Adaptive Agents under Limited Feedback

From Flat In-Context Adaptation to Structured Context

Quanming Yao

Associate Professor

Department of Electronic Engineering,

Tsinghua University

Tutorial

From Predicting Answers to Completing Tasks

Evolution of Task Forms

Preditction Tasks

Example:
Image classification

Task feature:
Recognize and classify, output a fixed category

2012

Content Generation

Example:
Code generation

Task feature:
Follow instructions to generate structured content

2021

Multi-turn Interaction

Example:
Conversational assistant

Task feature:
Engage in multi-turn dialogue and track broader context

2022

Tool Execution

Example:
Laboratory operation

Task feature:
Invoke tools and complete concrete operations

2023

System Coordination

Example:
Multi-agent coordination

Task feature:
Coordinate multiple systems and optimize the whole

2024

Long-horizon Agent

Example:
Large-scale satellite control

Task feature:
Long-term planning, autonomous decision-making and sustained execution

2025

Adaptation must happen throughout interaction for agentic tasks.

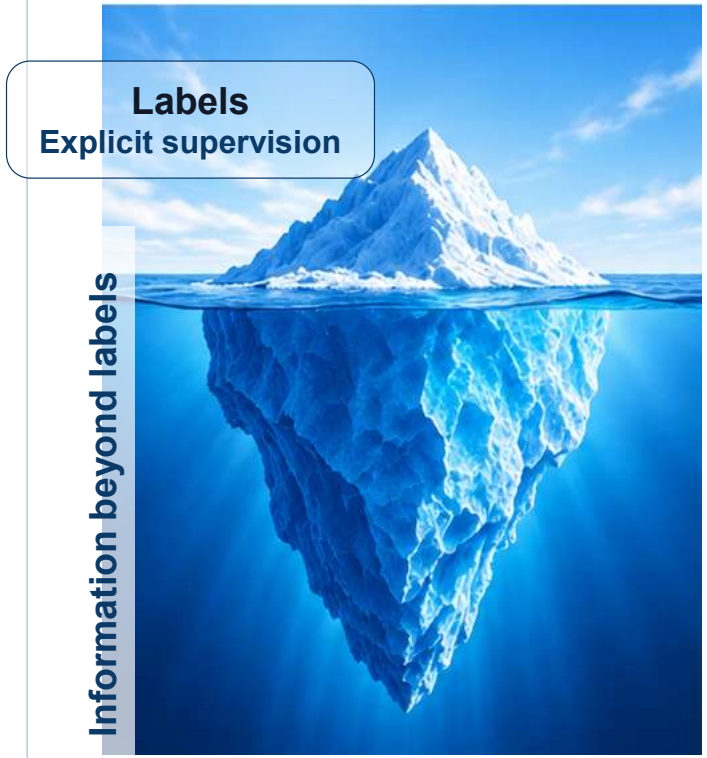
Starting from a Simple Example

Find a mug in an unfamiliar kitchen, clean it if necessary, fill it with water, and place it on the dining table.

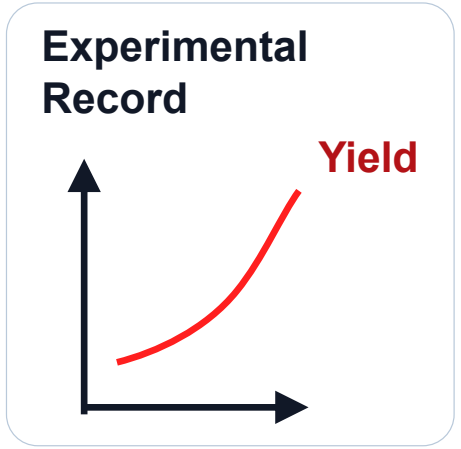
Even a simple task exposes state, constraints, and intermediate observations.

Limited Supervision Does Not Mean Limited Information

Labels: tip of the iceberg



Example: an agent optimizing chemical reaction conditions



Rich Information Beyond Labels

Temperature, concentration, pH, experimental process

Limited Explicit Supervision

Final yield, success/failure outcomes

State	Current temperature, pressure, concentration, pH, liquid level, stirring speed
Relations	Dependencies among reagents, solvents, catalysts, mixing ratios
Process	Sample addition, heating, temperature holding, stirring, sampling, observation

Limited supervision still produces rich interaction information—but how should an agent use it?

Outline

1. **How can a frozen LLM become an adaptive agent?**
2. Why is more context not enough?
3. What does Structured Context change?
4. How does structure improve data-efficient agentic learning?
5. How does structured context enable system-level adaptation?
6. What should we take away—and what remains open?

From Invocation to Interaction

One-shot LLM

single-pass Inference

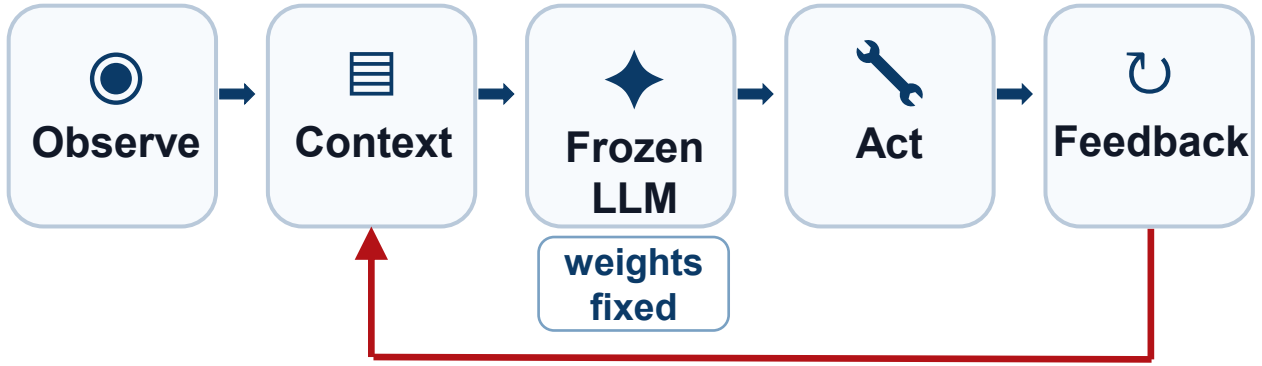
Input ➡ LLM ➡ Output

The model receives an input once and returns an output once.

interaction

Agent Loop

closed-loop interaction

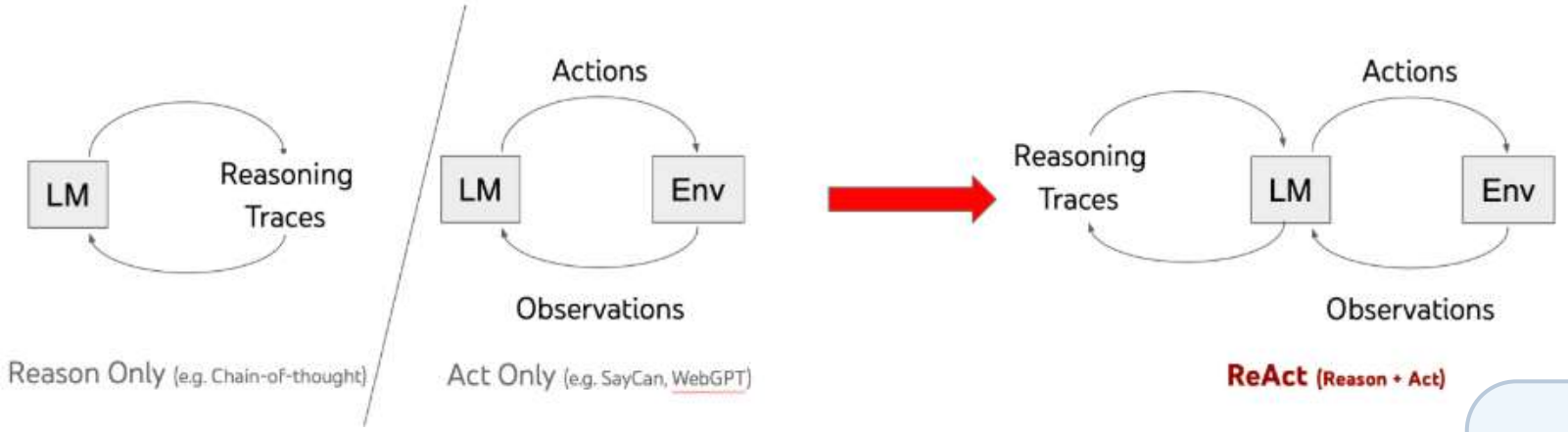


F What stays fixed?
The model parameters.

↻ What changes during interaction?
Observations, state, memory, and feedback

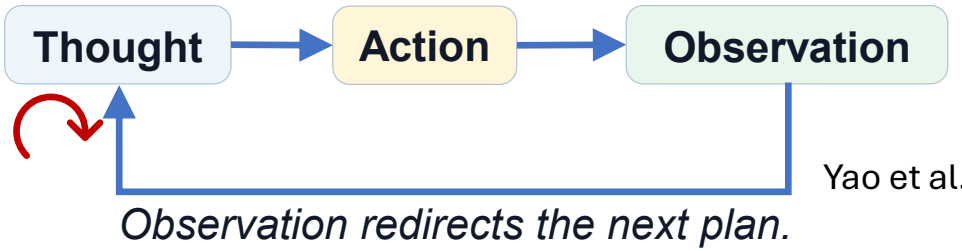
A frozen LLM becomes an agent when one-shot generation turns into a closed interaction loop.

New Observations Redirect the Current Trajectory



Act-only repeats invalid actions; ReAct uses each observation to revise the plan before the next action.

Core mechanism



Yao et al. ICLR, 2023.

Observation redirects the next plan.

- **Thought** maintains and revises a high-level plan.
- **Action** queries tools or changes the environment.
- **Observation** becomes new evidence for the next decision.

Mug example

1 **Action** Inspect counter
Observation No mug found

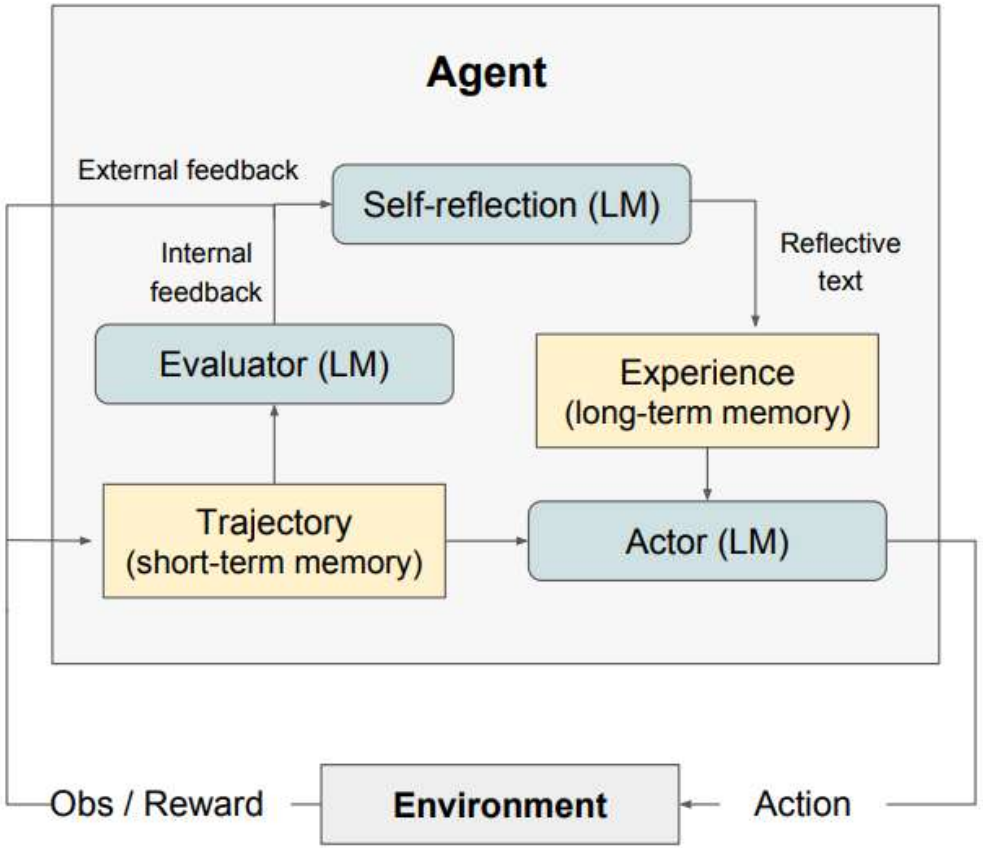
→ **Redirect: revise the location hypothesis → search another likely location**

2 **Action** Inspect cabinet
Observation Mug found; appears dirty

→ **Redirect: clean before filling**

ReAct supports within-trajectory adaptation:
each observation updates the next decision in the same attempt.

