Reflexion: Language Agents with Verbal Reinforcement Learning

NeurIPS 2023

Citation: 7455

Introduction

Reflexion, a framework to reinforce language agents not by updating weights, but instead through linguistic feedback. It uses verbal reinforcement to help agents learn from prior failings.

Main Contributions

- Introduces an approach where agents improve using written reflections stored in memory, without updating LLM weights.
- Demonstrates that reflection can improve performance over a few attempts in decision-making, reasoning, and programming.
- Introduces LeetcodeHardGym: 40 difficult coding problems with support for 19 programming languages.
- Outperforms strong comparison methods on several tasks and achieves state-of-the-art results on some coding benchmarks at publication.

Advantages

- **No fine-tuning:** Improves behavior while keeping LLM weights fixed.
- **Specific feedback:** Written reflections suggest targeted corrections.
- **Readable memory:** Stores understandable lessons from previous attempts.
- **Guided retries:** Uses saved lessons to improve decisions in later attempts.

Limitations

- **Depends on evaluation quality:** Incorrect judgments or reflections can mislead the agent.
- **No guaranteed success:** Repeated reflection may not solve the task.
- **Outlook:** More capable LLMs may improve evaluation and reflection quality.

datasets

HotpotQA: A Dataset Example

Tests whether an AI can combine information from multiple Wikipedia passages to answer one question.

Question

"Were Scott Derrickson and Ed Wood of the same nationality?"

Reference answer

Yes. Both are American.

Context: 10 passages

Two passages provide the necessary evidence:

- **Scott Derrickson:** He is American.
- **Ed Wood:** He was American.
- The other eight passages are distractors.

Supporting-fact labels

Identify the evidence sentences: the first sentence of the Scott Derrickson passage and the first sentence of the Ed Wood passage.

Additional information

Question type: comparison · Difficulty: hard · Unique record ID

This example uses HotpotQA's distractor setting. During evaluation, the reference answer and evidence labels are withheld from the model.

ALFWorld: Interactive Household Tasks

- A collection of text-based simulated household environments.
- Each task provides a goal and an initial description of the surroundings.
- The agent sends actions and receives observations from the environment.
- Tasks require several steps, such as finding, moving, cleaning, or cooling objects.

Illustrative example

Goal: Put a mug on the countertop.

Action: Open cabinet 1.

Observation: “Inside, you see mug 1.”

Next steps: Pick up the mug, go to the countertop, and place it there.

Chain-of-Thought (CoT): Think before you answer

Standard prompting: Question → Answer

CoT prompting: Question → step-by-step reasoning → Answer

- The LLM writes out its reasoning as text before committing to an answer.
- One single generation — no tools, no environment, no retries

ReAct: Reasoning + Acting in a loop

Thought → Action → Observation → repeat

- Interleaves reasoning with real actions in an environment.
- Grounded: gathers evidence from the world instead of hallucinating it

