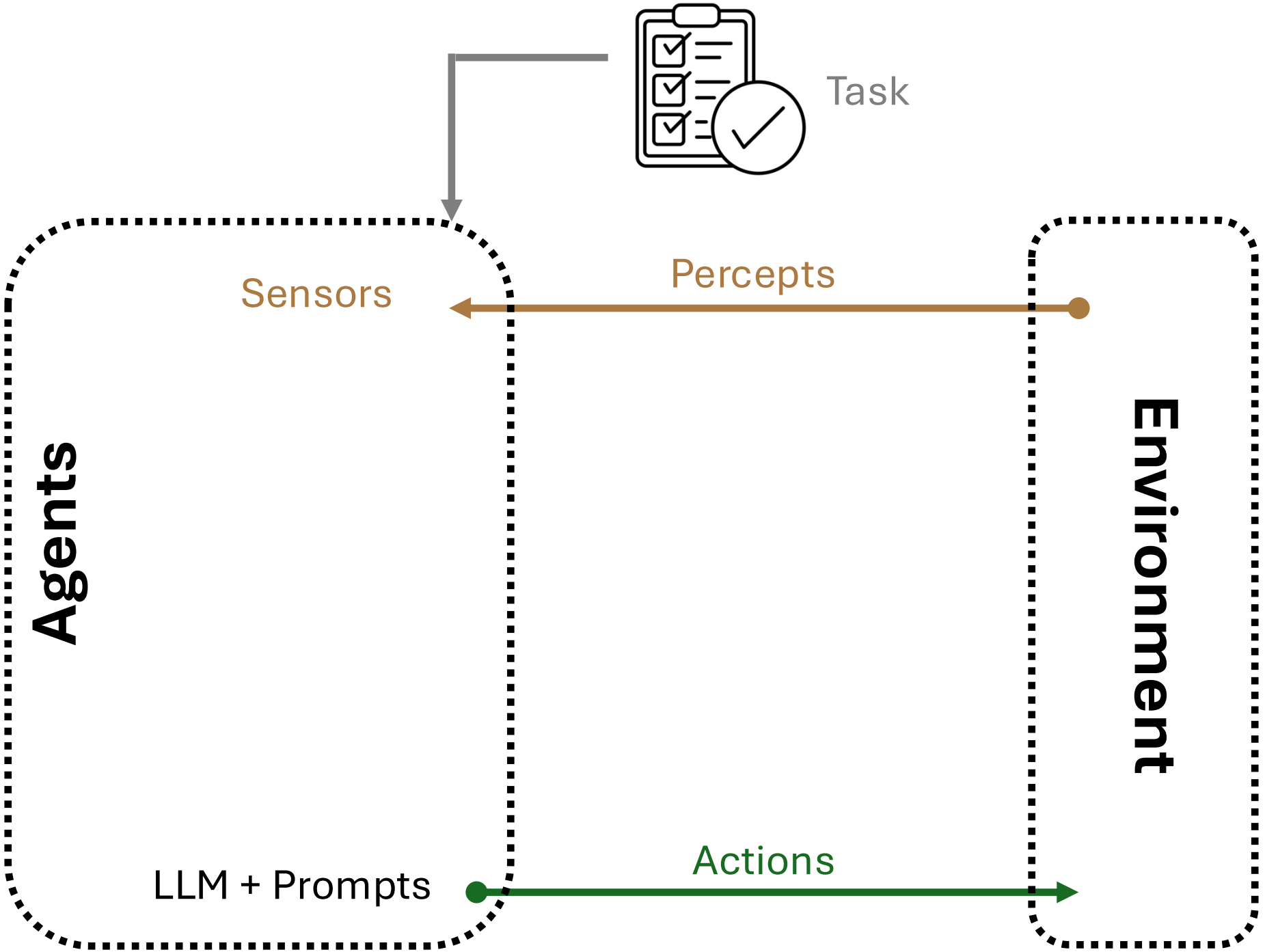
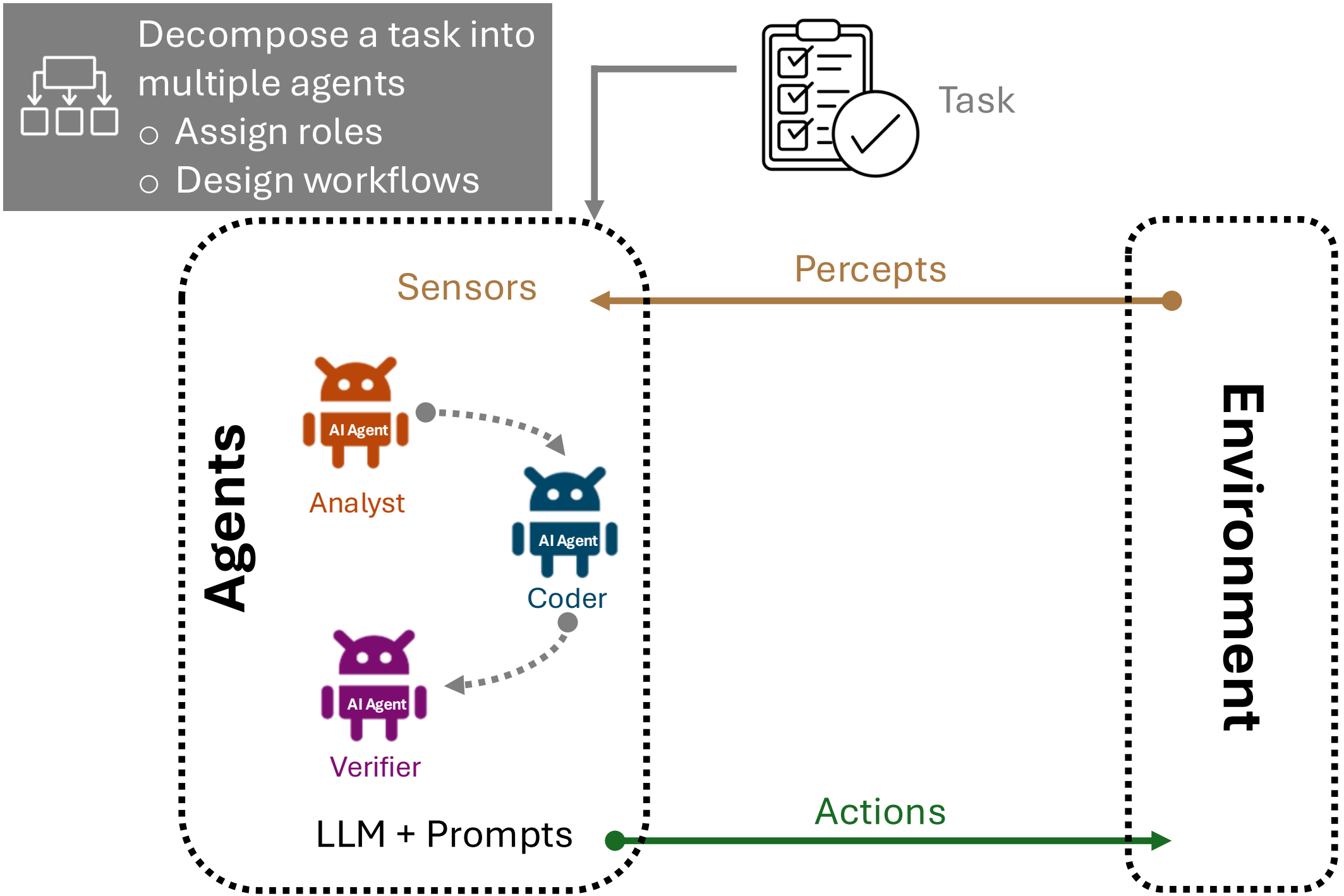
Reasoning as Control

