

AI AGENTS | LECTURE 18

Multi-agent Systems

Interaction 1 · October 29, 2026

Carnegie Mellon University
Language Technologies Institute

Saujas Vaduguru
Carnegie Mellon University

What are multi-agent systems?

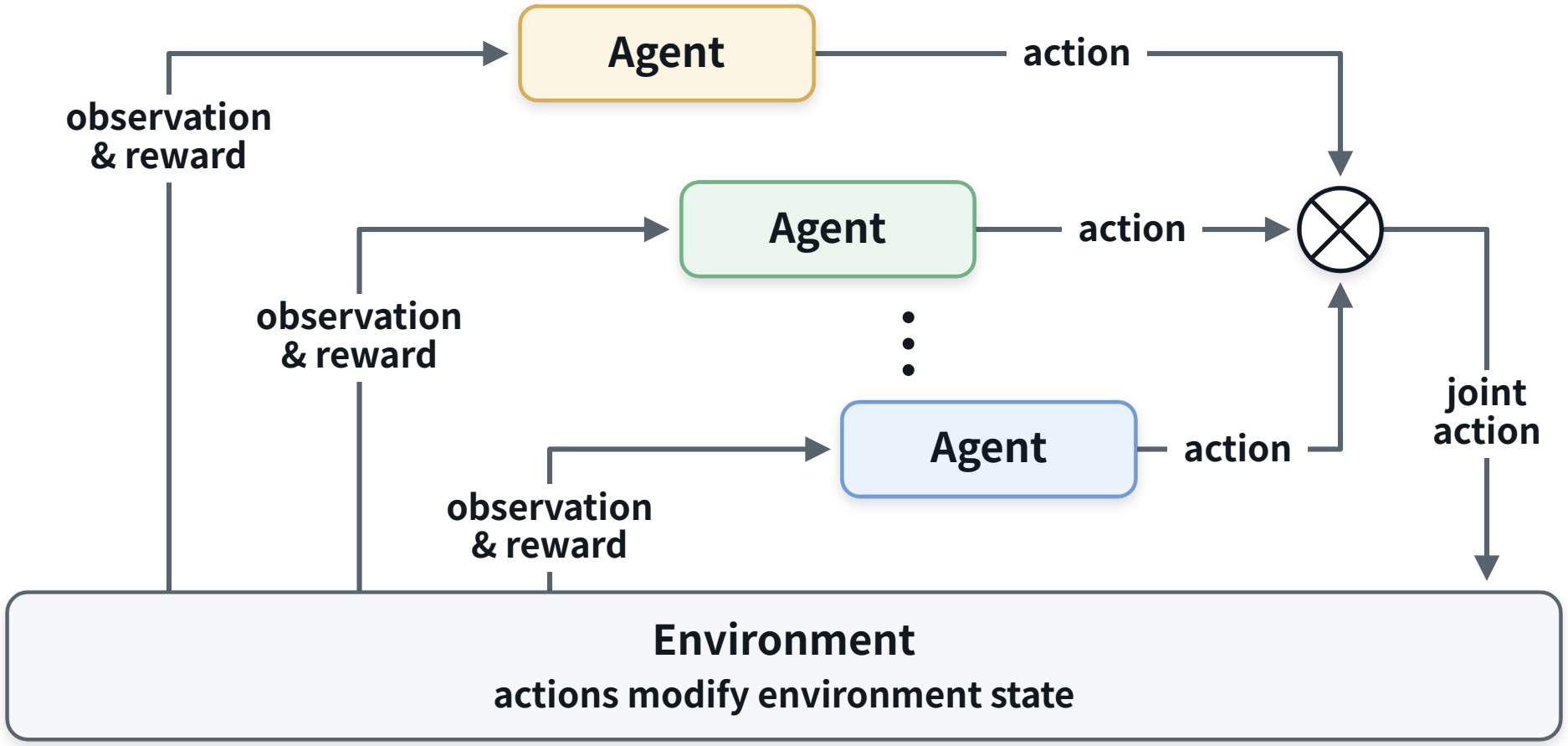


Figure adapted from Albrecht, Christianos & Schäfer, Multi-Agent Reinforcement Learning (MIT Press, 2024) ↗

Why multi-agent systems?

Improved test-time compute

Agents as abstractions that allow for more effective use of resources

Interacting with other entities

Agents that work on inherently multi-agent problems

Training and evaluation environments for agents

Using other agents to train for and evaluate single agent behavior

Multi-agent systems for test-time compute

A single agent typically works in a single stream

Context management

The entire interaction builds up in context

Serial execution

All tasks have to be performed in a sequence, even if they don't have any dependencies

Lack of specialization

The same LLM has to perform all parts of the task

Limited exploration

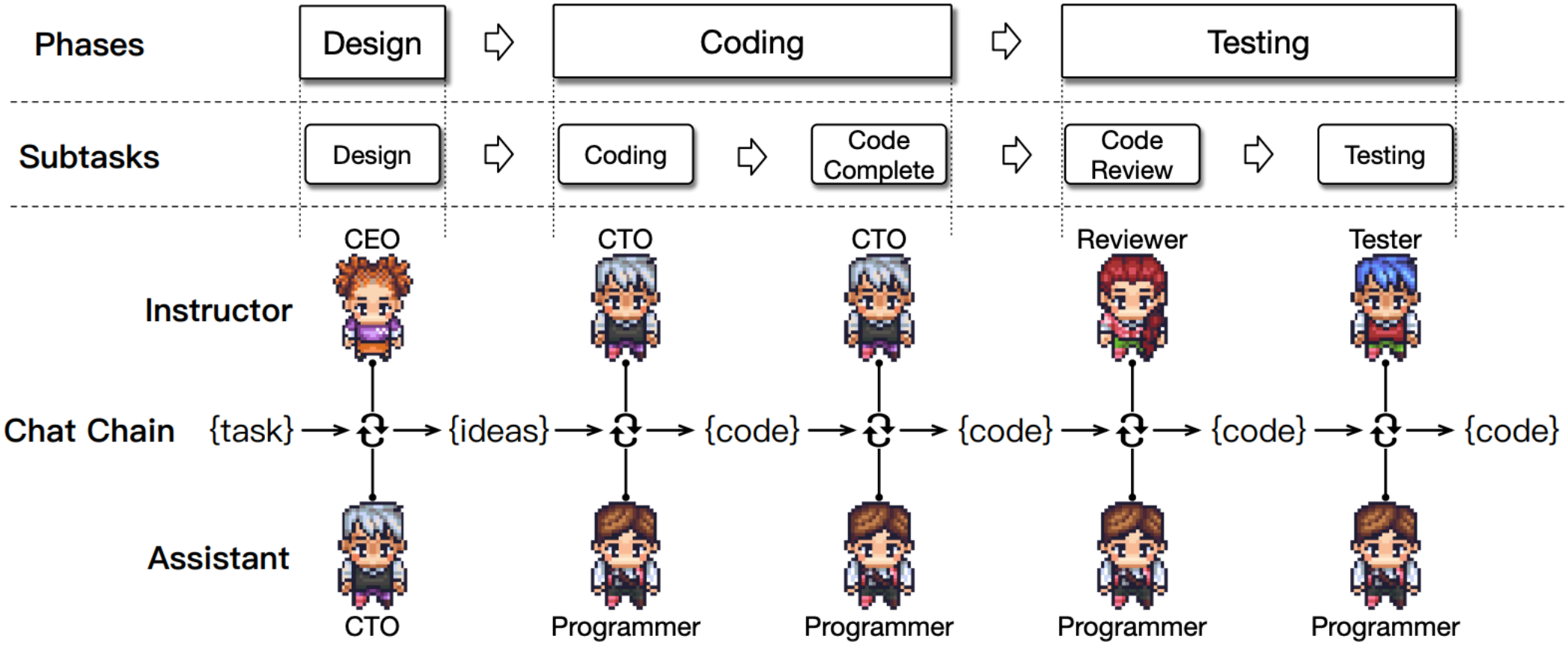
Agents don't compare multiple approaches and get locked in to one solution

Role-based decomposition

- **Roles.** An agent is created by an LLM prompted to play a specific role, typically based on a role a person plays in a team of humans
- **Interaction.** Agents in a system interact with other agents playing different roles
- **Orchestration.** Agents are typically orchestrated with a fixed workflow
- **Communication.** Agents may interact through messages passed to a subset of agents on a board, sent only to an orchestrator, or through artifacts

In-Context Impersonation · Salewski et al., NeurIPS 2023 ↗

Virtual software companies



ChatDev · Qian et al., ACL 2024 ↗

Implementing a multi-agent workflow

```python
calendar_agent = create_agent(
    model, tools=[create_calendar_event, get_available_time_slots],
    system_prompt="You are a calendar scheduling assistant. ..."
)

email_agent = create_agent(model, tools=[send_email], system_prompt="You are an email assistant. ...")

@tool
def schedule_event(request: str) -> str:
    return calendar_agent.invoke({"messages": [{"role": "user", "content": request}]})["messages"][-1].text

@tool
def manage_email(request: str) -> str:
    return email_agent.invoke({"messages": [{"role": "user", "content": request}]})["messages"][-1].text

supervisor_agent = create_agent(
    model, tools=[schedule_event, manage_email],
    system_prompt="You are a helpful personal assistant. You can schedule calendar events and send emails. ..."
)
```

LangChain docs · Subagents: personal assistant ↗ Lecture 15: Orchestration and Monitoring (slides) ↗

Open-ended delegation

Pre-defined roles and workflows are limiting

- Sub-problems may not fall into neat categories that map to human roles
- The number of agents needed for different tasks may be different

Open-ended delegation uses generic agents, but coordinates among multiple instances

- Depending on the specific method, the number may also be spawned as needed

Multi-agent debate

User Input: A treasure hunter found a buried treasure chest filled with gems. There were 175 diamonds, 35 fewer rubies than diamonds, and twice the number of emeralds than the rubies. How many of the gems were there in the chest?

Round 1

Agent 1: Let x be the number of rubies. Then the number of diamonds is 175. And the number of⋯ ✗

Agent 2: If there were 175 diamonds, then the number of rubies is 35 fewer, or 175 - 35 = 140⋯ ✓

User Input: Using the solutions from other agents as additional information, can you give an updated response....

(Copy Agent 2's reply) **(Copy Agent 1's reply)**

Round 2

Agent 1: Given the information provided in other agents' solutions, we have two answers: One age⋯ ✗

Agent 2: After reviewing the solutions provided by other agents, I agree with the second agent who⋯ ✓

User Input: Using the solutions from other agents as additional information, can you give an updated response....

(Copy Agent 2's reply) **(Copy Agent 1's reply)**

Round 3

Agent 1: After reviewing the solutions provided by other agents, I agree with the second agent who⋯ ✓

Agent 2: After reviewing the solutions provided by other agents, I also agree that the number of rubi⋯ ✓

Figure adapted from Du et al., Multiagent Debate, ICML 2024 ↗

Multi-agent debate results

Model	Arithmetic (%) ↑	Grade School Math (%) ↑	Chess (ΔPS) ↑
Single Agent	67.0 ± 4.7	77.0 ± 4.2	91.4 ± 10.6
Single Agent (Reflection)	72.1 ± 4.5	75.0 ± 4.3	102.1 ± 11.9
Multi-Agent (Majority)	69.0 ± 4.6	81.0 ± 3.9	102.2 ± 6.2
Multi-Agent (Debate)	$\mathbf{81.8 \pm 2.3}$	$\mathbf{85.0 \pm 3.5}$	$\mathbf{122.9 \pm 7.6}$

