Introduction to Agentic AI

Week 1: From LLMs to Autonomous Agents

PhD Course in Agentic Artificial Intelligence

12-Week Research-Level Course

Learning Objectives

Bloom's Taxonomy Levels Covered

- **Remember**: Define agent, tool use, ReAct paradigm, orchestration
- **Understand**: Explain difference between LLM (Large Language Model) inference and agentic behavior
- **Apply**: Implement a basic ReAct agent using LangChain
- **Analyze**: Compare reactive vs. deliberative architectures
- **Evaluate**: Assess capabilities and limitations of current LLM agents
- **Create**: Design an agent architecture for a novel problem

---

By end of this lecture, you will understand what makes an "agent" different from an LLM.

What is an Agent?

Classical Definition (Russell & Norvig, 2021)

- An **agent** is anything that perceives its environment through **sensors** and acts upon it through **actuators**
- Agents operate in an **environment** with **goals** to achieve

Formal Definition

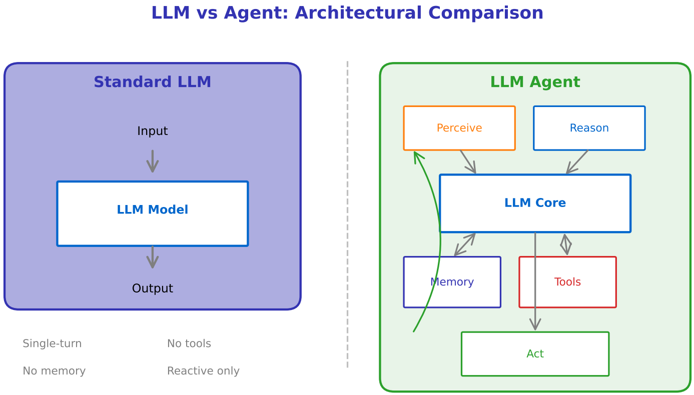
- Agent function: $f : P^* \to A$ (percept history to action)
- Agent program implements the agent function

Key Properties

- **Autonomy**: Operates without direct human intervention
- **Reactivity**: Responds to environment changes
- **Pro-activeness**: Takes initiative toward goals

Wooldridge & Jennings (1995): "Intelligent Agents: Theory and Practice"

From LLMs to Agents



Agents extend LLMs with perception, memory, tools, and action capabilities.

Why Agents Now?

The Capability Gap

- LLMs excel at single-turn generation but struggle with multi-step tasks
- Real-world problems require: planning, tool use, memory, iteration

Enabling Factors (2023-2024)

- **Reasoning**: Chain-of-Thought (CoT), Tree-of-Thoughts (ToT) prompting
- **Tool Use**: Function calling via APIs (Application Programming Interfaces)
- **Frameworks**: LangChain/LangGraph, AutoGen, CrewAI (agent orchestration)
- **Context**: 100K+ token (text unit) windows enable complex interactions

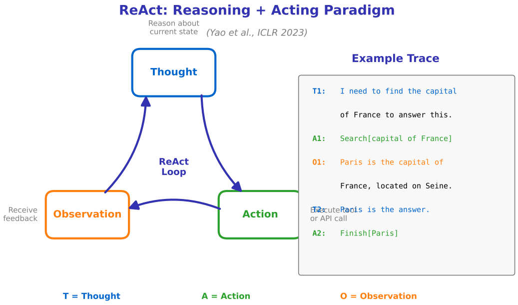
Current State

- Production agents: GitHub Copilot, Cursor, Claude Computer Use
- Research frontier: Multi-agent systems, embodied agents

2024 marked "Year of Agents" – from research prototypes to production systems.

ReAct: Reasoning + Acting (Yao et al., ICLR 2023)

- Interleave **reasoning traces** with **action execution**
- Thought: Internal reasoning about current state
- Action: External operation (search, calculate, API call)
- Observation: Feedback from environment



ReAct significantly outperforms action-only or reasoning-only baselines.

ReAct: Formal Definition

Mathematical Formulation

- State space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$
- Policy: $\pi(a_t | s_t, h_{<t})$ where h is interaction history

ReAct Trajectory

$$\tau = (s_0, t_1, a_1, o_1, t_2, a_2, o_2, \ldots, t_n, a_n, o_n)$$

where t_i = thought, a_i = action, o_i = observation

Key Insight

- Thoughts provide interpretable reasoning traces
- Actions ground the agent in external world
- Observations close the loop for iterative refinement

The trajectory τ provides a complete audit trail of agent reasoning.