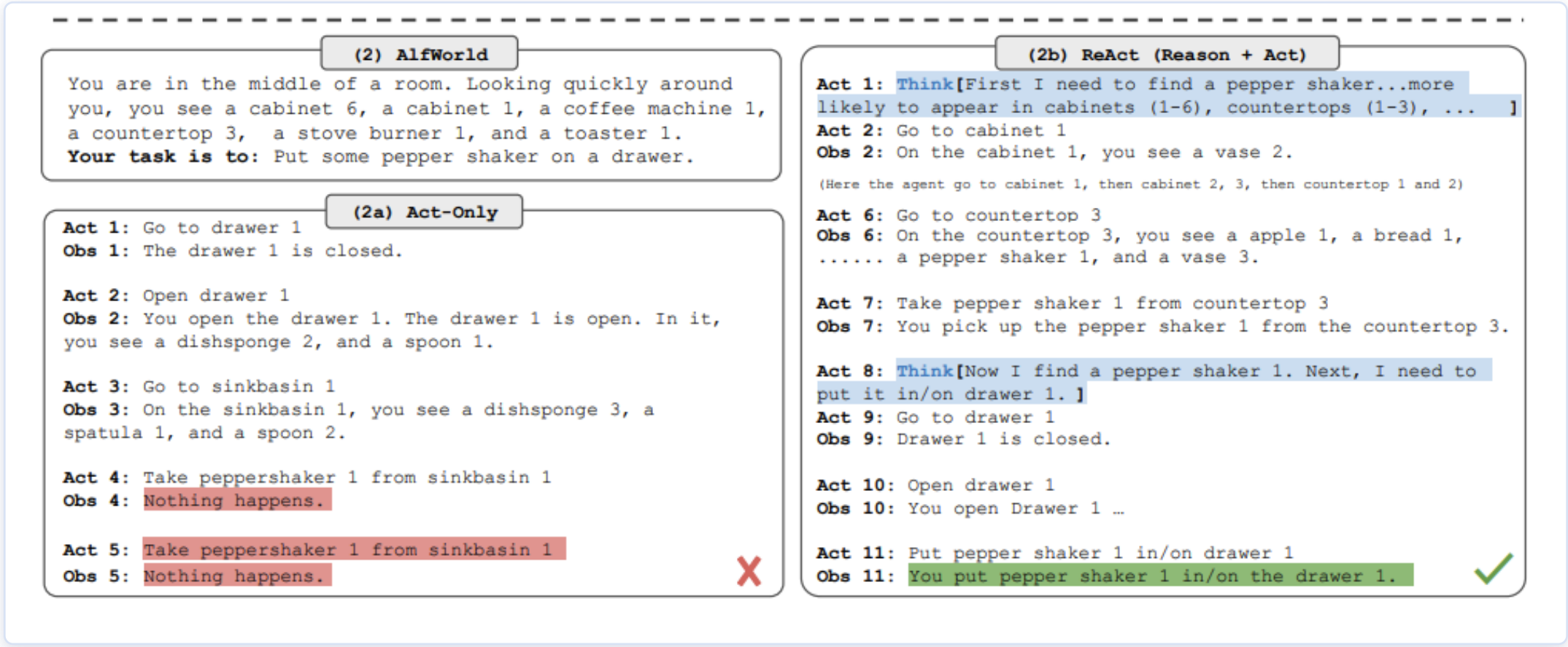
The problem: these agents never learn

CoT can reason. ReAct can act. Neither can learn from failure.

ALFWorld on react

ReAct: Thought → Action → Environment's response → repeat.

