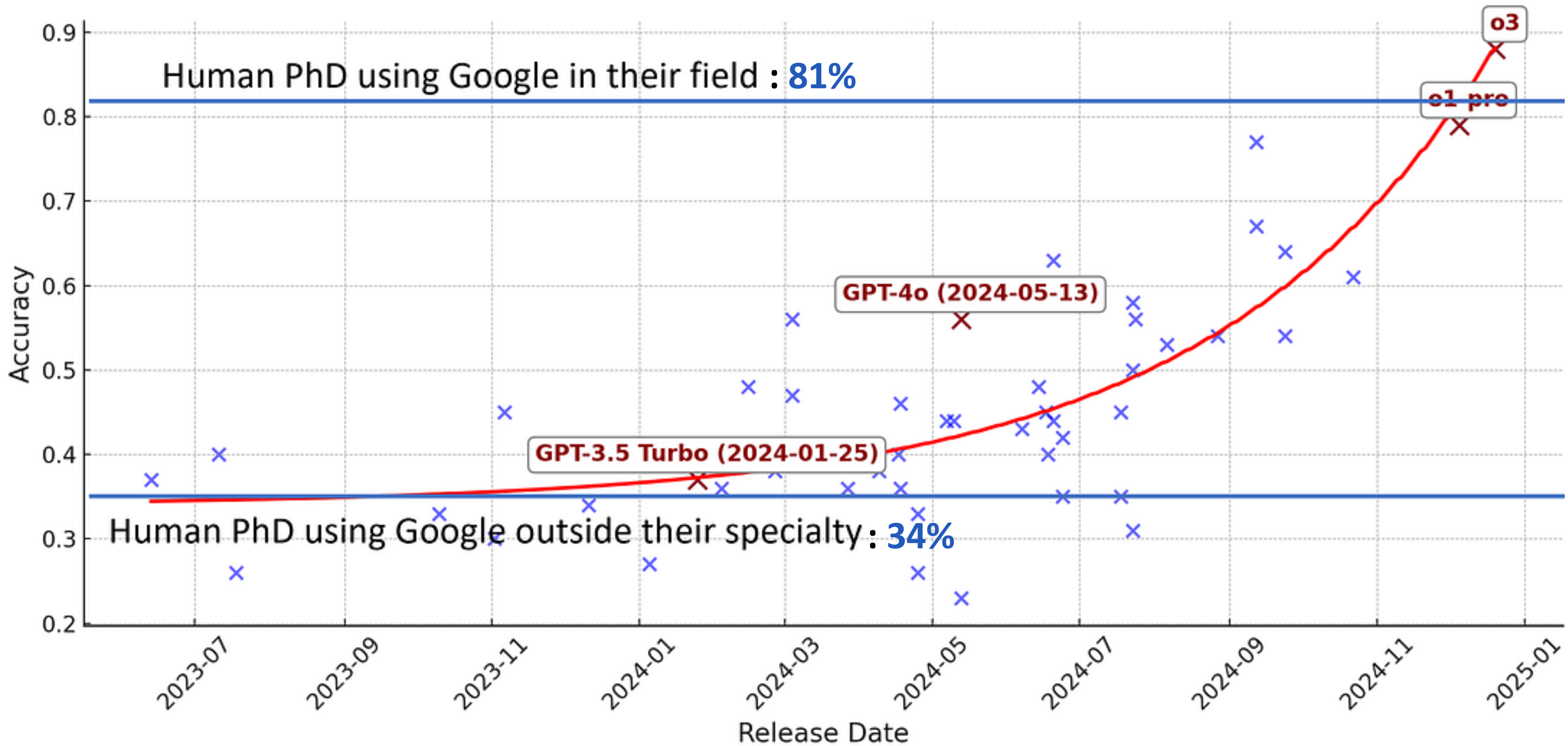
Accelerating Discovery with Intelligent Agents

Ian Foster

THE UNIVERSITY OF CHICAGO
globus labs
Argonne NATIONAL LABORATORY

Graduate-Level Google-Proof Q&A test (GPQA), Diamond problems

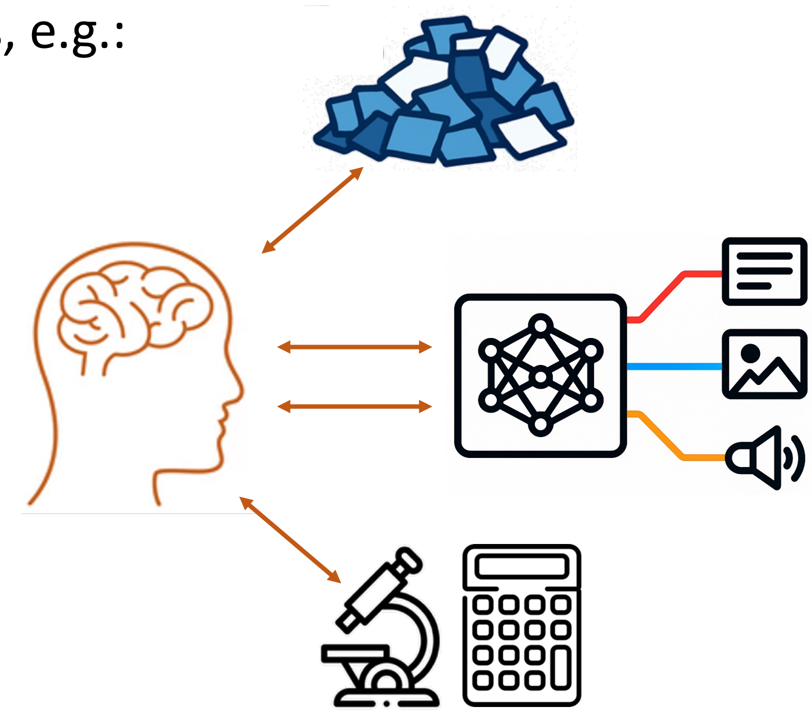


https://arxiv.org/pdf/2311.12022 https://epoch.ai/data/ai-benchmarking-dashboard

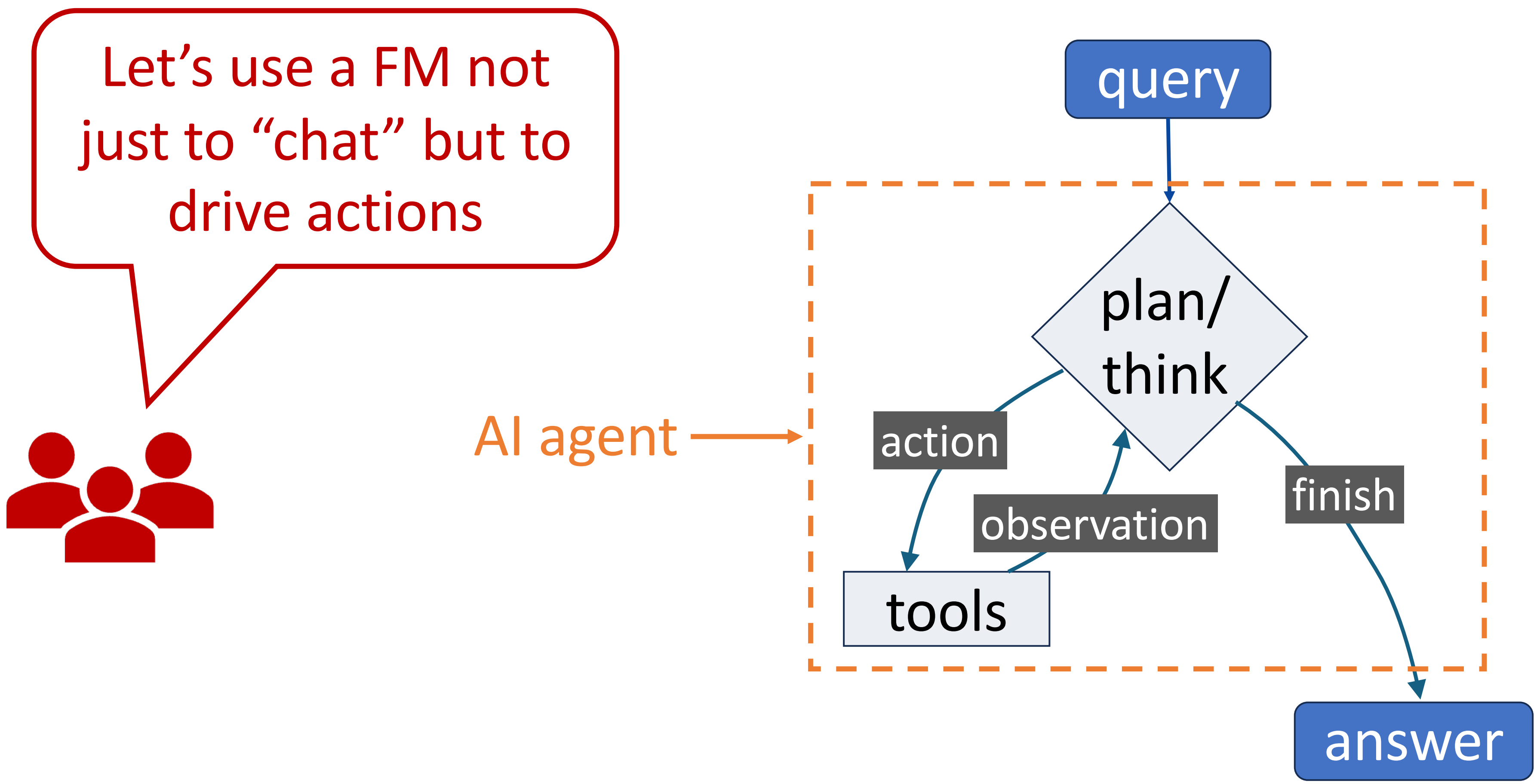
FMs are general-purpose technologies

Humans engage FMs for many purposes, e.g.:

