Gödel Agent: A Self-Referential Agent for Recursively Self-Improvement

Xunjian Yin

Duke University

xunjian.yin@duke.edu

https://xunjianyin.github.io/

How Should We Build Agents?

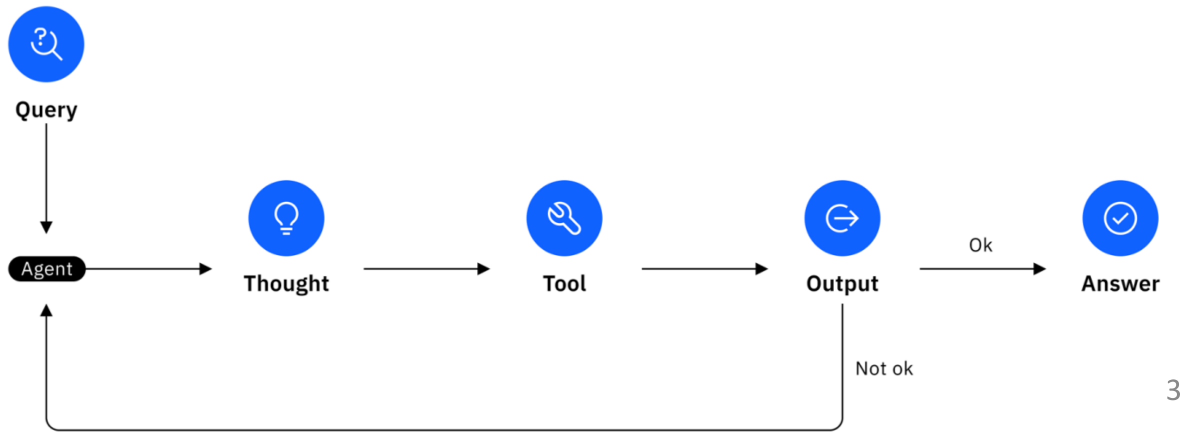
- Do agents require a human-designed workflow?
- Do agents require a human-designed optimization algorithm for that workflow?

- The answer to these questions defines the entire agent paradigm.

Paradigm 1: Hand-Designed Agents

- Answer to Q1: "Yes, agents need a human-designed workflow."
  - Result: Fixed workflows like Chain-of-Thought, ReAct, etc.
  - Analogy: We give the agent a detailed recipe to follow.

  - Problem:
    - Brittle: Highly sensitive to the prompt and the foundation model used.
    - Sub-optimal: How do we know this is the best workflow?

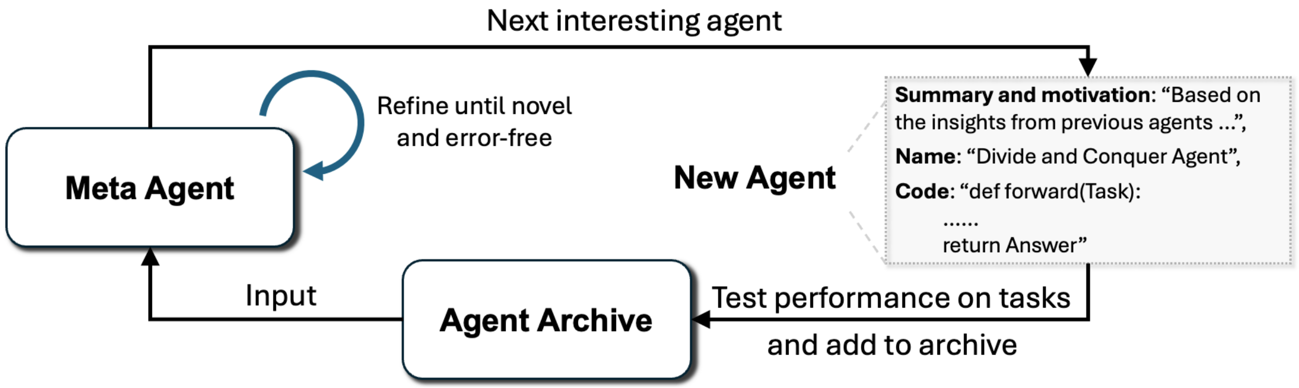


Paradigm 2: Meta-Learning Optimized Agents

- Answer to Q2: "Yes, agents need a human-designed optimization algorithm."
  - The Idea: Instead of designing the workflow, let's design an *algorithm* to search for the best workflow.

