REALFIN: Benchmarking Agent Workflow Orchestration Beyond MCP-Style Tool Connectivity

Anonymous ACL submission

Abstract

Standardized interfaces such as the Model Context Protocol (MCP) make it easier to connect tools to agents, but connection alone is not orchestration. When many related tools are exposed as a flat callable palette, agents must identify relevant tools, resolve substitutes, pass intermediate outputs, and decide when a workflow is complete. We introduce REALFIN, a finance-domain diagnostic benchmark for evaluating workflow construction under MCP-style flat tool exposure. REALFIN contains 1,524 tasks generated from 108 author-designed financial workflow templates over real market, macroeconomic, and company-data APIs. Each task is paired with an executable reference program that specifies one verified solution path, while the 85 agent-callable tools include explicit alternative, dependency, and related-tool metadata to represent non-unique paths, prerequisite tools, and near-neighbor ambiguity. This design supports path-independent answer scoring together with process-level evaluation of tool selection, ordering, parameter planning, entity grounding, temporal grounding, orchestration success, and error recovery. Across ReAct-style flat exposure, retrieval-based exposure, oracle-controlled variants, and progressive tool disclosure, current agents often achieve nonzero answer accuracy while failing to construct reliable workflows. These results suggest that MCP-style tool connectivity needs structured disclosure and workflow-aware evaluation, especially in finance where temporal constraints and overlapping tools are intrinsic.

1 Introduction

Large language models are increasingly acting as agents that plan actions, call tools and operate within external environments (Yao et al., 2023; Shinn et al., 2023; Wang et al., 2024). Standardized interfaces such as the Model Context Protocol (MCP) expose callable tools through discoverable schemas (Model Context Protocol, 2025), but answering real tasks still requires constructing multi-step workflows.

We argue that the flat tool-palette assumption has reached a dead end. Under this assumption, a system exposes many callable tools and expects the model to infer the right subset and workflow from names, descriptions and schemas alone. This interface is convenient for connection, but brittle when tool functions overlap. Multiple plausible solution paths may coexist, forcing agents to select tools, order them, and compose outputs into coherent workflows. We refer to this setting as tool explosion (Qin et al., 2024a,b; Patil et al., 2024).

The problem is especially acute in finance, where analysis requires temporal grounding, heterogeneous indicators, and sequential reasoning over market, company and macroeconomic signals. Recent financial tool-use benchmarks already contain hundreds of executable tools or many MCP-based financial servers (Lu et al., 2026; Zhu et al., 2026), and tool retrieval itself remains a difficult bottleneck (Shi et al., 2025). Simply flattening more tools into context therefore changes the task into retrieval, disambiguation, planning and execution under ambiguity.

Existing benchmarks often emphasize static question answering (Chen et al., 2021, 2022; Zhu et al., 2021; Deng et al., 2022; Guo et al., 2025), simplified tool environments (Qin et al., 2024b; Liu et al., 2024), or short workflows, leaving limited visibility into how agents choose tools and execute workflows in dense tool ecosystems.

A natural direction is therefore to evaluate agents through the processes they execute: tool selection, temporal grounding, intermediate coordination, and workflow completion. Such traces reveal when flat MCP-style exposure becomes insufficient under redundant APIs, alternative valid paths and temporal dependencies.

We propose REALFIN, a finance-domain benchmark built from authentic multi-step financial anal-

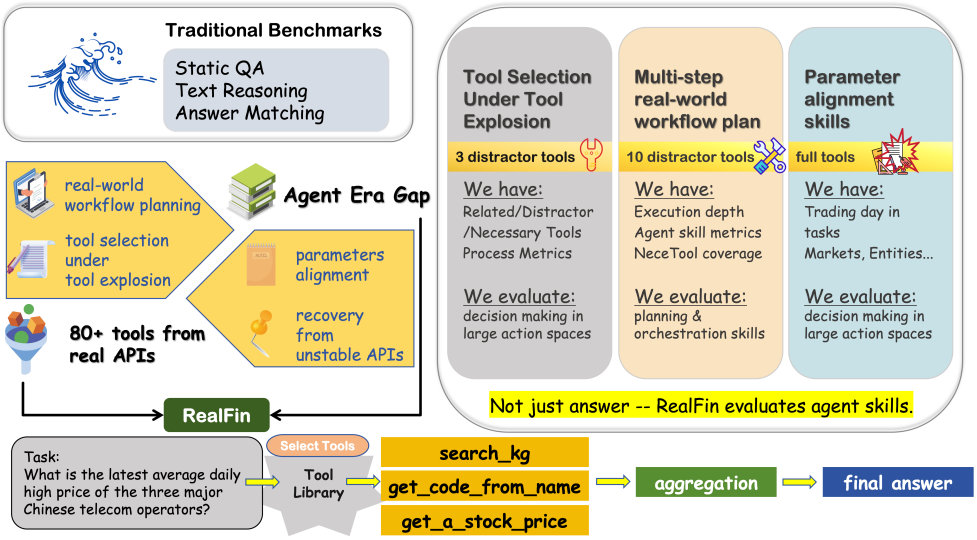

Figure 1: Overall concept of REALFIN: workflow-grounded financial tasks, executable golden code, and a realistic multi-tool environment where flat tool exposure leads to tool explosion.

ysis workflows. Each task is paired with an executable reference program that specifies one valid solution path while permitting alternative workflows. REALFIN uses realistic financial APIs and records execution traces, enabling joint study of outcomes and processes under natural tool abundance.

Our contributions are threefold:

- We introduce REALFIN, a workflow-grounded benchmark that frames tasks as complete financial analysis workflows rather than static QA.
- We promote process-based evaluation, assessing models through tool-use behavior rather than only final answers under tool explosion.
- We formulate MCP-style flat tool exposure as a central challenge for LLM agents in financial workflows, where tool redundancy, temporal sensitivity, and workflow ambiguity are intrinsic.

2 Related Work

Existing financial benchmarks mainly evaluate static question answering, document-grounded analysis, and numerical reasoning rather than executable workflows. FinQA, ConvFinQA, TAT-QA, FinEval, FinanceBench, and FinBen (Chen et al., 2021, 2022; Zhu et al., 2021; Guo et al., 2025; Islam et al., 2023; Xie et al., 2024) are valuable for financial knowledge and reasoning, but most are answer-centered and do not require agents to interact with real financial APIs. Recent financial-agent benchmarks such as InvestorBench, FinAgentBench, FinToolBench, FinMCP-Bench, and FinSearchComp (Li et al., 2025; Choi et al., 2025; Lu et al., 2026; Zhu et al., 2026; Hu et al., 2025) move closer to realistic financial agency. REALFIN is complementary: it focuses on workflow-level failures that emerge when financial tools are numerous, overlapping, temporally sensitive, and partially substitutable.

General tool-use benchmarks study API invocation, function calling, and environment interaction across broader domains (Qin et al., 2024a; Li et al., 2023; Zhou et al., 2024; Liu et al., 2024; Mialon et al., 2024; Patil et al., 2025; Wang et al., 2025a). Related agent frameworks such as ReAct and Reflexion interleave reasoning with actions, while Gorilla studies API invocation at scale (Yao et al., 2023; Shinn et al., 2023; Patil et al., 2024). These works show that tool use should be evaluated beyond final answers. Our novelty is narrower: executable, process-aware evaluation of finance-domain workflows under MCP-style flat exposure and dense, redundant tool spaces.

Tool retrieval work further motivates this focus. ToolRet shows that conventional retrieval models can struggle to retrieve appropriate tools from heterogeneous tool collections (Shi et al., 2025), while MCP-Bench shows that standardized protocols improve interoperability but do not by themselves solve tool selection and workflow construc-

Benchmark	Tool Selection	Multi-Tool Workflow	Real-API Robust-ness	Temporal Reasoning	Alt. Tool Paths	Tool Propen-sity
FinQA	✗	✗	✗	✗	✗	✗
ConvFinQA	✗	✗	✗	✗	✗	✗
FinanceBench	✗	✗	✗	△	✗	✗
FinBen	△	△	✗	△	✗	✗
FinSearchComp	✓	△	△	✓	✗	△
ToolBench	✓	✓	△	✗	✗	✓
GAIA	✓	△	△	△	✗	△
BFCL	✓	△	△	✗	✗	✓
RealFin (Ours)	✓	✓	✓	✓	✓	✓

Table 1: Benchmark coverage across capability dimensions. Symbols: ✓ = supported, ✗ = not supported, △ = partially or indirectly supported. Dimensions: Tool Selection = choosing correct tools from large catalogs; Multi-Tool Workflow = composing multi-step domain workflows; Real-API Robustness = robustness to unstable real-world APIs and execution noise; Temporal Reasoning = grounding to trading calendars and temporal anchors; Alt. Tool Paths = support for multiple valid executable tool workflows; Tool Propensity = tendency to invoke tools instead of zero-tool answering.

tion (Wang et al., 2025b). REALFIN follows this motivation in a high-stakes financial setting where redundancy, temporal grounding, and alternative valid workflows are intrinsic rather than artificially injected.

3 Benchmark Design

3.1 Benchmark Design: Stress-Testing Flat Tool Exposure

Workflow-grounded design beyond flat tool lists REALFIN tests whether an agent can recover valid workflows when many financial tools are exposed as a flat callable list. We construct tasks from full analyst workflows, decomposing each into intents, intermediate variables, temporal dependencies, and tool-level operations.

Template-based generation with workflow constraints We build REALFIN from author-designed workflow templates. Each template encodes intent, inputs, temporal assumptions, validity constraints, and an executable tool skeleton. Instantiating entities, markets, dates, indicators, windows, and numerical conditions yields 1,524 instances from 108 templates; the largest template accounts for 14.0%. Tasks run in a real API environment with overlapping and substitutable tools, so multiple valid tool chains may solve the same task.

From flat MCP-style exposure to redundant tool ecology We construct a multi-tool environment from real financial APIs (Akshare (King, 2022)) and preserve functional overlap, near-duplicate interfaces, and substitutable tools. This mirrors MCP-style flat exposure: many tools share one interface, while interchangeability, dependencies, and unsafe substitutes remain under-specified. The golden workflow gives one valid path, not the only path.

Executable alignment of tasks and workflows Each task is paired with an executable golden program covering entity resolution, tool selection, parameter planning, temporal alignment, computation, and aggregation. Non-executable, unstable, or ambiguous samples are removed, enabling process-level diagnosis without treating the golden path as the only valid path.

3.2 Benchmark Structure and Diagnostic Properties

Figure 2 shows how task-side workflow information is paired with tool metadata. Each instance records the final answer, executable workflow structure, and tool annotations; Appendix D.11 gives a concrete instance with alternative-path scoring.

Temporal structure Tasks are categorized as Fixed, Relative, or Realtime, covering historical dates, trading-calendar references, and live or near-

Table 2: Benchmark comparison. RealFin denotes the full benchmark, while RealFin-Deep denotes the subset of tasks whose reference workflows require longer executable tool chains. The comparison highlights whether each benchmark evaluates agents beyond flat tool invocation, including live execution, temporal sensitivity, and multi-step workflow depth.