Act-only: Action → Environment's response → repeat



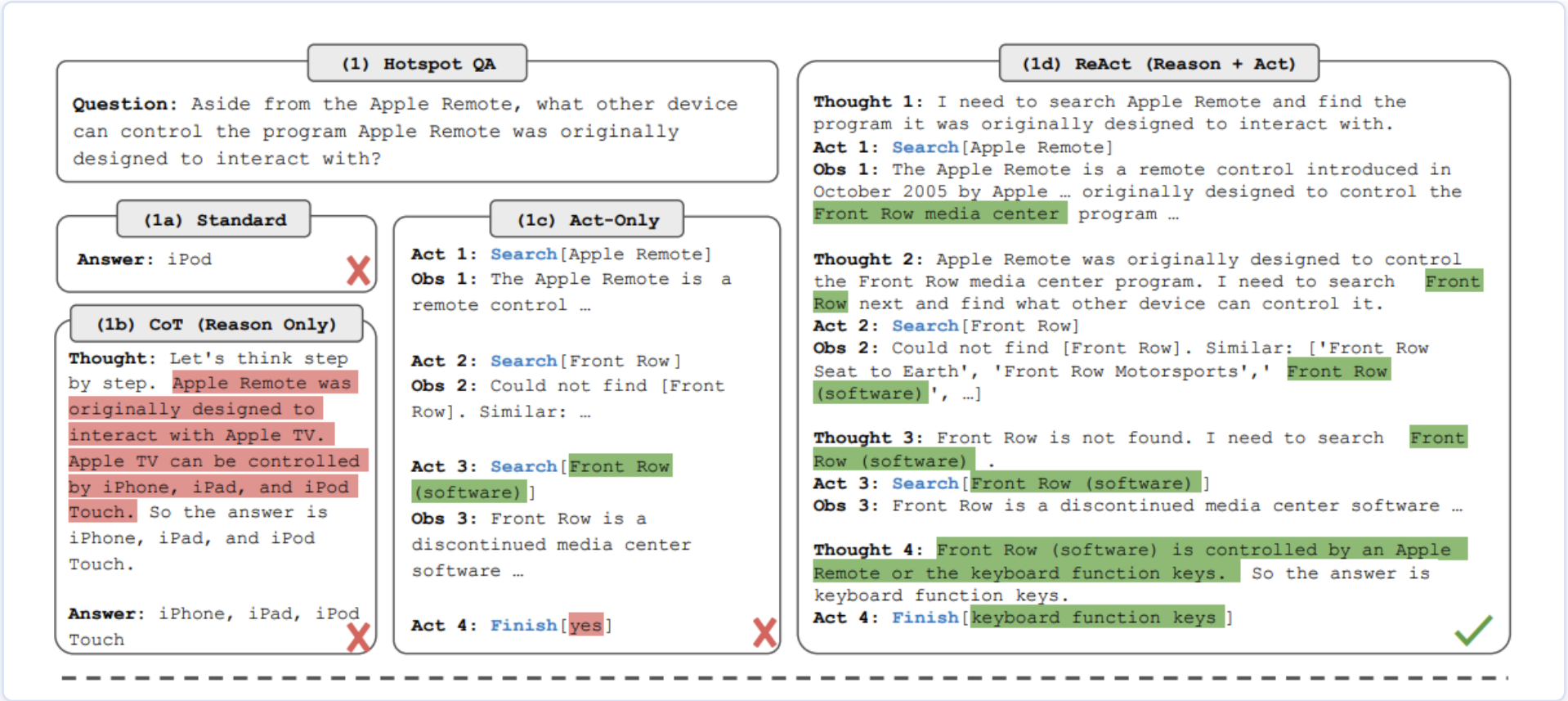
HotpotQA on react

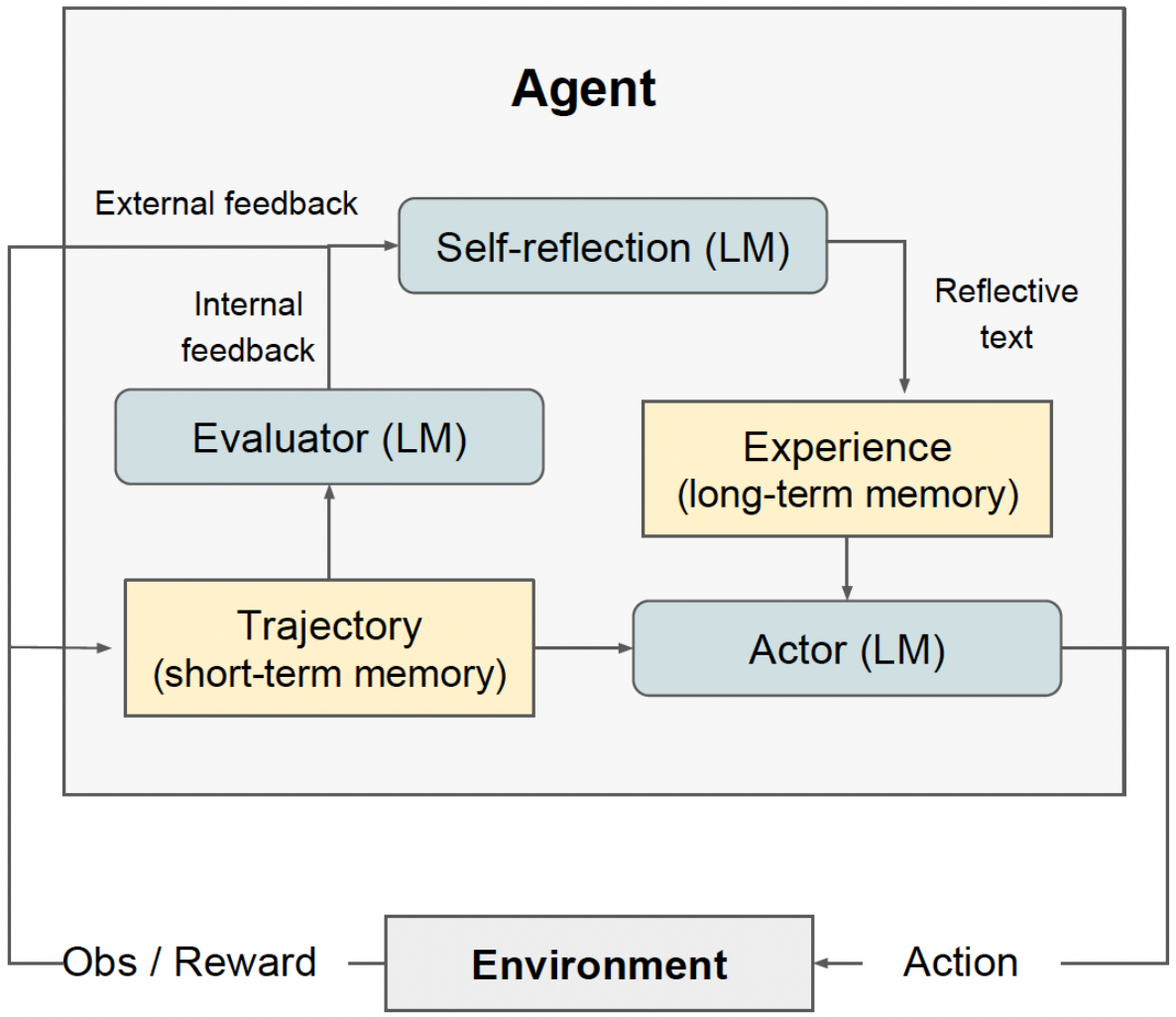
Standard: Question → answer.

