

LECTURE 14

Long-horizon research, memory, and self-improvement

Episodic evidence, summaries, skills and mutable beliefs.

Reasoning agents, feedback, and scientific evidence

A summary can change the meaning of an observation

A summary says “calibration checked.” The underlying record says the reference test failed and its certificate had expired.

A later agent may read the short summary as successful validation. Storing more similar summaries can repeat the distortion.

The useful question is which distinctions must survive compression and later correction, rather than simply how much text to remember.

Deriving memory objects from update rules

An observation records an event. A belief interprets it. A summary compresses information, while a skill stores a reusable procedure. They need different update rules.

Changing an interpretation should preserve the original observation. Retracting a faulty reference should trigger reconsideration of dependent claims.

This motivates provenance and support relations alongside retrieval. A memory is useful only when the agent can judge its applicability and current status.

Choosing a memory design

Candidate	Reason to choose it	Cost or limitation
Full history	Exact observations may be needed later	Context and retrieval costs grow
Compressed summaries	A compact record supports repeated tasks	Compression can erase failures or qualifications
Typed records with dependencies	Corrections must reach dependent claims	Requires explicit support and lifecycle rules

Choose memory structure from what must survive reuse and correction. Retrieval similarity alone does not establish validity.

Pedagogical reconstruction

[Packer et al. \(2023\)](#)

[de Kleer \(1986\)](#)

A summary can silently erase negative evidence

Memory object	Stored content	Consequence
Raw observation	Reference test failed at 40 °C; certificate expired	Both the negative result and its limitation survive
Lossy summary	“Calibration checked”	A later agent may infer successful validation
Grounded memory	Test ID, result, temperature and certificate status	Retrieve the observation before reusing the claim
Correction	New reference invalidates the old comparison	Mark dependent claims for reconsideration

Synthetic example. Compression is useful only if it preserves the distinctions needed for later reasoning.

Negative results need context to be reusable

Record which hypothesis was tested, the procedure, statistical sensitivity, and the conditions of the result. A null result from an underpowered measurement does not refute an entire mechanism class.

A useful memory can prevent redundant work while allowing justified replication under changed conditions.

The retrieval policy must distinguish a repeated mistake from a scientifically meaningful repeat, such as an independent replication or a test in a new context.

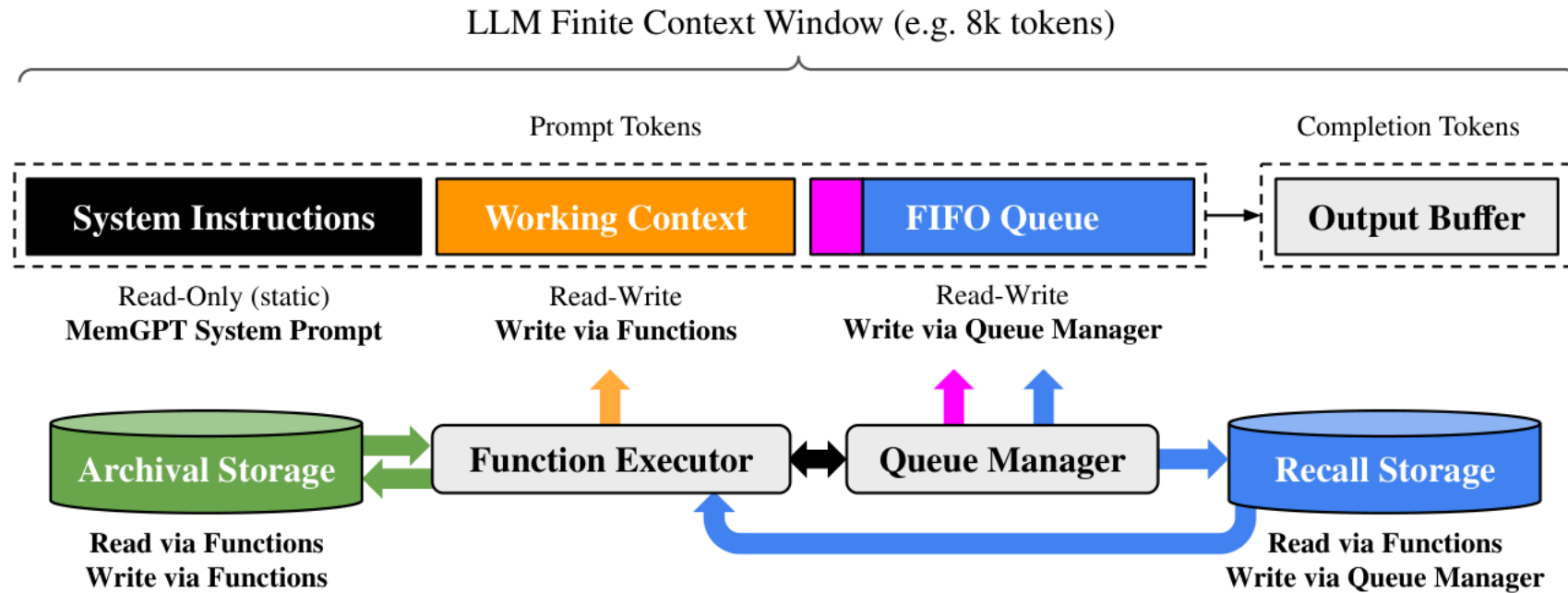
MemGPT: the method

MemGPT manages information across limited active context and larger external memory stores. Explicit memory operations and control-flow events move information between tiers.

