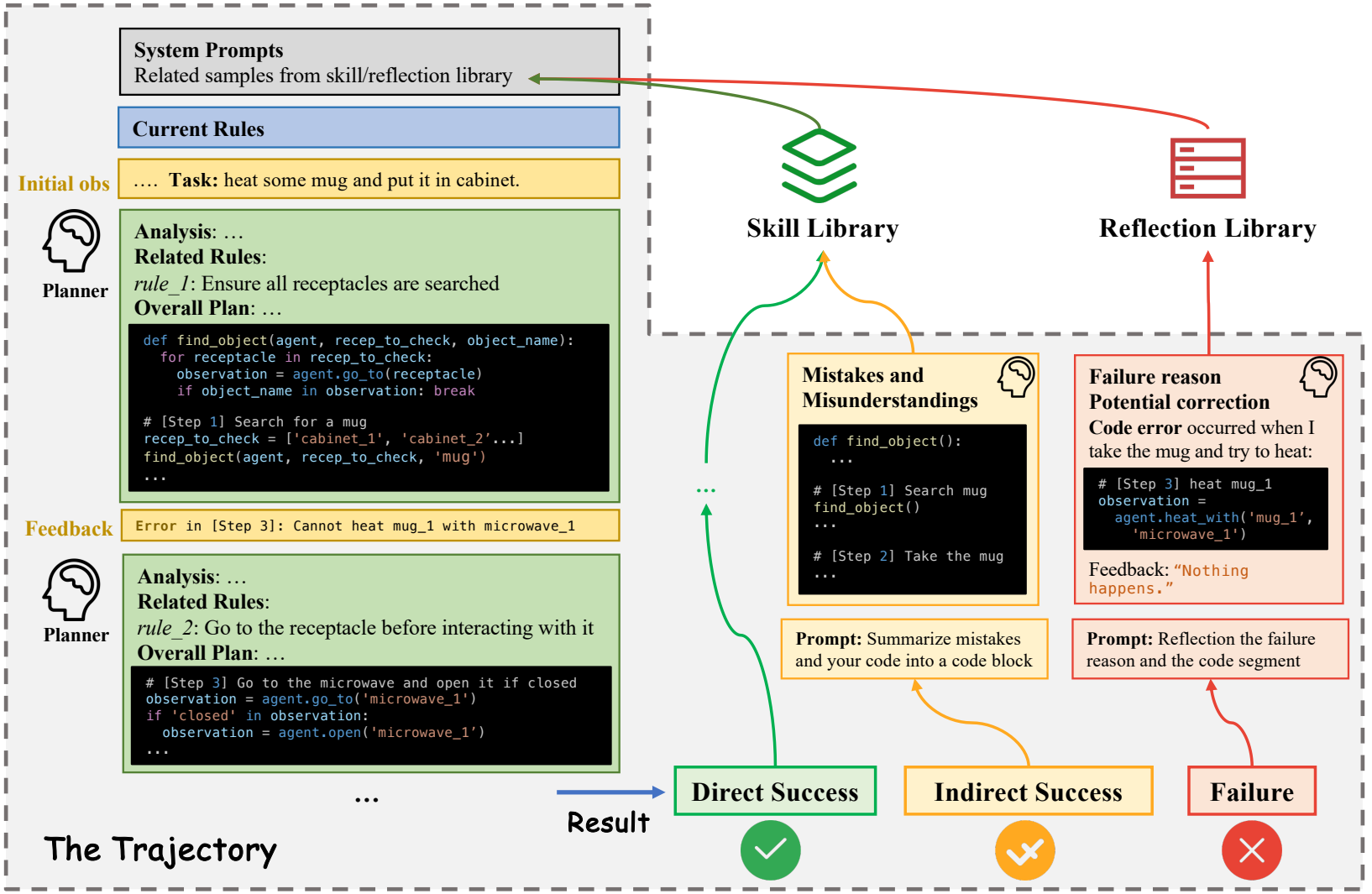
- **The Planner agent** interacts with the environment for an episode.
- **The Builder agent** updates the rules through a rule system.



AutoManual: Planner Agent

The output of the Planner:

1. **Analysis**
2. **Related Rules**
3. **Overall Plan**
4. **Code**.