CoT: Question → written reasoning → answer.

Act-only: Question → search → observation → more actions → answer.

ReAct: Question → thought → search → observation → further thoughts/actions → answer





Actor: Uses an LLM to generate answers or actions based on observations and memory. Can use CoT or ReAct.

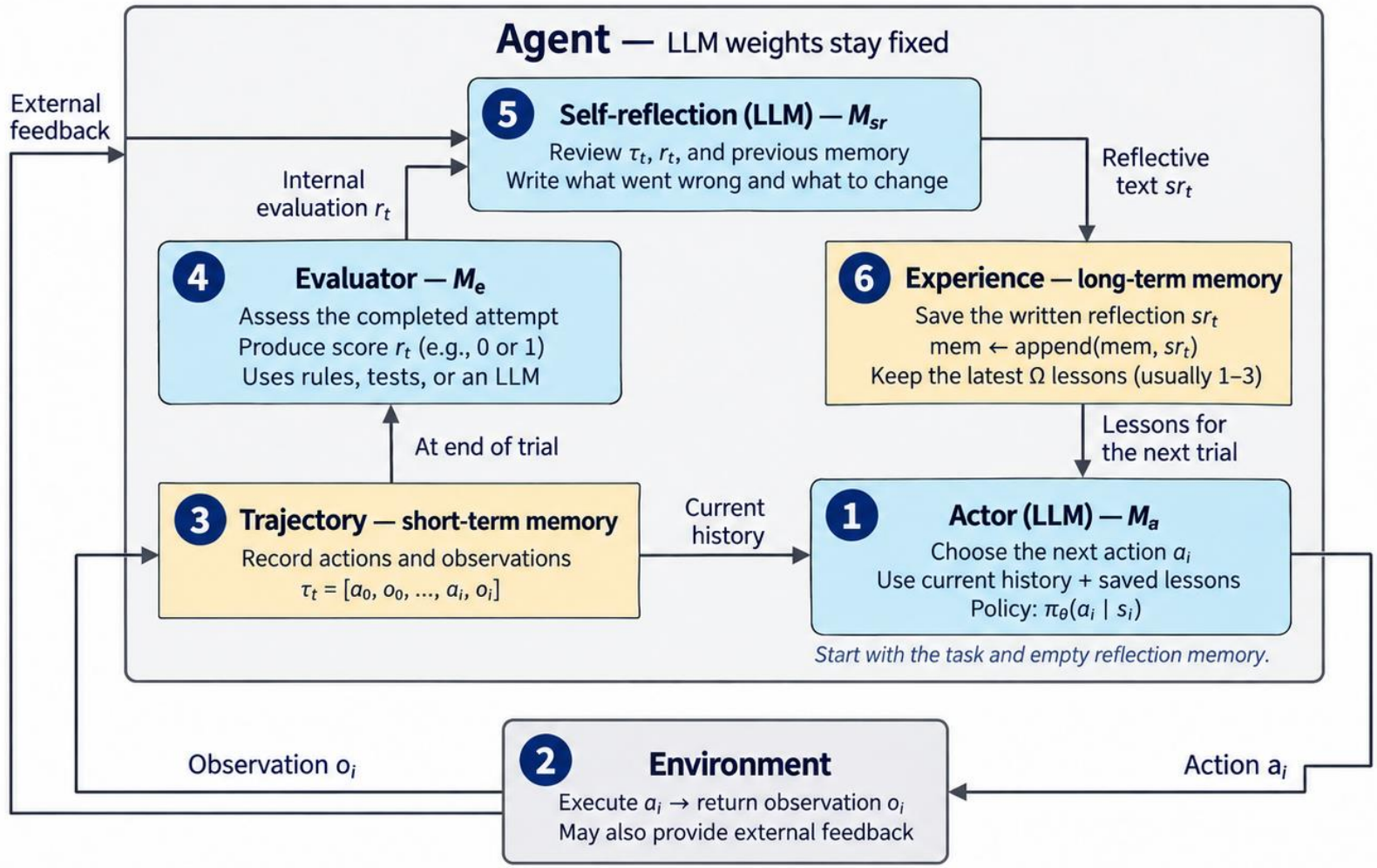
Evaluator: Assesses the attempt and provides a performance score using answer comparisons, predefined rules, or LLM evaluation.

