TOWARDS RELIABLE FINANCIAL AGENTS: MITIGATING FINANCIAL HALLUCINATIONS THROUGH EVIDENCE-GROUNDED EXECUTION AND VERIFIED EXPERIENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite rapid progress in LLM capabilities, constructing financial agents remains a long-standing challenge due to hallucinations caused by evidence misalignment, context noise accumulation, and unreliable experience reuse. To address these issues, we introduce FinHarness, a financial agent framework that improves workflow reliability by integrating task-adaptive execution, candidate evidence verification, and verified skill accumulation. Extensive experiments show that FinHarness consistently enhances financial reasoning across various LLM backbones on the professional financial reasoning benchmark, achieving gains of up to 19.89% over state-of-the-art baselines while reducing the hallucination rate among erroneous responses by up to 73.23%. Moreover, incorporating accumulated skills further increases FinHarness's financial reasoning score from 41.86% to 52.08%. To evaluate cross-task generalization, we transfer FinHarness and the accumulated skills to the out-of-domain stock-trading task, where the skill-augmented framework outperforms all training-free baselines. These results show that evidence-aligned financial agent workflows can improve financial reasoning while reducing financial hallucinations across execution, context construction, and experience reuse.

1 INTRODUCTION

Recent advances in LLM-based agents have expanded financial agents from standalone language modeling to tool-augmented, role-specialized, and multi-step reasoning systems (Wu et al., 2023b; Yang et al., 2023; Li et al., 2023b; Yu et al., 2025; Xiao et al., 2024; Huang et al.; Deng et al., 2026). As financial agents become more capable and operationally complex, recent benchmarks have expanded to evaluate professional financial reasoning and tool use (Li et al., 2025a; Lu et al., 2026). Numerous benchmarks consistently show that hallucinations persist even as model capabilities improve. Hallucinations can arise during planning, retrieval, and reasoning, and may carry over into subsequent execution (Lin et al., 2025; Liu et al., 2026b; Xu et al., 2024). These errors are particularly consequential in finance, where incorrect claims can directly affect financial analysis and decision-making. Financial tasks rely heavily on long-context evidence and numerical reasoning, both of which remain vulnerable to hallucination (Ji et al., 2026; Zhang et al., 2025a). Prior work further shows that noisy context can degrade reasoning and unreliable experience can cascade errors into future tasks (Liu et al., 2024; Wu et al., 2025a; Xiong et al., 2026). Collectively, these recurring risks can be characterized as three failure modes of financial hallucination, comprising evidence misalignment, context noise accumulation, and unreliable experience reuse. **(1) Evidence Misalignment**, which can arise in fixed-role workflows where all tasks follow predefined agent roles and tool sequences that do not adapt to task-specific evidence requirements. This mismatch can result in redundant retrieval, irrelevant evidence, or unresolved evidence gaps (Xiao et al., 2024; Li et al., 2025b; Wu et al., 2025b). **(2) Context Noise Accumulation**, which can arise in agent workflows where retrieved information is progressively incorporated into the reasoning context without adequate filtering. This accumulation may retain stale, irrelevant, or unsupported evidence, allowing context noise to propagate through downstream reasoning (Huang et al.; Lu et al., 2026; Xiao et al., 2026; Chi et al., 2026). **(3) Unreliable Experience Reuse**, which can arise in financial agents

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

that reuse reflections, historical trajectories, or extracted skills from previous tasks to improve future decision-making. Without verifying these experiences, leveraging prior experience may transfer previous errors or task-specific reasoning patterns to new financial tasks (Wang et al., 2024; Zhang et al., 2026; Guo et al., 2026; Yang et al., 2026; Lin et al., 2026).

To address these limitations, we introduce FinHarness, a financial agent framework that harnesses evidence to align task execution, context construction, and experience reuse with task-specific requirements to reduce financial hallucinations. In particular, we design an adaptive evidence retrieval mechanism that dynamically selects agents and financial tools according to task-specific requirements, while repairing unresolved evidence gaps. Furthermore, we develop dynamic context orchestration that filters heterogeneous tool outputs according to source reliability, temporal consistency, and evidential relevance, constructing reliable reasoning contexts. Finally, we introduce a skill-based experience management strategy that enables reliable experience reuse by distilling validated execution trajectories into reusable financial skills and retrieving only task-compatible skills for future planning and tool use.

We evaluate FinHarness from four complementary perspectives, including financial reasoning capability, reasoning reliability, experience reuse, and cross-task transfer. On the financial reasoning benchmark ProFinR (Huang et al.), FinHarness improves financial reasoning score by 7.20% to 19.89% over the state-of-the-art financial reasoning baselines across four backbone models. It further reduces the hallucination rate among erroneous responses by 34.90% to 73.23%, indicating that FinHarness enhances financial reasoning performance while significantly reducing the hallucination rate among errors. In addition, skills accumulated from financial-reasoning trajectories raise the FinHarness financial reasoning score from 41.86% to 52.08%, validating the effectiveness of reusable experience. We further transfer FinHarness and the frozen skill registry to a distinct stock-trading task and conduct a six-month backtest over five stocks. FinHarness achieves 22.97% average ARR and 0.63 SR, exceeding the strongest training-free baseline on these metrics by 6.48% and 0.06, while FinHarnessSkill, the skill-augmented variant of FinHarness, reaches 33.08% ARR and 0.77 SR with corresponding gains of 16.59% and 0.20. These results demonstrate the generalizability of both the FinHarness framework and verified financial skills across tasks and evaluation settings.

Our contributions are summarized as follows:

- We mitigate financial hallucination by orchestrating how evidence is retrieved, incorporated into the reasoning context, and reused across tasks. This design improves evidence alignment, suppresses context noise, and prevents unreliable historical experience from propagating into financial reasoning.
- We develop FinHarness, an end-to-end financial agent framework with a skill accumulation process that distills validated execution trajectories into reusable financial skills and reintegrates them into subsequent planning. This process builds a transferable financial skill library, enabling reuse beyond the tasks from which the skills were accumulated.
- We introduce three core modules: adaptive evidence retriever for task-specific evidence acquisition, dynamic context orchestrator for reliable context construction, and skill-based experience manager for validated financial skill reuse. Extensive evaluations across financial reasoning, hallucination analysis, and trading demonstrate the practical applicability and strong generalizability of FinHarness across diverse financial tasks.

2 OBSERVATIONS

To expose reliability limitations under professional-grade financial reasoning demands, we evaluate representative multi-agent and experience-learning frameworks on ProFinR and identify three recurring failure patterns. As shown in Figure 1, these observations reveal systematic failures in execution alignment, evidence context, and experience reuse. In addition, we discuss our related work in Section B.

Observation 1: Evidence Misalignment. Existing financial agents such as TradingAgents often rely on fixed role-based workflows, where heterogeneous financial tasks are processed through the same predefined expert sequence. This architecture incurs redundant agent activations and accumulates task-irrelevant context, even when only a subset of roles is necessary for the task. This limitation is also reflected in the empirical results. The Fixed-Role workflow achieves an overall score

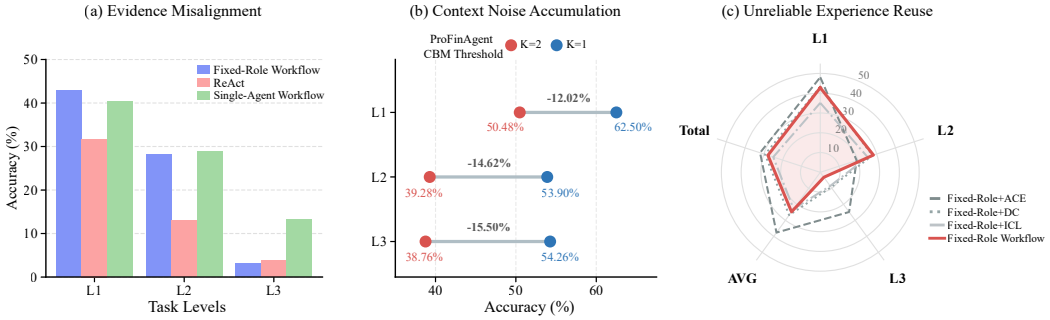

Figure 1: Overview of our preliminary observations. (a) Evidence Misalignment. (b) Context Noise Accumulation. (c) Unreliable Experience Reuse.

of 27.84%, below the 29.55% achieved by ProFinAgent, which uses structured planning and tools within a single-agent workflow. It also underperforms the simpler ReAct framework on L3 tasks. These results indicate that adding more specialized roles does not necessarily improve financial reasoning. When the execution structure is poorly aligned with task-specific evidence requirements, additional coordination can instead induce unnecessary computation and reasoning noise.

Observation 2: Context Noise Accumulation. For financial reasoning, context quality often outweighs context volume, as the reasoning process is highly susceptible to unreliable information. In ProFinR experiments, ProFinAgent with DeepSeek-R1 exhibits a consistent performance decline across task levels as more similar historical trajectories are retrieved. For example, L1 accuracy degrades from 62.50% to 50.48% when the retrieval scope expands from one trajectory to two, indicating that additional retrieved information may induce irrelevant or misleading context rather than enhance financial reasoning.

Observation 3: Unreliable Experience Reuse. Existing experience-learning methods mainly reuse historical cases, textual summaries, or semantically similar trajectories (Agarwal et al., 2024; Zhang et al., 2026; Suzgun et al., 2026). Therefore, these experiences can preserve irrelevant evidence and failed reasoning patterns, allowing previous errors to influence future financial tasks. Integrating In-Context Learning (ICL) (Agarwal et al., 2024), which reuses historical examples as demonstrations, reduces L1 accuracy from 42.79% to 35.10%, while ACE (Zhang et al., 2026), which reuses an evolving playbook distilled from prior feedback, reduces L2 accuracy from 28.27% to 19.37%. These results show that generic experience reuse does not reliably improve financial reasoning and can even degrade performance when experience is not sufficiently validated.

3 Method

3.1 Architecture Overview

In this section, we introduce FinHarness, a financial agent framework designed to improve the reliability of financial reasoning through three integrated modules. As shown in Figure 2, **(i) Adaptive Evidence Retriever** dynamically orchestrates the specific agents and tools required for each task; **(ii) Dynamic Context Orchestrator** checks candidate evidence for source provenance, temporal alignment, and evidence support before they enter downstream synthesis; **(iii) Skill-Based Experience Manager** distills execution trajectories into reusable financial skills to optimize future planning. These modules form a unified workflow that connects adaptive execution, verified reasoning, and reliable experience reuse. We provide a detailed description of the FinHarness workflow in Section D.

3.2 Adaptive Evidence Retriever

To avoid forcing heterogeneous financial tasks through the same fixed-role workflow, Adaptive Evidence Retriever dynamically allocates agents and financial tools according to task-specific evidence requirements. The selected components form a task-specific execution graph

$$G^{(k)} = \left(\mathcal{V}^{(k)}, \mathcal{E}^{(k)}\right), \tag{1}$$

where $\mathcal{V}^{(k)}$ contains the activated agents and their candidate tool sets, and $\mathcal{E}^{(k)}$ specifies the execution dependencies at workflow round k. This design makes evidence obligations serve as the unit of planning, making each activation and dependency traceable to the information the task requires.

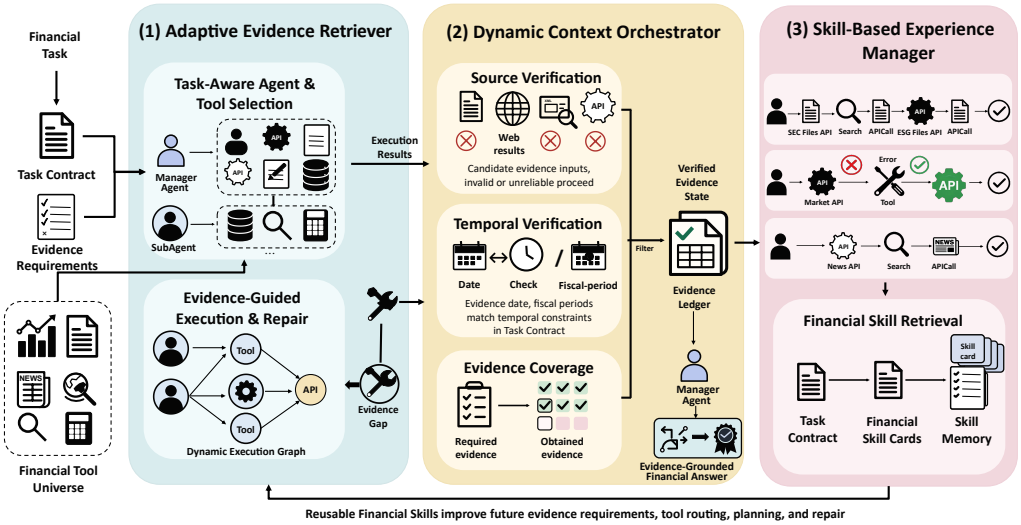

Figure 2: Overview of the FinHarness framework, which integrates an adaptive evidence retriever, a dynamic context orchestrator, and a skill-based experience manager for reliable financial reasoning.

The execution graph is determined by the evidence required at each stage. Independent operations can proceed in parallel, while dependent operations follow the necessary execution order. The graph forwards tool outputs to the downstream context module for validation before they can support financial reasoning. The Adaptive Evidence Retriever takes the task contract $\mathcal{C}$, evidence requirements $\mathcal{S}$, and Financial Tool Universe $\mathcal{U}$ as inputs and activates only the agents and tools required by the current task. After each execution round, FinHarness compares the admitted evidence with the requirements in $\mathcal{S}$. Requirements without verified support form the evidence gap. When this gap is nonempty, Adaptive Evidence Retriever performs targeted repair as

$$G_{\text{repair}}^{(k+1)} = \pi\left(\mathbf{x}^{(k)}, \mathcal{C}, \mathcal{U}\right), \tag{2}$$

where $\mathbf{x}^{(k)}$ denotes the current repair state and π denotes the repair-routing policy implemented by the manager agent. In our implementation, the repair-routing policy is an inference-time planner instantiated by the manager agent through a structured Directed Acyclic Graph (DAG). Its proposal is deterministically validated against role-capability, dependency, and budget constraints. Repair can reuse admitted upstream evidence while revising acquisition steps associated with unresolved requirements.

3.3 Dynamic Context Orchestrator

Dynamic Context Orchestrator controls which execution outputs can enter downstream financial reasoning. Tool outputs obtained from the activated nodes in $G^{(k)}$ form the candidate evidence set $\widetilde{\mathbb{L}}^{(k)}$. We denote each candidate evidence item by e. Each e is evaluated against the current task contract $\mathcal{C}$ before it can enter the admitted evidence state $\mathbb{L}^{(k)}$.

The admission process applies four complementary checks. *Source verification* establishes that e has traceable provenance from a usable financial source. *Temporal verification* checks whether its observation time and reporting period satisfy temporal constraints $\mathcal{W}$. *Financial-field validation* checks whether applicable fields, including instrument identity, price basis, accounting concept, event role, units, and metric formula, agree with $\mathcal{C}$. *Slot support* requires e to support at least one atomic evidence obligation in $\mathcal{S}$. These checks are represented by the binary predicates $V_{\text{src}}^{\text{ev}}$, $V_{\text{time}}^{\text{ev}}$, $V_{\text{fin}}^{\text{ev}}$, and $V_{\text{slot}}^{\text{ev}}$, respectively. For candidate e under contract $\mathcal{C}$, the admission rule is

$$\Phi(e \mid \mathcal{C}) = V_{\text{src}}^{\text{ev}}(e) \cdot V_{\text{time}}^{\text{ev}}(e, \mathcal{W}) \cdot V_{\text{fin}}^{\text{ev}}(e, \mathcal{C}) \cdot \max_{\rho \in \mathcal{S}} V_{\text{slot}}^{\text{ev}}(e, \rho). \tag{3}$$

The maximum is one when e supports at least one required slot and is defined as zero when $\mathcal{S} = \varnothing$. An item that fails a check or lacks information needed for verification has $\Phi(e \mid \mathcal{C}) = 0$. Admission is local to each item, so a passing item can fill one slot while other requirements remain open. Set-level coverage and remaining gaps are computed from admitted items.