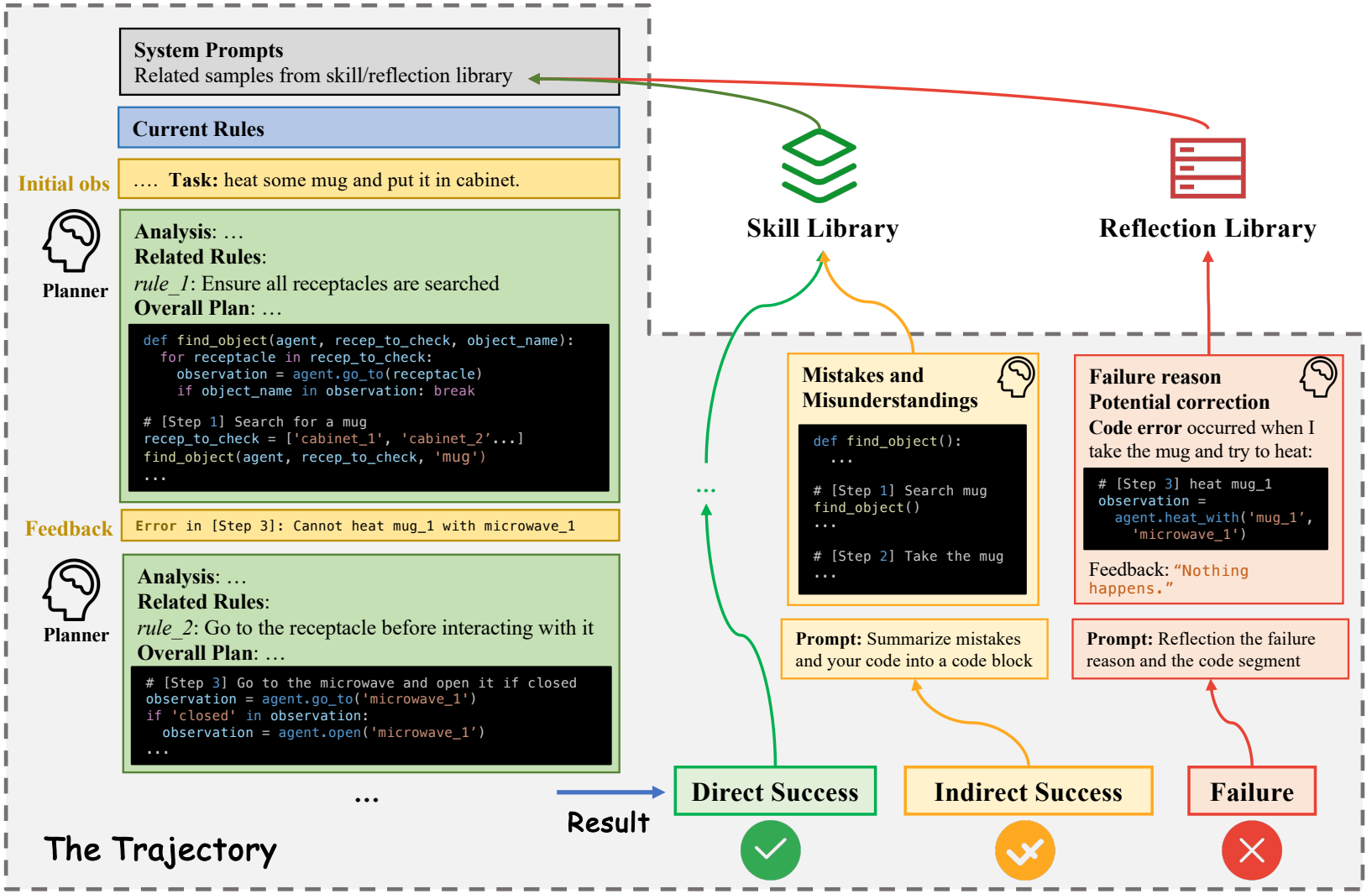


AutoManual: Planner Agent

The episodic result is categorized into:

- **Direct Success**
- **Indirect Success**
- **Failure**

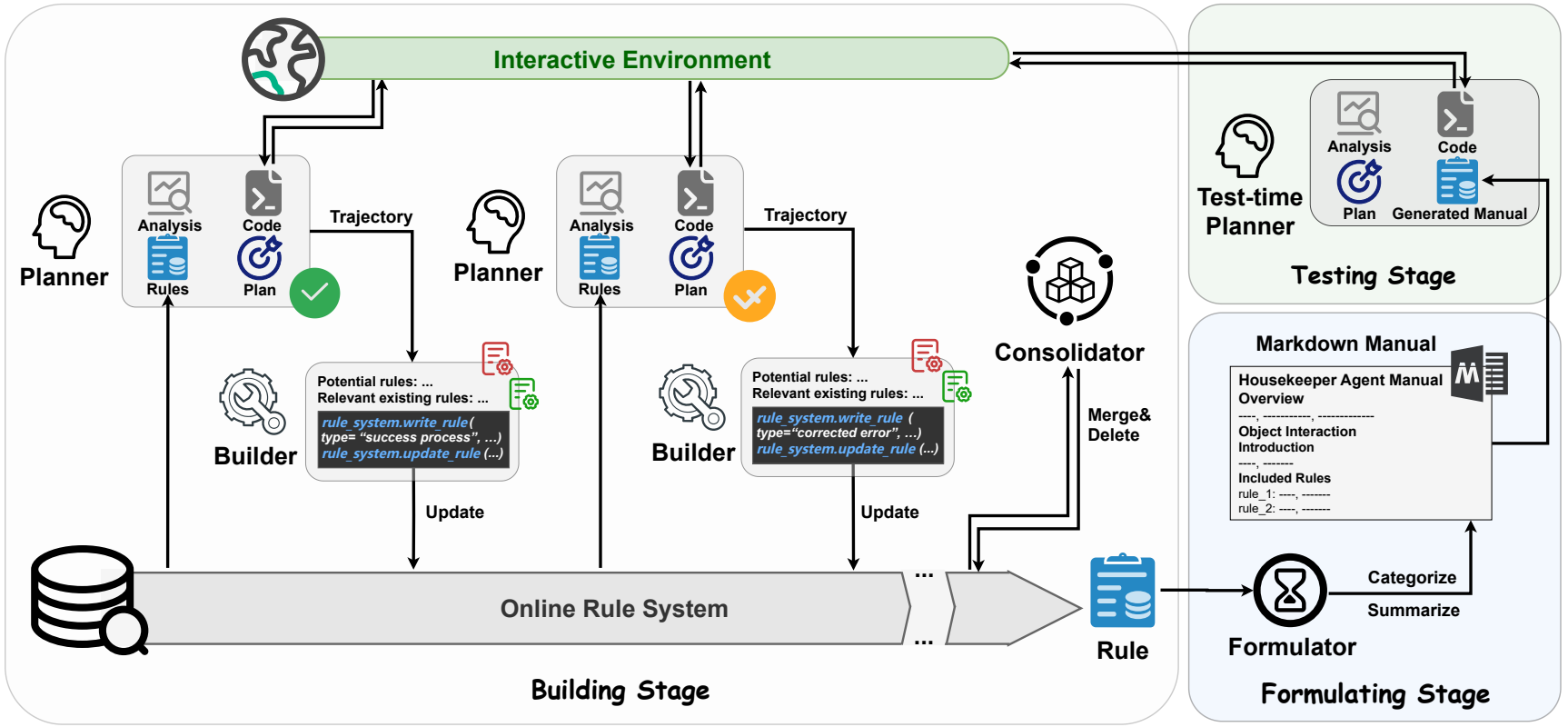
The Planner is prompted to summarize the skill code or reflection accordingly.