Toward Self-Improving Agentic Systems

Across Thinking, Actions, and Workflow

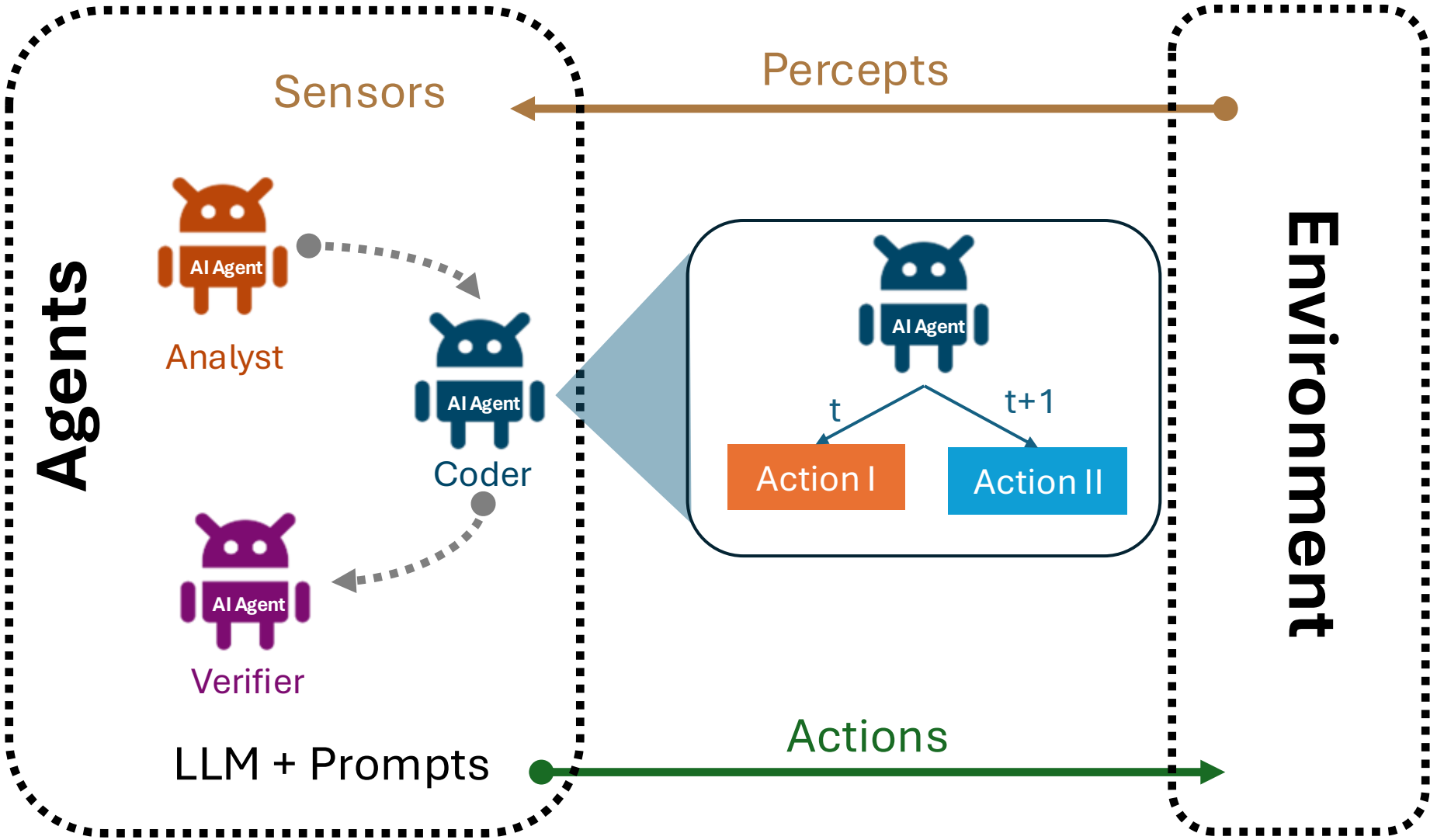
Furong Huang | https://furong-huang.com/
University of Maryland


Task
Agents
Sensors
Percepts
Environment
LLM + Prompts
Actions


Decompose a task into multiple agents
Assign roles
Design workflows
Task
Agents
Sensors
Percepts
AI Agent
Analyst
AI Agent
Coder
AI Agent
Verifier
LLM + Prompts
Actions
Environment

Decompose a task into multiple agents
- Assign roles
- Design workflows

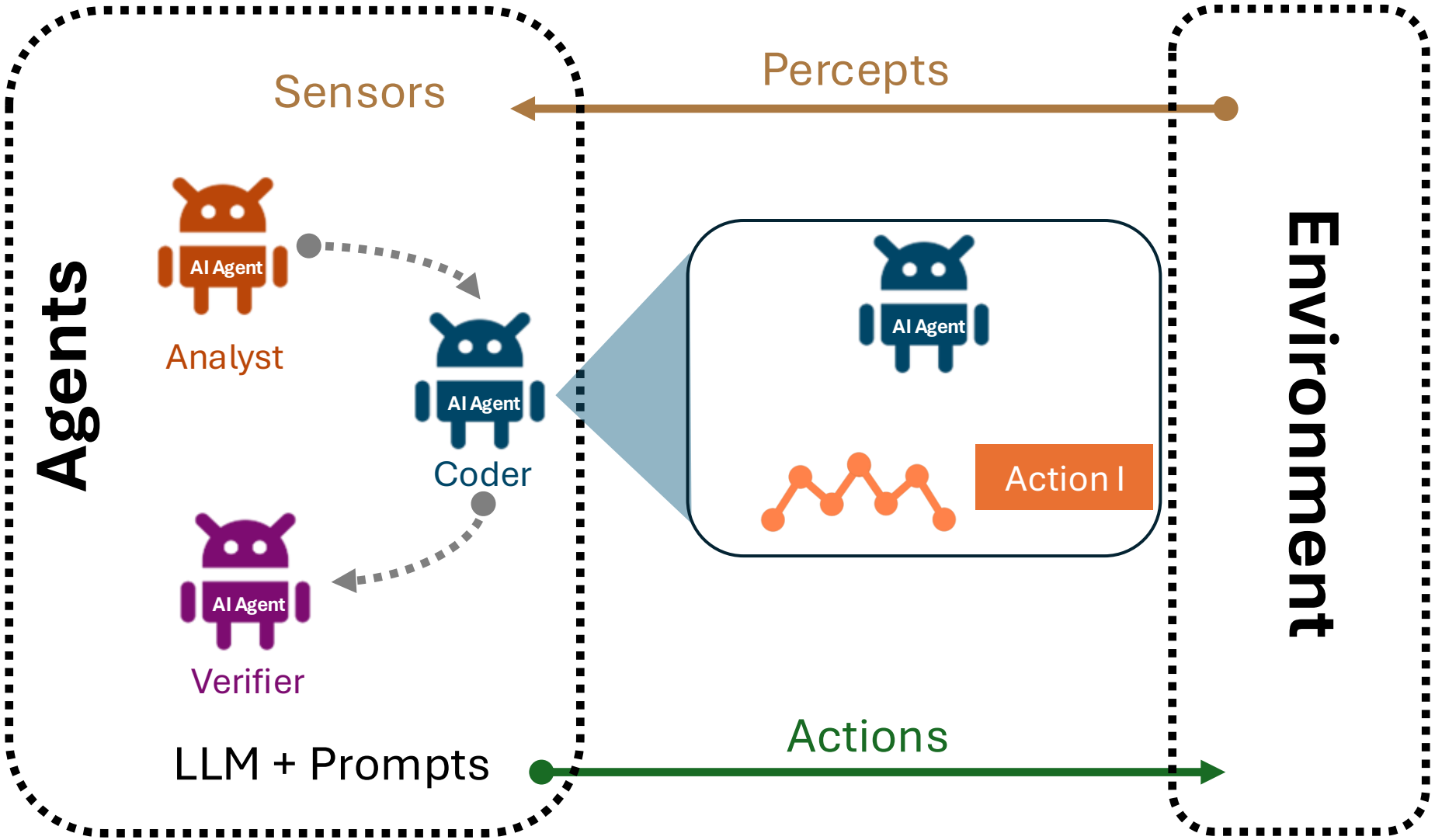
Each agent decides the action at every time step.

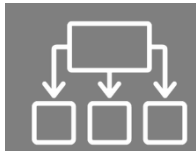


Decompose a task into multiple agents
- Assign roles
- Design workflows

Each agent decides the action at every time step.

For each action decision, the agent generates a reasoning trace.

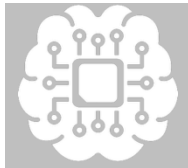




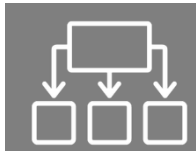
Decompose a task into multiple agents

- Assign roles
- Design workflows

Each agent decides the action at every time step.