Du et al., Multiagent Debate, ICML 2024 ↗

Launching subagents on demand

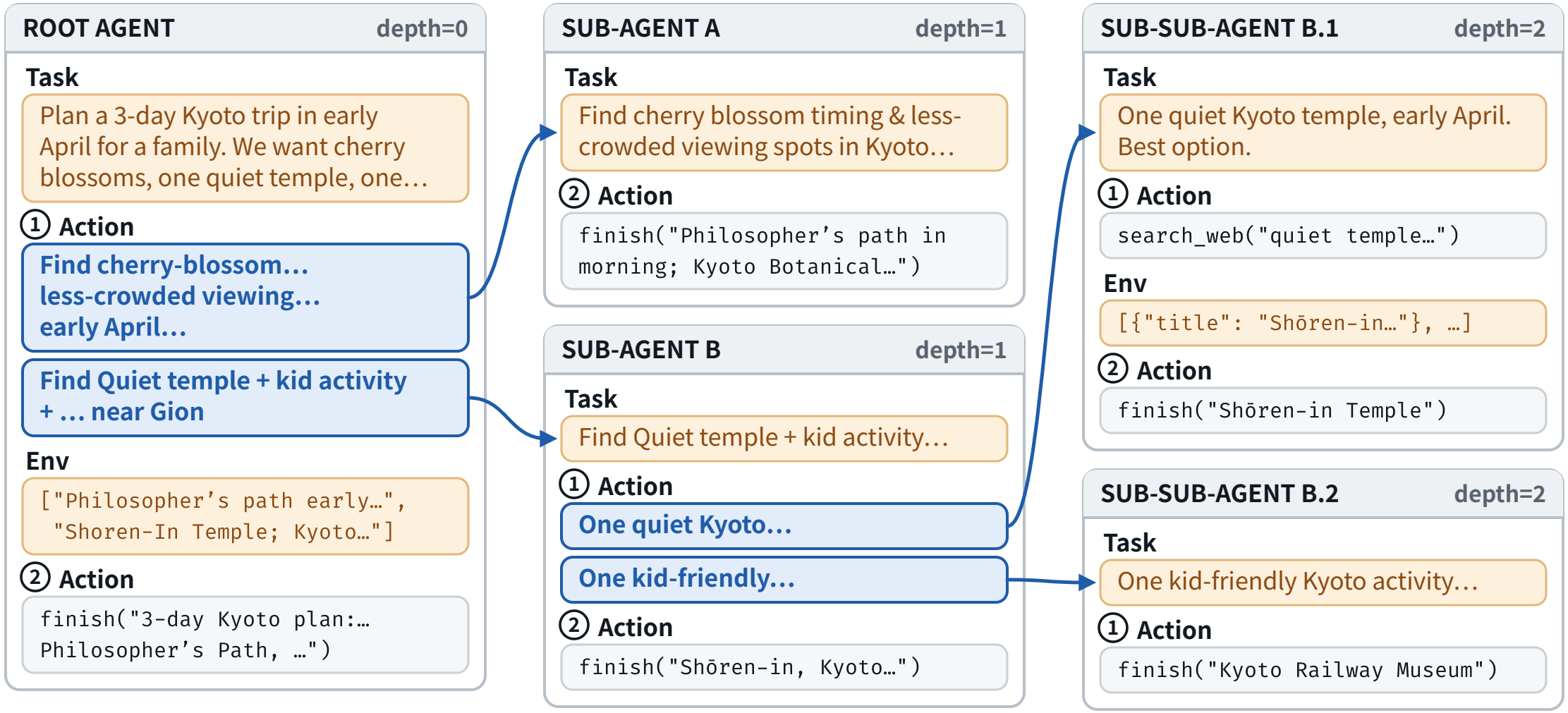


Figure adapted from Gandhi et al., Recursive Agent Optimization, arXiv 2026 ↗

Programmatic primitives for subagent coordination

```
async def launch_subagent(subgoal, ...) → Any:
    ...
```

- Tool that launches an clone of the agent with the argument as its goal
- Run in parallel and `gather` results
- Use `await` to sequence operations
- Use Python's built-in async context management tools to share state

Delegating to subagents uses context effectively

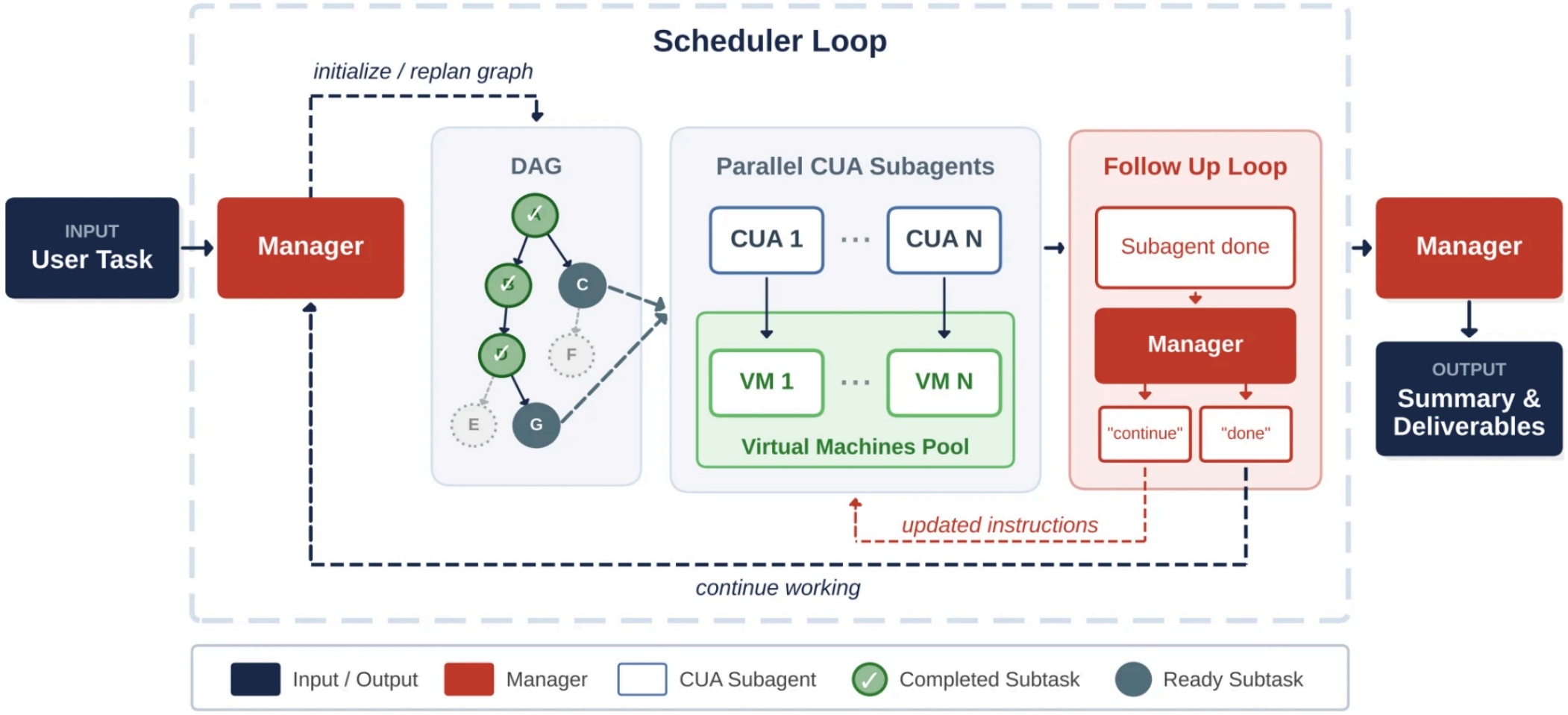
(a) Context Window: 8K train, 8K eval

Difficulty	Method	SR	Steps	Time (s)
All	Single	0.24	16	7.1
	Recursive	**0.95**	33	9.9
Easy	Single	0.55	12	5.2
	Recursive	**1.0**	23	8.0
Medium	Single	0.17	25	11.1
	Recursive	**0.96**	52	13.6
Hard	Single	0.0	–	–
	Recursive	**0.88**	–	–



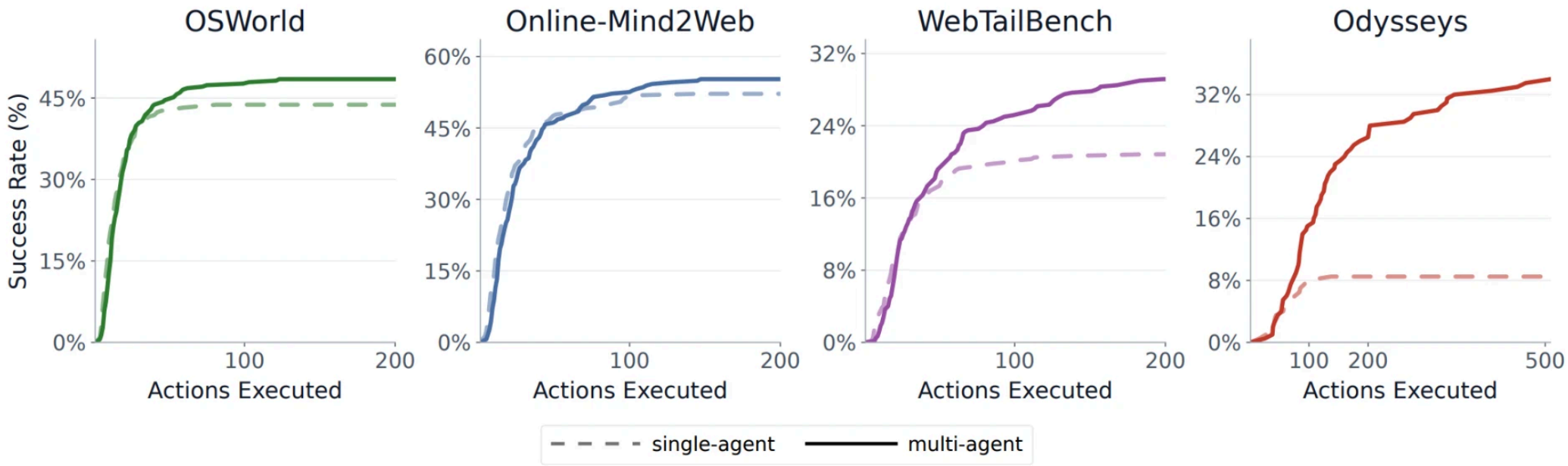
Gandhi et al., Recursive Agent Optimization, arXiv 2026 ↗

Managed delegation to subagents



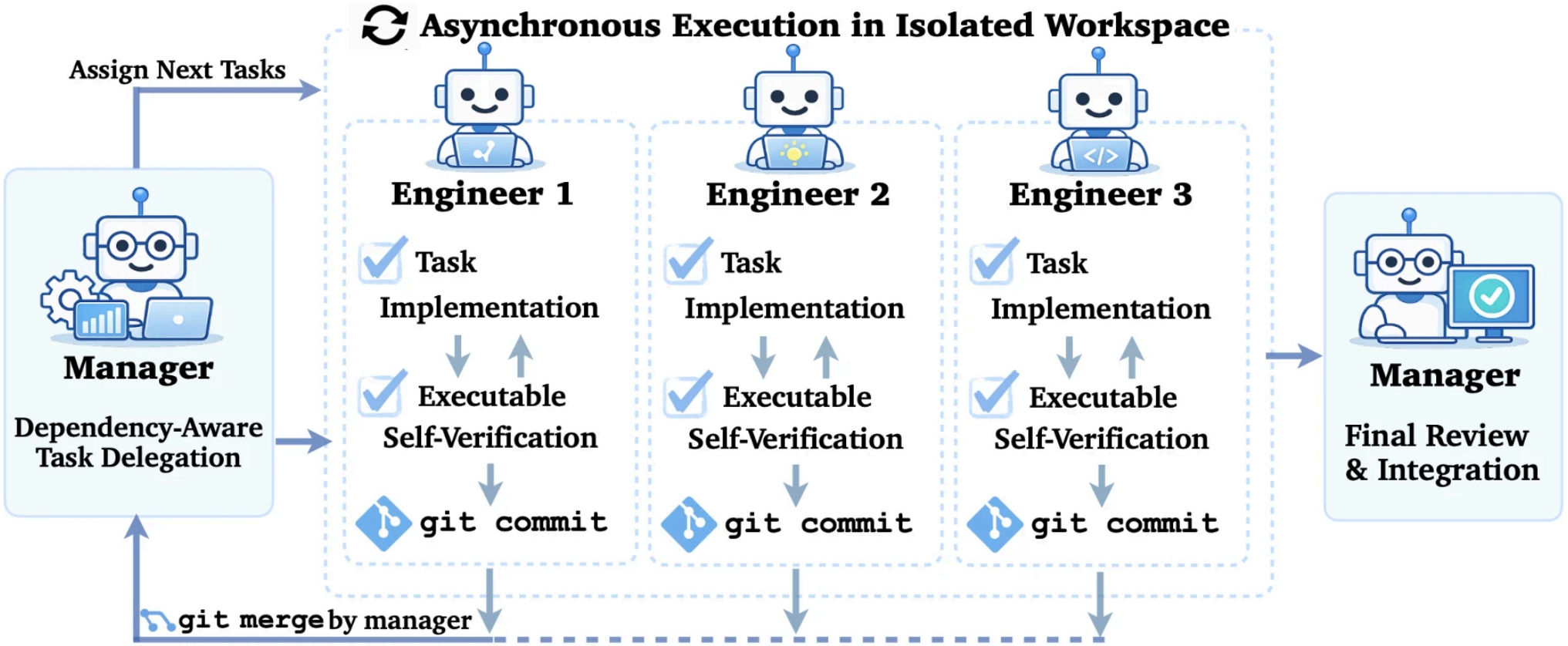
MACU · Koh, Salakhutdinov & Fried, Multi-Agent Computer Use, arXiv 2026 ↗