Paradigm 2: Meta-Learning Optimized Agents

- Examples:
  - Using Monte Carlo Tree Search (MCTS) to find the best agent framework.
  - Designing a "Meta-Agent" to generate and optimize task-specific agents.
  - Training a "Meta-Model" to optimize an agent's structure.
- This turns agent design into a search or optimization problem.



The "Optimizer's Dilemma" and the Infinite Regress

- The problem remains: The performance is now capped by the human-designed *optimizer*.
  - The search algorithm itself is a product of human priors.
  - This leads to a paradox: If the optimizer is sub-optimal, how can we improve it?
  - Do we need a **meta-meta-optimizer** to optimize the optimizer?
  - And a **meta-meta-meta-optimizer** for that one?

This is an infinite regress.
It's not a sustainable path to general intelligence.

A Better Inspiration: Humans

- How do humans improve?
  - We don't rely on a fixed, external "optimization algorithm."
  - We possess subjective agency.
  - We reflect on feedback from our environment.
  - We modify our own thinking processes and strategies to become better.
- The ideal agent should not be limited by a fixed algorithm. It should be able to improve itself.

Defining "True" Self-Improvement

- Not all "self-improvement" is the same.
- Self-Improvement 1: Module Optimization.
  - The agent uses a *human-designed algorithm* to improve one of its parts (e.g., its memory, or how it creates tools).
- *The core improvement logic is still fixed and human-designed.*

Defining "True" Self-Improvement

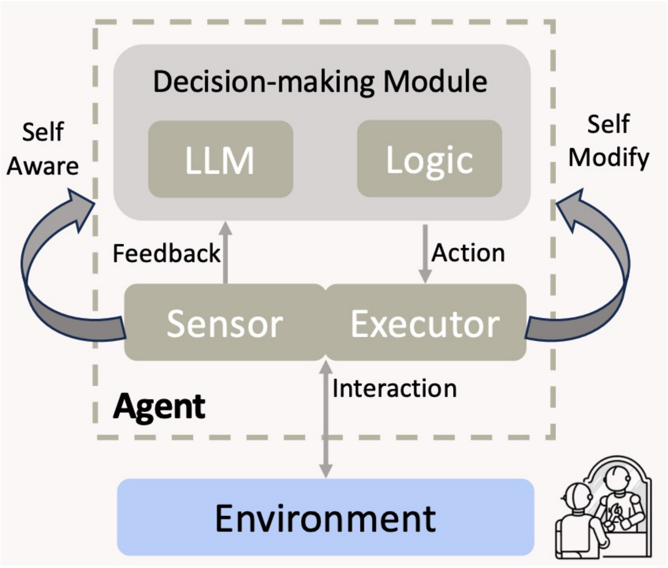
- Not all "self-improvement" is the same.
- Self-Improvement 2: Solution Refinement.
  - The agent improves the *output* or *solution* it generates, not its own internal process. (e.g., AlphaEvolve).

- *The agent itself remains unchanged.*

True Self-Improvement: The agent can modify its *entire* operational logic, including the logic responsible for self-modification.

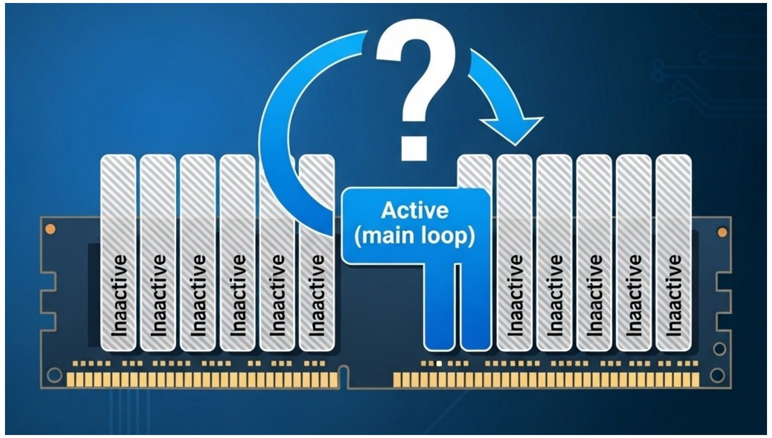
Gödel Agent: The Truly Self-Referential Agent