Agent Architecture Taxonomy

By Reasoning Strategy

- **Reactive**: Direct stimulus-response (simple reflex)
- **Deliberative**: Plan then execute (model-based)
- **Hybrid**: Combine reactive and deliberative layers

By Organization

- **Single-agent**: One LLM handles all tasks
- **Multi-agent**: Specialized agents collaborate
- **Hierarchical**: Manager agents delegate to workers

By Memory

- **Stateless**: No memory between interactions
- **Short-term**: In-context memory (conversation history)
- **Long-term**: Vector DB (embedding search), knowledge graph (entities)

Architecture choice depends on task complexity and latency requirements.

Core Agent Components

1. **LLM Core (Brain)**
   - Reasoning, planning, decision-making
   - GPT-4, Claude, Gemini, open-source alternatives
2. **Memory System**
   - Short-term: Conversation context, working memory
   - Long-term: Vector stores, knowledge graphs
3. **Tool Interface**
   - Function calling, MCP (Model Context Protocol)
   - APIs, databases, code execution
4. **Planning Module**
   - Task decomposition, goal tracking
   - Reflexion (learning from failures), self-correction

These four components form the backbone of modern agent systems.

Current Agent Landscape

Commercial Agents

- **GitHub Copilot**: Code completion and generation
- **Devin** (Cognition): Autonomous software engineer
- **Claude Computer Use**: Desktop automation

Research Systems

- **AutoGPT**: Goal-directed autonomous agent
- **BabyAGI**: Task-driven autonomous agent
- **Voyager** (NVIDIA): Minecraft exploration agent

Frameworks

- LangChain/LangGraph, AutoGen, CrewAI, Semantic Kernel

The field is rapidly evolving – new systems appear weekly.

Agent Capabilities

What Agents Can Do Well

- Multi-step task execution with tool use
- Information retrieval and synthesis
- Code generation and debugging
- Document analysis and summarization

Current Limitations

- **Reliability**: Hallucinations (fabricated facts), error propagation
- **Planning**: Struggle with long-horizon tasks
- **Evaluation**: Hard to measure agent quality
- **Safety**: Unintended actions, jailbreaks (safety bypass)

Key Metric

- AgentBench: GPT-4 achieves 30% on complex tasks

Agents are powerful but not yet reliable for high-stakes autonomous operation.

Research Frontiers

Open Problems

- **Scalable alignment** (human values): How to align agents?
- **Compositional generalization**: Transfer to new tasks
- **Long-term memory**: Efficient persistent memory
- **Multi-agent coordination**: Emergent communication

Emerging Directions

- Embodied agents (robotics, simulation)
- Agent-agent learning and co-evolution
- Constitutional AI (principle-based safety)
- World models (environment simulation) for planning

These open problems define the research agenda for the next 3-5 years.

Course Roadmap: 12 Weeks

Week	Topic	Key Concept
1	Introduction to Agentic AI	ReAct paradigm
2	LLM Foundations	CoT, ToT, prompting
3	Tool Use & Function Calling	MCP protocol
4	Planning & Reasoning	Reflexion, LATS
5	Multi-Agent Architectures	Coordination
6	Agent Frameworks	LangGraph, AutoGen
7	RAG (Retrieval-Augmented Gen.)	Self-RAG, CRAG
8	GraphRAG & Knowledge	Knowledge graphs
9	Hallucination Prevention	Verification
10	Agent Evaluation	Benchmarks
11	Domain Applications	Finance, Healthcare
12	Research Frontiers	Projects

Each week includes slides, notebook, exercise, and paper reading.

Required Readings

This Week

- Yao et al. (2023). "ReAct: Synergizing Reasoning and Acting in Language Models." *ICLR 2023*. arXiv:2210.03629

Supplementary

- Wang et al. (2024). "A Survey on Large Language Model based Autonomous Agents." arXiv:2308.11432
- Xi et al. (2023). "The Rise and Potential of LLM Based Agents." arXiv:2309.07864
- Sumers et al. (2024). "Language Agents: From Next-Token Prediction to Digital Automation." arXiv:2403.12897

Start with ReAct paper – it's foundational for everything that follows.

Summary and Key Takeaways

Key Concepts

- **Agent**: Autonomous system that perceives, reasons, and acts
- **ReAct**: Interleave reasoning traces with actions
- **Components**: LLM core, memory, tools, planning

Key Equations

- Policy: $\pi(a_t|s_t, h_{<t})$
- Trajectory: $\tau = (s_0, t_1, a_1, o_1, \ldots)$

Next Week

- LLM Foundations for Agents
- Chain-of-Thought and Tree-of-Thought prompting
- Context window management

Agents = LLM + Memory + Tools + Planning