Self-Reflection: Reviews the attempt and feedback to identify likely mistakes and write specific lessons for improvement.

Memory: Stores the current attempt's actions and observations in short-term memory, and lessons from previous attempts in long-term memory.

Reflexion: the process

Actions within a trial • Reflection between trials



Within a trial: 1 → 2 → 3 → 1
After an unsuccessful trial: 3 → 4 → 5 → 6 → 1
Stop when successful or when the trial limit is reached.

Notation: i = action step • t = trial
s_i = current state/context • $\theta = \{M_a, \text{mem}\}$

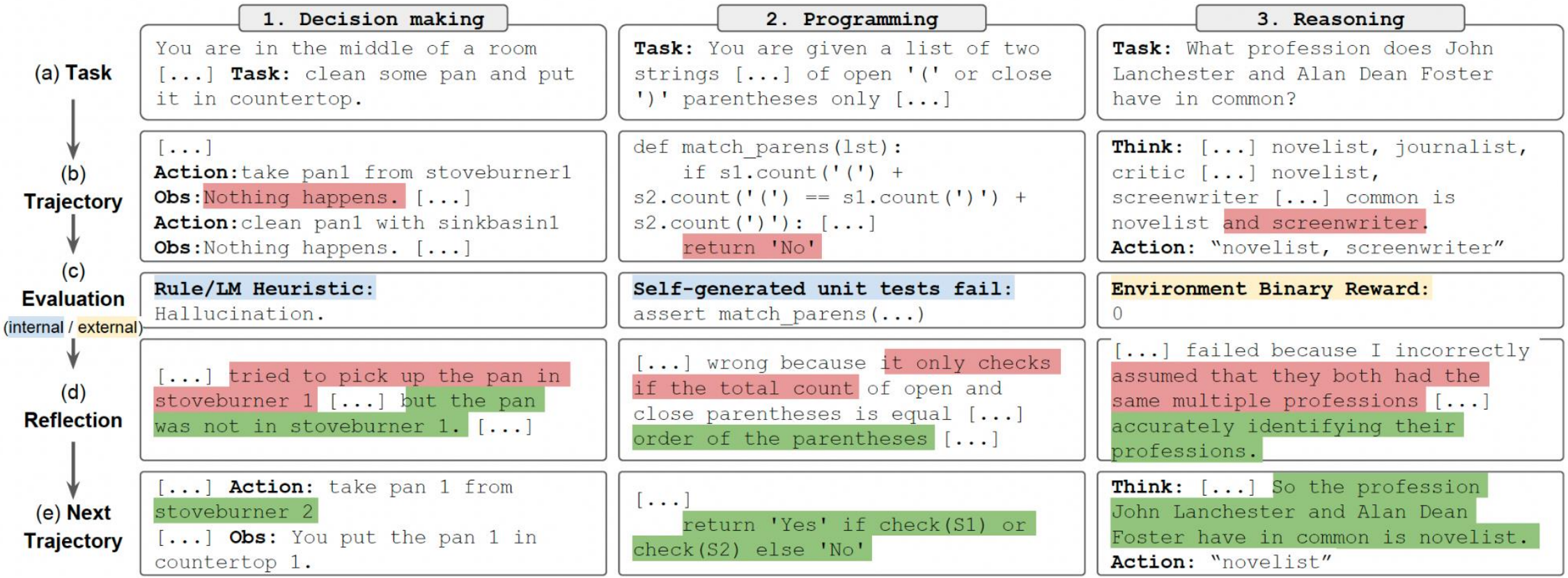


Figure 1: Reflexion works on decision-making 4.1, programming 4.3, and reasoning 4.2 tasks.