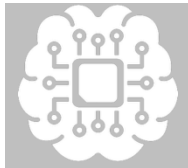
For each action decision, the agent generates a reasoning trace.



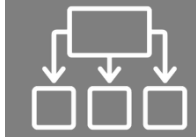
Decompose a task into multiple agents

- Assign roles
- Design workflows

Each agent decides the action at every time step.



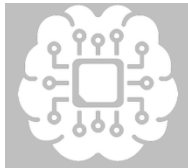
Thinking Trace/Token Control



Decompose a task into multiple agents

- Assign roles
- Design workflows

Action Control

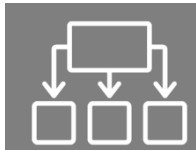


Thinking Trace/Token Control

Toward Self-Improving Agentic Systems

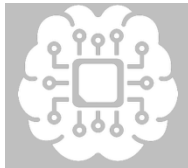
Across Thinking, Actions, and Workflow

Foundation models are becoming **Runtime Decision-Makers**



Workflow Control

Action Control



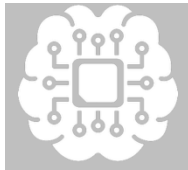
Thinking Trace/Token Control

Toward Self-Improving Agentic Systems

Across Thinking, Actions, and Workflow

Foundation models are becoming **Runtime Decision-Makers**

Level 1 Thinking Trace /Token Control



Thinking Trace/Token Control

The Steering Problem Formulation

$$\pi^*_{\text{dec}}(\cdot|\mathbf{s}_t) := \arg\max_{\pi \in \Pi} \mathbb{E}_{z \sim \pi(\cdot|\mathbf{s}_t)} \left[Q^*(\mathbf{s}_t, z) \right] - \alpha \mathbb{D}_{\text{KL}} \left[\pi(\cdot|\mathbf{s}_t) || \pi_{\text{sft}}(\cdot|\mathbf{s}_t) \right]$$

state $\mathbf{s}_t$ =

$$\pi^*_{\text{dec}}(z|\mathbf{s}_t) = \pi_{\text{sft}}(z|\mathbf{s}_t) \frac{\exp\left(\frac{1}{\alpha} Q^*(\mathbf{s}_t, z)\right)}{C_\alpha}$$

ıess to
ıodel

Mudgal, Sidharth, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen et al. "Controlled decoding from language models." in ICML 2024.

The Steering Problem Formulation

LLM

ρ_π , π

Steer toward

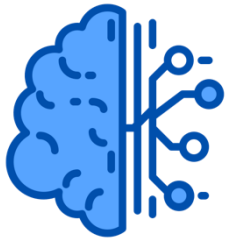
Reward Model

$r(\mathbf{x}, \mathbf{y})$

$$\pi^*_{\text{dec}}(z|\mathbf{s}_t) = \pi_{\text{sft}}(z|\mathbf{s}_t)\frac{\exp\left(\frac{1}{\alpha}Q^*(\mathbf{s}_t, z)\right)}{C_\alpha}$$

In log scale

$$\log \pi_{\text{decode}} = \text{Base LLM} + \text{Reward}$$



Reward $Q^*(\mathbf{s}_t, y_t)$

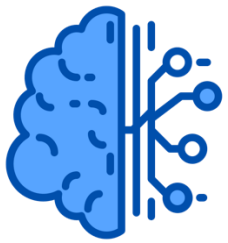
$$= \max_\pi \mathbb{E}_{\tau \sim \rho^*(\cdot|\mathbf{s}_t, y_t)}\left[r\left([\mathbf{x}, \mathbf{y}_{<t}, y_t], \tau\right)\right]$$

Trajectory reward under **optimal policy**

Mudgal, Sidharth, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen et al. "Controlled decoding from language models." in ICML 2024.

The Trajectory-Reward Problem

$$\log \pi_{\text{decode}} = \text{Base LLM} + \text{Reward}$$



Reward $Q^*(\mathbf{s}_t, y_t)$

$$= \max_{\pi} \mathbb{E}_{\tau \sim \rho^*(\cdot|\mathbf{s}_t, y_t)} [r([\mathbf{x}, \mathbf{y}_{<t}, y_t], \tau)]$$

Trajectory reward under optimal policy

$$\log \pi_{\text{decode}}(y|x) = -\log Z(x) + \log \pi_{\text{base}}(y|x) + \frac{1}{\beta} r(x, y)$$

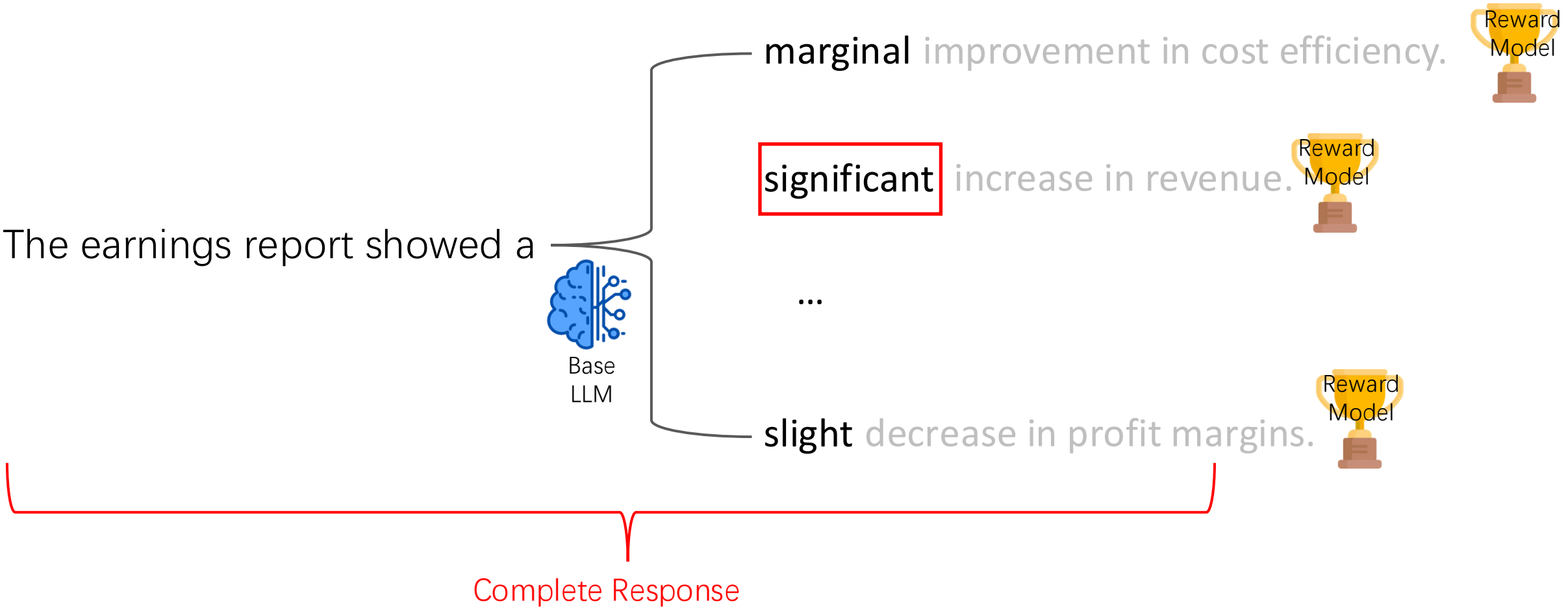
Next token sampling

$$\log \pi_{\text{decode}}(y_t|x, y_{:t}) \propto \log \pi_{\text{base}}(y_t|x, y_{:t}) + \frac{1}{\beta} r(y_t|x, y_{:t})$$

But only trajectory-level reward available

Transfer Q* (NeurIPS 2024) **& DeAL** (ACL 2025)

