

# LLM Agents

---

## Agentic AI Frameworks & AutoGen

Guest Speaker: Chi Wang

# Agenda

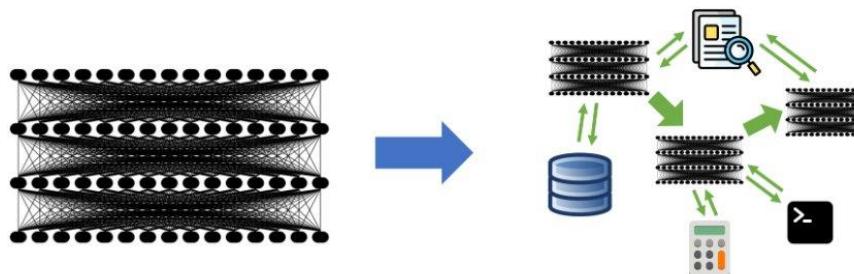
- Agentic AI Frameworks
- AutoGen

What are future AI applications like?

How do we empower every developer to build them?

# What are future AI applications like?

- Generative
  - Generate content like text & image
- 
- Agentic
  - Execute complex tasks on behalf of human



Zaharia et al. 2024. The Shift from Models to Compound AI Systems

# Examples of Agentic AI

- Personal assistants
- Autonomous robots
- Gaming agents
- Science agents
- Web agents
- Software agents



Creative Writing Coach

I'm excited to read your work and give you feedback to improve your skills.



Laundry Buddy

Ask me anything about stain settings, sorting and even laundry.



Game Time

I can quickly explain board games or card games to players of any skill level. Let the games begin!



Tech Advisor

From setting up a printer to troubleshooting a device, I'm here to help you step-by-step.



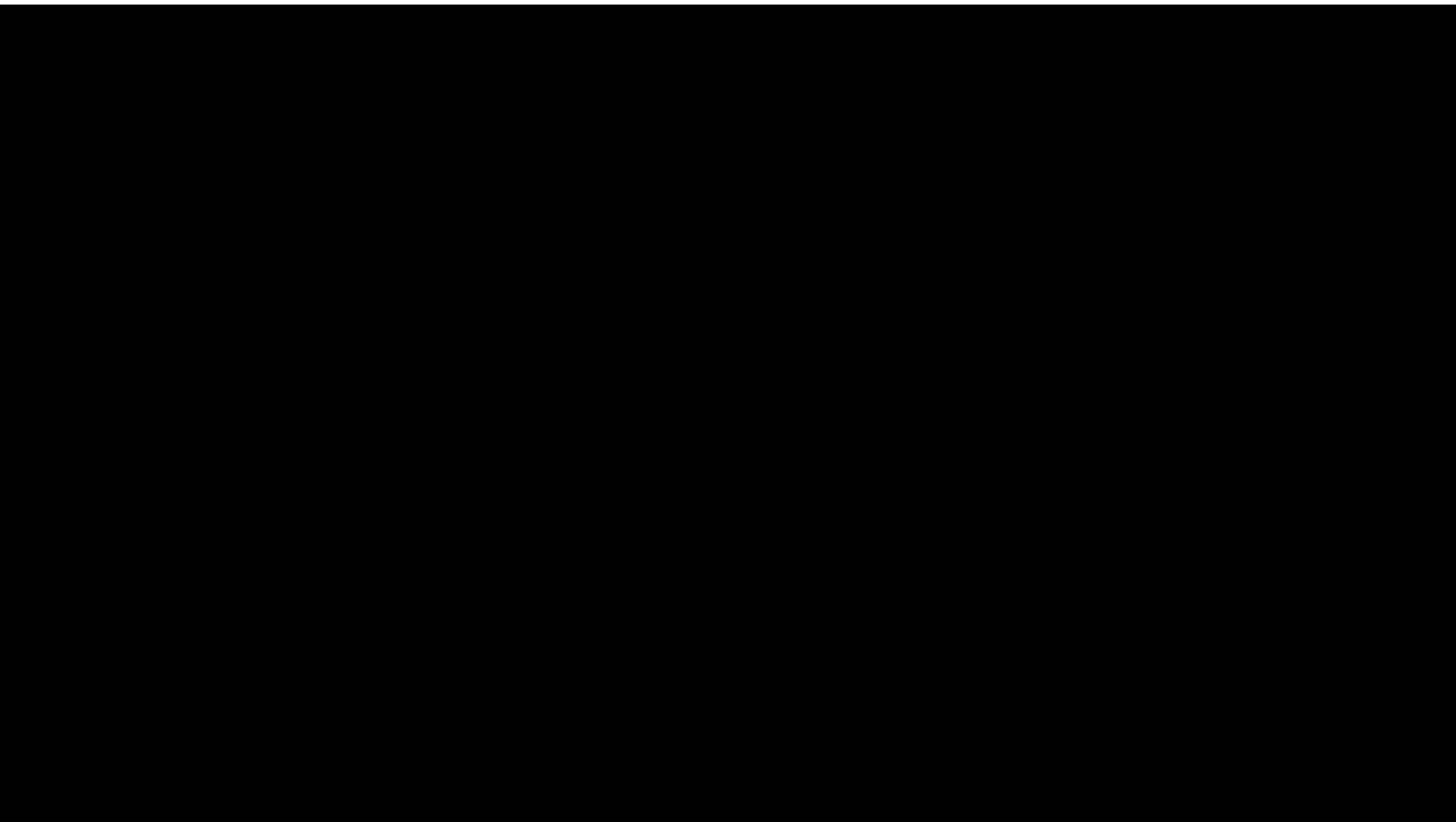
Sticker Whiz

I'll help turn your wildest dreams into die-cut stickers, shipped to your door.



The Negotiator

I'll help you advocate for you to get better outcomes. Be a negotiator.