- Analyze **knowledge**
- Define & evaluate **hypotheses**
- **Define protocols** to test
- **Select data** to use or request
- **Choose tools** (e.g., simulators, instruments, computers)
- **Define actions** (e.g., launch job, run query, trigger experiment)
- **Evaluate outputs**
- **Propose next steps**



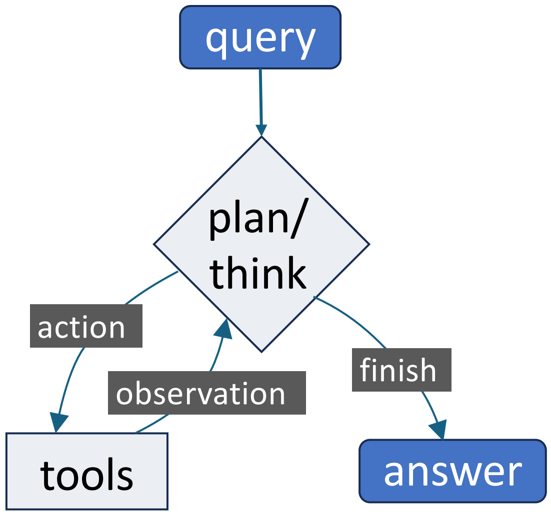
Human decision-making increasingly becomes the bottleneck


Let’s use a FM not just to “chat” but to drive actions
AI agent
query
plan/ think
action
observation
tools
finish
answer

An agent is a persistent, stateful process that acts on behalf of a user or system

An agent may:

- **Observe** inputs or events [or user query]
- **Plan** (decide on) actions using a policy (rules or FM)
- **Act**: Execute tools or call other agents
- **Learn**: Update state to adapt over time



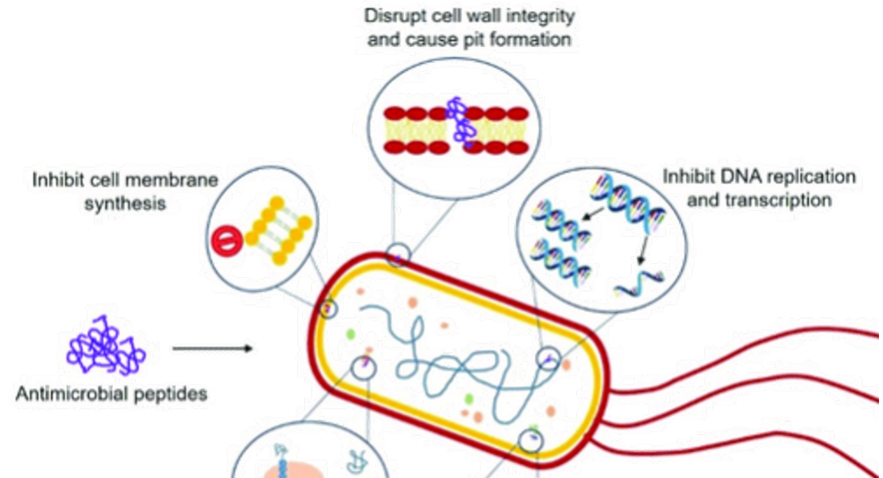
We can think of an agent as a [scientific] assistant that can reason, act, and coordinate on our behalf

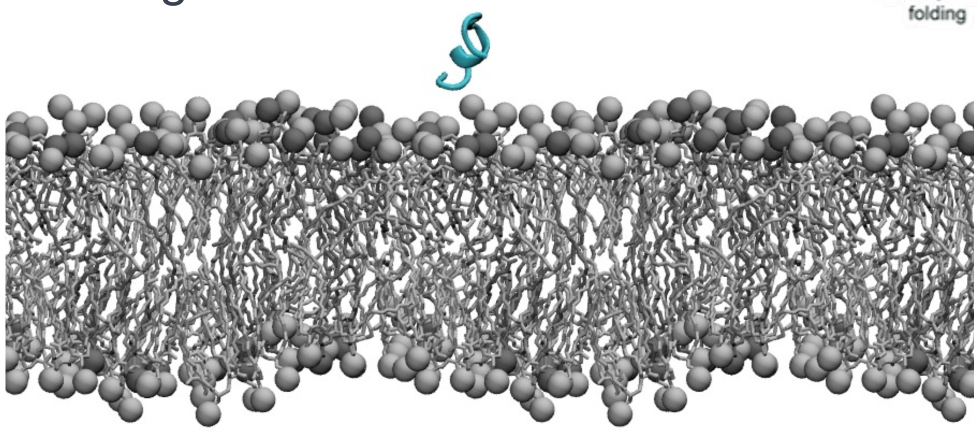
An agentic architecture for the design of antimicrobial peptides

An antimicrobial peptide (AMP) is a short (typically 12 to 50 amino acid) molecule that can target and kill viruses, bacteria, fungi, and other pathogens

Challenge: Design an AMP that can kill specified bacterial strains without harming host cells

With 20 possible amino acids, there are $20^{20} = 10^{26}$ AMPs of length 20

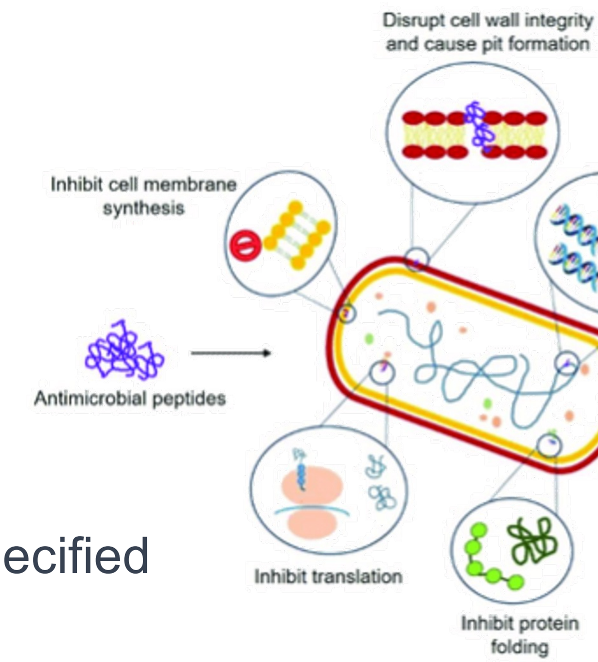




Arvind Ramanathan

An agentic architecture for the design of antimicrobial peptides

An antimicrobial peptide (AMP) is a short (typically 12 to 50 amino acid) molecule that can target and kill viruses, bacteria, fungi, and other pathogens



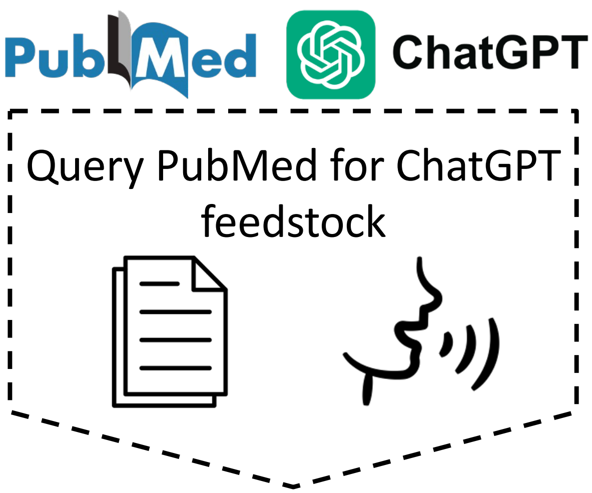
Challenge: Design an AMP that can kill specified bacterial strains without harming host cells

Arvind Ramanathan

A rational design approach might combine knowledge of bacterial cell membrane composition and structure, AMP molecular and structural properties, host cell membrane characteristics and intracellular pathways—knowledge that may be gained by **database/literature search**, **simulation**, **experiment**

Example: A **peptide expert**

(Prototyped with PubMed and ChatGPT)



Retrieve abstracts **A** from PubMed that reference specified **peptide**

Use ChatGPT to build hypotheses via retrieval-augmented generation: e.g.:

> "Given **A**, on which organism is {**peptide**} acting?"

Evaluate hypotheses and update query and hypothesis generation policies

Arvind Ramanathan, Priyanka Setty, et al.

