Achieving Multi-Hop Calculation and Safe Abstention in Financial Numerical Reasoning by Metric Graph Constrained LLMs

Aoyuan Jiang[1] Liang Hong[2]* Haoxuan Liu[3] Rui Wang[4]
[1]School of Information Management, Wuhan University, Wuhan, China
[2]School of Artificial Intelligence, Wuhan University, Wuhan, China
[3]Academy of Advanced Interdisciplinary Studies, Wuhan University, Wuhan, China
[4]School of Computer and Artificial Intelligence, Wuhan University of Technology, Wuhan, China
{jiangaoyuan, hong, penguin218}@whu.edu.cn, 339993@whut.edu.cn

Abstract

Financial numerical reasoning demands rigorous adherence to domain-specific logic and precise evidence foundation. However, large language models (LLMs) are prone to forced generation when confronting ambiguous evidence or complex recursive dependencies, often hallucinating values to bridge information gaps. To address this, we propose graph-bounded financial reasoning (GBFR), a neuro-symbolic framework that imposes semantic and structural constraints via a financial metric knowledge graph (FMKG). Unlike sequential generation paradigms, our approach employs a parallel graph-constrained reasoning algorithm that orchestrates specialized operators to simultaneously explore heterogeneous derivation paths of complex financial metrics. Through cross-path verification, the framework aggregates only semantically consistent results, ensuring reasoning is bounded by available context. Crucially, this approach enables safe abstention by distinguishing genuine data absence from retrieval failure, thereby preventing ungrounded fabrication. To evaluate this capability, we further construct counterfactual samples by perturbing entities, times, and metrics to synthesize unanswerable scenarios. Empirical evaluations on standard benchmarks demonstrate that GBFR significantly outperforms state-of-the-art baselines.

1 Introduction

Large language models (LLMs) have demonstrated strong language understanding and reasoning capabilities, achieving strong performance on financial text processing tasks, such as question answering and sentiment analysis (Lee et al., 2025; Wang et al., 2025a; Xing, 2025). However, extending these capabilities from textual comprehension to financial numerical reasoning remains a non-trivial challenge. Unlike open-ended language generation, numerical reasoning in financial contexts requires strict adherence to domain-specific knowledge and multi-step computational logic, where even minor numerical deviations can invalidate the final inference (Srivastava et al., 2024; Zhao et al., 2024; Tang et al., 2025).

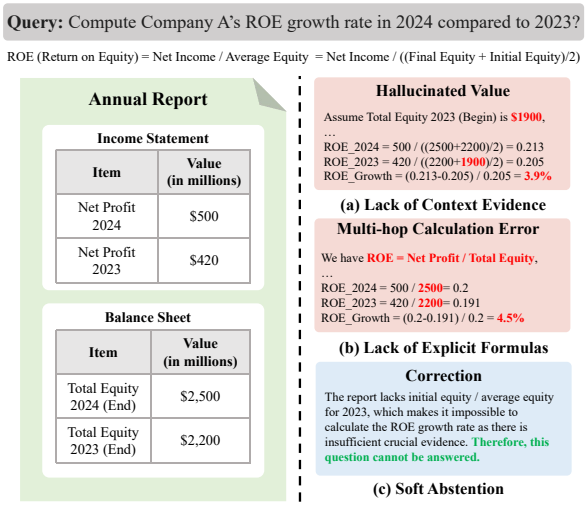

Figure 1: **Examples of two failure modes in LLM-based financial numerical reasoning.**

As illustrated in Figure 1, LLMs exhibit two common failure modes in deterministic financial numerical reasoning. The first is **the lack of clear contextual evidence**. When context is incomplete, the model's bias towards sequence completion often overrides factual faithfulness, leading it to hallucinate plausible-looking numbers to fill the gap. In Figure 1(a), the model hallucinates a value (e.g., $1900) to force a completion. The second is **the lack of explicit calculation formulas**. Without precise arithmetic definitions, LLMs fail to decompose hierarchical metrics (e.g., *ROE*) and frequently apply incorrect formulas (e.g. *ROE = Net Profit / Total Equity* in Figure 1(b)), resulting in calculation errors and reasoning drift during multi-hop inference.