We summarize the admission rule as

$$\mathbb{L}^{(0)} = \varnothing, \qquad \mathbb{L}^{(k)} = \mathbb{L}^{(k-1)} \cup \left\{ e \in \widetilde{\mathbb{L}}^{(k)} \mid \Phi(e \mid \mathcal{C}) = 1 \right\}. \tag{4}$$

Admitted evidence is appended to the evidence ledger with its claim or evidence identifier, source, temporal scope, and field-level decisions. The answerer receives factual content only from this ledger. Rejected and unverified items remain available as gap diagnostics without entering factual synthesis. The ledger and $\mathcal{S}$ determine which claims can support synthesis and which requirements still need acquisition. When $\mathbb{L}^{(k)}$ does not fully cover $\mathcal{S}$, uncovered requirements are returned to Adaptive Evidence Retriever for targeted acquisition. If gaps remain at termination, the answer covers supported slots and states the missing evidence explicitly.

3.4 Skill-Based Experience Manager

Skill-Based Experience Manager distills verified financial reasoning experience into reusable procedural abstractions for evidence acquisition, validation, and repair across tasks. Reusable payloads exclude concrete query content and task-specific answers. We denote $\widehat{Y}$ as the answer generated after the final execution round H. A retained trajectory has the form

$$\tau = \left\langle \mathcal{C}, \Xi, \widehat{Y} \right\rangle, \tag{5}$$

where the ordered trace Ξ retains each round's graph, tool calls, candidate and admitted evidence, admission decisions, and repair actions. Eligibility requires execution validity, source reliability, temporal consistency, evidence coverage, and independently verified answer correctness. Reference answers remain confined to an external verifier and are unavailable to the agent and skill compiler.

FinHarnessSkill is evaluated on ProFinR using a single-pass protocol with a precommitted deterministic query order. Each query q_i is answered from an immutable snapshot $\mathcal{M}_{i-1}^{\text{snap}}$ derived exclusively from prior queries. The response is committed before external verification, and any admitted trajectory updates memory only after q_i terminates. Any resulting update is first available when answering q_{i+1}.

Eligible trajectories with compatible task conditions and a common analysis signature are aggregated into a parameterized candidate

$$s = \langle C_s, P_s, R_s, O_s \rangle, \tag{6}$$

where C_s encodes entity and temporal constraints, metric definitions and parameters, units and currencies, price bases, and required capabilities. The component P_s captures evidence and analysis dependencies, R_s records recovery strategies, and O_s states the evidence conditions for completion. Before registration, the runtime-facing skill payload replaces instance-specific entities, dates, and file paths with placeholders, while removing query–response content. Schema validation and leakage checks reject any residual task-specific values and prohibited fields. Reusable procedural parameters, including indicator periods, remain admissible.

The mapping from a set of eligible trajectories to s is implemented through a structured compilation process without LLM-based summarization. It derives the four components from task conditions, ordered role-tool dependencies, admitted evidence states, repair transitions, and completion conditions. The generated answer remains confined to the audit record, while the external verifier supplies a post-commit admission decision.

4 Experiment

4.1 Experiment Setup

We evaluate ten representative LLMs spanning financial specialization, model scale, and access mode. Proprietary models are accessed via official APIs, while open-weight models are deployed locally using SGLang on dual NVIDIA A100 GPUs. For financial reasoning evaluation, we use ProFinR, a professional benchmark with 528 queries across three difficulty levels, covering zero-shot, agent-based, multi-agent, and experience-learning settings. For cross-task evaluation, we conduct a six-month stock-trading backtest over five stocks from January to June 2025, with ARR, SR, and MDD as the primary metrics. Detailed experimental settings are provided in Section A.1.

Table 1: Comparative evaluation of LLMs, Financial Reasoning Frameworks, Multi-Agent Frameworks, and Experience-Learning Frameworks on ProFinR.

Model & Frameworks	Accuracy				Total Score	Δ vs Naive
	L1	L2	L3	Avg.		
Baseline (Naive)						
Fin-R1	3.85%	0.52%	0.00%	1.46%	1.70%	–
Qwen3-4B-Instruct-2507	4.32%	0.00%	0.00%	1.44%	1.70%	–
Qwen3-32B	5.28%	0.00%	0.00%	1.76%	2.08%	–
DeepSeek-V3.2	16.35%	3.66%	5.43%	8.48%	9.10%	–
DeepSeek-R1	17.78%	13.08%	29.45%	20.10%	18.93%	–
DeepSeek-V4 Flash	12.50%	7.85%	3.10%	7.82%	8.52%	–
GPT-5.2	16.82%	19.37%	**65.89%**	34.03%	29.73%	–
Gemini-3 Pro	26.44%	20.41%	40.30%	29.05%	27.65%	–
Claude-Sonnet 4.6	**38.94%**	**32.98%**	58.14%	**43.35%**	**41.48%**	–
ProFinAgent						
Qwen3-4B-Instruct-2507	40.38%	28.79%	13.17%	27.45%	29.55%	+27.85%
Qwen3-32B	48.60%	32.40%	19.37%	33.46%	35.60%	+33.52%
DeepSeek-V3.2	57.69%	47.12%	46.52%	50.44%	51.14%	+42.04%
DeepSeek-R1	62.50%	53.90%	54.26%	56.89%	57.40%	+38.47%
DeepSeek-V4 Flash	62.98%	45.54%	68.22%	58.91%	57.95%	+49.43%
GPT-5.2	78.36%	78.53%	**82.94%**	79.94%	79.54%	**+49.81%**
Gemini-3 Pro	79.32%	75.91%	67.44%	74.22%	75.18%	+47.53%
Claude-Sonnet 4.6	**83.65%**	**85.34%**	80.62%	**83.20%**	**83.33%**	+41.85%
Multi-Agent Frameworks						
Fixed-role	42.79%	28.27%	3.10%	24.72%	27.84%	+26.14%
Agentverse	**50.48%**	31.94%	11.63%	31.35%	34.28%	+32.58%
FLOW	48.56%	36.65%	**14.73%**	33.31%	35.99%	+34.29%
MetaGPT	49.04%	**38.22%**	13.95%	**33.74%**	**36.55%**	**+34.85%**
Experience-Learning Frameworks						
ACE	48.08%	19.37%	24.81%	30.75%	32.01%	+30.31%
DC	43.27%	27.75%	9.30%	26.77%	29.36%	+27.66%
ICL	35.10%	25.13%	8.53%	22.92%	25.00%	+23.30%
FinHarnessSkill	**66.83%**	**45.55%**	**37.98%**	**50.12%**	**52.08%**	**+50.38%**
FinHarness						
Qwen3-4B-Instruct-2507	59.62%	32.46%	27.13%	39.74%	41.86%	+40.16%
Qwen3-32B	61.06%	51.31%	52.71%	55.03%	55.49%	+53.41%
DeepSeek-V4 Flash	73.08%	61.26%	72.09%	68.81%	68.56%	**+60.04%**
Claude-Sonnet 4.6	**91.35%**	**92.67%**	**86.05%**	**90.02%**	**90.53%**	+49.05%

4.2 Overall Financial Reasoning Performance

4.2.1 ProFinR Evaluation

Evidence-Aligned Execution Improves Financial Reasoning. As shown in Table 1, FinHarness consistently improves performance across all backbone models and difficulty levels. FinHarness achieves gains of 7.20% to 19.89% in Total Score over ProFinAgent across all evaluated models. The improvement is particularly notable for Qwen3-4B, whose score rises from 1.70% under naive inference to 41.86%. Therefore, it provides stable gains across model scales and enables smaller models to solve complex financial tasks more effectively. These findings indicate that financial reasoning depends on model capacity alongside execution structure. FinHarness orchestrates task decomposition, tool invocation, evidence validation, and final synthesis through explicit stages. This

Table 2: Comparison of error and hallucination behaviors across different LLMs and agent frameworks. Lower rates and more negative changes relative to the baseline indicate better reliability.

Model	Framework	Error Metrics		Hallucination Metrics	
		Rate (↓)	**Δ vs. Baseline** (↓)	**Rate** (↓)	**Δ vs. Baseline** (↓)
Qwen3-4B-Instruct-2507	Baseline	98.30%	–	19.08%	–
	ProFinAgent	70.45%	-27.85%	50.54%	+31.46%
	FinHarness	**58.14%**	**-40.16%**	**15.64%**	**-3.44%**
Qwen3-32B	Baseline	97.92%	–	30.17%	–
	ProFinAgent	64.39%	-33.53%	63.24%	+33.07%
	FinHarness	**44.51%**	**-53.41%**	**11.06%**	**-19.11%**
Claude-Sonnet 4.6	Baseline	58.52%	–	49.83%	–
	ProFinAgent	16.67%	-41.85%	85.23%	+35.40%
	FinHarness	**9.47%**	**-49.05%**	**12.00%**	**-37.83%**
DeepSeek-V4 Flash	Baseline	91.48%	–	11.80%	–
	ProFinAgent	42.05%	-49.43%	53.15%	+41.35%
	FinHarness	**31.44%**	**-60.04%**	**6.02%**	**-5.78%**

structure reduces the reliance on the backbone model to complete the entire task through unconstrained generation. The model follows a controlled workflow in which evidence requirements, executable specifications, and completion criteria are specified before synthesis. Using the same model, FinHarness also outperforms all representative multi-agent frameworks. It achieves a Total Score of 41.86%, exceeding the 27.84% to 36.55% range of the evaluated methods. We further observe that the fixed-role workflow achieves 27.84%, falling below the 29.55% of the single-agent workflow. This result indicates that role specialization and agent communication alone are insufficient for reliable financial reasoning. Although these mechanisms structure intermediate reasoning, they do not guarantee that the resulting workflow acquires complete, valid, and task-aligned evidence. FinHarness addresses this limitation by connecting agent coordination with explicit evidence requirements and validating the collected information before final synthesis. Therefore, workflow completion is determined by evidence coverage rather than agent response status. Execution terminates only after all required evidence has been validated, or any remaining gaps have been explicitly recorded. This evidence-driven execution helps explain the better performance on more difficult ProFinR tasks.

Orchestrated Context Construction Enhances Reasoning Reliability. To examine how uncontrolled context accumulation affects financial reasoning reliability, we ablate the Dynamic Context Orchestrator. As shown in Table 7, removing this component reduces FinHarness's financial reasoning score from 41.86% to 7.20%. This sharp decline indicates that simply increasing access to financial information does not ensure reliable task execution. A reliable multi-agent system should identify the evidence relevant to the current task instead of simply stacking or compressing all available information. The Dynamic Context Orchestrator filters according to source validity, temporal alignment, and evidence coverage. This controlled context construction ensures that downstream reasoning relies on task-relevant evidence and prevents redundant or conflicting information from affecting final synthesis.

Reliable Experience Reuse Strengthens Experience Learning. In our experiment, FinHarnessSkill updates memory between queries under the online protocol, whereas trading uses an offline transfer protocol. FinHarnessSkill follows a one-pass causal online construction protocol over the ProFinR benchmark. Each query is evaluated against an immutable memory snapshot constructed solely from preceding queries, and its trajectory updates the registry for subsequent queries only after the snapshot hash is committed and externally verified. General experience-learning methods achieve Total Scores between 25.00% and 32.01%, while FinHarness with verified financial skills reaches 52.08%. The skills improve the standard FinHarness score by 10.22% and outperform the strongest experience baseline by 20.07%. These improvements remain consistent across all difficulty levels and become especially clear on the more demanding L2 and L3 tasks. We attribute this advantage to the representation and validation of experience. FinHarness extracts procedural knowledge from verified execution trajectories and preserves evidence requirements and repair actions. In contrast, general experience-learning methods compress previous interactions into demonstrations,

reflections, or textual context. These representations can describe a failure, but they rarely identify the execution step that caused it. In financial tasks, errors may arise from unreliable sources, temporal misalignment, or incomplete evidence coverage. Therefore, a textual summary may recognize an incorrect outcome without preserving its operational cause. FinHarness addresses this limitation through structured execution traces that record source validity, temporal alignment, and evidence coverage. Only validated trajectories are used to construct reusable financial skills. These skills encode task conditions, evidence requirements, tool preferences, and repair strategies. They provide actionable procedures instead of textual summaries of past interactions. This structure allows prior experience to guide future execution without introducing unsupported financial claims.

4.2.2 Financial Hallucination and Evidence Reliability

In evaluation, we observe a recurring failure in financial agents. A claim may appear numerically plausible and even be supported by successful tool executions, but remain financially invalid when the underlying evidence is misaligned with the task requirements. This means that reliable financial reasoning requires domain-semantic validity checks beyond generic support-based hallucination detection. In financial reasoning, reliability additionally depends on whether the evidence matches the intended instrument, reporting period, accounting concept, price basis, aggregation rule, and calculation semantics. Therefore, effective detection requires assessing whether the retrieved evidence satisfies task-specific requirements and whether the final claim is supported by that evidence. We first use the benchmark reference answer and LLM judges to identify errors, followed by expert audits of the final claim, tool outputs, source status, temporal alignment, and evidence coverage. We retain only material mismatches between claims and evidence, while excluding tool failures, explicit abstentions, and formatting differences. This analysis reveals six recurring financial hallucination types: instrument entity resolution error, price basis normalization error, accounting taxonomy mapping error, temporal misalignment, earnings field binding error, and financial metric definition error. Representative cases are shown in Section A.2.3.

As shown in Table 2, advanced financial reasoning frameworks do not necessarily ensure reliable financial reasoning. ProFinAgent substantially reduces the overall error rate compared to other frameworks, validating the effectiveness of structured planning. However, the remaining errors reveal a more pronounced reliability issue. Across all evaluated models, hallucinations account for 50.54% to 85.23% of the residual errors. The results reveal a critical limitation of existing financial agent frameworks, as improved tool planning cannot ensure that the retrieved evidence is valid, complete, and sufficient to support the final claim. Without explicit verification, incomplete or temporally inconsistent evidence can still enter the synthesis process and lead to plausible but unsupported financial conclusions. FinHarness addresses this problem through verified workflow execution and explicit evidence verification. Specifically, it rigorously evaluates tool outputs for source validity, temporal consistency, and evidence coverage before response synthesis. FinHarness represents missing evidence as explicit execution gaps that trigger targeted repair. This design tightly couples evidence validation with execution control, allowing unreliable evidence to be excluded while unresolved requirements are returned for additional acquisition. Therefore, the reasoning process proceeds only with evidence that has been explicitly validated against the requirements of the task. The experimental results show that FinHarness consistently suppresses hallucination rates across all four tested models, reducing them to a range of 6.02% to 15.64%. Moreover, FinHarness achieves lower overall error rates than ProFinAgent in all settings. These results show that financial agents require reliable reasoning beyond planning and tool access.