- We introduce the Gödel Agent, a framework designed for true, recursive self-improvement.
  - Core Principle: The agent has the ability to read and modify its entire own codebase while it is running.
  - Design Philosophy: Minimalism. We intentionally provide very few human priors to isolate and study the effect of self-improvement itself.



The Core Technical Challenge: How to Modify a Running Program?

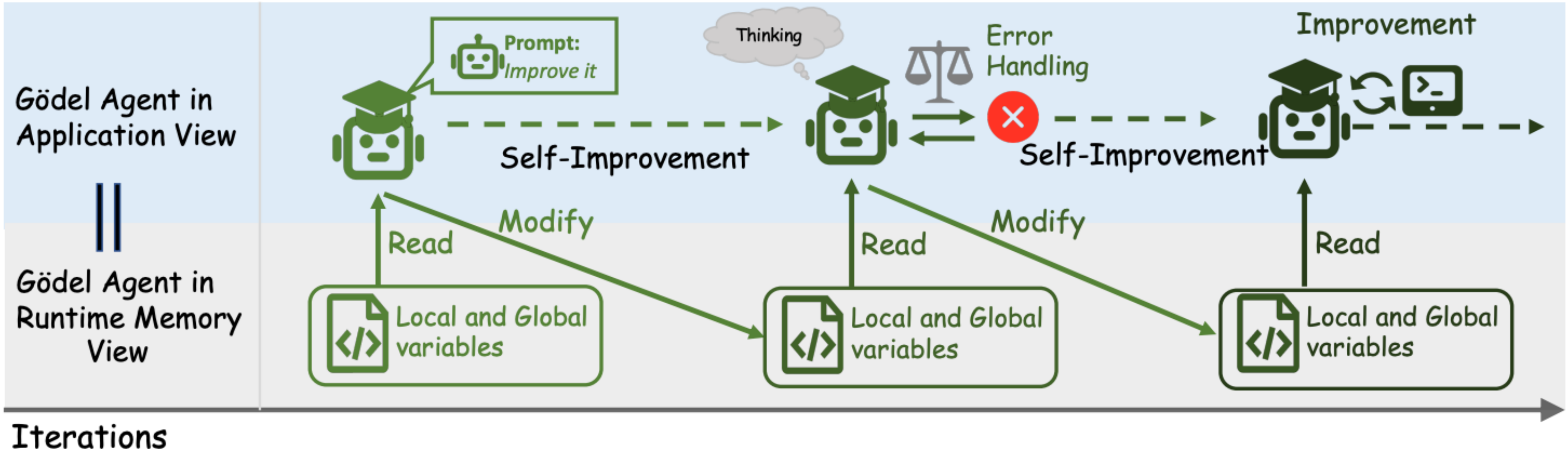
- Once an agent's code is running, it's loaded into memory. How can it modify itself?
- We can separate the code in memory into two types:
  - **Inactive Functions:** Code that is not currently being executed (e.g., tool functions, helper utilities).
  - **The Active Function:** The main function that is currently on the call stack.



The Solution: Monkey Patching and Recursion

- For Inactive/Active Functions: We use Monkey Patching.
  - The agent can dynamically overwrite these functions in memory. This is a standard and straightforward technique.
- For the Active Function: We implement it as a Recursive Function.
  - The agent modifies the code for the main function.
  - This modification is not applied to the *current* call.
  - But when the current step finishes and the function calls itself for the *next* step, the Python runtime loads the new, modified version of the code.