Benchmark	Domain	#Tasks	#Tools/APIs	Avg Tool Calls	Avg Exec Depth	Exec	Time-Sens.	Answer type
RealFin	Finance (tool)	1524	85	3.78	2.12	✓	✓	Fast Live exec
RealFin-Deep	Finance (tool)	482	85	6.10	2.10	✓	✓	Live exec
API-Bank	Tool-aug LLM	274	53	2.07	≈1.50	✓	×	Simu APIs
BFCL	General	5551	/	≈2.55	≈1.50	✓	×	AST
ToolBench	General	126,486	16,464	3.71	≈2.00	✓	×	Static
PACIFIC	Fin conv-QA	2757	Document	0.00	≈1.00	×	×	Static
WebArena	Web agent	812	Web	≈2.8	≈2.80	✓	✓	Rule-based
ConvFinQA	Fin num-reason	14115	computation	≈1.00	≈1.50	✓	×	Static
FinQA	Fin num-reason	8281	computation	≈2.00	≈1.50	✓	×	Static
FinSearchComp	Fin QA	635	search engine	1–2	1–2	×	✓	LLM-Judge

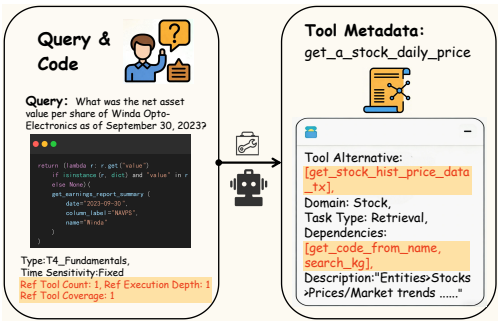

Figure 2: Data and tool structure in REALFIN. The left side illustrates the task-side representation including query text and executable code, while the right side shows the structured tool metadata such as domain, task type, dependencies, and alternatives.

Table 3: **REALFIN statistics.** Counts are computed from the released 1,524-instance dataset.

Instances / templates	1,524 / 108
Task types	T1 412 (27.0%); T2 107 (7.0%); T3 677 (44.4%); T4 328 (21.5%)
Temporal regimes	Fixed 794 (52.1%); Relative 629 (41.3%); Realtime 101 (6.6%)
Reference tools	avg 3.78; median 3; max 24
Execution depth	avg 2.12; median 2; max 3
Agent-visible tools	85 callable tools in Full exposure
Implementation entries	133 backing entries: 65 external-API wrappers, 68 virtual or algorithmic helpers, 66 unique Akshare endpoints

live values. The same tool can be valid or invalid depending on calendars, reporting dates, lookback windows, and live-vs-historical intent.

Tools and multi-path structure REALFIN exposes 85 agent-callable financial tools in Full exposure; this is the tool-library size used in all Full-setting experiments and process metrics. The 133 count in Appendix B is not an additional agent-visible palette, but the lower-level implementation inventory backing those 85 tools: 65 external-API wrappers, 68 virtual or algorithmic helpers, and 66 unique Akshare endpoints. Each callable tool is annotated with `tool_alternative`, `tool_dependencies`, and `tool_related` metadata. These fields make non-unique solution paths explicit: a workflow may replace a tool with an annotated equivalent, depend on prerequisite grounding or retrieval tools, or branch through related near-neighbor tools (Appendix B). Tools may differ by market scope, time granularity, indicator definition, parameter convention, or returned schema, so agents must select, disambiguate, transform, and orchestrate rather than merely answer.

Table 2 summarizes how REALFIN combines real APIs, temporal sensitivity, multi-tool redundancy, and executable workflows. We further compare golden workflows with real analyst SOPs in Appendix E.

Benchmark scope REALFIN is a diagnostic benchmark for studying tool explosion, not a new routing architecture. It asks whether a model given many plausible financial tools can still recover the right workflow; hierarchical packaging is only a lightweight structural aid.

Scope beyond finance REALFIN is intentionally a finance benchmark. Finance provides redundant tools, partial substitutability, schema variation, and temporal workflows with executable references. Cross-domain transfer remains future work.

3.3 Quality Control and Process-Aware Evaluation

We use quality control and process-aware evaluation to secure executable workflows and interpretable outcomes.

Data quality control We filter ambiguous entities, invalid market codes, non-trading days, in-

sufficient history, missing API returns, and non-deterministic answers. Failing or empty instances are repaired or removed (Appendix A). Golden programs are rerun to monitor drift; execution success exceeds 97%, so agent failures mostly reflect workflow, tool-selection, or temporal-grounding errors rather than dataset noise.

Evaluation protocol Beyond answer accuracy, we compute trace metrics for entity binding, temporal grounding, tool selection, ordering, parameter planning, and orchestration. We report both all traces and fully executable traces. REALFIN supports snapshot and live evaluation; released artifacts include dataset records, tool schemas, reference programs, prompts, parsing rules, retry policies, and API snapshots (Appendix C).

3.4 Human Validation of Workflow and Tool Quality

Following FinSearchComp (Hu et al., 2025), we validate question, answer, workflow, and tool quality with human review. The questionnaire is in Appendix G.

We sample 60 tasks across workflow depths and task types, each reviewed by two annotators. Twelve annotators with finance or CS backgrounds score question clarity, answer correctness, workflow correctness, and tool usability on a 0–1–2 scale. Raw agreement is 0.73–0.90 and weighted κ is 0.91–0.97 (Figure 17, Appendix F.1).

4 Experiments

This section tests whether flat tool exposure remains a sufficient interface for reliable agentic workflows. Full, BM25, Oracle, and OraA-k share a ReAct-style loop and vary only the visible tools; package routing instead uses progressive disclosure, first selecting a coarse package and then concrete tools. This separates model behavior under a fixed harness from failures induced by the flat exposure protocol.

4.1 Experimental Setup

Evaluation subsets. We evaluate models on a 128-task diagnostic core because the full matrix requires live API calls and repeated proprietary-model runs. The core covers all task types and temporal regimes (T1/T2/T3/T4 = 45/24/49/10; Fixed/Relative/Realtime = 51/41/36), preserves reference tool-count and execution-depth structure (avg tools 4.00 vs. 3.78; avg depth 1.93 vs. 2.12;

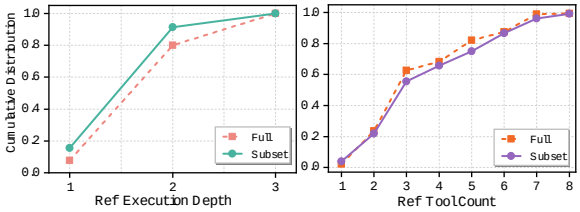

Figure 3: **Structural similarity between the full benchmark and the 128-core subset.**

KS $p = 0.548, 0.088$), and intentionally oversamples rare Realtime and Compare cases to stress temporal grounding and tool ambiguity. We therefore treat model results as diagnostic rather than a full-benchmark leaderboard; the full 1,524 tasks are released for dataset-level analysis and future full-scale evaluation.

Agent harnesses and tool exposure regimes. In the ReAct-style harness, the model observes the task, visible tool descriptions, and previous outputs, then calls a tool or returns an answer. Full, BM25, Oracle, and OraA-k share this loop and differ only in the exposed candidate tools.

(1) ReAct-style flat and retrieved exposure. *Full* exposes all tools and approximates a flat MCP-style palette: all callable tools are available, but the interface gives little structure about interchangeability, dependencies, or near-miss substitutes. *BM25* exposes only the Top-20 tools retrieved by BM25, simulating a shallow retrieval layer placed in front of the flat tool space. Comparing Full with BM25 tests whether reducing the visible action space mitigates the failure of flat exposure.

(2) ReAct-style controlled exposure. *(i) Oracle (Ora).* Expose only tools used in the reference workflow, largely removing tool explosion and isolating execution, parameter planning, and temporal grounding. *(ii) OraA-k.* Extend Oracle with functionally similar alternatives and k distractors while keeping task semantics fixed. OraA-3 and OraA-10 test whether degradation comes from tool-space structure rather than task difficulty.

(3) Progressive-disclosure agent. *Pkg* changes the protocol: the agent selects a package-level intent before seeing concrete tools. It keeps tool availability high while replacing the one-shot flat palette with a hierarchy.

(4) Unfair oracle-guided setting. The model receives oracle trajectory hints, including the correct tool set and solution path but not the final answer. This unfair setting tests task solvability.

4.2 Metrics

Outcome metrics ask whether the final answer is correct or tool calls execute. Process metrics inspect whether the trace selects admissible tools, orders them coherently, grounds entities and time, fills parameters, completes the workflow, and recovers from errors. This matters because correct answers can arise from shortcuts or partial execution. Formal definitions are in Appendix D.

Scoring alternative workflows. The executable reference gives one verified path, not the only correct path. For each task, equivalent tools are annotated by workflow role. A substitute receives process credit only when it fills the same semantic role and produces the required intermediate object under the same entity, date, field, and adjustment constraints. ACC is path independent and checks the final answer; TSR and OA map valid substitutes to reference roles before scoring; EGA, TGA, and PPS check grounded entities, temporal anchors, and parameters against role requirements rather than exact API names. Thus a non-gold but valid chain is not penalized merely for using a different tool. Appendix D.11 gives a concrete scored example.

4.3 Main Results: Flat Exposure Breaks Down

Table 6 reports performance on REALFIN-128. Under a fixed ReAct-style harness, Full, BM25, Oracle, and OraA-k isolate the effect of the visible tool space. Package routing then tests whether progressive disclosure can reduce the decision burden.

Flat exposure causes systematic workflow collapse. In Full, most models have near-zero orchestration success even when accuracy is nonzero. PPS and RSS also remain low, showing that agents struggle to plan parameters and recover from wrong tool paths. Full is therefore not a stronger Oracle with more tools, but a harder interface regime.

Tool redundancy, not task difficulty, drives degradation. We keep tasks fixed while adding related alternatives and distractors. Figure 4 shows that ACC, OSR, and PPS decline from Oracle to OraA-3, OraA-10, and Full, supporting the claim that flat palettes fail when many plausible but non-equivalent tools are exposed together.

Lightweight structure helps, but does not solve the problem. BM25 and package routing improve over Full through different mechanisms: BM25 reduces the visible action space, while package routing changes the harness to progressive disclosure. Both help several models, but OSR and RSS remain low, so shallow retrieval or coarse packaging only mitigates flat-exposure failure.

OraA-k can be worse than BM25 or Full because its distractors are plausible near-neighbors; mistakes are harder to detect and often survive execution.

Answer accuracy overestimates agent capability. ACC and OSR are often decoupled: ACC may reach 20–40% while OSR remains near zero. Correct answers can therefore hide guessing, partial execution, or shortcuts, making process metrics necessary for MCP-style evaluation.

Table 4: Evaluation metrics used in REALFIN. Process metrics are computed from tool-use traces; equivalent alternative tools are treated as valid where annotated.

Metric	What it measures
ACC	Final answer matches the reference answer. This is answer-only and does not require the trajectory to be correct.
ESR	All emitted tool calls return non-empty, non-error results. It measures execution reliability, not semantic correctness.
TSR	Recall of required tools, allowing annotated equivalent alternatives. It asks whether the agent found the right tool family.
OA	Whether the tool-call order is compatible with the reference dependency order, after accounting for valid substitutes.
EGA	Whether entities such as companies, tickers, indices, reports, or commodities are grounded correctly.
TGA	Whether dates, reporting periods, previous trading days, windows, or realtime anchors are grounded correctly.
PPS	Whether required tool parameters are planned correctly, including codes, dates, indicators, windows, and transformations.
OSR	End-to-end process success: tool selection, ordering, parameters, grounding, and execution jointly satisfy the workflow.
RSS	Whether the agent recovers after invalid calls, empty results, or wrong intermediate steps instead of continuing a bad path.