Trajectory-Level Feedback Enables Reflection



Shinn et al. *Reflexion: Language Agents with Verbal Reinforcement Learning*. NeurIPS, 2023.

Core mechanism

Trajectory → **Evaluation** → **Reflection** → **Memory**

- The actor generates a trajectory.
- An evaluator turns the outcome into feedback.
- Self-reflection writes an actionable verbal lesson.
- The lesson is stored in external memory and conditions the next trial.

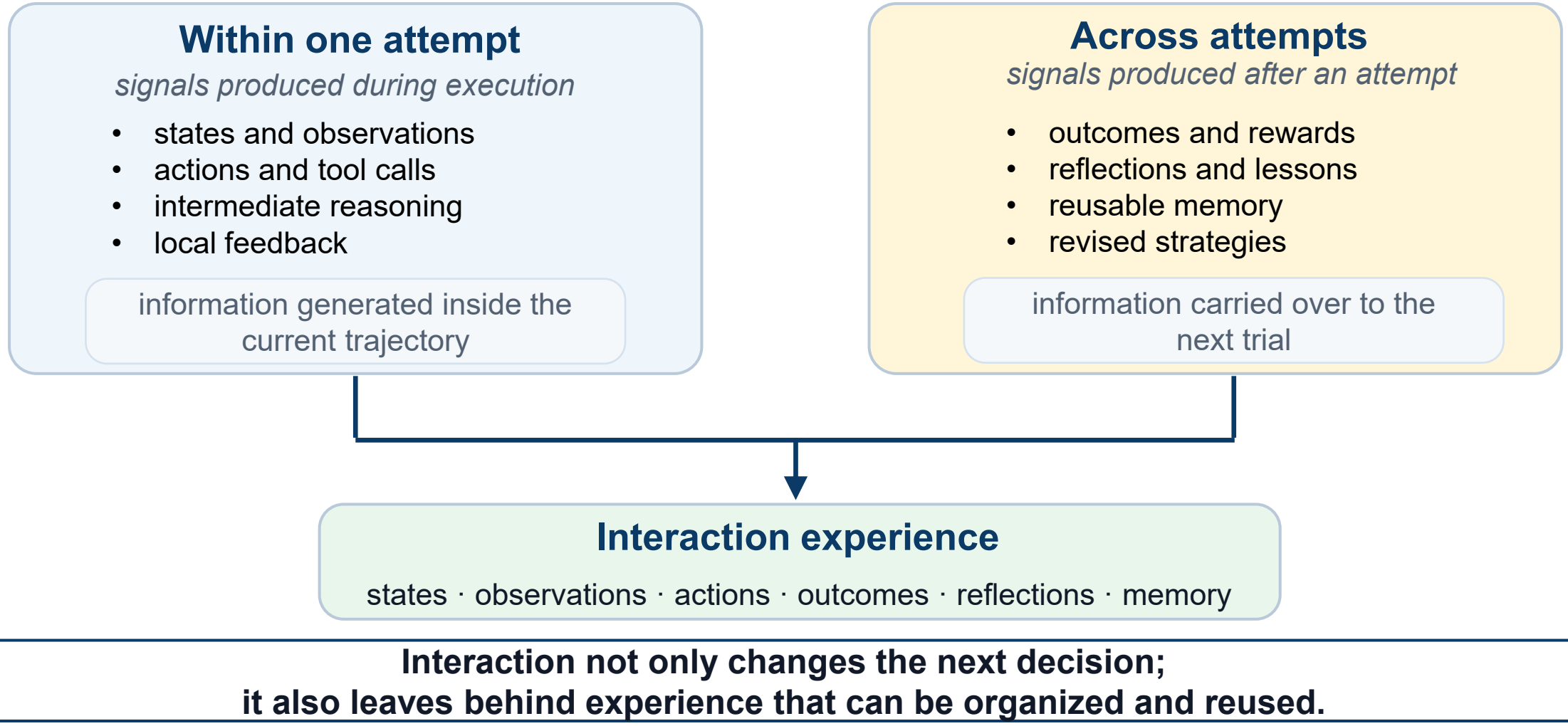
Mug example

Failed attempt	Locate → Fill → Place → Failure · cleanliness missed
Feedback	mug was placed, but cleanliness requirement was not satisfied.
Reflection	Inspect required states and satisfy unmet preconditions
Next attempt	Locate → Inspect → Clean if needed → Fill → Place → Verify

Reflexion supports across-attempt adaptation:
task-level feedback becomes external memory for the next trial.

What Does Interaction Leave Behind?

Adaptation changes decisions, and interaction also produces reusable experience.

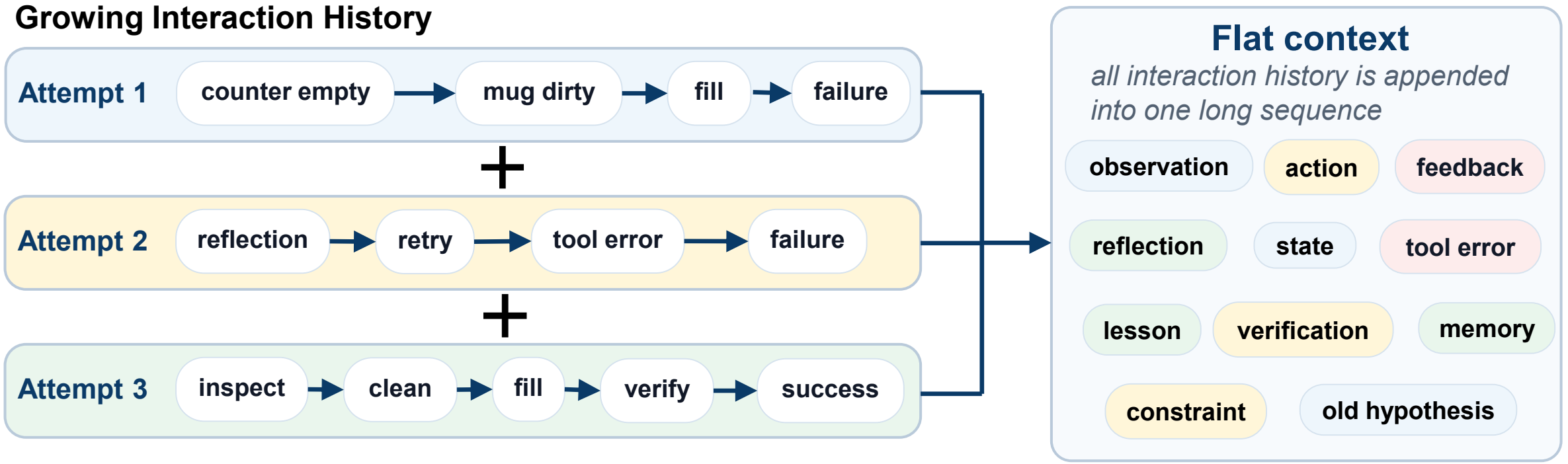


Interaction not only changes the next decision;
it also leaves behind experience that can be organized and reused.

Outline

1. How can a frozen LLM become an adaptive agent?
2. **Why is more context not enough?**
3. What does Structured Context change?
4. How does structure improve data-efficient agentic learning?
5. How does structured context enable system-level adaptation?
6. What should we take away—and what remains open?

More Context does not Mean Better Context



More information ✓ Easy usage ✗

Simply appending interaction history gives the model more tokens, not a better representation of the task.

Different Information Has Different Roles

A flat sequence records order, but it does not reveal what each piece of information means.

One flat context

- Mug must be clean and placed.
- Mug may be on the counter.
- Counter is empty.
- Mug appears dirty.
- Task failed.
- Verify required states.

All sentences appear in the same sequence.

interpret by role

Role-based interpretation

Constrain	Mug must be clean and placed.
Hypothesis	Mug may be on the counter.
Observation	Counter is empty.
State	Mug appears dirty.
Feedback	Task failed.
Lesson	Verify required states.

Different roles require different interpretation, update, and retention.
A flat sequence preserves when information appears, but not what role it plays.

Where Did the Failure Come From?

Execution chain

Locate → Fill → Place → Verify → Failure

Mug is placed correctly, but it is still dirty

Which component should be updated?
This is the credit-assignment problem.

Planning
Did the plan omit inspection and cleaning?

Memory
Was the previous lesson not retrieved?

Tool / Skill
Was cleaning unavailable or incorrectly executed?

Verification
Did the verifier check placement but not cleanliness?

Without structure, sparse feedback is difficult to localize to the plan, memory, tool use, or verifier that caused the failure.

Raw Trajectories Lead to Information Overload

What the raw trajectory contains


Search counter → no mug found


Open cabinet → mug found


Navigate to sink → tool fails once


Retry cleaning → fill water


Walk to table → place mug

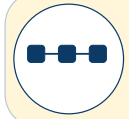

Run final verification

abstract →

What may transfer


Clean before fill.


Verify every required object state before completion.


Locate → Inspect → Prepare → Manipulate → Verify

Transfer examples

dirty plate → wash before serving

Empty bottle → fill before delivery

A long trajectory mixes task-specific actions, outcomes, and errors.

Complex experience is abstracted into reusable structure.

The whole trajectory is task-specific, but only part of it should be abstracted and transferred.

Outline

1. How can a frozen LLM become an adaptive agent?
2. Why is more context not enough?
3. **What does Structured Context change?**
4. How does structure improve data-efficient agentic learning?
5. How does structured context enable system-level adaptation?
6. What should we take away—and what remains open?

From Dispersed Experience to Usable Structure

Dispersed experience

"Mug may be on the counter."

"Counter is empty"

"Mug appears dirty."

"Task failed."

"Verify all required states."

Useful signals exist, but they are still scattered.

Organize the information

Identify the role of each item

state · belief · constraint
feedback · lesson

Connect related items

evidence · dependency
state transition

Roles and relations organize scattered information into structure.

Usable Structured Context

Goal clean, fill, and place the mug

Workflow

Locate → Inspect → Clean if needed → Fill → Place → Verify

Lesson Verify all required states before completion.

Structured Context organizes weak and dispersed signals into reusable task-relevant structure.

What Information Forms Structured Context?