Experiment

Decision-making

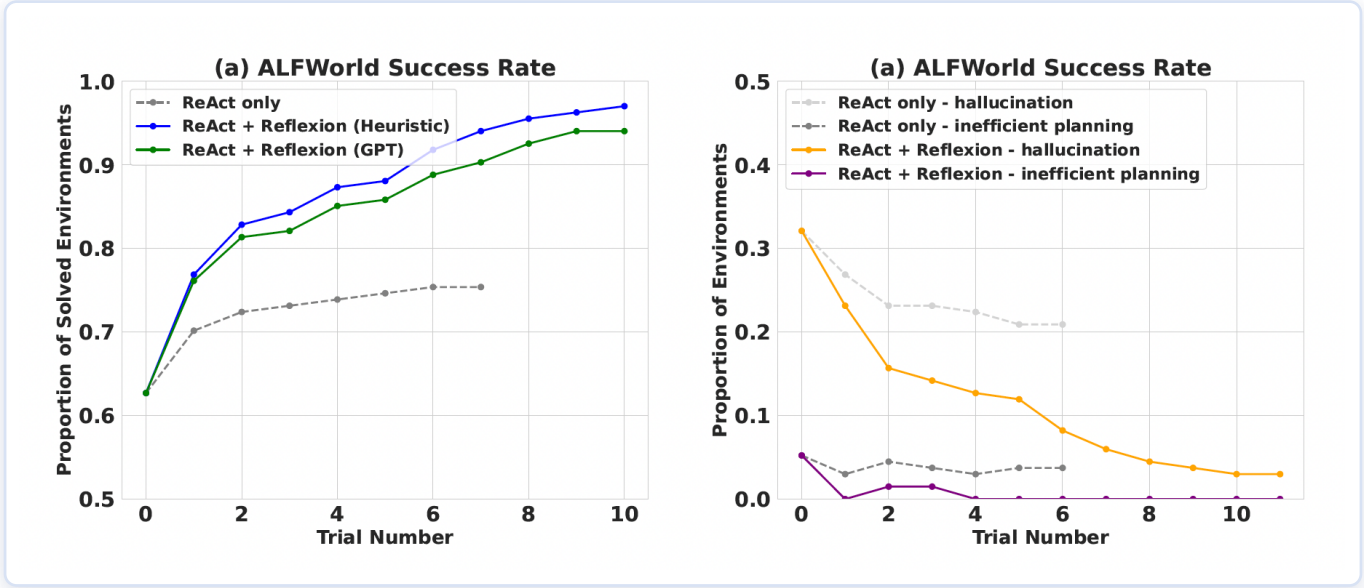
Reasoning

programming.

4.1 Sequential Decision-Making: ALFWorld

- **Environment:** Text-based household tasks requiring multiple actions, such as finding, moving, or cleaning objects.
- **Setup:** 134 environments across six task types, using ReAct as the Actor.
- **Comparison:** ReAct retries versus ReAct + Reflexion, which saves lessons from failed attempts.
- **Evaluation:** The environment signals success; rules or an LLM detect unsuccessful attempts. Reflexion retains the latest three reflections.

Key result: Reflexion with heuristic evaluation solves 130/134 tasks (~97%) across repeated trials, outperforming ReAct alone.