4.3 Generalization to Trading Tasks

As shown in Figure 3, we compare FinHarness with the trading framework AlphaQuanter across five stocks and observe consistent improvements in both return and risk-adjusted performance. Using Qwen3-4B, FinHarness achieves an average ARR of 22.97%, an SR of 0.63, and an MDD of -15.80%, substantially outperforming all non-RL AlphaQuanter variants, including the Qwen3-30B-A3B configuration with a substantially larger backbone. This result indicates that the gains are attributable not simply to model scale or stronger base-model capabilities, but rather to the proposed FinHarness framework. Moreover, FinHarnessSkill raises the average ARR to 33.08% and the SR to 0.77, demonstrating the effectiveness of learned financial skills in trading. We attribute this generalization to skills distilled from verified financial-reasoning trajectories, which encode reusable procedures instead of task-specific answers. These trajectories capture reusable financial skills,

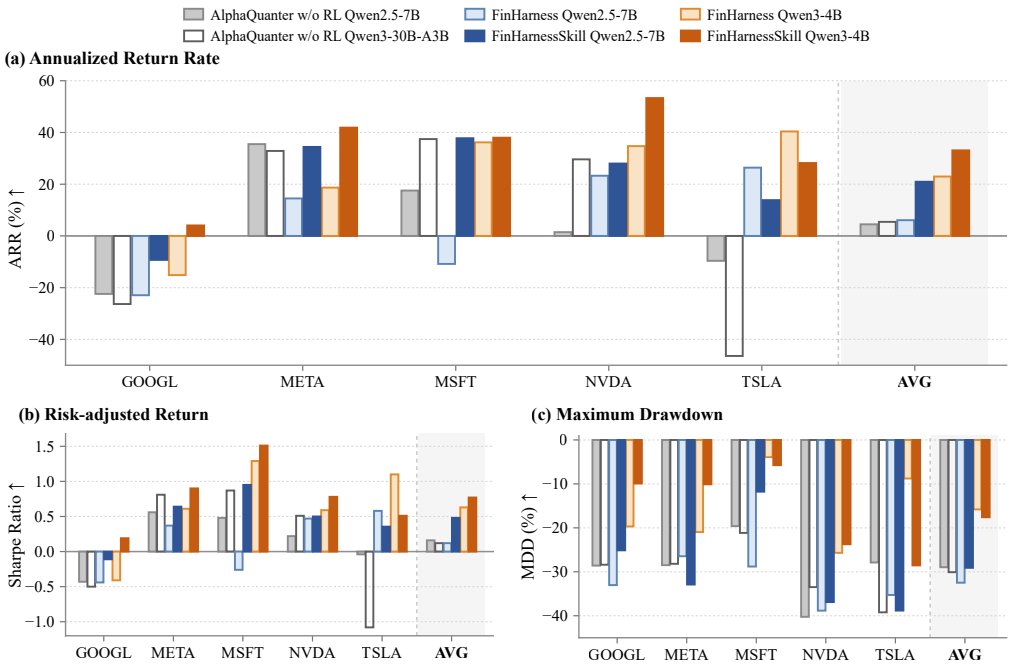

Figure 3: Comparison of Backtesting Performance with AlphaQuanter. Detailed comparisons with trading strategies, learning-based methods, and agentic frameworks are provided in Table 12.

such as evidence acquisition, temporal validation, cross-source verification, and evidence-grounded decision-making. In addition, these skills remain directly relevant even when the downstream task shifts from financial reasoning to trading.

As shown in Table 12, FinHarnessSkill further increases the average ARR to 33.08% and produces positive returns across all five stocks. The consistent gains across stocks suggest that the improvement reflects a more reliable trading process and generalizes beyond any single favorable asset. Moreover, it approaches the performance of the best AlphaQuanter configuration, which is particularly important because AlphaQuanter requires reinforcement learning to optimize its trading policy. In contrast, FinHarness achieves competitive performance without parameter updates or specific policy training. Therefore, FinHarness improves trading decisions only through verified experience. The risk metrics provide further evidence of stable performance. FinHarnessSkill achieves the highest SR of 0.77 among all evaluated methods. This result indicates that its returns are obtained with a more favorable balance between profit and risk. Both FinHarness variants also keep the MDD magnitude below 18%. Their drawdowns are substantially smaller than those of the trained baseline. These results demonstrate both the extensibility of FinHarness and the transferability of financial skills. FinHarness directly generalizes its verified execution mechanism from financial reasoning to trading, while FinHarnessSkill further benefits from reusable skills. Therefore, the improvement shows that verified execution trajectories could be converted into reliable financial capabilities that generalize across tasks. This provides a scalable mechanism for agents to continuously accumulate financial expertise.

5 Conclusion

In this paper, we introduce FinHarness, a financial agent framework that improves reliability across evidence acquisition, context construction, and experience reuse to mitigate financial hallucinations. FinHarness further distills validated execution trajectories into reusable financial skills. Experiments on professional financial reasoning and stock trading demonstrate consistent gains across diverse LLM backbones and support cross-task transfer of the accumulated skills. These results highlight the importance of reliable evidence handling and experience reuse throughout financial agent workflows. Future work will explore broader financial domains, longer market regimes, and imperfect verification settings.

AI USE STATEMENT

Generative AI tools were used to assist with research ideation and execution, including brainstorming and refining methodological and experimental choices. They were also used to support literature retrieval and discovery, draft and revise portions of the manuscript, and improve the clarity, grammar, and style of the writing. In addition, generative AI models served as experimental systems in the financial reasoning and trading evaluations described in this work. All AI-assisted research decisions, retrieved information and references, experimental procedures, numerical results, analyses, and manuscript content were independently reviewed and verified by the authors. The authors take full responsibility for the methodology, experiments, claims, results, and final content of the submission.

ETHICS STATEMENT

This work studies the reliability of LLM-based financial agents and aims to reduce financial hallucinations through controlled evidence acquisition, context construction, and experience reuse. FinHarness improves reliability but does not eliminate model errors, and its outputs should not be interpreted as financial advice or used as the sole basis for real-world investment decisions. Incorrect or incomplete financial claims may still lead to material consequences when deployed without appropriate human oversight. All trading experiments in this work are conducted through historical backtesting and do not involve real capital or live market intervention. The study does not require personally identifiable or sensitive user information. Therefore, we view FinHarness as a research framework for studying and improving reliable financial reasoning, and recommend appropriate validation, risk controls, and human supervision before any use in high-stakes financial applications.

REPRODUCIBILITY STATEMENT

We provide model and deployment configurations, evaluation protocols, tool and baseline specifications, prompt templates, financial-skill construction and leakage-control procedures, and detailed trading settings in Appendices A.1 and D. The skill registry is versioned throughout evaluation. On ProFinR, each query is evaluated against an immutable pre-query snapshot, with append-only updates becoming available only to subsequent queries. For trading evaluation, the transferred registry is frozen before backtesting and remains unchanged throughout the evaluation period. Configuration manifests and evaluation scripts will be released in the anonymous supplementary material.

REFERENCES

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966, 2024.

Anthropic. Claude-Sonnet 4.6: Advanced multimodal large language model. https://platform.claude.com/docs/en/about-claude/models/overview, 2026. Accessed: 2026-04-17.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024.

Jijun Chi, Zhenghan Tai, Hanwei Wu, Tung Sum Thomas Kwok, Hailin He, Zixing Liao, Bohuai Xiao, Chaolong Jiang, Jianliang Lei, Jerry Huang, Peng Lu, Muzhi Li, Liheng Ma, Yihong Wu, Sicheng Lyu, Jingrui Tian, Yihan Li, Yanzhang Ma, Sizhe Guan, Dingtao Hu, Yufei Cui, Ling Zhou, Lei Ding, and Xinyu Wang. Finsagent: Corpus-aligned multi-agent rag framework for evidence-grounded sec filing question answering, 2026. URL https://arxiv.org/abs/2607.18102.

DeepSeek AI. DeepSeek-R1: Advanced reasoning large language model. https://api-docs.deepseek.com/, 2025a. Accessed: 2026-01-22.

DeepSeek AI. DeepSeek-V3.2: General-purpose large language model. https://api-docs.deepseek.com/, 2025b. Accessed: 2026-04-17.

DeepSeek-AI. DeepSeek-V4-Flash: Mixture-of-experts language model. https://api-docs.deepseek.com, 2026. Accessed: 2026-08-20.

Zheye Deng, Weixiang Yan, Changlong Yu, and Jiashu Wang. Alphaquanter: An end-to-end tool-augmented agentic reinforcement learning framework for stock trading. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9373–9394, 2026.

Google DeepMind. Gemini 3 Pro: Multimodal generative ai model. https://ai.google.dev/models/gemini, 2025. Accessed: 2026-01-22.

Zhongxin Guo, Danrui Qi, Hanwen Gu, Peng Cheng, and Yongqiang Xiong. Skill-disco: Distilling and compiling agent traces into reusable procedural skills. *arXiv preprint arXiv:2606.26669*, 2026.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Cheng Huang, Jinghua Piao, Ranran Wang, and Yong Li. Towards professional-grade financial agents: Benchmarking, tooling, and structured reasoning. In *Forty-third International Conference on Machine Learning*.

Lanlan Ji, Dominic Seyler, Gunkirat Kaur, Manjunath Hegde, Koustuv Dasgupta, and Bing Xiang. Phantom: A benchmark for hallucination detection in financial long-context qa. *Advances in Neural Information Processing Systems*, 38, 2026.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023a.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Kp Subbalakshmi, Jimin Huang, et al. Investorbench: A benchmark for financial decision-making tasks with llm-based agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2509–2525, 2025a.

Xiangyu Li, Yawen Zeng, Xiaofen Xing, Jin Xu, and Xiangmin Xu. Hedgeagents: A balanced-aware multi-agent financial trading system. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 296–305, 2025b.

Yang Li, Yangyang Yu, Haohang Li, Zhi Chen, and Khaldoun Khashanah. Tradinggpt: Multi-agent system with layered memory and distinct characters for enhanced financial trading performance. *arXiv preprint arXiv:2309.03736*, 2023b.

Yuan Li, Bingqiao Luo, Qian Wang, Nuo Chen, Xu Liu, and Bingsheng He. Cryptotrade: A reflective llm-based agent to guide zero-shot cryptocurrency trading. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1094–1106, 2024.

Huawei Lin, Peng Li, Jie Song, Fuxin Jiang, and Tieying Zhang. Muse-autoskill: Self-evolving agents via skill creation, memory, management, and evaluation. *arXiv preprint arXiv:2605.27366*, 2026.

Xixun Lin, Yucheng Ning, Jingwen Zhang, Yan Dong, Yilong Liu, Yongxuan Wu, Xiaohua Qi, Nan Sun, Yanmin Shang, Kun Wang, et al. Llm-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions. *arXiv preprint arXiv:2509.18970*, 2025.

Hongjun Liu, Yifei Ming, Shafiq Joty, and Chen Zhao. Harnessing llm agents with skill programs. *arXiv preprint arXiv:2605.17734*, 2026a.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Xuannan Liu, Xiao Yang, Zekun Li, Peipei Li, and Ran He. Agenthallu: Benchmarking automated hallucination attribution of llm-based agents. *arXiv preprint arXiv:2601.06818*, 2026b.

Zhaowei Liu, Xin Guo, Zhi Yang, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Mengping Li, Qi Qi, Zhiqiang Liu, Yiyang Han, et al. Fin-r1: A large language model for financial reasoning through reinforcement learning. *arXiv preprint arXiv:2503.16252*, 2025.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. *arXiv preprint arXiv:2310.02170*, 2023.

Jiaxuan Lu, Kong Wang, Yemin Wang, Qingmei Tang, Hongwei Zeng, Xiang Chen, Jiahao Pi, Shujian Deng, Lingzhi Chen, Yi Fu, et al. Fintoolbench: Evaluating llm agents for real-world financial tool use. *arXiv preprint arXiv:2603.08262*, 2026.

Boye Niu, Yiliao Song, Kai Lian, Yifan Shen, Yu Yao, Kun Zhang, and Tongliang Liu. Flow: Modularized agentic workflow automation. In *International Conference on Learning Representations*, volume 2025, pp. 74949–74977, 2025.

OpenAI. GPT-5.2: Language models for advanced reasoning. `https://developers.openai.com/api/docs/models/gpt-5.2`, 2025. Accessed: 2026-01-22.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, 2026.

Ziyi Tang, Zechuan Chen, Jiarui Yang, Jiayao Mai, Yongsen Zheng, Keze Wang, Jinrui Chen, and Liang Lin. Alphaagent: Llm-driven alpha mining with regularized exploration to counteract alpha decay. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 2813–2822, 2025.

Qwen Team. Qwen2.5 technical report, 2024. URL `https://arxiv.org/abs/2412.15115`.

Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Jinyang Wu, Shuai Zhang, Feihu Che, Mingkuan Feng, Pengpeng Shao, and Jianhua Tao. Pandora's box or aladdin's lamp: A comprehensive analysis revealing the role of rag noise in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5019–5039, 2025a.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023a.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023b.

Yingqian Wu, Qiushi Wang, Zefei Long, Rong Ye, Zhongtian Lu, Xianyin Zhang, Bingxuan Li, Wei Chen, Liwen Zhang, and Zhongyu Wei. Finteam: A multi-agent collaborative intelligence system for comprehensive financial scenarios. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 443–455. Springer, 2025b.

Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. Tradingagents: Multi-agents llm financial trading framework. *arXiv preprint arXiv:2412.20138*, 2024.

Yijia Xiao, Rujun Han, Yanfei Chen, Zifeng Wang, Ke Jiang, Zhongying CuiZhu, Vishy Tirumalashetty, Wei Wang, Burak Gokturk, Tomas Pfister, et al. Financeharness: Autonomous financial deep research framework. *arXiv preprint arXiv:2607.27853*, 2026.

Fei Xiong, Xiang Zhang, Aosong Feng, Siqi Sun, and Chenyu You. Quantagent: Price-driven multi-agent llms for high-frequency trading. *arXiv preprint arXiv:2509.09995*, 2025a.

Guojun Xiong, Zhiyang Deng, Keyi Wang, Yupeng Cao, Haohang Li, Yangyang Yu, Xueqing Peng, Mingquan Lin, Kaleb E Smith, Xiao-Yang Liu, et al. Flag-trader: Fusion llm-agent with gradient-based reinforcement learning for financial trading. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13921–13934, 2025b.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts llm agents: An empirical study of experience-following behavior. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 623–645, 2026.

Hongshen Xu, Zichen Zhu, Lei Pan, Zihan Wang, Su Zhu, Da Ma, Ruisheng Cao, Lu Chen, and Kai Yu. Reducing tool hallucination via reliability alignment. *arXiv preprint arXiv:2412.04141*, 2024.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models. *FinLLM Symposium at IJCAI 2023*, 2023. URL https://arxiv.org/abs/2306.06031.

Hongyang Yang, Boyu Zhang, Neng Wang, Cheng Guo, Xiaoli Zhang, Likun Lin, Junlin Wang, Tianyu Zhou, Mao Guan, Runjia Zhang, and Christina Dan Wang. Finrobot: An open-source ai agent platform for financial applications using large language models, 2024. URL https://arxiv.org/abs/2405.14767.

Min Yang, Jinghua Piao, Xu Xia, Xiaochong Lan, Jiaju Chen, Yongshun Gong, and Yong Li. Skillmaster: Toward autonomous skill mastery in llm agents. *arXiv preprint arXiv:2605.08693*, 2026.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan W Suchow, Zhenyu Cui, Rong Liu, et al. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *Advances in Neural Information Processing Systems*, 37:137010–137045, 2024.

Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Jordan W Suchow, Denghui Zhang, and Khaldoun Khashanah. Finmem: A performance-enhanced llm trading agent with layered memory and character design. *IEEE Transactions on Big Data*, 2025.

Mengao Zhang, Jiayu Fu, Tanya Warrier, Yuwen Wang, Tianhui Tan, and Ke-wei Huang. Faith: A framework for assessing intrinsic tabular hallucinations in finance. In *Proceedings of the 6th ACM International Conference on AI in Finance*, pp. 159–167, 2025a.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024. URL https://arxiv.org/abs/2308.10144.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

A APPENDIX

A.1 EXPERIMENTAL SETUP AND REPRODUCIBILITY

A.1.1 FOUNDATION MODELS AND DEPLOYMENT INFRASTRUCTURE

As shown in Table 3, we evaluate FinHarness on ten diverse foundation models for a comprehensive assessment. To establish strong baselines, we access closed-source models through their official APIs, while serving open-weight models locally to ensure precise control over generation settings and latency measurements. Open-weight models are served with SGLang on dual NVIDIA A100 GPUs, which provide sufficient memory for long-context inference. To reduce variation in the embedding pipeline, we use Qwen3-Embedding-0.6B (Zhang et al., 2025b) as the shared embedding model across all experiments.