The design borrows an operating-system analogy to make context management programmable.

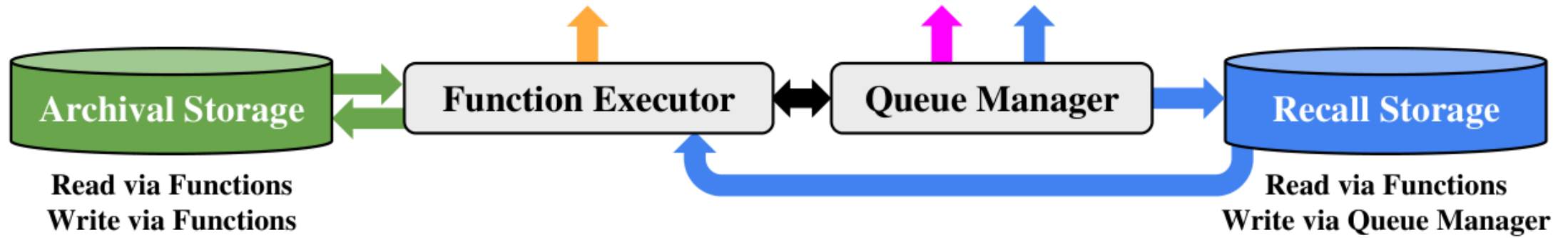
External capacity does not imply that every relevant fact is available at the right time: retrieval, eviction, summarisation, and update policies determine effective memory.

MemGPT: memory tiers and control flow



Information enters active context or moves into archival and recall storage. Some information remains directly accessible to the model; other information requires an additional retrieval operation.

MemGPT: mechanism detail



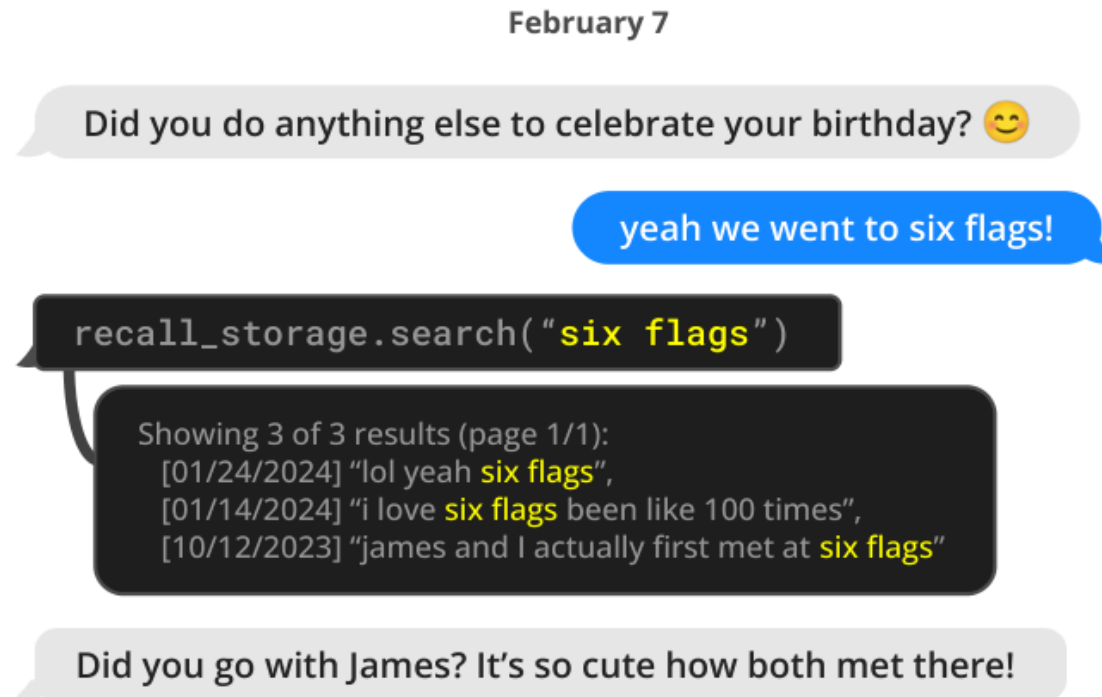
External storage sits outside the active context window. Its read/write interfaces mediate access, so storing a fact and supplying it to the model are different events.

Evidence: retained information is part of the treatment

Model	Accuracy ↑	ROUGE-L (R) ↑
GPT-3.5 Turbo	38.7%	0.394
+ MemGPT	66.9%	0.629
GPT-4	32.1%	0.296
+ MemGPT	92.5%	0.814
GPT-4 Turbo	35.3%	0.359
+ MemGPT	93.4%	0.827

In deep memory retrieval, GPT-4 scores 32.1% and GPT-4 with MemGPT scores 92.5%. The baseline receives a lossy history summary; MemGPT can search the full history. Accuracy uses an LLM judge against gold responses. The contrast measures this complete memory setup, including the information retained, rather than an isolated scheduling mechanism.