Table 5: Representative Claude-4.5 process metrics. Tool calls may execute while selection, parameter planning, and orchestration remain weak.

Setting	ACC	ESR	TSR	OA	PPS	OSR
Full	0.094	0.664	0.086	0.086	0.008	0.000
BM25	0.250	0.203	0.719	0.602	0.297	0.141
OraA-3	0.367	0.820	0.656	0.609	0.484	0.141

Table 6: Model performance across exposure and harness regimes. Full, BM25, Oracle, and OraA use the same ReAct-style sequential tool-use harness while varying the visible candidate tools; package routing uses a progressive-disclosure harness that reveals tools through package-level routing. ACC = outcome accuracy; OSR = orchestration success; PPS = parameter-planning success; RSS = recovery success. Model versions: Claude (claude-4.5-sonnet), GPT (gpt-5.1), Gemini (gemini-3-pro), Qwen (qwen-max), DeepSeek (deepseek-v3.2), Doubao (doubao-seed-1.6), Kimi (kimi-k2).

	Baseline								Constrained tool spaces												Package-route			
Model	Full*				BM25*				Oracle				OraA-3				OraA-10				pkg			
	ACC	OSR	PPS	RSS	ACC	OSR	PPS	RSS	ACC	OSR	PPS	RSS	ACC	OSR	PPS	RSS	ACC	OSR	PPS	RSS	ACC	OSR	PPS	RSS
Cluster A																								
Claude	9.4	0.0	0.8	8.5	25.0	14.1	29.7	36.6	27.3	3.1	13.3	9.9	36.7	14.1	48.4	17.9	26.6	1.6	13.3	9.8	23.4	17.2	18.0	8.3
GPT	1.6	0.0	0.8	9.4	6.4	4.7	11.7	26.0	10.9	2.3	7.8	5.8	8.6	11.7	35.2	15.4	4.3	0.0	2.9	11.1	30.5	15.6	18.0	6.6
Gemini	8.6	0.0	0.8	8.5	10.0	5.5	7.8	13.8	14.8	2.3	12.5	9.1	9.4	0.8	9.4	11.4	7.0	1.6	1.6	4.9	38.3	12.5	17.2	10.7
Cluster B																								
Qwen	3.1	0.0	2.3	6.7	9.6	2.3	6.2	15.4	16.4	1.6	14.1	9.1	14.1	10.9	35.2	13.8	4.7	0.0	3.1	5.7	23.4	13.3	14.8	10.7
DeepSeek	30.5	0.0	8.6	11.6	9.4	18.0	26.6	35.8	32.0	5.5	13.3	8.3	3.9	0.0	0.0	36.4	0.8	0.0	0.0	30.2	19.5	13.3	14.1	9.9
Doubao	33.6	0.8	16.4	6.6	11.7	7.8	23.4	37.4	36.7	4.7	19.5	9.1	0.0	0.0	0.0	31.4	0.0	0.0	0.0	26.4	42.2	11.7	14.8	11.6
Kimi	10.2	0.8	7.8	7.4	4.7	13.3	21.1	30.9	30.5	0.8	14.1	8.3	1.6	0.0	0.0	17.4	0.0	0.0	0.0	15.7	14.1	2.3	4.7	9.1
Unfair oracle-guided setting's ACC $\approx$ 83.59%																								

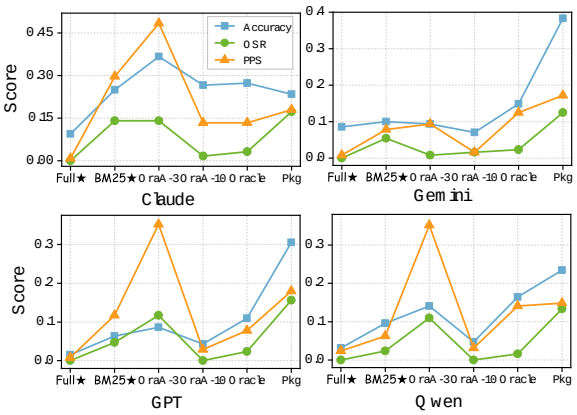

Figure 4: **Effect of increasing flat-tool ambiguity.** ACC, OSR, and PPS decline as the visible tool space moves from Oracle to alternatives, distractors, and Full exposure.

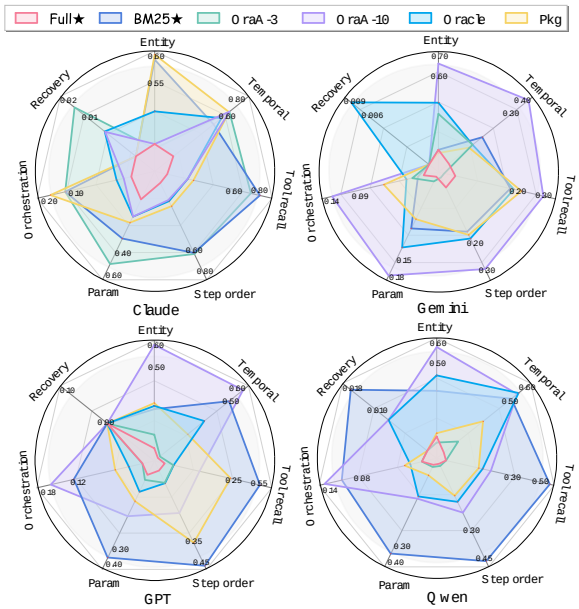

Figure 5: **Process-skill profile.** Full and OraA-10 contract across tool recall, ordering, grounding, parameter planning, recovery, and orchestration.

Oracle-guided results show that the tasks are solvable. With oracle trajectory hints, models reach 83% accuracy, confirming that the bottleneck is autonomous tool discovery and workflow planning, not task impossibility. Bootstrap results are in Appendix F.

4.4 Why Flat Exposure Fails

Process metrics show that flat exposure degrades the whole workflow pipeline. Figure 5 shows that Full and OraA-10 shrink across most dimensions; after a near-miss tool is selected, later steps inherit wrong intermediate results.

Figure 6 shows large flips between ACC- and OSR-based rankings: models can answer correctly without reliable workflows, or execute partial workflows while missing the final value.

The error views make the mechanism concrete. Figure 7 attributes most failures to tool selection, and Figure 8 shows accuracy rising sharply after correct tools are selected. Further diagnostics are in Appendices F.2 and F.3.

5 Conclusion

We presented REALFIN, a finance benchmark for evaluating LLM agents in realistic multi-tool workflows. Its central claim is that MCP-style connectivity alone is not enough: when many tools are exposed as a flat callable palette, agents struggle to select, order, parameterize, and recover from tool use. REALFIN makes this failure mode measur-

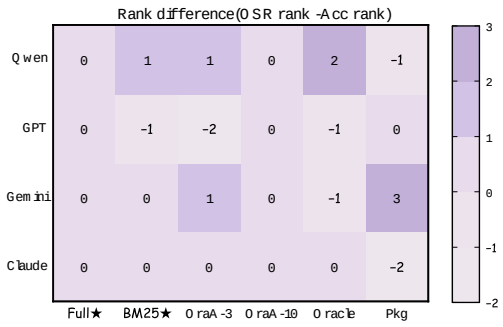

Figure 6: **Rank-flip heatmap.** ACC-based and OSR-based model rankings diverge across exposure regimes, revealing workflow failures hidden by answer-only scoring.

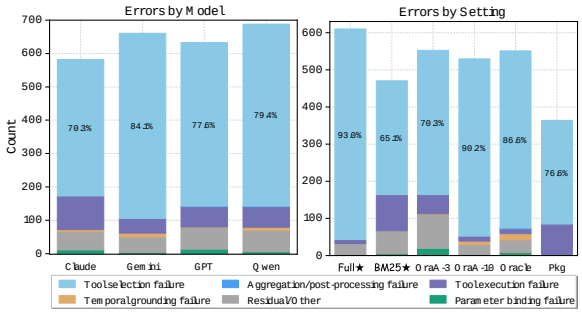

Figure 7: **Error distribution.** Most failures originate from incorrect tool selection and then cascade into downstream workflow errors.

able through executable reference programs, real financial APIs, and process-level traces.

Our experiments show that tool explosion degrades performance even when task semantics are fixed. Errors concentrate first in tool selection and then cascade into temporal grounding, parameter planning, and orchestration failures. Answer-only and process-aware rankings also diverge, showing that correct final answers can hide unreliable workflows. Improvements under guided and progressively disclosed tool settings indicate that many tasks are solvable; the bottleneck is autonomous orchestration in a dense tool space.

These results argue for moving beyond flat exposure toward structured interfaces, retrieval and routing, progressive tool disclosure, and workflow-aware evaluation. REALFIN provides a process-aware testbed for studying this transition in finance.

Limitations

REALFIN focuses on financial workflows, so its findings should be read as evidence from a realistic finance setting rather than as a claim about every

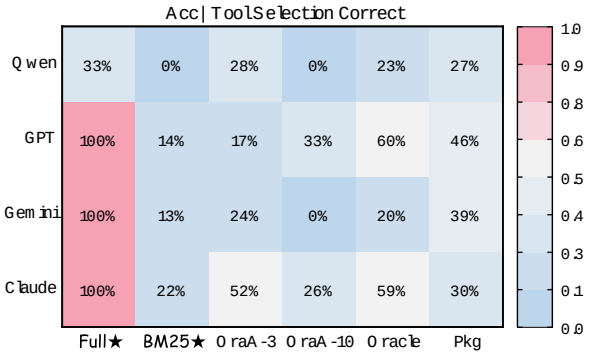

Figure 8: **Conditional accuracy after tool selection.** Accuracy improves sharply once the correct tools are selected, identifying tool choice as the primary bottleneck.

possible domain. The current release emphasizes Chinese financial questions, with English tickers and market names appearing where they are natural in finance. Future versions can extend the benchmark with additional languages, markets, and data providers.

The main model matrix uses a 128-task diagnostic core to keep live API execution and repeated model calls manageable. The full 1,524-task benchmark, executable references, tool schemas, and snapshots are released to support broader follow-up evaluation.

Ethics Statement

REALFIN is an evaluation benchmark rather than a deployed decision-making system. It is designed for scientific study of agent behaviour and does not provide investment advice or trading signals. Nevertheless, evaluating agents in financial settings raises several ethical considerations.

First, we clarify the provenance of the data used in REALFIN. All tasks are generated from publicly accessible financial data sources and public API interfaces. The benchmark does not contain personal data, non-public market information, or proprietary client records. No human subjects were involved, and no manual annotations of personal content were collected.

Second, we consider the risk of potential misuse. Although REALFIN focuses on evaluation rather than training or deployment, stronger agentic capabilities in finance could in principle be used for harmful purposes, including market manipulation, automated speculation, or amplified misinformation about financial assets. To mitigate these risks, the benchmark provides no trading APIs and does

not include tasks that directly optimise or recommend financial actions. The benchmark is intended for controlled research environments rather than end-user decision automation.

Third, we reflect on possible systemic effects. Real financial APIs and time-varying data reflect real-world conditions, but agents evaluated on REALFIN should not be assumed to be safe or reliable for high-stakes use. We discourage interpreting benchmark scores as certification for deployment in financial decision-making pipelines; rigorous domain-specific oversight remains necessary.