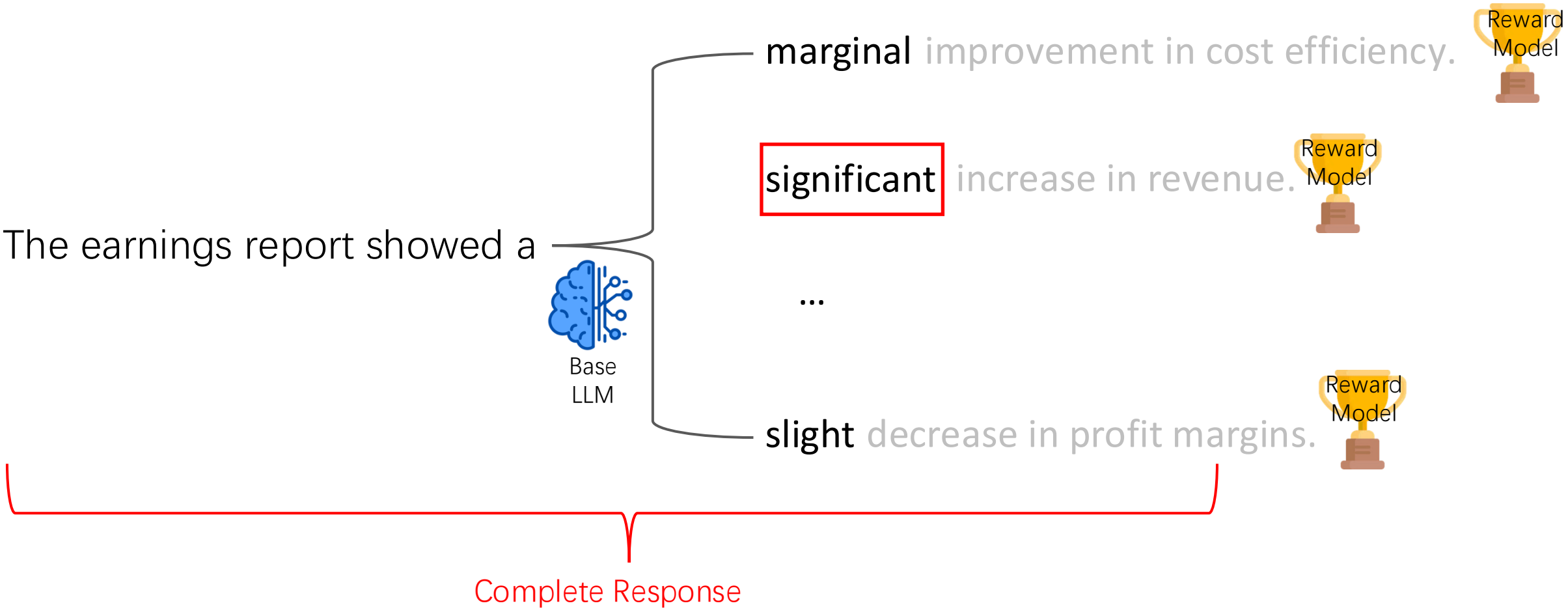
Use trajectory-level rewards via trajectory completion/rollout



CGYMWBH, Transfer Q Star: Principled Decoding for LLM Alignment, NeurIPS 2024.
Huang, James Y., et al. Deal: Decoding-time alignment for large language models. ACL 2025.

Transfer Q* (NeurIPS 2024) **& DeAL** (ACL 2025) **are correct**

Use trajectory-level rewards via trajectory completion/rollout



But Slow, require generating the full response for each next token sampling

Generating a response with 500 tokens:

Transfer Q*/DeAL

Trajectory-level reward model
(Evaluate complete responses)

14 hours

Generating a response with 500 tokens:

Transfer Q*/DeAL	**Our work**
Trajectory-level reward model (Evaluate complete responses)	Autoregressive reward model (Generate next token reward)
14 hours	20 seconds

Parametrization of $r(x, y)$

$$r(x, y) = \log \pi_r(y|x)$$

$$= \sum_t \log \pi_r(y_t|x, y_{:t})$$

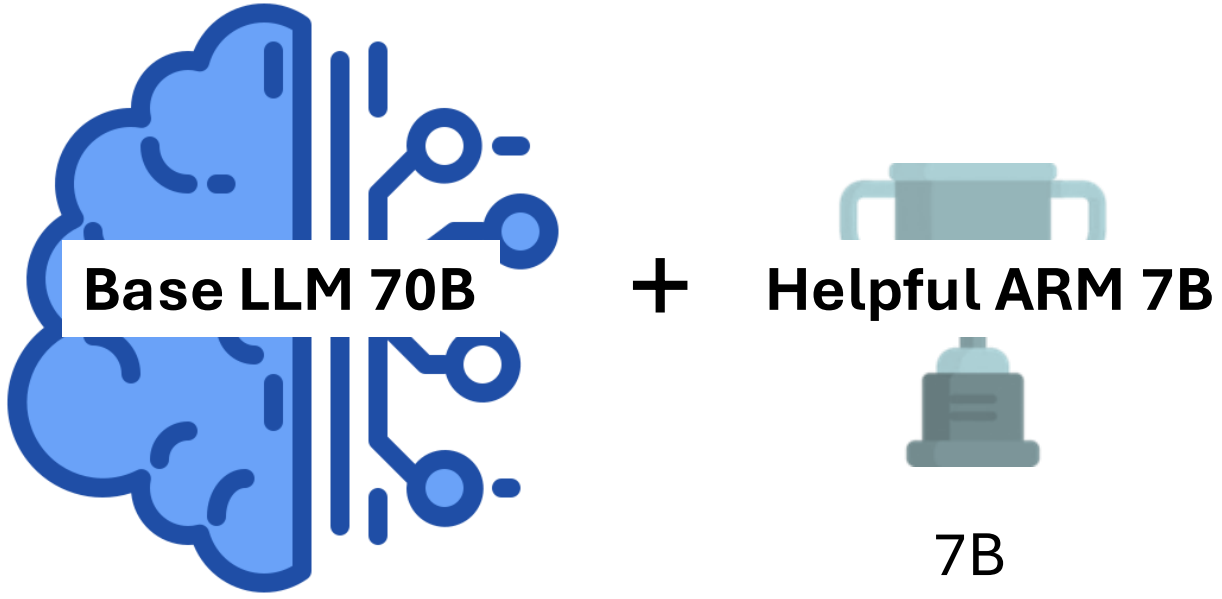
Generating a response with 500 tokens:

Transfer Q*/DeAL	**Our work**
Trajectory-level reward model (Evaluate complete responses)	Autoregressive reward model (Generate next token reward)
14 hours	20 seconds

XSKZAHG, GenARM: Reward Guided Generation with Autoregressive Reward Model for Test-time Alignment, ICLR 2025.

Exp 1: Aligning with general human preference

Exp 2: Weak-to-strong guidance



Exp 3: Multi-objective alignment (diverse preferences)

XSKZAHG, GenARM: Reward Guided Generation with Autoregressive Reward Model for Test-time Alignment, ICLR 2025.

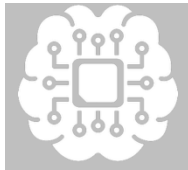
Toward Self-Improving Agentic Systems

Across Thinking, Actions, and Workflow

Foundation models are becoming **Runtime Decision-Makers**

Workflow Control

Action Control



Thinking Trace/Token Control

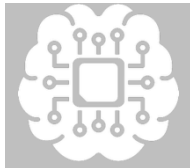
Toward Self-Improving Agentic Systems

Across Thinking, Actions, and Workflow

Foundation models are becoming **Runtime Decision-Makers**

Level 2 Action Control

Action Control



Thinking Trace/Token Control

Imitation Learning: The “Stuck” Loop

Agent’s Faulty Plan (Imitation Learning):