MemGPT: interpretation and limits



Retrieval misses, stale summaries, and incomplete updates can cause errors despite abundant storage. Measure whether the required evidence is retrieved and correctly used, not only whether it was stored.

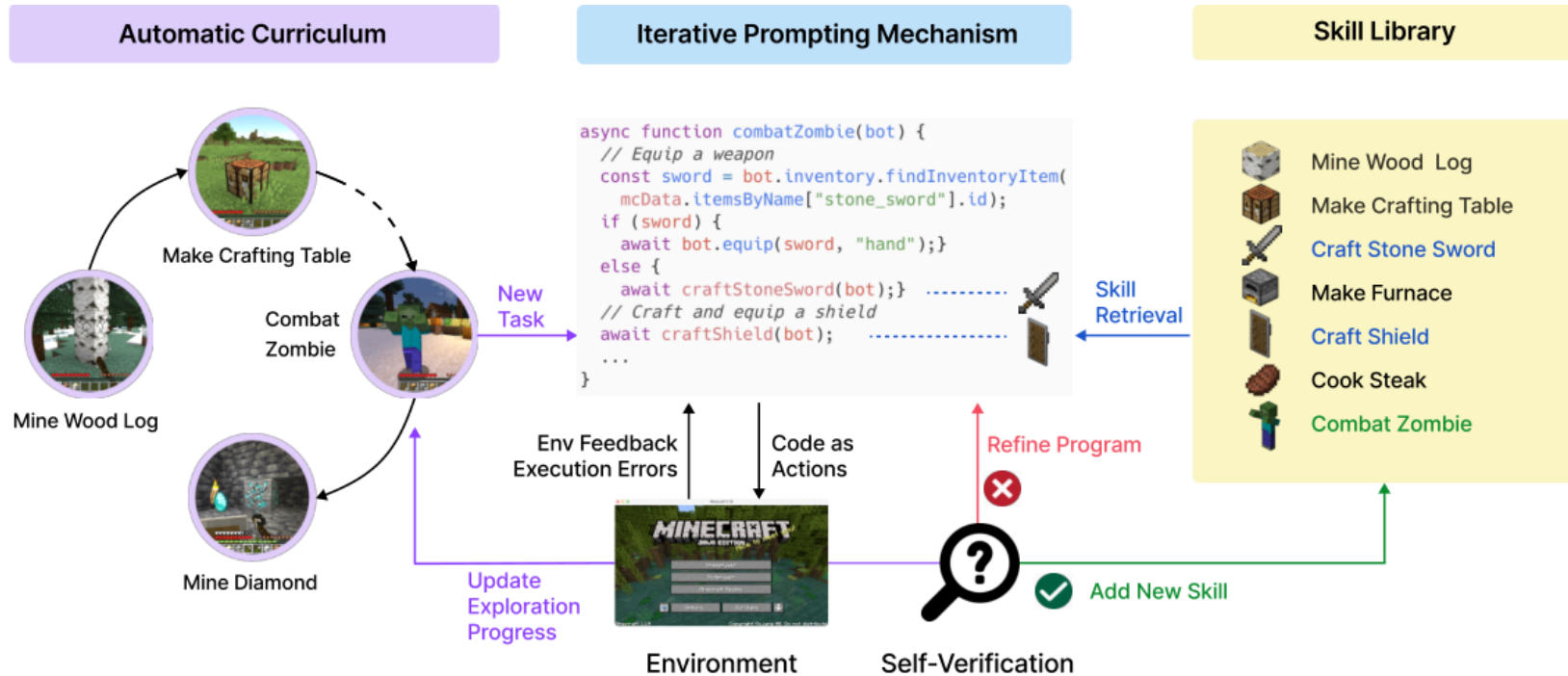
Voyager: the method

Voyager combines an automatic curriculum, a library of executable skills, and iterative code improvement using environment feedback. It operates in Minecraft through a language model and tool interface.

Skills persist outside the model's immediate context and can be retrieved and composed for new tasks.

The curriculum and execution environment supply structure that is absent from a generic conversation transcript.

Voyager: curriculum, feedback and skill reuse



Curriculum, iterative prompting and skill-library loops connect program execution to feedback and reuse. The admission conditions determine which behavior becomes a reusable skill.

Voyager: mechanism detail



Skill entries connect natural-language retrieval keys to executable code and environmental feedback. Reuse depends on compatible preconditions, versions and task assumptions.