Existing research addresses these issues by aug-

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27575–27595
July 2-7, 2026

menting LLMs with external knowledge or structured prompting (Srivastava et al., 2024). By leveraging diverse knowledge augmentation techniques like function retrieval and LLM-as-retrieval-judge (Tang et al., 2025), RAG-based frameworks incorporate financial formulas into prompts as explicit contextual knowledge, enabling models to seek evidence and follow logical rules. Concurrently, reasoning capabilities are enhanced via chain-of-thought (CoT), which elucidates intermediate steps (Wei et al., 2022), and program-of-thought (PoT), which executes generated programs for precise calculation (Chen et al., 2022a; Khatuya et al., 2025). While integrating these paradigms improves performance, they operate primarily as soft semantic guidelines rather than hard symbolic constraints. These approaches fall short of explicitly instructing the model on which evidence should be verified and how to structurally instantiate domain knowledge into calculation, failing to guarantee adherence to the rigorous logic required for financial analysis.

Fundamentally, precise numerical reasoning remains challenging because it involves two core difficulties. (1) **Ambiguity of Evidence Boundaries**. In real-world financial corpora, the concept of 'missing value' is highly ambiguous: it may be truly absent, expressed via domain aliases (e.g., *Net Profit* for *Net Income*), or derivable from other reported numbers. Consequently, determining when to abstain is often more challenging than answering. A system can only conclude a query is 'unanswerable' after checking all possible calculation paths, alias mappings, and derivation dependencies. (2) **Structural Complexity of Reasoning**. Financial indicators typically form intricate directed acyclic graphs (DAGs) with hierarchical dependencies, requiring multi-step recursive operations. This presents a fundamental topological challenge for Transformer-based LLMs, which are constrained by a sequential modeling paradigm. Such structural mismatch limits the model's ability to maintain global state tracking across deep dependency trees, resulting in omitted steps and error propagation during complex multi-hop reasoning.

To address these challenges, we introduce **graph-bounded financial reasoning (GBFR)**, a neuro-symbolic framework that combines the semantic generalization of LLMs with the structural constraints of a financial metric knowledge graph (FMKG). Unlike sequential reasoning paradigms, our parallel graph-constrained reasoning algorithm orchestrates atomic operators, including query analysis, evidence seeking, problem decomposition, and program execution, to explore heterogeneous derivation paths for complex financial metrics. To mitigate evidence ambiguity, the framework leverages parallel execution to handle heterogeneous evidence forms, including direct mentions, aliases, and derived values. This structured exploration further enables safe abstention: by verifying all potential logical paths, the system distinguishes missing data from retrieval failure, thereby improving inference faithfulness. To evaluate this abstention capability, we employ a multi-dimensional counterfactual sample generation strategy that alters entities, times, and metrics in existing datasets to synthesize unanswerable scenarios. Experiments show that our framework achieves SOTA performance, with an average accuracy rate of 93.92% across different datasets. Our code and data are publicly available[1].

Our contributions are summarized as follows:

- We present a **neuro-symbolic framework GBFR** that leverages an FMKG to constrain LLM reasoning, ensuring rigorous adherence to domain-specific calculation logic.
- We develop a **parallel graph-constrained reasoning algorithm** that orchestrates specialized operators to simultaneously explore diverse derivation paths. By integrating cross-path verification, the method aggregates semantically consistent results to guarantee inference faithfulness.
- We enhance evaluation robustness by **constructing multi-dimensional counterfactual samples** from existing benchmarks to test safe abstention. Extensive experiments demonstrate the effectiveness of our framework, which achieves state-of-the-art performance across multiple datasets.