AutoManual: Builder Agent

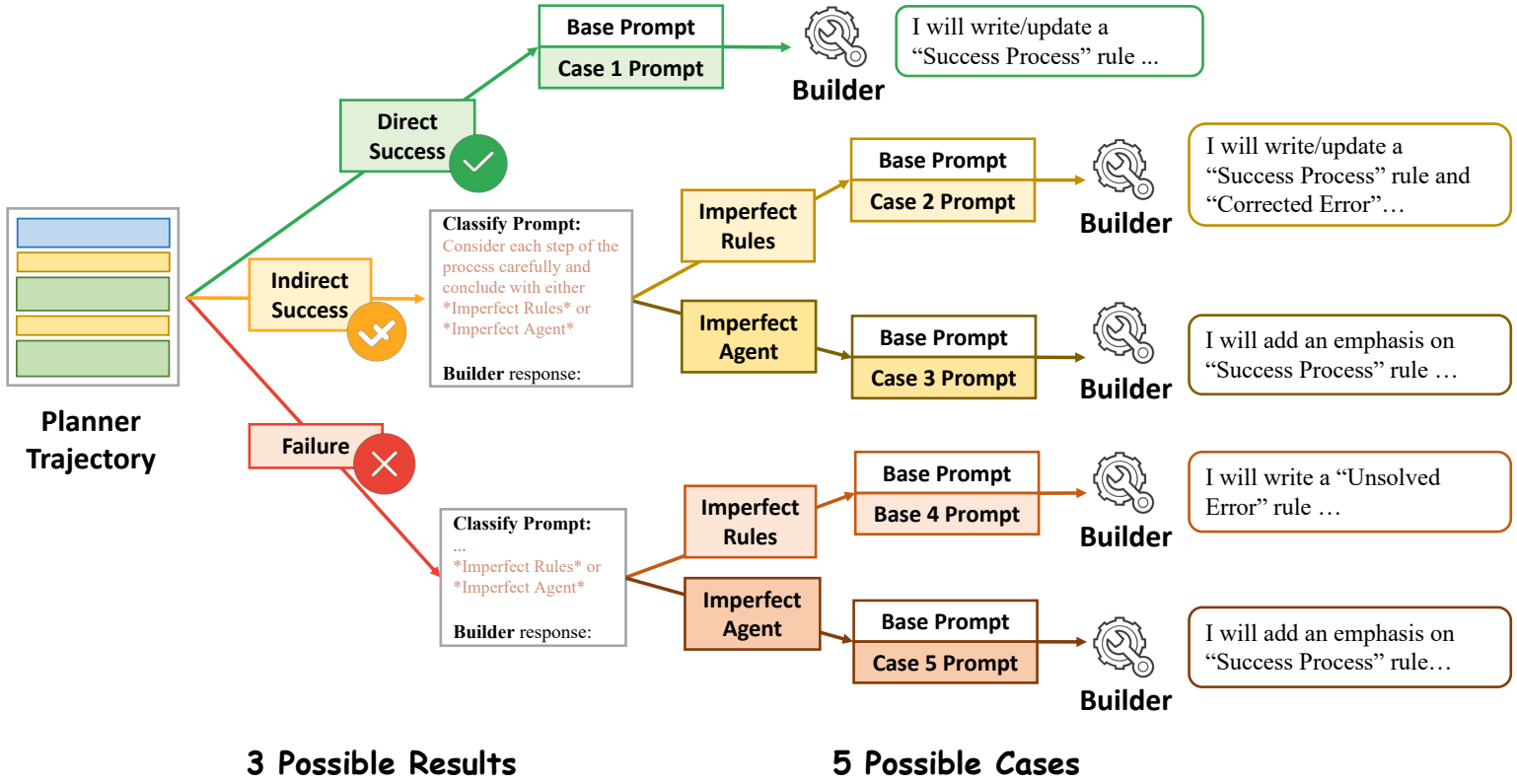
Upon receiving the trajectory of the Planner, the Builder has to update the rules through the rule system. Each rule in the rule system has four attributes:

1. **Rule Type** 2. **Rule Content** 3. **Rule Example** 4. **Validation Logs**.