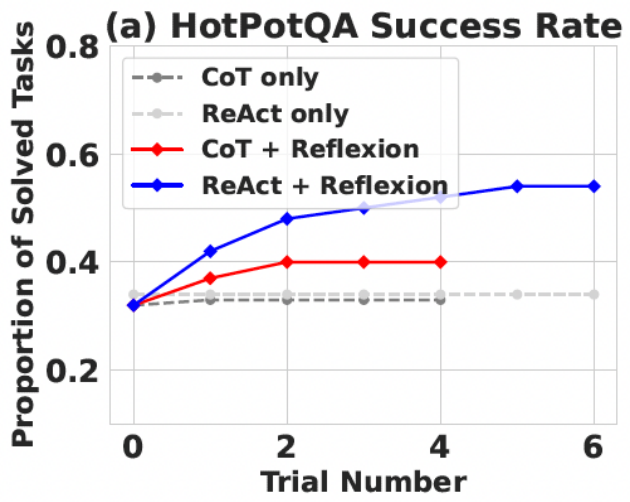


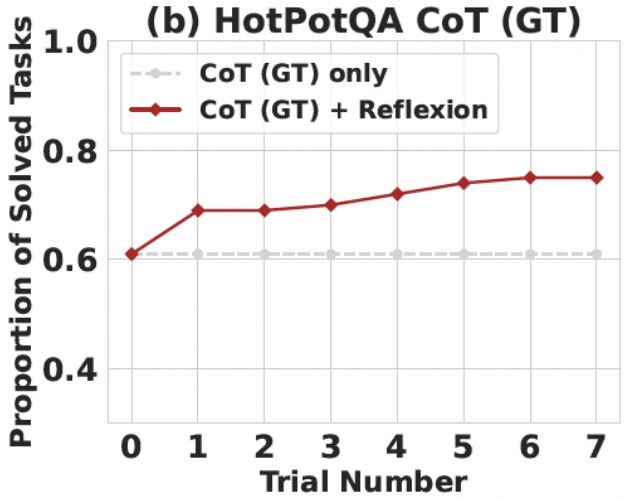
4.2 Reasoning: HotpotQA

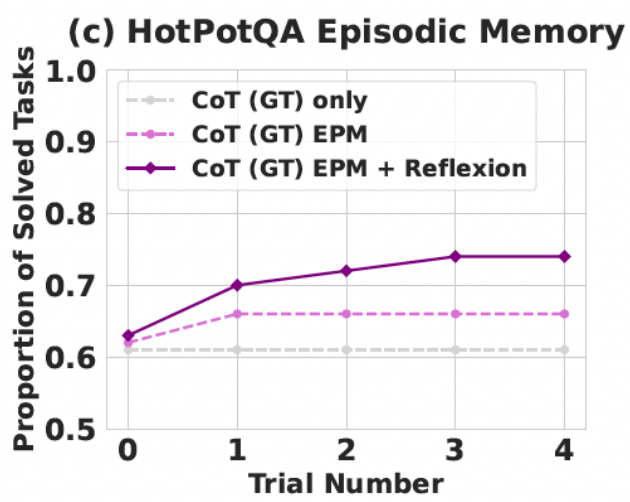
- **Task:** Answer questions requiring evidence from multiple Wikipedia documents; evaluated on 100 questions.
- **Approaches:** CoT, CoT with provided supporting context (GT), and ReAct with Wikipedia search.
- **Evaluation:** Exact-match grading provides success/failure feedback.
- **Memory:** Retain the latest three reflections to guide retries.

Results — Figure 4(a–b): Reflexion improves cumulative success across trials: ReAct ≈32% → 54%; CoT (GT) ≈61% → 75%.

Memory comparison — Figure 4(c): Adding reflection to episodic memory improves success from 66% to 74%.







4.3 Programming

- **Task:** Generate working code from natural-language problem descriptions.
- **Benchmarks:** HumanEval, MBPP, and LeetcodeHardGym, with experiments in Python and Rust.
- **Evaluation during retries:** Generate and execute up to six unit tests.
- **Reflection:** Use execution feedback to revise the code, retaining the latest reflection.

- **HumanEval:** Write functions from descriptions; evaluate with tests.
- **MBPP:** Solve basic programming problems.
- **LeetcodeHardGym:** Solve 40 difficult LeetCode problems.

Main result: HumanEval Python improves from 80.1% to 91%.

Limitation: Improvements are not universal—MBPP Python decreases from 80.1% to 77.1%.

Benchmark + Language	Prev SOTA Pass@1	SOTA Pass@1	Reflexion Pass@1
HumanEval (PY)	65.8 (CodeT [5] + GPT-3.5)	80.1 (GPT-4)	**91.0**
HumanEval (RS)	–	60.0 (GPT-4)	**68.0**
MBPP (PY)	67.7 (CodeT [5] + Codex [6])	**80.1** (GPT-4)	77.1
MBPP (RS)	–	70.9 (GPT-4)	**75.4**
Leetcode Hard (PY)	–	7.5 (GPT-4)	**15.0**

Thank you

Questions?