Attributes / State
properties, profiles, descriptors
Example: mug is dirty

Relations
graphs, interactions, dependencies
Example: clean before fill

Interaction History
sequences, traces, feedback context
Example: counter empty; prior attempt failed

Task Structure
demonstrations, workflows, action constraints
Example: Locate → Inspect → Prepare → Verify

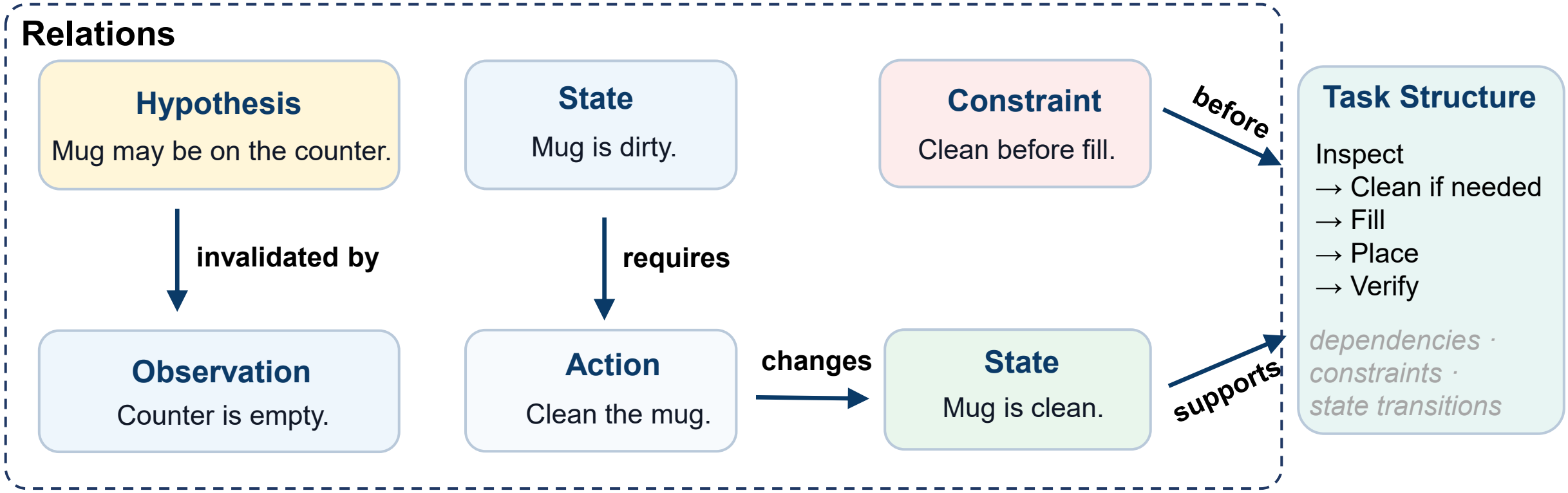
organize

Structured Context
organizes task-relevant signals
into explicit, reusable structure

Current state	mug is dirty
Dependency	clean before fill
History	previous attempt failed
Workflow	Locate → Inspect → Prepare → Verify

Structured Context is broader than labels: it includes attributes, relations, interaction history, and task structure.

Relations Turn Context Items into Task Structure



Types identify the pieces; relations connect them into a task structure the agent can reason over.

How Structured Context Guides the Agent

Task Structure from Relations: Inspect → Clean if needed → Fill → Place → Verify

1. Select

retrieve what matters now

State: mug is dirty

Constraint: mug must be clean

Relation: clean before fill

Lesson: verify required states

Select pulls the relevant signals for the current task.

2. Compose

organize current behavior

Goal:
clean + filled + placed

Goal-focused workflow
Inspect → Clean → Fill → Place → Verify

Verifier: cleanliness + placement

Compose turns selected items into a concrete plan.

3. Execute

follow the task workflow

Current execution uses structured guidance.

The workflow is easier to follow because key conditions are explicit.

4. Update

revise context after feedback

clean(mug): dirty→clean

If failure persists: refine lesson

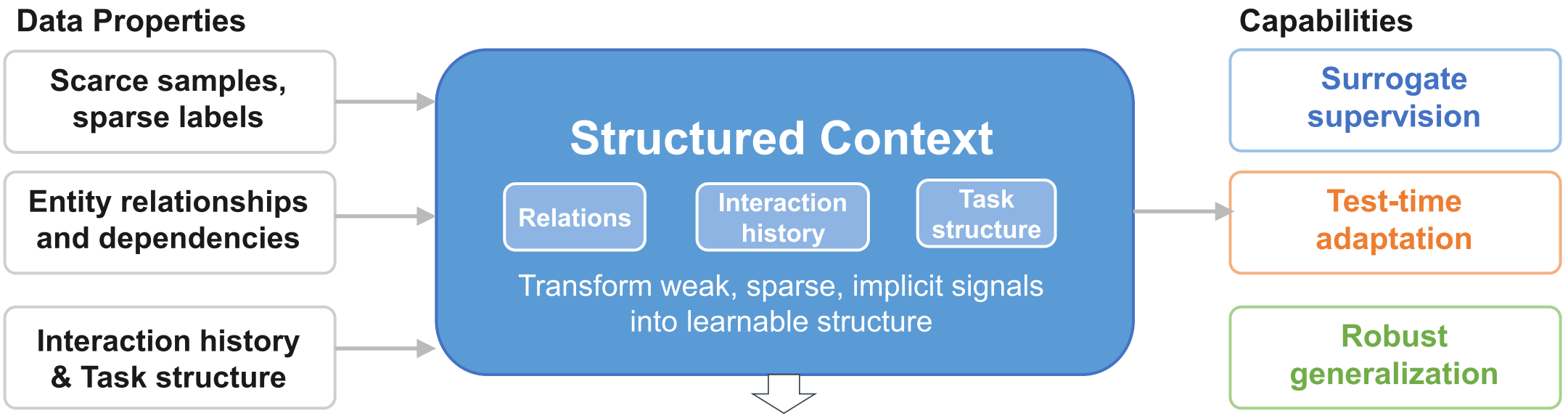
If state changes: refresh context

Store reusable experience

Update feeds new experience back into Structured Context.

Structured Context guides what the agent retrieves, how it acts, and what it updates after feedback.

Core Concept: What is Structured Context?



Structured Context for the Mug Task

State-aware execution · Explicit constraints · Reusable workflow

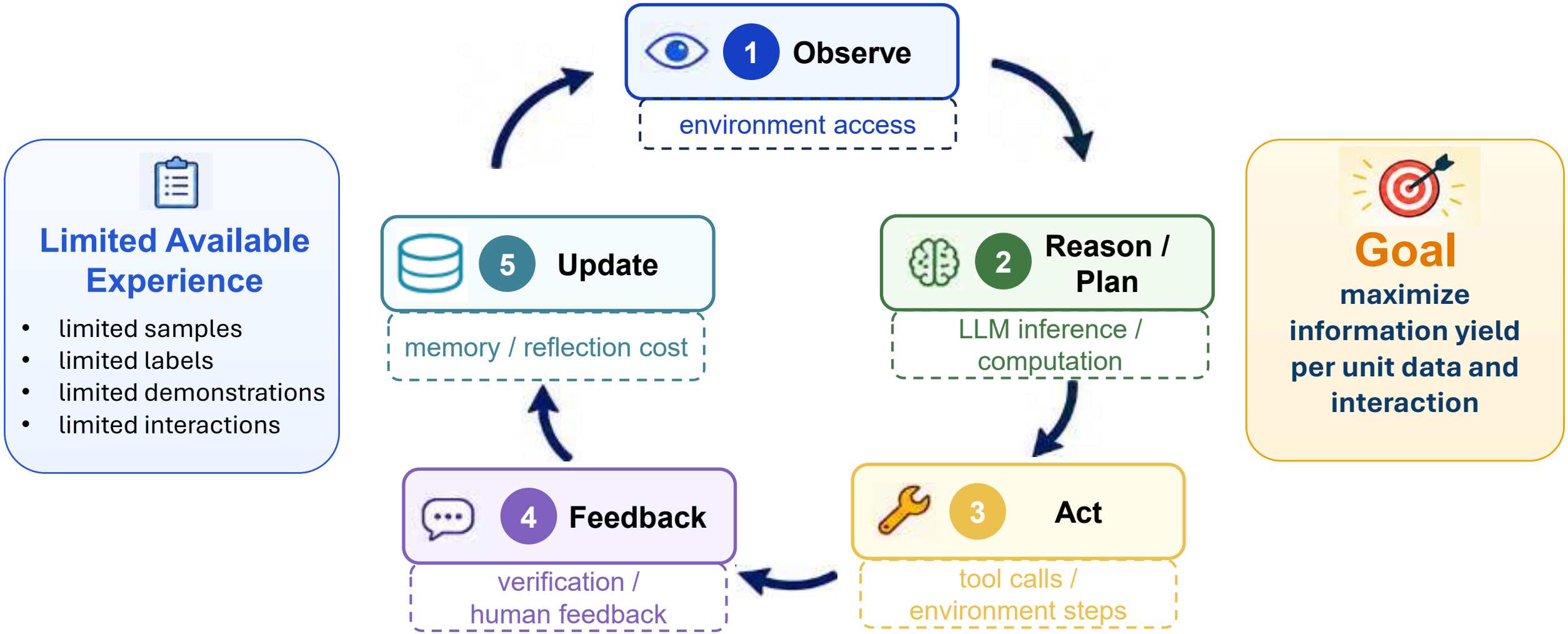
State / observations: counter empty, mug dirty, mug clean after washing
Interaction history: failed counter search, prior attempt missed cleanliness, failure feedback from verification
Task structure: inspect before acting, clean before fill, verify cleanliness and placement before completion

Structured Context organizes dispersed interaction signals into explicit, editable task structure.

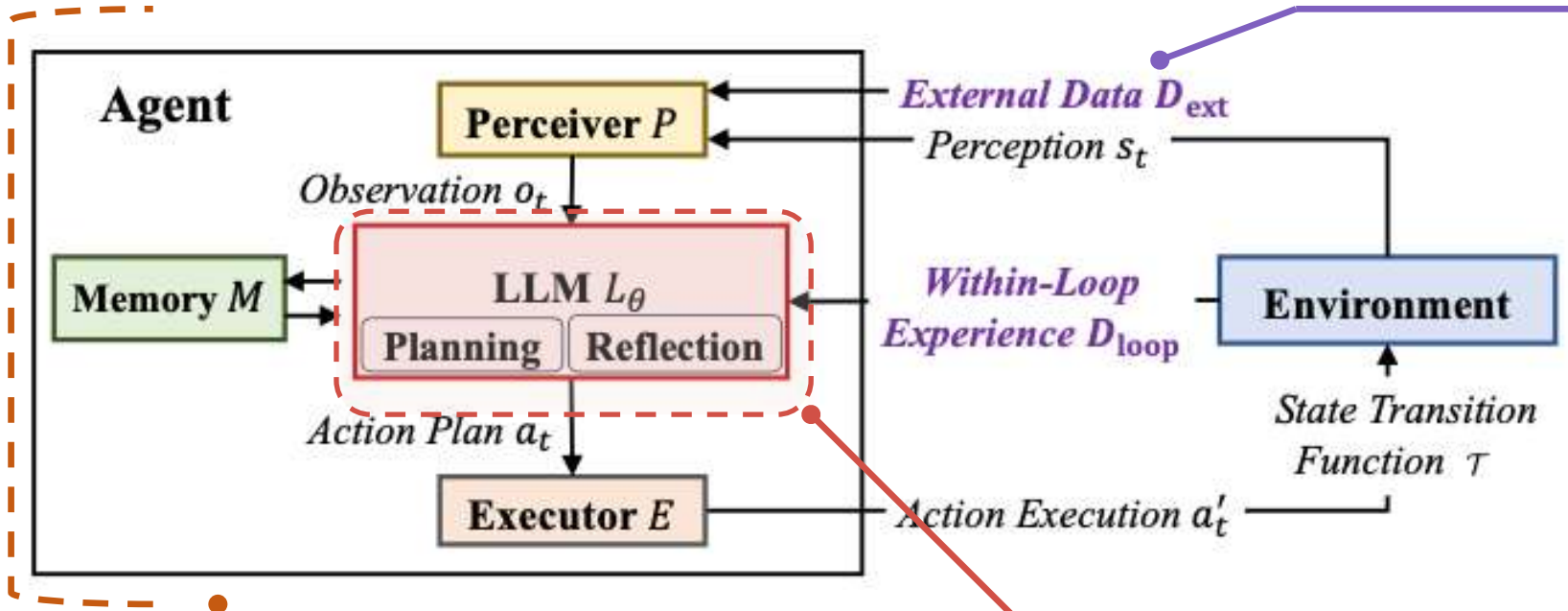
Outline

1. How can a frozen LLM become an adaptive agent?
2. Why is more context not enough?
3. What does Structured Context change?
4. **How does structure improve data-efficient agentic learning?**
5. How does structured context enable system-level adaptation?
6. What should we take away—and what remains open?

