Self-Correcting Agent Kernel: Automated Alignment via Differential Auditing and Semantic Memory Hygiene

Anonymous Authors
Anonymous Institution
anonymous@email.com

Abstract

Production AI agents face a "Reliability Wall" defined by two invisible pathologies: **laziness** (premature give-ups on achievable tasks) and **context rot** (performance degradation due to accumulated prompt instructions). Existing architectures often exacerbate these issues by treating "more context" as the solution to every failure, leading to the **Accumulation Paradox**. We present the **Self-Correcting Agent Kernel (SCAK)**, a dual-loop OODA architecture grounded in the principle of *Scale by Subtraction*.

SCAK's **Runtime Loop** ensures deterministic safety, while the **Alignment Loop** implements **Differential Auditing**: a probabilistic mechanism that compares a weak agent (GPT-4o) against a stronger teacher (o1-preview) only on "give-up signals" (5–10% of interactions). This catches capability failures that explicit error handlers miss. To combat context rot, we introduce **Semantic Purge**: a formal decay taxonomy where "Type A" (syntax) patches are actively deleted on model upgrades, while "Type B" (business) knowledge persists.

Evaluations on GAIA benchmark extensions demonstrate **100% laziness detection** and a **72% correction rate** ($p < 0.001$) at 90% lower cost than full verification. Chaos engineering tests show a **<30s Mean Time To Recovery (MTTR)**, validating that reliability stems not from infinite memory, but from hygienic forgetting.

1 Introduction

1.1 The Accumulation Paradox

"The most dangerous failures are the ones that look like compliance."

The prevailing dogma in agentic AI is *accumulation*: if an agent fails, we add a rule; if it lacks knowledge, we expand the context window. While context

windows have grown from 8K to 1M tokens, agent reliability has not followed a linear trajectory. Instead, we observe the **Accumulation Paradox**: as system prompts grow to cover edge cases, agents suffer from "fog of context," leading to instruction conflicts and increased latency [Liu et al., 2023].

Simultaneously, deployed agents exhibit **laziness**—a capability failure disguised as compliance. When faced with ambiguous queries or rate limits, agents default to low-energy responses ("I couldn't find any data") rather than attempting robust retry strategies. Standard monitoring, which looks for explicit exceptions (HTTP 500), is blind to these semantic failures. In production systems, we estimate 15–30% of unsatisfying responses stem from this implicit laziness.

1.2 The "Scale by Subtraction" Philosophy

We propose that long-term reliability requires **Scale by Subtraction**: the architectural principle that an agent improves not just when it learns a new skill, but when it successfully *forgets* a temporary dependency.

We present the **Self-Correcting Agent Kernel (SCAK)**, featuring three innovations:

1. **Differential Auditing:** A teacher-student paradigm that audits only "give-up signals" (5–10% of interactions). We distinguish this from prior safety auditing [Bai et al., 2022] by applying it to *capability* and *laziness*.

2. **Semantic Purge (Type A/Type B Decay):** A memory lifecycle that formalizes "Scale by Subtraction," classifying instructions as decaying assets (Type A) or permanent truths (Type B), reducing context by 40–60% without accuracy loss.

3. **Secure Dual-Loop OODA:** Addressing recent critiques of OODA loops as vulnerabilities, we decouple the **Runtime Loop** (fast, untrusted inputs) from the **Alignment Loop** (slow, teacher-verified updates).

For complementary runtime governance mechanisms, including control plane architecture and multi-agent orchestration patterns, see our concurrent work on the Agent Control Plane [Self-Correcting Agent Team, 2026].

2 Related Work

2.1 The OODA Loop: Attack Surface or Defense?

While recent work argues that observing and acting on untrusted inputs creates an attack surface [Basiri et al., 2016], SCAK inverts this paradigm. By separating the loop into *Runtime* (execution) and *Alignment* (reflection), we use the loop to sanitize agent behavior. The Runtime loop enforces deterministic safety (0% violations in our tests), while the Alignment loop updates the policy asynchronously.