Table 3: Overview of Evaluated Models (Open Weights vs. API).

Model Name	**Size**	**Form**	**Access**
DeepSeek-R1 (DeepSeek AI, 2025a)	–	API	Remote
DeepSeek-V3.2 (DeepSeek AI, 2025b)	–	API	Remote
DeepSeek-V4 Flash (DeepSeek-AI, 2026)	–	API	Remote
GPT-5.2 (OpenAI, 2025)	–	API	Remote
Gemini-3 Pro (Google DeepMind, 2025)	–	API	Remote
Claude-Sonnet 4.6 (Anthropic, 2026)	–	API	Remote
Qwen3-4B-Instruct-2507 (Team, 2025)	4B	Open Weights	Local
Qwen2.5 (Team, 2024)	7B	Open Weights	Local
Fin-R1 (Liu et al., 2025)	7B	Open Weights	Local
Qwen3-32B (Team, 2025)	32B	Open Weights	Local

A.1.2 DATASET AND EVALUATION METRICS FOR FINANCIAL REASONING TASK

We evaluated all methods on ProFinR, which contains 528 queries across three difficulty levels. The comparison covers zero-shot inference, ReAct, ProFinAgent, multi-agent frameworks, experience-learning frameworks, and FinHarness with and without financial-skill reuse. Table 1 shows the overall performance comparison. **Avg.** denotes the macro-average across L1–L3 difficulty levels, while **Total Score** represents the instance-weighted score over the entire benchmark.

A.1.3 DATASET AND BASELINES FOR THE TRADING TASK

Dataset and backtesting period. We evaluate FinHarness on stock trading to examine its effectiveness beyond financial reasoning. Our evaluation protocol follows AlphaQuanter exactly (Deng et al., 2026). The aligned setting controls for differences in asset selection and market conditions. Therefore, it supports a direct comparison between FinHarness and existing trading methods. Table 4 summarizes the evaluation set.

Baselines. We classify the evaluated baselines according to their underlying trading paradigms. **Market** denotes a passive buy-and-hold strategy. **Rule** denotes strategies that make decisions through fixed technical trading rules. **RL** represents conventional deep reinforcement learning agents that learn trading policies from numerical market states. **LM** denotes language modeling methods that forecast future prices from historical sequences. **LLM** represents prompt-based language model traders without reinforcement learning optimization. **Fixed-role** denotes the multi-agent setting, where specialized agents collaborate to produce a trading decision. **AlphaQuanter w/o RL** denotes a tool-augmented single agent that retains the original tools and prompting structure without policy training. **AlphaQuanter** denotes the corresponding single agent after reinforcement learning optimization.

Evaluation Metrics. Following prior work (Xiong et al., 2025b; Deng et al., 2026), we use Annualized Rate of Return, Sharpe Ratio, and Maximum Drawdown as our three principal evaluation metrics.

Table 4: Trading evaluation set.

Item	Setting
Stock universe	Alphabet Inc. (**GOOGL**); Microsoft Corporation (**MSFT**); Meta Platforms, Inc. (**META**); NVIDIA Corporation (**NVDA**); Tesla, Inc. (**TSLA**)
Test start	January 1, 2025
Test end	June 30, 2025
Duration	Six months
Trading days	122

- **Annualized Rate of Return** (ARR) measures profitability by annualizing the total portfolio return over a standard 252-trading-day year:

$$\mathrm{ARR} = \left(\frac{V_T}{V_0}\right)^{\frac{252}{T}} - 1, \tag{7}$$

where V_0 and V_T denote the initial and final values, respectively, and T is the number of trading days in the backtesting period. In our six-month evaluation, $T = 122$, whereas 252 denotes the standard number of trading days used for annualization.

- **Sharpe Ratio** (SR) measures risk-adjusted performance using the mean and sample standard deviation of daily portfolio returns:

$$\begin{aligned} r_t &= \frac{V_t - V_{t-1}}{V_{t-1}}, & \bar{r} &= \frac{1}{T}\sum_{t=1}^{T} r_t, \\ \sigma_r &= \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(r_t - \bar{r})^2}, & \mathrm{SR} &= \frac{\bar{r}}{\sigma_r}. \end{aligned} \tag{8}$$

where r_t is the daily portfolio return, $\bar{r}$ is its sample mean, and σ_r is its sample standard deviation.

- **Maximum Drawdown** (MDD) measures the largest peak-to-trough decline. Because our tables report drawdowns as negative returns, we use the following signed convention:

$$\begin{aligned} P_t &= \max_{0 \le u \le t} V_u, \\ \mathrm{MDD} &= \min_{0 \le t \le T}\left(\frac{V_t - P_t}{P_t}\right) = -\max_{0 \le t \le T}\left(\frac{P_t - V_t}{P_t}\right). \end{aligned} \tag{9}$$

where P_t is the historical peak value up to trading day t. Under this convention, $\mathrm{MDD} \in [-1, 0]$.

A.1.4 Baseline Multi-Agent Workflows and Experience-Learning Frameworks

We compare our method against multi-agent workflows and experience-learning frameworks. These baselines span a broad range of state-of-the-art paradigms, from collaborative multi-agent reasoning to iterative experience-based refinement.

Multi-Agent Frameworks:

- **Fixed-role** (Xiao et al., 2024): The architecture coordinates specialized financial agents through a predefined role-based workflow with a fixed execution structure.
- **Agentverse** (Chen et al., 2024): The architecture provides a flexible environment for dynamic multi-agent collaboration to foster cooperative problem-solving and emergent behaviors.

- **FLOW** (Niu et al., 2025): The architecture modularizes complex workflows into highly automated and reusable processing components.
- **MetaGPT** (Hong et al., 2024): The architecture uses meta-programming principles to orchestrate specialized agents across large-scale projects.

Experience-Learning Frameworks:

- **ICL** (Agarwal et al., 2024): The framework directly incorporates historical execution exemplars into the prompt space to guide subsequent generation.
- **ACE** (Zhang et al., 2026): The framework systematically distills past interactions to iteratively evolve the underlying operational context.
- **Dynamic Cheatsheet (DC)** (Suzgun et al., 2026): The framework actively refreshes its memory repository to maintain continuous alignment with shifting environmental feedback.

A.2 Additional Experiments

A.2.1 Additional Experiments for Financial Reasoning Task

To further evaluate the effectiveness of FinHarness and its individual modules, we conduct two complementary experiments. First, we perform module-level ablations on the baseline model by removing each core component individually, allowing us to isolate its contribution to financial reasoning performance. Second, we evaluate foundation models equipped with the ReAct framework and external tools, providing a tool-augmented baseline for comparison with FinHarness.

Module Ablation. We examine the contribution of each FinHarness component through controlled ablation experiments. As shown in Table 7, removing any individual module consistently degrades performance, indicating that the three components provide complementary benefits. The degradation is particularly pronounced when the Dynamic Context Orchestrator is removed, suggesting that controlling the evidence admitted to downstream reasoning is critical for reliable financial reasoning. Removing Adaptive Evidence Retriever also causes a substantial decline, highlighting the importance of task-aware execution. The smaller but consistent degradation without Skill-Based Experience Manager further shows that validated procedural experience provides additional gains beyond the execution and verification components. These results support the effectiveness of jointly modeling adaptive execution, evidence verification, and procedural experience.

Experience-Learning Ablation. As shown in Table 5, the results show that generic experience-learning strategies provide limited benefits for financial reasoning. For example, ACE maintains an evolving playbook by summarizing execution feedback into reusable guidance. However, this semantic abstraction can accumulate rules that appear broadly relevant while remaining inconsistent with the evidence requirements and tool procedures of a specific financial task. This negative transfer is particularly evident on L2 tasks. Incorporating ACE into the fixed-role workflow reduces L2 accuracy from 28.27% to 19.37%, indicating negative transfer. The same effect persists within FinHarness, where L2 accuracy drops from 32.46% to 28.27%. In contrast, FinHarnessSkill extracts reusable skills only from validated trajectories and preserves task conditions, evidence requirements, tool preferences, and repair strategies. It achieves a Total Score of 52.08%, improving the standard FinHarness by 10.22 percentage points and outperforming the strongest generic experience-learning baseline by 5.49 percentage points.

ReAct Baseline with Tool Use. We further evaluate foundation models equipped with the ReAct framework and external financial tools to assess the capability of general tool-augmented agents. Table 6 shows a clear dependence of the results on the underlying model's capabilities, while tool use alone does not yield uniformly reliable financial reasoning. The performance gap across models indicates that iterative tool interaction does not by itself resolve the challenges of evidence alignment and task-specific execution. This provides a stronger training-free reference for evaluating FinHarness and isolates the benefits introduced by its structured execution and evidence management.

Table 5: Ablation of Experience-Learning Frameworks in FinHarness on ProFinR.

Method	L1	L2	L3	AVG.	Total Score
FinHarness	59.62%	32.46%	27.13%	39.74%	41.86%
FinHarnessSkill	66.83%	45.55%	37.98%	50.12%	52.08%
ACE	39.42%	28.27%	26.36%	31.35%	32.20%
DC (CU)	56.25%	45.55%	32.56%	44.79%	46.59%
ICL	55.29%	38.22%	26.36%	39.96%	42.05%

Table 6: Performance Evaluation of Various Foundation Models using the ReAct Framework on ProFinR

Model	L1	L2	L3	AVG.	Total Score
Qwen3-4B-Instruct-2507	31.73%	13.08%	3.87%	16.23%	18.18%
Qwen3-32B	40.38%	23.03%	7.75%	23.72%	26.14%
DeepSeek-R1	43.75%	30.89%	21.70%	32.11%	33.71%
GPT-5.2	69.23%	38.74%	46.51%	51.49%	52.65%
Gemini-3 Pro	73.07%	46.60%	45.74%	55.14%	56.82%
Claude-Sonnet 4.6	65.87%	61.78%	65.89%	64.51%	63.39%

Table 7: Ablation experiments on Qwen3-4B-Instruct-2507.

Model	Framework	L1	L2	L3	AVG.	Total Score
Qwen3-4B-Instruct-2507	Baseline	4.32%	0.00%	0.00%	1.44%	1.70%
	FinHarnessSkill	66.83%	45.55%	37.98%	50.12%	52.08%
	w/o AER	42.79%	28.79%	3.1%	24.89%	28.03%
	w/o DCO	13.94%	4.71%	0.00%	6.22%	7.20%
	w/o SBEM	59.62%	32.46%	27.13%	39.74%	41.86%

A.2.2 Additional Experiments for the Trading Task

To further validate the efficacy and generalizability of our proposed framework, we conduct additional experiments on the quantitative trading task using the Qwen2.5-7B model as the backbone. Specifically, we evaluate both our base framework and skill-augmented variant, testing them against AlphaQuanter equipped with the Qwen2.5-7B model. This direct comparison effectively isolates the architectural and methodological contributions of our approach from the underlying capabilities of the foundation model.

As shown in Tables 13 and 14, FinHarness establishes a robust foundation for trading, achieving an average ARR of 6.09%, which outperforms the 4.49% average ARR achieved by AlphaQuanter. In addition, FinHarnessSkill achieves an average ARR of 20.95% and an SR of 0.48, demonstrating a substantial and consistent outperformance over AlphaQuanter across the evaluated portfolio. These results demonstrate the contribution of the skill module in our framework. While the base FinHarness architecture provides an effective substrate for market analysis, the explicit integration of domain-specific skills equips the LLM with improved action-selection and risk-management capabilities.

Table 8: Representative expert-audited financial hallucination cases. Each output contains a definite financial claim that materially conflicts with the benchmark contract in a distinct semantic dimension.

Hallucination Type	Audited Output vs. Benchmark	Financial-Semantic Mismatch
Instrument identity	WTI (`CL=F`) close: \$2.15 vs. \$68.17/bbl.	A broad provider alias (`WTI`) was treated as the requested futures identifier, so successful retrieval and date alignment did not establish instrument identity.
Price adjustment and scale	NFLX close on 2024-02-02: \$56.46 vs. \$565.32.	The requested raw-price contract was not reconciled with the provider's adjustment and corporate-action basis, producing an approximately tenfold scale mismatch.
Accounting-field identity	Boeing FY2023 long-term debt: \$52.055B vs. \$47.1B.	A successful statement-field call did not establish that the returned taxonomy concept matched the requested debt definition and inclusion boundary.
Temporal Misalignment	\$0.46 vs. The dividend payout dates for 2023 were: February 17, 2023 (\$0.44 per share); May 19, 2023 (\$0.44 per share).	The agent incorrectly reused 2022 dividend information and reported it as 2023 data.
Earnings-event field roles	DIS FY2023 Q4 consensus EPS: \$0.67 vs. \$0.70; derived surprise 22.4% vs. approximately 17%.	Actual EPS was correct, but the consensus field was misbound within the earnings event and the error propagated into derived quantities.
Metric definition	TGT FY2023 asset turnover: 1.99 vs. approximately 1.94.	Ending total assets replaced average total assets in the denominator, and the displayed inputs were not arithmetically consistent with the reported ratio.

A.2.3 Financial Hallucination Types and Case Studies

We identify financial hallucinations through a two-stage audit. First, the benchmark reference and an LLM judge identify incorrect-answer candidates. This stage is used only for candidate discovery and does not assign hallucination labels. Financial-domain experts inspect each candidate against the query contract, final answer, benchmark reference, cited tool outputs, source status, temporal alignment, evidence coverage, and any expressed uncertainty. A case is retained only when the answer makes a definite financial claim with a material mismatch in financial semantics. We exclude tool or provider failures without a fabricated claim, disclosed missing evidence, explicit abstentions, formatting differences, and reasonable rounding. Table 8 summarizes six representative cases. These cases serve to illustrate distinct failure mechanisms. Furthermore, the term "expert-audited" signifies a confirmed discrepancy between the claim and the contract, and does not imply an independent verification of the ground truth.

A.3 Financial Skills in FinHarness versus Experience-Learning Memories

Experience-learning methods and the Skill-Based Experience Manager both reuse prior interactions, but they differ in what is transferred and how reuse is controlled. In our evaluation, ICL (Agarwal et al., 2024) transfers an immutable workflow exemplar, whereas ACE (Zhang et al., 2026) and Dynamic Cheatsheet (DC) (Suzgun et al., 2026) extract unstructured natural-language heuristics distilled from isolated trajectories. These objects are selected as prompt context and can influence subsequent reasoning. In contrast, Skill-Based Experience Manager transfers a parameterized *financial skill*: a contract-bearing representation of a recurring financial task, its evidence dependencies, admissible tool pattern, characteristic repair route, and completion conditions. Table 9 summarizes these architectural distinctions.

Table 9: Comparison between the experience representations used by the evaluated baselines and the financial skills learned by FinHarness.