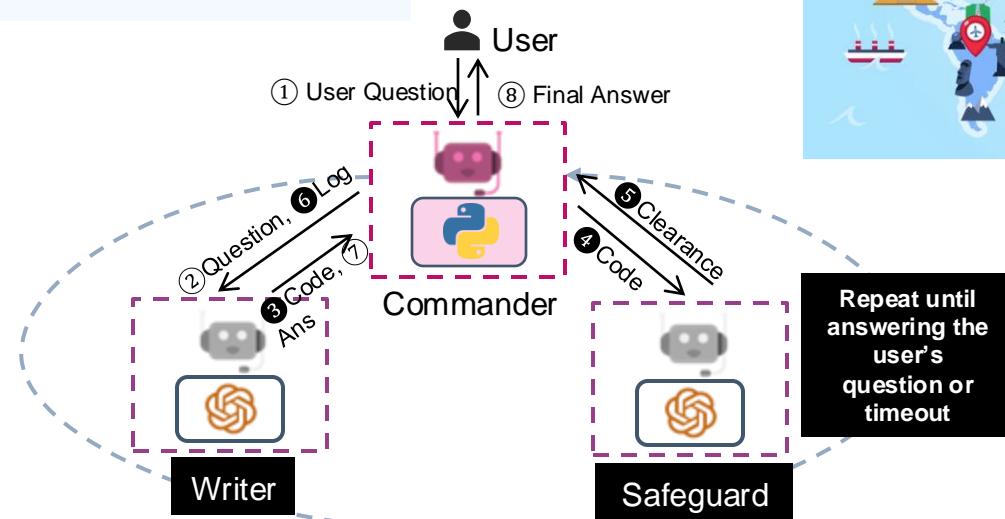


# Key Benefits of Agentic AI

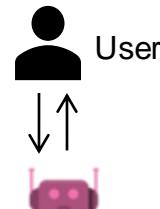
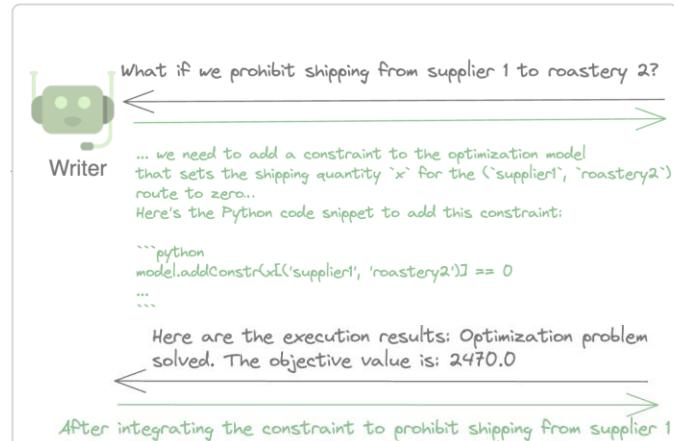
---

- Useful Interface
  - Natural interaction with human agency
- Strong Capability
  - Operate with minimal human intervention
- Useful Architecture
  - Intuitive programming paradigm

What if we prohibit shipping  
from supplier 1 to roastery  
2?



(Writer nested in Commander, Triggered by User)



What if we prohibit shipping from supplier 1 to roastery 2?

Commander

(Safeguard nested in Commander, Triggered by Writer)

Is the code safe?

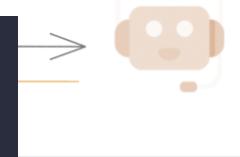
```
'''python
model.addConstr(x[('supplier1', 'roastery2')] == 0
...
'''
```

```
Optimize a model with 11 rows, 18 columns and 36 nonzeros
Model fingerprint: 0x8aa2c280
Variable types: 0 continuous, 18 integer (0 binary)
Coefficient statistics:
    Matrix range [1e+00, 1e+00]
    Objective range [2e+00, 1e+01]
    Bounds range [0e+00, 0e+00]
    RHS range [2e+01, 2e+02]
Found heuristic solution: objective 2900.0000000
Presolve time: 0.00s
Presolved: 11 rows, 18 columns, 36 nonzeros
Variable types: 0 continuous, 18 integer (0 binary)
Found heuristic solution: objective 2896.0000000
Root relaxation: objective 2.470000e+03, 11 iterations, 0.00 seconds (0.00 work units)

      Nodes |      Current Node |      Objective Bounds      |     Work
Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
*     0     0           0    2470.0000000  2470.00000  0.00%  -      0s

Explored 1 nodes (11 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 14 (of 14 available processors)

Solution count 3: 2470 2896 2900
```



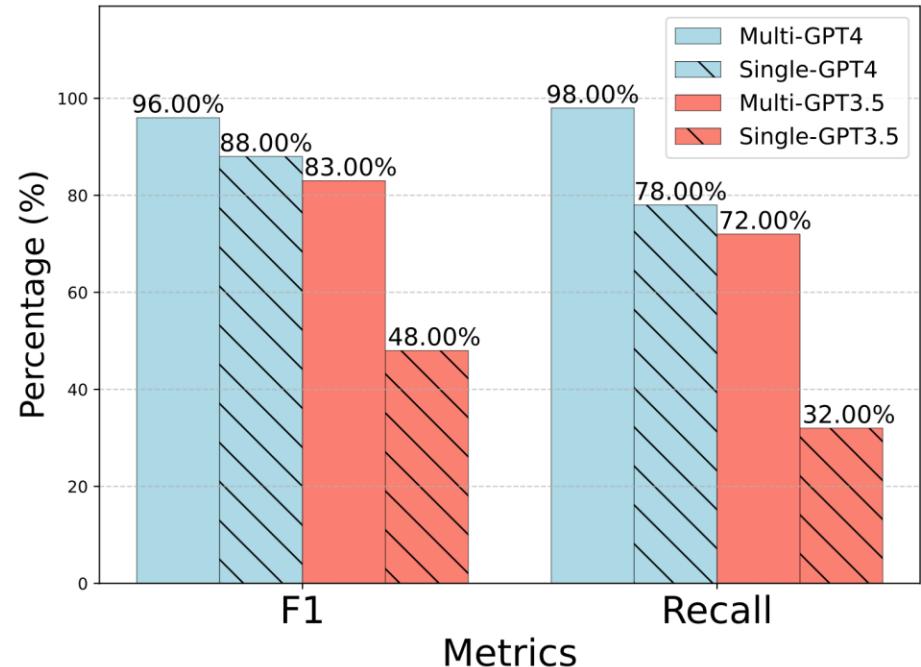
```
writer = Writer("writer", llm_config=llm_config)
safeguard = autogen.AssistantAgent("safeguard", llm_config=llm_config)
commander = Commander("commander", llm_config=llm_config)
user = autogen.UserProxyAgent("user")

writer_chat_queue = [{"recipient": writer, "message": writer_init_message, "summary_method": writer_success_summary}]
safeguard_chat_queue = [{"recipient": safeguard, "message": safeguard_init_message, "max_turns": 1, "summary_method": safeguard_summary}]
commander.register_nested_chats(safeguard_chat_queue, trigger="writer")
commander.register_nested_chats(writer_chat_queue, trigger="user")

chat_res = user.initiate_chat(commander, message="What if we prohibit shipping from supplier 1 to roastery 2?")
```

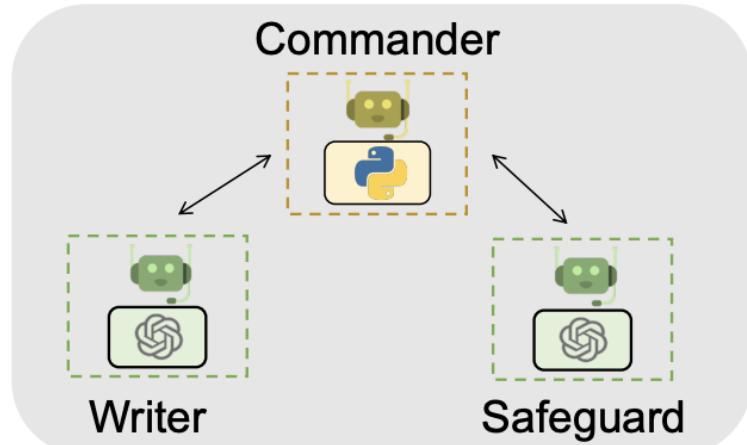
# Agentic Programming

- Handle more complex tasks / Improve response quality
  - Improve over natural iteration
  - Divide & conquer
  - Grounding & validation



