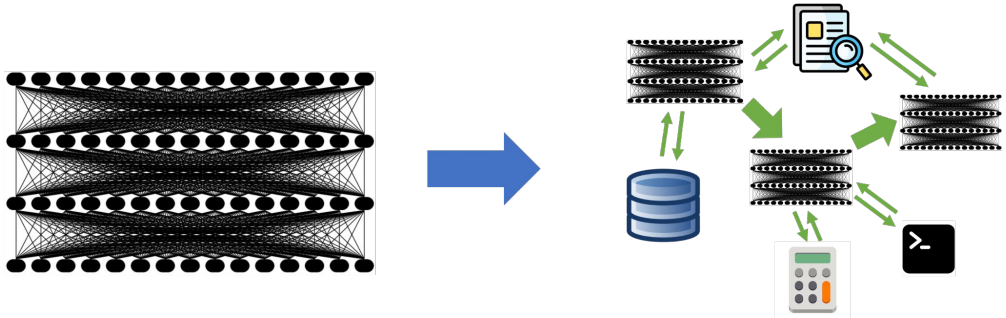
WHAT IS THE NEXT STEP BEYOND AGENTS?

Compound systems

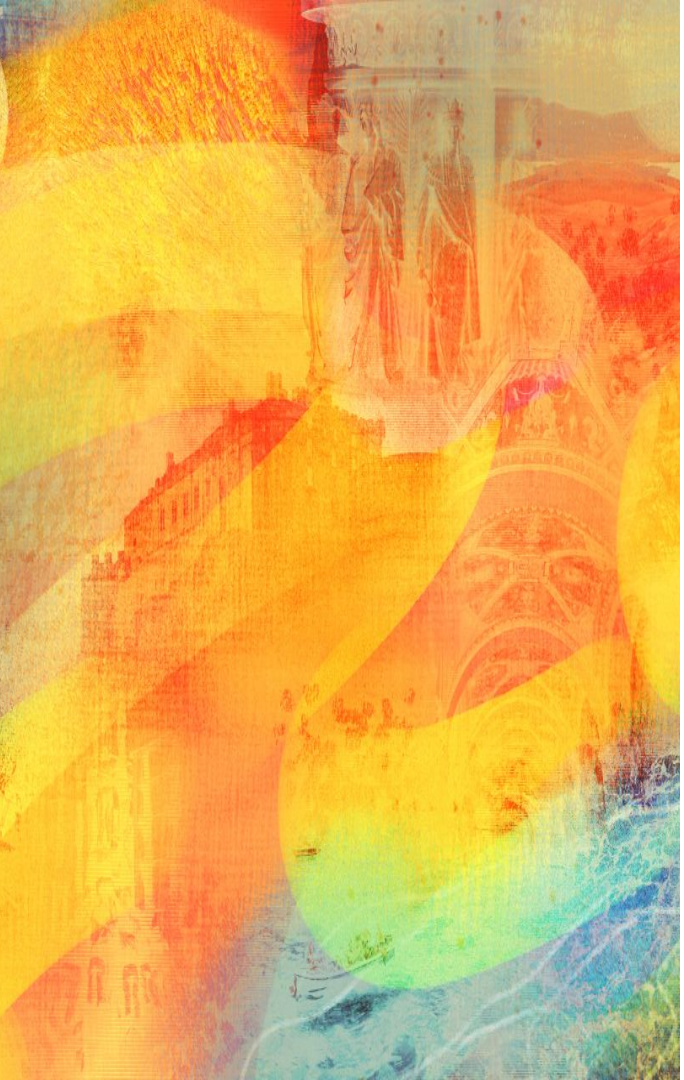


"We define a Compound AI System as a system that tackles AI tasks using multiple interacting components"

https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/

Make them nested!