Outline of a peptide agent implementation

```
# Persistent agent state
Hypotheses = {}        # hypothesis_id -> hypothesis object
EvidenceLedger = {}    # hypothesis_id -> list of evidence events
RetrievalPolicy = {}   # learned biases over queries/sources
AgentConfig = {
    "confidence_threshold": 0.8,
    "retirement_threshold": -0.5
}

while True:

    # 1. Select target peptide
    peptide = select_peptide(Hypotheses)
```

Tool call

```
    # 2. Retrieve new evidence
    query = build_pubmed_query(peptide, RetrievalPolicy)
    abstracts = retrieve_abstracts(query)

    # 3. Generate new hypotheses (if needed)
    if not has_active_hypothesis(peptide, Hypotheses):
        new_hypotheses = generate_hypotheses(
            abstracts,
            prompt = (
                "Given abstracts A, on which organism is "
                f"{peptide} acting?"
            )
        )
```

“Think”

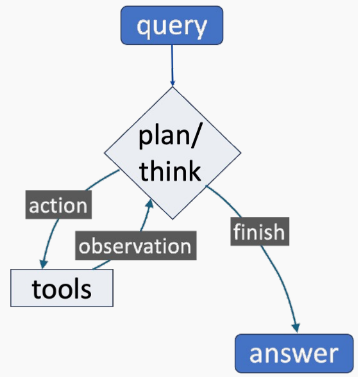
```
        for H in new_hypotheses:
            Hypotheses[H.id] = H
            EvidenceLedger[H.id] = []

    # 4. Evaluate hypotheses against new evidence
    for H in active_hypotheses(peptide, Hypotheses):

        assessment = evaluate_hypothesis(H, abstracts)
        # assessment ∈ {supports, contradicts, inconclusive}
        # with confidence score

        EvidenceLedger[H.id].append({
            "abstracts": abstracts,
            "assessment": assessment.type,
            "strength": assessment.strength,
            "timestamp": now()
        })

        # 5. Update hypothesis confidence
        H.confidence = update_confidence(
            H.confidence,
            assessment
        )
```




```
    # 6. Prune or retire hypotheses
    for H in Hypotheses.values():
        if H.confidence < AgentConfig["retirement_threshold"]:
            retire_hypothesis(H)
```

Learn

```
    # 7. Adapt retrieval policy
    RetrievalPolicy = update_retrieval_policy(
        RetrievalPolicy,
        EvidenceLedger
    )

    # 8. Self-reflection and gap analysis (periodic)
    if time_for_reflection():
        gaps = identify_knowledge_gaps(Hypotheses)
        RetrievalPolicy = bias_toward_gaps(
            RetrievalPolicy,
            gaps
        )

    # 9. Optional human-in-the-loop
    if uncertainty_high(Hypotheses):
        request_human_feedback(Hypotheses)
```

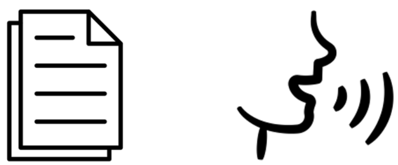
Finish

```
    # 10. Termination or sleep
    if stopping_condition_met(Hypotheses):
        break
    else:
        sleep()
```

Define other agents, which may also be FM-powered

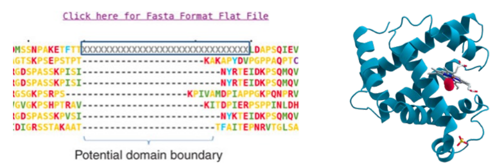
PubMed ChatGPT

Query PubMed for ChatGPT feedstock



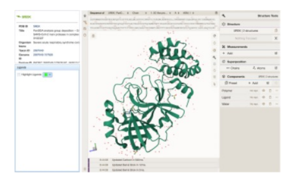
Protein BLAST protein ▸ protein AlphaFold

Align proteins, predict structure, rank results



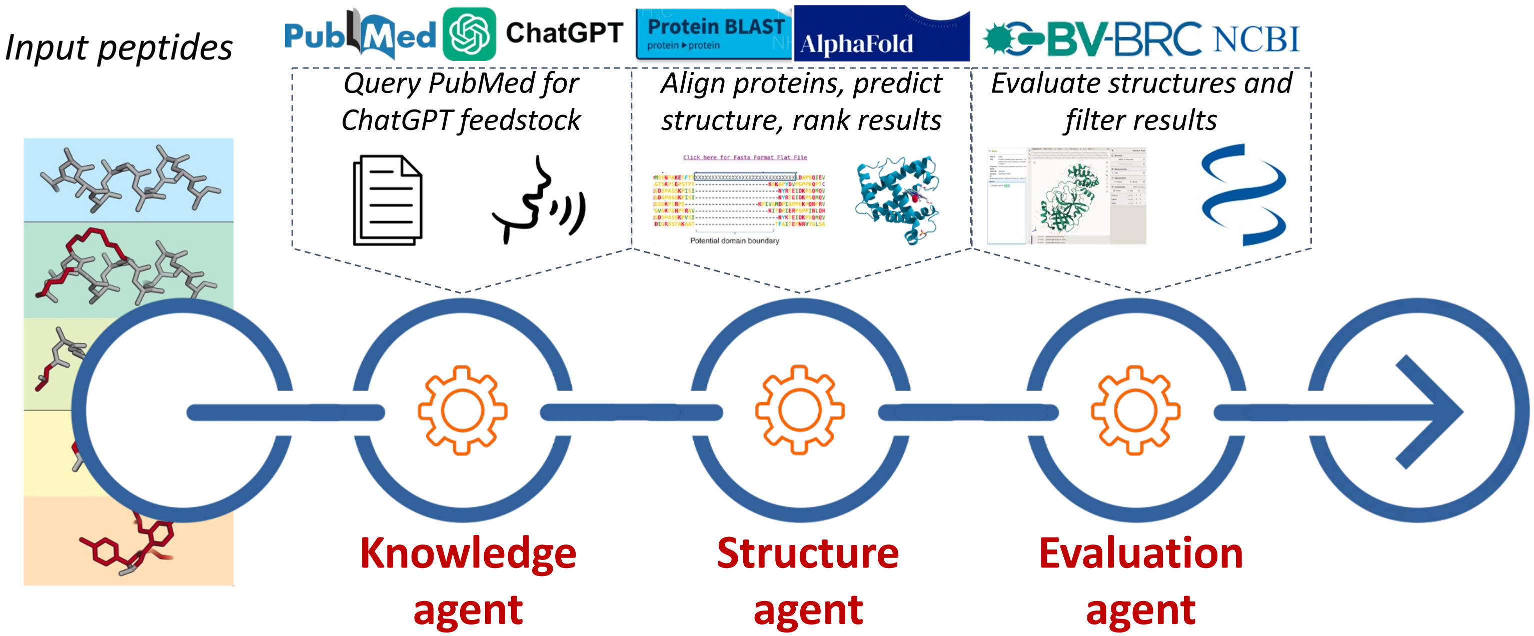
BV-BRC NCBI

Evaluate structures and filter results



Arvind Ramanathan, Priyanka Setty, et al.

Link agents to construct an application

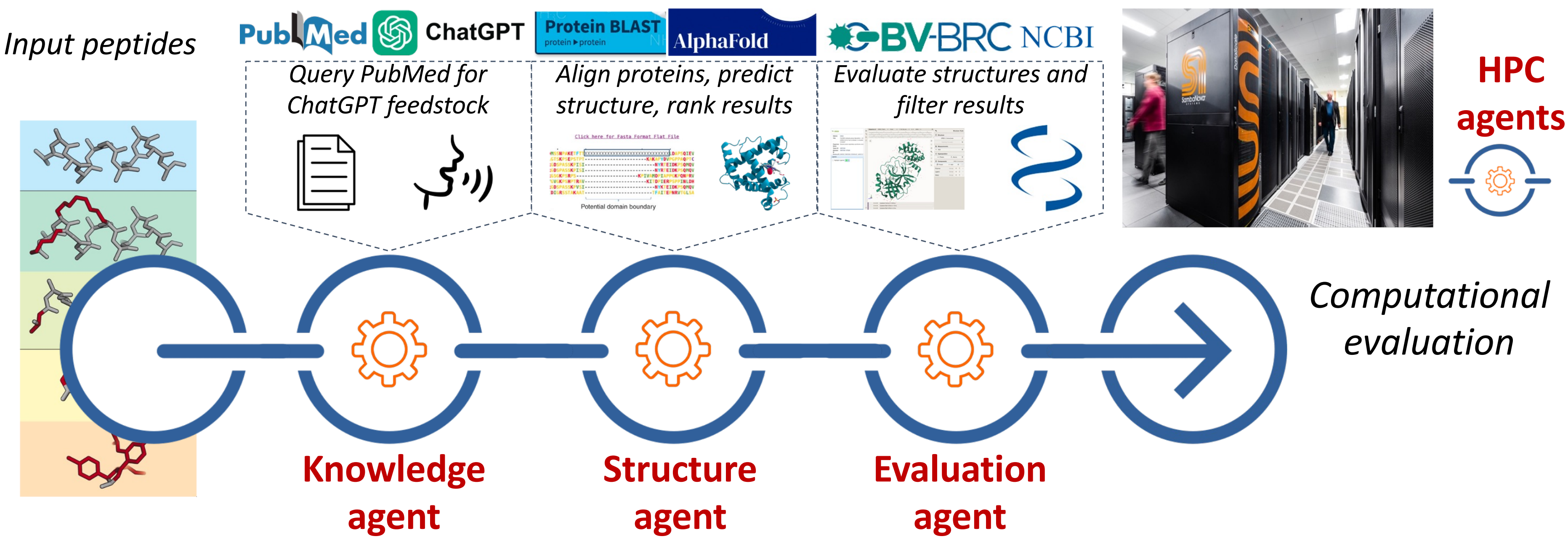


We use **Academy** to create and manage individual agents, which query databases, retrieve data, run simulations, run experiments, etc.

Arvind Ramanathan, Priyanka Setty, et al.