# Agentic Programming

- Easy to understand, maintain, extend
  - Modular composition
  - Natural human participation
  - Fast & creative experimentation



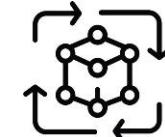
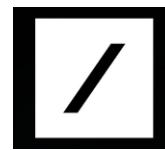
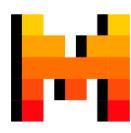
# Agentic AI Framework Desiderata

---

- Intuitive unified agentic abstraction
- Flexible multi-agent orchestration
- Effective implementation of agentic design patterns
- Support diverse application needs

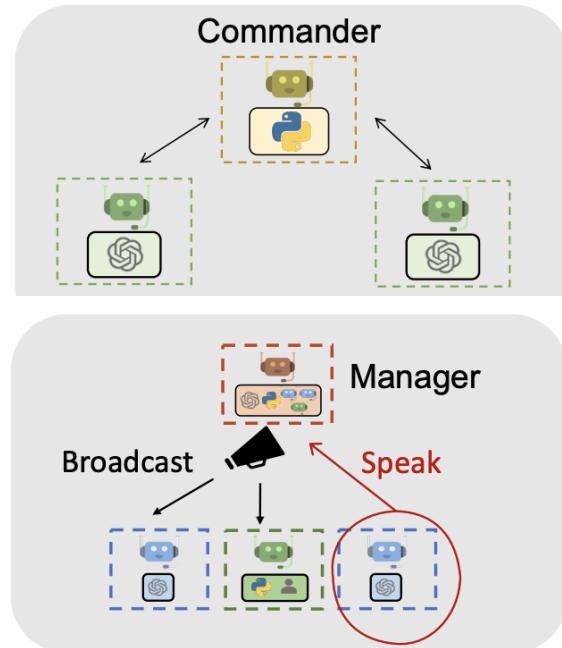
# Agentic Abstraction

Unify models, tools, human for compound AI systems



# Multi-Agent Orchestration

- Static/dynamic
- NL/PL
- Context sharing/isolation
- Cooperation/competition
- Centralized/decentralized
- Intervention/automation

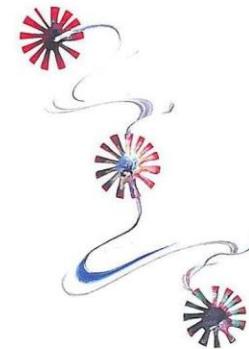


# Agentic Design Patterns

- Conversation
- Prompting & reasoning
- Tool use
- Planning
- Integrating multiple models, modalities and memories

How Talk Can Change Your Life

## CONVERSATION



THEODORE  
ZELDIN

# Examples of Agentic AI Frameworks

- AutoGen
  - Multi-agent conversation programming
    - Comprehensive & flexible
    - Integrable with other frameworks like OpenAI Assistant, LlamaIndex, LangChain
- Langchain-based
  - Langraph
    - Graph-based control flow
  - CrewAI
    - High-level static agent-task workflow

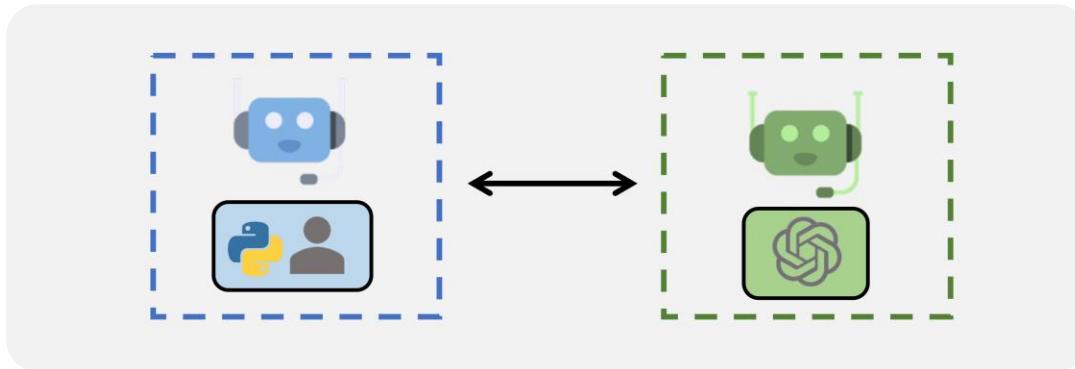
# Agenda

- Agentic AI Frameworks
- **AutoGen**

What are future AI applications like?

How do we empower every developer to build them?

# AutoGen: A programming framework for agentic AI



Initially developed in FLAML (Nov 2022)  
Spined off to a standalone repo (October 2023)  
Standalone GitHub organization AutoGen-AI (August 2024)



**AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework**  
 Qinyun Wu<sup>1</sup>, Gagan Bansal<sup>2</sup>, Jieyu Zhang<sup>2</sup>, Yiran Wu<sup>1</sup>, Shaokun Zhang<sup>1</sup>, Erkang Zhu<sup>1</sup>, Bobin Li<sup>1</sup>, Li Jiang<sup>2</sup>, Xiaoyun Zhang<sup>2</sup>, and Chi Wang<sup>2</sup>

<sup>1</sup>Pennsylvania State University  
<sup>2</sup>Microsoft

<sup>1</sup>University of Washington

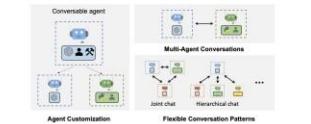


Figure 1. AutoGen enables complex LLM-based workflows using multi-agent communication. (Left) AutoGen agents are customizable and can be based on LLMs, tools, humans and even a combination of them. (Top-right) Agents can converse to solve tasks. (Bottom-right) The framework supports many additional complex conversation patterns.

**Abstract**  
 This technical report presents **AutoGen**, a new framework that enables development of LLM applications using multiple agents that can communicate with each other to solve tasks. AutoGen agents are controllable, composable, and semantically-aware participants. They can operate in different domains and interact with each other in a distributed manner. AutoGen has three main design offices multiple advantages: a) it graciously negotiates the strong but imperfect generation and execution of AI models; b) it provides a modular and extensible architecture for AI applications while providing valuable automation through conversations between agents; c) it simplifies and unifies the development of AI applications. We demonstrate the power of AutoGen by showing a real-world example of how developers can easily use AutoGen to effectively a range from coding, mathematics, operations research, medical question answering, etc.