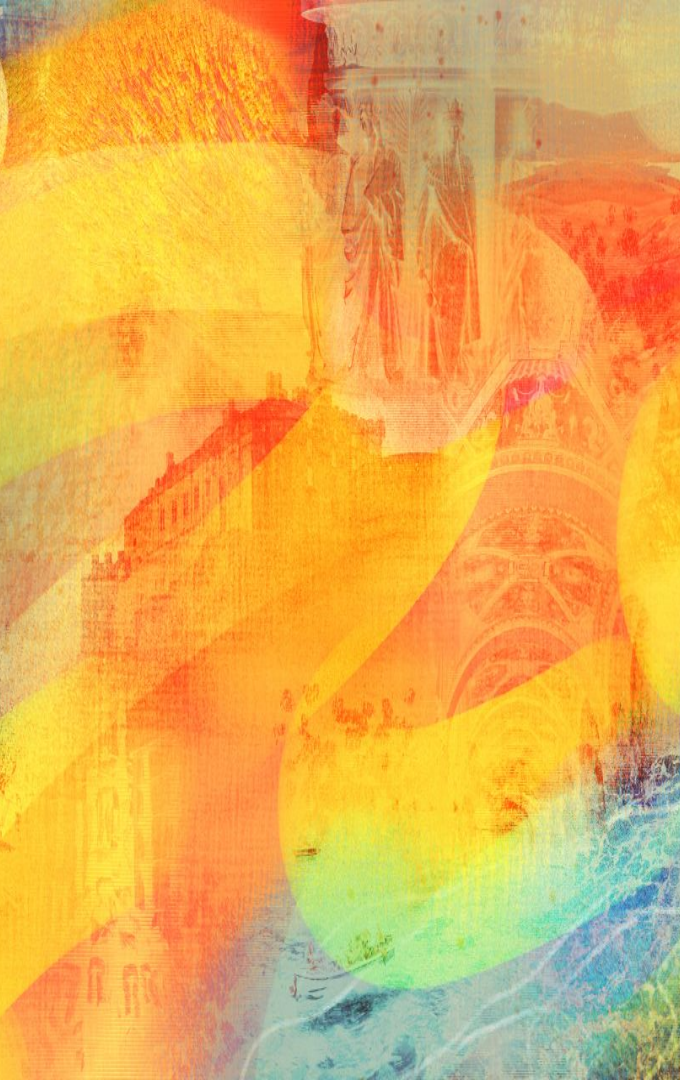
- Event-driven graphs can get large and hard to manage
- To make them easier to manage, and reason about, make them **hierarchical**: let some of the nodes themselves be event flow graphs (with exactly one input and output node)!



Are nested, maybe cyclical event computation flows all you need?

In principle, any instance of business logic (at least, that I can think of) can be expressed as nested event computation flows, with some nodes making LLM calls

So is it useful to reason in terms of agents at all, or is it all just graphs?

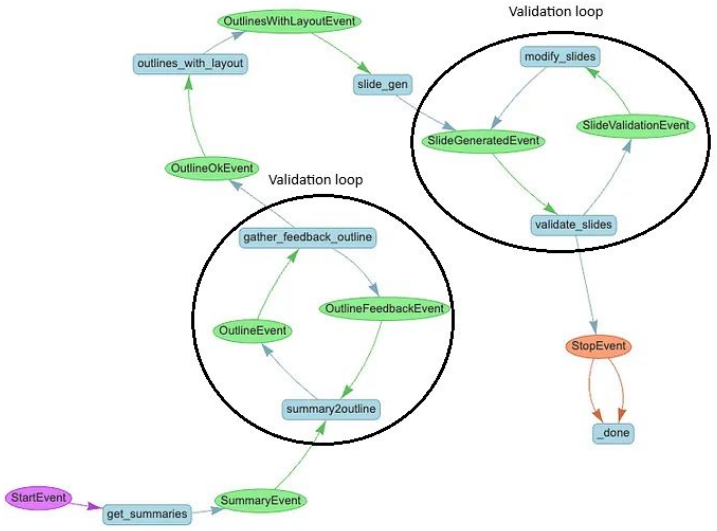


Is the agent concept useful?

I would argue it is!

- Having to express everything as an explicit graph leads to the Turing-complete swamp, where everything is possible and nothing is easy
- Communication: just like with software architecture patterns, calling certain kinds of sub-graphs “agents” makes it easier to reason about, and communicate the intent of one’s logic