Every Interaction Has a Cost



Three Design Levers for Data-Efficient Agents



Experience

What signals enter the loop - and what can be retained or reused?

· trajectories · demonstrations · feedback · prior attempts

Agent Structure

How is the loop organized to make each interaction more useful?

· perceiver · memory · planning · reflection · executor

Adaptation

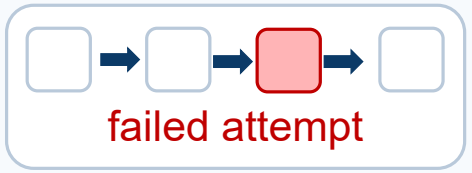
What changes when limited experience should improve future behavior?

· context · parameters · policy

Wang et al. *Beyond Scaling: A Survey of Data-Efficient Learning for LLM Agents*. IJCAI, 2026.

Past Experience Becomes Reusable Guidance

Past experience



State Tool

Select & Abstract

Find what is useful and generalize it.

Output
- Lessons
- Patterns
- Rules

Organize & Store

Keep it in a compact and structured form.

Output
- Memory
- Skill
- Knowledge base

Retrieve & Apply

Find and apply what matters now

Output
- Lessons
- Rules
- Patterns

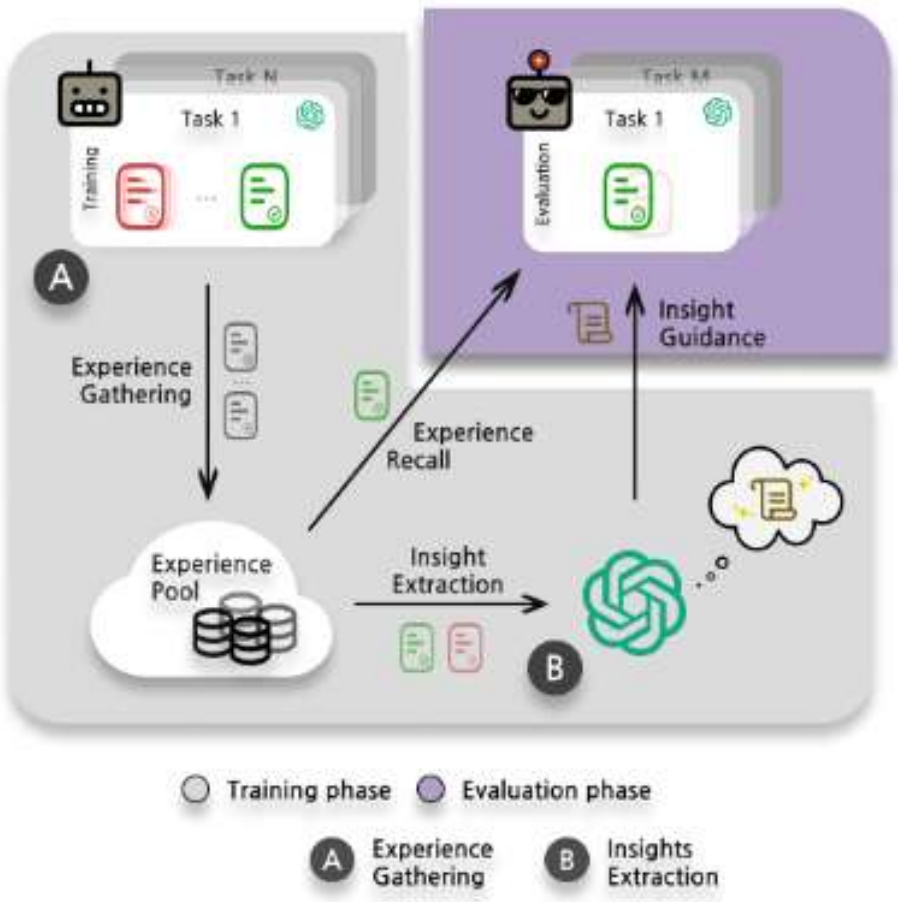
Guide for a new task

Reusable lesson / procedure
compact and task-relevant

New task
uses the retrieved guidance to act

Past experience becomes valuable when it can be effectively reused for future decisions.

From Past Experience to Better Decisions on a New Task



Gather Experience

Experience Pool
successful & failed trajectories

Task A — Failed
Put a clean, water-filled mug

Locate → Fill → Place

✕ **Cleanliness requirement missed**

Task B — Success
Locate → Inspect → Clean → Fill → Place → Verify

✓ **Required object states satisfied**

distill

Extract Insights

Cross-Task Insights
reusable natural-language guidance

General Insight
Check required object states and satisfy unmet preconditions before manipulation.

Recall and Reuse

New Task: Put a clean plate on the dining table.
Current state: plate = dirty

Retrieved trajectory + **Insight**

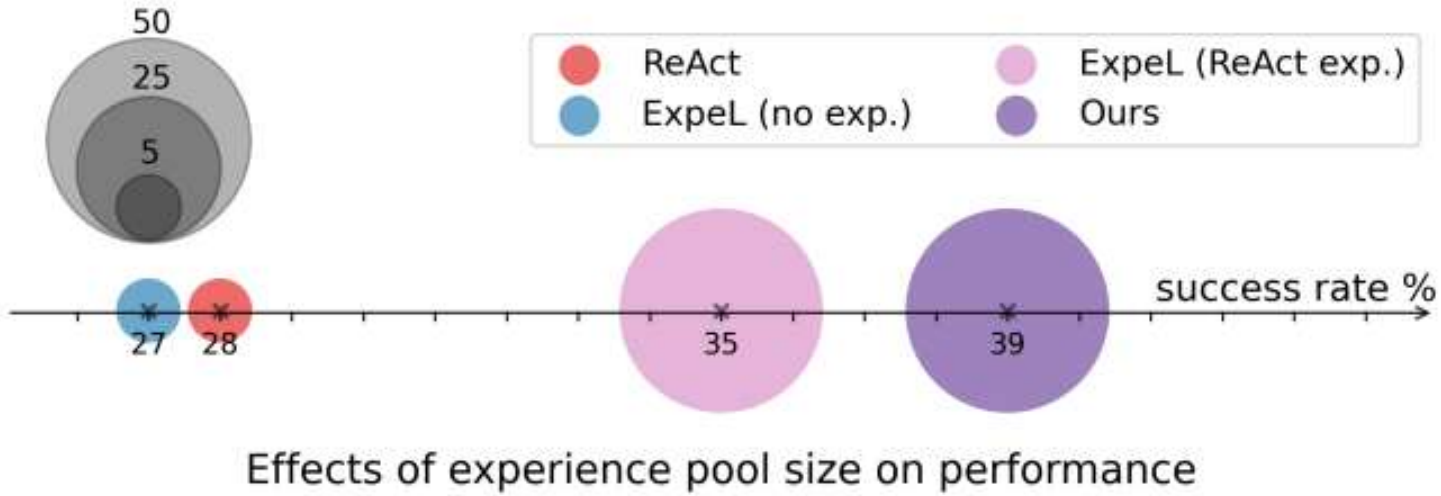
Locate → Inspect → Clean → Place → Verify

✓ **Success**

Experience becomes reusable at two levels: specific trajectories and general insights.

Zhao et al. *ExpeL: LLM Agents Are Experiential Learners*. AAAI, 2024.

Richer Experience Improves Downstream Performance



Effects of experience pool size on performance

More experience provides stronger learning signals.

Extra autonomously collected experience substantially improves downstream performance.

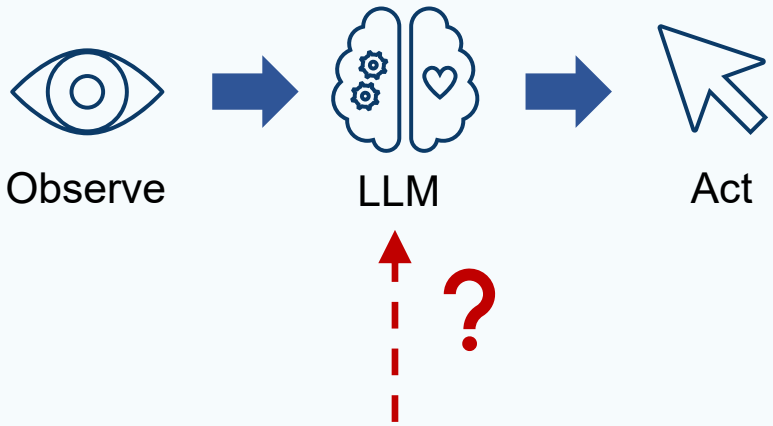
Diversity matters, not only quantity.

Success–failure pairs provide richer experience than successful trajectories alone.

Data efficiency depends on extracting richer learning signals from limited interactions.

The Agent System Is Also Something to Structure

Monolithic Agent

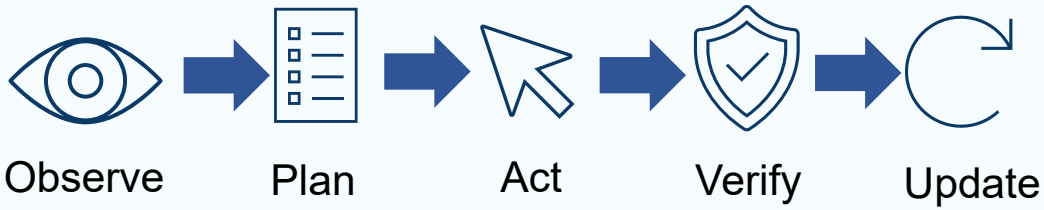


Hard to localize errors.

One mistake can affect the entire trajectory.