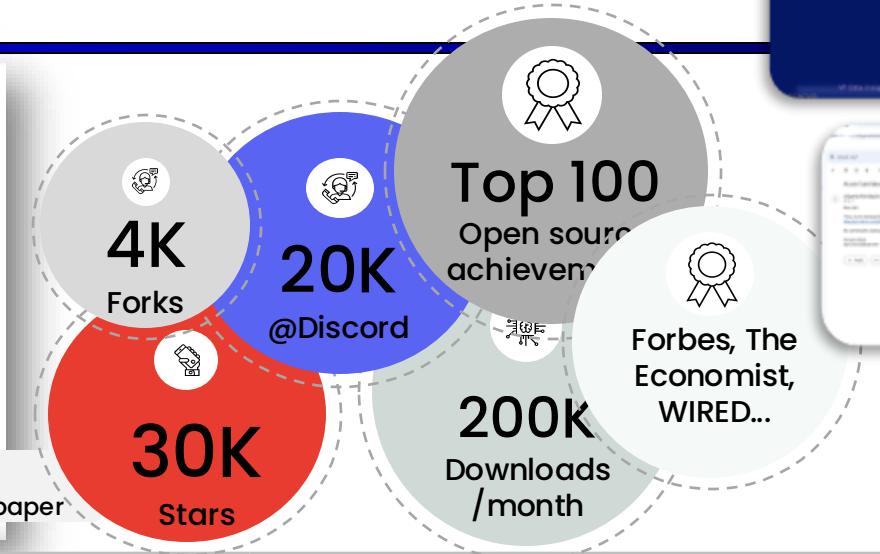
**1 Introduction**  
 Large Language Models (LLMs), like GPT-3, have demonstrated remarkable, and sometimes bizarre, AI applications, ranging from music generation to generating financial reports. However, there is a lack of general-purpose frameworks that enable users to build their own AI applications.

<https://arxiv.org/pdf/2308.08155v1.pdf>

**03.2023: Initial Prototype**  
 - Flexible multi-agent conversation framework  
 - Code/function execution

**10.2023: Top trending on GitHub**

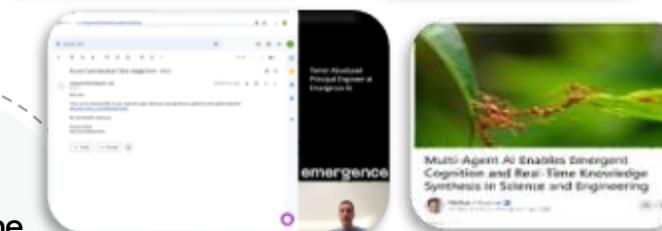
**12.2023: 5 favorite AI papers by The Sequence**



**Best Paper at ICLR 2024 LLM Agents Workshop**



**COLM**

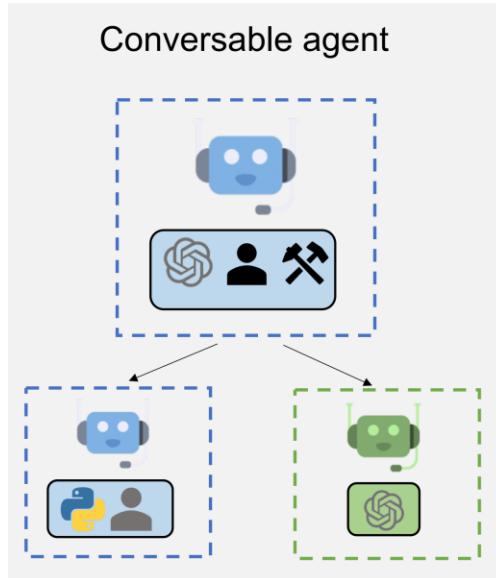


**BETTERFUTURELABS**  
 Uses AutoGen

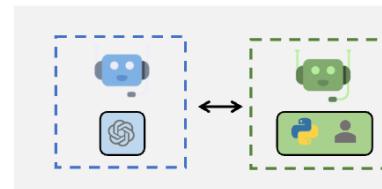
Andrew Ng  
 Columbia & Stanford



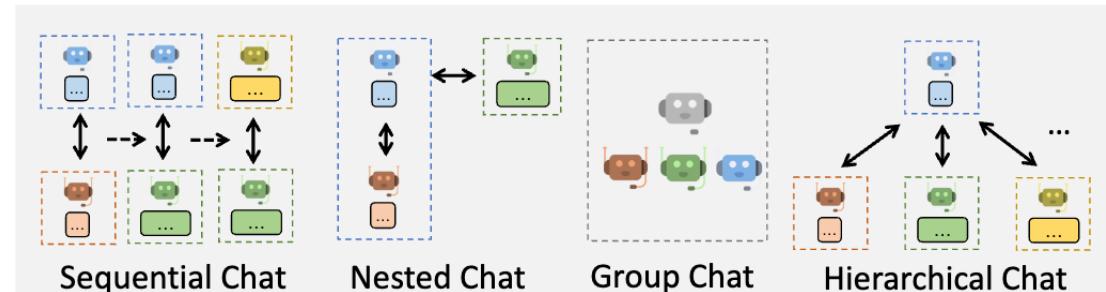
Define agents:  
Conversable & Customizable



Get them to talk:  
Conversation Programming



Multi-Agent Conversations

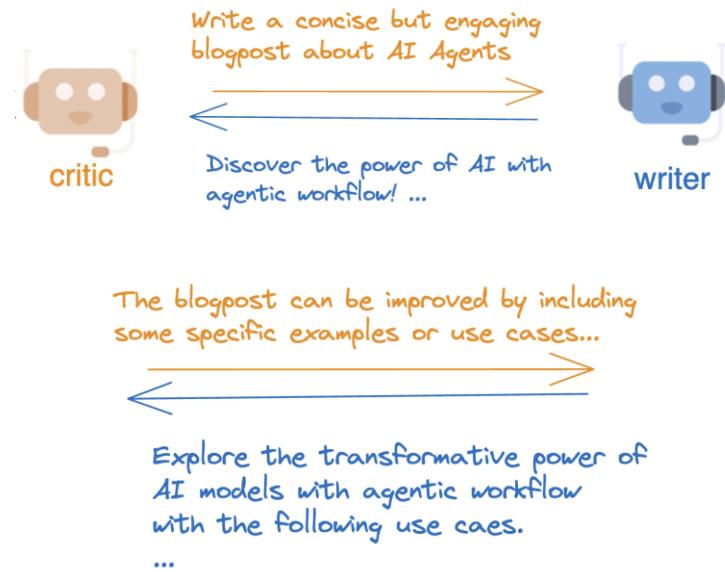


Flexible Conversation Patterns

# Blogpost Writing with Reflection

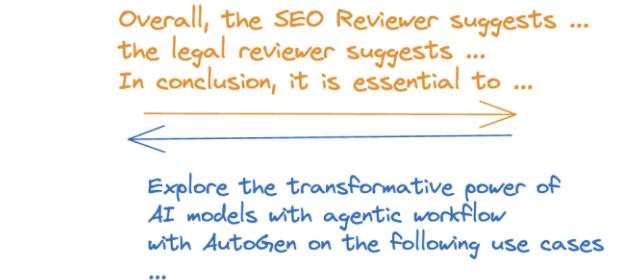
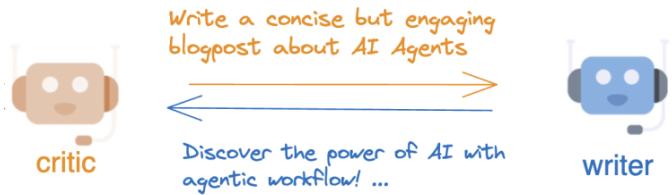
```
writer = autogen.AssistantAgent(  
    name="Writer",  
    system_message="You are a writer...",  
    llm_config=llm_config,  
)  
  
critic = autogen.AssistantAgent(  
    name="Critic",  
    is_termination_msg=lambda x: x.get("content", "").find("TERMINATE") >= 0,  
    llm_config=llm_config,  
    system_message="You are a critic...",  
)  
  
critic.initiate_chat(  
    recipient=writer,  
    message=task,  
    max_turns=2,  
    summary_method="last_msg"  
)
```