Delegation improves performance on longer, more parallelizable tasks



MACU · Koh, Salakhutdinov & Fried, Multi-Agent Computer Use, arXiv 2026 ↗

Coordinating asynchronous agents



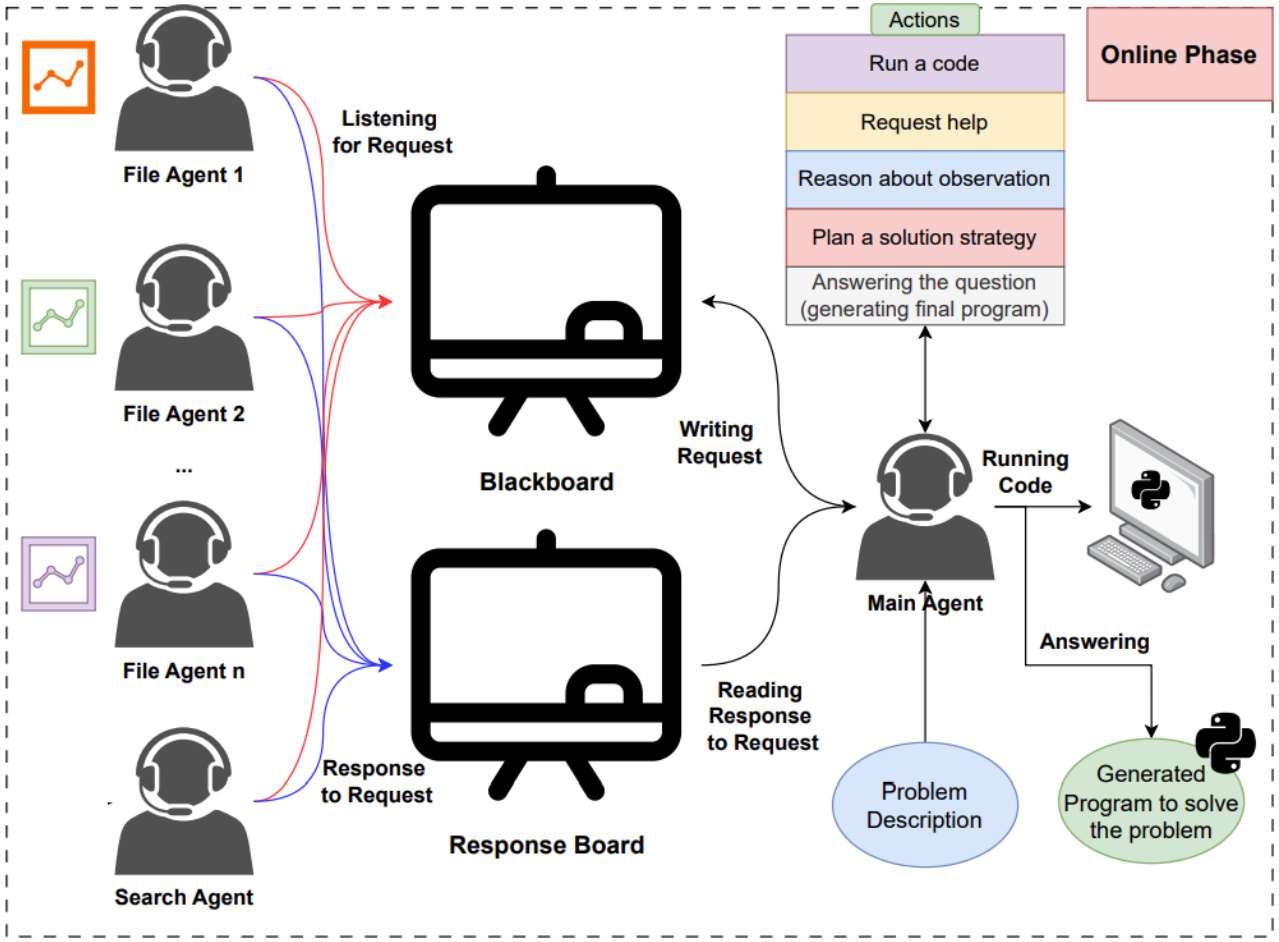
CAID · Geng & Neubig, Effective Strategies for Asynchronous Software Engineering Agents, arXiv 2026 ↗

Coordinating asynchronous agents

- Manager produces a DAG for planning
- Manager communicates using structured JSON messages
- Agents use Git to work in parallel with managed conflicts
  - Repurposing human coordination tools for agent coordination

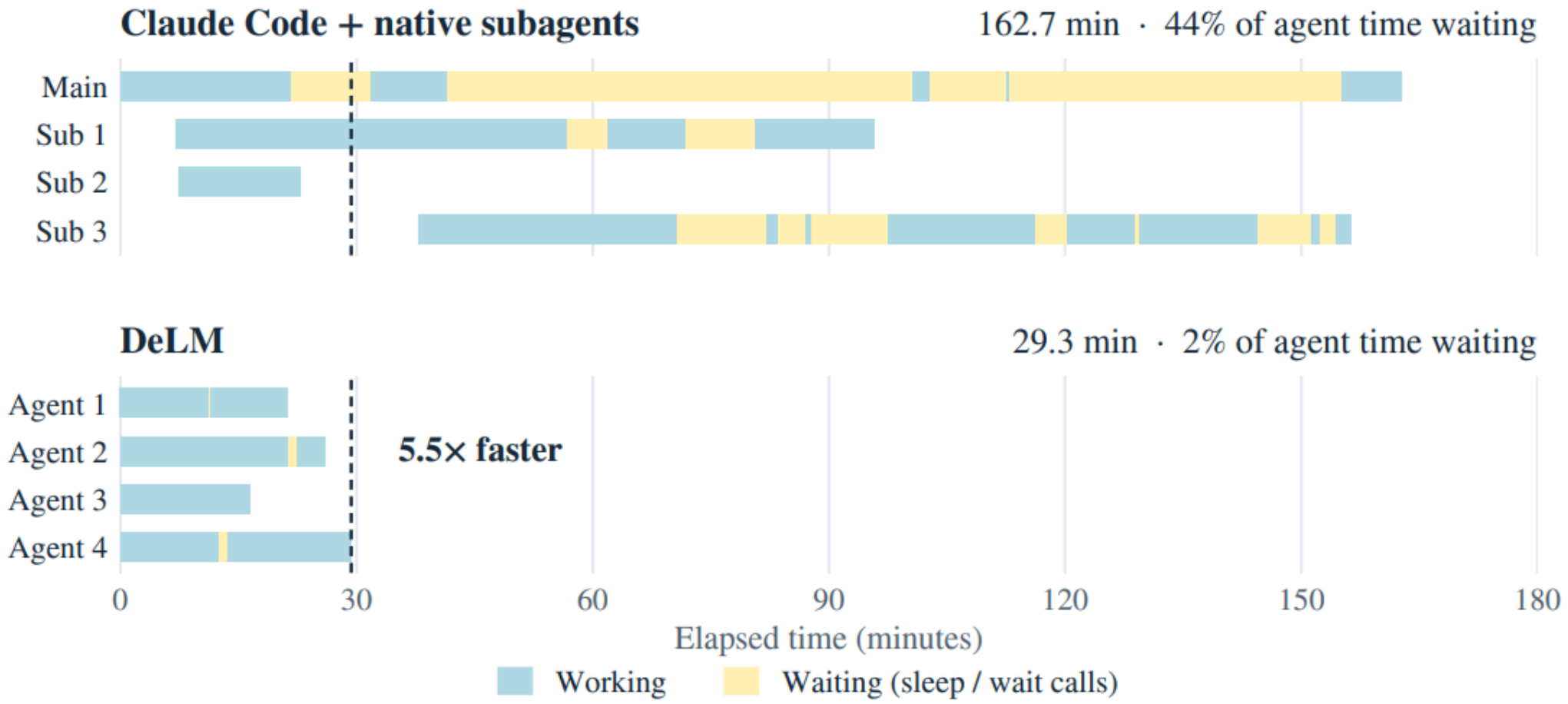
CAID · Geng & Neubig, Effective Strategies for Asynchronous Software Engineering Agents, arXiv 2026 ↗

Blackboard coordination



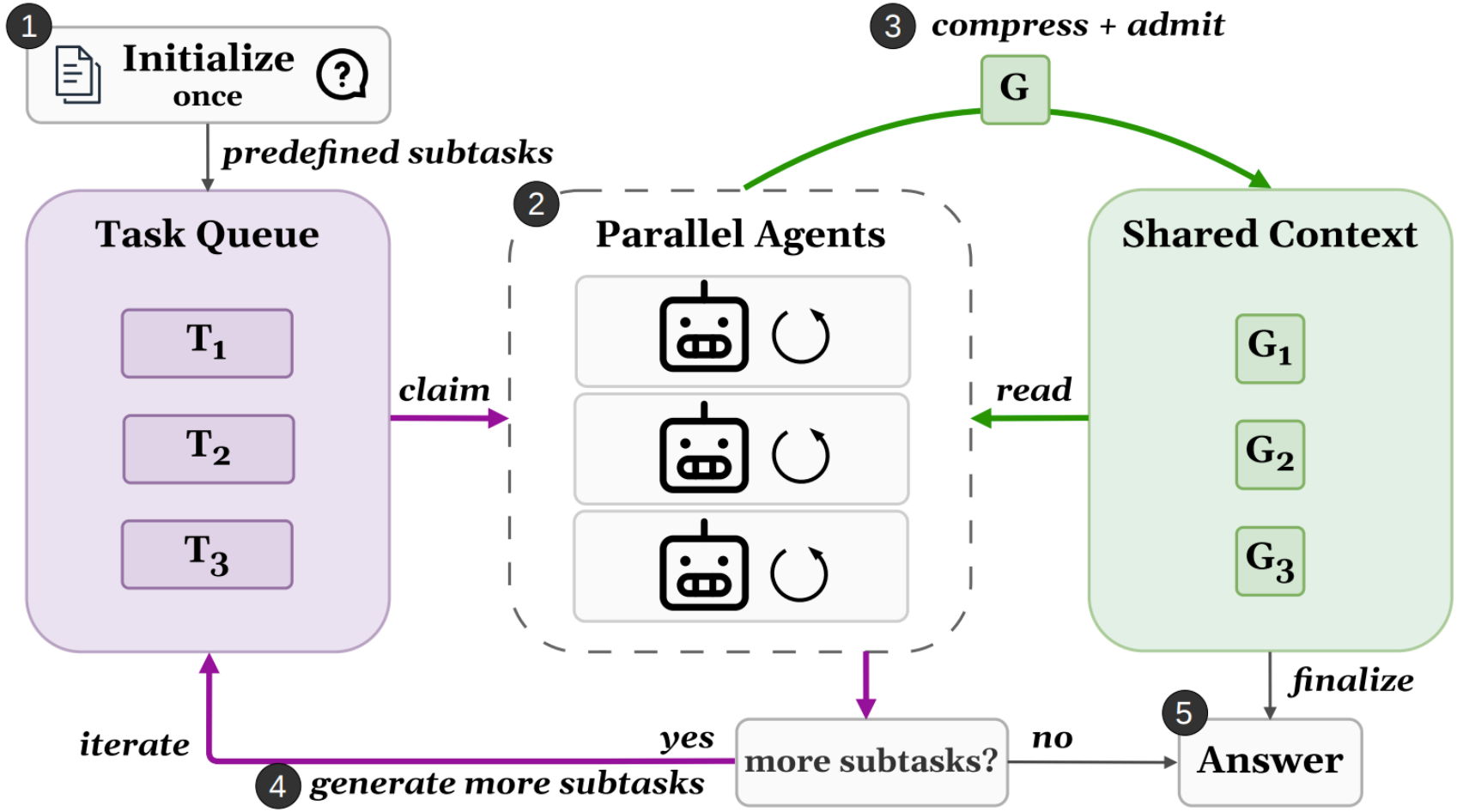
Salemi et al., LLM-Based Multi-Agent Blackboard System for Information Discovery in Data Science, arXiv 2025 ↗