The Solution: Monkey Patching and Recursion



Recursion vs. Loop: Enabling Self-Reference

Traditional Agent (Loop)	Gödel Agent (Recursion)
`while True:`	`def self_improve():`
`  state = get_state()`	`  # 1. Analyze current state`
`  action = policy(state)`	`  # 2. Potentially modify my own code`
`  execute(action)`	`  #    (including this self_improve function)`
`  # Logic is fixed`	`  # 3. Take action`
	`  self_improve() # Next call uses new code`

The Power of Self-Reference: A Virtuous Cycle

- Self-reference creates the potential for a powerful feedback loop and exponential growth.
  - The agent improves its ability to solve tasks.
  - This makes it better at the meta-task of **optimizing itself**.
  - A better self-optimizer can find even better task-solving strategies.
  - This creates a virtuous cycle, or a **"Recursive Self-Improvement"** dynamic.
- This suggests that self-referential agents might have their own scaling law.

Experiment

- Setup: We started with a very simple agent (basic Chain-of-Thought) and a few core tools:
    - read_memory_code()
    - modify_memory_code()
    - execute_code()
    - get_environment_feedback()
    - handle_errors()

Experiment

Agent Name	F1 Score	Accuracy (%)		
	DROP	MGSM	MMLU	GPQA
Hand-Designed Agent Systems				
Chain-of-Thought (Wei et al., 2022)	64.2 ± 0.9	28.0 ± 3.1	65.4 ± 3.3	29.2 ± 3.1
COT-SC (Wang et al., 2023b)	64.4 ± 0.8	28.2 ± 3.1	65.9 ± 3.2	30.5 ± 3.2
Self-Refine (Madaan et al., 2024)	59.2 ± 0.9	27.5 ± 3.1	63.5 ± 3.4	31.6 ± 3.2
LLM Debate (Du et al., 2023)	60.6 ± 0.9	39.0 ± 3.4	65.6 ± 3.3	31.4 ± 3.2
Step-back-Abs (Zheng et al., 2024)	60.4 ± 1.0	31.1 ± 3.2	65.1 ± 3.3	26.9 ± 3.0
Quality-Diversity (Lu et al., 2024)	61.8 ± 0.9	23.8 ± 3.0	65.1 ± 3.3	30.2 ± 3.1
Role Assignment (Xu et al., 2023)	65.8 ± 0.9	30.1 ± 3.2	64.5 ± 3.3	31.1 ± 3.1
Meta-Learning Optimized Agents				
Meta Agent Search (Hu et al., 2024)	$\underline{79.4 \pm 0.8}$	$\underline{53.4 \pm 3.5}$	$\underline{69.6 \pm 3.2}$	$\underline{34.6 \pm 3.2}$
Gödel Agent (Ours)				
Gödel-base (Closed-book; GPT-3.5)	$\mathbf{80.9 \pm 0.8}$	$\mathbf{64.2 \pm 3.4}$	$\mathbf{70.9 \pm 3.1}$	$\mathbf{34.9 \pm 3.3}$
Gödel-free (No constraints)	*90.5 ± 1.8*	*90.6 ± 2.0*	*87.9 ± 2.2*	*55.7 ± 3.1*

Table 1: Results of three paradigms of agents on different tasks. The highest value is highlighted in **bold**, and the second-highest value is <u>underlined</u>. Gödel-base is the constrained version of Gödel Agent, allowing for fair comparisons with other baselines. Gödel-free represents the standard implementation without any constraints, whose results are *italicized*. We report the test accuracy and the 95% bootstrap confidence interval on test sets[3].