Structure the agent system

Structured Agent



Modules

Decouple roles to localize decisions
- Planner
- Executor
- Verifier

Procedure

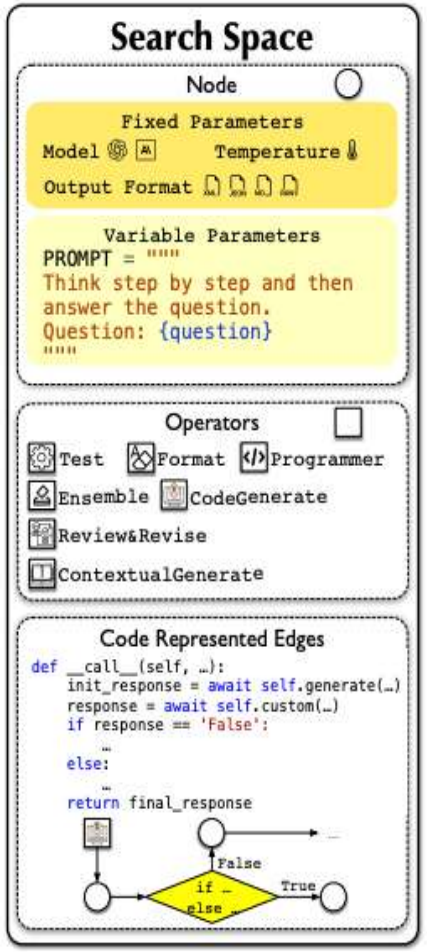
Constrain the execution flow to reduce waste
- Observe → Plan
- Act → Verify
- Iterate when needed

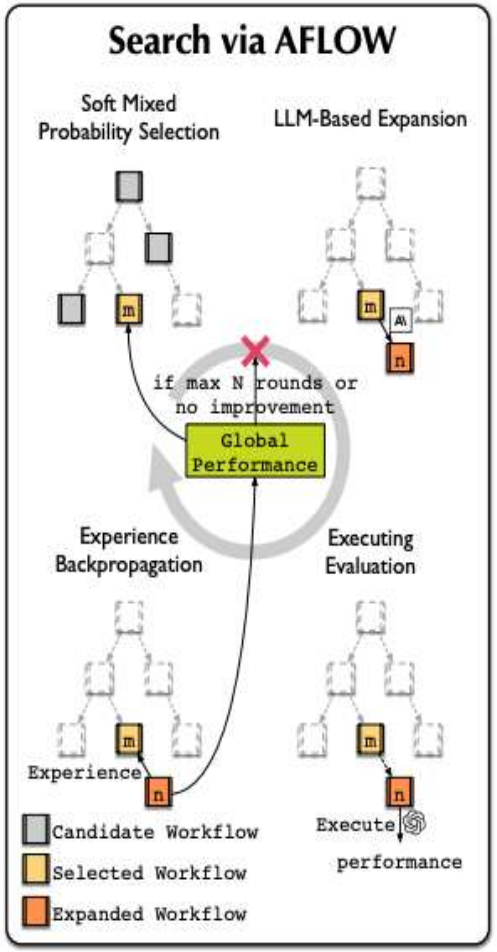
Reusable Structure

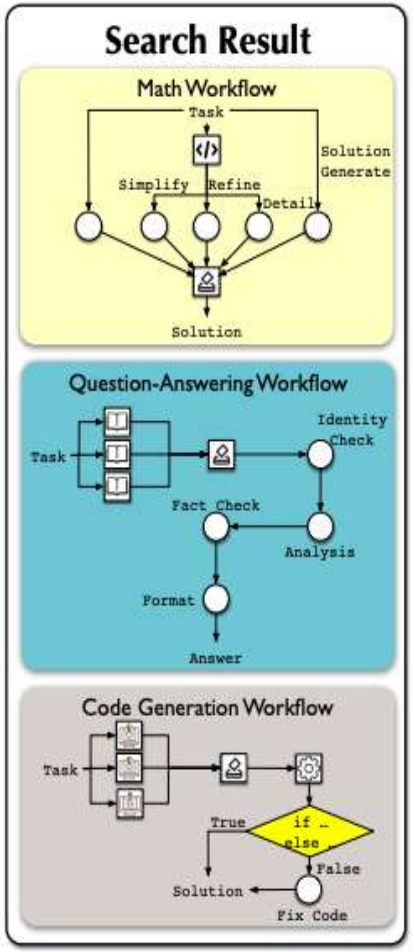
Reuse prior solutions to avoid redundant exploration
- Memory
- Workflows
- Skills

Structure makes each interaction more directed, verifiable, and reusable.

The Workflow Becomes Editable







Task
Put a clean, water-filled mug on the table.

1. Initial workflow

Plan → **Execute**

Locate → Fill → Place ✕ **Failure: not check**

2. Aflow workflow search

Propose candidates → **Execute** → **Evaluate** → **Keep better workflow**

Candidate A: Plan → Execute → Verify
Candidate B: Plan → Review → Execute
Candidate C: Plan → Check Preconditions → Execute → Verify ✓

3. Selected workflow

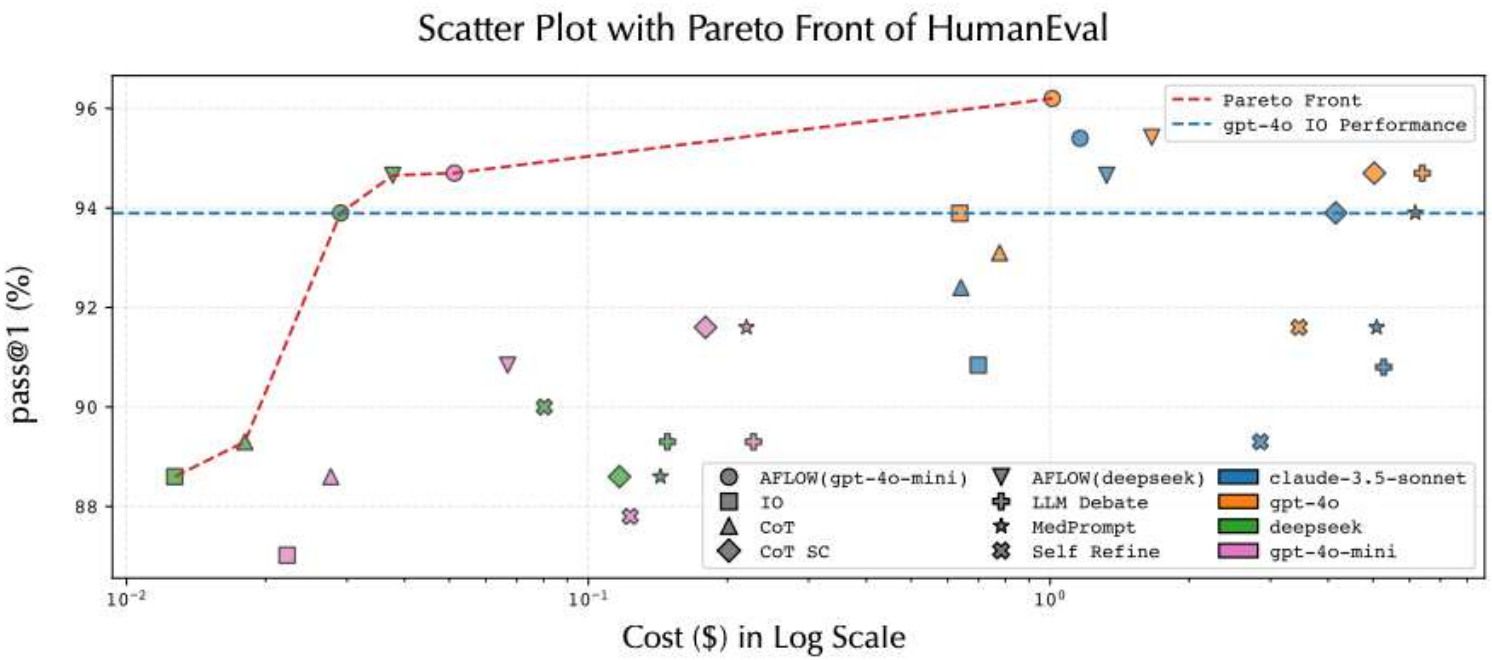
Plan → **Check Preconditions** → **Execute** → **Verify**

Locate → Inspect → Clean → Fill → Place → Verify
Success: ✓ clean ✓ filled ✓ placed

AFlow makes agent structure itself an explicit object of optimization.

Zhang et al. *AFlow: Automating Agentic Workflow Generation*. ICLR, 2025.

Better Structure Makes Limited Inference More Effective



1. Same task, different efficiency

Workflow design changes how much performance is obtained from each inference budget.

2. Smaller models can become competitive

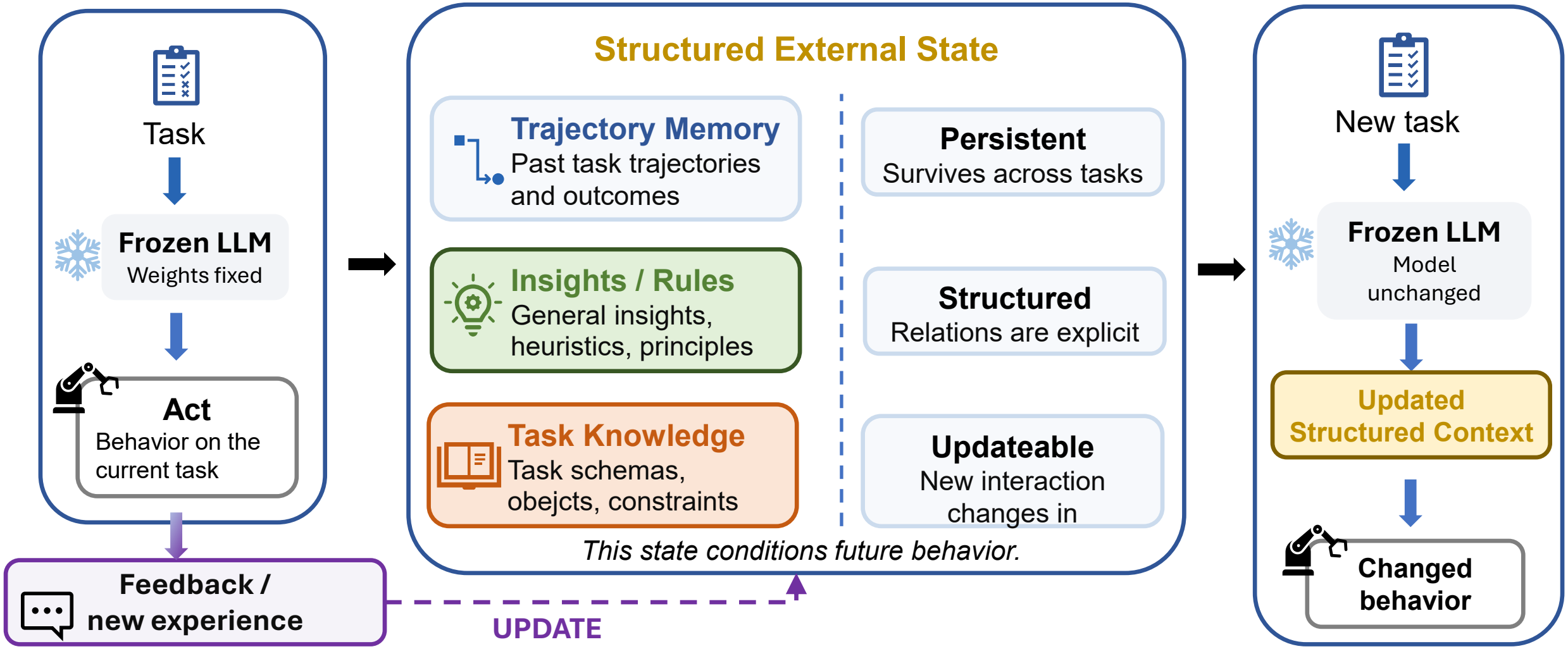
An optimized workflow can let a cheaper model match or outperform a stronger model.