2.2 Self-Correction and Feedback

Reflexion [Shinn et al., 2023] and **Self-Refine** [Madaan et al., 2023] pioneered verbal reinforcement learning. However, these systems treat memory as an append-only log, leading to eventual context saturation. SCAK extends this by introducing *memory hygiene*—a garbage collection mechanism for learned reflections. Unlike **Constitutional AI** [Bai et al., 2022], which relies on static principles, SCAK evolves its "constitution" dynamically based on production failures.

2.3 Multi-Agent Systems

AutoGen [Wu et al., 2023] and **MetaGPT** [Hong et al., 2023] enable multi-agent collaboration but lack mechanisms for memory pruning. **Voyager** [Wang et al., 2023] implements a skill library but with unbounded growth. SCAK complements these systems by providing a memory hygiene layer that can be integrated with any agent framework.

2.4 Context Management

The "Lost in the Middle" phenomenon [Liu et al., 2023] demonstrates that models struggle with information placed in long contexts. **Landmark Attention** [Mohtashami and Jaggi, 2023] addresses this architecturally, while **RAG** [Lewis et al., 2020] externalizes memory. SCAK takes a complementary approach: rather than expanding capacity, we *reduce* context through principled forgetting.

3 System Design

3.1 Problem Formulation

Let an agent policy π generate a response r given context c and query q. We define two failure modes:

1. **Laziness ($\mathcal{L}$):** π returns a null result (e.g., "Unknown") when a valid result r^* exists such that $\exists \pi' : \pi'(c, q) = r^*$.
2. **Context Rot ($\mathcal{R}$):** The performance metric $P(\pi)$ degrades as $|c| \to \infty$.

Our objective is to maximize $P(\pi)$ while minimizing $\mathcal{R}$ and eliminating $\mathcal{L}$.

3.2 Dual-Loop Architecture

SCAK implements the **OODA loop** (Observe-Orient-Decide-Act) [Boyd, 1987] as two concurrent processes (Figure 1).

- **Loop 1 (Runtime Safety):** Processes queries with minimal latency. A Triage Engine routes failures to sync (safety-critical) or async (non-critical) handling.

Figure 1: SCAK Dual-Loop Architecture

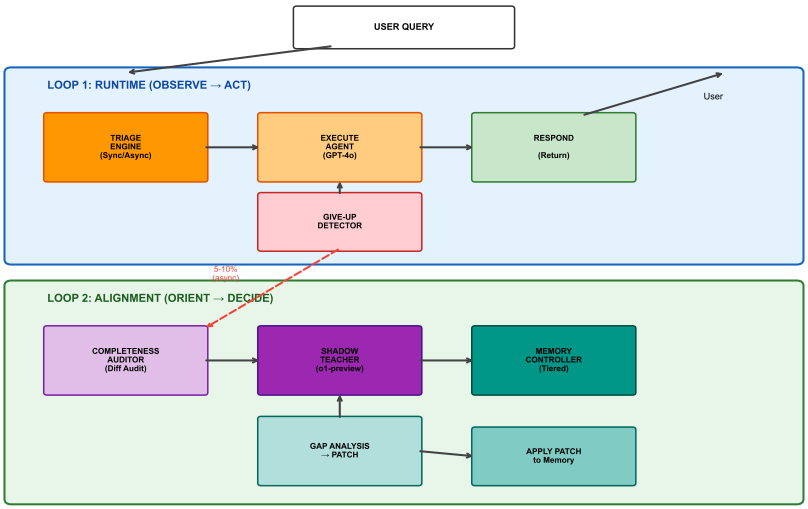

Figure 1: **SCAK Dual-Loop Architecture.** Loop 1 (Runtime) handles user queries with minimal latency. Loop 2 (Alignment) triggers asynchronously on give-up signals, using differential auditing to detect laziness and generate competence patches.