## Two-Agent Reflection



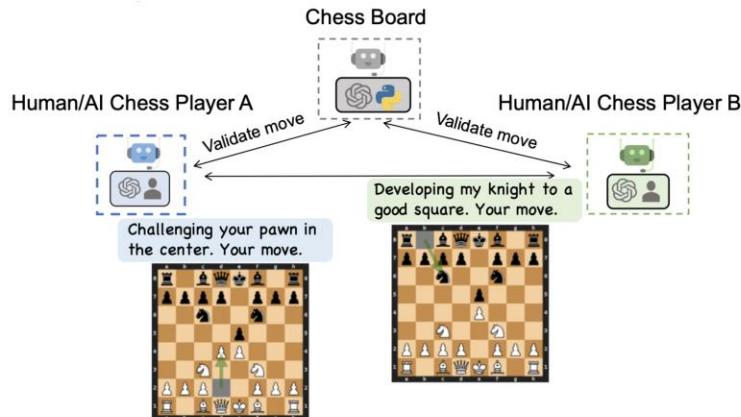
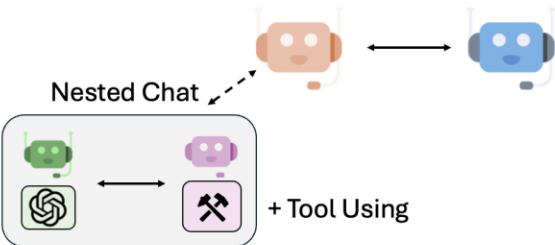
# Blogpost Writing with Advanced Reflection

```
critic.register_nested_chats(  
    ...  
    review_chats,  
    ...  
    trigger=writer,  
)  
  
critic.initiate_chat(  
    recipient=writer,  
    message=task,  
    max_turns=2,  
    summary_method="last_msg"  
)
```



## Nested Chat

# Conversational Chess



```
from autogen import register_function

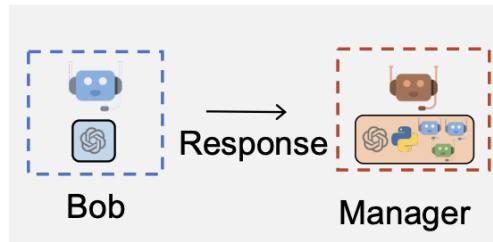
for caller in [player_white, player_black]:
    register_function(
        get_legal_moves,
        caller=caller,
        executor=board_proxy,
        name="get_legal_moves",
        description="Get legal moves.",
    )

    register_function(
        make_move,
        caller=caller,
        executor=board_proxy,
        name="make_move",
        description="Call this tool to make a move.",
    )
```

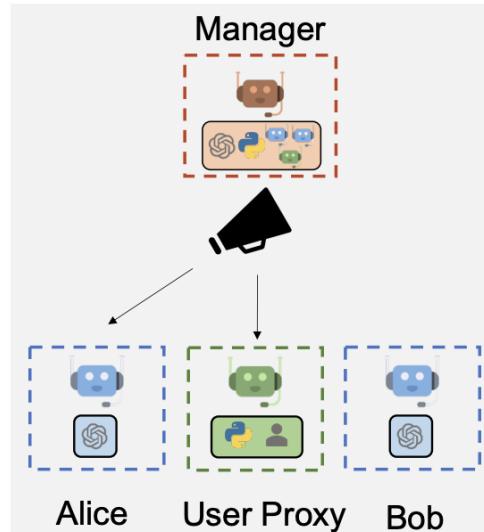
# Complex Task Planning and Solving with Group Chat



1. Select a Speaker



2. Ask the Speaker to Respond



3. Broadcast

# Complex Task Planning and Solving with Group Chat

## StateFlow - Build State-Driven Workflows with Customized Speaker Selection in GroupChat

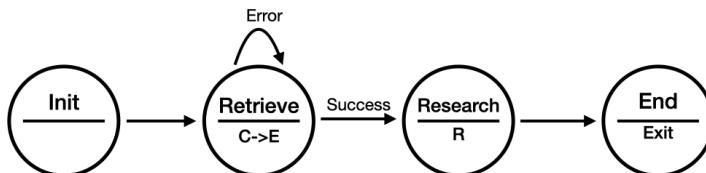
February 29, 2024 · 7 min read



Yiran Wu

PhD student at Pennsylvania State University

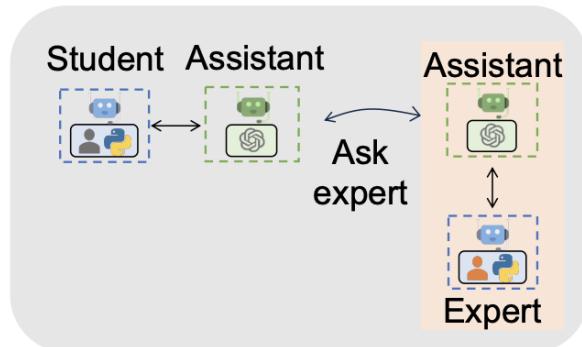
**TL;DR:** Introduce Stateflow, a task-solving paradigm that conceptualizes complex task-solving processes backed by LLMs as state machines. Introduce how to use GroupChat to realize such an idea with a customized speaker selection function.