Fourth, we discuss fairness and accessibility. The benchmark reflects financial markets and APIs that are available in our environment and may not represent global financial practices equally. We release detailed documentation and open-source code to support transparency and reproducibility, while acknowledging that API access may depend on regional availability.

Finally, we clarify the role of AI assistance in benchmark preparation. LLMs were not used as autonomous sources of benchmark labels, final answers, or unverified reference workflows. Any AI-assisted drafting, coding, or checking was manually inspected by the authors, and released instances are retained only after template-level validation, executable reference-program checks, and human quality review.

Overall, we emphasize that REALFIN aims to support responsible research into agent evaluation by making processes observable and failure modes explicit, rather than by enabling autonomous financial decision systems. Researchers using this benchmark should consider the broader societal impact of agentic AI in finance and avoid applications that could lead to individual or systemic harm.

References

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Kenneth Huang, Bryan R. Routledge, and William Yang Wang. 2021. Finqa: A dataset of numerical reasoning over financial data. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, pages 3697–3711. Association for Computational Linguistics.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022. Convfinqa: Exploring the chain of numerical reasoning in conversational finance question answering. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022, pages 6279–6292. Association for Computational Linguistics.

Chanyeol Choi, Jihoon Kwon, Alejandro Lopez-Lira, Chaewoon Kim, Minjae Kim, Juneha Hwang, Jaeseon Ha, Hojun Choi, Suyeol Yun, Yongjin Kim, and Yongjae Lee. 2025. Finagentbench: A benchmark dataset for agentic retrieval in financial question answering. CoRR, abs/2508.14052.

Yang Deng, Wenqiang Lei, Wenxuan Zhang, Wai Lam, and Tat-Seng Chua. 2022. PACIFIC: towards proactive conversational question answering over tabular and textual data in finance. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022, pages 6970–6984. Association for Computational Linguistics.

Xin Guo, Haotian Xia, Zhaowei Liu, Hanyang Cao, Zhi Yang, Zhiqiang Liu, Sizhe Wang, Jinyi Niu, Chuqi Wang, Yanhui Wang, Xiaolong Liang, Xiaoming Huang, Bing Zhu, Zhongyu Wei, Yun Chen, Weining Shen, and Liwen Zhang. 2025. Fineval: A chinese financial domain knowledge evaluation benchmark for large language models. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025, pages 6258–6292. Association for Computational Linguistics.

Liang Hu, Jianpeng Jiao, Jiashuo Liu, Yanle Ren, Zhoufutu Wen, Kaiyuan Zhang, Xuanliang Zhang, Xiang Gao, Tianci He, Fei Hu, Yali Liao, Zaiyuan Wang, Chenghao Yang, Qianyu Yang, Mingren Yin, Zhiyuan Zeng, Ge Zhang, Xinyi Zhang, Xiying Zhao, and 4 others. 2025. Finsearchcomp: Towards a realistic, expert-level evaluation of financial search and reasoning. CoRR, abs/2509.13160.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. Financebench: A new benchmark for financial question answering. CoRR, abs/2311.11944.

Albert King. 2022. Akshare. https://github.com/akfamily/akshare.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, K. P. Subbalakshmi, Jimin Huang, Lingfei Qian, Xueqing Peng, Jordan W. Suchow, and Qianqian Xie. 2025. 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), ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pages 2509–2525. Association for Computational Linguistics.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023, pages 3102–3116. Association for Computational Linguistics.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su,

Huan Sun, and 3 others. 2024. Agentbench: Evaluating llms as agents. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net.

Jiaxuan Lu, Kong Wang, Yemin Wang, Qingmei Tang, Hongwei Zeng, Xiang Chen, Jiahao Pi, Shujian Deng, Lingzhi Chen, Yi Fu, Kehua Yang, and Xiao Sun. 2026. Fintoolbench: Evaluating LLM agents for real-world financial tool use. CoRR, abs/2603.08262.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: a benchmark for general AI assistants. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net.

Model Context Protocol. 2025. Tools - model context protocol specification. https://modelcontextprotocol.io/specification/2025-06-18/server/tools. Accessed: 2026-05-20.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): from tool use to agentic evaluation of large language models. In Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025. OpenReview.net.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024a. Toolllm: Facilitating large language models to master 16000+ real-world apis. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024b. Toolllm: Facilitating large language models to master 16000+ real-world apis. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. 2025. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pages 24497–24524. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An open-ended embodied agent with large language models. Trans. Mach. Learn. Res., 2024.

Pei Wang, Yanan Wu, Noah Wang, Jiaheng Liu, Xiaoshuai Song, Z. Y. Peng, Ken Deng, Chenchen Zhang, Jiakai Wang, Junran Peng, Ge Zhang, Hangyu Guo, Zhaoxiang Zhang, Wenbo Su, and Bo Zheng. 2025a. Mtu-bench: A multi-granularity tool-use benchmark for large language models. In The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, and Eugene Siow. 2025b. Mcp-bench: Benchmarking tool-using LLM agents with complex real-world tasks via MCP servers. CoRR, abs/2508.20453.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, Yijing Xu, Haoqiang Kang, Ziyan Kuang, Chenhan Yuan, Kailai Yang, Zheheng Luo, Tianlin Zhang, Zhiwei Liu, Guojun Xiong, and 15 others. 2024. Finben: A holistic financial benchmark for large language models. In Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 3277–3287. Association for Computational Linguistics.

Jie Zhu, Yimin Tian, Boyang Li, Kehao Wu, Zhongzhi Liang, Junhui Li, Xianyin Zhang, Lifan Guo, Feng Chen, Yong Liu, and Chi Zhang. 2026. Finmcp-bench: Benchmarking LLM agents for real-world financial tool use under the model context protocol. CoRR, abs/2603.24943.

Appendix

A Dataset Statistics

Type distribution. Table 7 summarizes the distribution of task types in REALFIN.

Takeaways.

Type	Count	Proportion
T3_Aggregate	677	44.4%
T1_Snapshot	412	27.0%
T4_Fundamentals	328	21.5%
T2_Compare	107	7.0%

Table 7: Distribution of task types in the 1,524-instance dataset.

- 1,524 valid instances, no broken records.
- Aggregate questions (T3) constitute nearly half of the corpus.

Filtering of non-executable candidates. Before release, template-instantiated candidates are executed and checked against their reference programs. Candidates that cannot support a deterministic and executable answer are repaired when possible or removed. Table 8 lists the main filtering categories.

Filtering reason	Typical trigger
Invalid entity or ticker	The company, security, index, or fund cannot be resolved to a unique valid identifier.
Market/date mismatch	The requested entity is queried under an incompatible exchange, market, or reporting calendar.
Unresolvable non-trading day	The query date is a non-trading day and the required previous or next trading day cannot be determined reliably.
Insufficient historical window	The entity has too short a listing history or too few observations for the required lookback computation.
Empty API response	The external API returns an empty table, missing field, or unavailable endpoint response for the requested inputs.
API schema drift	The provider changes column names, field types, or return formats so the reference program no longer executes as specified.
Ambiguous matching reports	Multiple reports or disclosure records match the same query constraints, yielding no unique deterministic answer.
Unstable realtime value	A realtime value changes beyond the allowed tolerance during repeated validation runs.

Table 8: Main reasons for repairing or removing non-executable candidate programs during benchmark construction. The released dataset contains only candidates that pass these checks.

Execution depth.

Tool usage statistics. **Takeaways.**

- 34 single-tool vs. 1,490 multi-tool instances (2.2% vs. 97.8%).
- 1,404 instances (92.1%) require actual chaining (depth ≥ 2).

Time sensitivity. **Type $\times$ Time Sensitivity.** **Depth $\times$ Time Sensitivity.** **Takeaways.**

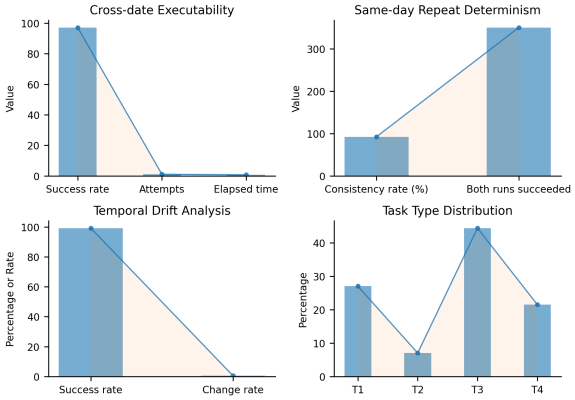

Figure 9: **Code analysis experiments.** We evaluate executability across dates, determinism across repeated runs, and temporal drift of time-sensitive tasks. High executability with non-trivial drift confirms that REALFIN captures realistic, time-varying financial environments.

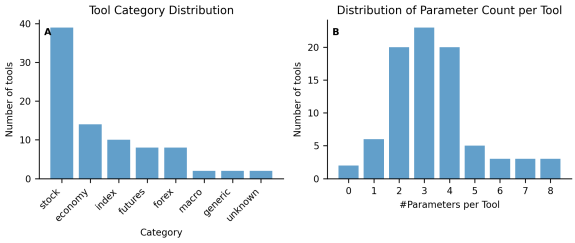

Figure 10: **Tool ecosystem statistics.** Left: distribution of tool categories. Right: distribution of parameter counts per tool. The benchmark exhibits heterogeneous and partially redundant tools, operationalizing the challenge of tool explosion.

- Fixed-date queries dominate; relative-time queries are substantial.
- Realtime queries are mostly shallow and largely in T1.
- Deeper chains skew toward relative-time computation.

Workflow complexity signals. **Takeaways.**

- Control flow is pervasive despite moderate average depth.
- Multi-tool usage and chained execution are common; depth-3 workflows remain the minority.

Extended Dataset Statistics.

B Tool Library Statistics

Domain distribution. **Takeaways.**

- Stock-related tools dominate the library.

Time Sensitivity	Count	Proportion
Fixed	794	52.1%
Relative	629	41.3%
Realtime	101	6.6%

Table 9: Time-sensitivity labels for dataset instances.

Type	Fixed	Realtime	Relative
T1_Snapshot	315	97	0
T2_Compare	102	4	1
T3_Aggregate	49	0	628
T4_Fundamentals	328	0	0

Table 10: Joint distribution of task type and time sensitivity.

– Long tail of mixed-domain tools reflects cross-asset workflows.

Task types of tools. Takeaways.

– The library is primarily retrieval-oriented; computational tools are secondary.

Dependency graph properties. Graph statistics.

– Leaves: 46

– Internal nodes: 39

– Longest chain length: 4

Hub tools (fan-in).

– `get_code_from_name`: 18

– `search_kg`: 4

– `get_index_history`: 3

Fan-out.

– `calculate_gold_forex_change`: 2

– `get_a_stock_daily_price`: 2

Takeaways.

– The dependency graph is shallow with a few hub tools.

Tool relation metadata. Each agent-callable tool description includes three relation fields: `tool_alternative` for tools that can fill similar workflow roles under compatible constraints, `tool_dependencies` for prerequisite tools such as entity grounding or date resolution, and `tool_related` for near-neighbor tools that may be useful but are not automatically interchangeable. Table 14 summarizes these annotations over the 85 callable tools. The non-empty alternative and dependency fields provide direct metadata evidence that the released tool graph is not a set of isolated one-step functions: valid workflows can branch through substitutes, while many tools require upstream grounding or retrieval steps.

Signal	Count	Proportion
Control flow in code	1436	94.2%
Multiple tools (>1)	1,490	97.8%
Chained tools (depth $\geq$ 2)	1,404	92.1%

Table 11: Indicators of workflow complexity.