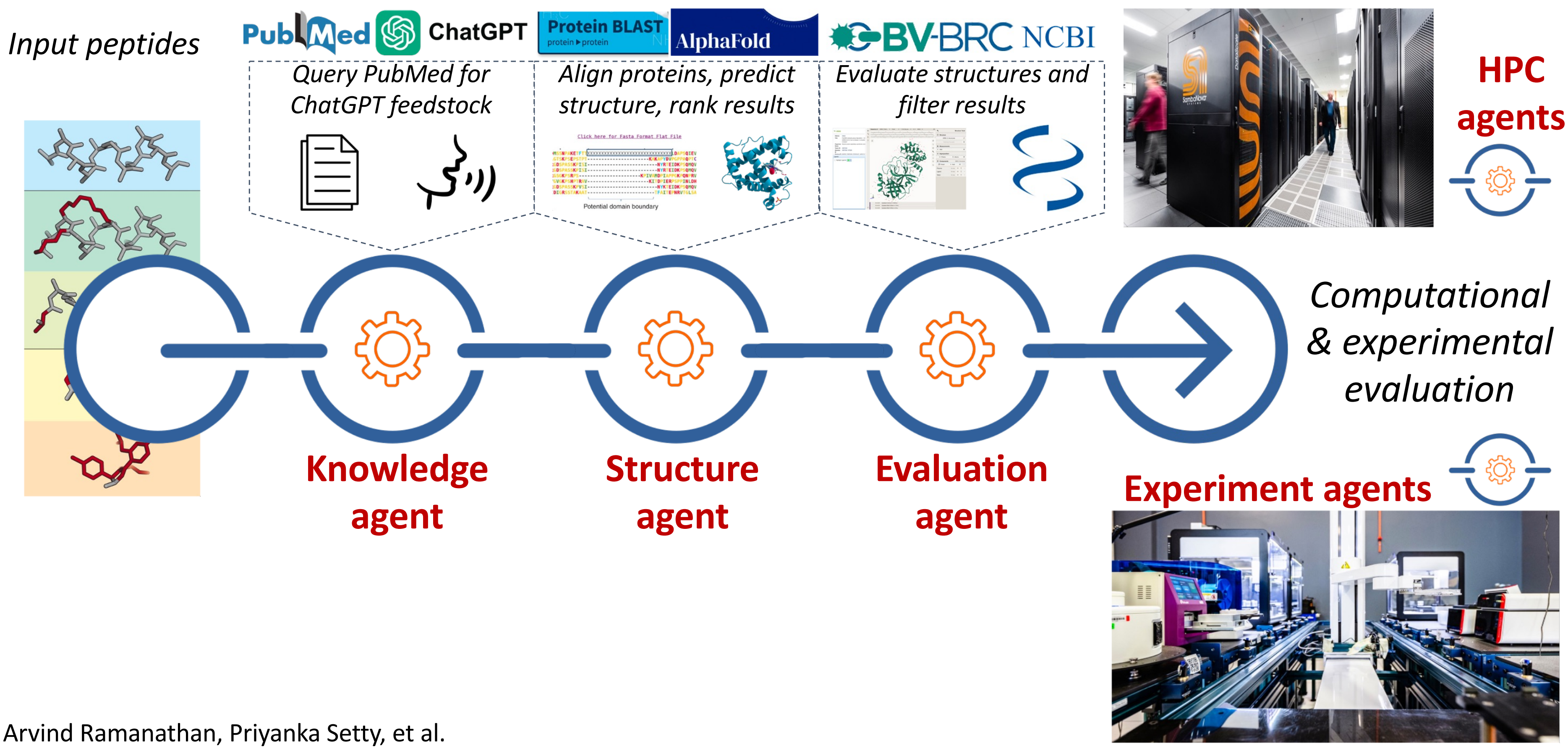
https://academy-agents.org

Link with HPC for **computational evaluation**



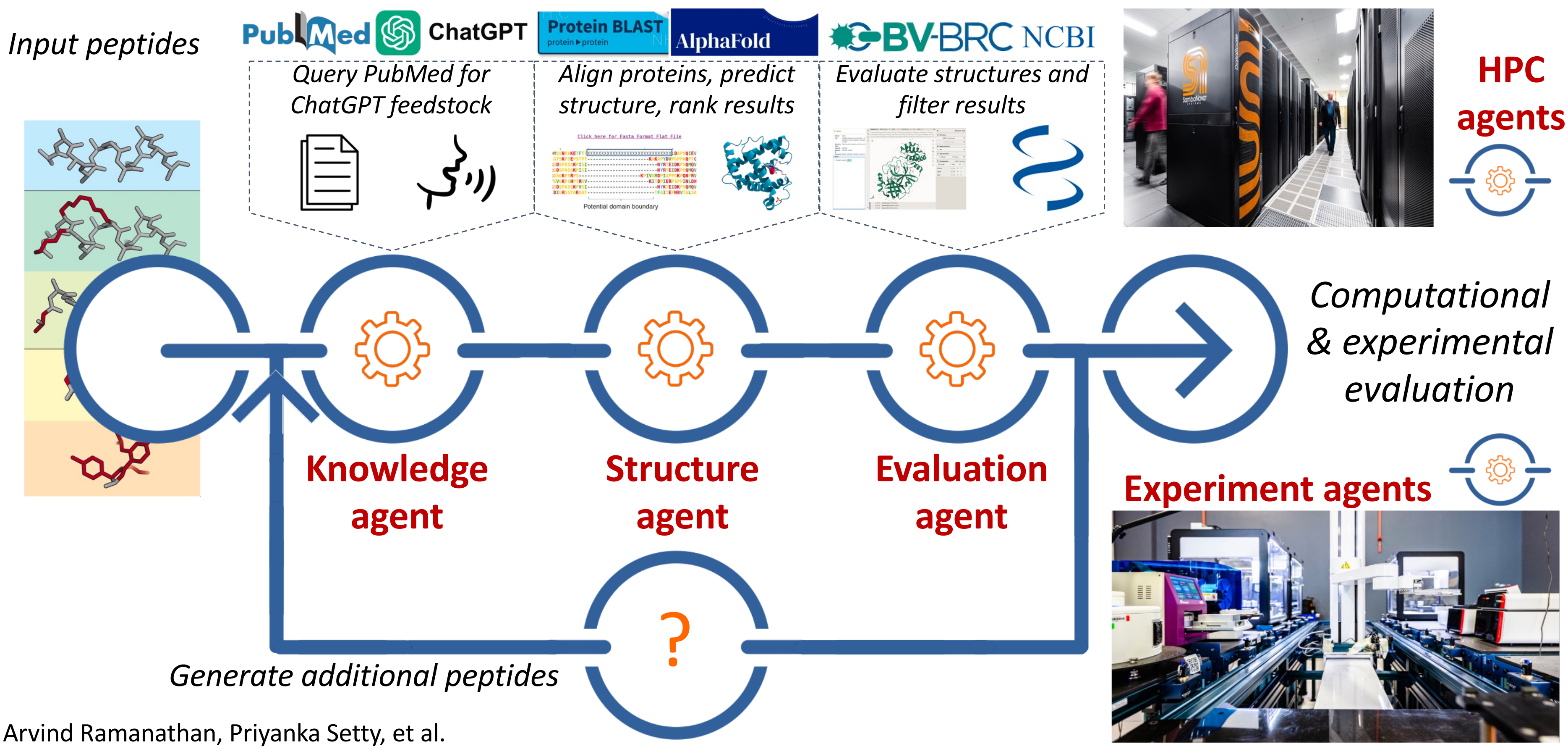
Arvind Ramanathan, Priyanka Setty, et al.

Link with self-driving labs for **experimental evaluation**



Arvind Ramanathan, Priyanka Setty, et al.

Close the loop for autonomous discovery



Arvind Ramanathan, Priyanka Setty, et al.