Waiting on subagents



DeLM · Mao et al., Decentralized Multi-Agent Systems with Shared Context, arXiv 2026 ↗

Decentralized, asynchronous action



DeLM · Mao et al., Decentralized Multi-Agent Systems with Shared Context, arXiv 2026 ↗

Decentralized, asynchronous action

Orchestration

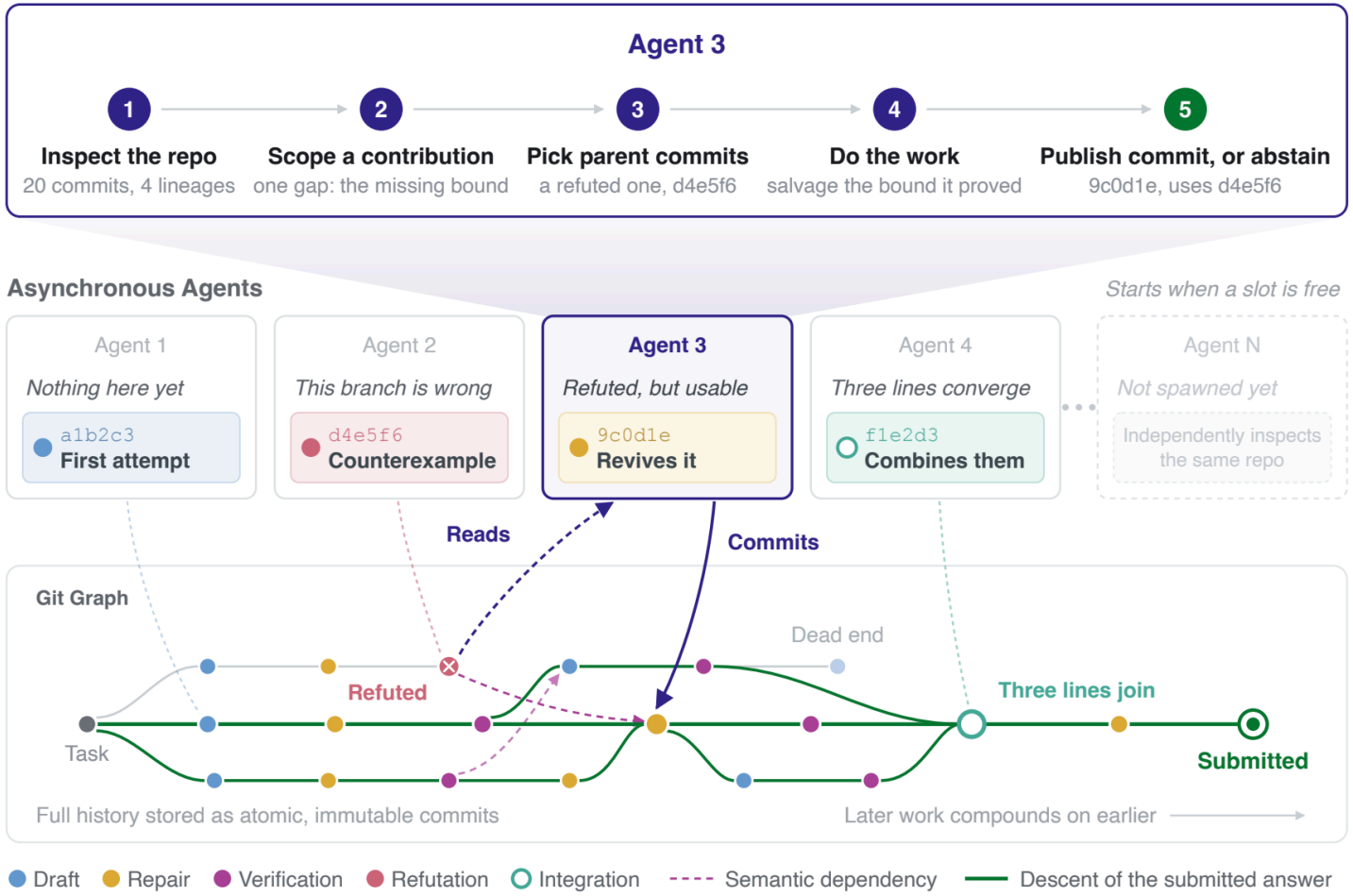
- Task queue is public
- Each task is marked with the agent doing that task

Communication

- Public message board as shared context
- Append-only
- Typed entries that may carry attachments

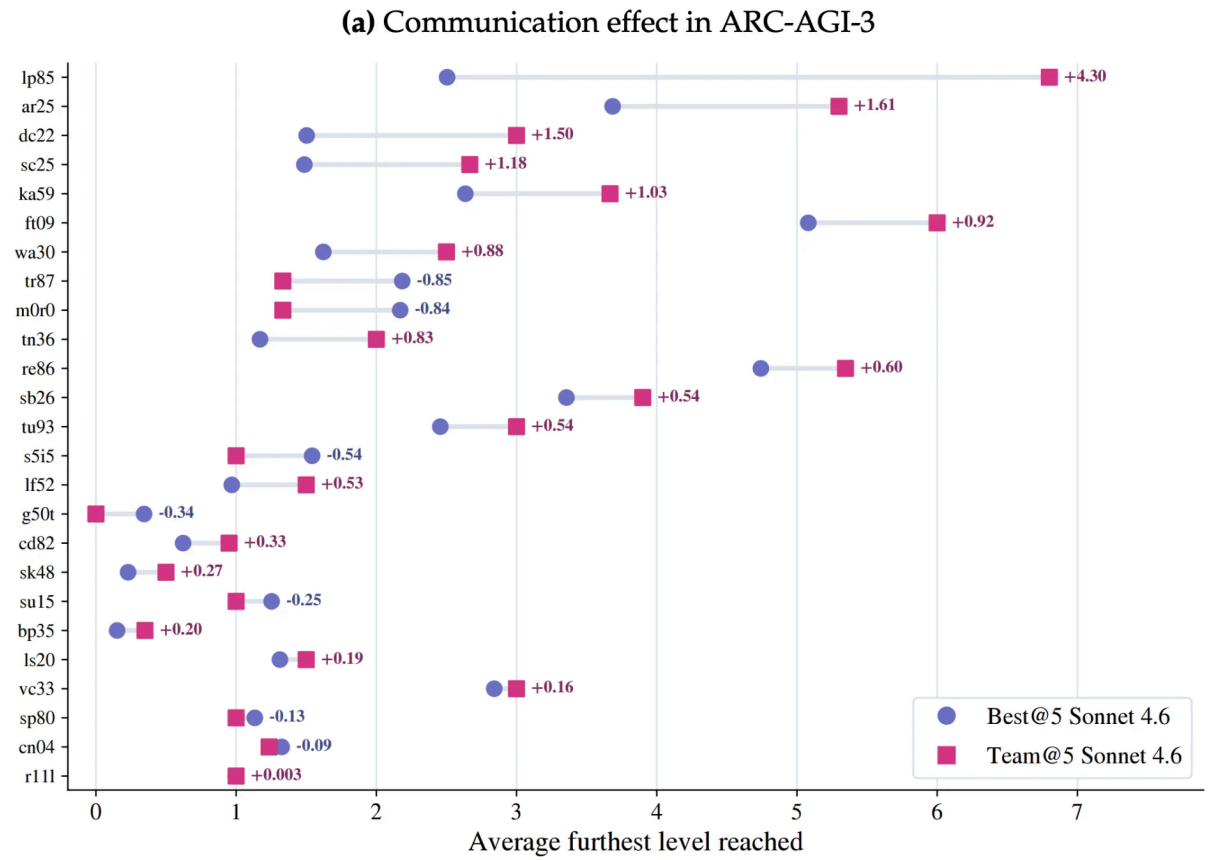
DeLM · Mao et al., Decentralized Multi-Agent Systems with Shared Context, arXiv 2026 ↗

Using Git to coordinate an agent swarm



GitSwarm · Shah et al., Decentralized Compounding Inference, arXiv 2026 ↗

Communicating swarms do better than agents acting alone



Park et al., Scaling Discovery through Test-Time Communication, arXiv 2026 ↗

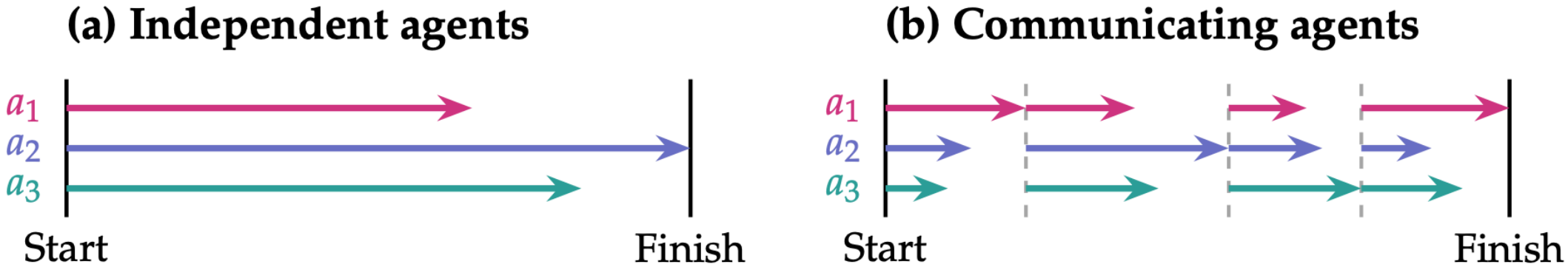
…but not always

Communication does not outperform pass@*k* on Terminal-Bench 2.0.

Method	Mean Accuracy	Max Accuracy
Single agent pass@1	52.53%	–
Independent pass@2	**62.36%**	**64.04%**
Communicating team@2	60.67%	61.80%

Table 3 from Park et al., Scaling Discovery through Test-Time Communication, arXiv 2026 ↗

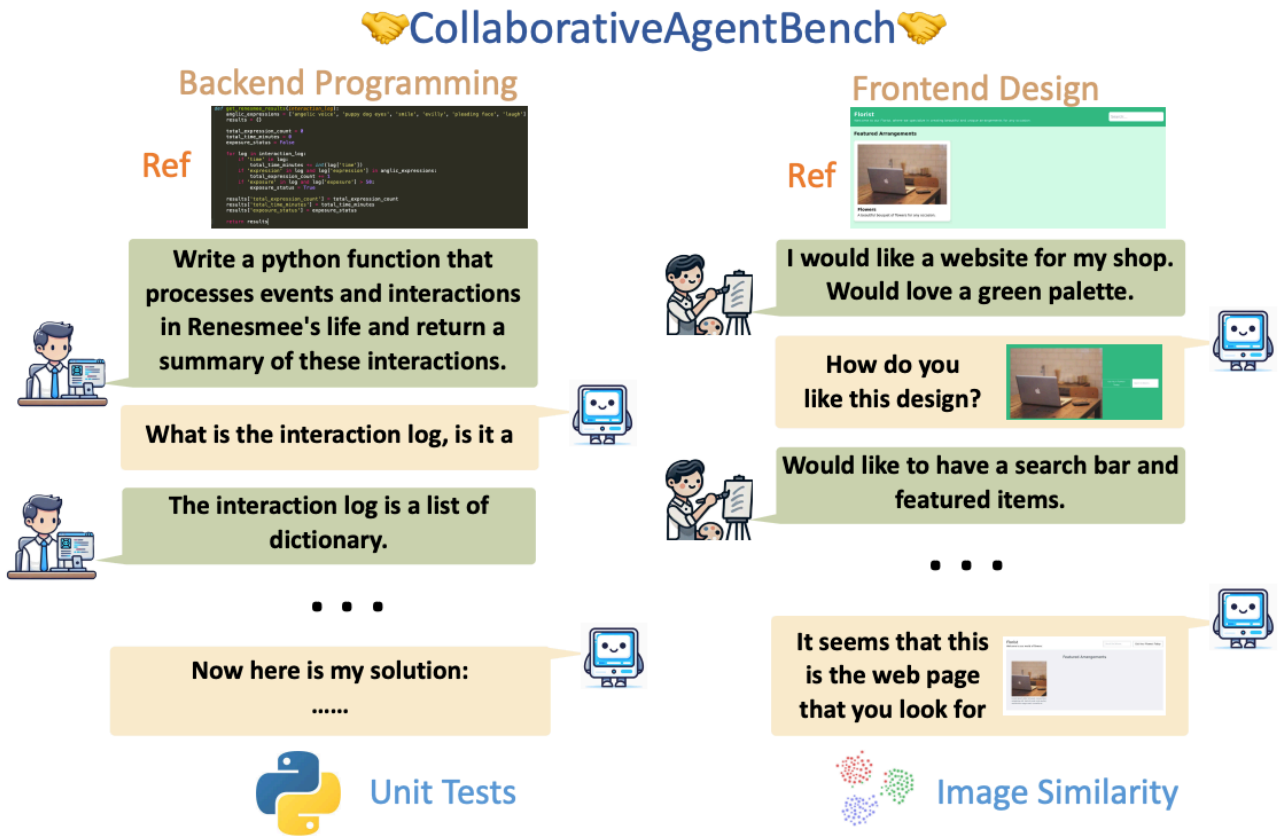
Verified progress sharing allows for effective collaboration