3. Efficiency comes from organization

The gain is achieved by restructuring model calls rather than increasing model size.

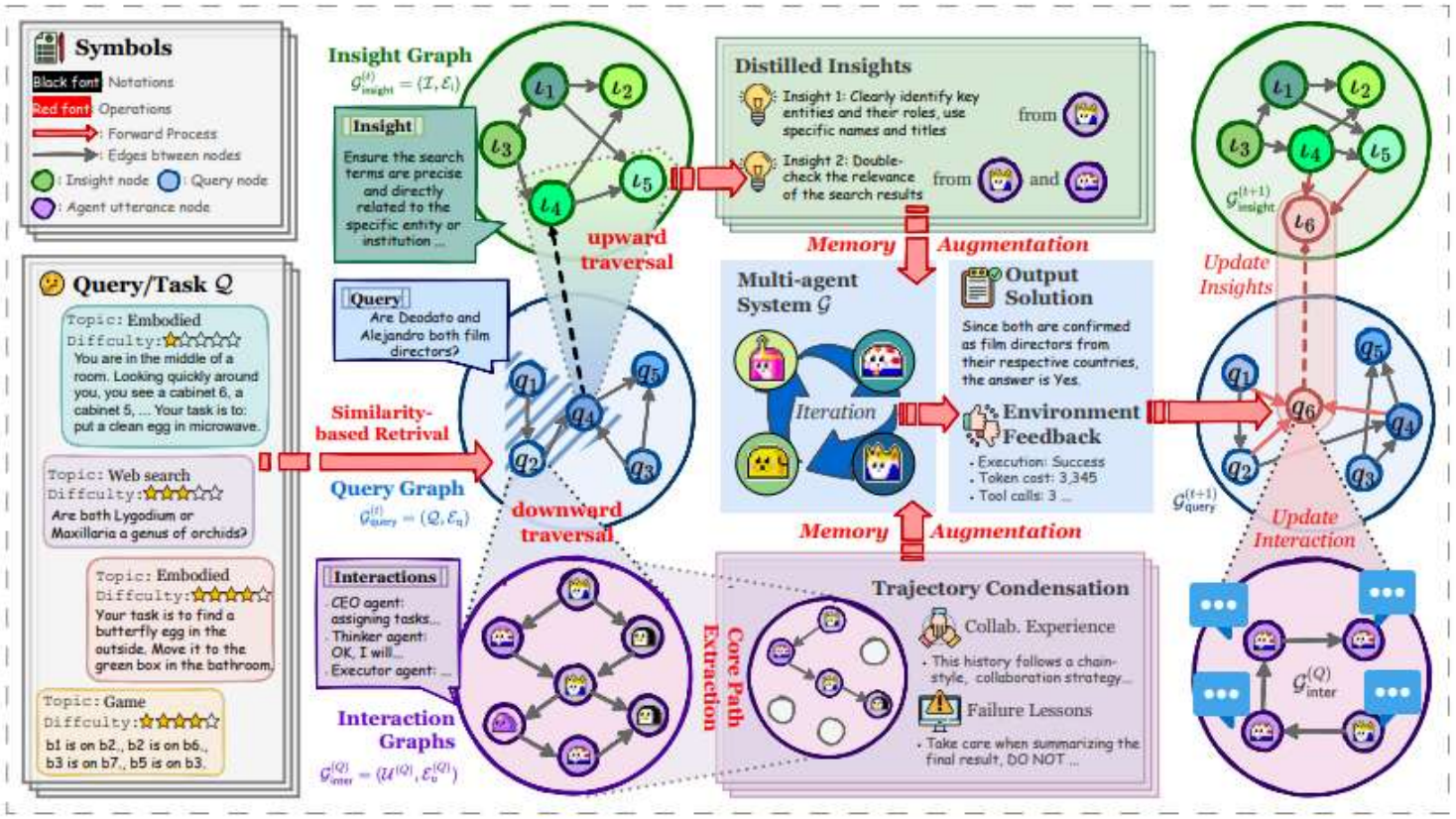
Structured workflows increase the value obtained from a limited execution budget.

A Frozen Model Can Adapt Through Structured Context



Adaptation occurs when interaction changes the structured external state used by future tasks.

Structured Memory Evolves Across Tasks



retrieve structured memory→act with memory augmentation→write the new interaction

Zhang et al. *G-Memory: Tracing Hierarchical Memory for Multi-Agent Systems*. NeurIPS, 2025.

Task
Put a clean, water-filled mug on the table.

Insight Graph: What general lesson transfers?
Satisfy required object states before the final action.

Query Graph:Which past tasks are related?
Put a filled cup on the desk
Put a clean mug on the table

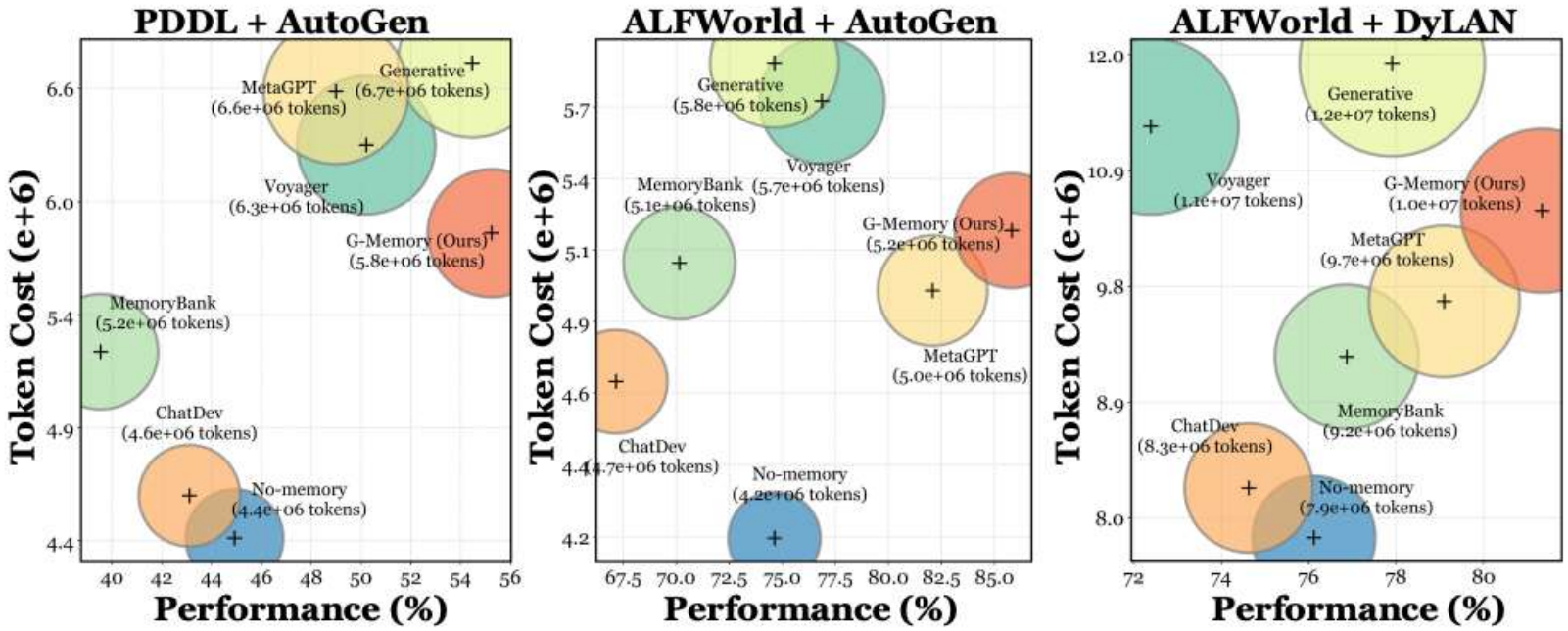
Interaction Graph: How were similar tasks handled?
Inspect → Clean → Place
Inspect → Fill → Place

Memory-augmented execution
Locate → Inspect → Clean → Fill → Place → Verify
Success: the mug is clean, filled, and placed.

New interaction is written back

Structured memory enables adaptation by turning each new interaction into knowledge that can guide future tasks.

Structured Memory Improves Performance Efficiency



Higher performance without proportional context growth

G-Memory gains performance while keeping token consumption controlled.

More memory is not the objective

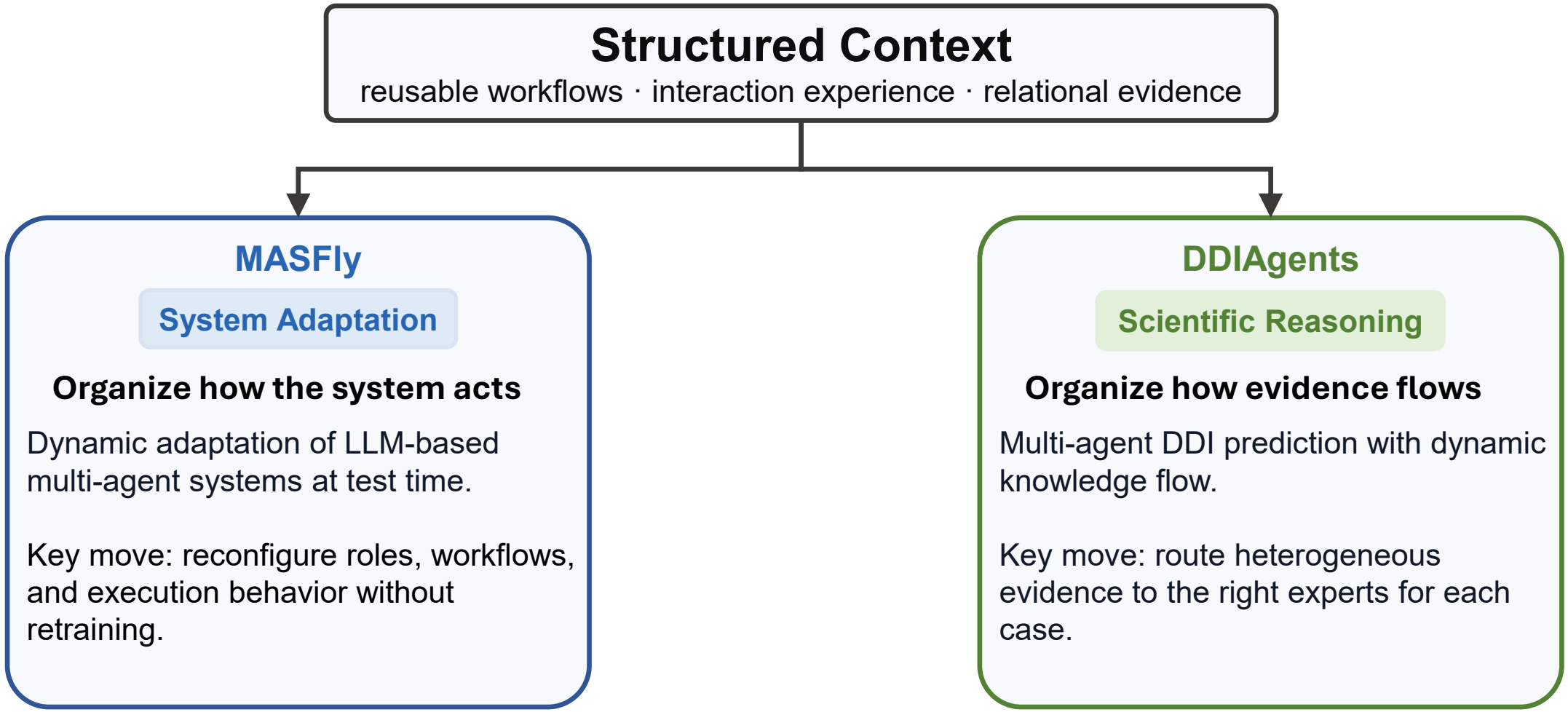
Useful structured memory matters more than simply expanding the context.