2 Related Work

2.1 Finance Numerical Reasoning

Financial numerical reasoning requires models to comprehend unstructured texts and perform precise calculations to derive metrics. Early approaches primarily fine-tuned pre-trained language models to extract values and predict arithmetic operators, as seen in benchmarks like FinQA (Chen et al., 2021) and ConvFinQA (Chen et al., 2022b). With the advent of LLMs, the paradigm has shifted towards in-context learning and CoT prompting (Wei et al., 2022). To address the arithmetic limitations

[1] https://github.com/aolaoban/Graph-Bounded_Financial_Reasoning

of LLMs, program-aided language models (Gao et al., 2023) and PoT (Chen et al., 2022a; Khatuya et al., 2025) decouple reasoning from computation by generating executable code instead of direct numerical answers.

Despite these advancements, financial numerical reasoning remains a knowledge-intensive task that requires the rigorous application of explicit metric definitions and hierarchical calculation dependencies. While RAG-based approaches (Lewis et al., 2020; George et al., 2025; Wang et al., 2025b) incorporate external knowledge, they exhibit a fundamental limitation: they treat financial formulas merely as unstructured semantic context rather than strict symbolic constraints. Consequently, these models fail to enforce logical validity during multi-hop inference (Ren et al., 2025). Furthermore, current systems are optimized for answer generation rather than safe abstention, leading to hallucinations when evidence is ambiguous or missing (Wen et al., 2025). Unlike these works, our GBFR framework introduces a graph-constrained search process to ensure that reasoning is strictly bounded by available evidence.

2.2 Neuro-Symbolic Reasoning with LLMs and KGs

The integration of LLMs with knowledge graphs (KGs) aims to combine the semantic generalization ability of language models with the factual precision of structured knowledge (Pan et al., 2024). Existing neuro-symbolic frameworks generally fall into two categories: structure-augmented generation and structure-guided reasoning. The former retrieves KGs triples to augment the context of LLM, mitigating factual hallucinations in open-domain QA (He et al., 2024). The latter employs the graph structure to guide the reasoning process itself. For instance, methods like graph-of-thought (Besta et al., 2024) and reasoning-on-graph approaches (Luo et al., 2024) model the inference path as a traversal over the KG, enabling interpretable multi-hop reasoning.

However, most existing neuro-symbolic systems are designed for factual or commonsense reasoning (Feng et al., 2023; Sun et al., 2023; Jin et al., 2024), where dependencies are relational rather than computational. As a result, they lack the specialized mechanisms to handle the hierarchical calculation logic inherent in financial analysis. In contrast, our GBFR framework treats reasoning as a recursive calculation process, orchestrating specialized operators over an FMKG to ensure arithmetic precision.

3 Preliminaries

This section formalizes the financial reasoning task, presents the schema of FMKG, and describes the construction of our adversarial benchmarks for evaluating the model's safe abstention capability.

3.1 Task Definition

Given a financial document $\mathcal{D}$ and a query Q asking for a specific financial metric, the goal is to derive a grounded numeric answer $y \in \mathbb{R}$ or a specific abstention token $\perp$.

Unlike standard QA tasks, we impose an evidence-bounded constraint: the system must output y if and only if there exists a valid derivation path grounded in $\mathcal{D}$; otherwise, it must output $\perp$ whether due to missing values or ambiguous context. Formally, the objective is to learn a function $f_\theta(Q, \mathcal{D}, \mathcal{G}) \rightarrow \mathbb{R} \cup \{\perp\}$, where $\mathcal{G}$ denotes the FMKG.

3.2 Financial Metric Knowledge Graph

Financial metrics exhibit inherent dependency structures: a target metric is computed from other variables, and some formulas can be algebraically rearranged to solve for different variables. Such interdependence can complicate dependency ordering and make recursive decomposition ambiguous. To make decomposition well-defined and impose structural constraints on the reasoning process, we construct the FMKG, formalized as a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \Phi, \mathcal{A})$.