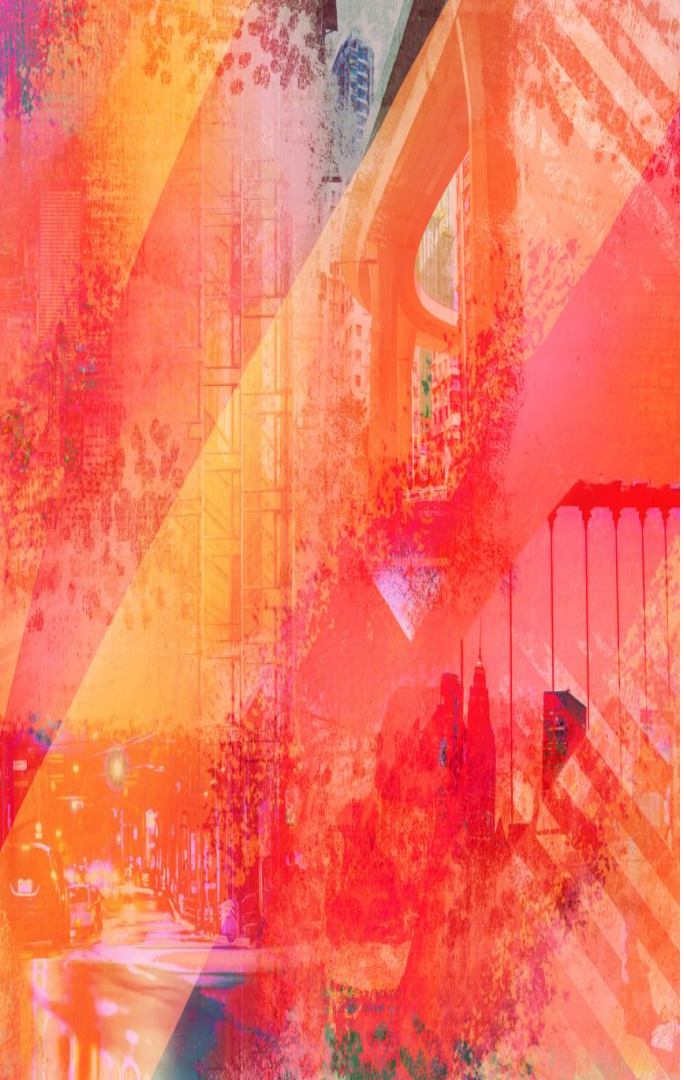
My mental model: agents are just cyclical subgraphs in the event propagation graph!



Do you need other kinds of cycles in your graph?

- I have a hard time imagining a practical business flow that can’t be represented by a combination of
  - An overall DAG flow
  - Tool-calling agents as nodes
  - Forced validation loops as nodes
  - Agents using any nodes as tools
- Do you have a counterexample? If so, I’d love to hear about it!

HOW MUCH OF THIS IS NEW?