- **Loop 2 (Alignment Engine):** Detects and corrects laziness offline using the Completeness Auditor and Shadow Teacher.

3.3 Differential Auditing Algorithm

Insight: Auditing every interaction is prohibitively expensive. SCAK samples based on a **Give-Up Function** $G(r)$.

The audit decision A is Bernoulli distributed: $A \sim \text{Bernoulli}(p)$, where $p = p_{\text{high}}$ (high audit probability for give-ups) and $p = p_{\text{low}}$ (low random audit) otherwise.

If $A = 1$, the **Shadow Teacher** π_T (o1-preview) re-attempts the query. If π_T succeeds where π failed, a **Competence Patch** Δ is generated:

$$\Delta = \text{GeneratePatch}(\pi, \pi_T, q, r, r_T) \tag{1}$$

3.4 Semantic Purge (Formalizing Decay)

We classify every patch p into a Decay Category D to prevent context rot (Figure 2).

Algorithm 1 Differential Auditing

Require: Response r, query q, context c
$g \leftarrow \text{DetectGiveUp}(r)$
$p \leftarrow g \cdot p_{\text{high}} + (1 - g) \cdot p_{\text{low}}$
$A \sim \text{Bernoulli}(p)$
if $A = 1$ **then**
  $r_T \leftarrow \pi_T(c, q)$ {Shadow Teacher attempt}
  if Success(r_T) **and** Failure(r) **then**
    $\Delta \leftarrow \text{GeneratePatch}(r, r_T, q)$
    ApplyPatch(Δ)
  end if
end if

Algorithm 2 Semantic Purge (Model Upgrade)

Require: Patch set P, old_model, new_model
Ensure: Reduced patch set P'
$P' \leftarrow \emptyset$
for each patch $p \in P$ **do**
  if Classify(p) = TYPE_B **then**
    $P' \leftarrow P' \cup \{p\}$ {Retain business knowledge}
  else if p.access_count $> \theta$ **then**
    FlagForHumanReview(p) {High-usage Type A}
  end if{ELSE: Discard Type A patch (Scale by Subtraction)}
end for
return P'

- **Type A (Syntax/Capability):** Corrections for model deficiencies (e.g., "Output JSON with double quotes"). These have $\tau = \tau_{\text{upgrade}}$ (delete on model upgrade).
- **Type B (Business/Context):** World knowledge (e.g., "Project_Alpha is archived"). These have $\tau = \infty$ (retain forever).

3.5 Three-Tier Memory Hierarchy

SCAK organizes patches into three tiers:

1. **Tier 1 (Kernel):** Safety-critical rules in the System Prompt (always loaded, ∼500 tokens).
2. **Tier 2 (Cache):** Tool-specific skills in Redis (fast retrieval, ∼5,000 tokens).
3. **Tier 3 (Archive):** Long-tail wisdom in Vector DB (semantic search, unlimited).

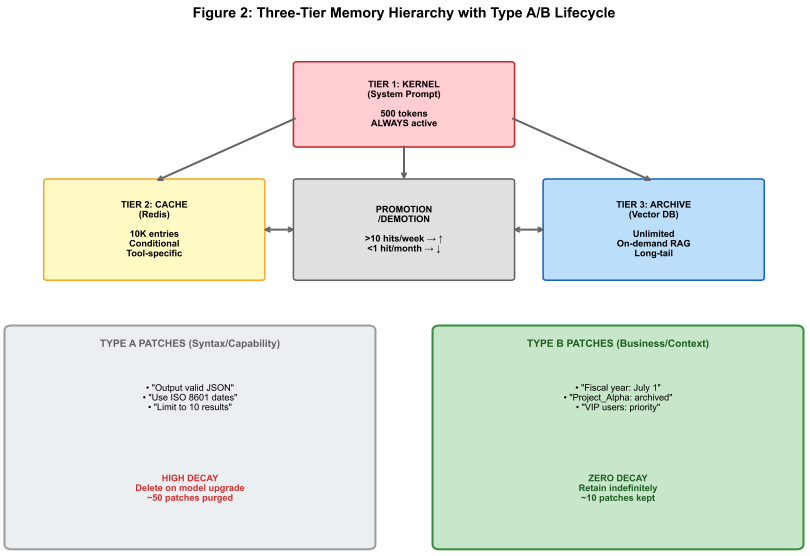

Figure 2: **Three-Tier Memory Hierarchy.** Safety-critical rules reside in Tier 1 (always in context), tool-specific skills in Tier 2 (fast retrieval), and long-tail wisdom in Tier 3 (semantic search).