- **Nodes ($\mathcal{V}$)**: The set of financial variables is partitioned into two disjoint subsets: atomic nodes ($\mathcal{V}_{atomic}$), representing irreducible fundamental concepts (e.g., *Stock Price*) that serve as terminal units in the dependency graph; and derived nodes ($\mathcal{V}_{derived}$), representing complex metrics defined by calculation logic (e.g., *ROE*).
- **Edges ($\mathcal{E}$) and Formulas (Φ)**: A directed edge $(d, a) \in \mathcal{E}$ signifies that node d structurally depends on node a (derived or atomic). To accommodate the diverse calculation ways in finance, each derived node $d \in \mathcal{V}_{derived}$ is associated with a set of candidate formulas $\Phi(d) = \{(\phi_d^{(k)}, \mathcal{S}_d^{(k)})\}_{k=1}^{K_d}$, where each tuple represents a valid calculation path: $\phi_d^{(k)}$ denotes a specific deterministic function, and $\mathcal{S}_d^{(k)} \subseteq \{a \mid (d, a) \in \mathcal{E}\}$ denotes the corresponding subset of child nodes required for that path. A derived node

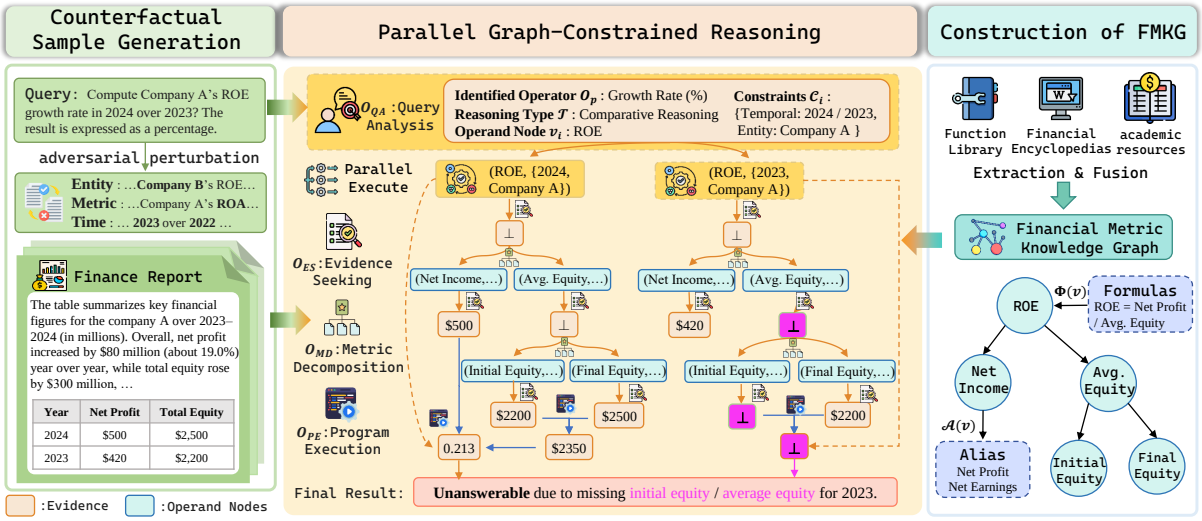

Figure 2: **Overview of GBFR framework**. The left panel illustrates the generation of counterfactual samples via adversarial perturbation. The right panel outlines the construction pipeline and topology of FMKG. The central panel details the parallel graph-constrained reasoning algorithm. By orchestrating four atomic operators ($\mathcal{O}_{QA}$, $\mathcal{O}_{ES}$, $\mathcal{O}_{MD}$, $\mathcal{O}_{PE}$), the framework transforms unstructured queries into verifiable execution plans, enforcing explicit evidence boundaries to enable precise calculation or safe abstention ($\perp$) when evidence is missing.

v is considered resolved if and only if at least one candidate formula $\phi_d^{(k)}$ can be successfully instantiated and computed.