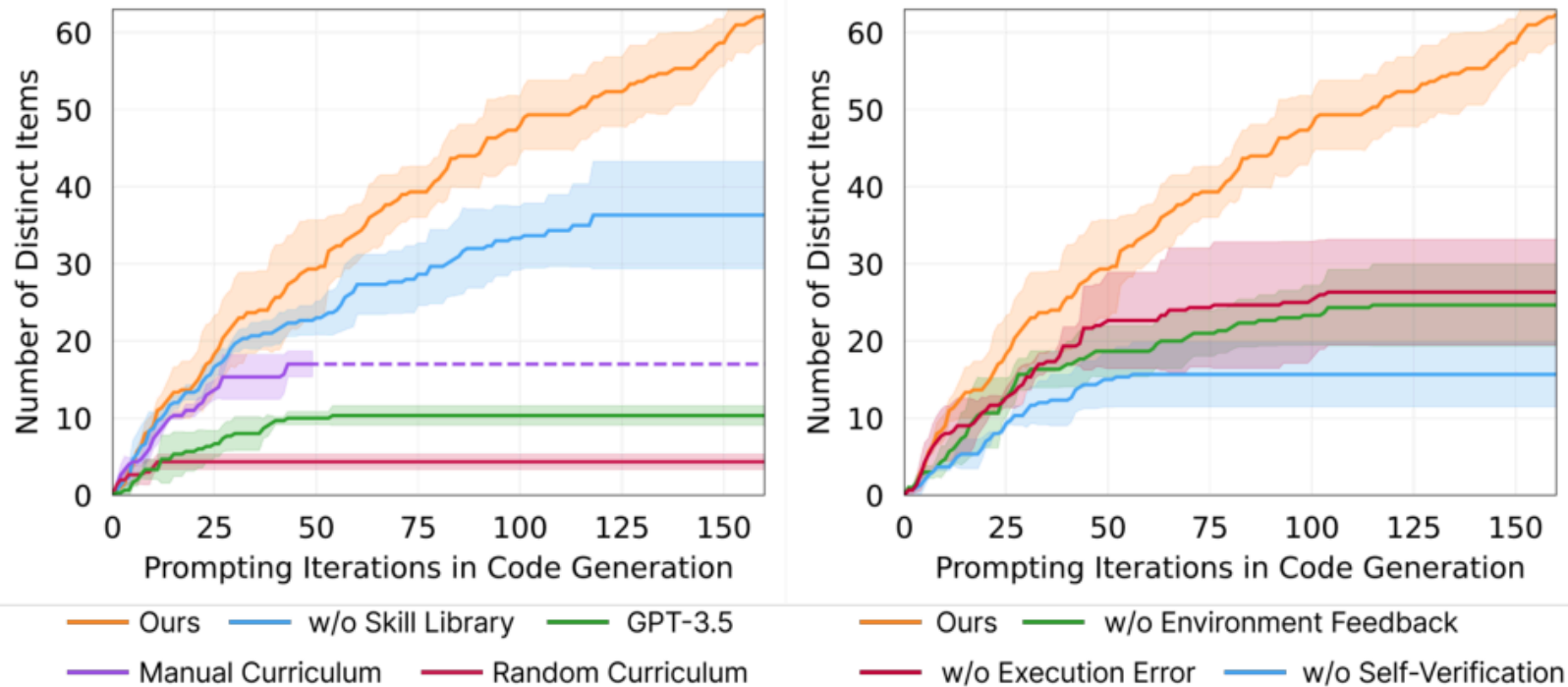
Evaluation scope

The paper measures exploration and task progress in the evaluated Minecraft environments and studies component ablations.

The selected ablation visual compares conditions affecting skill acquisition and reuse.

Persistent code can improve later performance without updating the language model's parameters. Transfer to science depends on whether analogous executable skills and reliable success checks exist.

Voyager: interpretation and limits



A skill that succeeds in one environment can fail under changed preconditions. Reuse requires versioned interfaces and tests for the conditions that determine applicability.

Skills require preconditions and postconditions

A stored procedure can save future effort when its assumptions match the new task. Version its code, dependencies, and validation evidence.

Before reuse, check resource availability, input schema, and domain compatibility. After execution, verify the promised output rather than assuming that a retrieved skill is already correct.

Measure transfer success and repair cost; accumulating more skills is not by itself evidence of improved research capability.

Self-improvement can overfit its evaluator

An agent that repeatedly modifies prompts, code, or memory against one benchmark can adapt to the evaluator's quirks.

Separate development feedback from a locked final evaluation. Preserve all attempted variants, selection decisions, and resource use.

Test transfer to new tasks, contexts, and executors. Improvements on the optimisation benchmark support a narrower claim than general autonomous scientific improvement.

Memory invalidation is scientific maintenance

When an observation or source is corrected, identify dependent interpretations, summaries, and skills. Mark them stale and recompute or review them as appropriate.

Do not treat every descendant as false: some may remain valid through independent evidence. Maintain support relationships at the claim level.

This is a dependency-management problem with epistemic consequences; the system must represent why it believes a claim to know what a correction changes.