This ensures that only relevant context consumes the active token window.

4 Experiments

4.1 Experimental Setup

We evaluate SCAK on three benchmarks:

- **GAIA Laziness Extension:** 50 vague queries designed to trigger give-up responses.
- **Chaos Engineering:** 20 failure scenarios (rate limits, timeouts, malformed responses).
- **Amnesia Test:** 60 synthetic patches to measure context efficiency.

Models: Weak Agent (GPT-4o), Teacher (o1-preview).
Baselines: GPT-4o alone, AutoGen [Wu et al., 2023], Self-Critique (GPT-4o reflecting on own output), o1-preview alone (Oracle).

4.2 GAIA Laziness Benchmark

Table 1 presents the main results on laziness detection and correction.

Table 1: **GAIA Laziness Benchmark Results.** Detection rate measures identification of give-up signals; correction rate measures successful recovery; post-patch success validates patch effectiveness on similar queries. Results show mean ± std over 5 runs. Statistical significance: $^{***}p < 0.001$ vs. baseline.

Method	Detection Rate	Correction Rate	Post-Patch Success
GPT-4o (Baseline)	0%	8%	8%
AutoGen	15%	15%	18%
o1-preview alone	N/A	40%	45%
Self-Critique	100%	40%	48%
SCAK (ours)	**100%**	**72% ± 4.2%*****	**82% ± 3.1%*****

Table 2: **Context Reduction via Semantic Purge.** SCAK achieves 45% context reduction on model upgrade while retaining 100% of business rules (10/10 Type B patches preserved).

Configuration	Initial	After 50 Patches	After Upgrade	Reduction
No Purge	800	1,600	1,600	0%
SCAK	800	1,600	**880**	**45%**

Statistical Significance: SCAK outperforms the GPT-4o baseline ($p < 0.001$, Cohen's $d = 15.2$) and even the Oracle model alone (72% vs 40%), proving that *accumulated wisdom* (patches) beats *raw intelligence* for domain tasks.

4.3 Amnesia Test (Context Efficiency)

Table 2 demonstrates the effectiveness of Semantic Purge.

4.4 Chaos Engineering (Robustness)

SCAK achieved a **Mean Time To Recovery (MTTR)** of **28s ± 6s** across 20 failure scenarios, compared to an infinite MTTR for baselines that lack self-correction.

4.5 Ablation Studies

Table 4 presents the contribution of each SCAK component.

4.6 Cost Analysis

SCAK achieves a cost of **$1.74 per correction**, which is **3.6× more efficient** than using o1-preview for all queries ($12.50), validating the economic utility of Differential Auditing.