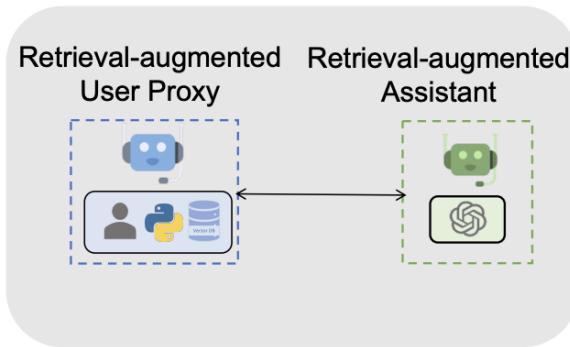
C: Coder  
E: Code Executor  
R: Research

```
def state_transition(last_speaker, groupchat):  
    messages = groupchat.messages  
  
    if last_speaker is initializer:  
        # init -> retrieve  
        return coder  
    elif last_speaker is coder:  
        # retrieve: action 1 -> action 2  
        return executor  
    elif last_speaker is executor:  
        if messages[-1]["content"] == "exitcode: 1":  
            # retrieve --(execution failed)--> retrieve  
            return coder  
        else:  
            # retrieve --(execution success)--> research  
            return scientist  
    elif last_speaker == "Scientist":  
        # research -> end  
        return None
```

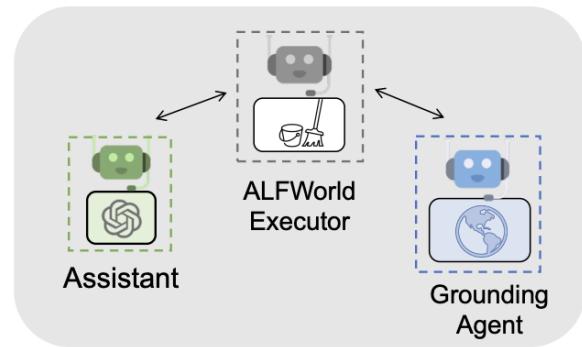
```
groupchat = autogen.GroupChat(  
    agents=[initializer, coder, executor, scientist],  
    messages=[],  
    max_round=20,  
    speaker_selection_method=state_transition,  
)
```



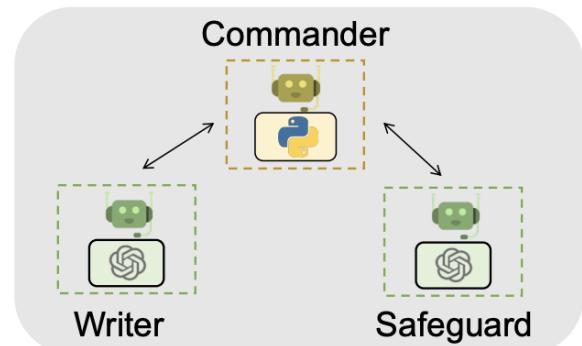
A1. Math Problem Solving



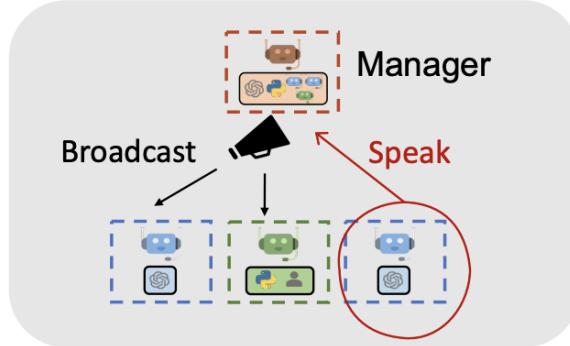
A2. Retrieval-augmented Q&A



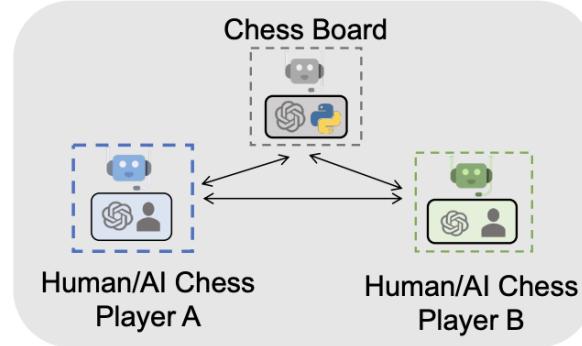
A3. Decision Making in Embodied Agents



A4. Supply-Chain Optimization



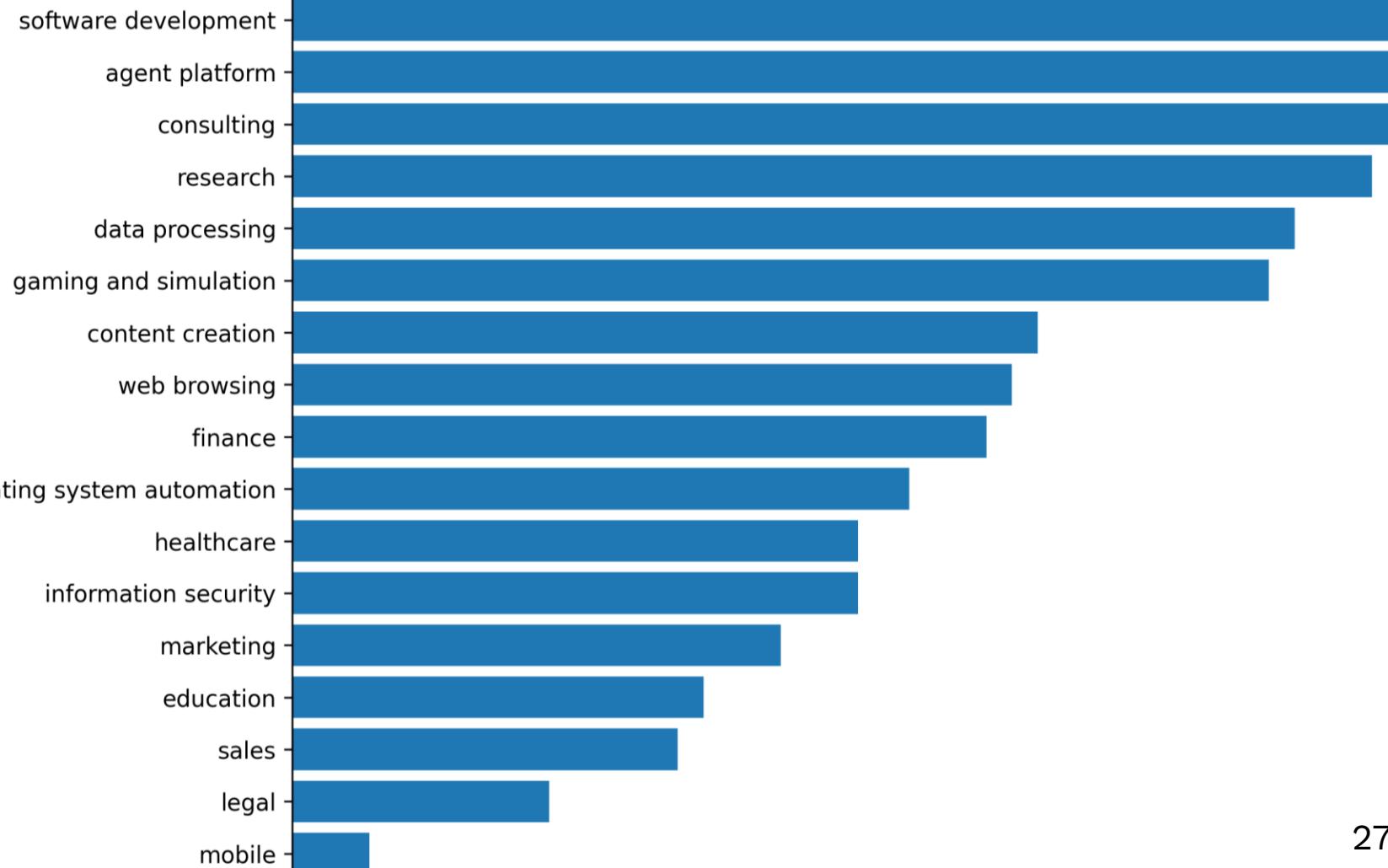
A5. Dynamic Task Solving with Group Chat