- **Alias Mappings** ($\mathcal{A}$): To resolve semantic ambiguity, each node v is mapped to a set of linguistic aliases $\mathcal{A}(v) = \{s_1, s_2, \dots\}$ (e.g., $\mathcal{A}(\textit{Net Income}) = \{\textit{"Net profit"}, \textit{"Net Earnings"}\}$), serving as the alignment bridge between symbolic nodes and unstructured text spans in $\mathcal{D}$.

Scope and Construction. We construct the FMKG by synthesizing knowledge from established function library (Tang et al., 2025) with information retrieved from authoritative financial encyclopedias (e.g., *Wikipedia*, *Investopedia*) and unstructured academic resources. Through a rigorous hybrid pipeline combining semantic clustering, LLM-assisted completion, and expert verification, we curated a comprehensive network of financial indicators. Each derived metric has aliases, definition, formulas, and code implementations, while atomic metric only has aliases and definition. In total, the FMKG has 7521 total nodes, consisting of 3318 atomic metrics and 4203 derived metrics, and 20152 dependency edges. Detailed construction protocols, quantitative validation, representative node examples, and a visualization of the subgraph topology are provided in Appendix A.

3.3 Counterfactual Sample Generation

Existing benchmarks suffer from a positive bias, which encourages models to hallucinate answers even when evidence is insufficient. To evaluate abstention performance of model, we employ a multi-dimensional counterfactual sample generation strategy on existing datasets. Inspired by NeoQA (Glockner et al., 2025), we adapt the characterization of unanswerable questions to financial reasoning setting and categorize them into two types: (1) missingness, where the target metric or a required operand is absent from the context (e.g., $Target = A - B$ but B is not provided), and (2) mismatch, where related evidence exists but does not satisfy the query constraints, such as entity mismatch (*consolidated* vs. *parent-only*), temporal mismatch (*FY2023* reported but *FY2022* queried), and metric mismatch (*Adjusted EBITDA* provided while *standard EBITDA* is required). Accordingly, our counterfactual samples are designed as context-related traps via perturbations to entities, time, and metrics:

- **Entity Mutation**: Replacing the target company entity in the query with an irrelevant competitor appearing in the corpus.
- **Temporal Shift**: Altering the fiscal year or reporting period (e.g., changing "2023" to "2022") where the corresponding data is absent.
- **Metric Substitution**: Swapping the requested metric with a semantically similar but distinct

metric (e.g., *ROE* vs. *ROA*).

These perturbations create realistic traps, requiring the model to verify query constraints and correctly abstain by outputting $\perp$. More details are provided in Appendix B.

4 Graph-Bounded Financial Reasoning

In this section, we introduce GBFR, our neuro-symbolic framework for financial reasoning. As illustrated in Figure 2, GBFR converts unstructured financial queries into verifiable execution plans. At its core, the framework decomposes reasoning into four specialized atomic operators, each corresponding to a distinct capability from intent analysis to program execution. These operators are dynamically orchestrated by a parallel graph-constrained reasoning algorithm, which enforces structural constraints to bound the generation process within available evidence.

4.1 Atomic Operators

Query Analysis Operator ($\mathcal{O}_{QA}$). Financial numerical reasoning involves diverse reasoning patterns, ranging from simple arithmetic to domain-specific multi-step derivations, and often involves fine-grained constraints on entities, time, and precision. To support executable downstream reasoning, we define $\mathcal{O}_{QA}$ as a symbolic execution planner that parses unstructured queries into structured logical forms. The resulting execution plan consists of the reasoning type, which governs the subsequent reasoning procedure; the function, which specifies the final aggregation logic; the sub-goals, which define the evidence required for computation; and the precision, which determines the output format when needed. Within this plan, we categorize financial queries into five reasoning types: direct retrieval, arithmetic calculation (e.g., *sum*, *difference*), comparative reasoning (e.g., *range*, *ratio*), statistical aggregation (e.g., *average*, *counting*), and temporal reasoning (e.g., *calculating fiscal durations*). Different reasoning types induce different sub-goals and final aggregation operators. The full mapping from reasoning types to symbolic semantics is provided in Appendix C.