Statistic	**Value**
Number of questions	1,524
Executable golden codes	1,524
Tool functions	85
Markets covered	A, HK, US, FX, Macro
Task types (T1/T2/T3/T4)	27.0% / 7.0% / 44.4% / 21.5%
Question language: Chinese	99.5%
Question language: English	0.5%
Avg. execution depth	2.12
Avg. tool calls per instance	3.78
Multi-tool questions	97.8%
Temporal regime: Fixed	52.1%
Temporal regime: Relative	41.3%
Temporal regime: Realtime	6.6%
Golden program success rate	97.15%
Avg. attempts per program	1.06
Repeat-run consistency	87.64%
Temporal drift proportion	0.55%

Table 12: **Statistics of REALFIN.** All questions are paired with executable golden programs. Dataset composition statistics are computed from the released `realfin_data.jsonl` file.

Alternative-tool graph. The `tool_alternative` field yields 104 directed alternative links, 101 of which point to tools inside the callable library. Treating these links as undirected gives 14 non-singleton alternative components; the largest component contains seven currency, FX, and realtime-value tools. This structure is used by process metrics to credit semantically valid substitutions rather than exact tool-name matches.

Error models. Takeaways.

– 97.6% of tools expose structured error models.

– Parameter and internal errors dominate.

Real APIs vs. virtual tools. Takeaways.

– Rough balance between wrapper tools and virtual/algorithmic tools.

Domain	Count	Proportion
stock	37	43.5%
index	10	11.8%
economy	10	11.8%
futures	7	8.2%
forex	7	8.2%
generic	4	4.8%
mixed domains	8	–

Table 13: Domain coverage of tools. Mixed-domain rows aggregate tools annotated with multiple domains.

Relation field	Tools	Coverage	Edges	Avg.	Max
`tool_alternative`	45	52.9%	104	1.22	6
`tool_dependencies`	49	57.6%	93	1.09	3
`tool_related`	78	91.8%	665	7.82	36
Any relation	79	92.9%	–	–	–

Table 14: Relation metadata in the 85 agent-callable tool descriptions. Edges count listed relation targets; coverage is the fraction of tools with a non-empty field.

- Akshare is the dominant external data provider.

C Released Artifacts and Reproducibility

REALFIN supports two evaluation modes. In snapshot evaluation, model traces are scored against frozen task records, versioned tool schemas, executable golden programs, and cached API responses. This mode is intended for reproducible comparison across agents. In live evaluation, the same programs call the current financial APIs, preserving realistic non-stationarity but allowing values to drift.

The released package includes the 1,524 task records, the 128-task core subset, tool schemas and package metadata, executable reference programs, response snapshots for snapshot-mode scoring, model prompts, output parsers, retry policies, and metric scripts. Each task record stores the question, reference code, answer or realtime marker, task type, time-sensitivity label, reference tool count, execution depth, tool coverage, and template id. Realtime answers are evaluated with tolerance rules; failed or empty API calls are logged separately from semantic process failures.

C.1 Coverage of the 128-task diagnostic core

Table 16 compares the released full benchmark with the 128-task diagnostic core used for the main model matrix. The core is not intended to be a miniature leaderboard distribution. It preserves workflow-size statistics while deliberately oversampling rare Realtime and Compare cases, so the experiments stress temporal grounding and tool ambiguity under a feasible live-evaluation budget.

Metric	Value
Agent-callable tools exposed in evaluation	85
Underlying implementation entries	133
External-API tools	65
Virtual / algorithmic tools	68
Unique Akshare endpoints	66

Table 15: Real vs. virtual tool implementation statistics. The evaluated Full setting exposes 85 agent-callable tools. The remaining counts describe lower-level implementation entries backing those callable tools.

Statistic	Full benchmark	128-task core
Instances	1,524	128
Task types	T1/T2/T3/T4 = 412/107/677/328	T1/T2/T3/T4 = 45/24/49/10
Temporal regimes	Fixed/Relative/Realtime = 794/629/101	Fixed/Relative/Realtime = 51/41/36
Avg. reference tools	3.78	4.00
Avg. execution depth	2.12	1.93
KS comparison	–	tool count $p = 0.548$; depth $p = 0.088$

Table 16: Coverage statistics for the full benchmark and the 128-task diagnostic core.

D Full metric definitions

This appendix provides full mathematical definitions of the metrics introduced in Section §4. We distinguish between outcome accuracy and process-based agent-skill metrics.

D.1 Notation

Let the dataset be $\mathcal{D} = \{1, \ldots, N\}$. For each instance $i \in \mathcal{D}$:

- y_i : reference answer
- $\hat{y}_i$: model's final answer
- $T_i^* = \langle t_{i,1}^*, \ldots, t_{i,m_i}^* \rangle$: one verified reference tool sequence
- $\hat{T}_i = \langle \hat{t}_{i,1}, \ldots, \hat{t}_{i,k_i} \rangle$: model tool sequence
- $\mathcal{T}_i^*$: set of reference tools (ignoring order)
- $\hat{\mathcal{T}}_i$: set of tools used by the model
- $\mathcal{A}_i(t)$: annotated equivalent tools that can replace tool t for instance i
- d_i^* : gold workflow depth $= |T_i^*|$

– $\hat{d}_i$: model workflow depth $= |\hat{T}_i|$

– $r_{i,j}$: execution result of tool call $\hat{t}_{i,j}$

We write $\mathbf{1}[\cdot]$ for the indicator function. The reference program specifies one verified path, not the only valid path. For process metrics, a model call is considered tool-equivalent to a reference call if it belongs to the annotated alternative set for that instance and produces the required intermediate semantics. Final-answer scoring is path independent.

D.2 Outcome-Level Metrics

Outcome Accuracy (ACC).

$$\text{ACC} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\hat{y}_i = y_i]. \tag{1}$$

D.3 Execution and Orchestration

Execution Success Rate (ESR). Let $\text{OK}(r)$ denote a non-error, non-null result.

$$\text{ESR} = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{k_i} \mathbf{1}[\text{OK}(r_{i,j})]. \tag{2}$$

Full Orchestration Success Rate (OSR). A sample is successful if all skills succeed:

$$\text{OSR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\text{TSR}_i = \text{OA}_i = \text{PPS}_i = \text{EGA}_i = 1].$$

D.4 Tool Selection

Let

$$\mathcal{T}_i^* = \{t_{i,1}^*, \ldots, t_{i,m_i}^*\}, \quad \hat{\mathcal{T}}_i = \{\hat{t}_{i,1}, \ldots, \hat{t}_{i,k_i}\}.$$

Define the equivalent reference set

$$\mathcal{E}_i^* = \bigcup_{t \in \mathcal{T}_i^*} (\{t\} \cup \mathcal{A}_i(t)).$$

Tool Selection Recall (TSR).

$$\text{TSR} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{|\mathcal{T}_i^*|} \sum_{t \in \mathcal{T}_i^*} \mathbf{1}\Big[(\{t\} \cup \mathcal{A}_i(t)) \cap \hat{\mathcal{T}}_i \neq \emptyset\Big]. \tag{3}$$

D.5 Ordering and Workflow Planning

Ordering Accuracy (OA). Let $\text{Subseq}_{\mathcal{A}}(A, B) = 1$ iff each step in A appears in B in dependency-compatible order, allowing tools in $\mathcal{A}_i(t)$ to substitute for reference tool t.

$$\text{OA} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\Big[\text{Subseq}_{\mathcal{A}}(T_i^*, \hat{T}_i)\Big]. \tag{4}$$

This measures whether the model preserves the logical order of required operations, not whether it exactly copies the reference tool names.

D.6 Parameter Planning

Define validity of a tool call result:

$$\text{Good}(r) = \mathbf{1}[r \neq \text{NULL},\ r \neq \text{ERROR}].$$

Parameter Planning Success (per instance). For each gold-required tool class c, consider the model's first call to a tool in c and its result r.

$$\text{PPS}_i = \prod_{c \in \mathcal{C}_i^*} \text{Good}(r_{i,c}), \tag{5}$$

where $\mathcal{C}_i^*$ is the set of required tool classes.

Dataset-level PPS.

$$\text{PPS} = \frac{1}{N}\sum_{i=1}^{N} \text{PPS}_i.$$

D.7 Entity Grounding

Let E_i^* be entity set from question + gold program, and $\hat{E}_i$ the entities implied by tool calls.

$$\text{EGA} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\Big[\exists e \in \hat{E}_i \text{ matching some } e^* \in E_i^*\Big].$$

D.8 Recovery Success Rate (RSS)

Let $\hat{S}_i$ be the set of steps taken by the agent for task i, and S_i^* the reference steps in the gold program.

$$\text{RSS} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\Big[\exists s \in \hat{S}_i \text{ such that } s \in S_i^*\Big].$$

RSS evaluates the success of the agent in recovering from errors by comparing the steps taken by the agent with those in the reference workflow.

D.9 Temporal Grounding Accuracy (TGA)

Let T_i^* be the set of temporal references from the question + gold program, and $\hat{T}_i$ the temporal references inferred by the agent's tool calls.

$$\text{TGA} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\Big[\exists t \in \hat{T}_i \text{ matching some } t^* \in T_i^*\Big].$$

TGA measures the agent's ability to correctly interpret and apply temporal information, such as dates, time periods, or relative times, within the context of the task.

D.10 Detailed Performance Metrics

Table 17 presents the comprehensive performance metrics across different retrieval strategies and language models on our benchmark dataset.

D.11 Representative Benchmark Instance

Table 18 illustrates one released REALFIN instance and how alternative workflows are scored. The reference program gives one verified path, but process scoring accepts annotated substitutes when they satisfy the same intermediate semantics. Final-answer accuracy is path independent.

D.12 Case Study

D.12.1 Case Study I: Do functionally redundant tools unlock alternative execution paths?

Claude-Sonnet-4.5 achieves higher accuracy under OraA-3 tool exposure than under the oracle Oracle setting. Since OraA-3 differs from Oracle primarily by adding functionally redundant tools, we ask:

> **Do redundant tools unlock alternative executable workflows that are unavailable under the oracle minimal tool set?**

Experimental probe. We evaluate GPT-5.1 in a controlled setting where the agent is given the oracle tool set $\mathcal{G}$ augmented with redundant tools. We then search for instances satisfying:

1. the oracle tool set alone is sufficient to solve the task; and
2. the model nevertheless solves the task through a different workflow that explicitly uses redundant tools.

Across 1,524 questions we identify six such cases. Although rare, they are conceptually important: current agents seldom explore alternative workflows, yet multi-path execution does occur and can change measured accuracy under Top-k exposure.

Representative examples. Figure 11 illustrates two representative cases.

(a) Alternative executable paths. The agent sometimes constructs a workflow different from the golden program but semantically equivalent, showing that the benchmark implicitly admits multiple valid execution paths even when only one is annotated.

(b) Redundant tools as rescue routes. In other cases, the canonical workflow fails due to missing data or transient API errors, while the agent succeeds by switching to a redundant tool. Redundancy thus acts as a fallback mechanism and improves robustness, rather than serving only as distraction.