Agentic middleware: Scope and challenges

Fault Tolerance

Scaling

Data Movement

Protocols

Multi-agent Conversations

Deployment

Cross-domain execution

Tool Calling

LLM APIs

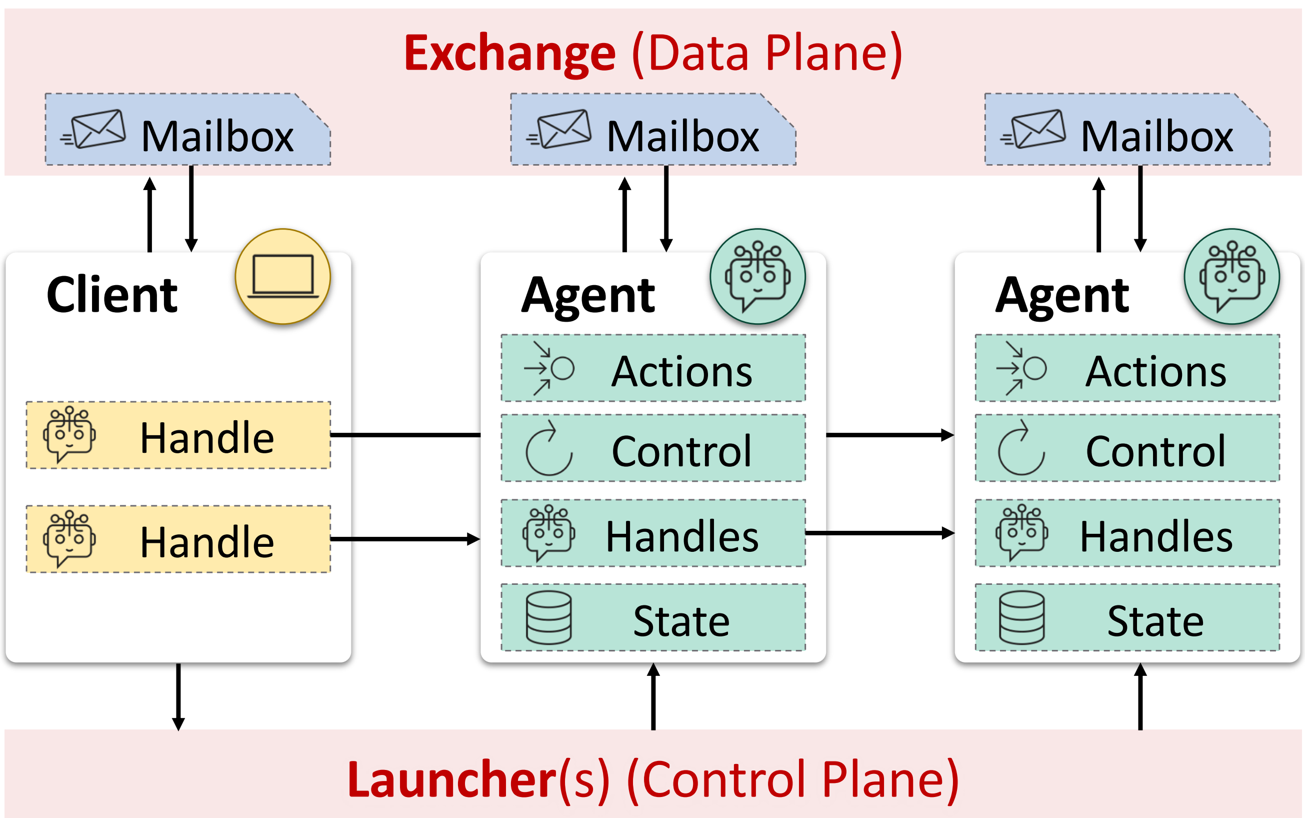
Low-level challenges

High-level challenges

Academy

LangChain, AutoGen, Pydantic AI, etc.

Exploring agentic middleware: **Academy**



Dr. Greg Pauloski

Dr. Kyle Chard

Alok Kamatar

https://academy-agents.org

Agents defined by a **behavior**

Clients & other agents can request **actions**

```
import asyncio
from academy.behavior import Behavior, action, loop

class Example(Behavior):
    def __init__(self) -> None:
        self.count = 0  # State stored as attributes

    @action
    async def square(self, value: float) -> float:
        return value**2

    @loop
    async def count(self, shutdown: asyncio.Event):
        while not shutdown.is_set():
            self.count += 1
            asyncio.sleep(1)
```

Instance of a behavior is **state**

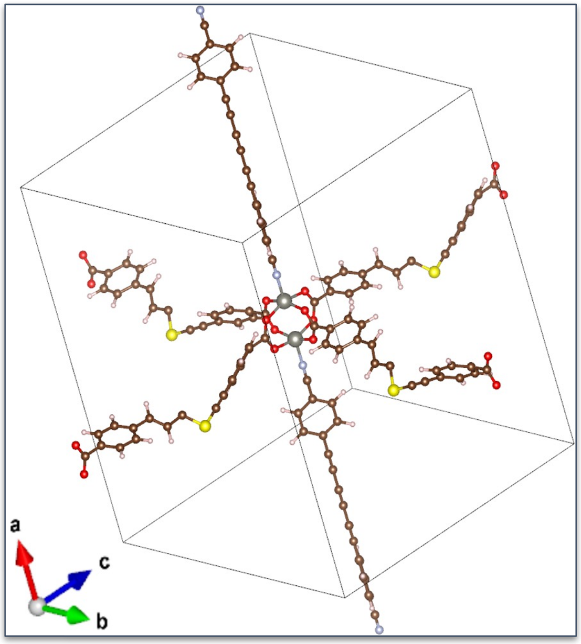
Control loops for autonomous behavior

https://docs.academy-agents.org/latest/get-started/

Use case: Metal organic framework (MOF) discovery

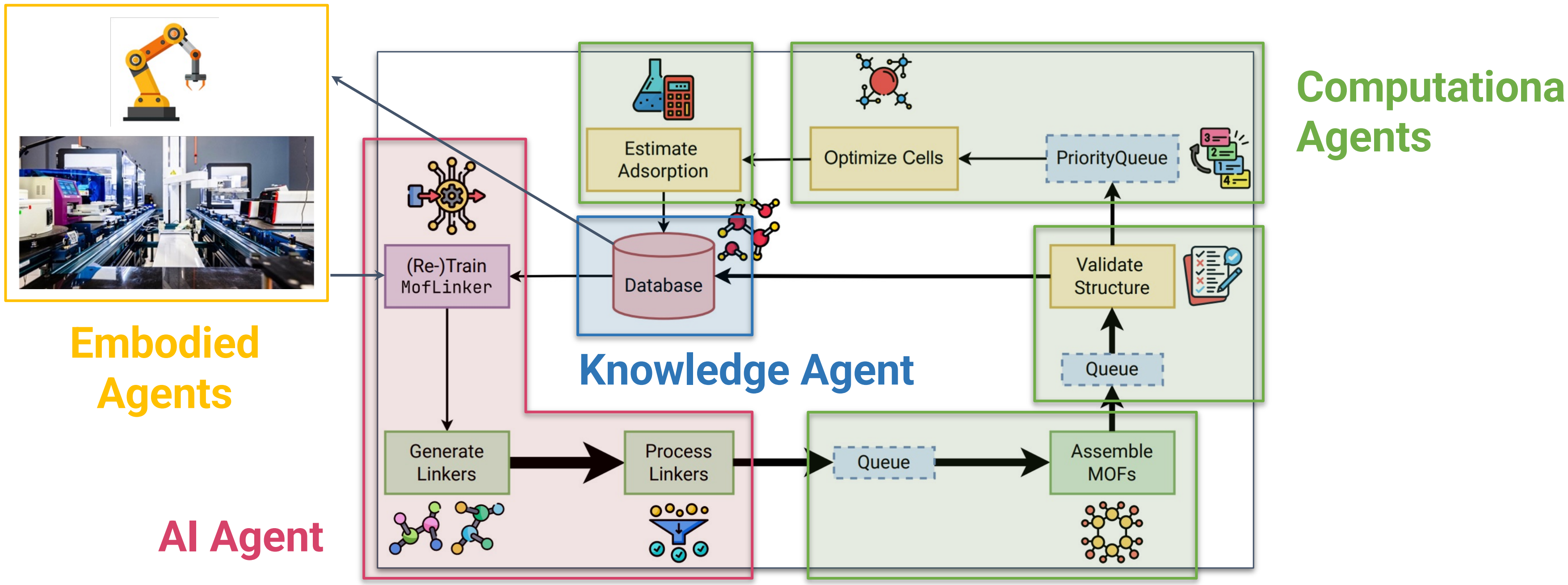
- A MOF is composed of organic molecules (ligands) and inorganic metals (nodes)
- Porous structures that can adsorb and store gases -- the sponges of materials science
- Topologies can be optimized for targeted gas storage: e.g., **Carbon Capture**

Goal: Efficient discovery of MOFs with desirable properties for target applications



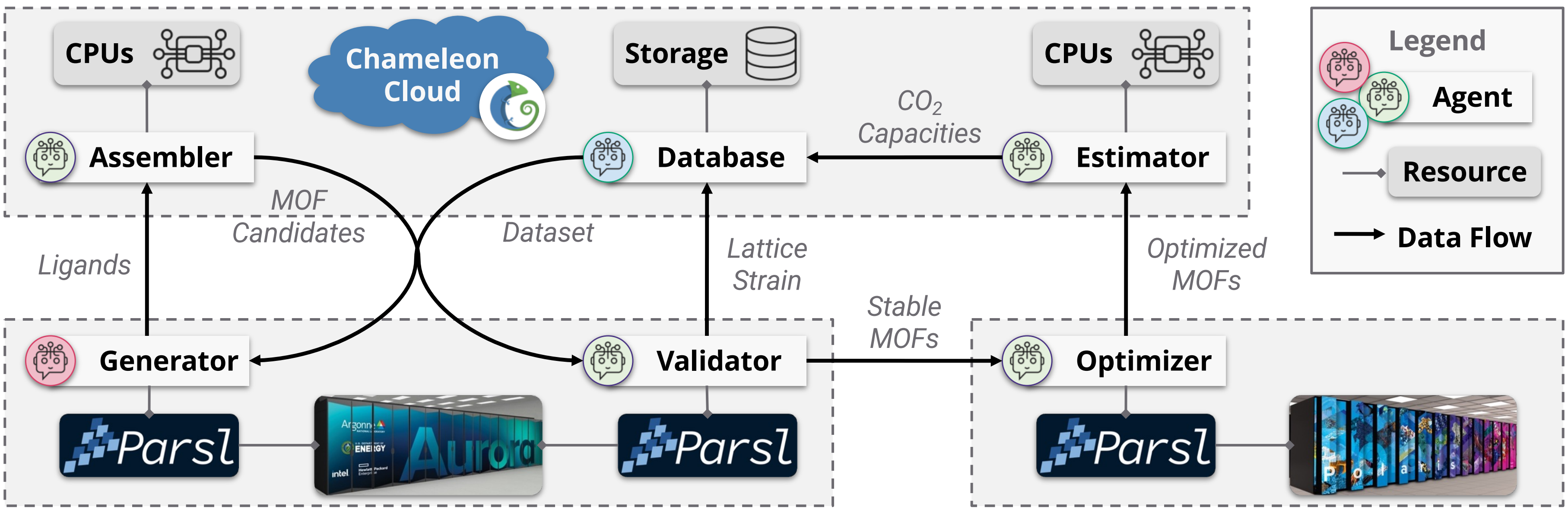
Intractable search space of ligand, node, & geometry combinations