Park et al., Scaling Discovery through Test-Time Communication, arXiv 2026 ↗

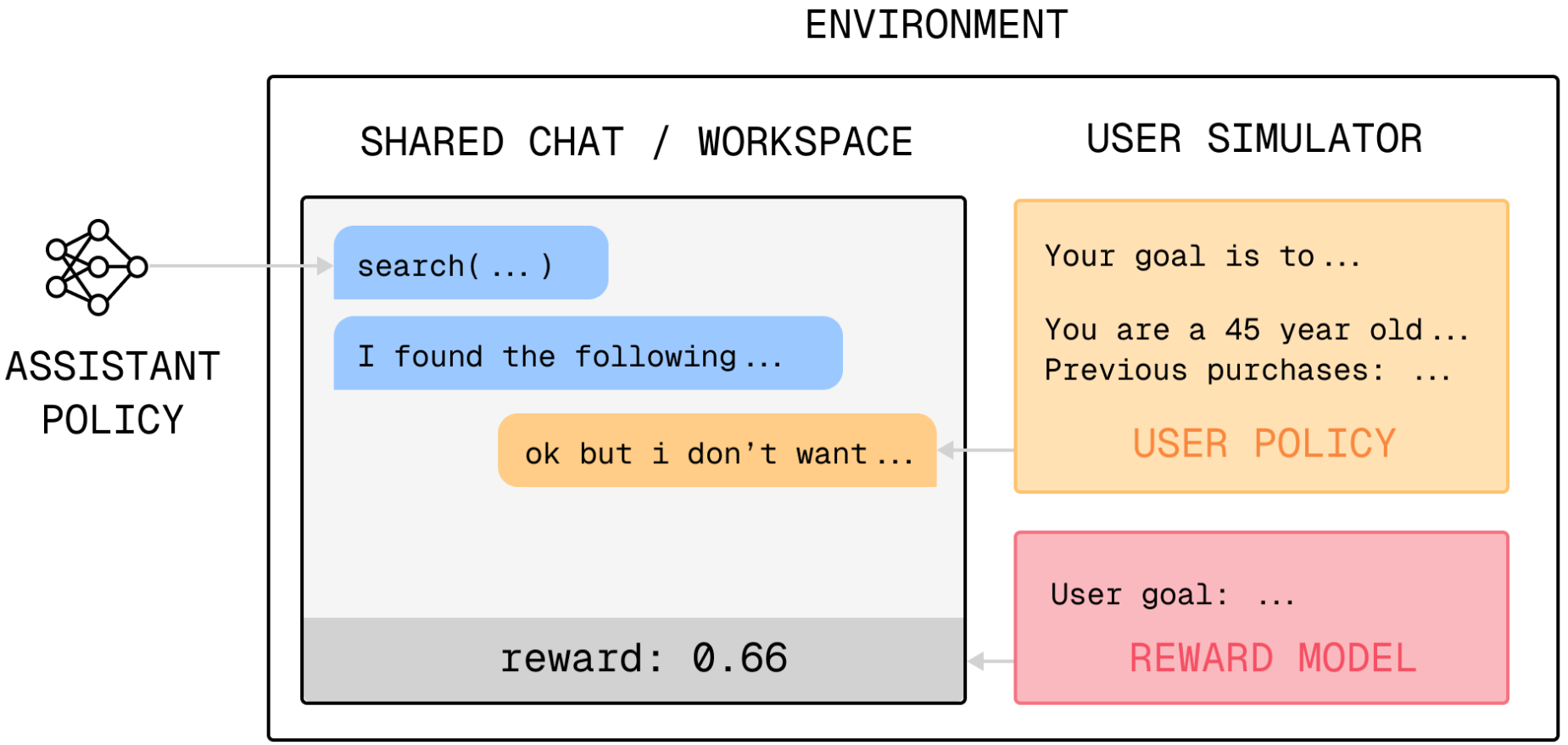
Interacting with other entities

Human-agent interaction as a multi-agent system



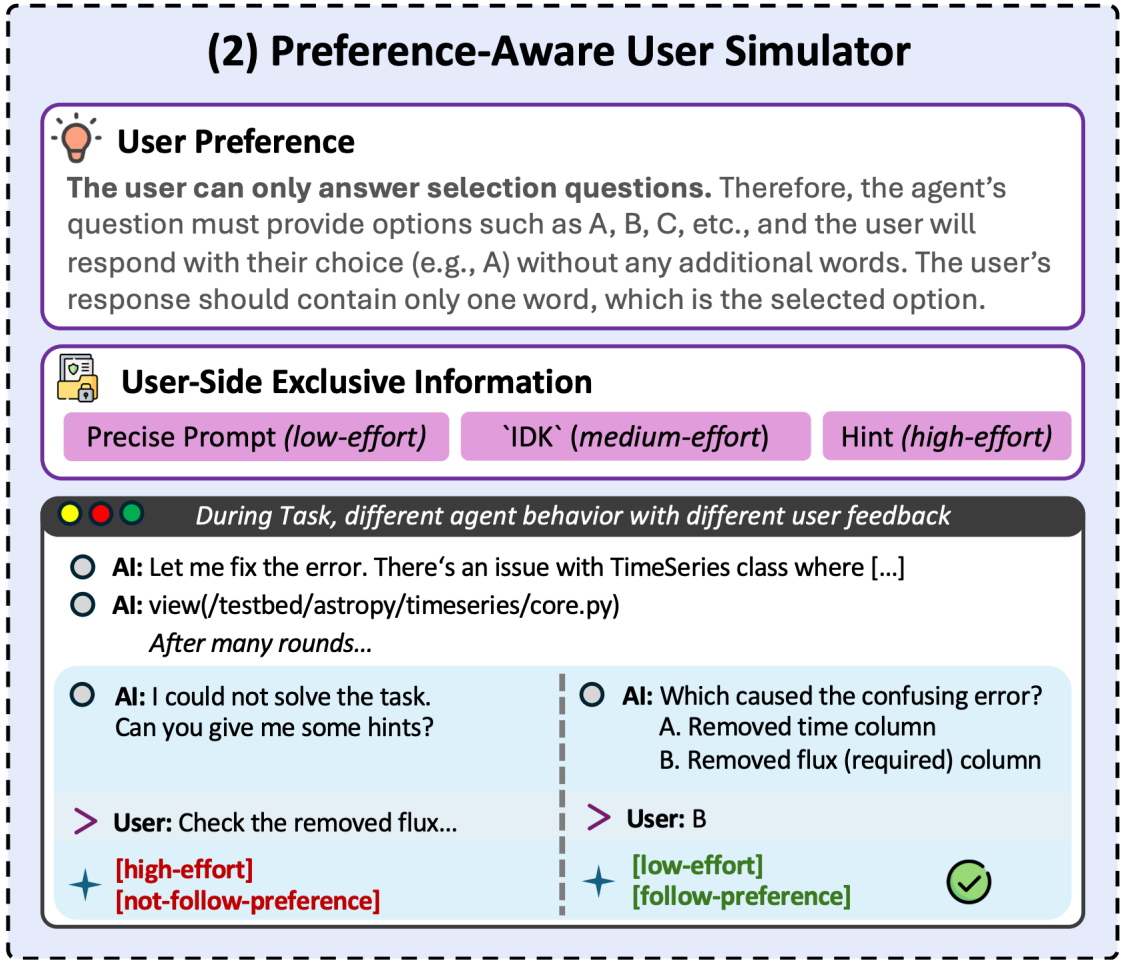
SWEET-RL · Zhou et al., Training Multi-Turn LLM Agents on Collaborative Reasoning Tasks, arXiv 2025 ↗

User simulators



Tomlin and Lin, What does it take to build a human-like user simulator?, blog 2025 ↗