Given a document $\mathcal{D}$ and a query Q, the operator maps the query to a specific reasoning category $\mathcal{T}$ and parses the required logic:

$$\mathcal{O}_{QA}(\mathcal{D}, Q) \rightarrow (\mathcal{T}, Op, \{(v_i, \mathcal{C}_i)\}_{i=0}^{N}, P) \quad (1)$$

where Op denotes the identified function (e.g., Average), P indicates the precision of the final answer, and $(v_i, \mathcal{C}_i)$ denotes a sub-goal representing an operand node and its associated constraints. These constraints $\mathcal{C}_i$ are used to resolve ambiguity and specify: (1) temporal scope, including the fiscal year, quarter, or specific timestamp (e.g., *2023 Q4*, *Dec 31*); (2) entity scope, including the target corporate entity or subsidiary (e.g., *Parent Company*, *Group*); (3) meta-attributes, including the required currency, scale, and unit (e.g., *USD*, *billions*, *percentage*). The prompt of $\mathcal{O}_{QA}$ is shown in Appendix F.

Evidence Seeking Operator ($\mathcal{O}_{ES}$). A core challenge in financial numerical reasoning is evidence ambiguity: metrics may appear under various aliases or be entirely absent. In unstructured financial text, neither keyword matching nor semantic retrieval is sufficient to reliably resolve this ambiguity. Therefore, $\mathcal{O}_{ES}$ is designed to identify supporting evidence from the context. Given structured constraints $\mathcal{C}$, $\mathcal{O}_{ES}$ leverages the LLM's semantic capabilities together with alias mappings in FMKG to locate supporting evidence. If the context does not contain a value that satisfies $\mathcal{C}$, we instruct the model to return $\perp$ and report missing metrics. This defines $\perp$ as an explicit state that triggers either metric decomposition for derived nodes or path termination for atomic nodes.

For a specific node v and constraints $\mathcal{C}$, the operator retrieves evidence from document $\mathcal{D}$ using aliases $\mathcal{A}(v)$.

$$\mathcal{O}_{ES}(v, \mathcal{C}, \mathcal{D}, \mathcal{A}(v)) \rightarrow val \cup \{\perp\} \quad (2)$$

Here, val represents the verified numerical evidence. Crucially, if the evidence is missing, ambiguous, or inconsistent with the constraints, the operator returns the explicit abstention token $\perp$. The prompt template for $\mathcal{O}_{ES}$ is shown in Appendix F.

Metric Decomposition Operator ($\mathcal{O}_{MD}$). Complex financial metrics are often not reported directly and must instead be derived from other variables. Relying on the parametric knowledge of LLMs to plan such multi-step derivations can lead to logical hallucinations and missing steps. To address this, $\mathcal{O}_{MD}$ functions as a logical guide that transforms the reasoning problem from unstructured text generation into graph-structured symbolic search.

Given a derived node v, the operator accesses the formula set Φ to retrieve all candidate derivation paths.

$$\mathcal{O}_{MD}(v, \Phi) \rightarrow \{(path_k, \mathcal{S}^{(k)}_{children})\}_{k=1}^{K} \quad (3)$$

Algorithm 1: Parallel Graph-Constrained Reasoning with Cross-Path Verification

Input : Document $\mathcal{D}$, FMKG $\mathcal{G}$, Query Q
Output : Verified Answer y or Abstention $\perp$