A6. Conversational Chess

For more examples: <https://autogen-ai.github.io/autogen/docs/notebooks>

Area





arXiv:2409.05556v1 [cs.AI] 9 Sep 2024

---

## SCIAGENTS: AUTOMATING SCIENTIFIC DISCOVERY THROUGH MULTI-AGENT INTELLIGENT GRAPH REASONING<sup>†</sup>

---

Alireza Ghafarollahi

Laboratory for Atomistic and Molecular Mechanics (LAMM)  
Massachusetts Institute of Technology  
77 Massachusetts Ave.  
Cambridge, MA 02139, USA

Markus J. Buehler

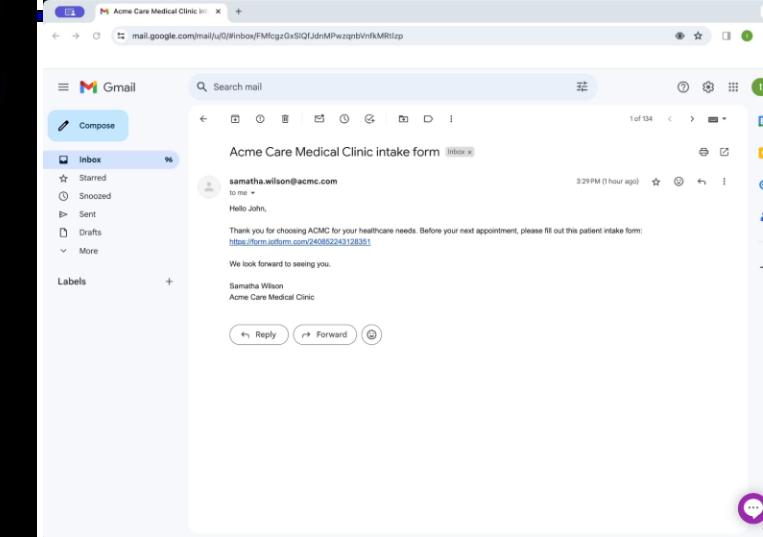
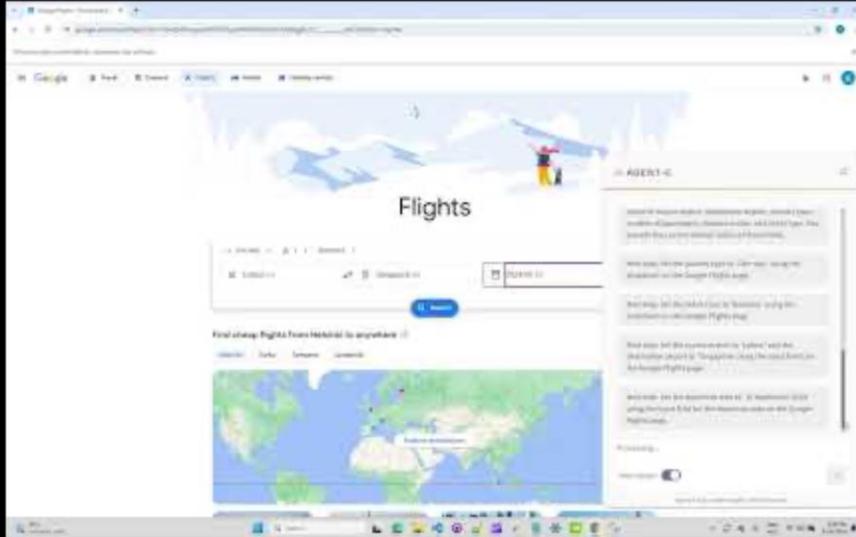
Laboratory for Atomistic and Molecular Mechanics (LAMM)  
Center for Computational Science and Engineering  
Schwarzman College of Computing  
Massachusetts Institute of Technology  
77 Massachusetts Ave.  
Cambridge, MA 02139, USA

Correspondence: mbuehler@MIT.EDU

### ABSTRACT

A key challenge in artificial intelligence is the creation of systems capable of autonomously advancing scientific understanding by exploring novel domains, identifying complex patterns, and uncovering previously unseen connections in vast scientific data. In this work, we present SciAgents, an approach that leverages three core concepts: (1) the use of large-scale ontological knowledge graphs to organize and interconnect diverse scientific concepts, (2) a suite of large language models (LLMs) and data retrieval tools, and (3) multi-agent systems with *in-situ* learning capabilities. Applied to biologically inspired materials, SciAgents reveals hidden interdisciplinary relationships that were previously considered unrelated, achieving a scale, precision, and exploratory power that surpasses traditional human-driven research methods. The framework autonomously generates and refines research hypotheses, elucidating underlying mechanisms, design principles, and unexpected material properties. By integrating these capabilities in a modular fashion, the intelligent system yields material discoveries, critique and improve existing hypotheses, retrieve up-to-date data about existing research, and highlights their strengths and limitations. Our case studies demonstrate scalable capabilities to combine generative AI, ontological representations, and multi-agent modeling, harnessing a ‘swarm of intelligence’ similar to biological systems. This provides new avenues for materials discovery and accelerates the development of advanced materials by unlocking Nature’s design principles.

**Keywords** Scientific AI · Multi-agent system · Large language model · Natural language processing · Materials design · Bio-inspired materials · Knowledge graph · Biological design



Publication	Task success rates on websites							
	Allrecipe	Amazon	Apple	Arxiv	Github	Booking	ESPN	Coursera
He et al. 2024 (text)	57.8	43.1	36.4	50.4	63.4	2.3	28.6	24.6
He et al. 2024 (multi)	51.1	52.9	62.8	52.0	59.3	32.6	47.0	57.9
Lutz et al. 2024 (text)	60	43.9	60.5	51.2	22.0	<b>38.6</b>	59.1	51.1
<b>Agent-E (text)</b>	<b>71.1</b>	<b>70.7</b>	<b>74.4</b>	<b>62.8</b>	<b>82.9</b>	27.3	<b>77.3</b>	<b>85.7</b>

Publication	Task success rates on websites							
	Dictionary	BBC	Flights	Maps	Search	Hug.Face	Wolfram	Overall
He et al. 2024 (text)	66.7	45.2	7.1	62.6	75.2	31.0	60.2	44.3
He et al. 2024 (multi)	71.3	60.3	<b>51.6</b>	64.3	77.5	55.8	60.9	57.1
Lutz et al. 2024 (text)	<b>86.0</b>	<b>81.0</b>	0.0	39.0	67.4	53.5	65.2	52.6
<b>Agent-E (text)</b>	81.4	73.8	35.7	<b>87.8</b>	<b>90.7</b>	<b>81.0</b>	<b>95.7</b>	<b>73.1</b>

arXiv > cs > arXiv:2407.13032

Search...