1. Pick up cloth.
2. Go to sink.
3. Clean cloth.
4. Go to cabinet.
5. **Put cloth in cabinet.**

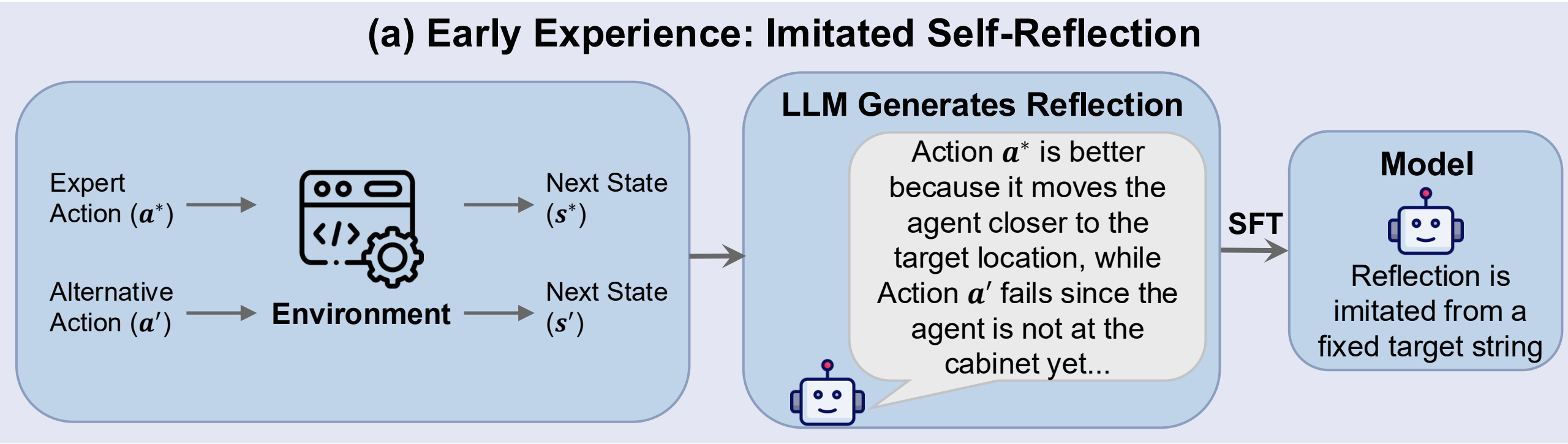
Step 9: Put cloth in cabinet.

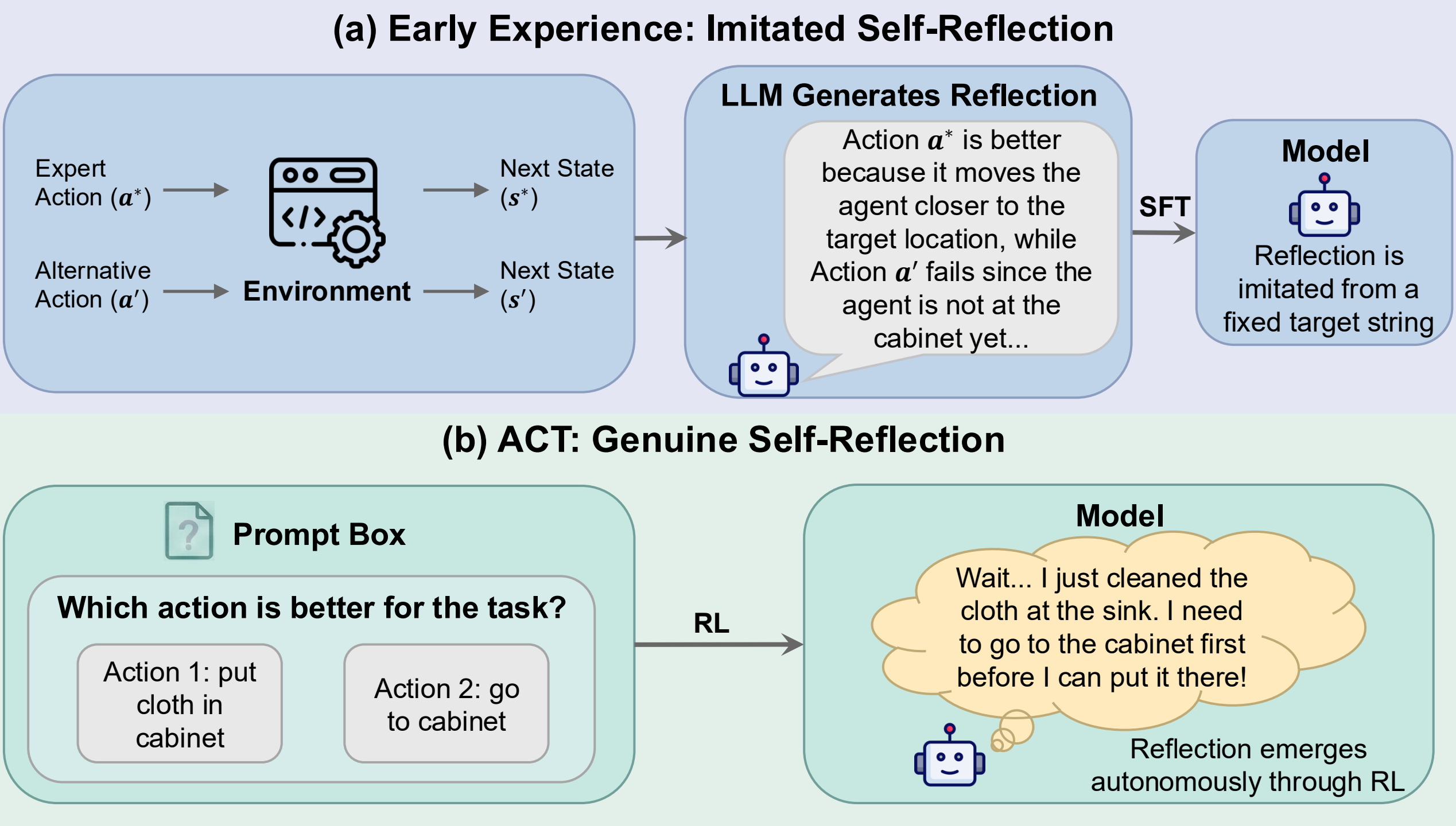
❌ **FAILS** ⚠

Step 10: Put cloth in cabinet.

Step 11: Put cloth in cabinet.

Model repeats failed action for 30+ steps until termination.


(a) Early Experience: Imitated Self-Reflection
Expert Action (a^*)
Alternative Action (a')
Environment
Next State (s^*)
Next State (s')
LLM Generates Reflection
Action a^* is better because it moves the agent closer to the target location, while Action a' fails since the agent is not at the cabinet yet...
SFT
Model
Reflection is imitated from a fixed target string


(a) Early Experience: Imitated Self-Reflection
Expert Action (a^*)
Alternative Action (a')
Environment
Next State (s^*)
Next State (s')
LLM Generates Reflection
Action a^* is better because it moves the agent closer to the target location, while Action a' fails since the agent is not at the cabinet yet...
SFT
Model
Reflection is imitated from a fixed target string
(b) ACT: Genuine Self-Reflection
Prompt Box
Which action is better for the task?
Action 1: put cloth in cabinet
Action 2: go to cabinet
RL
Model
Wait... I just cleaned the cloth at the sink. I need to go to the cabinet first before I can put it there!
Reflection emerges autonomously through RL

Method	ALFWorld		WebShop	ScienceWorld
	ID	OOD		
Prompt w/o CoT thinking	35.71	27.61	2.80	28.01
Prompt w/ CoT thinking	56.43	50.00	3.00	25.21
ACT	72.86	72.39	7.40	26.71
Imitation Learning	85.71	82.84	28.00	42.80
Early Experience (Self-Reflection)	87.86	85.82	31.00	45.60
IL w/ ACT	91.43	87.31	31.60	48.69
RL	90.71	84.33	29.40	43.04
RL w/ ACT	**92.86**	**88.06**	**33.80**	**50.34**

Main results on Qwen3-8B (%). ALFWorld and WebShop report success rates; ScienceWorld reports next-action prediction accuracy.

ACT improves IL and RL

ACT improves OOD generalization

ACT outperforms Early Experience:
RL-driven genuine self-reflection > imitating pre-generated reflection text

Generalization to General Reasoning

Model trained on ALFWorld agentic data → MATH-500, GPQA-Diamond

Method	MATH-500	GPQA-Diamond
Prompt w/o CoT thinking	78.6±0.33	42.93±1.09
Prompt w/ CoT thinking	86.93±0.74	51.52±1.89
Imitation Learning	87±0.33	44.61±0.95
Early Experience (Self-Reflection)	86.86±0.25	51.85±0.63
RL	87.07±0.77	52.36±1.32
ACT	**87.73±0.19**	**53.37±0.63**

Agentic RL environment, when **combined with the ACT objective,** can be a viable pathway for enhancing **general reasoning capabilities**

Controlled decoding steers *how the agent* ***thinks***;

ACT trains the agent to judge *which* ***action*** *is actually better*.