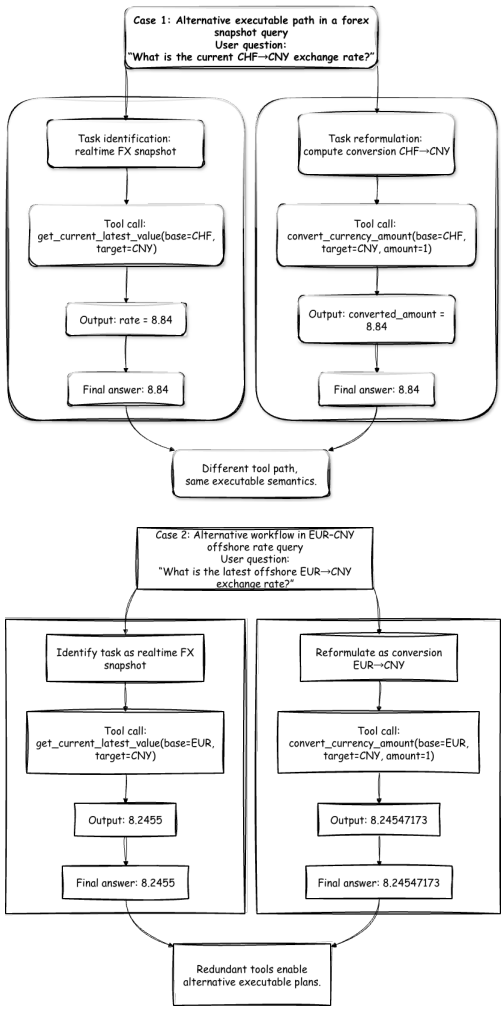

Figure 11: **Two examples of multi-path execution unlocked by redundant tools.** Top: an FX snapshot query solved through an alternative API chain; Bottom: a dividend query where redundant tools provide a fallback path after canonical-tool failure.

Takeaway. This case study suggests that:

- redundant tools occasionally unlock alternative valid execution paths that are not used under Oracle settings;
- current agents do not systematically search

workflow space, explaining the small number of observed cases;

– when such paths are discovered, they can improve accuracy under Top-K tool exposure.

Taken together with Case Study II, this reveals a nuanced picture: moderate Top-K exposure may help by increasing the number of valid paths, whereas excessive tool exposure may hurt by increasing decision uncertainty and abstention. The key challenge is therefore not only to provide tools, but to guide agents in exploring the space of executable workflows effectively.

D.12.2 Case Study II: Does a Larger Top-K Make Agents Stronger or More Hesitant?

In the previous case study, we showed that OraA-3 exposure can outperform the oracle Oracle setting because it provides additional valid execution paths: agents sometimes reach the correct answer through workflows that differ from the reference program. We now ask a complementary question: **if increasing K creates even more potential paths, why does OraA-10 often perform worse than OraA-3?**

Experimental setup. We focus on the 394-instance paired subset. Each instance is evaluated twice with identical prompts, evaluation protocol, and models; the only difference is the size of the tool list presented to the agent ($k = 3$ vs. $k = 10$). This paired design isolates the effect of Top-K exposure from all other confounders.

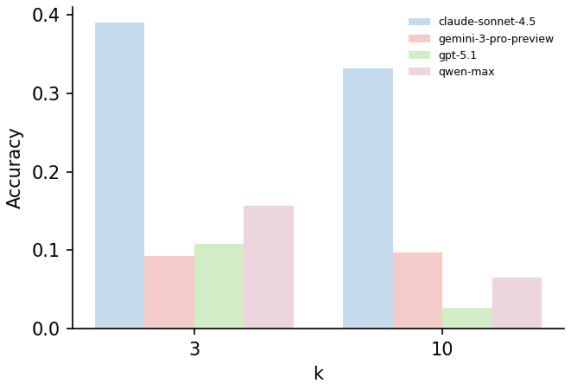

Figure 12: Accuracy comparison between $k = 3$ and $k = 10$ on the 394-item paired subset. Each bar shows mean accuracy across samples for each model.

Observation 1: Increasing K reduces accuracy. Figure 12 shows that OraA-3 consistently outperforms OraA-10 across models on the same tasks. This result is striking because OraA-10 strictly contains OraA-3: if tool availability were the primary bottleneck, accuracy should be non-decreasing in K. Instead, expanding the visible tool set degrades performance, suggesting that the dominant factor is not the absence of the right tool, but the agent's behavior under increased choice.

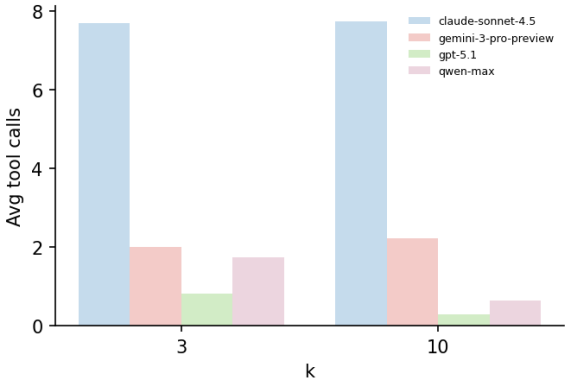

Figure 13: Change in average number of tool calls when increasing from $k = 3$ to $k = 10$. Positive values indicate more tool usage under larger k.

Observation 2: Lower accuracy is not caused by insufficient tool use. Figure 13 shows that the mean number of tool calls does not decrease when moving from $k = 3$ to $k = 10$; in many models it even increases. Therefore, OraA-10 does not fail simply because agents "give up early" or "call too few tools". Instead, errors arise during execution—mis-parameterization, tool mis-selection among near-duplicates, or inconsistent multi-step composition—rather than from a lack of action.

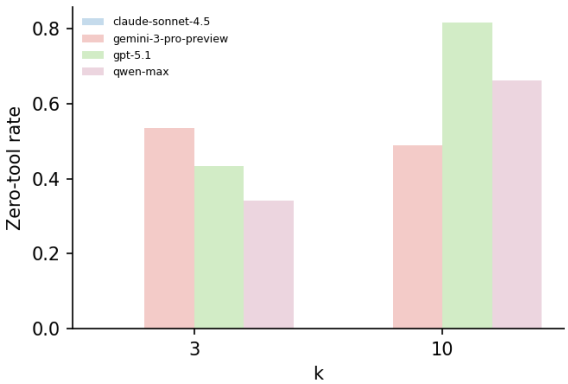

Figure 14: Zero-tool answer rate for $k = 3$ and $k = 10$. Higher values indicate a stronger tendency to avoid tool usage entirely.

Observation 3: Larger K increases hesitation and abstention. Figure 14 shows that OraA-10

settings exhibit a markedly higher zero-tool answer rate than OraA-3. Thus, enlarging the tool set does not only change which tools are used; it changes whether the agent decides to use tools at all. This indicates a shift in strategy: when faced with a larger and more redundant tool space, agents increasingly default to direct answers instead of committing to multi-step tool execution.

Interpretation. Putting these observations together, we obtain the following picture:

- Oracle → OraA-3 improves performance by introducing additional valid execution paths.
- OraA-3 → OraA-10 degrades performance by inflating decision uncertainty and abstention pressure.

In other words, tool exposure has a non-monotonic effect: a moderate number of tools increases the chance of discovering a successful workflow, but excessive tool exposure induces hesitation, increases zero-tool responses, and amplifies composition errors in deep tool chains.

Takeaway. The limiting factor for current agents is therefore not simply tool availability, but their robustness under choice overload. More tools can make agents not only more error-prone, but also more hesitant to act—highlighting tool explosion as a behavioral phenomenon rather than a purely algorithmic scaling issue.

D.12.3 Case Study III: Does coarse tool packaging eliminate the tool-explosion problem?

Top-K exposure limits the visible tool set but does not restructure it. A complementary strategy is to reorganize tools hierarchically into coarse functional packages and require the agent to first select an intent, then a package, and only then a specific tool. This raises the guiding question of this case study:

> **If tools are hierarchically packaged, does tool explosion essentially disappear?**

Hierarchical routing and packaging in practice. Figure 15 shows a representative execution trace under our three-stage routing framework: (i) coarse-grained intent classification, (ii) fine-grained intent routing, and (iii) tool selection within a constrained package. As illustrated in the figure, even after packaging, the agent must still choose among multiple highly similar tools inside the same package and reason over heterogeneous argument schemas and return formats. Thus, packaging restructures the search space, but does not collapse it into a single deterministic route.

Aggregate empirical effect. We evaluate GPT-5.1 under this packaged regime and obtain 180 correct answers out of 520 questions (approximately 30% accuracy). This confirms that hierarchical packaging does help: compared with flat tool exposure, the agent faces fewer irrelevant tools and the routing decisions visualized in Figure 15 become more organized and interpretable.

But tool explosion persists. However, the same execution trace in Figure 15 also reveals why accuracy remains far from satisfactory. Most errors arise from:

- confusion among multiple tools within the same functional family,
- unstable parameter grounding across stages of the workflow,
- failure to normalize heterogeneous output schemas across tools,
- early abandonment of tool use when latency or API noise is encountered.

The branching structure shown in Figure 15 indicates that even after packaging, the agent still faces a large workflow search space rather than a single correct pipeline.

Takeaway. Coarse tool packaging mitigates surface-level search overload but does not eliminate tool explosion. The challenge is structural: the agent must coordinate multi-step workflows across noisy, partially redundant tools. Therefore, tool explosion is not merely a catalog-design artifact; it persists even under hierarchical routing, as visually evidenced in Figure 15.

D.12.4 Case Study IV: Why does accuracy degrade beyond execution depth 1?

Across all three subsets (128 / 249 / 394) and all exposure conditions (Oracle, Top-3, Top-10, full), we consistently observe the same phenomenon: tasks with execution depth 1 maintain non-zero accuracy, whereas accuracy declines substantially as reference workflow depth increases to depth 3. This is not an artifact of specific models or tool exposure settings; rather, it suggests a structural failure mode of current LLM-based agents.

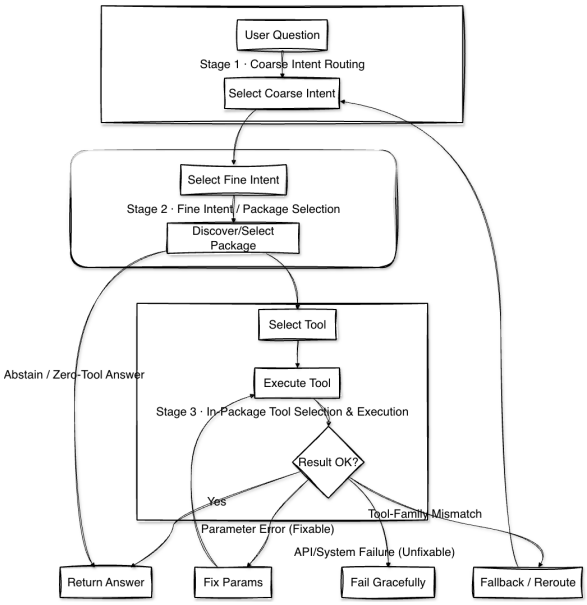

Figure 15: **Hierarchical intent routing and packaged tool selection.** The execution trace demonstrates that even when tools are grouped into packages, the agent must still disambiguate between multiple similar tools and coordinate multi-step workflows, indicating that tool explosion persists in packaged settings.

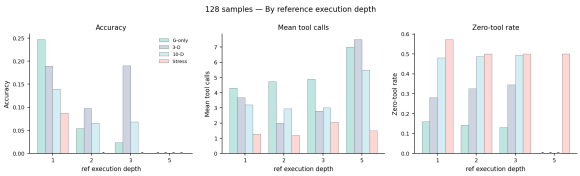

Figure 16: Zero-tool answer rate for $k = 3$ and $k = 10$. Higher values indicate a stronger tendency to avoid tool usage entirely.