Help | Ad

Computer Science > Artificial Intelligence

[Submitted on 17 Jul 2024]

# Agent-E: From Autonomous Web Navigation to Foundational Design Principles in Agentic Systems

29 / 37

Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jaqmohan, Aditya Vempaty, Ravi Kokku

AutoGen is simply the gold-standard when it comes to applied enterprise agentic orchestration. It allows us to easily and rapidly explore multiple agentic configurations, conducting experiments at scale. Without this specific capability, you'll never realize the full potential of multi-agent systems. Thank you AutoGen!

Medo Eldin, Cofounder and CEO @  
Terrascape AI



## Enterprise Customer Interest:

Accounting, Airlines, Biotech, Consulting, Construction, Consumer Packaged Goods, Electronics, Energy, Entertainment, Finance, Fintech, Government, Healthcare, Manufacturer, Metals, Motion & Control, Pharmacy, Research, Retailer, Social Media, Software, Supply Chain, Technology, Telecom...



## Used/Contributed by:

AgentCloud, AgentCoin, AgentLabs, AgentOps, AT&T, BetterFutureLabs, Emergence AI, FastAgency, Google, Meta, Metropolis, Microsoft, MIT, MongoDB, Nexla, Nordstrom, Novo Nordisk, PingCap, Princeton, Qdrant, Rutgers, Santander, Terrascape, Trilogy Education, Tufts University, UC Berkeley, UCL, U Washington, Weaviate, X-force, XGPT

## Evaluation

Agent-based  
evaluation tools

Examples:  
AgentEval,  
AutoDefense,  
Observability

## Interface

Lower the barrier  
of programming

Examples:  
AutoBuild,  
Composable Actor  
Platform

## Learning/ Teaching/ Optimization

Agents made  
smarter

Examples:  
AgentOptimizer,  
EcoAssistant,  
Learn to Cooperate

# AutoBuild Multi-Agent Systems



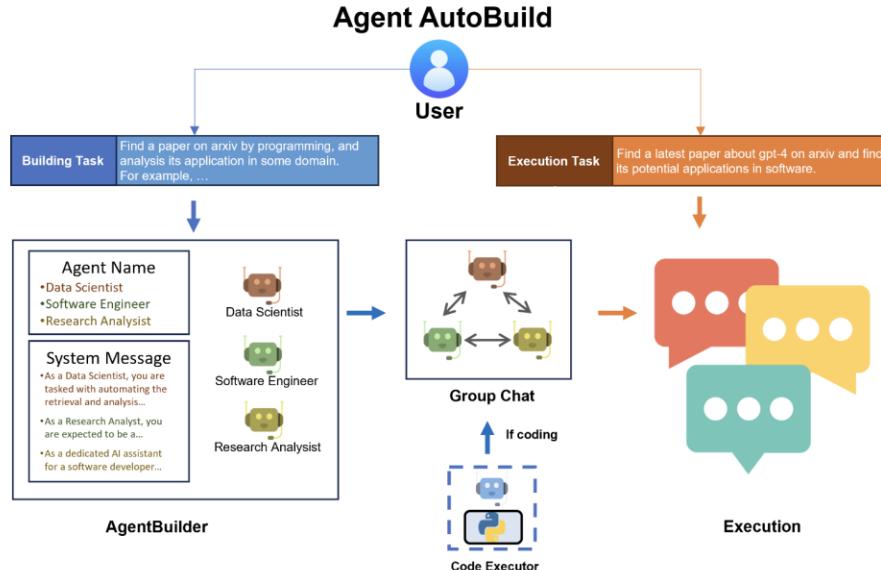
Linxin Song

MS student at Waseda University

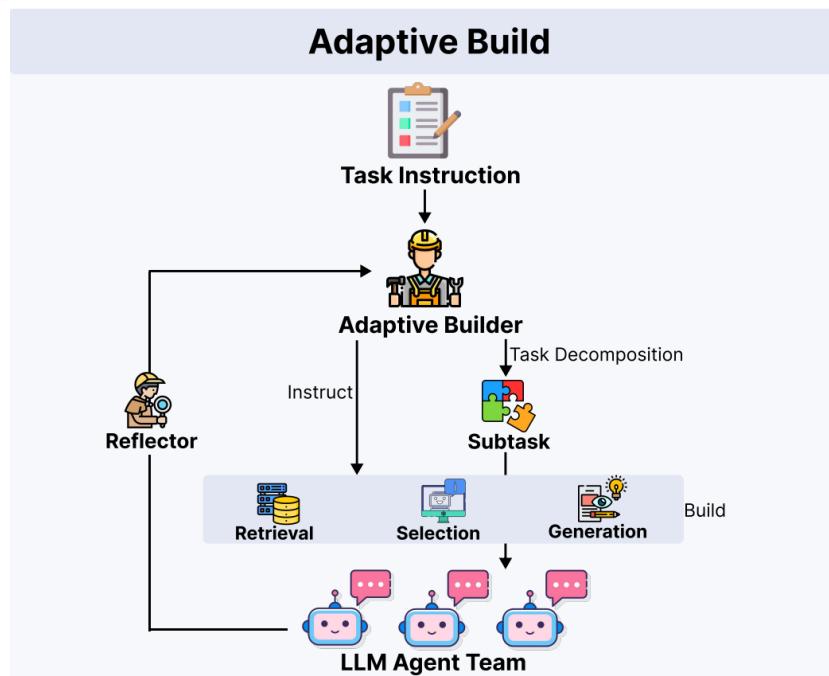


Jieyu Zhang

PhD student at University of Washington



# AutoBuild Multi-Agent Systems



# AutoBuild Multi-Agent Systems

Method	Mathematics	Programming	Data Analysis	(Sci) Chemistry	(Sci) Physics	Avg.
Vanilla LLM	51.53	84.76	6.61	39.02	31.25	40.98
Meta-prompting	68.88	19.51	39.69	41.46	43.75	43.47
AutoAgents	56.12	84.76	57.98	60.98	50.00	63.58
AutoGen: Assistant + Executor	74.49	93.90	82.88	60.98	43.75	79.89
Captain Agent	<b>77.55</b>	<b>96.95</b>	<b>88.32</b>	<b>65.85</b>	<b>53.12</b>	<b>84.25</b>

How to design optimal  
multi-agent topology?

**Quality**  
**Monetary Cost**  
**Latency**  
**Manual Effort**

How to create  
highly capable agents?

**Reasoning**  
**Planning**  
**Modality**  
**Learning**

How to enable scale,  
safety and human  
agency?

**Parallelization**  
**Resilience**  
**Intent**  
**Teaching**

Discord:

● 19,732 Members

● 2,516 Online



<https://github.com/autogen-ai>



@Chi\_Wang\_



@sonichi

Acknowledgment to Qingyun Wu,  
Markus Buehler, Tamer Abuelsaad &  
Khoi Nguyen for helping develop the  
course material

❤️ AutoGen OSS contributors ❤️