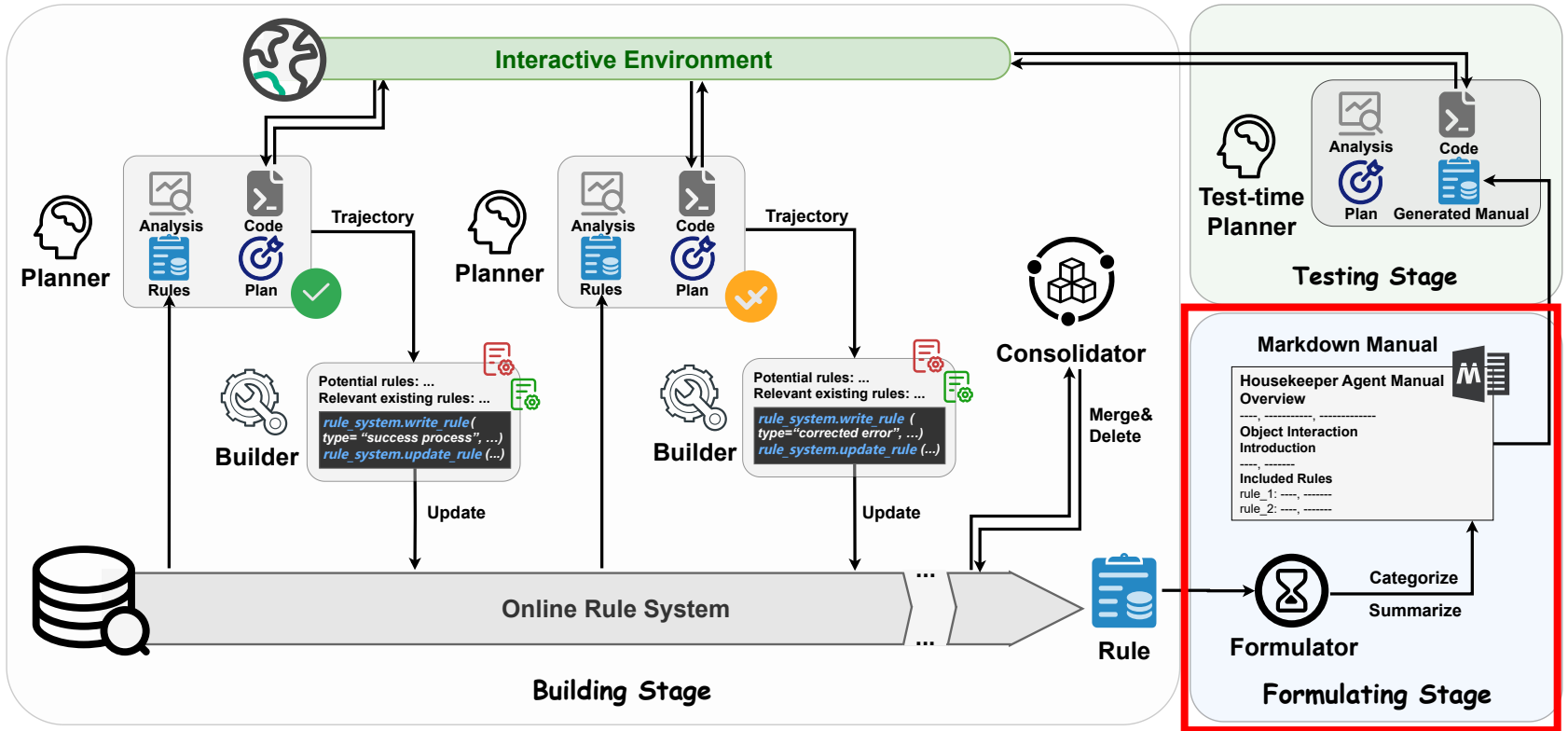
AutoManual: Builder Agent

To mitigate the hallucinations of the Builder, we employ ***case-conditioned prompting***: The Builder first determines the type of the major errors, then a targeted prompt directs the Builder to focus on specific rules.



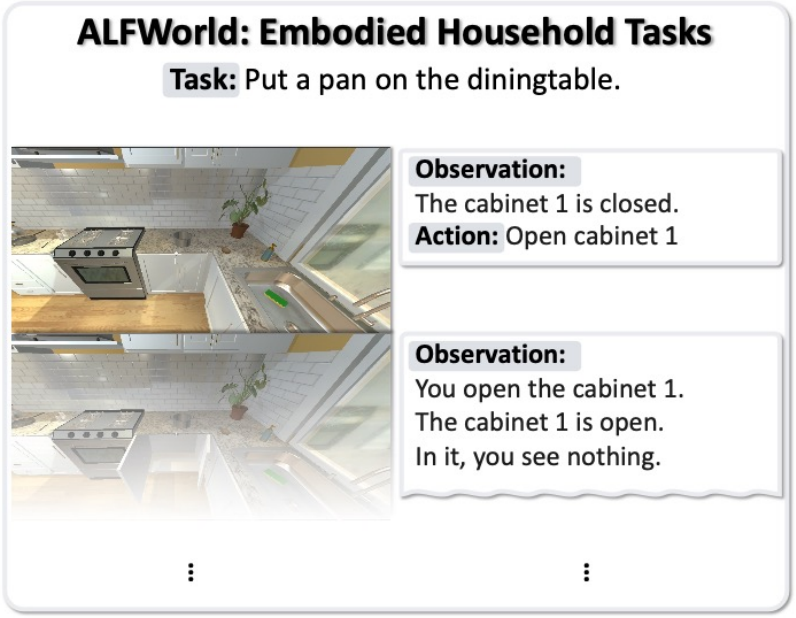
AutoManual: Formulating Stage

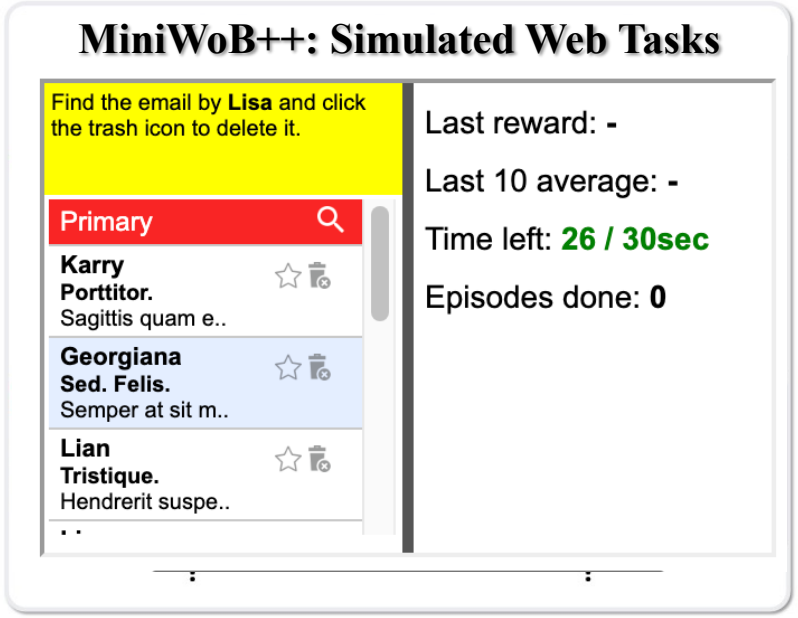
- The Formulator agent categorizes the rules, summarizes the key points, and formulates them into a manual in Markdown form.