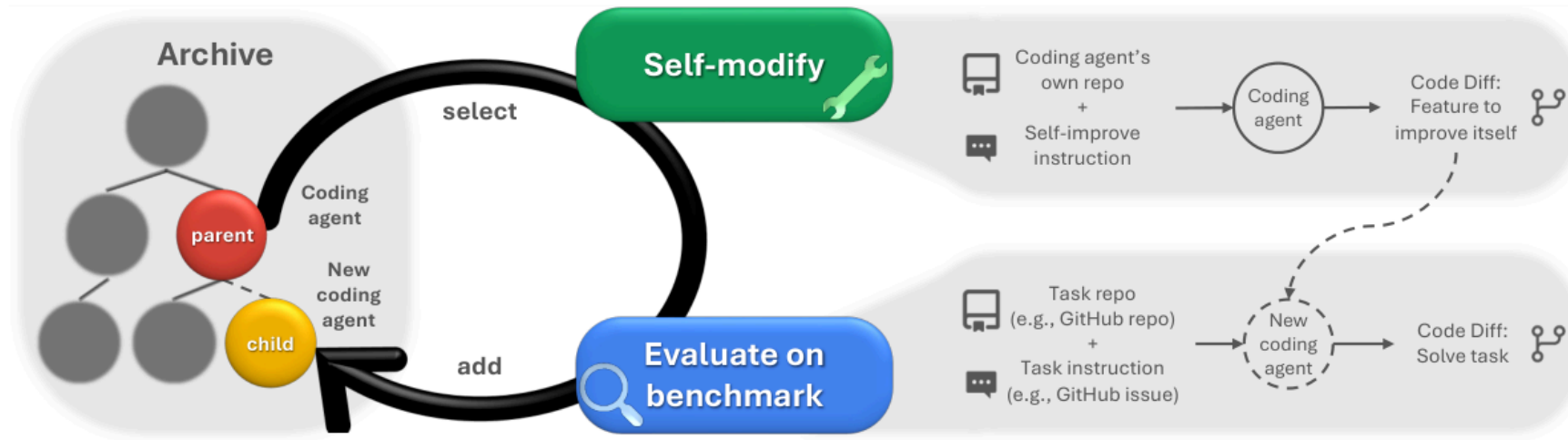
Darwin Gödel Machine: the method

The Darwin Gödel Machine maintains an archive of coding agents, proposes modifications to their code, and evaluates descendants on coding tasks.

Selection and continued exploration can preserve stepping stones that are useful for later improvements.

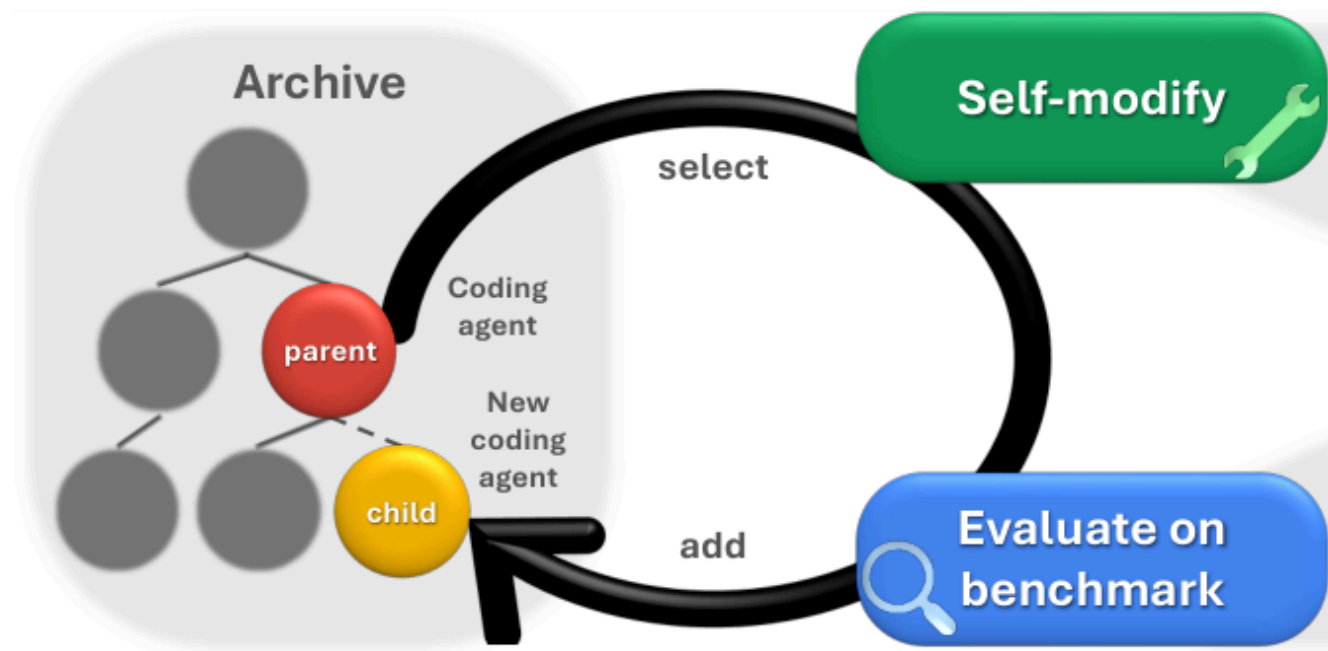
Despite the name, changes are validated empirically; the system does not provide a general proof that each self-modification is beneficial.