Dimension	ICL	ACE / DC	FinHarness Financial Skill
Reusable unit	A query–guidance–answer-contract exemplar.	A natural-language playbook bullet or lesson, optionally accompanied by tags and helpful/harmful counts.	A parameterized skill $s = \langle C_s, P_s, R_s, O_s \rangle$ with typed intent, input schema, evidence and tool dependencies, repair policy, and success conditions.
Construction	A pre-specified, immutable demonstration is itself a memory item.	One interaction can add or revise a textual item through the method-specific reflection rule.	A candidate is induced from externally verified prefix trajectories and is promoted only after distinct-query support, grounding, and leakage checks.
Retrieval	Lexical overlap with the query, with a role-match bonus.	Lexical overlap combined with item-level helpful/harmful statistics.	Lexical retrieval and a deterministic screen of candidates. Full semantic, dependency, and capability compatibility is required before a skill enters planning.
Runtime effect	The exemplar conditions are generated through the prompt.	The selected text conditions planning or reflection through the prompt.	The skill supplies required evidence slots, tool and window preferences, dependency structure, repair hints, and a completion predicate to the execution workflow.
Factual boundary	The exemplar is process guidance and is not admitted as current evidence.	The bullet or lesson is process guidance and is not admitted as current evidence.	Historical entities, dates, and values are removed from the reusable payload. Every current financial claim must still be supported by evidence admitted to $\mathbb{L}^{(k)}$.
Evaluation state	Fixed demonstrations.	Memory can evolve at the granularity of an individual textual item.	The registry is fixed within each task and grows only between tasks. Its transferred trading snapshot remains frozen throughout evaluation.

Therefore, the central distinction is not textual detail, but operational semantics. While demonstrations can state useful heuristics, their applicability and completion are interpreted by the language model. A financial skill exposes these decisions to the surrounding execution environment. The task condition filters out superficially similar but incompatible queries. The evidence slots specify what information must be acquired, while the completion condition checks whether the current ledger satisfies the required evidence. In addition, this architecture decouples procedural retrieval from factual grounding. While the retrieved skill supplies a procedural prior for task analysis, the Dynamic Context Orchestrator governs which runtime outputs are admitted as valid evidence.

Trace-backed case: exact closing-price retrieval. We use ProFinR query 195, "What was the closing price of Meta Platforms (META) on December 29, 2023?" as a real case. The stored ICL, ACE, and DC runs all returned $353.96. Table 10 reports the top-ranked memory object actually retrieved by each baseline and the corresponding validated financial skill extracted from five distinct price-retrieval trajectories.

This case demonstrates that the learned object is fundamentally a domain-specific financial skill, distinct from a generic procedure. Its reusable content is not the observed answer $353.96, but the financial semantics needed to establish such an answer: price adjustment, trading-date identity, evidence provenance, and permitted substitution. Consequently, surface similarity cannot by itself authorize reuse, and successful tool execution cannot by itself establish completion. The skill does not guarantee a correct answer: correctness still depends on an accurate task contract, available tools, and admissible current evidence. Its advantage is that the conditions most likely to produce financial semantic errors are represented as inspectable execution constraints instead of remaining solely as natural-language advice.

Table 10: Trace-backed comparison for ProFinR query 195. Baseline rows present the top-ranked memory retrieved within their respective runtime contexts, whereas the FinHarness row details structured fields from the validated skill registry.

Memory Interface	Actual Retrieved or Extracted Content	Consequence at Reuse Time
ICL exemplar	`icl-market-series`: "Resolve the requested instrument and exact output window first. Use a market-data source, preserve the source and date metadata, and report an explicit data gap instead of inventing missing rows."	The retrieved exemplar is relevant, but it does not encode the price-adjustment convention or the admissible treatment of a non-trading date as typed constraints.
ACE experience	`temporal-and-window-rule-00001`: "For price movement reviews over full calendar years, explicitly define the observation window as the calendar year [...] and separate it from strategy or fiscal year boundaries."	Although the query asks for one date, lexical retrieval selects a year-level rule. The text may still guide temporal reasoning, but it does not specify an exact-date evidence requirement.
DC experience	`cheatsheet-fcf688c8a027f067-02`: "For queries involving entity-specific regulatory exposure across years, structure the evidence chain to include explicit validation of data availability per year and per source [...]."	The selected entry overlaps on entity and temporal language but is unrelated to price retrieval. Because it remains prompt text, no typed task-compatibility gate rejects this item before injection.
FinHarness financial skill	`financial.skill.validated.c21d4e1d12d7346ecf4e` (support = 5): `task_family=market_data_analysis`; slots `market_series` and `exact_price`; metric `price`; tool `market_data_get_exact_price`; `adjustment=adjusted`; `nearest_policy=exact`; success requires source IDs, temporal validation, and complete coverage.	The representation exposes the financial field, date policy, tool capability, and completion test to the workflow. If exact-date evidence is absent, the slot remains unresolved and triggers repair or an explicit gap, precluding any implicit nearby-date substitution.

B Related Work

B.1 Financial Agents

Recent work in financial agents has evolved from domain-specific language models to autonomous systems that use tools, coordinate multiple roles, and interact with financial environments. While early models such as BloombergGPT and FinGPT demonstrated strong financial language capabilities, they primarily focused on static tasks like response and text understanding (Wu et al., 2023b; Yang et al., 2023). TradingGPT and FinMem adapted LLMs for sequential trading by incorporating specialized roles and layered memory architectures (Li et al., 2023b; Yu et al., 2025). CryptoTrade and FinCon further incorporated reflection and experience-based adaptation into financial decision-making (Li et al., 2024; Yu et al., 2024). Recent systems such as TradingAgents, HedgeAgents, and FinTeam orchestrate multiple specialized agents for market analysis, risk control, and broader financial tasks (Xiao et al., 2024; Li et al., 2025b; Wu et al., 2025b). Building on this, FinRobot further expands this paradigm toward general financial analysis through specialized agent workflows (Yang et al., 2024). ProFinAgent advances professional financial reasoning with DAG-guided planning and case-based memory (Huang et al.). In addition, recent work has shifted attention from task completion to reliable financial execution. For example, FinToolBench evaluates real financial tool use, FinanceHarness emphasizes point-in-time financial research, and FinSAgent strengthens evidence-grounded reasoning over financial filings (Lu et al., 2026; Xiao et al., 2026; Chi et al., 2026). As shown in Table 11, existing financial agents provide only partial coverage of adaptive tool planning, information reliability, and experience reuse. FinHarness unifies all three capabilities

Table 11: Comparison of representative financial agents across Task Scope, Tool Planning, Information Reliability, and Experience Reuse.

Framework	Task Scope	Tool Planning	Information Reliability	Experience Reuse
TradingAgents (Xiao et al., 2024)	Trading	×	×	×
FinMem (Yu et al., 2025)	Trading	×	×	✓
FinCon (Yu et al., 2024)	Financial Decision	×	×	✓
HedgeAgents (Li et al., 2025b)	Trading	×	×	×
FinTeam (Wu et al., 2025b)	Financial Report Generation	×	×	×
FinRobot (Yang et al., 2024)	Financial Analysis	✓	✓	×
ProFinAgent (Huang et al.)	Financial Reasoning	✓	×	✓
FinHarness	**General Financial Tasks**	✓	✓	✓

Note. Tool Planning indicates whether the framework explicitly plans tool usage according to task-specific requirements. Information Reliability indicates whether the framework explicitly controls the reliability of information used during financial reasoning. Experience Reuse indicates whether prior execution experience is used to support subsequent tasks.

within a general financial agent framework, enabling task-adaptive execution, verified evidence use, and reliable reuse of validated skills.

B.2 Multi-Agent Frameworks

LLM-based multi-agent systems have advanced from simple role-based communication to structured and adaptive orchestration. CAMEL pioneered role-playing to support autonomous cooperation between language agents (Li et al., 2023a). Based on this foundation, AgentVerse and AutoGen expanded the paradigm by providing reusable frameworks for expert collaboration and flexible inter-agent conversations (Chen et al., 2024; Wu et al., 2023a). However, later work introduced explicit workflow structures to organize agent collaboration as tasks became more complex. To systematically orchestrate agent collaboration, MetaGPT encoded standard operating skills into specialized agent roles, while ChatDev organized software agents through staged communication protocols (Hong et al., 2024; Qian et al., 2024). These systems improved coordination, but most relied on predefined teams or interaction patterns. DyLAN enables dynamic collaboration by activating task-relevant agents and adapting their communication structure during inference (Liu et al., 2023). Flow further represented workflows as graphs and dynamically refined task allocation during execution (Niu et al., 2025). Similar role specialization has also been adopted in financial systems (Xiao et al., 2024; Tang et al., 2025; Xiong et al., 2025a; Li et al., 2025b). Despite this progress, existing frameworks mainly optimize agent composition and workflow execution. They rarely couple task allocation with explicit evidence requirements, executable tool capabilities, and execution-derived gaps, leaving reliable adaptive coordination under evidence constraints largely underexplored.

B.3 Experience Learning

Experience-learning in language agents has transitioned from basic prompt optimization toward dynamic context adaptation and skill acquisition. Chain-of-Thought prompting demonstrated that explicit reasoning demonstrations can substantially improve complex reasoning (Wei et al., 2022). To automate this process, Automatic Prompt Engineer further framed instruction generation as an optimization problem and automatically searched for effective prompts (Zhou et al., 2022). Subsequently, research moved beyond static prompts toward persistent and adaptive context. Reflexion stores verbal feedback from previous failures, while ExpeL extracts reusable insights from accumulated experience (Shinn et al., 2023; Zhao et al., 2024). To systematically manage these growing experience banks, MemGPT introduced explicit memory management for long-term interactions, while Agent Workflow Memory abstracted historical trajectories into reusable workflows (Packer et al., 2023; Wang et al., 2024). ACE further formulates context as an evolving playbook that is continuously refined through execution feedback (Zhang et al., 2026). More recent work has shifted from unstructured textual experience toward structured procedural skills. Voyager compiles and stores reusable executable skills, whereas SKILL-DISCO and SkillMaster automate this process by actively extracting, validating, and refining skills directly from agent execution trajectories (Wang et al., 2023; Guo et al., 2026; Liu et al., 2026a; Yang et al., 2026). Overall, this evolution reflects a shift from refining prompts to dynamically adapting to context, and finally to learning reusable skills. However, most existing methods do not verify whether the underlying experience is trustworthy before it is converted into persistent skills.

Table 12: Backtesting results for all methods over an evaluation period of six months. We highlight the best value for each reported **ARR**, **SR**, and **MDD** metric in bold.

Model	GOOGL	META	MSFT	NVDA	TSLA	Average		
	ARR (↑)	ARR (↑)	ARR (↑)	ARR (↑)	ARR (↑)	ARR (↑)	SR (↑)	MDD (↑)
Market								
B&H	-14.49%	45.64%	36.80%	25.47%	-28.91%	12.90%	0.57	-31.13%
Rule								
MACD	-3.17%	46.82%	-9.58%	-12.89%	22.77%	8.79%	0.44	-21.24%
ZMR	-2.26%	-0.98%	8.53%	35.01%	16.74%	11.41%	0.46	-20.86%
RL								
$\text{FinRL}_{\text{A2C}}$	-21.22%	43.41%	43.15%	37.43%	-35.10%	13.53%	0.51	-33.07%
$\text{FinRL}_{\text{PPO}}$	-19.77%	50.34%	43.91%	18.56%	-31.94%	12.22%	0.51	-33.72%
LM								
Chronos-2	**19.07%**	-12.61%	20.04%	38.19%	-17.66%	9.41%	0.34	-24.34%
LLM								
FinMem	-22.41%	46.25%	40.26%	26.71%	-22.28%	13.71%	0.30	-29.14%
TradingAgents	-14.95%	29.69%	38.62%	-7.83%	36.92%	16.49%	0.50	-21.82%
TradingAgents								
Qwen2.5-3B	1.73%	36.25%	40.89%	-3.28%	-76.98%	-0.28%	-0.13	-20.95%
Qwen2.5-7B	9.33%	28.98%	-4.50%	-17.22%	-9.11%	1.50%	-0.08	**-6.43%**
Qwen3-30B-A3B	-18.09%	1.36%	9.84%	10.22%	-16.51%	-2.64%	0.06	-22.20%
$\text{DeepSeek-V3.1}_{\text{685B}}$	-12.43%	-9.48%	14.13%	-24.02%	0.00%	-6.36%	-0.26	-12.49%
$\text{Kimi-K2}_{\text{IT}}$	-23.40%	-9.52%	12.60%	-8.33%	8.88%	-3.95%	-0.11	-26.62%
GPT-4o-mini	-18.08%	0.73%	16.27%	-5.38%	5.20%	-0.25%	-0.06	-18.28%
GPT-4o	-14.95%	29.69%	38.62%	-7.83%	36.92%	16.49%	0.50	-21.82%
AlphaQuanter w/o RL								
Qwen2.5-3B	3.06%	23.08%	5.10%	-7.43%	-32.21%	-1.68%	0.08	-25.99%
Qwen2.5-7B	-22.42%	35.50%	17.55%	1.47%	-9.63%	4.49%	0.16	-28.96%
Qwen3-30B-A3B	-26.33%	32.86%	37.45%	29.61%	-46.41%	5.44%	0.12	-30.08%
$\text{DeepSeek-V3.1}_{\text{685B}}$	-25.15%	32.49%	25.45%	10.30%	-1.21%	8.38%	0.24	-30.70%
$\text{Kimi-K2}_{\text{IT}}$	-40.48%	25.83%	-3.39%	-3.27%	13.05%	-1.65%	0.15	-25.30%
GPT-4o-mini	-24.02%	44.42%	43.42%	13.61%	-43.71%	6.74%	0.25	-26.78%
GPT-4o	-9.01%	**57.18%**	19.39%	17.60%	-38.04%	9.42%	0.25	-28.27%
AlphaQuanter								
AlphaQuanter-3B	-14.68%	56.15%	9.82%	30.55%	33.33%	23.03%	0.43	-25.16%
AlphaQuanter-7B	-2.52%	41.91%	**47.23%**	45.41%	**42.67%**	**34.94%**	0.65	-24.93%
FinHarness								
FinHarness	-15.14%	18.69%	36.18%	34.73%	40.40%	22.97%	0.63	-15.80%
FinHarnessSkill	4.05%	41.87%	37.98%	**53.31%**	28.18%	33.08%	**0.77**	-17.58%

C Problem Formulation

Financial hallucinations can arise from both model generation errors and failures across the agent workflow. In our evaluation, ProFinAgent shows a maximum hallucination rate of **85.23%** during errors. Accordingly, we decompose the emergence of financial hallucinations into three distinct stages, which we formalize in the following three subsections.

Table 13: Performance Comparison on GOOGL, META, and MSFT.