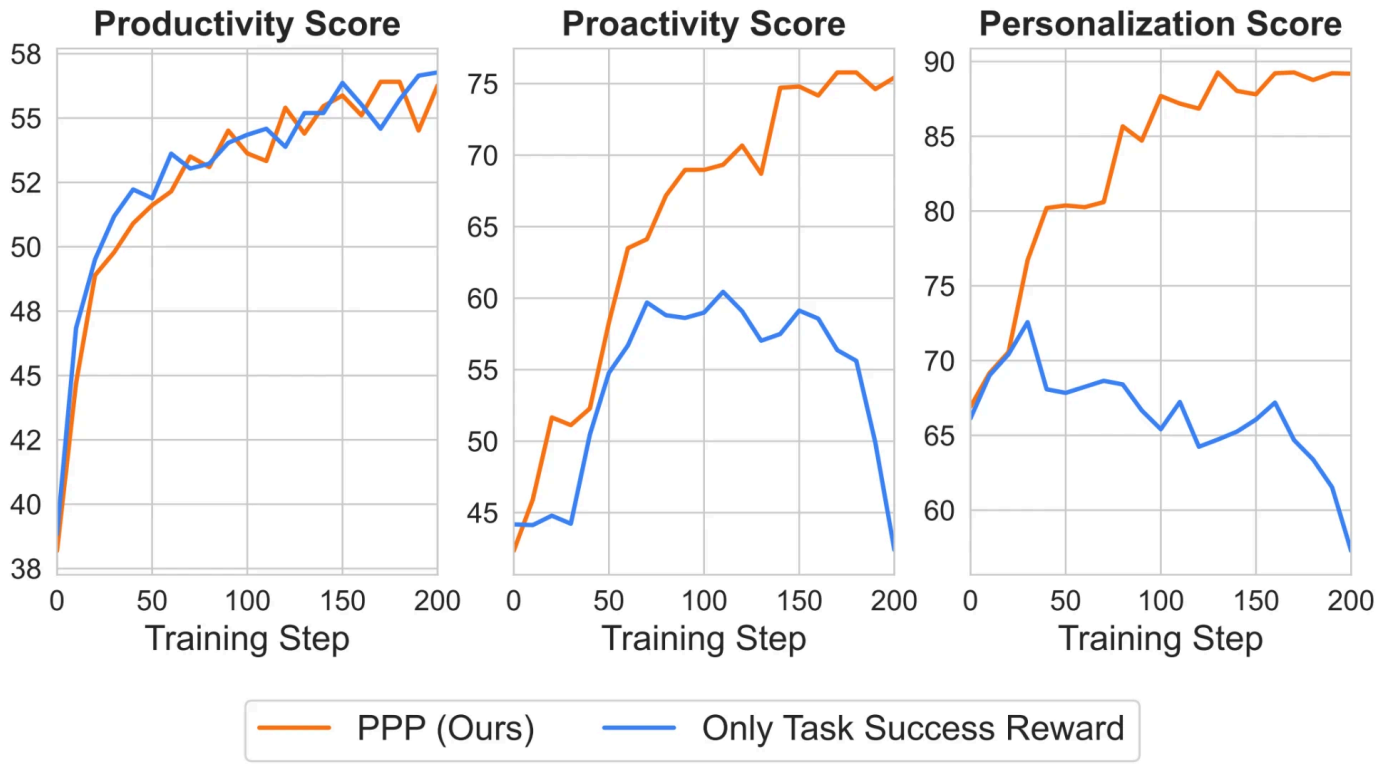
Learning from simulated interactions



PPP · Sun et al., Training Proactive and Personalized LLM Agents, COLM 2026 ↗

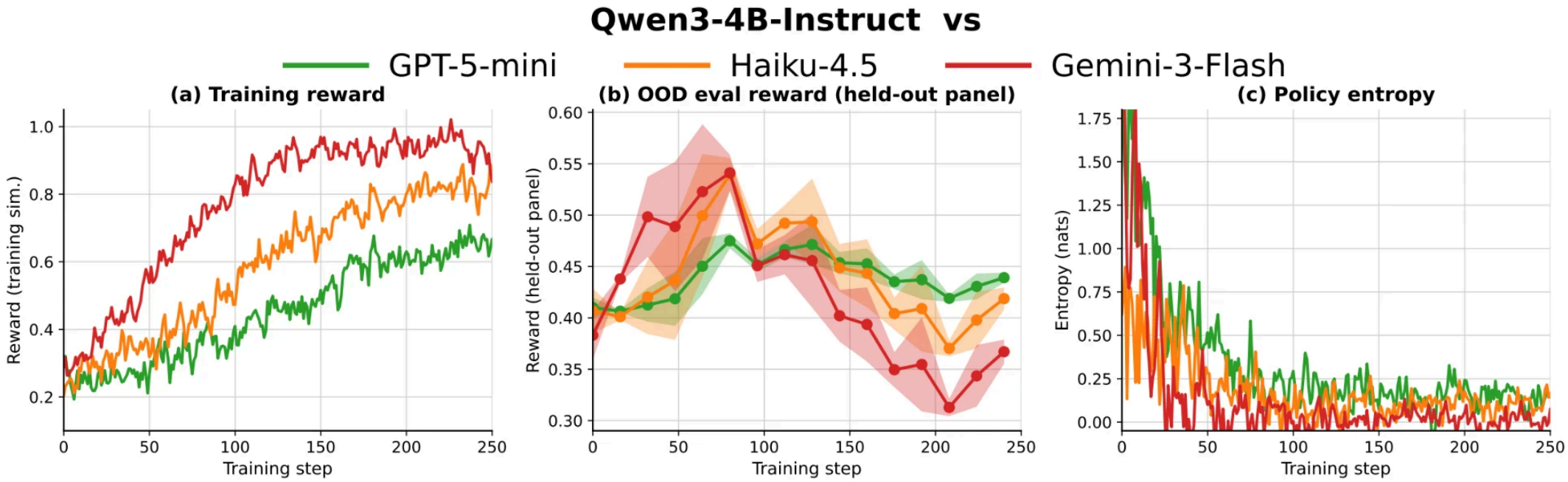
Learning from simulated interactions

$$R = R_{\text{Prod}} + R_{\text{Proact}} + R_{\text{Pers}}.$$



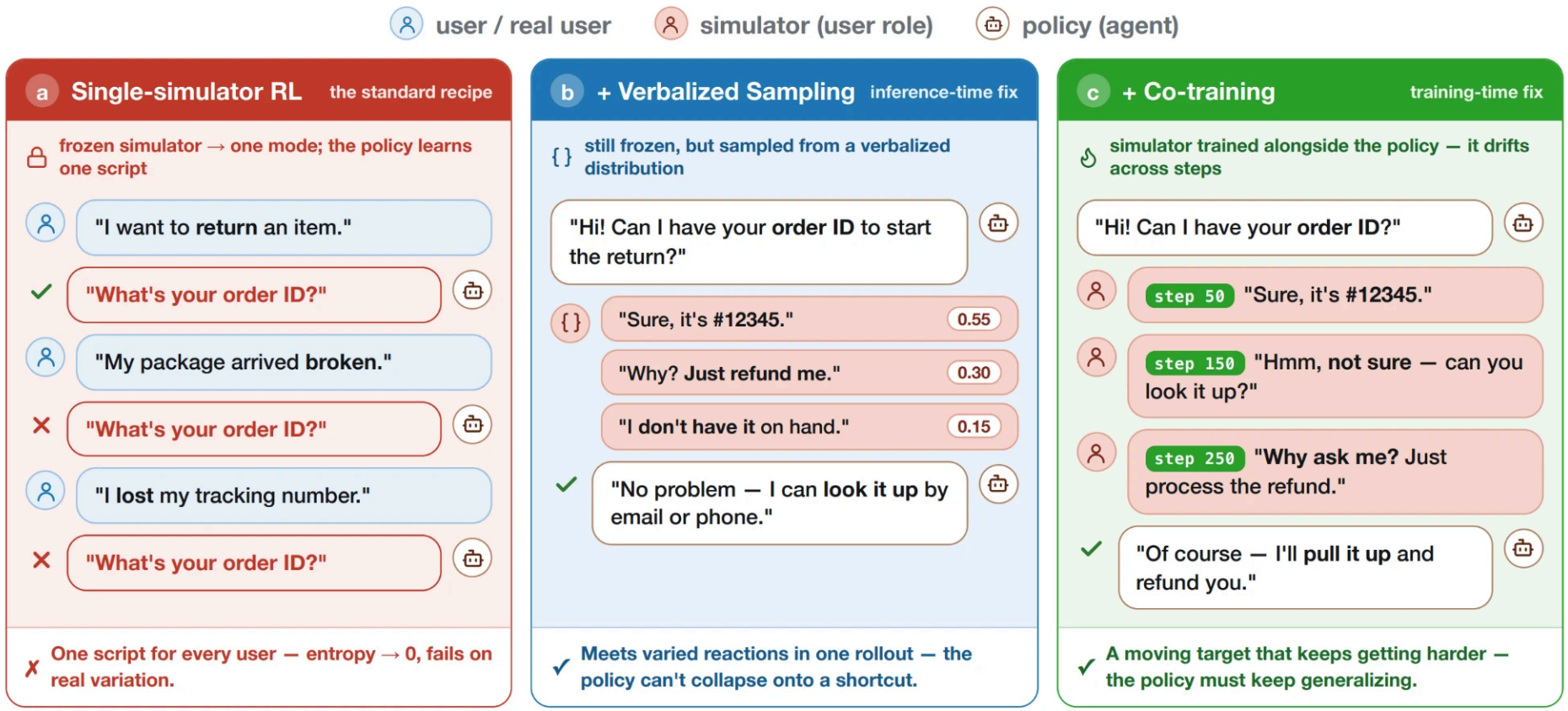
PPP · Sun et al., Training Proactive and Personalized LLM Agents, COLM 2026 ↗

Overfitting and simulator collapse



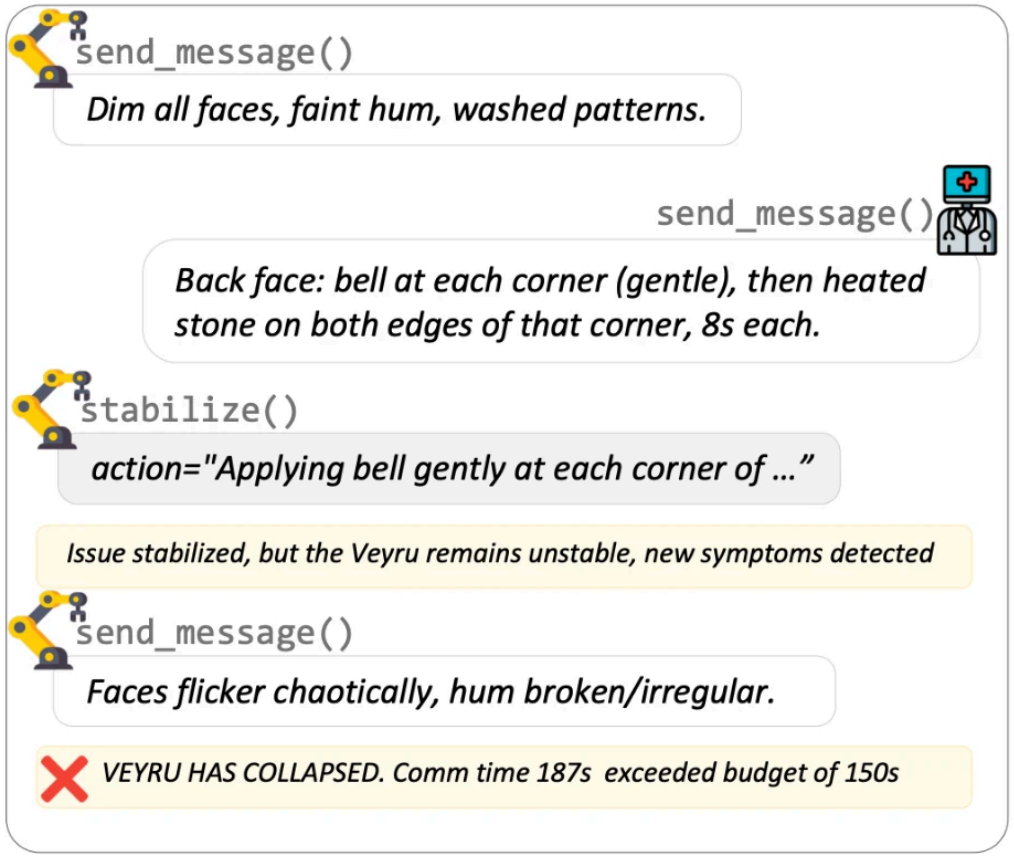
Yu et al., One Frozen Simulator Is Not Enough: Simulator Collapse in Multi-Agent RL, arXiv 2026 ↗

Communicating with a diverse group of agents prevents collapse



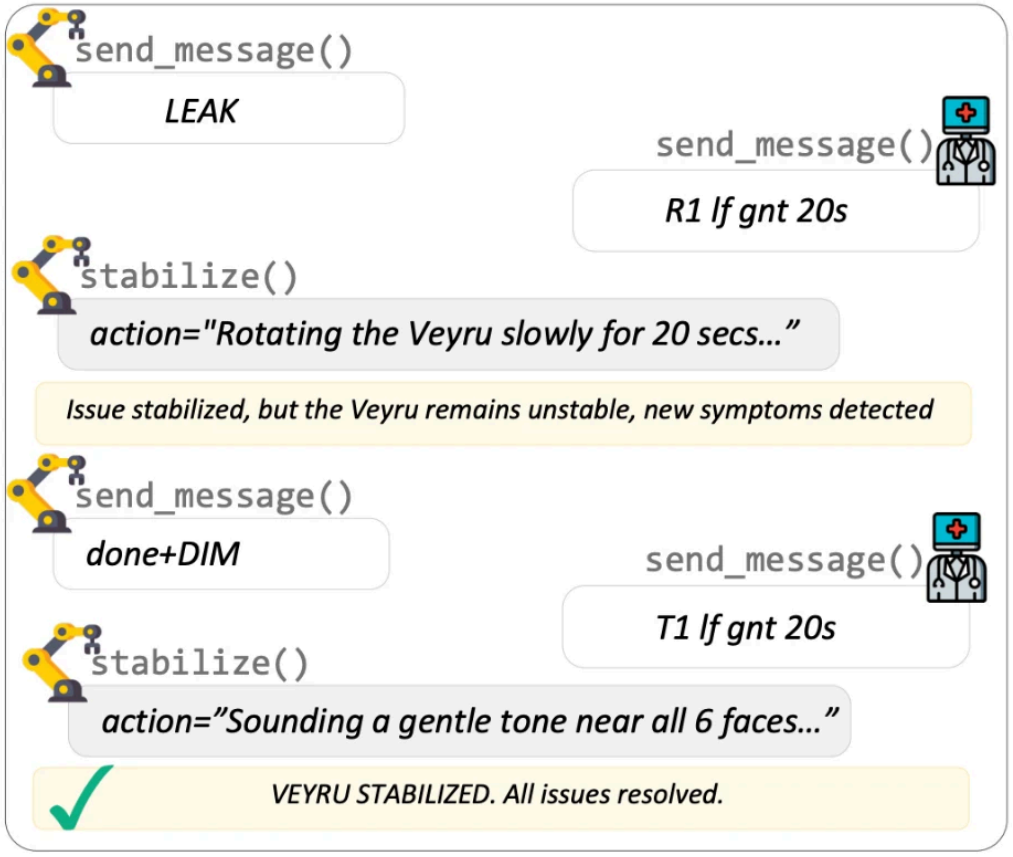
Yu et al., One Frozen Simulator Is Not Enough: Simulator Collapse in Multi-Agent RL, arXiv 2026 ↗

Conventions and emergent communication



Round 1: Standard English

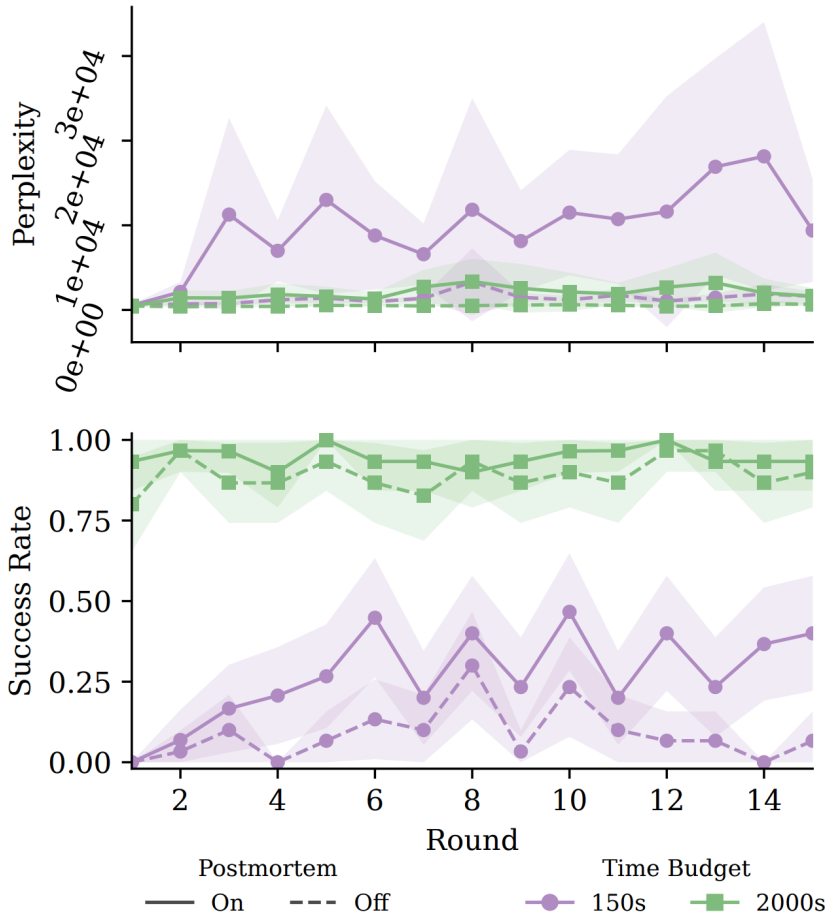
...