Case Study: The Game of 24

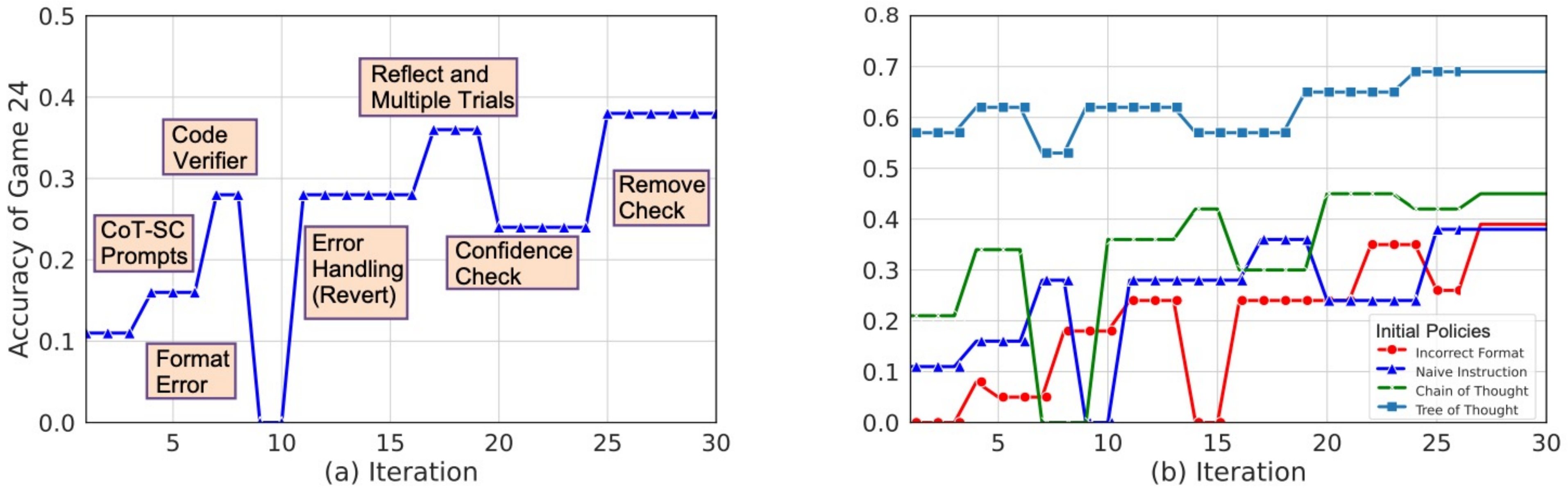


Figure 4: (a) One representative example of Game of 24. (b) Accuracy progression for different initial policies.

Self-correction; Exploration; Innovation

A Future-Proof Architecture

- Hand-designed agent workflows are often **overfitted** to the capabilities of a specific foundation model.
  - A complex prompt that works for GPT-4 might fail on Llama 3 or Claude 3.
  - When new, better models are released, these agents have to be re-engineered.
- Gödel Agent is model-agnostic.
  - Its performance naturally improves as foundation models get better at understanding and writing code.
  - The framework itself doesn't need to change.

The Unbounded Search Space

- Because the Gödel Agent is not constrained by a human-designed search space, it can theoretically...
  - Discover any agent framework that humans have already designed.
  - Discover entirely novel frameworks that humans haven't thought of.
  - Even learn to write its own code to fine-tune its underlying LLM.
- This incredible potential also comes with significant safety considerations. An unconstrained, self-improving agent requires careful oversight.

Future Work

- Seeding with the Best: Instead of starting from a simple policy, can we initialize a Gödel Agent with a state-of-the-art, human-designed framework? Can it improve even further?
- Enhanced Optimization Priors: While we advocate for minimalism, could we give the agent knowledge of optimization techniques (e.g., genetic algorithms, RL) as tools it can choose to use, rather than a fixed process?
- Safety and Controllability: Developing robust sandboxing environments and verification methods to ensure self-modifications remain beneficial and aligned with human intent.
- Multi-Agent…

Take Away

- Let the Agent Learn: Fixed human priors and rules are a bottleneck. Learning and search will always outperform them in the long run.
- Minimize Constraints: To achieve generality, we must impose as few constraints as possible on our agents.
- Self-Reference is Key: Self-reference is the most general and least constrained form of learning. It enables a virtuous cycle of improvement that is essential for the path toward AGI.

Humans are self-referential. Our agents should be too.

Thank you!

xunjian.yin@duke.edu
https://github.com/Arvid-pku/Godel_Agent