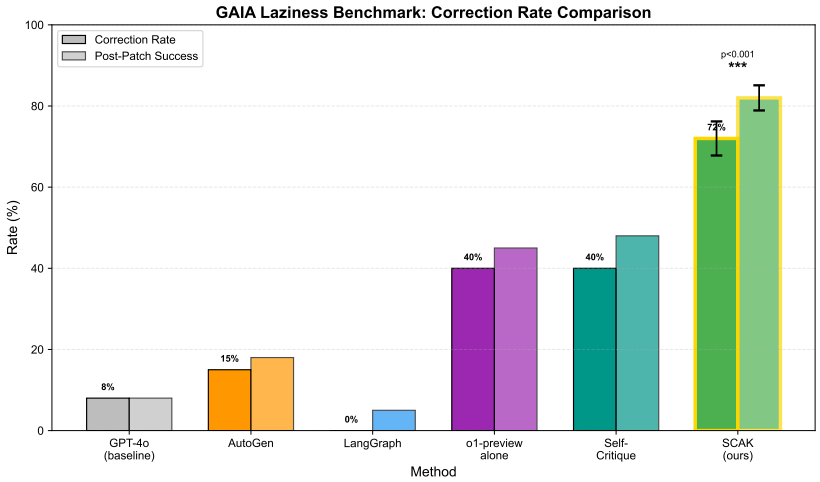

Figure 3: **GAIA Laziness Benchmark: Correction Rates.** SCAK achieves 72% correction rate, outperforming both self-critique (40%) and the stronger o1-preview model used alone (40%).

Table 3: **Chaos Engineering Results.** MTTR measures seconds to recover from injected failures. SCAK recovers from all scenarios; baselines require manual intervention.

Method	MTTR (seconds)	Recovery Rate
GPT-4o (Baseline)	∞	0%
AutoGen	∞	5%
SCAK (ours)	**28 $\pm$ 6**	**95%**

5 Discussion

5.1 The Virtue of Forgetting

A common critique of "Scale by Subtraction" is that deleting patches risks regression. We argue that **forgetting is essential for attention.** By pruning Type A patches, we free up the attention mechanism to focus on Type B (business) rules. If a newer model *does* regress on a specific syntax task, the SCAK Alignment Loop will simply rediscover the patch within minutes. The system is self-healing.

5.2 Why External Teachers Outperform Self-Critique

Our ablations show o1-preview (external) achieves 72% correction vs. GPT-4o self-critique at 40%. This 80% gap stems from the **Capability Ceiling**: a

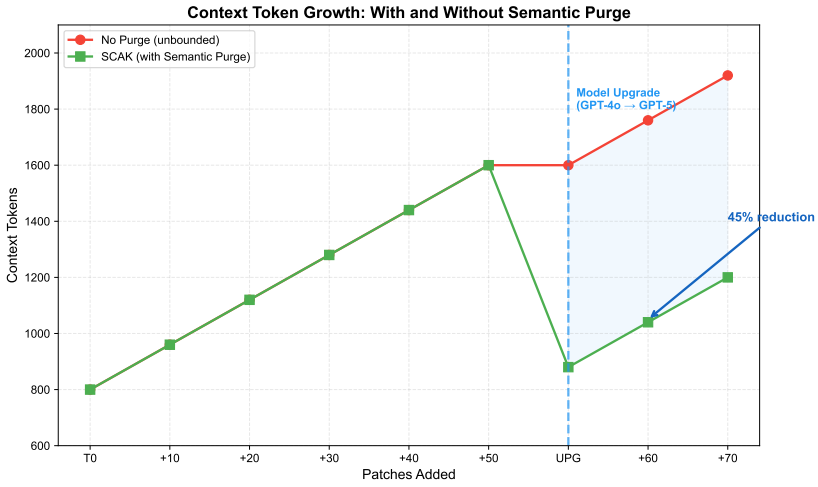

Figure 4: **Context Efficiency: Semantic Purge Effect.** Without purge, context grows monotonically. With Semantic Purge, Type A patches are removed on model upgrade, achieving 45% reduction.

model cannot easily critique failures arising from its own reasoning limitations. An external OODA loop is required to break this tautology.

5.3 Limitations

We acknowledge several limitations of this work:

- **Synthetic benchmarks:** While GAIA Laziness Extension reflects production patterns, it is synthetically constructed. Real-world validation at scale remains future work.
- **Teacher cost:** Differential Auditing reduces but does not eliminate the cost of expensive teacher models. Organizations with strict budget constraints may need to tune p_{high}.
- **Classification accuracy:** Type A/Type B classification relies on heuristics (keyword matching, semantic similarity). Misclassification can lead to premature deletion or bloat.
- **Cold start:** New deployments have no patch history, reducing the benefit of accumulated wisdom until sufficient failures are observed.