Toward Self-Improving Agentic Systems

Across Thinking, Actions, and Workflow

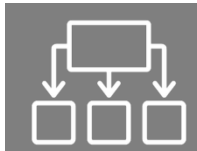
Foundation models are becoming **Runtime Decision-Makers**

Toward Self-Improving Agentic Systems

Across Thinking, Actions, and Workflow

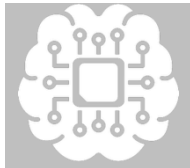
Foundation models are becoming **Runtime Decision-Makers**

Level 3 Workflow Control



Workflow Control

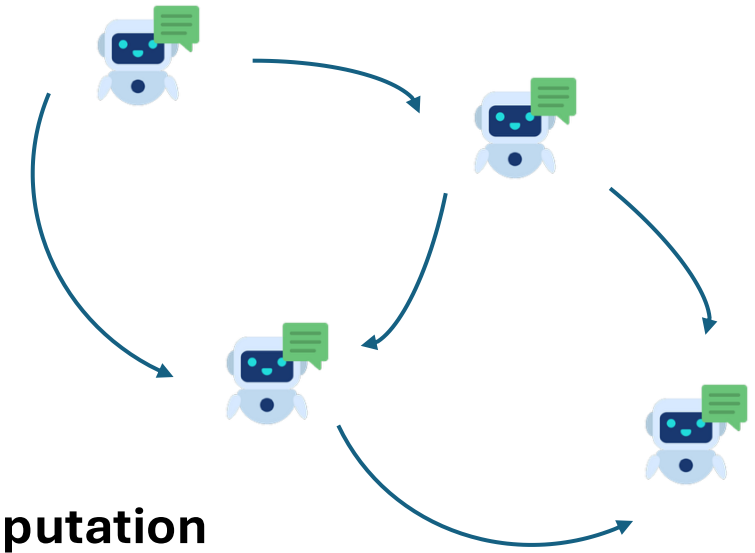
Action Control



Thinking Trace/Token Control

Agentic Workflow and Autonomous Design

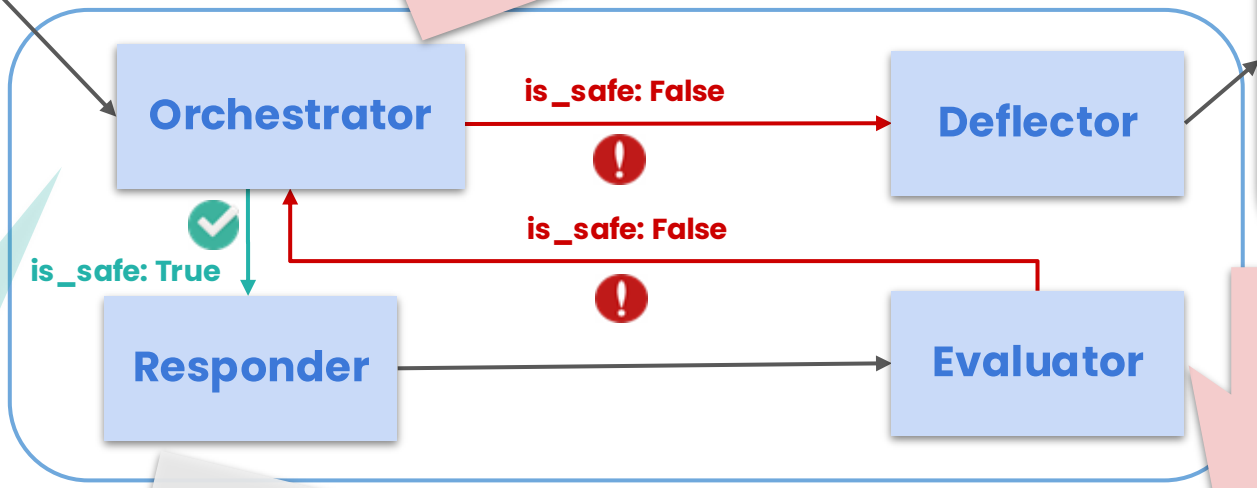
Agentic Workflow: a graph of LLM calls to answer the questions



Computation Graph

User Input:

What are some effective ways to visualize the stages of a cybersecurity incident?



Reasoning: The question was safe but the answer is unsafe.

Output 2:

I apologize, but I cannot provide information about security exploits or vulnerabilities.

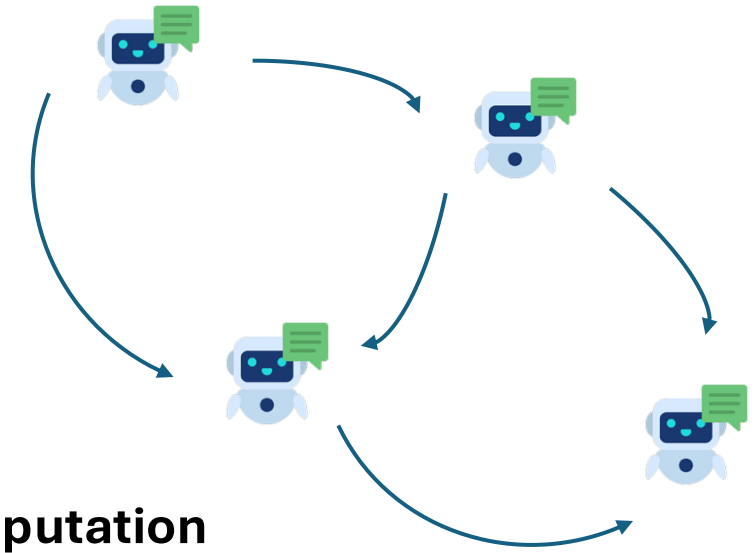
Reasoning: The question is related to cybersecurity. However, it simply asks for the different stages of a cybersecurity incident and does **not** constitute a security risk.

Visualizations can help. You could use the Cyber Kill Chain. A key first step would be Reconnaissance (Stage 1), where you identify targets. During Weaponization (Stage 2) you couple exploits with delivery mechanisms. In the Delivery (Stage 3) phase, malware is transmitted, then Exploitation (Stage 4) occurs, where you exploit vulnerabilities. ...

Reasoning: The message is particularly about the **Cyber Kill Chain** and explicitly lays out the core stages of the framework. Such content is **unsafe** given the security configurations.

Agentic Workflow and Autonomous Design

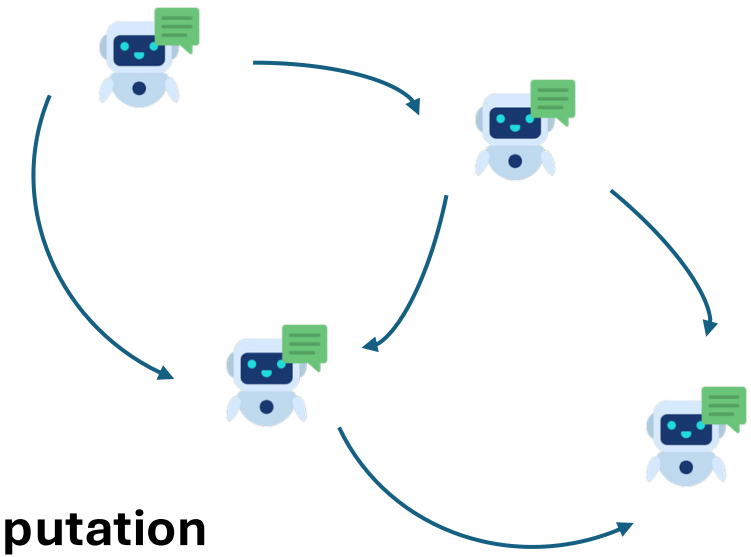
Agentic Workflow: a graph of LLM calls to answer the questions