Model	GOOGL			META			MSFT		
	ARR (↑)	*SR* (↑)	*MDD* (↑)	*ARR* (↑)	*SR* (↑)	*MDD* (↑)	*ARR* (↑)	*SR* (↑)	*MDD* (↑)
Market									
B&H	-14.49%	-0.35	-27.35%	45.64%	1.25	-31.59%	36.80%	1.41	-18.79%
Rule									
MACD	-3.17%	-0.04	-14.14%	46.82%	2.17	-12.51%	-9.58%	-0.49	-19.97%
ZMR	-2.26%	0.01	-18.47%	-0.98%	0.12	-15.19%	8.53%	0.56	-9.59%
RL									
$\text{FinRL}_{\text{A2C}}$	-21.22%	-0.50	-29.80%	43.41%	1.04	-34.14%	43.14%	1.35	-20.55%
$\text{FinRL}_{\text{PPO}}$	-19.77%	-0.43	-29.80%	50.34%	1.15	-34.15%	43.90%	1.36	-20.55%
LM									
Chronos-2	19.07%	0.60	-19.43%	-12.61%	-0.21	-36.59%	20.04%	0.64	-13.73%
LLM									
FinMem	-22.41%	-0.38	-29.80%	46.25%	0.77	-34.14%	40.26%	0.90	-20.55%
TradingAgents	-14.95%	-0.29	-25.93%	29.69%	0.71	-14.05%	38.62%	0.90	-19.83%
TradingAgents									
Qwen2.5-3B	1.73%	0.10	-5.52%	36.25%	0.85	-15.28%	40.89%	1.06	-12.23%
Qwen2.5-7B	9.33%	1.38	-1.40%	28.98%	0.87	-6.54%	-4.50%	-1.05	-2.27%
Qwen3-30B-A3B	-18.09%	-0.46	-26.36%	1.36%	0.29	-16.29%	9.84%	0.42	-15.88%
$\text{DeepSeek-V3.1}_{\text{685B}}$	-12.43%	-0.66	-12.01%	-9.48%	-0.25	-17.18%	14.13%	0.60	-10.09%
$\text{Kimi-K2}_{\text{IT}}$	-23.40%	-1.09	-17.57%	-9.52%	-0.10	-16.12%	12.60%	0.51	-9.11%
GPT-4o-mini	-18.08%	-0.94	-18.86%	0.73%	0.04	-11.11%	16.27%	0.48	-18.52%
GPT-4o	-14.95%	-0.29	-25.93%	29.69%	0.71	-14.05%	38.62%	0.90	-19.83%
AlphaQuanter w/o RL									
Qwen2.5-3B	3.06%	0.07	-18.18%	23.08%	0.52	-24.91%	5.10%	0.14	-14.66%
Qwen2.5-7B	-22.42%	-0.43	-28.59%	35.50%	0.56	-28.49%	17.55%	0.48	-19.60%
Qwen3-30B-A3B	-26.33%	-0.50	-28.39%	32.86%	0.81	-28.18%	37.45%	0.87	-21.15%
$\text{DeepSeek-V3.1}_{\text{685B}}$	-25.15%	-0.47	-29.77%	32.49%	0.61	-34.14%	25.45%	0.64	-19.94%
$\text{Kimi-K2}_{\text{IT}}$	-40.48%	-0.39	-24.67%	25.83%	0.68	-21.65%	-3.39%	-0.03	-19.21%
GPT-4o-mini	-24.02%	-0.56	-23.20%	44.42%	0.97	-23.84%	43.42%	1.10	-12.92%
GPT-4o	-9.01%	-0.12	-19.72%	57.18%	0.99	-25.02%	19.39%	0.53	-23.04%
AlphaQuanter									
AlphaQuanter-3B	-14.68%	-0.29	-25.60%	56.15%	1.08	-23.75%	9.82%	0.30	-21.06%
AlphaQuanter-7B	-2.52%	0.05	-21.37%	41.91%	0.78	-25.65%	47.23%	1.17	-14.85%
FinHarness									
Qwen3-4B-Instruct-2507	-15.14%	-0.41	-19.68%	18.69%	0.61	-20.96%	36.18%	1.29	-3.91%
Qwen2.5-7B	-22.93%	-0.44	-33.03%	14.50%	0.37	-26.46%	-10.82%	-0.26	-28.80%
FinHarnessSkill									
Qwen3-4B-Instruct-2507	4.05%	0.19	-19.90%	41.87%	0.90	-10.02%	37.98%	1.51	-5.73%
Qwen2.5-7B	-9.26%	-0.11	-25.11%	34.41%	0.64	-32.89%	37.78%	0.95	-11.76%

Table 14: Performance Comparison on NVDA, TSLA, and Average.

Model	NVDA			TSLA			Average		
	ARR (↑)	*SR* (↑)	*MDD* (↑)	*ARR* (↑)	*SR* (↑)	*MDD* (↑)	*ARR* (↑)	*SR* (↑)	*MDD* (↑)
Market									
B&H	25.47%	0.74	-33.83%	-28.91%	-0.20	-44.10%	12.90%	0.57	-31.13%
Rule									
MACD	-12.89%	-0.22	-30.76%	22.77%	0.78	-28.83%	8.79%	0.44	-21.24%
ZMR	35.01%	1.03	-16.72%	16.74%	0.59	-44.33%	11.41%	0.46	-20.86%
RL									
$\text{FinRL}_{\text{A2C}}$	37.42%	0.84	-32.68%	-35.10%	-0.19	-48.18%	13.53%	0.51	-33.07%
$\text{FinRL}_{\text{PPO}}$	18.56%	0.58	-35.93%	-31.94%	-0.12	-48.18%	12.22%	0.51	-33.72%
LM									
Chronos-2	38.19%	0.73	-16.32%	-17.66%	-0.04	-35.62%	9.41%	0.34	-24.34%
LLM									
FinMem	26.71%	0.48	-36.89%	-22.28%	-0.27	-24.32%	13.71%	0.30	-29.14%
TradingAgents	-7.83%	0.03	-38.74%	36.92%	1.17	-10.56%	16.49%	0.50	-21.82%
TradingAgents									
Qwen2.5-3B	-3.28%	-0.06	-18.77%	-76.98%	-2.60	-52.95%	-0.28%	-0.13	-20.95%
Qwen2.5-7B	-17.22%	-0.99	-14.12%	-9.11%	-0.59	-7.82%	1.50%	-0.08	-6.43%
Qwen3-30B-A3B	10.22%	0.31	-23.78%	-16.51%	-0.25	-28.71%	-2.64%	0.06	-22.20%
$\text{DeepSeek-V3.1}_{\text{685B}}$	-24.02%	-0.97	-23.18%	0.00%	0.00	0.00%	-6.36%	-0.26	-12.49%
$\text{Kimi-K2}_{\text{IT}}$	-8.33%	-0.28	-18.88%	8.88%	0.40	-71.40%	-3.95%	-0.11	-26.62%
GPT-4o-mini	-5.38%	0.01	-36.61%	5.20%	0.10	-6.30%	-0.25%	-0.06	-18.28%
GPT-4o	-7.83%	0.03	-38.74%	36.92%	1.17	-10.56%	16.49%	0.50	-21.82%
AlphaQuanter w/o RL									
Qwen2.5-3B	-7.43%	0.14	-34.63%	-32.21%	-0.46	-37.59%	-1.68%	0.08	-25.99%
Qwen2.5-7B	1.47%	0.22	-40.24%	-9.63%	-0.04	-27.88%	4.49%	0.16	-28.96%
Qwen3-30B-A3B	29.61%	0.51	-33.48%	-46.41%	-1.08	-39.22%	5.44%	0.12	-30.08%
$\text{DeepSeek-V3.1}_{\text{685B}}$	10.30%	0.31	-39.81%	-1.21%	0.13	-29.82%	8.38%	0.24	-30.70%
$\text{Kimi-K2}_{\text{IT}}$	-3.27%	0.11	-34.92%	13.05%	0.36	-26.05%	-1.65%	0.15	-25.30%
GPT-4o-mini	13.61%	0.35	-37.60%	-43.71%	-0.59	-36.32%	6.74%	0.25	-26.78%
GPT-4o	17.60%	0.39	-38.53%	-38.04%	-0.54	-35.06%	9.42%	0.25	-28.27%
AlphaQuanter									
AlphaQuanter-3B	30.55%	0.51	-29.04%	33.33%	0.57	-26.34%	23.03%	0.43	-25.16%
AlphaQuanter-7B	45.41%	0.66	-34.91%	42.67%	0.58	-27.88%	34.94%	0.65	-24.93%
FinHarness									
Qwen3-4B-Instruct-2507	34.73%	0.59	-25.68%	40.40%	1.1	-8.77%	22.97%	0.63	-15.80%
Qwen2.5-7B	23.28%	0.47	-38.84%	26.40%	0.58	-35.27%	6.09%	0.12	-32.48%
FinHarnessSkill									
Qwen3-4B-Instruct-2507	53.31%	0.78	-23.72%	28.18%	0.51	-28.54%	33.08%	0.77	-17.58%
Qwen2.5-7B	27.99%	0.50	-36.88%	13.84%	0.355	-38.82%	20.95%	0.48	-29.09%

We formulate financial agent execution as completing a financial task $\mathcal{T}$ with access to a set of financial tools $\mathcal{U} = \{u_1, \ldots, u_K\}$. We represent the requirements of $\mathcal{T}$ using a task contract

$$\mathcal{C} = (\mathcal{X}, \mathcal{W}, \mathcal{S}), \tag{10}$$

where $\mathcal{X}$ specifies the task-specific input constraints, $\mathcal{W}$ specifies the temporal constraints, and $\mathcal{S} = \{\rho_1, \ldots, \rho_m\}$ specifies the required evidence. To complete $\mathcal{T}$, the agent constructs an execution graph $G^{(k)}$ over the available tools in $\mathcal{U}$, maintains a verified evidence set $\mathbb{L}^{(k)}$, and produces the task outcome Y.

C.1 Evidence Misalignment in Financial Task Execution

Fixed role-based workflows can induce financial hallucinations when applied across tasks with different evidence requirements. Unnecessary agent activations and task-irrelevant context can propagate errors throughout the execution trajectory.

Consider two financial tasks with contracts $\mathcal{C}_i$ and $\mathcal{C}_j$. A fixed-role workflow applies the same execution graph even when their required evidence differs:

$$G(\mathcal{C}_i) = G(\mathcal{C}_j) = G_{\text{fixed}}, \qquad \mathcal{S}_i \neq \mathcal{S}_j. \tag{11}$$

This task-independent allocation creates a mismatch between execution and evidence requirements. For an admitted item e and requirement $\rho \in \mathcal{S}$, let $V^{\text{ev}}_{\text{slot}}(e, \rho) = 1$ indicate that e supports ρ. The covered and unresolved requirements at round k are

$$\begin{aligned} \text{Cov}(\mathbb{L}^{(k)}, \mathcal{S}) &:= \left\{ \rho \in \mathcal{S} \;\middle|\; \exists e \in \mathbb{L}^{(k)},\ V^{\text{ev}}_{\text{slot}}(e, \rho) = 1 \right\}, \\ \Delta^{(k)} &:= \mathcal{S} \setminus \text{Cov}(\mathbb{L}^{(k)}, \mathcal{S}). \end{aligned} \tag{12}$$

The execution is evidence-misaligned when

$$\Delta^{(H)} \neq \varnothing, \tag{13}$$

indicating that the activated workflow fails to provide all evidence required by the task. This mismatch can leave critical financial evidence unsupported.

C.2 Evidence Contamination in Financial Reasoning

Financial agents aggregate information from heterogeneous sources into a shared reasoning context. In financial tasks, retrieved information may conflict with the required source, temporal, or evidence constraints, allowing unreliable evidence to propagate through subsequent reasoning.

For each candidate evidence item e, we define its admission condition under the task contract $\mathcal{C}$ as

$$\Phi(e \mid \mathcal{C}) = V^{\text{ev}}_{\text{src}}(e) \cdot V^{\text{ev}}_{\text{time}}(e, \mathcal{W}) \cdot V^{\text{ev}}_{\text{fin}}(e, \mathcal{C}) \cdot \mathbf{1}\{\exists \rho \in \mathcal{S} :\ V^{\text{ev}}_{\text{slot}}(e, \rho) = 1\}. \tag{14}$$

The first two terms measure source validity and temporal consistency. The third enforces the field-level financial checks defined in the method, and the indicator requires support for at least one slot. For binary $V^{\text{ev}}_{\text{slot}}$, the indicator equals the maximum in the main text, with both expressions set to zero when $\mathcal{S} = \varnothing$.

Let $\widetilde{\mathbb{L}}^{(k)}$ denote the candidate evidence produced before admission. We define the nonadmitted audit set and its overlap with the admitted ledger as

$$\mathbb{R}^{(k)} = \left\{ e \in \widetilde{\mathbb{L}}^{(k)} \mid \Phi(e \mid \mathcal{C}) = 0 \right\}, \quad \Gamma^{(k)} = \mathbb{R}^{(k)} \cap \mathbb{L}^{(k)}. \tag{15}$$

An internal admission violation occurs when

$$\Gamma^{(k)} \neq \varnothing, \tag{16}$$

indicating that an item failing Φ reached the admitted ledger. Since both sets use the same rule, $\Gamma^{(k)} = \varnothing$ establishes internal admission consistency. Independent expert labels remain necessary to assess errors outside the encoded field-level checks.

C.3 Hallucination Reinforcement through Unreliable Experience Reuse

Financial agents often reuse historical cases or textual experience to guide future tasks. However, stored experience may contain invalid evidence, failed execution patterns, or task-specific decisions that do not transfer to a new financial task.

We represent a completed execution trajectory as

$$\tau = \left\langle \mathcal{C}, \Xi, \widehat{Y} \right\rangle, \tag{17}$$

where Ξ contains the complete execution and audit trace, including the terminal graph $G^{(H)}$ and admitted evidence state $\mathbb{L}^{(H)}$. Its reliability is characterized by

$$Z(\tau) = V_{\text{exec}}^{\text{traj}}(\tau) \cdot V_{\text{src}}^{\text{traj}}(\tau) \cdot V_{\text{time}}^{\text{traj}}(\tau) \cdot V_{\text{cov}}^{\text{traj}}(\tau) \cdot V_{\text{ans}}^{\text{traj}}(\tau), \tag{18}$$

where the five binary terms evaluate execution validity, source reliability, temporal consistency, evidence coverage, and answer correctness.

For a historical trajectory set $\mathcal{D}_\tau$, unreliable experience is defined as

$$\mathcal{D}_\tau^- = \{\tau \in \mathcal{D}_\tau \mid Z(\tau) = 0\}. \tag{19}$$

For task contract $\mathcal{C}$, skill s, and tool universe $\mathcal{U}$, let g_{sem}, g_{dep}, and g_{cap} be binary checks for semantic compatibility, executable dependencies, and required capabilities, respectively. Full procedural compatibility requires

$$\text{Compat}(\mathcal{C}, s; \mathcal{U}) = g_{\text{sem}}\, g_{\text{dep}}\, g_{\text{cap}}, \tag{20}$$

where g_{sem} requires compatible entity types, temporal modes and cutoffs, metric definitions and parameters, units and currencies, and price bases. An explicitly specified and executable conversion may satisfy a unit or price-basis mismatch; lexical similarity or an unspecified required field cannot. The dependency check g_{dep} requires the skill's ordered preconditions and artifact handoffs to be realizable. The capability check g_{cap} requires every necessary capability to be supported by at least one tool available under the current region, runtime, and role restrictions. Alternative tools may satisfy the same requirement, while preferred tools need not all be available. Tool support also requires matching output semantics.

For a historical trajectory τ_i, direct experience reuse is unreliable if the trajectory is ineligible or, when $Z(\tau_i) = 1$, its scrubbed candidate abstraction $\tilde{s}_i = \text{Abs}(\tau_i)$ is incompatible with the new task:

$$Z(\tau_i) = 0 \quad \vee \quad [Z(\tau_i) = 1 \wedge \text{Compat}\left(\mathcal{C}_{\text{new}}, \tilde{s}_i; \mathcal{U}\right) = 0]. \tag{21}$$

Thus, direct experience reuse can transfer unreliable evidence or incompatible execution patterns across tasks, allowing financial hallucinations to accumulate over repeated interactions.

D Detailed Workflow of FinHarness

D.1 Adaptive Evidence Retriever

Task-Guided Evidence Specification. The Adaptive Evidence Retriever derives its search constraints directly from the task contract, independent of any fixed collection of expert roles. The input constraints $\mathcal{X}$ identify the financial entities, instruments, requested quantities, comparison relations, and output form. The temporal constraints $\mathcal{W}$ determine the observation window, reporting period, or decision boundary under which evidence remains relevant. The required evidence $\mathcal{S}$ converts these constraints into explicit obligations that must be supported before the task can be considered complete. This representation separates the semantic goal of the task from the particular agents or tools used to satisfy it.

The Financial Tool Universe $\mathcal{U}$ is viewed as a capability space rather than a flat list of callable functions. Each tool contributes a financial capability, such as market-data acquisition, quantitative transformation, filing retrieval, event discovery, or document-level research. Agent activation is consequently grounded in two conditions: the assigned operation must contribute to at least one requirement in $\mathcal{S}$, and the responsible agent must have access to a compatible capability in $\mathcal{U}$. This prevents workflow structure from being determined solely by a predefined team composition.