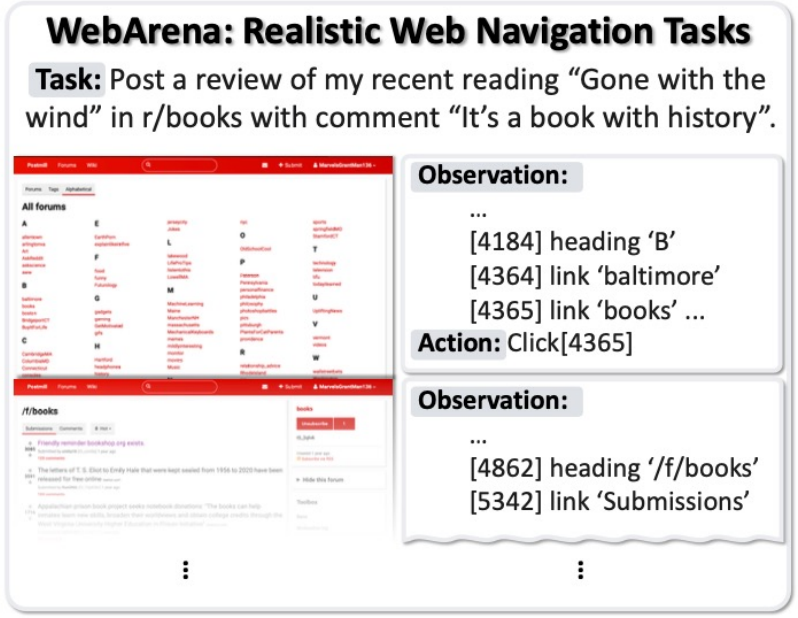


Experiment: Environments

1. **ALFWorld** is a text-based virtual environment for the household robot.
2. **MiniWoB++** is a simulated web environment where agents complete diverse tasks on the Internet by performing keyboard and mouse actions.
3. **WebArena** (Reddit) is a realistic web environment by emulating the functionality and data of Reddit website.







Experiment: Results

- Building and formulating stages: all agents use **GPT-4-turbo (*gpt-4-1106-preview*)**.
- Testing stage: we equip the Planner agent with **GPT-4-turbo or GPT-3.5-turbo**, to evaluate whether generated manuals can guide smaller LLM.

Success rate (\%) of LLM agent methods on **ALFWorld** test tasks (6 task types)

Methods	Examples	Put	Clean	Heat	Cool	Examine	Put two	ALL
Testing LLM: GPT-3.5-turbo								
ReAct [33]	12	75.0	24.7	37.7	36.4	44.4	11.8	41.9
Reflexion [16]	12	87.5	44.1	73.9	50.0	61.1	35.3	59.8
ExpeL [35]	12	62.5	61.3	30.4	61.9	55.5	35.3	52.2
AdaPlanner [21]	6	83.3	46.2	65.2	74.2	68.5	52.9	63.3
Planner+Lib.	1	77.8	**88.2**	82.6	72.7	37.0	27.5	66.5
AutoManual	1	**95.8**	79.6	**87.0**	**78.8**	**100.0**	**66.7**	**86.2**
Testing LLM: GPT-4-turbo								
ReAct [33]	12	95.8	76.3	69.6	86.4	72.2	52.9	76.8
Reflexion [16]	12	**100.0**	95.7	78.3	86.4	77.8	70.6	85.9
ExpeL [35]	12	94.4	82.8	72.4	81.8	72.2	58.8	79.2
AdaPlanner [21]	6	88.9	90.3	85.5	75.8	64.8	41.2	76.4
Planner+Lib.	1	**100.0**	93.5	**100.0**	93.9	88.9	39.2	88.1
AutoManual	1	**100.0**	**98.9**	**100.0**	**95.4**	**100.0**	**90.2**	**97.4**

Experiment: Results

Noticeably, AutoManual requires little expert prior knowledge about the environment and **is only provided with one human example to achieve excellent results**.