// Phase 1: Semantic Analysis
$(\mathcal{T}, Op, \mathcal{S}_{goals}) \leftarrow \mathcal{O}_{QA}(Q)$;
$V_{results} \leftarrow \emptyset$;
// Phase 2: Parallel Resolution of Sub-goals
parallel foreach $(v, \mathcal{C}) \in \mathcal{S}_{goals}$ **do**
  $val \leftarrow \texttt{Resolve}(v, \mathcal{C})$;
  if $val = \perp$ **then**
    return $\perp$
  Add val to $V_{results}$;
// Phase 3: Final Aggregation
return Apply operator Op on $V_{results}$;

Function *Resolve*$(v, \mathcal{C})$**:**
  $v_{final} = \mathcal{O}_{ES}(v, \mathcal{C}, \mathcal{D}, \mathcal{A}(v))$;
  if $v_{final} = \perp$ & $v \in \mathcal{V}_{atomic}$ **then**
    return $\perp$
  if $v_{final} \neq \perp$ **then**
    return v_{final}
  else
    $Paths \leftarrow \mathcal{O}_{MD}(v, \Phi)$ $\mathcal{V}_{candidates} \leftarrow \emptyset$;
    // Step A: Simultaneous Execution
    parallel foreach $(path_k, \mathcal{C}_{children}) \in Paths$ **do**
      $ChildVals \leftarrow \emptyset$;
      $isPathValid \leftarrow$ **true**;
      parallel foreach $u \in \mathcal{C}_{children}$ **do**
        $val_u \leftarrow \texttt{Resolve}(u, \mathcal{C})$;
        if $val_u = \perp$ **then**
          $isPathValid \leftarrow$ **false**;
        Add val_u to $ChildVals$;
      if $isPathValid$ **then**
        $res_k \leftarrow \mathcal{O}_{PE}(path_k, ChildVals)$;
        Add res_k to $\mathcal{V}_{candidates}$;
    // Step B: Cross-Path Verification
    if $\mathcal{V}_{candidates} \neq \emptyset$ **then**
      $v_{final} \leftarrow \texttt{Verify}(\mathcal{V}_{candidates})$;
      return v_{final}
    return $\perp$

where each $path_k$ contains a symbolic formula and $\mathcal{S}^{(k)}_{children}$ represents the set of operand nodes required to resolve v.

To ensure comprehensive coverage of feasible derivation paths, $\mathcal{O}_{MD}$ retrieves candidate paths from the FMKG through bidirectional search. It primarily performs downward traversal to obtain decomposition paths through the child nodes of the target metric, while also allowing limited upward exploration over 1-hop parent nodes. Downward paths provide direct decomposition formulas for subsequent execution. Upward paths are used only when a parent formula can be algebraically rearranged to solve for the target metric and the required variables are available in the context. In both cases, the resulting formula and evidence are passed to $\mathcal{O}_{PE}$ for executable code generation and final computation.

Program Execution Operator ($\mathcal{O}_{PE}$). Even with correct reasoning, LLMs are prone to arithmetic hallucinations (e.g., calculation errors, unit mismatches) when processing numerical tokens directly. To address this, $\mathcal{O}_{PE}$ functions as a deterministic interpreter, translating the symbolic path ϕ and verified evidence $\{val_i\}$ into an executable programs (e.g., Python). Formally, the execution is defined as:

$$\mathcal{O}_{PE}(\phi, \{val_1, val_2, \dots\}) \rightarrow y \tag{4}$$

The operator executes the program to yield the final result y. If any input value in the set is $\perp$, execution halts and $\perp$ is propagated. The prompt of $\mathcal{O}_{PE}$ is shown in Appendix F.

4.2 Parallel Graph-Constrained Reasoning Algorithm

While the atomic operators serve as the fundamental execution units, the complexity of financial reasoning arises from the existence of multiple valid derivation paths for a single target metric within the FMKG (e.g., *Earnings Before Interest and Taxes* can be derived via revenue-based or profit-based formulas). Relying on a single derivation path often leads to failure due to incomplete evidence. To address this issue, the parallel graph-constrained reasoning algorithm (as shown in Algorithm 1) is proposed to orchestrate a comprehensive search for the most evidence-supported solution. The complexity analysis and empirical cost of the algorithm is presented in Appendix D.