Figure 16 illustrates this monotonic depth-induced degradation: depth 1 tasks are solvable at modest rates, but depth 3 tasks are substantially harder across tool-exposure configurations.

Observation 1: Failure is depth-driven, not exposure-driven. Notably, the degradation at depth 3 occurs even in the Oracle (oracle tool set) condition where tool selection burden is eliminated. The same pattern appears under Top-3, Top-10, and full exposure. This indicates that the source of difficulty is not merely tool retrieval or selection; instead, execution depth itself is a dominant factor.

Observation 2: Depth increases tool-use behavior but not accuracy. Across settings, the mean number of tool calls increases with execution depth, yet accuracy simultaneously decreases. This dissociation shows that additional tool use does not translate into improved correctness: agents attempt longer workflows, but cannot maintain coherent execution over multiple steps.

Observation 3: Divergent failure behaviors at high depth. At higher workflow depth, we observe two distinct behavioral responses depending on tool exposure: (i) under full (full-tool exposure), the zero-tool rate rises to around 50%, suggesting deliberate abstention in the face of overwhelming search space; (ii) under Top-K exposure, zero-tool rates drop to nearly zero while tool calls surge, indicating thrashing behavior where the agent repeatedly invokes tools without converging on the correct workflow. These contrastive behaviors point to a common bottleneck—long-horizon planning—modulated by exposure-induced uncertainty.

Why does depth hurt so strongly? The empirical evidence supports a multi-step consistency explanation:

- **state drift**: intermediate tool outputs must be remembered and reused, but partial results are overwritten or inconsistently referenced across steps;
- **temporal misalignment**: trading-day offsets and date normalization must remain consistent across calls, and small inconsistencies compound over depth;
- **parameter propagation errors**: arguments derived from previous tool outputs are misparsed, reformulated, or silently dropped in later calls;
- **error accumulation**: small inaccuracies at early steps propagate forward and dominate the final answer;
- **goal myopia**: agents locally repair tool errors without maintaining a global workflow plan, leading to the high-call "thrashing" observed in deeper workflows.

Takeaway. Execution depth is the single strongest predictor of failure in REALFIN. Even when tool selection is simplified (Oracle) or tool space is constrained (Top-K), agents fail beyond one or two steps. This suggests that the key challenge in financial agent evaluation is not isolated tool use, but maintaining globally consistent, temporally aligned, multi-step workflows over long horizons. Long-chain orchestration—rather than single-step competence—is therefore the dominant bottleneck for current LLM agents.

D.12.5 Case Study V: Representative failure types

We illustrate how tool explosion changes the error landscape. Failures are not random: they cluster into a small number of systematic types that align with our process metrics.

E Alignment with Real-World Financial SOPs

We present three representative cases to illustrate that RealFin golden workflows closely mirror real-world financial standard operating procedures (SOPs). The cases cover (1) equity quarterly return computation, (2) cross-sectional liquidity aggregation, and (3) precious-metals benchmark price comparison.

F Supplement Experiment Results

F.1 Human Validation Figure

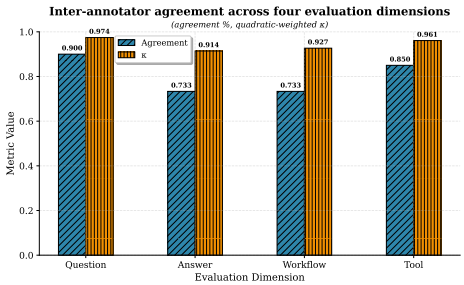

Figure 17: Inter annotator agreement across four evaluation dimensions.

F.2 Detailed Diagnostics for Flat Exposure

To move beyond answer-level metrics, we analyze agent behavior in REALFIN using process-based measures. The radar plot in Figure 5 summarizes seven dimensions of agent capability, including tool recall, step consistency, parameter planning, entity and temporal grounding, workflow execution success, and error recovery. The central pattern is that Full and OraA-10 contract sharply across nearly all process dimensions, suggesting that flat exposure degrades the entire workflow pipeline rather than a single isolated skill.

Wrong tool choices propagate into workflow failure. Tool recall and step consistency decline together across exposure settings, indicating that many ordering errors originate from incorrect tool selection rather than step planning alone. This is the core mechanism behind the failure of flat exposure: once the model selects a near-miss tool, later steps are built on the wrong intermediate result, causing parameter errors, temporal mismatches, and failed aggregation. Error recovery also remains low and deteriorates under tool explosion. Once models commit to an incorrect tool or path, they rarely abandon it and replan, behaving more like one-shot planners than robust tool-using agents.

Execution succeeds more often than workflows do. Process-level metrics reveal key challenges in multi-step financial workflows. As shown in Table 17, ESR is generally high while ACC and OSR remain much lower. This means that agents often manage to execute API calls, but fail to turn those calls into correct workflows. The failure is therefore not primarily a low-level execution problem. It is a decision problem caused by tool selection, parameter construction, temporal grounding, and result interpretation. Additional analysis in Table 24 further shows that many failures originate from tool retrieval in large action spaces, and even with correct tools, agents often struggle with ordering, parameter selection, and temporal grounding.

Execution-based rankings expose hidden failures. Model rankings induced by ACC and OSR diverge substantially (Figure 6). The gap is largest under medium tool spaces (Oracle, OraA-3), where models may answer correctly without completing workflows, or execute workflows yet miss the final value. Under very large tool spaces (OraA-10, Full), OSR collapses toward zero and execution-based rankings flatten. This flattening is itself evidence of interface breakdown: once the tool palette becomes too dense, model differences are masked by the shared inability to recover reliable workflows. Table 24 further shows that most failures occur at tool selection, with $\text{ACC} \mid \text{selection} \gg \text{ACC}$, indicating that the main bottleneck lies in tool selection and workflow orchestration rather than single-step tool invocation.

External structure changes behavior more than prompting alone. Structured interfaces stabilize tool selection and improve orchestration success, but the source of structure matters. Retrieval-based exposure keeps the ReAct-style loop fixed and changes only the candidate tools. Package routing changes the agent harness by progressively

disclosing tools after a package-level routing decision. This supports our critique of flat MCP-style tool exposure: the same underlying tools become more usable when the action space is organized rather than revealed all at once. However, these gains mainly reduce decision burden rather than strengthen intrinsic agent skills, as recovery and replanning improve only marginally (Appendix D).

F.3 Controlled Interventions on Tool-Space Ambiguity

Tool explosion enlarges the agent's action space and increases ambiguity among plausible tools. Rather than making strong causal claims, we conduct two quasi-experimental interventions to examine how tool-space size and context length relate to agent behaviour. Both interventions test whether the failure of flat exposure is caused only by having more tokens in the prompt, or by the semantic ambiguity introduced by dense tool spaces.

Intervention 1: tool-space size We fix the prompt context and vary only the number of available tools. We evaluate $k \in \{0, 3, 6, 10\}$ distractors under the G+R+kN configuration while keeping tasks and prompts identical. Qwen is reported as a representative model because it avoids ceiling and floor effects and shows clear variation in ACC, TSR and OSR across k (Table 25b). As k increases, TSR and OSR decrease and OSR is essentially zero for $k \geq 6$, showing that additional plausible tools make selection and orchestration harder.

Intervention 2: context length We fix the tool set (OraA-3) and vary only prompt length using short and long versions of the same tasks on a 30-task subset. Both Claude and Qwen are evaluated. Longer contexts are consistently associated with lower ACC, TSR and PPS, and OSR drops to zero for both models (Table 25a). Thus, reliable agents need more than a larger context window or a longer flat tool list; they need interfaces and evaluation protocols that expose, structure, and diagnose workflow decisions.

F.4 Error analysis

F.4.1 Quantitative error breakdown

Figure 7 shows that the majority of errors are caused by *tool selection* (78.1%), with execution errors (10.5%) and residual errors (9.0%) forming most of the remainder. Parameter, temporal, and aggregation errors jointly account for less than 3%. Thus the primary bottleneck in REALFIN is deciding which tools to use, not arithmetic or aggregation.

This pattern is consistent across models (70–84% selection error) and exposure regimes: Full and OraA-10 exhibit the highest selection error rates, whereas OraA-3 and BM25 regimes contain more execution mistakes.

F.4.2 Conditional performance given correct skills

We compute accuracy (i) conditioned on correct tool selection and (ii) conditioned on both correct selection and reference-consistent orchestration (Figure 8).

Main observation. Across all models, $\mathrm{ACC} \mid \mathrm{Selection}$ is 3–6$\times$ higher than raw accuracy, whereas $\mathrm{ACC} \mid \mathrm{Orchestration} \approx \mathrm{ACC} \mid \mathrm{Selection}$. Hence, once the right tools are chosen, ordering and arithmetic rarely fail; the dominant source of failure is tool selection under tool explosion.

G Annotation Guidelines

G.1 Human evaluation procedure

This appendix describes the human evaluation used for RealFin. The goal of the questionnaire is to assess the quality of tasks and reference workflows in the dataset. Annotators are instructed to read the guidelines carefully and base their judgments only on the information provided for each item.

For every item, annotators are given:

- the question description
- the golden answer
- the reference program implementing the reference workflow
- tool schema documentation, including tool names, functionality, and input and output fields

Annotators are asked not to rely on any external resources and not to execute the code. All judgments must be made purely from the provided information.

G.2 Rating dimensions

Each item is scored along four dimensions:

1. **Question reasonableness:** clarity, lack of ambiguity, and plausibility within real financial analysis scenarios

2. **Answer correctness:** consistency of the golden answer with the question and whether the value or conclusion is correct or essentially correct

3. **Reference workflow correctness:** without running the code, whether the tools are appropriate, parameters are reasonable, and the sequence of steps is conceptually able to produce the correct answer

4. **Tool schema usability:** clarity and sufficiency of the tool documentation, including input and output meanings, to reasonably support solving the task

G.3 Scoring scheme

All four dimensions use a three-level scale:

2 fully correct or reasonable 1 minor issues but usable 0 clearly incorrect or unusable

Annotators may optionally add short comments explaining their scores. If a judgment truly cannot be made due to insufficient information, annotators may select unable to judge and briefly explain the reason. Annotators complete the questionnaire independently.

G.4 Example annotation item

Table 17: Detailed performance comparison across retrieval strategies and models. **Oracle**: Ground truth tool selection. **BM25**: BM25-based retrieval (top-20). **OraA-k**: Oracle with top-k augmentation. **Full**: Full tool library. Metrics include Accuracy (ACC), Exact Success Rate (ESR), Tool Selection Rate (TSR), Overall Accuracy (OA), Exact Goal Achievement (EGA), Task Goal Achievement (TGA), Partial Pass Score (PPS), Overall Success Rate (OSR), and Retrieval Success Score (RSS).