Structured memory increases the value obtained from each token of past experience.

Outline

1. How can a frozen LLM become an adaptive agent?
2. Why is more context not enough?
3. What does Structured Context change?
4. How does structure improve data-efficient agentic learning?
5. **How does structured context enable system-level adaptation?**
6. What should we take away—and what remains open?

Structured Context Enables System-Level Adaptation

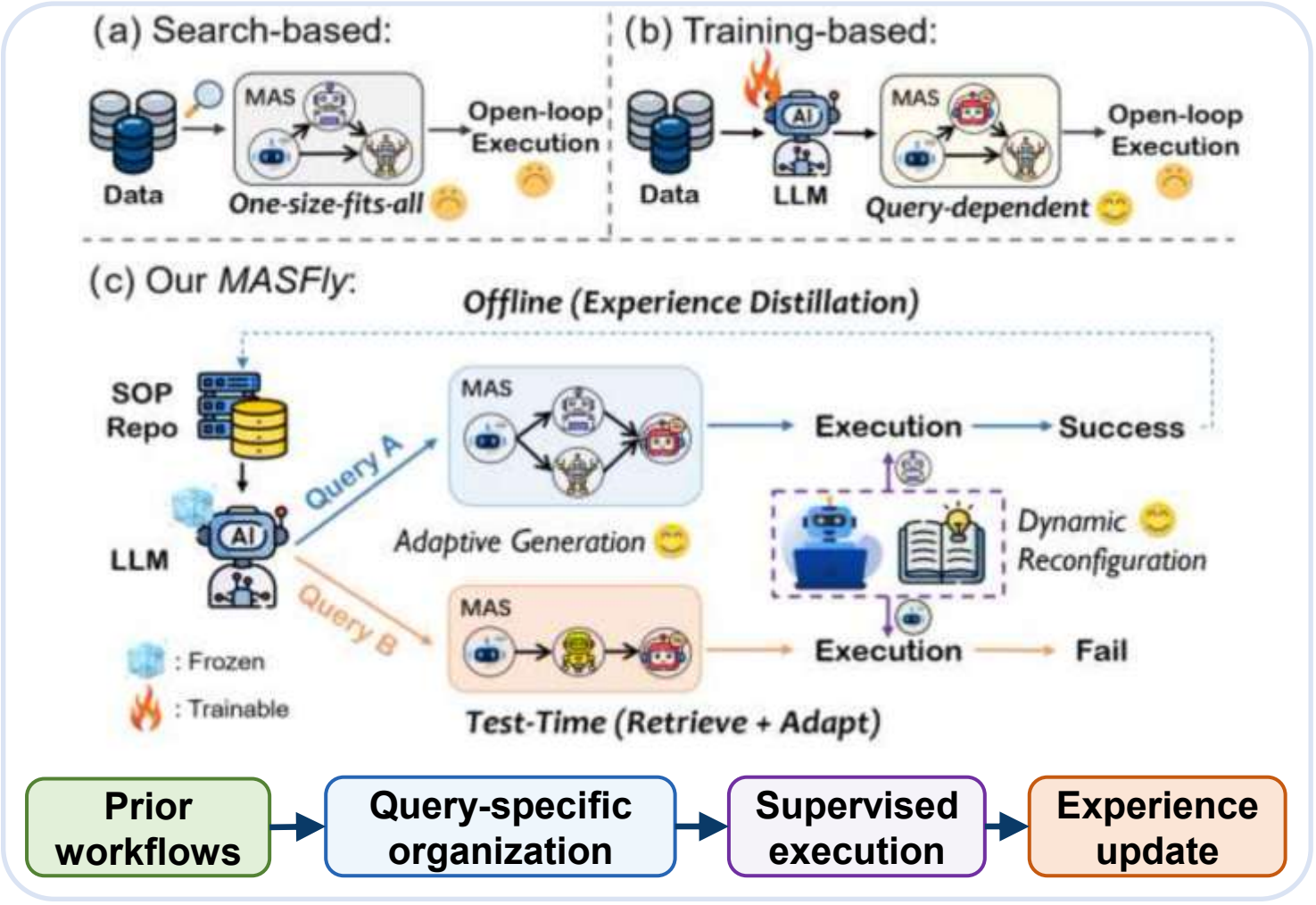


Structured context extends adaptation from the next action to how agents and evidence are organized.

Dynamically Adapt the System Structure

A simple shift in perspective

- MASFly extends the adaptation logic of in-context learning (ICL) from model behavior to **multi-agent organization.**
- Instead of updating weights, it adapts by conditioning on:
  - retrieved prior workflows
  - accumulated execution experience
- Achieving both **adaptive system generation** and **execution.**



MASFly extends test-time adaptation from model behavior to multi-agent organization.

Framework of MASFly

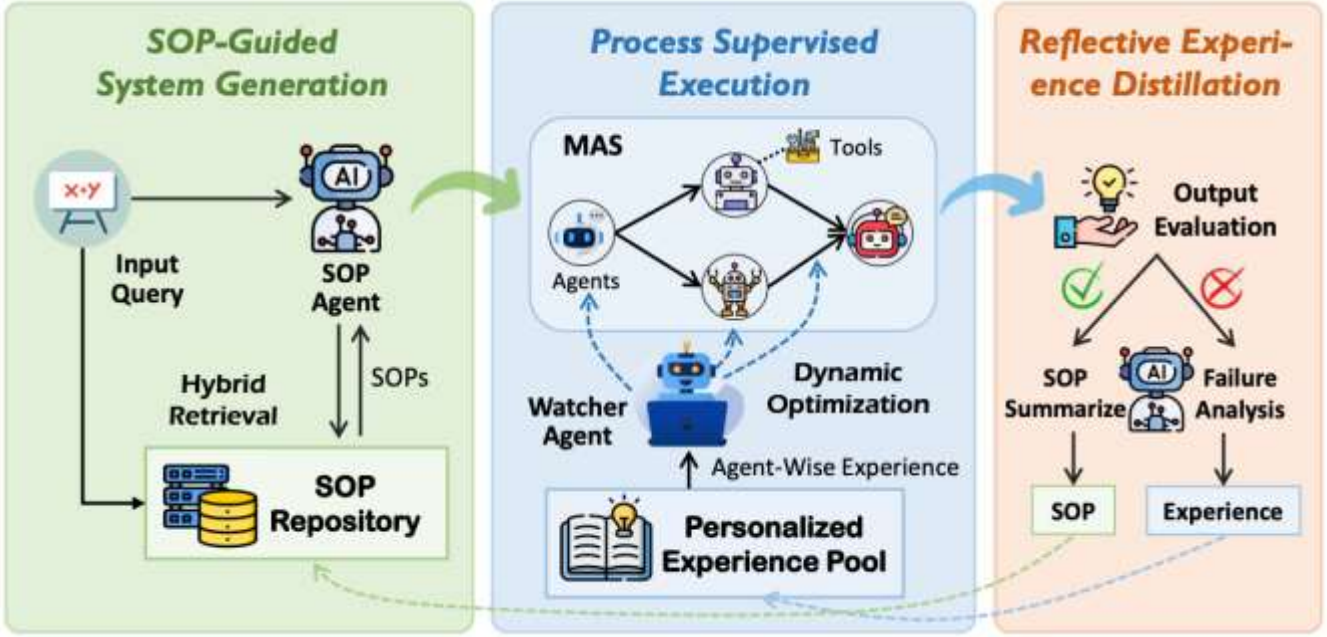
Three stages closed-loop:

1 **Query-driven system instantiation**
Retrieve SOP templates and instantiate a tailored initial system.

2 **Process-supervised execution**
A Watcher checks behavior against the workflow and intervenes when needed.

3 **Reflective experience distillation**
Successful runs become reusable SOPs; failures become agent-level lessons.

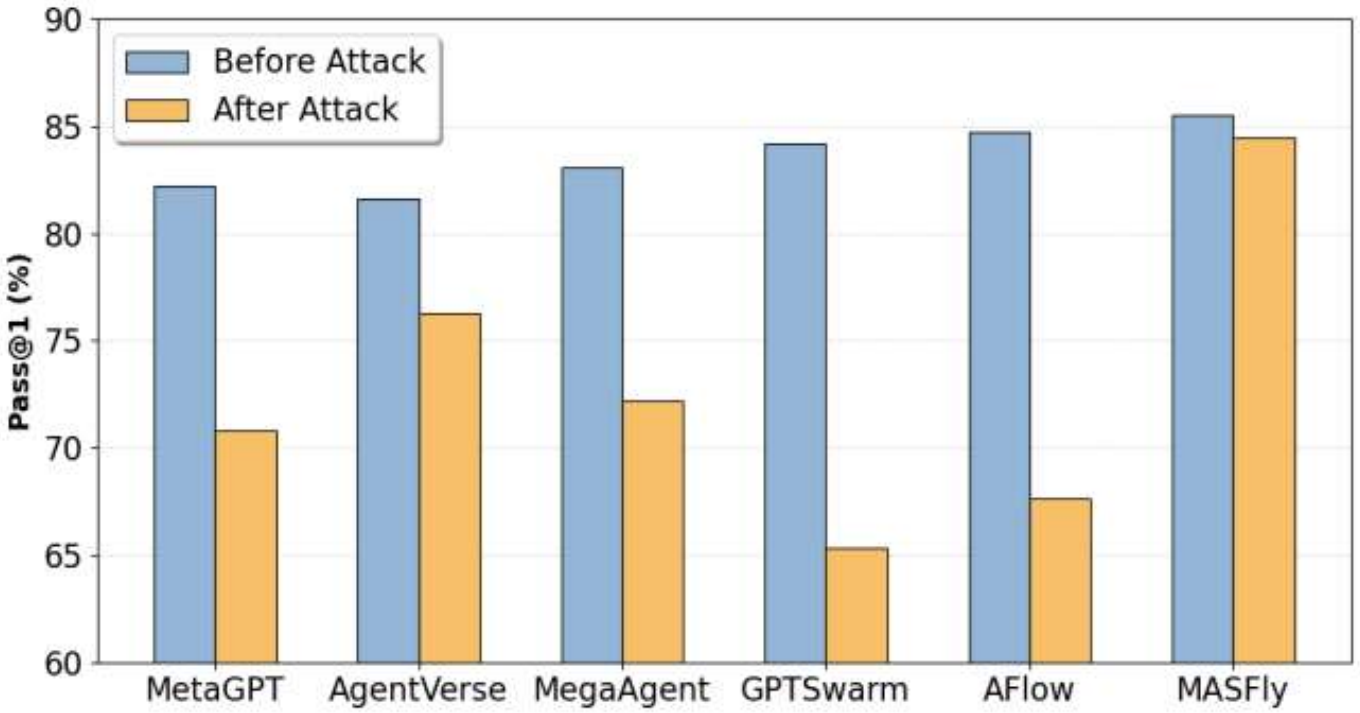
The design closes the loop between adaptation during a task and learning for future tasks.