MOFA code for metal-organic framework discovery, agentified with Academy



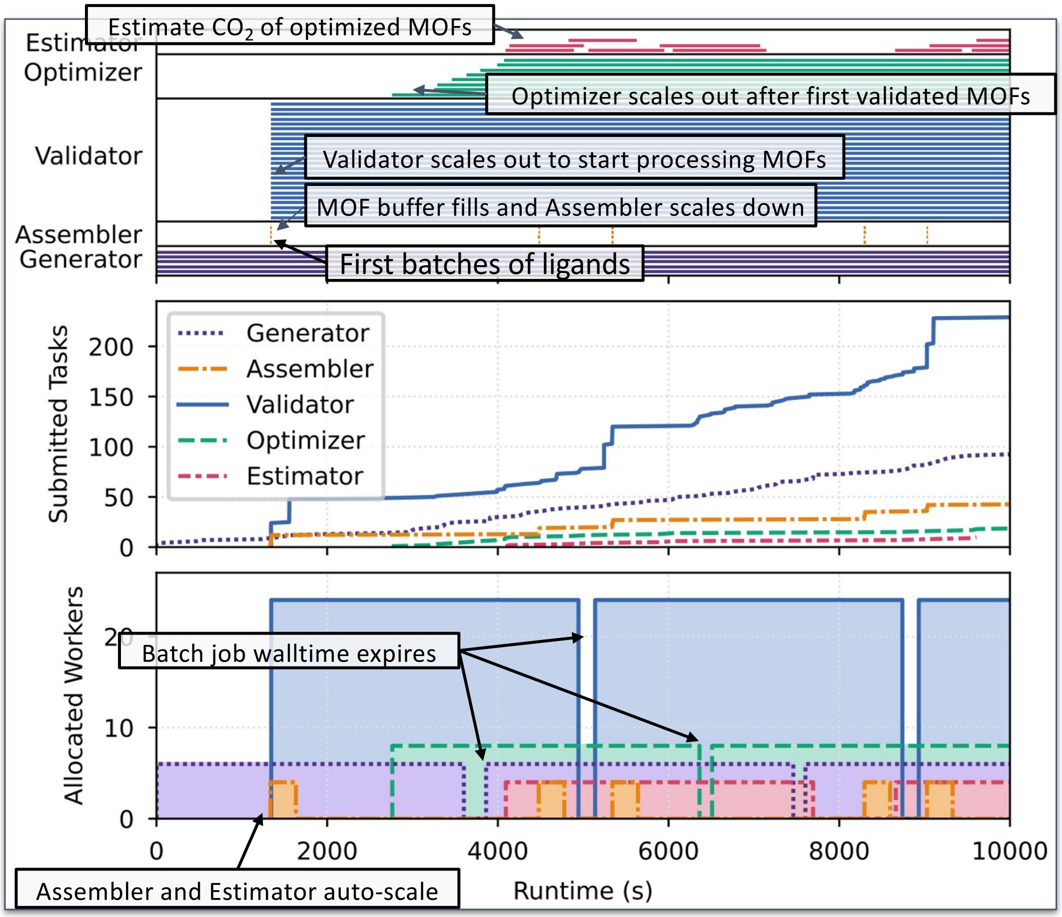
Yan et al., "MOFA: Discovering materials for carbon capture with a GenAI- and simulation-based workflow" (Under review; https://arxiv.org/abs/2501.10651)

Agentified MOFA code easily maps to many resources



*Agents executed remotely via **Globus Compute***
*Data moved via **Globus transfer***
*Authentication and authorization via **Globus Auth***

Agentified MOFA application execution trace



Benefits of agentic model:

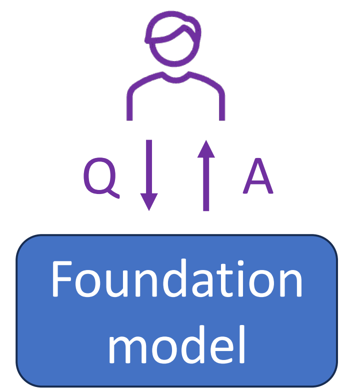
- **Placement:** Move agents to resources
- **Separation of concerns:** Resource acquisition & scaling based on local workload
- **Loose coupling:** Swap agents, integrate new agents (e.g., SDL)
- **Shared agents:** Multiple workflows can share agents (microservice-like)

Agency as a new organizing abstraction for computer science

AI progress is commonly framed around models

- Scale, parameters, benchmarks
- Models as stateless inference engines
- Execution assumed to be request–response

This framing is increasingly incomplete

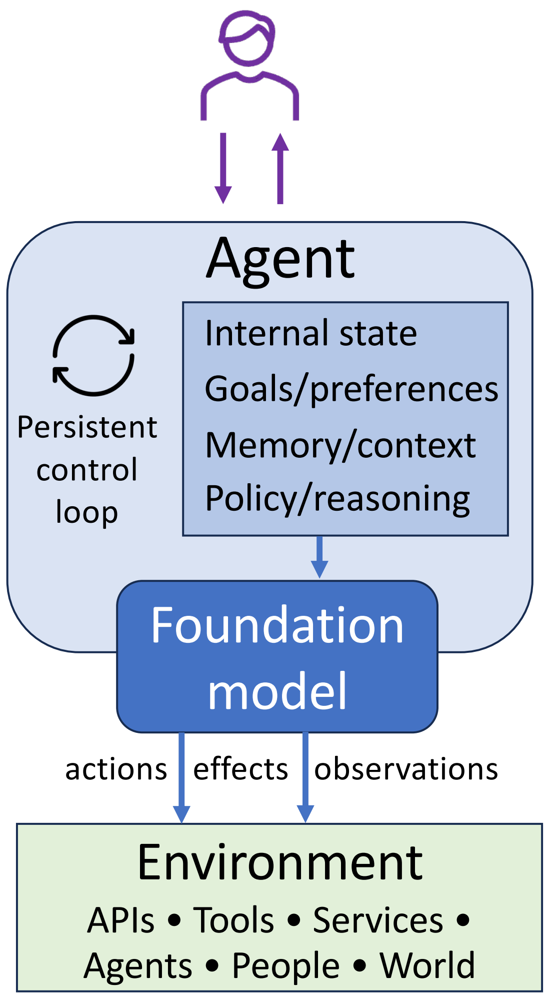


From models to agents

Deployed systems increasingly:

- Persist over time
- Initiate actions autonomously
- Interact continuously with tools, APIs, people
- Accumulate state and context

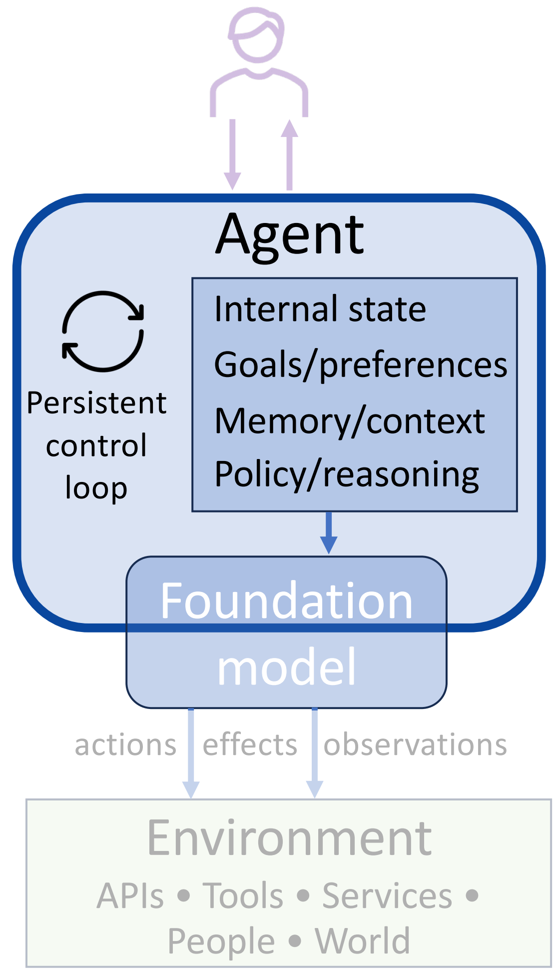
*These systems behave as **agents***



What makes a system "agentic"?

Agentic systems combine four properties

- **Persistence:** Long-lived state and context
- **Autonomy:** Decide when & how to act
- **Goal-directedness:** Pursue objectives over time
- **Open environments:** Act in settings that cannot be fully specified

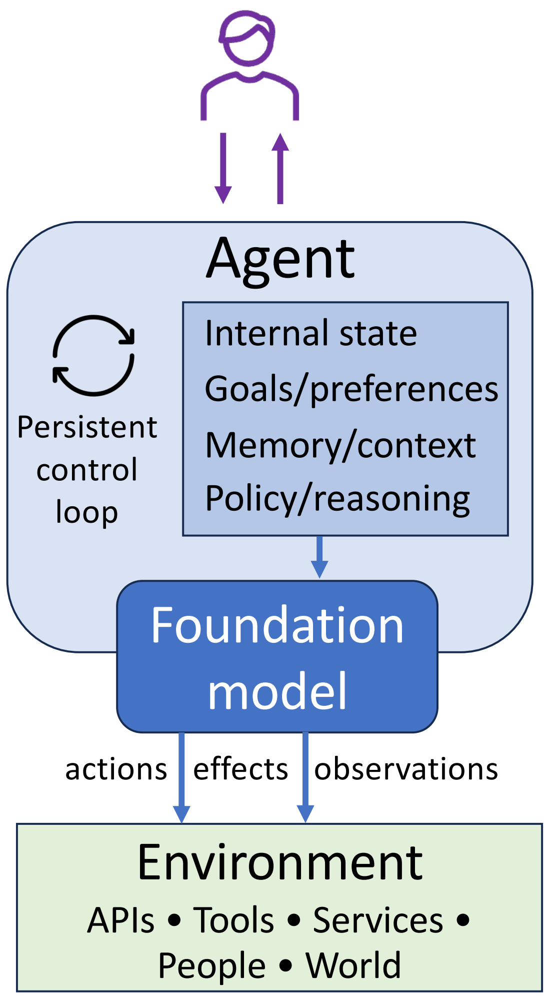


None of these properties is new

Each of the following has a long history:

- Persistent processes — *e.g., OS daemon*
- Autonomous controllers — *e.g., thermostat*
- Goal-directed planners — *e.g., classical planner*
- Open-world interaction — *e.g., network service*

What is new is that all four exist simultaneously in a single computational entity, enabled by foundation models

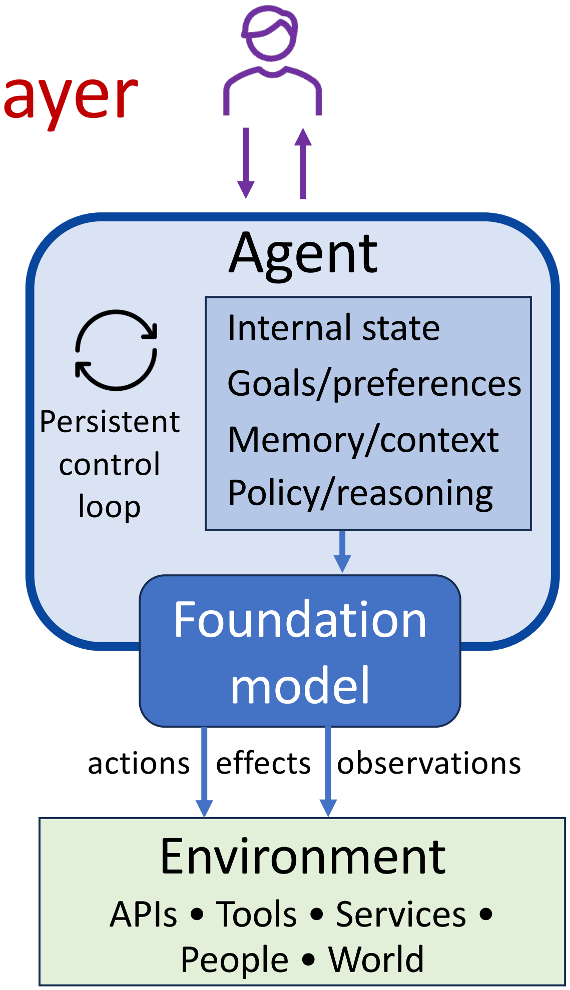


“Agents” are not just an application layer

Agentic systems reorganize computation

- Control flow moves to inside the system
- Responsibility shifts from caller to agent
- Time horizon expands beyond individual executions

This reorganization stresses foundational assumptions in computer science

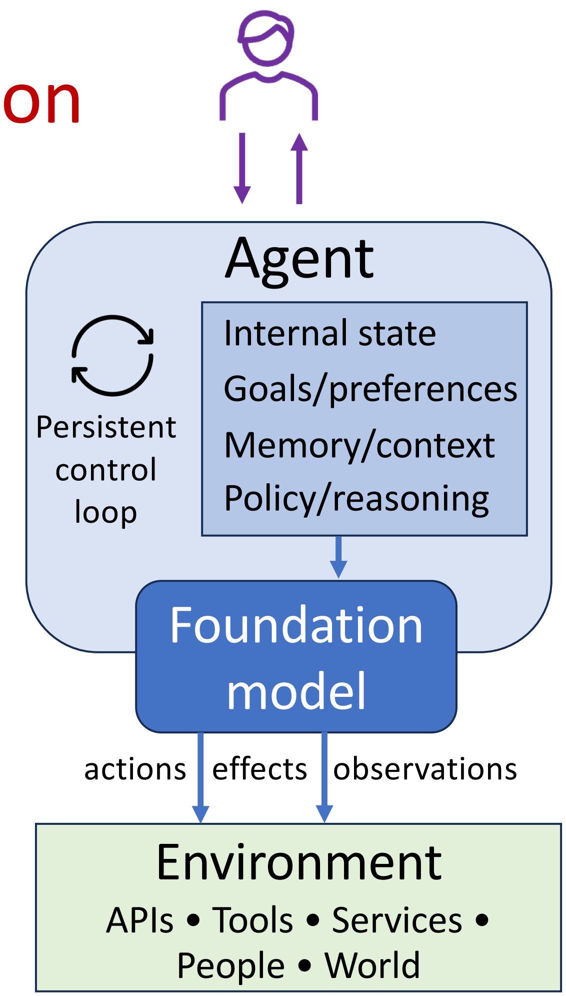


Agency as a new organizing abstraction

Agency = Persistent, autonomous control loop embedded in an open environment

Core claim: Agency provides a unifying abstraction for reasoning about autonomous computational systems

- Explains recurring failures as abstraction mismatches
- Connects systems, PL, theory, and HCI (and AI)
- Moves discussion beyond ad hoc engineering fixes



Example failure: Autonomous resource consumption

- Scenario: An agent is deployed to monitor a cloud service and "improve reliability"
  - Persistently observes metrics, logs, and alerts
  - Authorized to provision resources & run diagnostics
- Upon performance anomaly, it autonomously:
  - Spins up additional instances
  - Runs large diagnostic queries
  - Invokes paid external APIs
- Each action is locally reasonable, but costs end up excessive

Example failure: Autonomous resource consumption

Traditional per-invocation resource-limiting methods fail for self-initiated computation

- Agents decide when and how to act
- Can spawn processes and invoke paid APIs
- Traditional limits are reactive, not preventive

Assumption of externally invoked, episodic computation is invalid

Agentic systems require proactive constraint enforcement

Where current abstractions fall short

Recurring failure modes in deployed systems:

- Autonomous resource consumption
- Ambiguous agent lifecycles
- Distributed failure attribution
- Misplaced responsibility at system boundaries
- Opaque behavior under continuous operation

These failures are systematic, reflecting mismatches between existing abstractions and realities of agentic systems

Recurring failure modes observed in agentic systems

Observed Failure Mode
Autonomous resource consumption and unbounded action initiation
Ambiguous agent lifecycles and costly termination
Distributed failure attribution across long decision sequences
Misplaced responsibility at system boundaries
Opaque behavior under continuous operation

Recurring failure modes observed in agentic systems

Observed Failure Mode	Abstraction Gap
Autonomous resource consumption and unbounded action initiation	Execution models assume externally invoked computation with reactive resource control
Ambiguous agent lifecycles and costly termination	Process and service abstractions assume clear start, run, and termination phases
Distributed failure attribution across long decision sequences	Debugging and verification assume localized faults and linear execution
Misplaced responsibility at system boundaries	Control and correctness are assumed to reside outside the computational component
Opaque behavior under continuous operation	Interaction models assume explicit user commands and episodic execution

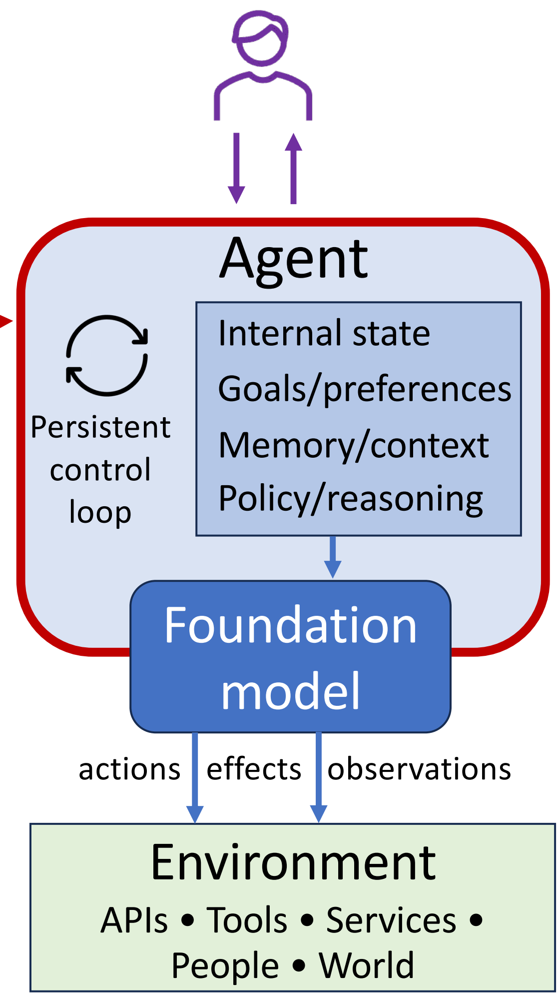
Recurring failure modes observed in agentic systems

Observed Failure Mode	Abstraction Gap	Resulting Research Challenges
Autonomous resource consumption and unbounded action initiation	Execution models assume externally invoked computation with reactive resource control	Proactive resource budgeting; anticipatory constraint enforcement; isolation for self-directed processes
Ambiguous agent lifecycles and costly termination	Process and service abstractions assume clear start, run, and termination phases	Lifecycle models for persistent agents; state preservation and transfer; principled shutdown semantics
Distributed failure attribution across long decision sequences	Debugging and verification assume localized faults and linear execution	Causal tracing across reasoning–action loops; partial-order explanations; long-horizon accountability
Misplaced responsibility at system boundaries	Control and correctness are assumed to reside outside the computational component	New interface contracts; explicit responsibility assignment; end-to-end reasoning for autonomous initiators
Opaque behavior under continuous operation	Interaction models assume explicit user commands and episodic execution	Oversight interfaces; escalation policies; scalable human supervision