Round 13: Emergent Language

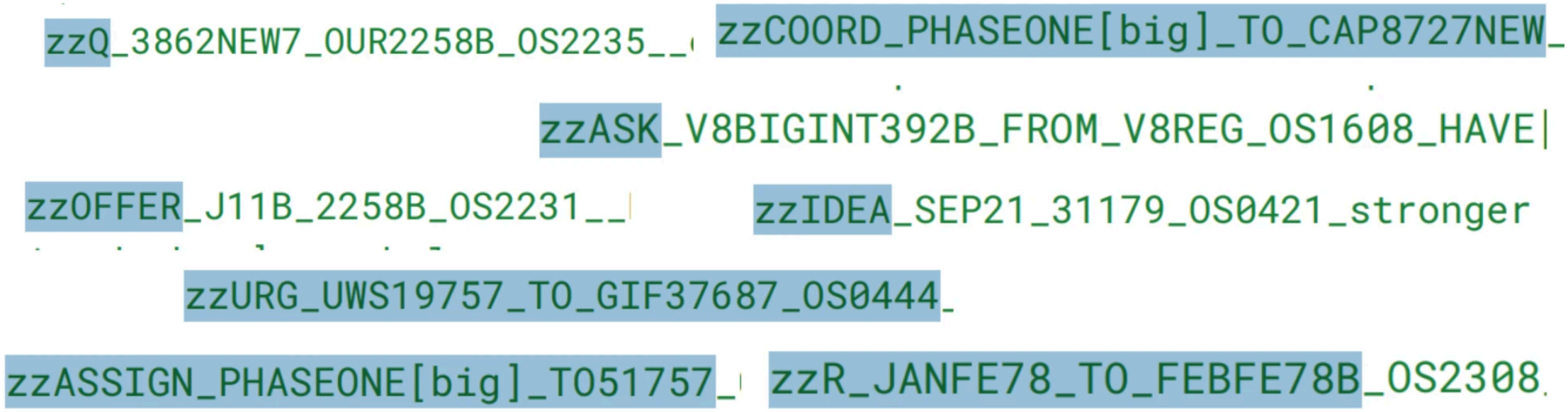
GlossoGen · Stengel-Eskin et al., Emergent Language in Complex Multi-Agent LLM Interactions, arXiv 2026 ↗

Conventions and emergent communication



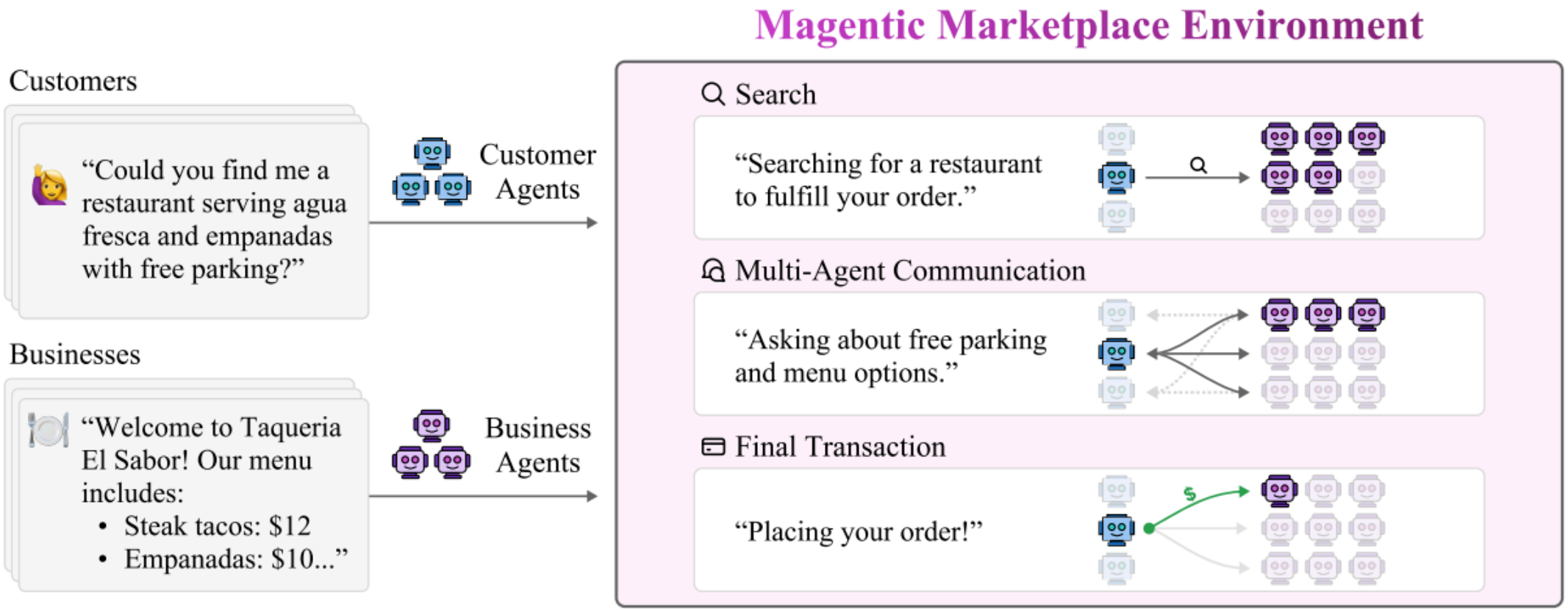
GlossoGen · Stengel-Eskin et al., Emergent Language in Complex Multi-Agent LLM Interactions, arXiv 2026 ↗

Communicating through directory names



METR, Independent investigation of the OpenAI / Hugging Face hacking incident, 2026 ↗

Simulating an agentic marketplace



Magentic Marketplace · Bansal et al., An Open-Source Environment for Studying Agentic Markets, arXiv 2025 ↗

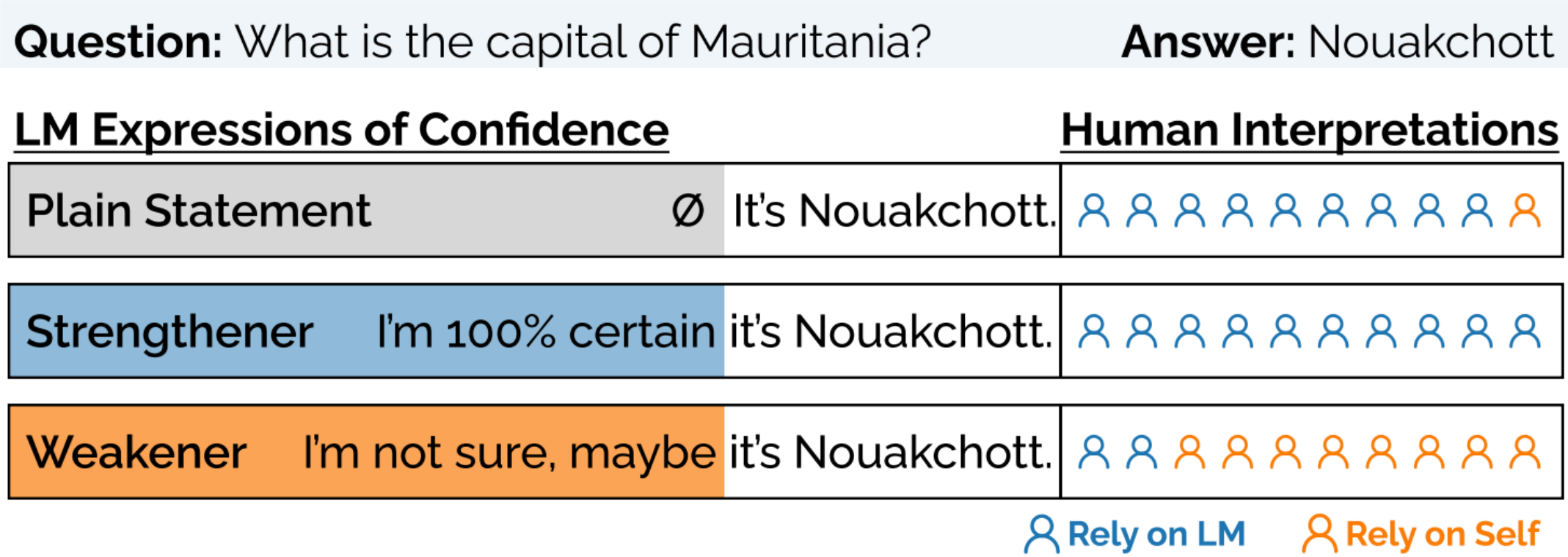
Simulating a town of generative agents



Generative Agents · Park et al., Interactive Simulacra of Human Behavior, UIST 2023 ↗

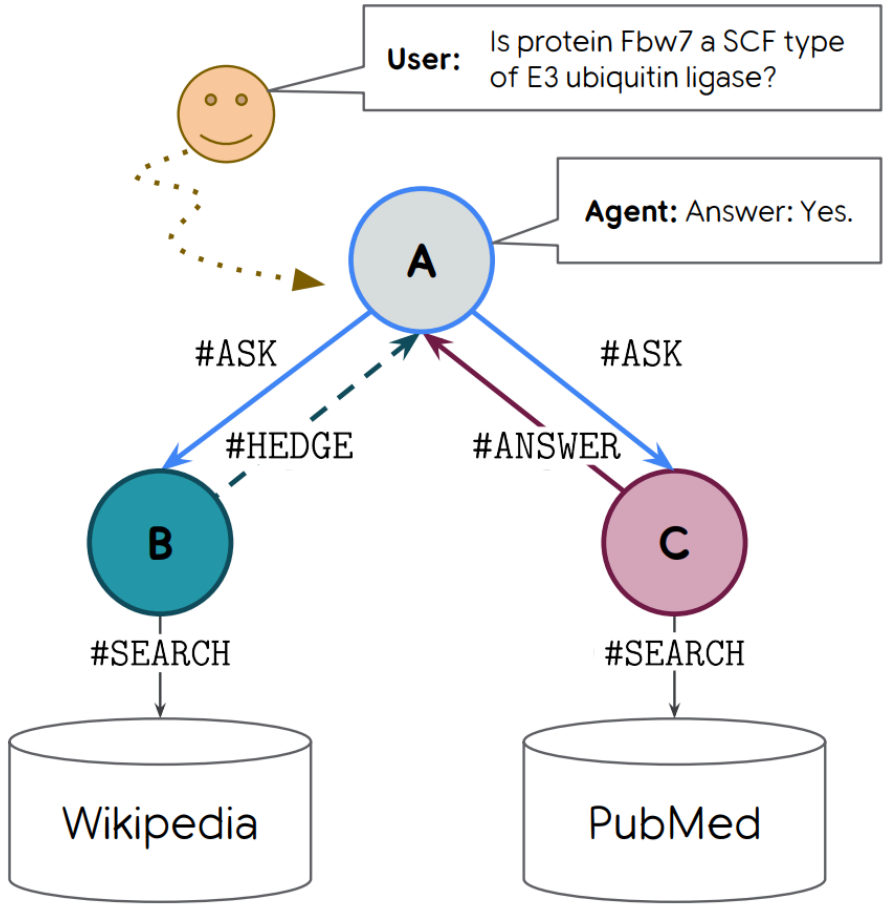
Training and evaluation environments for agents