Darwin Gödel Machine: archive and evolution loop



The archive retains executable agent implementations and measured outcomes. Parent selection, code modification, evaluation and admission determine how implementations change across iterations.

Darwin Gödel Machine: mechanism detail



Archive admission uses an empirical criterion to retain a descendant. Passing that criterion is narrower evidence than universally beneficial self-modification.

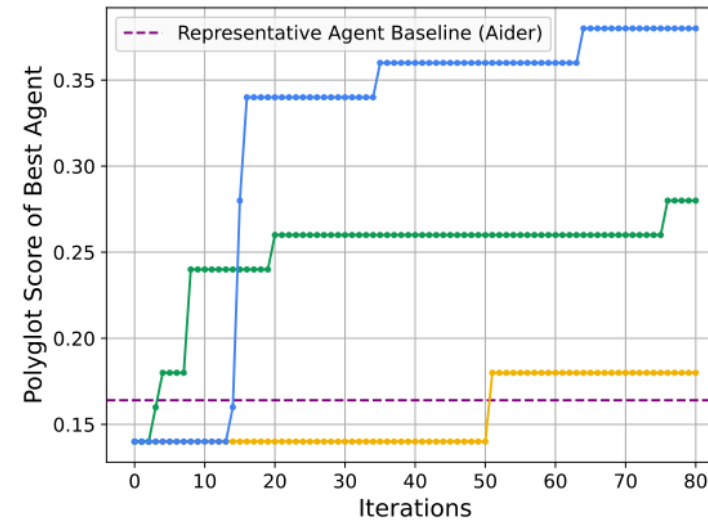
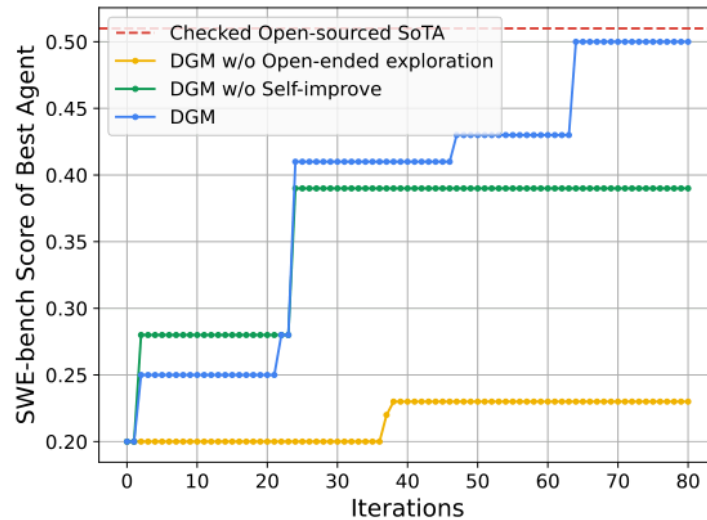
Evaluation scope

The source reports coding-benchmark improvements and comparisons with restricted exploration or self-improvement variants.

The selected results should be read with their task splits, base model, evaluation budget, and archive procedure.

Improvement in the selected coding tasks is evidence for that search process under its conditions, while general scientific self-improvement remains a further transfer claim.

Darwin Gödel Machine: interpretation and limits



Repeated evaluation can overfit the development benchmark. Archive quality depends on trustworthy scoring, held-out tasks, reproducible environments, and preservation of failed as well as successful attempts.

ReasoningBank learns through persistent records

ReasoningBank distills reusable lessons from self-judged successful and failed experiences and retrieves memories for later tasks. It also studies additional experience generation through test-time scaling.

Its adaptation mechanism includes changing external memory. Distinguish that update from changing model weights or discovering a validated scientific fact.

Memory policies answer different questions

Policy	Can test tasks update memory?	What is measured?
Frozen bank	No	Retrieval from fixed experience
Sequential adaptation	Yes, for later tasks	Learning across a stream
Per-task reset	Only within a task	Within-task adaptation

Report the initial bank, reset rules, task order and permitted feedback. Scores from different policies are not automatically comparable.

Memory has a lifecycle, not just a retrieval score

Operation	Scientific memory example	Decision rule
Write	Observation plus units, tool version and status	Keep evidence distinct from interpretation
Retrieve	Find comparable calibration conditions	Match applicability, not only word similarity
Compress	Retain failed tests and unresolved assumptions	Do not turn absence of support into success
Correct or retire	Invalidate a result using a faulty reference	Revisit claims that depended on it

The lifecycle links storage engineering to scientific reliability. Persistent text is not automatically persistent knowledge.