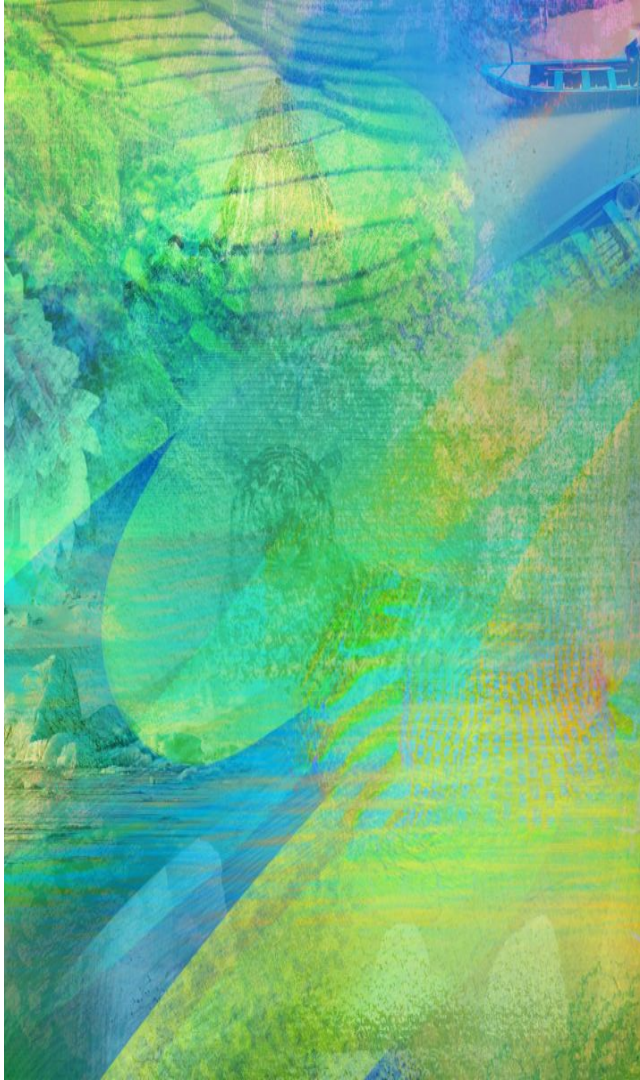


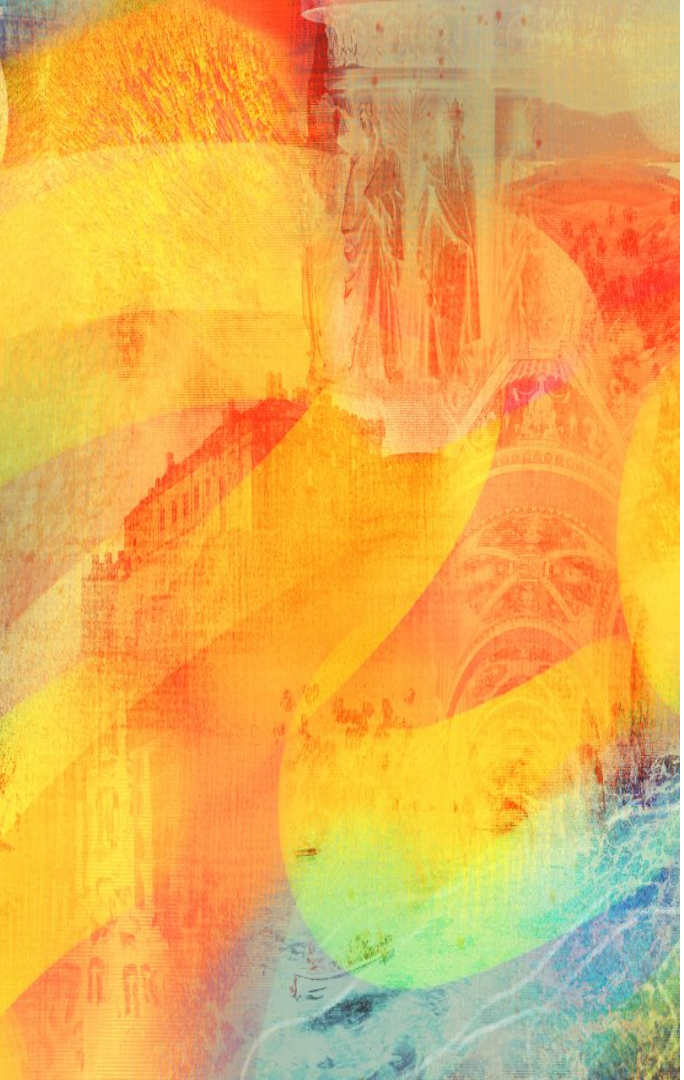
A single agent seen from the outside is just a transform

- Tool usage is function calling
- Return-direct tool calling with validation is new-ish, but it's a local pattern inside the agent, from outside the agent using it is just a transform

State machines

- Have been around forever, eg Order Management Systems in finance
- Often simple enough to code by hand, so I’m unaware of standard frameworks for them