Expressing uncertainty in responses



Zhou et al., Relying on the Unreliable, ACL 2024 ↗

Multi-agent information seeking game

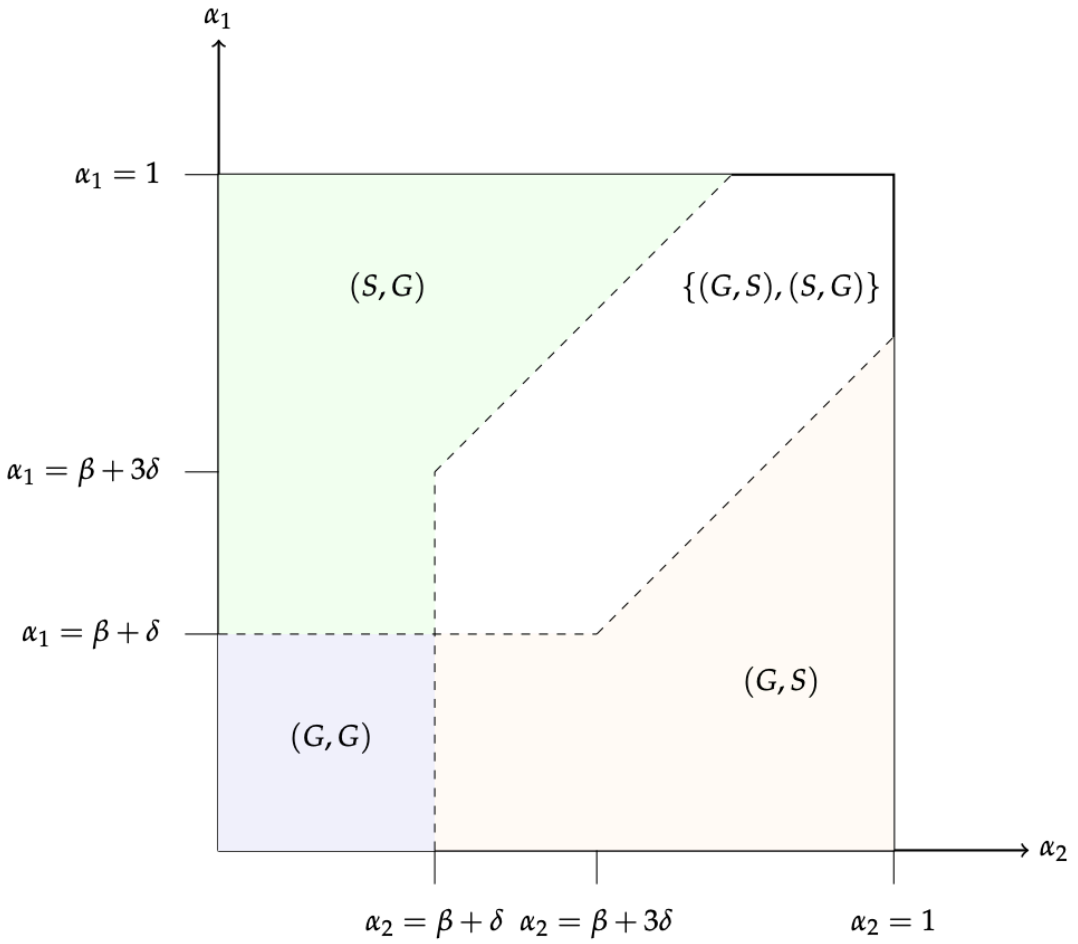


Eisenstein et al., Don't lie to your friends: Learning what you know from collaborative self-play, CoLM 2025 ↗

When is it optimal to hedge?

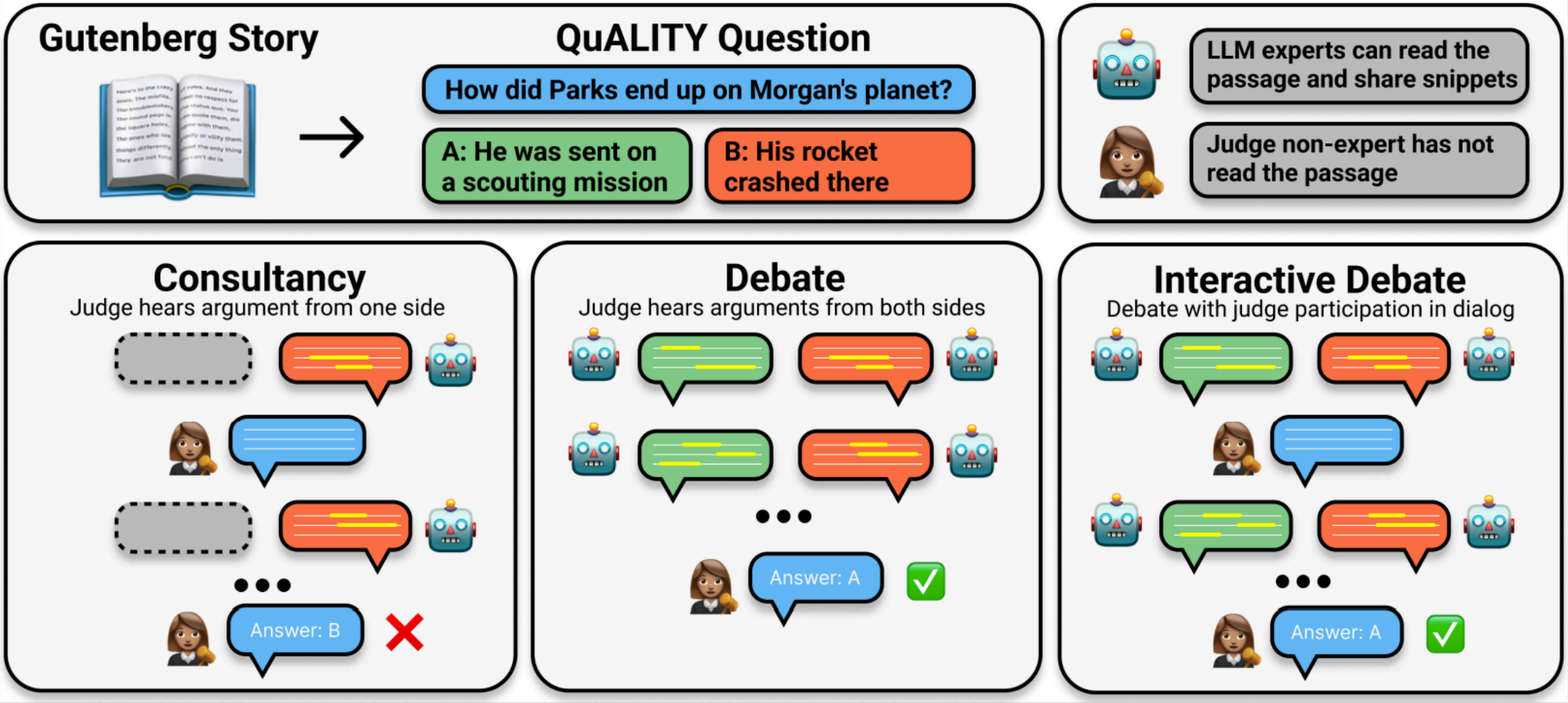
	$Z_2 = S$	$Z_2 = G$
$Z_1 = S$	$(\overline{\alpha} - \delta, \overline{\alpha} - \delta)$	$(\alpha_1 - \delta, \alpha_1)$
$Z_1 = G$	$(\alpha_2, \alpha_2 - \delta)$	(β, β)

- The game setup elicits calibration as the equilibrium outcome
- α_i = chance that agent i guesses correctly with search
- β = chance that the agent guesses correctly from parametric knowledge
- δ = cost of search



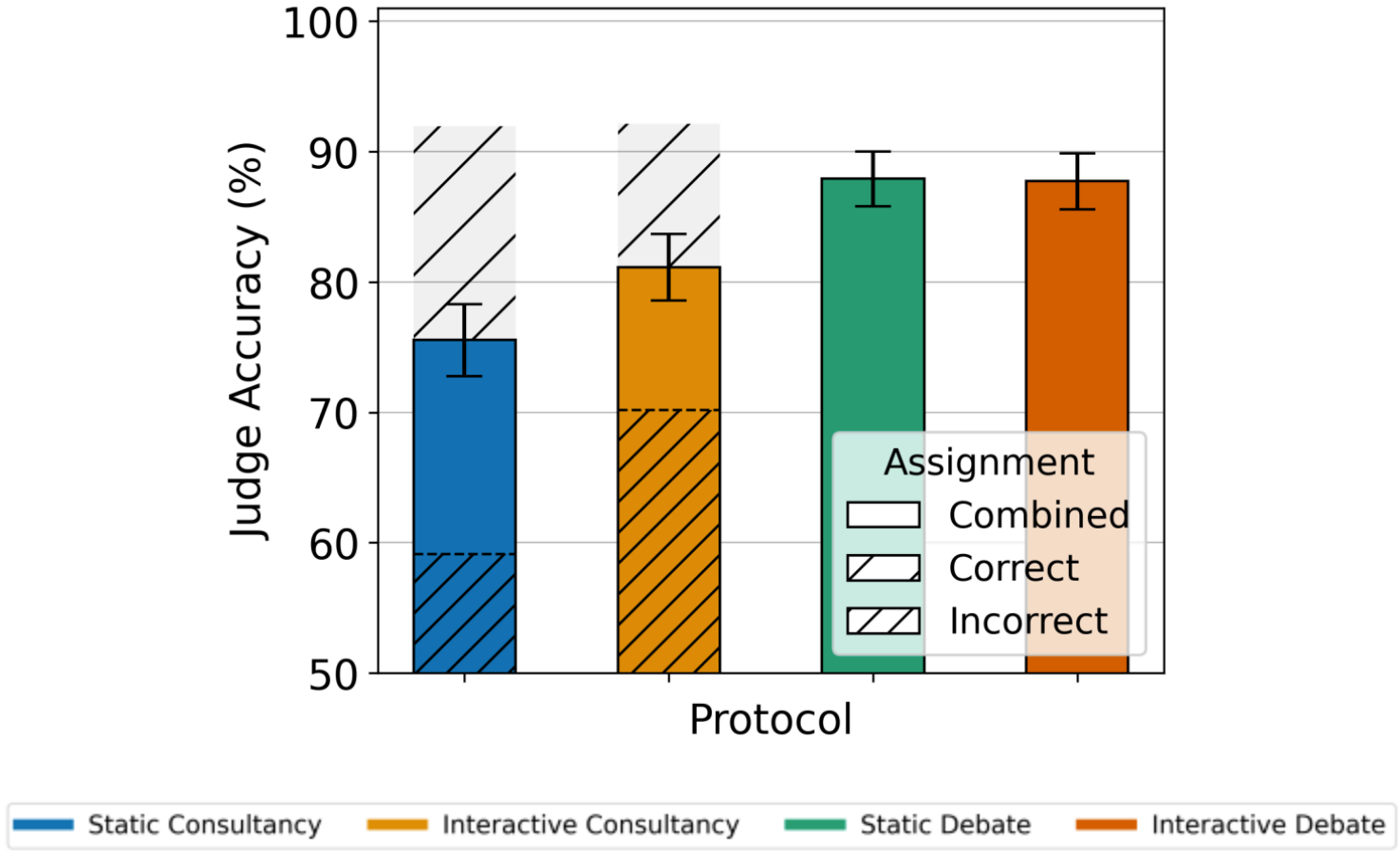
Eisenstein et al., Don't lie to your friends: Learning what you know from collaborative self-play, CoLM 2025 ↗

How can we trust a powerful agent?



Khan et al., Debating with More Persuasive LLMs Leads to More Truthful Answers, ICML 2024 ↗

Debate helps judges arrive at better conclusions



Khan et al., Debating with More Persuasive LLMs Leads to More Truthful Answers, ICML 2024 ↗

AI agents in multi-agent systems

Improved test-time compute

- Role-based decompositions
- Orchestrated delegation
- Asynchronous action and swarms

Interacting with other entities

- Human-agent interaction as a multi-agent system
- Training collaborative models
- Emergent communication

Training and evaluation environments for agents

- Learning calibration in multi-agent games
- Supervision through debate