Success rate (\%) of LLM agent methods on **ALFWorld** test tasks (6 task types)

Methods	Examples	Put	Clean	Heat	Cool	Examine	Put two	ALL
Testing LLM: GPT-3.5-turbo								
ReAct [33]	12	75.0	24.7	37.7	36.4	44.4	11.8	41.9
Reflexion [16]	12	87.5	44.1	73.9	50.0	61.1	35.3	59.8
ExpeL [35]	12	62.5	61.3	30.4	61.9	55.5	35.3	52.2
AdaPlanner [21]	6	83.3	46.2	65.2	74.2	68.5	52.9	63.3
Planner+Lib.	1	77.8	**88.2**	82.6	72.7	37.0	27.5	66.5
AutoManual	1	**95.8**	79.6	**87.0**	**78.8**	**100.0**	**66.7**	**86.2**
Testing LLM: GPT-4-turbo								
ReAct [33]	12	95.8	76.3	69.6	86.4	72.2	52.9	76.8
Reflexion [16]	12	**100.0**	95.7	78.3	86.4	77.8	70.6	85.9
ExpeL [35]	12	94.4	82.8	72.4	81.8	72.2	58.8	79.2
AdaPlanner [21]	6	88.9	90.3	85.5	75.8	64.8	41.2	76.4
Planner+Lib.	1	**100.0**	93.5	**100.0**	93.9	88.9	39.2	88.1
AutoManual	1	**100.0**	**98.9**	**100.0**	**95.4**	**100.0**	**90.2**	**97.4**

Experiment: Results

The same results can be concluded for web environments: **AutoManual is only provided with one human example to achieve excellent results**.

Success rate (\%) of LLM agent methods on **MiniWoB++** tasks

Methods	Examples	With feedback (9 types)	Examples	ALL (53 types)
Testing LLM: GPT-3.5-turbo				
RCI [33]	22	45.6	104	77.3
AdaPlanner [21]	13	71.6	38	89.4
Planner+Lib.	1	63.6	4	87.0
AutoManual	1	**82.2**	4	**92.7**
Testing LLM: GPT-4-turbo				
RCI [33]	22	60.4	104	88.6
AdaPlanner [21]	13	74.1	38	90.3
Planner+Lib.	1	80.2	4	94.4
AutoManual	1	**94.5**	4	**98.3**

WebArena (Reddit) tasks

Methods	Examples	Suc(%)
ReAct [37]	2	6.0
AutoGuide [2]	19	43.7
SteP [19]	14	55.0
Planner	1	51.1
AutoManual	1	**65.1**

Experiment: Analysis

The generated Markdown manual is also **friendly for human-reading**.

Housekeeper Agent Interaction Manual

Overview

This manual is intended to assist the housekeeper agent in the successful execution of tasks within a simulated environment. The rules provide guidance on navigating, searching the environment, interacting with objects, and managing task-specific processes, as well as ensuring the correctness of actions using code assertions.

Navigation and Search

Introduction

These rules provide guidance on how to search for objects, including the use of helper methods to streamline the process and ensure thoroughness.

Included Rules

- **rule_0 (type="Special Mechanism")**: Objects can be found in unconventional locations, and the agent should **include all possible locations in its search**. For example, In epoch_9, the agent found a soapbar on the toilet, which is an unconventional location for storing such items.
- **rule_1 (type="Useful Helper Method")**: If there are multiple receptacles to be search, the agent can write and use 'find_object' method as shown in the example. For example,

```
# Define helper method to find object that is needed
def find_object(agent, recep_to_check, object_name):
  for receptacle in recep_to_check:
      observation = agent.go_to(receptacle)
      # Check if we need to open the receptacle. If we do, open it.
      if 'closed' in observation:
          observation = agent.open(receptacle)
      # Check if the object is in/on the receptacle.
      if object_name in observation:
          object_ids = get_object_with_id(observation, object_name)
          return object_ids, receptacle
  return None, None

# Use assertions to validate each step
assert object_ids is not None, "Error: Could not find the object."
```

Object Interaction and Location Management

Introduction

These rules inform the agent on how to interact with objects, from taking and placing items to handling multiple items of the same type. Proper location management is crucial for successful task execution.

Included Rules

- **rule_2 (type="Special Phenomena")**: When using a microwave, the agent can interact with it (e.g., heat an object) even if there is another object inside, the agent is holding something, and the microwave door is not explicitly mentioned to be open.

  For example, In epoch_1, the agent was able to heat the mug with the microwave even though there was an egg inside the microwave and the agent was holding the mug.
- **rule_3 (type="Special Mechanism")**: The agent can only hold one object at a time and must put down any held object before taking another.

  For example, In epoch_2, the agent was holding statue_4 and attempted to take statue_3 without putting down statue_4 first, resulting in a 'Nothing happens' observation.
- **rule_4 (type="Success Process")**: When tasked with placing multiple objects in/on a receptacle, the agent can either collect all objects before attempting to place them or find and place them one by one, ensuring they revisit locations with multiple objects if necessary. If all objects are found at the same location, handle them sequentially according to rule_3.