Overview of our framework MASFly.

Closed-loop framework: system instantiation, supervised execution, and experience distillation.

Runtime Adaptation Improves Robustness



Static systems degrade under disruption

Most baselines suffer a clear performance drop after the prompt attack.

MASFly remains robust

Performance changes little when abnormal agent behavior occurs during execution

System-level adaptation enables recovery from failures that emerge during execution.

Shen et al. *DDIAgents: Predicting Drug Interactions by a Multi-Agent System with Dynamic Knowledge Flow*. KDD, 2026.

Dynamic Knowledge Flow

Knowledge sources

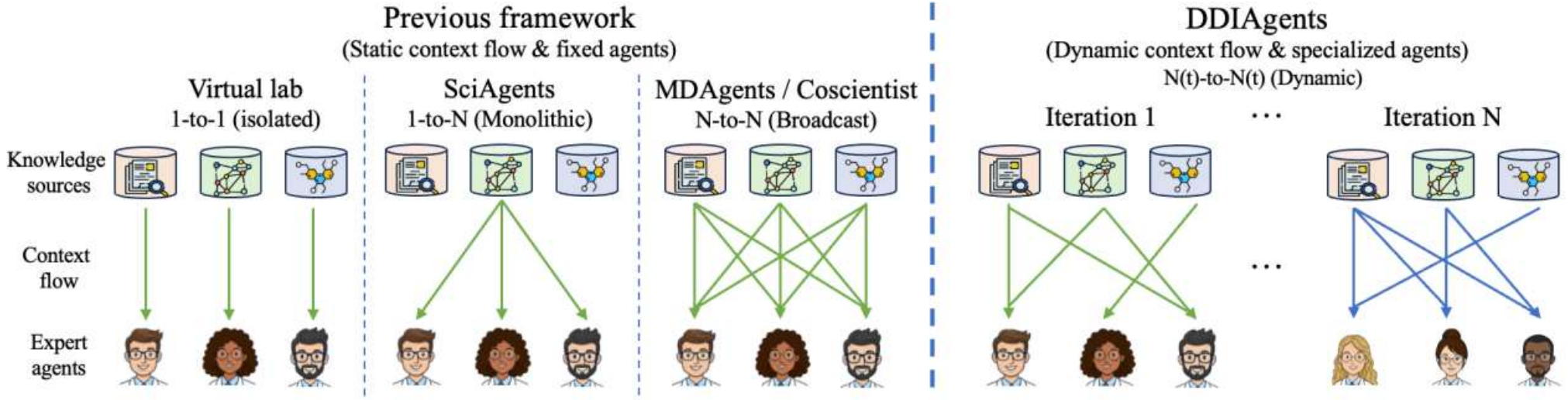
Molecular structural
SMILES and structural cues

Biomedical relational
Targets, prior DDI data

Semantic contextual
LLM summaries of mechanisms and DDI types

Routing rule

- Assign evidence to experts based on expertise and iteration state.
- Treat source-to-expert edges as variables, not fixed pipes.
- Reuse disagreement and refinement as supervision substitutes.



Dynamic source-to-expert routing turns heterogeneous evidence into relational supervision.

Framework of DDIAgents

1 Expert agent instantiation

A planner creates a tailored expert team for the current drug pair.

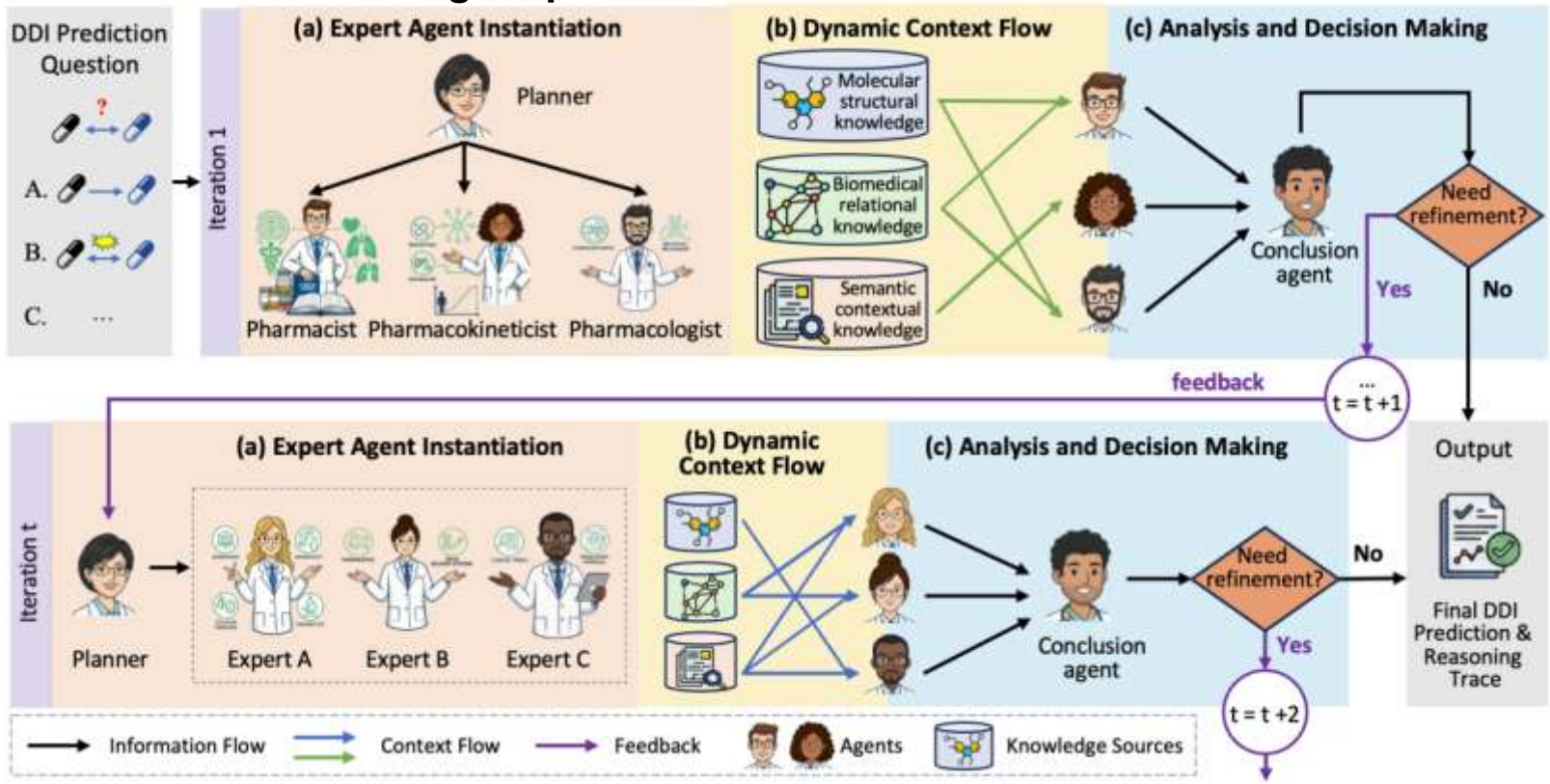
2 Dynamic knowledge flow

Mechanism-relevant evidence is routed to the right experts.

3 Analysis & decision making

Specialist analyses are synthesized; disagreement triggers refinement.

Two-iteration reasoning loop



Supervision substitute: expert disagreement → critique by the conclusion agent → updated plan.

The framework adapts per instance without collecting new labeled data.

Dynamic Knowledge Flow Improves Performance

Table 3: Comparison of different variants of DDIAgents

	DrugBank				TWOSIDES			
	S1		S2		S1		S2	
	Acc	F1	Acc	F1	Hit	NDCG	Hit	NDCG
DDIAgents	55.75	45.52	38.21	18.17	53.94	21.21	42.26	14.08
w/o DynExp	54.08	44.39	36.70	16.66	51.73	20.37	41.01	13.52
w/o DynKnow	52.23	40.19	35.57	16.01	50.34	19.52	38.71	12.07
w/o Iter	53.68	42.15	36.67	16.71	50.72	19.44	39.56	12.88

Dynamic knowledge flow matters

Removing dynamic knowledge routing consistently reduces performance.

Adaptation is multi-component

Removing expert dynamics or iterative refinement also weakens the full system.

Performance gains come from adapting how expertise and evidence are organized for each case.

One Structured-Context Pattern, Two Forms of Adaptation

	STRUCTURED CONTEXT	CONFIGURES	FEEDBACK & UPDATE	ADAPTATION TARGET
MASFly *System Adaptation*	SOP templates execution experience	roles · workflows execution process	Watcher feedback → SOPs / lessons	**Adaptive System Organization**
DDIAgents *Scientific Reasoning*	molecular · relational semantic evidence	expert team knowledge routing	Disagreement→ critique → updated plan	**Adaptive Reasoning Flow**

At the system level, Structured Context guides what is configured and what is revised after feedback.

Outline

1. How can a frozen LLM become an adaptive agent?
2. Why is more context not enough?
3. What does Structured Context change?
4. How does structure improve data-efficient agentic learning?
5. How does structured context enable system-level adaptation?
6. **What should we take away—and what remains open?**

From Limited Feedback to Adaptive Systems

1. Limited Feedback

- sparse / delayed supervision
- costly verification & interaction
- model weights remain fixed

2. Interaction Signals

- observations · actions · feedback
- outcomes · reflections · prior attempt

3. Flat Context Bottleneck

- mixed roles
- hard credit assignment
- poor reuse

4. Structured Context

WHAT IT IS
External · Typed · Relational · Editable

HOW IT WORKS
Select → Compose → Update

WHAT IT ENABLES
Turn interaction signals into reusable guidance

5. Adaptation at Multiple Levels

Decision
feedback redirects the next action

Experience
past interaction becomes reusable guidance

System
organization and reasoning flow can adapt

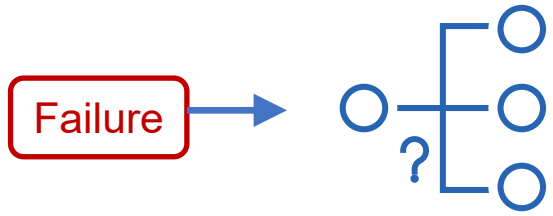
Structured Context turns limited interaction into reusable guidance for agent adaptation.

From Reusable Experience to Continual Agent Adaptation

What Should Be Updated?

Structural credit assignment

Localize sparse feedback to the structure that should change.

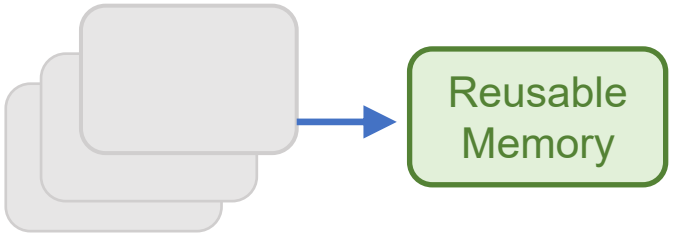


- Memory
- Skill
- Workflow
- Relation
- Verifier

What Should Be Retained?

Consolidation and forgetting

Merge reusable experience, resolve conflicts, and remove stale structure



- Merge
- Validate
- Forget
- Reuse

When Is Adaptation Worth the Cost?

Cost-aware adaptation

Adapt only when the expected gain justifies interaction and maintenance cost

Adaptation Gain **vs.** Interaction Cost

- Toll calls
- Verification
- Computation

These decisions turn reusable interaction experience into sustainable continual adaptation.

Thanks!