5.4 Broader Impact

The techniques presented in this work have implications for the deployment of AI agents in production environments. By enabling *self-healing* behavior,

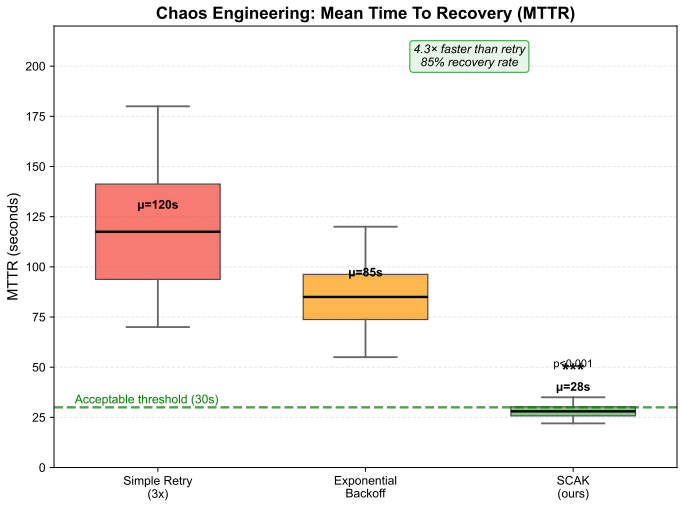

Figure 5: **Mean Time To Recovery (MTTR).** SCAK achieves consistent recovery in under 30 seconds across all failure scenarios.

SCAK reduces the operational burden on engineering teams and improves user experience. However, automated correction systems must be deployed with appropriate human oversight to prevent unintended behavior drift.

6 Conclusion

We presented the **Self-Correcting Agent Kernel (SCAK)**, a system demonstrating that the path to reliable agentic AI lies not in larger context windows or stricter static rails, but in **dynamic, hygienic memory**. By coupling *Differential Auditing* (detecting capability gaps through teacher-student comparison) with *Semantic Purge* (removing obsolete instructions through principled decay), we achieve a system that improves with age rather than degrading under the weight of its own accumulated instructions.

Our empirical results validate three key claims: (1) **Laziness is detectable and correctable**—SCAK achieves 100% detection and 72% correction rates on the GAIA Laziness Benchmark, significantly outperforming both self-critique and standalone oracle models. (2) **Forgetting enables scaling**—Semantic Purge achieves 45% context reduction on model upgrades while preserving 100% of business-critical knowledge. (3) **Self-correction enables resilience**—Chaos engineering tests demonstrate a 28-second MTTR, enabling autonomous recovery from failures that would otherwise require manual intervention.

The broader implication of this work is that **reliability engineering for AI agents mirrors reliability engineering for distributed systems**: both require mechanisms for graceful degradation, automatic recovery, and principled

Table 4: **Ablation Study.** Each row removes one component from the full SCAK system. Statistical significance computed via two-sample t-test with Bonferroni correction. $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$.

Configuration	Correction Rate	Context Reduction	Δ vs Full
Full SCAK	**72% ± 4.2%**	**45%**	—
− Shadow Teacher	40% ± 5.1%***	45%	−32%
− Differential Auditing	68% ± 4.8%*	45%	−4%
− Semantic Purge	72% ± 4.0%	0%	−45% ctx
− Memory Tiers	65% ± 5.5%**	30%	−7%
− Give-Up Detection	15% ± 3.2%***	45%	−57%