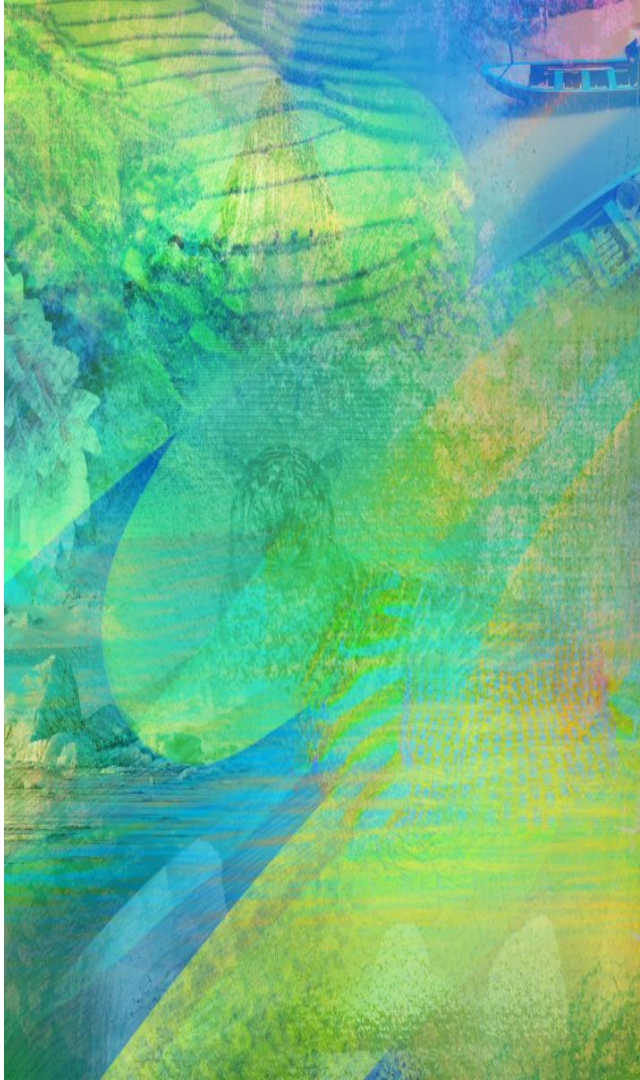


Event-driven calculation

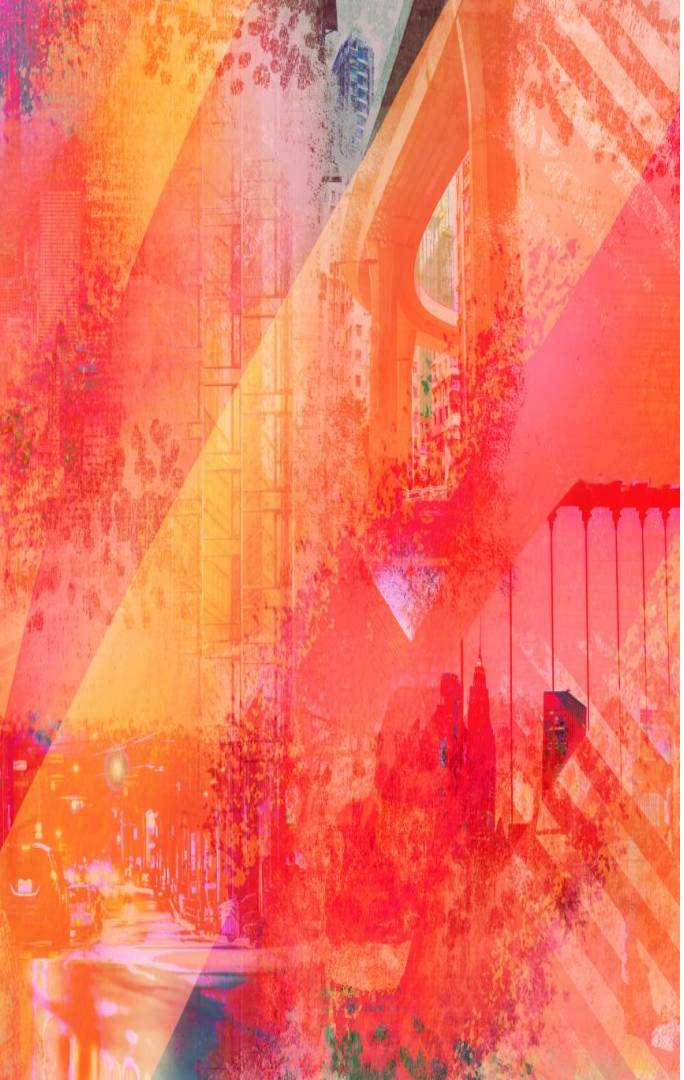
- Another thing that has been there forever
  - Akka, “Actors”
- LlamaIndex Workflows have nearly identical semantics to Faust(link)
- For calculation DAGs, there is Spark, Flink, Airflow, ...

So what is really new?

- Natural language (or code) as inputs
  - Including transforming natural language into structured data
- Natural language as instructions
- Natural language (or code) as output
- A modest degree of resilience/ad hoc error correction, but success not guaranteed



BASIC PATTERNS OF AGENT INTERACTION



Summary: Basic patterns of agent interaction

- Agent using another agent **as a tool**
  - Either tool-calling or forced validation pattern
- Agents **as nodes in a DAG**
  - Transforming input into output and sending it on
  - Often pass a state object down the line, eg whole conversation history
- Agents communicating via a **shared state**
  - In existing repos, mostly a group chat
  - Could be a KV store, knowledge graph, ...
- Nesting: using an event graph with one starting and one final node as a node in another graph