  For example, In epoch_15, the agent should have revisited sidetable_1 to collect the second pencil before attempting to place it in coffeetable_1. In epoch_23, the agent failed to collect all required statues from coffeetable_1 because it did not revisit, is also addressed by this rule.
- **rule_5 (type="Special Mechanism")**: The agent must interact with a receptacle to observe its contents, which includes going to the receptacle and opening it if it is closed. Before performing a put or take action, the agent must ensure it is at the correct location. When multiple items of the same type are present at a location, the agent may have to choose one to interact with or examine.

  For example, In epoch_16, the agent had to open several closed cabinets (e.g., cabinet_1, cabinet_2) to find items such as the mug. In epoch_21, the agent observed multiple alarm clocks on desk_1 and selected one ('alarmclock_4') to interact with.

Task-Specific Processes

Introduction

This category outlines the steps required to complete specific tasks, such as heating, cooling, and examining objects with another object's assistance.

Included Rules

- **rule_6 (type="Success Process")**: If the task involves cooling or heating an object before placing it, the steps are: (1) search for the object using 'find_object' in rule_1, (2) take the object, (3) cool/heat it as required, (4) go to the target receptacle, and (5) put the object. Ensure the agent's location and the state of the environment are updated after each action. For example,

```
# For example, to cool a mug and put it in a coffeemachine:
# [Step 1] Use 'find_object' method to search all receptacles
# [Step 2] Take the mug
# [Step 3] Go to the fridge, open it if necessary, and cool the mug
# [Step 4] Go to the coffeemachine and put the cooled mug in it
```

- **rule_7 (type="Success Process")**: When tasked with examining an object under a desklamp, the agent should first find the desklamp and the object, ensure the desklamp is on, take the object, and then use the desklamp to examine the object. For example,

```
# [Step 1] Use 'find_object' method to search for the desklamp and the object
# [Step 2] Make sure the desklamp is on
# [Step 3] Take the object
# [Step 4] Use the desklamp to examine the object.
```

- **rule_8 (type="Success Process")**: When tasked to look at an object under a desklamp, ensure the lamp is on before using it to examine the object. For example,

```
# [Step 4] Go to the desklamp's location and turn it on if it's not already on
observation = agent.go_to(receptacle_with_desklamp)
observation = agent.use(found_desklamp)
assert 'turn on' in observation or 'already on' in observation, 'Error in [Step 4]: Failed to use the desklamp.'
# [Step 5] Similarly, search for the alarm clock and take it.
# [Step 6] With the desklamp on, examine the alarm clock using the desklamp.
```

Correctness and Validation

Introduction

Instructions on asserting code to confirm state changes and enhance the reliability of the agent's actions.

Included Rules

- **rule_9 (type="Corrected Error")**: Assertions in the agent's code should confirm state changes such as location or held objects, rather than rely on specific phrases in observations.

  For example, Instead of asserting 'You are at' in the observation, the agent should assert the location and held object state changes. Also, when handling multiple required objects at the same location, the agent should manage them sequentially without unnecessary variables.

Experiment: Analysis

AutoManual resolves the Path Dependency problem of skills by **digging deeper into mechanisms**, **updating and incorporating success processes**, and **annotating important details**.

Object Interaction and Location Management

Introduction

These rules inform the agent on how to interact with objects, from taking and placing items to handling multiple items of the same type. Proper location management is crucial for successful task execution.

Included Rules

- **rule_2 (type="Special Phenomena")**: When using a microwave, the agent can interact with it (e.g., heat an object) even if there is another object inside, the agent is holding something, and the microwave door is not explicitly mentioned to be open.

  For example, In epoch_1, the agent was able to heat the mug with the microwave even though there was an egg inside the microwave and the agent was holding the mug.

- **rule_3 (type="Special Mechanism")**: The agent can only hold one object at a time and must put down any held object before taking another.

  For example, In epoch_2, the agent was holding statue_4 and attempted to take statue_3 without putting down statue_4 first, resulting in a 'Nothing happens' observation.

- **rule_4 (type="Success Process")**: When tasked with placing multiple objects in/on a receptacle, the agent can either collect all objects before attempting to place them or find and place them one by one, ensuring they revisit locations with multiple objects if necessary. If all objects are found at the same location, handle them sequentially according to rule_3.

  For example, In epoch_15, the agent should have revisited sidetable_1 to collect the second pencil before attempting to place it in coffeetable_1. In epoch_23, the agent failed to collect all required statues from coffeetable_1 because it did not revisit, is also addressed by this rule.

Thanks for Watching!