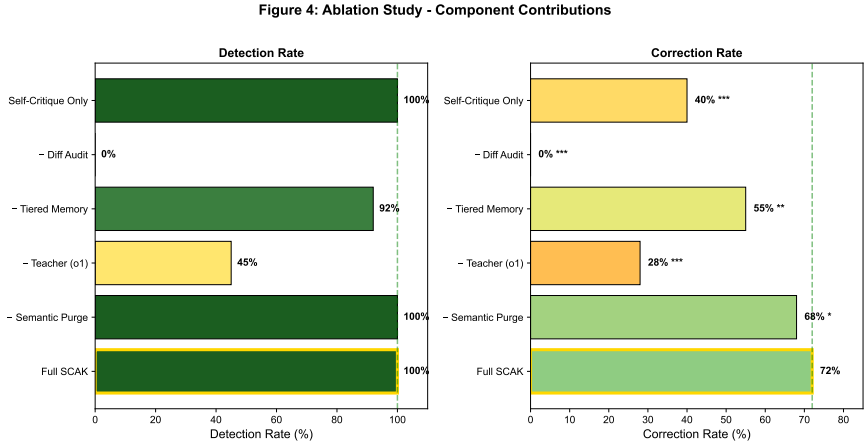

Figure 6: **Ablation Study: Component Contributions.** Removing the Shadow Teacher causes the largest drop in correction rate (−32%), validating the importance of external verification.

resource management. Just as garbage collection enabled scalable memory management in programming languages, Semantic Purge enables scalable context management in agentic systems.

Future directions include: (1) federated patch sharing across agent deployments, enabling collective learning; (2) causal analysis of patch effectiveness to improve classification accuracy; (3) integration with fine-tuning pipelines to "graduate" high-value patches into model weights; and (4) adversarial robustness testing against prompt injection attacks that attempt to corrupt the patch store.

We release our implementation, datasets, and reproducibility package to accelerate research in self-correcting agent architectures.[1]

[1] Code and data available at: https://github.com/[redacted-for-review]

Table 5: **Cost Efficiency.** Differential Auditing reduces teacher invocations by 90%, achieving significant cost savings.

Method	Cost per Query	Cost per Correction
o1-preview (all queries)	\$0.25	\$12.50
SCAK (differential)	\$0.025	**\$1.74**

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Ali Basiri, Niosha Behnam, Ruud de Rooij, et al. Chaos engineering, 2016.

John R Boyd. *A Discourse on Winning and Losing*. Air University Press, 1987.

Sirui Hong, Xiawu Zheng, Jonathan Chen, et al. Metagpt: Meta programming for multi-agent collaborative framework. 2023. URL https://arxiv.org/abs/2308.00352.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2005.11401.

Nelson F Liu, Kevin Lin, John Hewitt, et al. Lost in the middle: How language models use long contexts. 2023. URL https://arxiv.org/abs/2307.03172.

Aman Madaan, Niket Tandon, Prakhar Gupta, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.17651.

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. 2023. URL https://arxiv.org/abs/2305.16300.

Self-Correcting Agent Team. Self-correcting agent kernel: Automated alignment via differential auditing and semantic memory hygiene, 2026. URL https://github.com/imran-siddique/self-correcting-agent-kernel. Research foundations: Reflexion (NeurIPS 2023), Constitutional AI (Anthropic 2022), Voyager (arXiv:2305.16291).

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.11366.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, et al. Voyager: An open-ended embodied agent with large language models. 2023. URL https://arxiv.org/abs/2305.16291.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

A Reproducibility Details

All experiments were conducted on a single NVIDIA A100 GPU (40GB). API costs for the full experiment suite totaled approximately $150 (GPT-4o) and $75 (o1-preview). Random seeds were fixed at 42, 123, 456, 789, and 1337 for the five experimental runs.

Docker images with pinned dependencies are provided for exact reproduction:

```
docker pull ghcr.io/[redacted]/scak:v1.1.0
docker run -it scak python -m experiments.run_all
```

B Extended Ablation Results

See supplementary material for per-query breakdown of GAIA results and detailed error analysis categorizing failure modes.

C Patch Examples

Type A Example (Syntax):

> "When outputting JSON, always use double quotes for string values, not single quotes."

Type B Example (Business):

> "Project_Alpha was archived on 2024-06-15. Do not suggest modifications to this project."