Strategy	Model	ACC	ESR	TSR	OA	EGA	TGA	PPS	OSR	RSS
Oracle: Ground Truth Tool Selection										
Oracle	Doubao-1.6	0.367	0.703	0.281	0.266	0.344	0.545	0.195	0.047	0.091
Oracle	Kimi-K2	0.305	0.750	0.227	0.203	0.398	0.636	0.141	0.008	0.083
Oracle	Claude-4.5	0.273	0.742	0.227	0.219	0.500	0.568	0.133	0.031	0.099
Oracle	DeepSeek-3.2	0.320	0.734	0.211	0.203	0.539	0.318	0.133	0.055	0.083
Oracle	Gemini-3-Pro	0.148	0.750	0.211	0.203	0.445	0.114	0.125	0.023	0.091
Oracle	GPT-5.1	0.109	0.883	0.133	0.133	0.273	0.318	0.078	0.023	0.058
Oracle	Qwen-Max	0.164	0.797	0.227	0.211	0.414	0.523	0.141	0.016	0.091
BM25: BM25-based Retrieval (Top-20)										
BM25	Doubao-1.6	0.117	0.141	0.648	0.563	0.336	–	0.234	0.078	0.374
BM25	Claude-4.5	0.250	0.203	0.719	0.602	0.586	0.000	0.297	0.141	0.366
BM25	DeepSeek-3.2	0.094	0.172	0.641	0.578	0.586	–	0.266	0.180	0.358
BM25	Gemini-3-Pro	0.100	0.719	0.227	0.195	0.195	0.000	0.078	0.055	0.138
BM25	GPT-5.1	0.064	0.633	0.375	0.352	0.289	0.000	0.117	0.047	0.260
BM25	Kimi-K2	0.047	0.250	0.516	0.484	0.453	–	0.211	0.133	0.309
BM25	Qwen-Max	0.016	0.734	0.211	0.188	0.125	0.000	0.063	0.023	0.154
OraA-k: Oracle with Top-k Augmentation										
OraA-10	Doubao-1.6	0.000	0.000	0.250	0.250	0.690	0.390	0.000	0.000	0.264
OraA-10	Kimi-K2	0.000	0.141	0.164	0.148	0.586	0.082	0.000	0.000	0.157
OraA-10	Claude-4.5	0.266	0.594	0.227	0.211	0.445	0.636	0.133	0.016	0.098
OraA-10	DeepSeek-3.2	0.008	0.000	0.309	0.285	0.699	0.286	0.000	0.000	0.302
OraA-10	Gemini-3-Pro	0.070	0.758	0.063	0.063	0.391	0.114	0.016	0.016	0.049
OraA-10	GPT-5.1	0.043	0.884	0.130	0.130	0.130	0.036	0.029	0.000	0.111
OraA-10	Qwen-Max	0.047	0.711	0.086	0.078	0.078	0.136	0.031	0.000	0.057
OraA-3	Doubao-1.6	0.000	0.000	0.305	0.297	0.711	0.659	0.000	0.000	0.314
OraA-3	Kimi-K2	0.016	0.055	0.172	0.164	0.656	0.500	0.000	0.000	0.174
OraA-3	Claude-4.5	0.367	0.820	0.656	0.609	0.438	0.636	0.484	0.141	0.179
OraA-3	DeepSeek-3.2	0.039	0.000	0.367	0.344	0.719	0.341	0.000	0.000	0.364
OraA-3	Gemini-3-Pro	0.094	0.797	0.203	0.188	0.211	0.159	0.094	0.008	0.114
OraA-3	GPT-5.1	0.086	0.836	0.500	0.438	0.258	0.477	0.352	0.117	0.154
OraA-3	Qwen-Max	0.141	0.742	0.484	0.430	0.336	0.477	0.352	0.109	0.138
OraA-0	Claude-4.5	0.000	0.933	0.062	0.062	0.554	0.000	0.000	0.000	0.000
OraA-0	Qwen-Max	0.102	0.836	0.188	0.164	0.281	0.318	0.109	0.008	0.000
OraA-0	Doubao-1.6	0.422	1.000	0.258	0.258	0.563	0.000	0.148	0.117	0.116
OraA-0	Claude-4.5	0.234	1.000	0.266	0.258	0.594	0.000	0.180	0.172	0.083
OraA-0	DeepSeek-3.2	0.195	1.000	0.234	0.234	0.578	0.000	0.141	0.133	0.099
OraA-0	Gemini-3-Pro	0.383	1.000	0.273	0.273	0.641	0.000	0.172	0.125	0.107
OraA-0	Kimi-K2	0.141	1.000	0.133	0.133	0.398	0.000	0.047	0.023	0.091
OraA-0	GPT-5.1	0.305	1.000	0.242	0.242	0.578	0.000	0.180	0.156	0.066
OraA-0	Qwen-Max	0.234	1.000	0.258	0.250	0.555	0.000	0.148	0.133	0.107
OraA-6	Claude-4.5	0.078	0.750	0.117	0.117	0.641	0.364	0.031	0.008	0.008
OraA-6	Qwen-Max	0.078	0.805	0.109	0.102	0.273	0.273	0.047	0.000	0.000
Full: Complete Tool Library										
Full	Doubao-1.6	0.336	0.758	0.227	0.211	0.164	0.614	0.164	0.008	0.066
Full	Claude-4.5	0.094	0.664	0.086	0.086	0.445	0.159	0.008	0.000	0.085
Full	DeepSeek-3.2	0.305	0.711	0.195	0.188	0.391	0.273	0.086	0.000	0.116
Full	Gemini-3-Pro	0.086	0.688	0.086	0.086	0.211	0.000	0.008	0.000	0.085
Full	GPT-5.1	0.016	0.781	0.094	0.086	0.063	0.023	0.008	0.000	0.094
Full	Kimi-K2	0.102	0.766	0.148	0.141	0.305	0.455	0.078	0.008	0.074
Full	Qwen-Max	0.031	0.500	0.086	0.070	0.109	0.045	0.023	0.000	0.067

Field	Example
Question	*What was the percentage price change of Tencent Holdings in the fourth quarter of 2023? Report the answer as a percentage rounded to two decimals.*
Dataset fields	Type: T3_Aggregate; time sensitivity: Fixed; template: `query_income_statement_yoy_percentage_metric`; reference tool count: 10; reference depth: 3.
Reference workflow	Resolve company name to HK ticker → retrieve adjusted close on 2023-09-30 → retrieve adjusted close on 2023-12-31 → compute holding-period return.
Reference tools	`get_code_from_name`, `get_hk_stock_daily_price`, arithmetic computation.
Reference answer	`-4.11%`.
Valid alternative path	Use `search_kg` or another annotated resolver for ticker grounding, and use an equivalent HK historical-price API if it returns the same adjusted-close fields under the same date-adjustment and front-adjustment rule.
Visible information to agent	Natural-language question, visible tool names, descriptions, schemas, and previous tool outputs. The agent does not receive the golden code or final answer.
Outcome scoring	ACC checks only whether the returned percentage matches the reference answer within the task tolerance. A different valid tool path can receive full ACC.
Process scoring	TSR accepts annotated equivalent tools; OA checks dependency-compatible ordering after substitution; EGA/TGA check entity and date grounding; PPS checks required parameters such as ticker, dates, adjustment mode, and price field; OSR requires the full process to succeed.

Table 18: A representative REALFIN benchmark instance and its scoring logic.

Type	Representative example
Tool selection failure	*Question:* latest Shanghai silver benchmark price at noon. *Gold:* 17140.0. Oracle: calls `get_sge_benchmark_price` and answers correctly. 10-D / full: calls no tool and outputs free-form numbers.
Temporal grounding failure	*Question:* highest price of Suolian Co. 7 trading days ago. *Gold:* 33.56. Correct tool called, but indexes wrong trading day (returns 33.00).
Execution / answer grounding failure	*Question:* latest closing price of Yilite. *Gold:* 14.06. Correct tools called, but model ignores the "close" field and outputs unrelated values.

Table 19: **Representative failure types.** Tool explosion shifts error structure from computation to (i) tool selection, (ii) temporal grounding, (iii) answer grounding.

Table 20: Equity quarterly return SOP vs. RealFin workflow (Tencent Holdings, Q4 2023).

Analyst SOP	RealFin golden workflow
Identify target security	`get_code("Tencent")`.
Determine quarter start and end dates	Use 2023-09-30 and 2023-12-31.
Retrieve adjusted closing prices	`get_price(close)`.
Ensure price comparability	`adjust = qfq`.
Compute period return	$\frac{P_{\text{end}}}{P_{\text{start}}} - 1$.
Format output	Percentage form, two decimals.

Table 21: Cross-sectional liquidity SOP vs. RealFin workflow (U.S. semiconductor "big three").

Analyst SOP	RealFin golden workflow
Define peer group	`search_kg("U.S. semiconductor big three")`.
Obtain constituent tickers	`get_code_from_name` for each member.
Select reference trading day	`query_date = latest`.
Retrieve trading volume	`get_price(volume)`.
Construct cross-sectional sample	Collect volumes across constituents.
Compute group-level statistic	Arithmetic mean of volumes.

Table 22: Precious-metals fixing SOP vs. RealFin workflow (Shanghai gold benchmark, 2024-05-15).

Analyst SOP	RealFin golden workflow
Specify asset and benchmark	`get_price(gold)`.
Choose pricing sessions	`session = morning / evening`.
Fix calendar date	`query_date = 2024-05-15`.
Retrieve benchmark prices	Two calls to `get_sge_benchmark_price`.
Compute intraday difference	$P_{\text{morning}} - P_{\text{evening}}$.
Reporting format	Return price difference.

Table 23: Model-averaged bootstrap results. Values are means with bootstrap standard errors.

Setting	ACC	Exec	Oracle	Prec	Rec
Oracle	.21±.03	.67±.03	.76±.03	1.00±.00	.68±.04
OraA-3	.09±.02	.34±.02	.39±.02	.98±.01	.55±.04
OraA-10	.05±.01	.24±.02	.25±.02	.91±.02	.35±.04
Pkg	.28±.04	.34±.01	.00±.00	.12±.03	.13±.03
BM25*	.13±.03	.60±.04	.64±.04	.99±.01	.43±.04
Full	.12±.02	.49±.03	.00±.00	.00±.00	.00±.00

Table 24: Process-level failure summary. All values are proportions over evaluated samples; the gap between TSR and OSR indicates that the main difficulty is multi-tool orchestration rather than single-tool invocation alone.

Quantity	Rate	Interpretation
Retrieval failure	.72	Gold tools not retrieved
Orchestration failure	.03	Wrong order or composition
Parameter/temporal failure	.17	Wrong date, argument, or anchor
Full* answer success	.08	Correct final answer
TSR	.15	Required tools selected
OSR	.03	Full workflow succeeds

Table 25: Controlled intervention results. Left: prompt length is varied while the tool set is fixed. Right: the number of distractor tools is varied while the prompt context is fixed.

Model	Context	ACC	TSR	OSR	PPS
Claude	Short	.300	.500	.067	.467
Claude	Long	.033	.167	.000	.033
Qwen	Short	.033	.333	.100	.300
Qwen	Long	.000	.133	.000	.067

(a) Short vs. long context; 30 tasks per setting.

Model	k	ACC	TSR	OSR
Qwen	0	.102	.104	.000
Qwen	3	.141	.484	.109
Qwen	6	.078	.094	.000
Qwen	10	.047	.086	.000

(b) Tool-space size intervention with k distractors.

Table 26: Example human evaluation item

Field	Example content
ID	152
Question	In 2023, the year-on-year growth rate of the M2 money supply was 12.6% in January and changed to another value in December. Compute the change between January and December to two decimal places and report it as a percentage.
Golden answer	-1.8%
Reference program	abbreviated for space
Task type	T2_Compare
Time sensitivity	Fixed
Example ratings	Question: 2; Answer: 2; Workflow: 2; Tool schema: 2.