Computation Graph

Agentic Workflow and Autonomous Design

Agentic Workflow: a graph of LLM calls to answer the questions



Computation Graph

Python Code

```python
CLARIFY_PROMPT = "Refine the problem understanding and clarify the requirements."

SOLVE_PROMPT = "Solve the problem step by step and provide a clear solution."

REFINE_PROMPT = "Refine the solution to fix any errors or improve clarity."


class Workflow:
    def __init__(self, name, llm_config, dataset):
        self.name = name
        self.dataset = dataset
        self.llm = create_llm_instance(llm_config)
        self.custom = operator.Custom(self.llm)
        self.custom_code_generate = operator.CustomCodeGenerate(self.llm)
        self.test = operator.Test(self.llm, dataset=self.dataset)

    async def __call__(self, problem: str, entry_point: str):
        # Clarify the problem, then generate an initial solution
        refined_problem = await self.custom(input=problem, instruction=CLARIFY_PROMPT)
        solution = await self.custom_code_generate(
            problem=refined_problem, entry_point=entry_point, instruction=SOLVE_PROMPT
        )
        test_result = await self.test(problem=problem, solution=solution, entry_point=entry_point)

        # Up to 3 refinement passes; stop early as soon as tests pass
        for _ in range(3):
            if test_result['result']:
                break
            refined_solution = await self.custom(
                input=test_result['solution'], instruction=REFINE_PROMPT
            )
            test_result = await self.test(
                problem=problem, solution=refined_solution, entry_point=entry_point
            )

        return test_result['solution'], self.llm.usage_tracker.get_summary()["total_cost"]
```

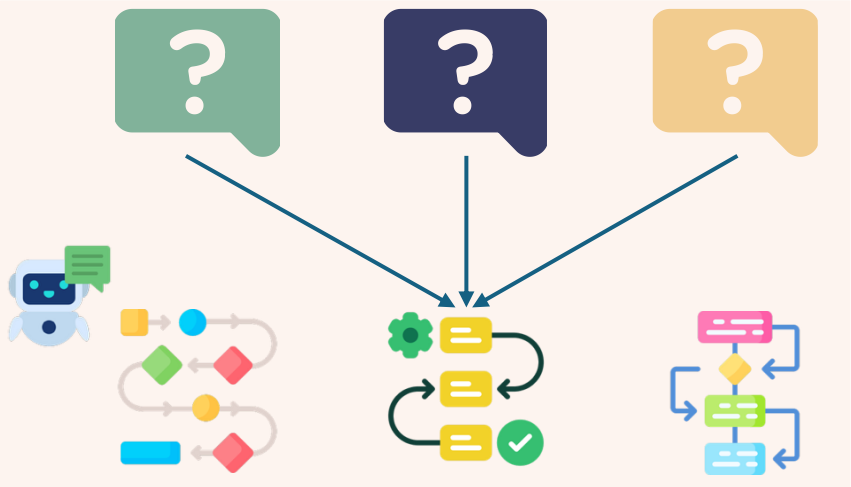
Code Representation

Meta Designer

Two Optimization Paradigm: Task-level v.s. Query-level

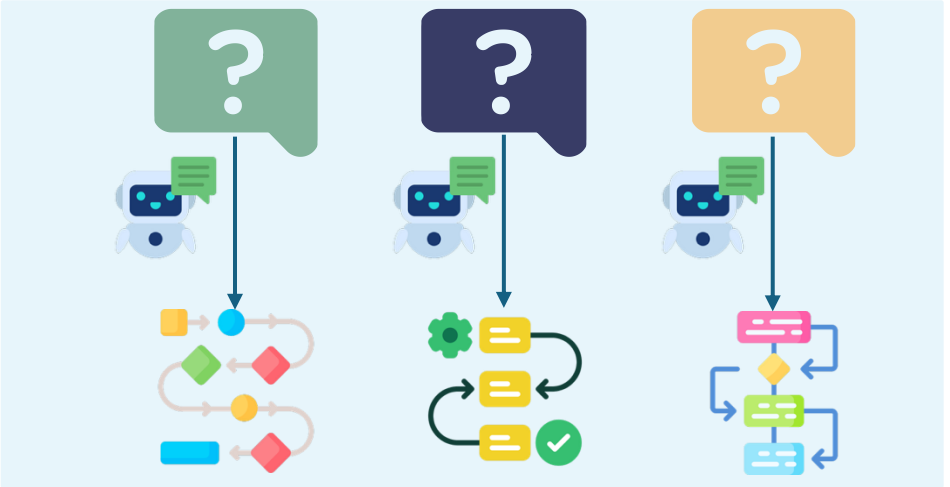
One for All

One Workflow for All Queries



One for Each

Dynamic Workflow for Each Query

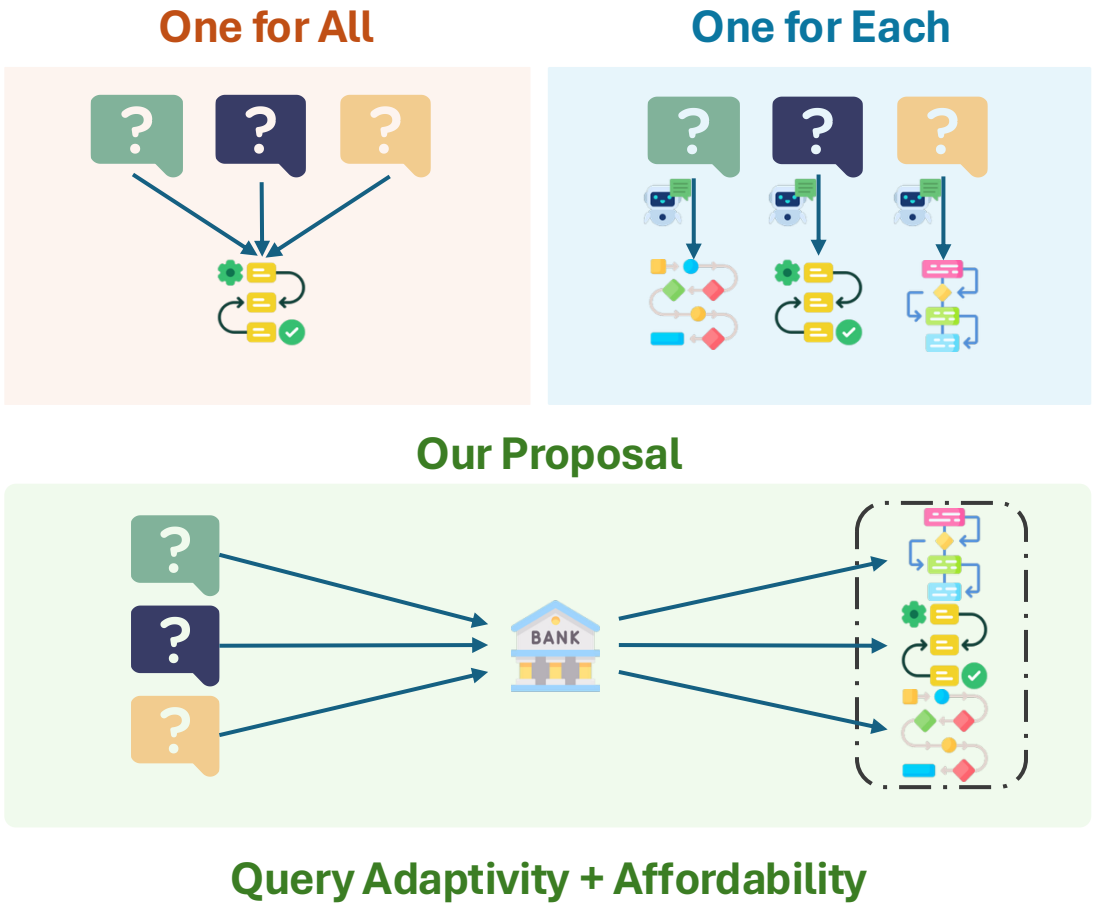


Lack Per-Query Adaptivity

Expensive to Learn

Can We Precompute Workflows in Training and Reuse in Deployment?

Can We Precompute Workflows in Training and Reuse in Deployment?



Challenges Solved:

(1) How to construct a bank?

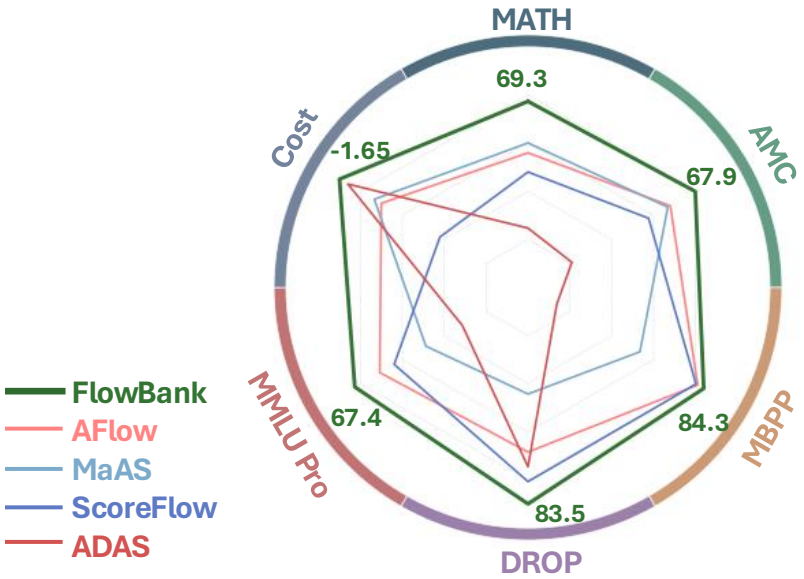
(2) How to choose during deployment?