New research problems

Beyond model capability and alignment

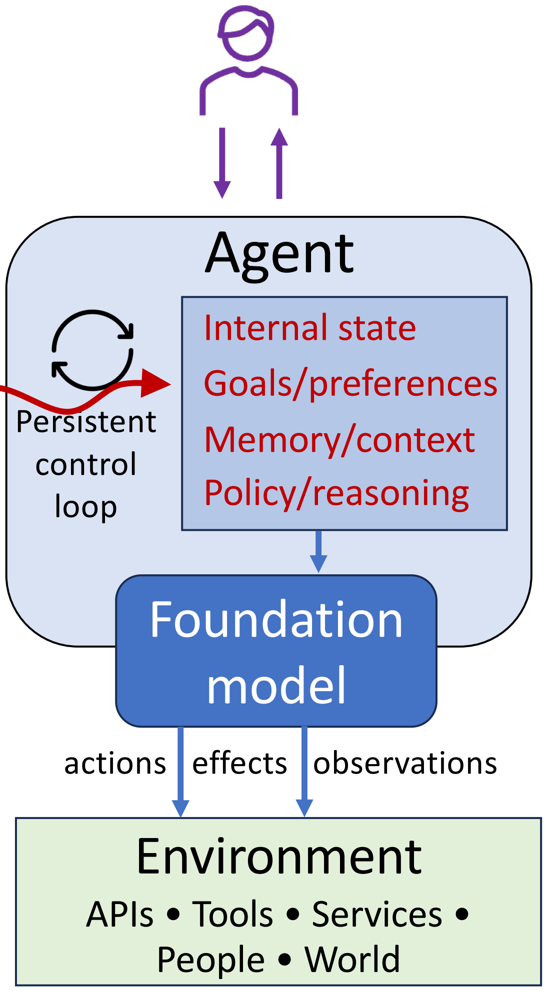
- Execution environments for persistent agents



New research problems

Beyond model capability and alignment

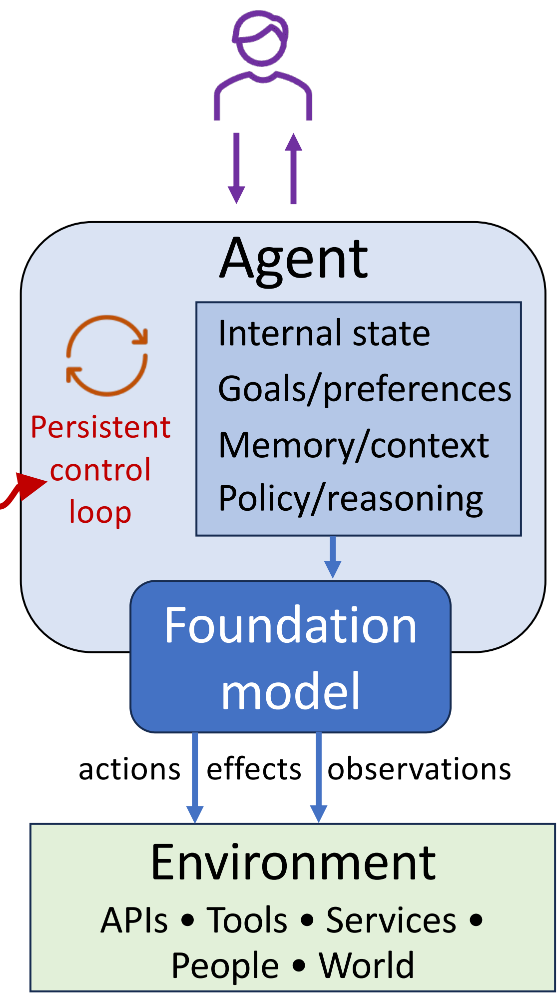
- Execution environments for persistent agents
- Programming models for constrained autonomy



New research problems

Beyond model capability and alignment

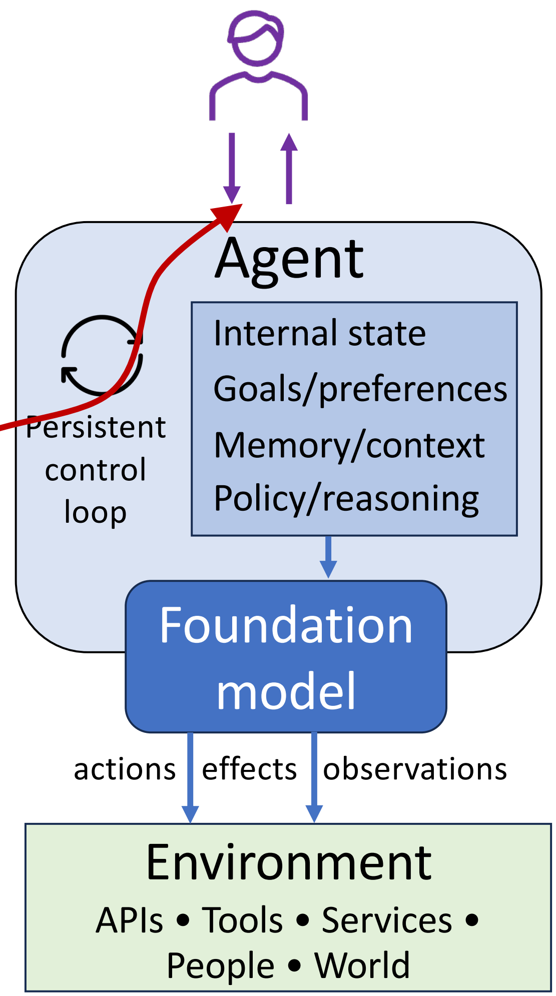
- Execution environments for persistent agents
- Programming models for constrained autonomy
- Correctness for adaptive, long-horizon behavior



New research problems

Beyond model capability and alignment

- Execution environments for persistent agents
- Programming models for constrained autonomy
- Correctness for adaptive, long-horizon behavior
- Oversight interfaces for continuous operation

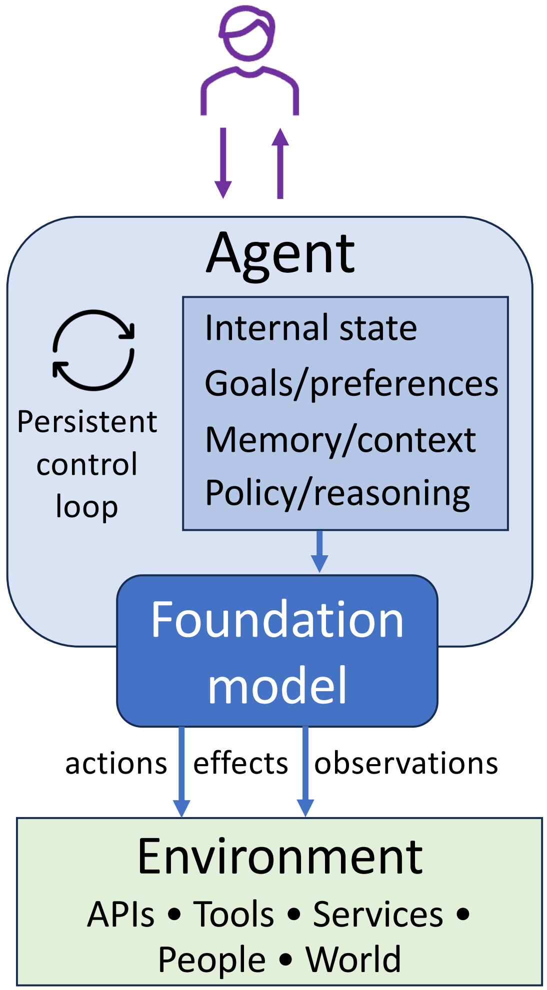


New research problems

Beyond model capability and alignment

- Execution environments for persistent agents
- Programming models for constrained autonomy
- Correctness for adaptive, long-horizon behavior
- Oversight interfaces for continuous operation

These CS research problems span systems, theory, HCI, etc.—and AI



Correctness under bounded rationality

Classical correctness assumes:

- Complete information
- Fixed objectives
- Known control flow
- Termination

Correctness under bounded rationality

Correctness in agentic systems must account for:

- Partial information and uncertainty
- Competing and evolving goals
- Long-horizon decision sequences
- Explicit resource and time limits

Thus:

- Safety envelopes instead of absolute guarantees
- Regret or performance bounds instead of optimality
- Temporal properties instead of postconditions

Human oversight at scale

Classical assumptions:

- Humans issue commands
- Systems execute deterministically
- Oversight is synchronous and local

Agentic reality

- Agents act continuously
- Decisions are distributed over time
- Human attention is the scarce resource

Systems research question

How do we design systems where:

- Oversight is **asynchronous**, not interactive
- Intervention is **exception-based**, not continuous
- Accountability is **aggregated over histories**, not events

Summary:
Three perspectives on agentic AI

globus labs
https://labs.globus.org

1) A **new approach to scientific discovery**
- Performed by semi-autonomous agents directed by high-level goals

2) A **new source of problems** for computer systems research
- Academy as an example and testbed; being used to tackle interesting problems in scalability, resilience, safety, … https://academy-agents.org

3) Agency as a **new organizing abstraction** for computer science
- An integrative concept, like networking and ML, that forces interactions across domains and exposes foundational abstraction gaps
- Agentic systems stress existing abstractions; agency provides a coherent way to understand why