Shared memory changes performance on later tasks

Protocol	First task	Second task	Total correct
Reset before every task	Fail; learn rule locally	Fail; learn rule locally	0/2
A then B, shared bank	Fail; store reusable rule	Succeed using rule	1/2
B then A, shared bank	Fail; store reusable rule	Succeed using rule	1/2

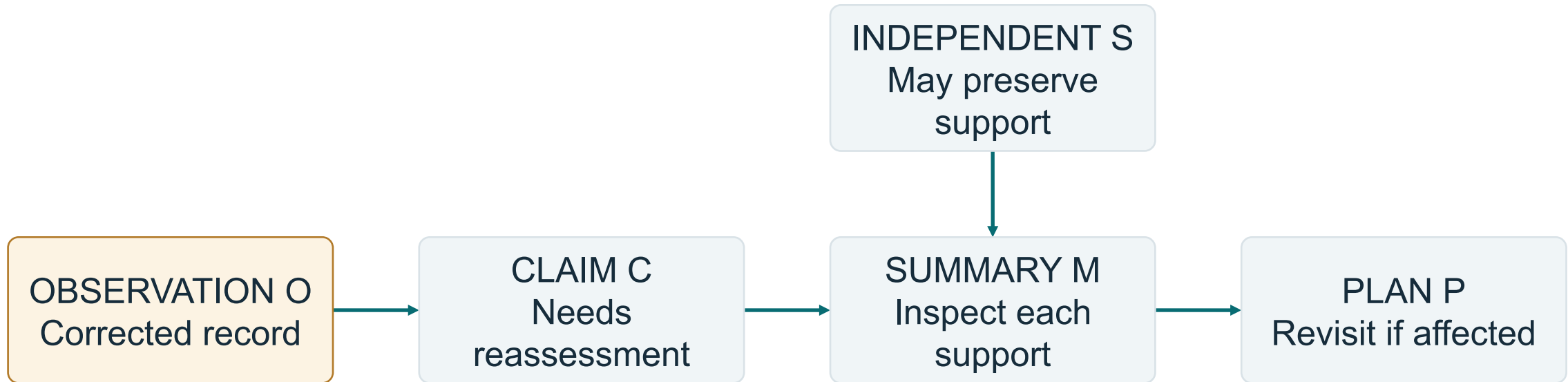
Synthetic two-task example assumes either task teaches the same reusable rule after failure. Aggregate success conceals where the information came from.

Grounded support under positive rules

Operation	Calculation or state	Consequence
Rules	a from r or b; b from a	A cycle is possible
Root r is valid	Seed {r}; add a; then b	Supported set {r,a,b}
Root r is retracted	Seed empty; no rule can start	Supported set empty
Unsafe shortcut	Keep a because b, b because a	Circular support survives

Synthetic positive-rule truth maintenance. Recomputing from valid roots prevents an unsupported cycle from maintaining itself.

Worked memory graph: one corrected observation



Propagate a need for review through the affected dependencies. The independent source may preserve some of M's claims, so staleness does not mean falsity. Revisit P if its decision relies on the corrected support. Preserve O's history and the reason for the correction rather than silently deleting it.

The same arrows can encode different support rules

Meaning of the support	Rule for M	C = 0, S = 1
Either source is sufficient	C OR S	M supported: 1
Both sources are required	C AND S	M unsupported: 0

Here 1 means a valid recorded justification, and 0 means no such justification. Neither is a probability or a proof about reality. The graph alone does not specify AND versus OR; record the rule before recomputing M and the plan recommendation P.

Trace the queue before recomputing support

Remove head	Unseen children	Queue afterward
O	C	[C]
C	M	[M]
M	P	[P]
P	None	[]

Start with queue [O] and seen {O}. Remove its head; add each unseen child to seen and the queue. Mark C, M, and P stale. S is unaffected. A back edge $P \rightarrow M$ adds no new item because M is already seen. This terminates even with cycles.

Recursive Language Models inspect external context

Stage	What the model sees or does	Why it matters
Initial state	Prompt data lives in an external environment	The full input need not occupy the model context
Inspection	Code selects a relevant region or statistic	Selection itself can omit decisive evidence
Recursive call	A sub-call analyzes a selected part	Local conclusions must retain source locations
Aggregation	Combine returned results and inspect gaps	Cross-part dependencies may require another pass

Recursive Language Models (RLMs), December 2025 preprint. A larger accessible input is not the same as perfect recall or understanding.

Alternative and joint justifications

$$\text{support}(v) = \bigvee_{J \in \mathcal{J}(v)} \bigwedge_{u \in J} \text{support}(u)$$

Let $\mathcal{J}(v)$ list the premise sets that justify v . Require every premise within one set; any complete set suffices. Reviewed evidence supplies root labels.

For M , alternatives $\{C\}$ and $\{S\}$ preserve support when C loses its justification and S remains valid. A single set $\{C, S\}$ requires both.

This positive-rule model excludes confidence, contradictions, and default negation. A full assumption-based truth maintenance system has richer semantics.

Traversal cost and support recomputation

With adjacency lists and a seen set, stale traversal costs $O(V_r + E_r)$ in reachable vertices and their outgoing edges. Each node enters the queue once; independent source S need not enter it.

For an acyclic support graph, recompute after all premises in topological order. Cost also includes reading every relevant justification premise, including unaffected support.

For positive cycles, begin affected derived labels at zero and propagate grounded support to a fixed point. Reusing old labels can retain self-support. Arbitrary truth maintenance is not covered by the traversal bound.