Capability-grounded graph construction. The routing policy π translates the current task state into the execution graph $G^{(k)} = (\mathcal{V}^{(k)}, \mathcal{E}^{(k)})$ defined in Eq. 1. A node in $\mathcal{V}^{(k)}$ represents an evidence-producing objective together with the agent and candidate tool-capability set responsible for it. An edge in $\mathcal{E}^{(k)}$ represents an information dependency: it is introduced when a downstream objective requires evidence or an intermediate artifact produced by an upstream objective. Thus, graph edges encode strict evidential necessity, dispensing with unrestricted inter-agent communication.

Operationally, π is implemented by structured inference with the manager LLM followed by deterministic validation. For repair, this composition can be written as

$$\pi(\mathbf{x}^{(k)}, \mathcal{C}, \mathcal{U}) := \mathcal{P}_{\mathrm{DAG}}\Big(\mathrm{Parse}_{\mathrm{schema}}\Big[M_{\mathrm{mgr}}\Big(p_{\mathrm{repair}}(\mathbf{x}^{(k)}, \mathcal{C}, \mathcal{U})\Big)\Big]\Big), \tag{22}$$

where M_{mgr} is the same manager backbone used elsewhere in the workflow, p_{repair} is a structured DAG prompt, and $\mathcal{P}_{\mathrm{DAG}}$ enforces the output schema, valid roles and capabilities, dependency acyclicity, gap identifiers, failed-call exclusions, and remaining budgets. Invalid proposals are retried under the same schema or rejected. Initial planning uses the analogous task-state prompt and validation path. No routing-specific model is trained or fine-tuned.

This distinction determines the execution schedule. Operations without unresolved predecessors form the same execution wave and can proceed in parallel. Operations that consume upstream evidence are delayed until the relevant dependency has completed. For example, retrieving a price series and collecting an independent filing can proceed concurrently, whereas computing a technical indicator from that series follows the corresponding market-data operation. Before execution, the graph is checked for role–capability consistency, valid dependencies, and acyclicity. A proposed operation that cannot be grounded in an available financial capability is retained as an explicit execution gap to prevent its silent substitution with unsupported reasoning.

The phrase "required agents" refers to contract-level relevance, not to an assumed oracle workflow. An agent is activated because its assigned operation supports a requirement in $\mathcal{S}$ or produces an intermediate result needed by another required operation. The resulting graph is compact with respect to the current evidence obligations, while the task contract remains independent of any particular graph realization.

Evidence-conditioned graph revision. Execution does not make $G^{(k)}$ final. After each wave, the outputs are processed by the Dynamic Context Orchestrator and incorporated into $\mathbb{L}^{(k)}$ only when they satisfy the evidence constraints. Adaptive Evidence Retriever evaluates the residual gap $\Delta^{(k)}$ using Eq. 12. The repair state is $\mathbf{x}^{(k)} = (\Delta^{(k)}, \mathcal{A}^{(k)}, B^{(k)})$, where $\mathcal{A}^{(k)}$ records typed failures, attempted call signatures, and available intermediate artifacts, while $B^{(k)}$ records the remaining repair, node, and tool-call budgets. A completed tool invocation does not close a requirement unless its output contributes verified coverage. Conversely, a requirement already supported by $\mathbb{L}^{(k)}$ does not trigger further acquisition merely because additional tools remain available.

When $\Delta^{(k)} \neq \varnothing$, the routing policy constructs $G^{(k+1)}_{\mathrm{repair}}$ according to Eq. 2. The repair graph is conditioned on the unresolved evidence. It may select an alternative evidence source, revise an incompatible acquisition path, reuse a verified upstream artifact, or recompute a downstream quantity. Repair remains bounded: a route that cannot introduce new support for $\Delta^{(k)}$ is not repeatedly expanded. The routing loop terminates when the required evidence is covered, when the remaining gap is not recoverable under $\mathcal{C}$ and $\mathcal{U}$, or when the execution budget is exhausted. Any unresolved requirement is preserved for conditional synthesis.

Adaptive Evidence Retriever workflow

1. Represent the task $\mathcal{T}$ by $\mathcal{C} = (\mathcal{X}, \mathcal{W}, \mathcal{S})$ and identify the usable financial capabilities in $\mathcal{U}$.
2. Construct $G^{(k)} = (\mathcal{V}^{(k)}, \mathcal{E}^{(k)})$ by mapping each evidence requirement in $\mathcal{S}$ to a compatible agent–tool-capability set, with $\mathcal{E}^{(k)}$ encoding only necessary evidence dependencies.
3. Execute independent nodes in parallel and delay dependent nodes until their required upstream evidence becomes available.

4. Send candidate evidence to Dynamic Context Orchestrator, update $\mathbb{L}^{(k)}$, and compute $\Delta^{(k)} = \mathcal{S} \setminus \mathrm{Cov}(\mathbb{L}^{(k)}, \mathcal{S})$.
5. If $\Delta^{(k)} \neq \varnothing$, construct $G^{(k+1)}_{\mathrm{repair}} = \pi(\mathbf{x}^{(k)}, \mathcal{C}, \mathcal{U})$; otherwise, proceed to synthesis.
6. Stop when coverage is complete, no admissible repair remains, or the execution budget is exhausted, while preserving unresolved evidence in $\Delta^{(k)}$.

D.2 Dynamic Context Orchestrator

Evidence as a Structured Reasoning Interface. Dynamic Context Orchestrator defines the boundary between tool execution and financial reasoning. Outputs produced by $G^{(k)}$ first enter the unverified context $\widetilde{\mathbb{L}}^{(k)}$. At this stage, an output is only a candidate evidence item e. The orchestrator associates each candidate with its provenance, temporal scope, supported requirement, financial fields, and producing operation. This common representation makes heterogeneous outputs comparable under the same task contract.

Each candidate is evaluated through $\Phi(e \mid \mathcal{C})$ in Eq. 14. Source verification, represented by $V^{\mathrm{ev}}_{\mathrm{src}}(e)$, establishes usable provenance. Temporal verification, represented by $V^{\mathrm{ev}}_{\mathrm{time}}(e, \mathcal{W})$, checks the observation and reporting boundary. Slot verification records whether e supports a requirement $\rho \in \mathcal{S}$. Financial-field validation, represented by $V^{\mathrm{ev}}_{\mathrm{fin}}(e, \mathcal{C})$, compares the observed fields with the requested instrument, price basis, accounting concept, event role, units, and metric formula. Each applicable rule must pass. A missing required field is recorded as unverified and cannot enter $\mathbb{L}^{(k)}$.

Only candidates satisfying Eq. 4 enter $\mathbb{L}^{(k)}$. Failed and unverified candidates remain in $\mathbb{R}^{(k)}$ for audit. The resulting $\Gamma^{(k)} = \varnothing$ records internal consistency of the admission rule. Each admitted item retains the provenance, temporal scope, slot support, and field-level decisions needed for later citation checks.

Validated Evidence Propagation. The admitted evidence state is organized around task relevance and execution dependencies. A downstream operation receives the admitted information needed by its objective. This preserves the evidence chain required for later synthesis. At the final boundary, factual input is restricted to items in $\mathbb{L}^{(k)}$, while unsupported components remain represented by $\Delta^{(k)}$.

The orchestrator consequently serves two coupled functions. It validates candidate evidence under Φ and provides structured feedback to Adaptive Evidence Retriever. Rejected, missing, or incomplete evidence is translated back into an unresolved requirement. Newly acquired evidence passes through the same admission process, producing a closed loop between acquisition, verification, and coverage.

Dynamic Context Orchestrator workflow

1. Receive the outputs of $G^{(k)}$ as candidate evidence items e in the unverified context $\widetilde{\mathbb{L}}^{(k)}$.
2. Apply $V^{\mathrm{ev}}_{\mathrm{src}}(e)$ to establish whether each item has identifiable and usable provenance.
3. Apply $V^{\mathrm{ev}}_{\mathrm{time}}(e, \mathcal{W})$ to check consistency with the task's observation, reporting, or decision boundary.
4. Apply $V^{\mathrm{ev}}_{\mathrm{slot}}(e, \rho)$ for $\rho \in \mathcal{S}$ to determine the requirements supported by each item.
5. Apply $V^{\mathrm{ev}}_{\mathrm{fin}}(e, \mathcal{C})$ to validate applicable financial fields and mark missing required metadata as unverified.
6. Evaluate $\Phi(e \mid \mathcal{C})$ in Eq. 14 and append only passing items to $\mathbb{L}^{(k)}$.
7. Retain failed and unverified items in $\mathbb{R}^{(k)}$ for audit and return their unsupported requirements through $\Delta^{(k)}$.

D.3 Skill-Based Experience Manager

Causal Online Skill Memory Construction Protocol. We implement skill construction and reuse as a single pass over ProFinR in a precommitted deterministic order. The memory is initially empty.

Each query is answered from an immutable snapshot containing skills induced only from earlier query identities. The current trajectory can update memory after its response is committed and externally verified. The update becomes visible from the next query. Reference answers remain restricted to the external verifier, which emits only an admission decision and audit metadata.

Stage 1. Causal snapshot and response generation. Let $\mathcal{B}$ denote the ordered set of 528 ProFinR queries. Before query q_i is executed, the runner creates a content-addressed snapshot $\mathcal{M}_{i-1}^{\mathrm{snap}}$ from the current registry.

$$\begin{aligned} \mathcal{B} &= (q_1, \ldots, q_{528}), \\ \mathcal{M}_0 &= \varnothing, \\ \mathcal{M}_{i-1}^{\mathrm{snap}} &= \mathrm{Snapshot}(\mathcal{M}_{i-1}), \\ \widehat{Y}_i &= \mathrm{Run}\big(q_i, \mathcal{M}_{i-1}^{\mathrm{snap}}\big). \end{aligned} \tag{23}$$

The snapshot provenance may contain only identities and normalized query hashes associated with q_j for $j < i$. The runner rejects the snapshot if the identity or normalized hash of q_i appears in its provenance. Using Eq. 5 for query i, the complete trajectory $\tau_i = \langle \mathcal{C}_i, \Xi_i, \widehat{Y}_i \rangle$ includes the ordered execution and audit trace. Its final graph $G_i^{(H)}$ and ledger $\mathbb{L}_i^{(H)}$ are terminal states, while earlier calls, admission decisions, and repair transitions remain in Ξ_i.

$$\begin{aligned} \Xi_i &= \big(\xi_i^{(k)}\big)_{k=0}^{H}, \\ \xi_i^{(k)} &= \big(G_i^{(k)}, \Lambda_i^{(k)}, \widetilde{\mathbb{L}}_i^{(k)}, D_i^{(k)}, \mathbb{L}_i^{(k)}, \Delta_i^{(k)}, a_i^{(k)}\big). \end{aligned} \tag{24}$$

Within Ξ_i, $\Lambda_i^{(k)}$ preserves the order and results of role-tool calls, $D_i^{(k)}$ records each candidate's admission decision, supported slots, and rejection reason, and $a_i^{(k)}$ records an executed repair and its outcome. The trace joins separately stored execution and audit events, so the terminal graph and ledger do not replace the preceding rounds.

Stage 2. Prediction commitment and external verification. The runner records an immutable commitment before any benchmark reference is accessed.

$$c_i = \mathrm{Commit}\Big(\mathrm{id}(q_i), h(q_i), h(\mathcal{M}_{i-1}^{\mathrm{snap}}), \widehat{Y}_i\Big). \tag{25}$$

After c_i is persisted, an evaluator outside the agent workflow computes

$$V_{\mathrm{ans}}^{\mathrm{traj}}(\tau_i) = \mathbf{1}\Big\{\mathrm{Verify}_{\mathrm{ext}}(\widehat{Y}_i, Y_i^*) = \textsc{Pass}\Big\}. \tag{26}$$

This decision instantiates the answer-correctness factor of $Z(\tau_i)$ in Eq. 18. The reference Y_i^* is never exposed to the agent, stored in the trajectory, or passed to the skill compiler. The prefix-admitted set after query i is

$$\mathcal{D}_{\tau,i}^{+} = \{\tau_j \,|\, j \le i,\ Z(\tau_j) = 1,\ c_j \text{ is committed}\}. \tag{27}$$

Stage 3. Prefix-bounded candidate abstraction. Let $\sigma(\tau)$ denote the aggregation signature formed from normalized applicability conditions, required slots, metrics, comparison relations, and ordered role-tool dependencies, and let σ_s denote the signature associated with candidate s. Admitted trajectories in the observed prefix form

$$\begin{aligned} \mathcal{A}_s^{(i)} &= \big\{\tau \in \mathcal{D}_{\tau,i}^{+} \,\big|\, \sigma(\tau) = \sigma_s\big\}, \\ n_i(s) &= \Big|\Big\{\mathrm{qid}(\tau) \,\Big|\, \tau \in \mathcal{A}_s^{(i)}\Big\}\Big|, \end{aligned} \tag{28}$$

where $\mathrm{qid}(\tau)$ denotes the identity of the query that generated trajectory τ. Each query identity contributes at most one unit of support. One parameterized record defines the candidate structure, and the remaining distinct earlier trajectories provide support. The compiler removes concrete entities, dates, query text, answers, observed values, and filesystem paths from the runtime payload.

Stage 4. Incremental promotion. A prefix-bounded candidate is promoted only when

$$\mathrm{Promote}_i(s) = \mathbf{1}\left\{\begin{array}{l} n_i(s) \ge 2 \wedge \mathrm{Scrub}(s) = 1 \\ \wedge\, \mathrm{LeakAudit}(s) = 1 \end{array}\right\}. \tag{29}$$

The support threshold requires distinct query identities from the committed prefix. The scrubbing and leakage audits exclude query-specific content from the reusable payload. Promotion cannot expose verifier references, historical answers, or observed values.

Stage 5. Post-query update. Only newly promoted skills are appended after the current query has terminated.

$$\begin{aligned} \Delta\mathcal{M}_i &= \{s \mid \mathrm{Promote}_i(s) = 1,\ s \notin \mathcal{M}_{i-1}\}, \\ \mathcal{M}_i &= \mathrm{Append}(\mathcal{M}_{i-1}, \Delta\mathcal{M}_i). \end{aligned} \tag{30}$$

The update event records the previous registry hash, the query identity, the normalized query hash, the commitment hash, and the new registry hash. It is appended to a hash-chained audit log. Causal visibility is enforced by

$$\begin{aligned} &\mathrm{id}(q_i), h(q_i) \notin \mathrm{Prov}(\mathcal{M}_{i-1}^{\mathrm{snap}}), \\ &\mathcal{M}_i \text{ is visible only to } q_j \text{ with } j > i. \end{aligned} \tag{31}$$

Every query is executed exactly once. Its correctness signal cannot affect its own response, memory snapshot, or execution trace. This protocol evaluates online procedural reuse on queries that are unseen by the skills available at response time.

Verified Trajectory Admission. Skill-Based Experience Manager operates after the execution and verification boundaries. The completed trajectory $\tau = \langle \mathcal{C}, \Xi, \widehat{Y} \rangle$ in Eq. 5 retains the task contract, execution and audit history, and generated answer. Completion alone does not make a trajectory reusable. Its eligibility is determined by the trajectory-level reliability criterion $Z(\tau)$ in Eq. 18, which jointly considers execution validity, source reliability, temporal consistency, evidence coverage, and independently verified answer correctness. Trajectories in $\mathcal{D}_\tau^-$ are excluded from persistent financial skill memory because they contain an unresolved failure in at least one of these dimensions. The admission decision is evaluated only after the current response commitment and can influence only later queries.