YDYCRH, FlowBank: Query-Adaptive Agentic Workflows Optimization through Precompute-and-Reuse, arXiv:2606.11290.

Experiments – Main Results

Compared to task-level/query-level methods over 5 benchmarks

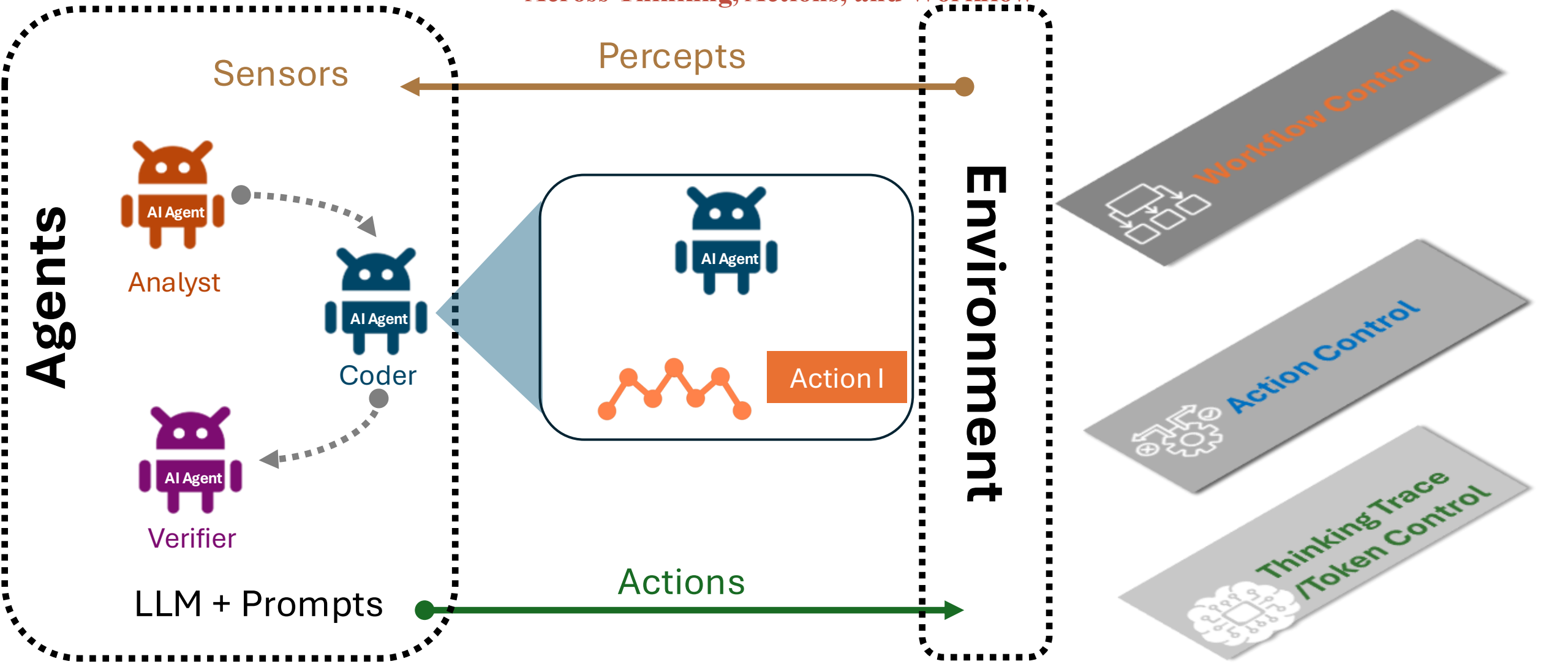
Method	MATH		AMC		MBPP		DROP		MMLU Pro		Average	
	Perf.	Cost	Perf.	Cost	Perf.	Cost	Perf.	Cost	Perf.	Cost	Perf.	Cost
IO	56.58	0.51	53.59	0.53	72.92	0.08	75.49	0.08	50.89	0.03	60.44	0.22
Manually Designed Agentic Workflows												
CoT	55.76	0.53	52.47	0.54	74.29	0.08	77.96	0.11	51.35	0.04	60.98	0.23
ComplexCoT	54.39	0.62	49.52	0.64	71.68	0.10	77.74	0.33	64.03	0.35	63.85	0.42
Self-Consistency	57.82	3.38	53.89	3.44	73.98	0.52	78.33	0.74	50.93	0.34	61.49	1.51
Multi-agent Debate	58.92	7.66	53.82	7.96	74.68	0.86	78.65	1.64	57.56	0.88	63.86	3.45
Self-Refine	55.14	1.08	52.88	1.06	71.07	0.27	76.71	0.34	43.75	0.46	57.96	0.62
MedPrompt	56.69	5.64	55.19	5.49	71.85	0.61	81.77	1.75	52.79	0.63	62.77	2.58
MultiPersona	57.41	0.56	53.74	0.58	70.38	0.28	76.13	0.29	61.70	0.36	63.87	0.41
Automated Agentic Workflow Optimization Methods												
GPTSwarm	54.94	1.69	55.42	9.04	71.26	**0.25**	78.37	**0.51**	58.37	1.14	63.43	2.54
ADAS	56.17	2.95	47.33	3.22	72.34	0.77	81.17	0.98	59.71	1.05	63.22	1.71
AgentSquare	62.76	**1.35**	58.78	**1.79**	72.73	0.52	76.84	0.76	63.75	**0.75**	66.70	**1.02**
AFlow (Qwen3-8B)	63.24	3.23	62.75	2.51	79.37	1.02	77.08	1.05	64.62	1.47	68.56	1.79
AFlow (GPT-4o)	63.99	2.77	63.77	4.86	83.77	0.57	80.26	1.02	65.61	0.89	70.40	1.95
MaAS	65.02	2.43	63.36	2.70	79.08	2.04	76.64	1.64	62.31	1.29	68.09	1.90
ScoreFlow	62.00	2.36	60.15	3.51	83.63	1.70	82.10	1.42	64.58	2.62	69.51	2.37
FlowBank	**69.34**	1.78	**67.94**	2.49	**84.26**	1.62	**83.49**	1.06	**67.40**	1.52	**73.40**	1.65



YDYCRH, FlowBank: Query-Adaptive Agentic Workflows Optimization through Precompute-and-Reuse, arXiv:2606.11290.

Toward Self-Improving Agentic Systems

Across Thinking, Actions, and Workflow



Toward Self-Improving Agentic Systems

Across Thinking, Actions, and Workflow

1. **[Transfer Q*]** Chakraborty, Ghosal, Yin, Manocha, Wang, Bedi, and Huang. "Transfer Q-star: Principled Decoding for LLM Alignment." NeurIPS 2024.
2. **[GenARM]** Xu, Sehwag, Koppel, Zhu, An, Huang, and Ganesh. "GenARM: Reward Guided Generation with Autoregressive Reward Model for Test-time Alignment." ICLR 2025.
3. **[ACT]** Liu, Liu, Ho, Chakraborty, Wang, and Huang. "Agentic Critical Training." arXiv:2603.08706.
4. **[FlowBank]** Yuan, Deng, Yu, Chakraborty, Rostami, and Huang. "FlowBank: Query-Adaptive Agentic Workflows Optimization through Precompute-and-Reuse." arXiv:2606.11290.
5. **[AegisLLM]** Cai, Shabihi, An, Che, Bartoldson, Kailkhura, Goldstein, Huang. "AegisLLM: Scaling Agentic Systems for Self-Reflective Defense in LLM Security." ICLR workshop 2025.

furongh@umd.edu https://furong-huang.com/ X: @furongh