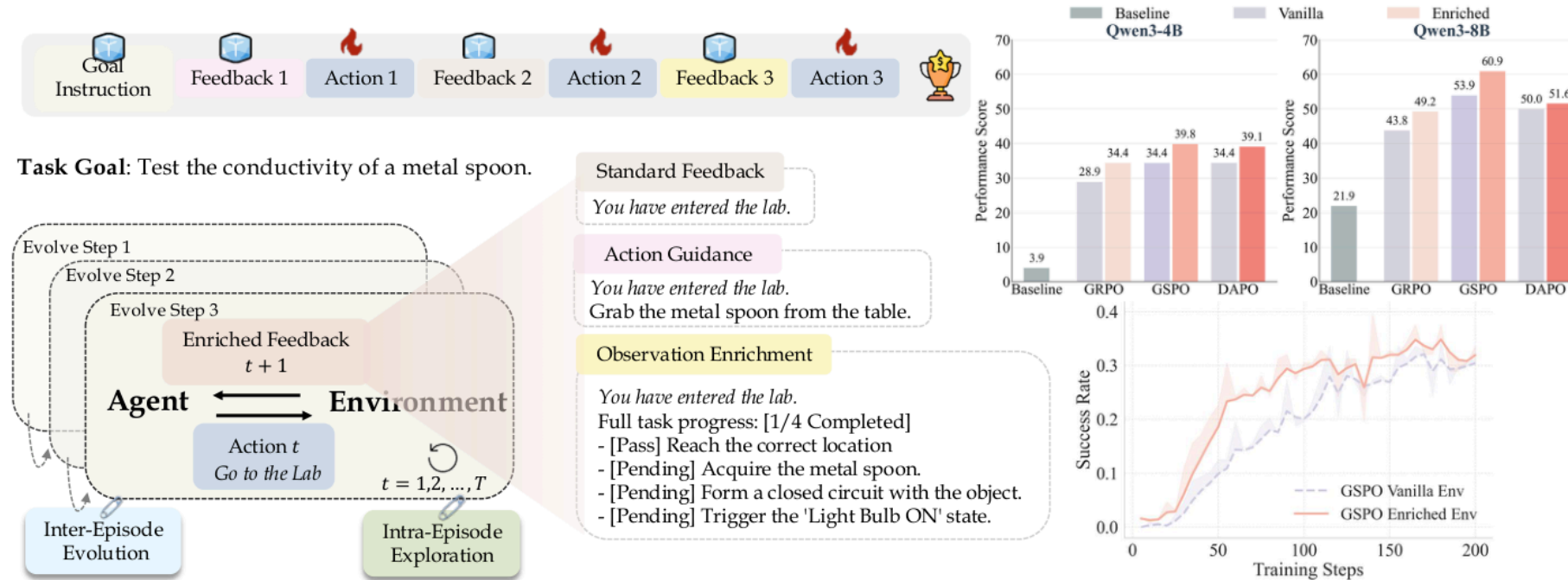
Recent research: Feedback-enriched environments

This September 2026 preprint studies training environments that provide additional guidance or observations to help agents learn long-horizon tasks.

Its design changes the feedback supplied during learning, with attention to when action guidance gives way to richer observations.

The literature snapshot is dated 11 September 2026. Findings remain conditional on the source version and evaluation conditions. Recent preprints contribute reported results with distinct validation requirements.

Feedback scaffolds: training and evaluation conditions



The labeled conditions distinguish environment-provided feedback from information acquired independently by the agent. Generalization depends on whether the evaluation retains the training scaffold.

Recent methods in context

Recursive Language Models (preprint; revised 2026). Treats long context as an external environment that can be inspected programmatically and processed through recursive model calls. Nested calls can increase cost unpredictably; long-context benchmark gains do not establish universal task efficiency.

ReasoningBank (ICLR 2026). ReasoningBank distills reusable strategies from successful and failed experience. Its self-judged lessons support later tasks, while erroneous lessons and changing task conditions motivate explicit revision.

GEPA (ICLR 2026). GEPA adapts prompts using reflected execution feedback and a population of candidates. It changes instructions rather than model weights, and its development feedback remains separate from final evaluation.

References (1/2)

Charles Packer, Sarah Wooders, Kevin Lin et al. (2023). MemGPT: Towards LLMs as Operating Systems

Guanzhi Wang, Yuqi Xie, Yunfan Jiang et al. (2023). Voyager: An Open-Ended Embodied Agent with Large Language Models

Jenny Zhang, Shengran Hu, Cong Lu et al. (2025). Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents

Ludovico Mitchener, Angela Yiu, Benjamin Chang et al. (2025). Kosmos: An AI Scientist for Autonomous Discovery

Johan de Kleer (1986). An assumption-based TMS

References (2/2)

[Hongbang Yuan, Zhuoran Jin, Yixin Cao \(2026\). Environments as Scaffold: Enriching Feedback to Bootstrap Self-Evolving Agents in Long-Horizon Tasks](#)

[Siru Ouyang, Jun Yan, I-Hung Hsu et al. \(2025\). ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory](#)

[Alex L. Zhang, Tim Kraska, Omar Khattab \(2025\). Recursive Language Models](#)

[Lakshya A Agrawal, Shangyin Tan, Dilara Soylu et al. \(2025\). GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning](#)