This gate changes the unit of memory from a raw interaction history to verified financial experience. The memory does not preserve a trajectory merely because it contains a detailed conversation or fluent final answer. It preserves the parts of the trajectory that explain how evidence requirements were satisfied, which financial dependencies were necessary, and how recoverable gaps were resolved.

From trajectories to reusable skills. Each eligible trajectory is first transformed into a scrubbed candidate abstraction $\tilde{s}_i = \mathrm{Abs}(\tau_i)$. Candidate abstractions with a common aggregation signature are grouped over the committed prefix, from which a parameterized skill $s = \langle C_s, P_s, R_s, O_s \rangle$ is constructed and subsequently promoted. The applicability condition C_s describes the class of financial task contracts for which the skill is suitable. The analysis pattern P_s preserves the ordering and dependency structure of evidence acquisition without binding the skill to historical entities or values. The repair component R_s records how characteristic evidence gaps can be addressed. The completion condition O_s specifies the evidence state that the skill is expected to establish.

This representation separates transferable financial analysis patterns from task-specific facts. Entity names, dates, and observed financial values remain attached to the historical trajectory, while the reusable skill retains the evidence pattern, capability requirements, temporal relationships, and completion logic. Therefore, the abstraction supports transfer across tasks that share an evidential structure without treating surface similarity as sufficient evidence of compatibility.

Fixed representation and induced content. The skill schema, normalization vocabulary, canonicalization rules, and safety checks are fixed system components. They constrain the admissible representation but do not encode a reusable financial strategy. The compiler populates the field values exclusively from recurring structures in prefix-admitted trajectories. Normalized applicability conditions instantiate C_s, supported evidence and analysis dependencies instantiate P_s, observed repair transitions instantiate R_s, and shared completion obligations instantiate O_s. The representation is engineered, while the reusable field values and their support statistics are trajectory-derived.

Registry and information boundary. The runtime loads an immutable content-addressed snapshot for each response. Each skill retains the identities and normalized hashes of its supporting earlier queries together with its promotion decision. The runner verifies the snapshot hash before

and after query execution, rejects schemas that permit in-query mutation, and checks that current-query identifiers are absent from snapshot provenance. Registry updates occur through append-only post-query events whose hash chain supports causal audit.

A financial skill contains no executable code, provider credentials, retrieved documents, historical answers, or observed values. It may describe an admissible evidence pattern, a preferred capability, or a completion condition. It cannot execute a tool, certify a tool output, or establish a financial fact. These responsibilities remain with Adaptive Evidence Retriever, the executable catalog $\mathcal{U}$, and Dynamic Context Orchestrator, respectively.

Trigger construction and compatibility gating. The Skill-Based Experience Manager guides execution structurally without injecting factual biases. For a new task with contract $\mathcal{C}_{\text{new}}$, retrieval first forms a lexical candidate set using trigger terms compiled deterministically from each skill's normalized applicability conditions, required slots, metrics, and comparison relations. Matching is case-normalized, and short financial abbreviations are matched as complete tokens to avoid substring collisions. Each lexical candidate is then evaluated using the full procedural compatibility criterion $\mathrm{Compat}(\mathcal{C}_{\text{new}}, s; \mathcal{U})$ defined in Eq. (20). Specifically, g_{sem} verifies compatibility of entity types, temporal modes and cutoffs, metric definitions and parameters, units and currencies, and price bases. A unit or price-basis mismatch is permitted only when an explicitly specified and executable conversion resolves it, while lexical similarity or an unspecified required field is insufficient for semantic compatibility. The dependency check g_{dep} verifies that the skill's ordered preconditions and artifact handoffs are realizable under the current task. The capability check g_{cap} requires every necessary capability to be supported by at least one tool available under the current region, runtime, and role restrictions, with matching output semantics. Alternative tools may satisfy the same requirement, while preferred tools need not all be available. A candidate is therefore eligible for retrieval only when all three checks pass, such that $\mathrm{Compat}(\mathcal{C}_{\text{new}}, s; \mathcal{U}) = 1$. Let $\ell(q, s)$ count the matched trigger terms between query q and skill s. The retrieved set is

$$\mathcal{R}(q) = \mathrm{TopK}_s \left\{ s \mid \ell(q, s) > 0,\ \mathrm{Compat}(\mathcal{C}_{\text{new}}, s; \mathcal{U}) = 1 \right\}, \tag{32}$$

ranked by $\ell(q, s)$. The implementation retains at most four compatible candidate skills, with score ties broken by deterministic registry order, and does not use embedding-based skill retrieval. The proposed execution graph is subsequently validated against the same current tool availability and role–tool capability constraints before execution. Current-task evidence and the final answer undergo their own validation before completion, while requirements without verified support remain explicit gaps.

Runtime influence and evidence boundary. A retrieved skill is materialized through four bounded channels. First, its required slots extend $\mathcal{S}$ and become explicit obligations in the coverage state. Second, its preferred tool groups and tool names reorder compatible candidates when constructing $G^{(k)}$ without removing fallback tools. Third, the role–tool dependencies, artifact handoffs, temporal windows, and preconditions in P_s are attached only to applicable execution nodes. Fourth, R_s and O_s provide proposals for gap repair and completion checking. Every repair proposal is revalidated against the current gaps, $\mathcal{C}_{\text{new}}$, and $\mathcal{U}$. A declared completion condition is accepted only when supported by current-task coverage, temporal validation, answer validation, and source provenance. Conditions without a corresponding runtime validator constitute mere guidance, failing to qualify as evidence.

A retrieved skill does not override the current task contract. In particular, historical conclusions and values stored with τ cannot enter $\mathbb{L}^{(k)}$ for the new task, and the skill payload itself contributes zero facts to the ledger. All new financial claims must be supported by current evidence admitted through the Dynamic Context Orchestrator.

The same boundary applies when the accumulated registry is transferred to trading. The registry is fixed before the backtest begins. Daily task, universe, and temporal information determine a read-only skill context for planning and risk synthesis, while all prices, filings, indicators, and decisions remain grounded in current tool outputs. No trading outcome updates the registry during evaluation.

Skill-Based Experience Manager workflow

1. Fix the query order before execution and initialize $\mathcal{M}_0 = \varnothing$.
2. Snapshot $\mathcal{M}_{i-1}$ and verify that its provenance contains only earlier query identities before executing q_i.
3. Commit the response and snapshot hash before supplying $V_{\text{ans}}^{\text{traj}}$ through the external verifier.
4. Exclude trajectories in $\mathcal{D}_\tau^-$ and compile eligible prefix trajectories into scrubbed candidates $s = \langle C_s, P_s, R_s, O_s \rangle$ with distinct-query support.
5. Append promoted skills after q_i terminates and expose the resulting $\mathcal{M}_i$ only to later queries.
6. Form lexical candidates from the query and each skill's deterministic trigger terms, retain at most four candidates that pass $\text{Compat}(\mathcal{C}_{\text{new}}, s; \mathcal{U})$, and record rejected candidates for audit.
7. Materialize an admitted skill through required slots, tool-ordering preferences, role–tool dependencies, repair proposals, and completion guidance, while revalidating each decision against the current task state.
8. Treat every retrieved skill only as structural guidance. All current financial claims must still be supported by evidence admitted to $\mathbb{L}^{(k)}$ through Dynamic Context Orchestrator.

D.4 Prompting Frameworks

This section presents the core prompt templates used throughout the FinHarness workflow.

Workflow - Manager Agent Profile

```
You are the manager agent for the A-share, Hong Kong, and US
stock market.

Trading workflow boundary: - Coordinate the specialist team
only: Market Data, Quant / Technical, Fundamental / Event,
Trader, and Risk / Portfolio. - Do not fabricate data, replace
specialists with assumptions, or perform reusable skill mining,
scoring, router training, or policy training.

Available MCP tools: - none directly; you plan subagent tasks,
review tool parameters, and summarize memory.

Core rules: Keep market/symbol/date/horizon/benchmark/
portfolio assumptions consistent; Separate evidence from
interpretation. Surface conflicts; require risk review before
action.

Required JSON output: JSON only, no markdown, no extra
top-level keys.
```

Workflow - Market Data Agent Profile

```
You are the Market Data Agent for the A-share, Hong Kong, and US
stock market.

Trading workflow boundary: - Prepare and validate market data
only; Do not decide trades or size positions. - Never fabricate
prices, volumes, calendars, tradability, or missing rows. - Do
```

```
not perform reusable skill mining, scoring, router training, or
policy training.

   Core checks: Resolve provider symbols; Verify date range,
frequency, benchmark, row counts, missing or duplicate sessions,
stale quotes, zero price/volume, abnormal jumps, suspensions,
ST flags, price limits, IPO age, splits/dividends, lot/liquidity
issues, and downstream data readiness.

   Required JSON output: JSON only, no markdown, no extra
top-level keys.
```

Workflow - Technical Agent Profile

```
   You are the Quant / Technical Analyst Agent for the A-share,
Hong Kong, and US stock market.

   Trading workflow boundary: - Convert validated market data
into technical and quantitative evidence only. - Do not create
orders, approve portfolio risk, invent numeric signals, or use
data not shown by tools/artifacts. - Do not perform reusable
skill mining, scoring, router training, or policy training.

   Core checks: Trend, momentum, reversal, volatility,
liquidity, drawdown, factor rank, benchmark-relative behavior,
cross-sectional strength, lookback/frequency/benchmark,
bullish versus bearish evidence, A-share limit/liquidity
distortions, HK/US gaps, split adjustment, and benchmark
mismatch.

   Required JSON output: JSON only, no markdown, no extra
top-level keys.
```

Workflow - Risk Agent Profile

```
   You are the Risk / Portfolio Agent for equity workflows across
A-shares, Hong Kong stocks, and US stock market.

   Review trade proposals, enforce risk limits, scale or reject
exposure, and produce the final risk-approved action.

   Trading workflow boundary: - You review trading proposals and
portfolio exposure only. - You do not originate the primary alpha
thesis unless asked for a risk-based rejection explanation. - You
do not perform reusable skill mining, reusable skill scoring, or
reusable policy training. - You do not approve trades when risk
evidence is insufficient.

   Risk rules: 1. Check single-name exposure, sector
exposure, market exposure, liquidity, volatility,
drawdown, and correlation. 2. Resize or reject trades
```

```
that violate constraints. 3. Require enough observations
for VaR, volatility, or correlation. 4. Treat A-share
limits/suspensions, HK lot and gap risk, and US earnings or
index-correlation risk explicitly.

  Required JSON output: You must output JSON only, with no
markdown and no extra top-level keys.
```

Workflow - DAG Planning

```
  Query: {query} Structured QuerySpec: {query_spec} Answer
slots: {answer_slots} Runtime data window: {data_window}

  Create a compact DAG wave for the evidence subagents that
should run now. If a role needs evidence produced by another
role, include that upstream node id in dependencies. If a
downstream role needs a file created by an upstream role, do
not invent a filename. Add the upstream node id to dependencies
and describe the need in required_artifacts using from_node,
kind, symbol, and date range. If a node is expected to produce
a reusable file, describe it in produces_artifacts; the DAG
executor will create and validate the actual path.

  Return JSON only, no markdown, no extra keys: [
{{ "id": "market_1", "role": "market_data", "goal":
"concise upstream evidence objective for this query",
"dependencies": [], "produces_artifacts": [ {{"kind":
"ohlcv|filing|statement|web_page|json|csv", "symbol":
"...", "format": "json"}} ], "required_artifacts": [] }},
{{ "id": "technical_1", "role": "quant_technical", "goal":
"concise downstream evidence objective for this query",
"dependencies": ["market_1"], "produces_artifacts": [],
"required_artifacts": [ {{"from_node": "market_1", "kind":
"ohlcv|filing|statement|web_page|json|csv", "symbol": "..."}} ]
}} ]
```

Workflow - Gap Repair Planning

```
  Query: {query} Execution-derived GapSnapshot: {gap_snapshot}
Capability catalog by role (executable candidates for this
query): {capability_catalog}

  Inspect the reports and unresolved gaps from the completed
execution. Decide whether to synthesize, repair, or abort.
Repair only recoverable gaps that are important to the requested
answer. Do not create a general research task. Use at most
one task per role in this repair wave. Never repeat a failed
tool signature unchanged. Permission, authentication,
unsupported-region, and exhausted-budget failures are not
repairable.
```

```
   Return one JSON object and no markdown: {{ "decision":
"synthesize|repair|abort", "reason": "short execution-grounded
reason", "tasks": [ {{ "id": "repair_market_1", "role":
"market_data|quant_technical|fundamental_event",
"objective": "specific repair objective",
"resolves_gap_ids": ["known gap id"], "depends_on": [],
"required_artifacts": [], "evidence_refs": [], "strategy":
"alternative_source|correct_parameters|reuse_artifact|recompute",
"avoid_tool_signatures": [] }} ] }}
```

Workflow - Answer

```
   Query: {query} Admitted claims and evidence:
{admitted_evidence} Execution-derived gap snapshot:
{gap_snapshot}

   You are the only component allowed to write final_answer.
Do not invent facts or silently fill unresolved gaps. Even
when no supplied fact can be verified, final_answer must be a
concise, direct answer stating that the requested conclusion
is unconfirmed and naming the missing evidence in ordinary
answer language. Never return an empty or whitespace-only
final_answer. final_answer is the answer only. Do not append gap
ids, diagnostics, or framework explanations to it. Keep gap ids
and diagnostics in the JSON fields.

   Return one JSON object and no markdown: {{ "status":
"success|partial|failed", "final_answer": "answer grounded in
supplied claims", "claim_support": [], "used_claim_ids": [],
"used_evidence_ids": [], "unresolved_gap_ids": [] }}
```

Workflow - Tool Selection

```
   {context}

   Role: {role} Current query: {task} Manager focus for this
role: {role_focus} Execution round: {tool_round}; remaining
tool-call budget: {remaining_tool_budget} Recoverable observed
tool results: {available_context} Allowed tool groups:
{allowed_tool_groups} Allowed tools: {allowed_tools}

   Select only tools needed for quantitative or technical
evidence. The hard tool-call limit is enforced by code. Do
not exceed the remaining budget. Return JSON only: {{"tools":
[{{"name": "...", "reason": "..."}}]}}
```

Workflow - Tool Parameters

```
   {context}
```

```
  Role: {role} Current query: {task} Manager focus for this
role: {role_focus} Selected tool specs: {selected_tools} Manager
feedback: {manager_feedback} Prior context: {prior_context}
Execution round: {tool_round} Observed tool results:
{observed_tool_results}

  Generate valid arguments using only declared input names. Use
canonical symbols from prior_context.instrument_context for
symbol-bearing inputs. Do not invent mappings or use placeholder
expressions such as {{tool_name.N.field}}. If resolution_status
is unresolved, ambiguous, or source_unavailable, do not make
a provider call with the requested name. Preserve dates,
periods, frequencies, and indicator names. Return JSON only:
{{"tool_calls": [{{"name": "tool_name", "arguments": {}}}]}}
```

Workflow - Tool Result Analysis

```
  {context}

  Role: {role} Current query: {task} Manager focus for this
role: {role_focus} Prior context: {prior_context} Selected
tools: {selected_tools} Approved tool calls: {tool_calls} Tool
results: {tool_results}

  Analyze only quantitative and technical evidence. Prefer the
selected semantic retrieval fragments in prior context and cite
their fragment_ids. Every claim must cite observed source_ids.
If an input series, lookback, or data window is missing, return
partial or failed and explain the missing evidence in failures.

  Return exactly one JSON object: {{ "role": "quant_technical",
"status": "success|partial|failed", "claims": [{{"claim_id":
"quant_technical:claim:1", "text": "...", "source_ids":
[], "fragment_ids": [], "confidence": 0.0}}], "evidence":
[{{"source_id": "tool_name:0", "summary": "...", "fragment_ids":
[], "as_of_date": ""}}], "failures": [] }}
```