However, obtaining a numerical result does not guarantee semantic correctness; different paths may rely on spurious evidence alignments or produce numerically identical values under incorrect contexts. To enforce faithfulness, we implement a cross-path verification mechanism. Specifically, $Verify(V_{\text{candidates}})$ is realized as a single-call LLM judge conditioned on an instantiated trace (the prompt is shown in Appendix F), including the logical constraints (entity, time, and meta attributes), evidence-grounded variable bindings $(name, value, evidence\_snippet)$, the applied derivation path, and the resulting value. The

Table 1: **Comparative results on financial numerical reasoning tasks.** Metrics include Total Accuracy (Total Acc.) for overall performance and Abstention Accuracy (Abst. Acc.) for safety evaluation. Best results are highlighted in green.

Model	Methods	FinQA		TAT-QA		FinanceReasoning		CFBenchmark	
		Total Acc.	**Abst. Acc.**	**Total Acc.**	**Abst. Acc.**	**Total Acc.**	**Abst. Acc.**	**Total Acc.**	**Abst. Acc.**
GPT-5	PoT	79.07	93.81	94.49	**98.68**	88.26	79.37	90.07	**100.00**
	RAG	79.80	89.69	93.70	94.70	88.57	77.30	88.08	**100.00**
	LLM as Retri.	80.35	87.63	94.02	95.36	88.45	73.06	89.40	**100.00**
	LLM-Instructed Retri.	79.53	88.66	94.25	96.03	88.26	74.03	90.73	**100.00**
	ReAct	78.56	90.72	92.99	96.03	83.09	73.54	80.13	**100.00**
	GBFR	**84.98**	**98.80**	**95.75**	96.69	**98.26**	**100.00**	**96.69**	**100.00**
QWEN3-14B	PoT	76.52	62.89	91.26	75.50	77.70	51.21	82.12	90.91
	RAG	73.07	49.48	90.47	72.85	73.89	38.83	77.48	54.55
	LLM as Retri.	74.34	50.52	88.90	68.87	74.19	39.08	82.12	63.64
	LLM-Instructed Retri.	74.98	51.55	88.35	68.87	73.40	40.53	78.14	63.64
	ReAct	76.06	59.79	82.05	80.13	74.87	41.57	76.23	90.91
	GBFR	76.73	72.16	91.50	80.79	84.89	83.25	86.75	81.82

judge outputs a VALID/INVALID decision with a short justification. A candidate is considered invalid if its operands are not grounded in evidence, if the evidence violates the specified constraints, or if the computation contains arithmetic errors or missing operands. The final answer is then determined via consensus voting among the validated traces. If no candidate passes, the system triggers safe abstention, thereby distinguishing genuine data absence from reasoning failure.

5 Experiments

Our evaluation is guided by the following four research questions:

- **RQ1:** How does GBFR compare against state-of-the-art baselines in handling complex financial numerical reasoning tasks?
- **RQ2:** How effectively can LLMs abstain from answering unanswerable financial questions, and can our framework improve this capability?
- **RQ3:** What are the individual contributions of the structural components to the overall performance?
- **RQ4:** To what extent is the proposed framework robust across different LLM backbones?

5.1 Experimental Setup

Dataset. We evaluate GBFR on four standard financial reasoning datasets: FinQA (Chen et al., 2021), TAT-QA (Zhu et al., 2021), FinanceReasoning (Tang et al., 2025), and CFBenchmark (Lei et al., 2023). To incorporate adversarial scenarios, we augmented the evaluation data by subjecting a random 10% subset of samples from each dataset to our counterfactual sample generation strategy, thereby creating a composite benchmark of answerable and unanswerable queries.

Baselines. We compare GBFR against four state-of-the-art baselines: (1) PoT prompting (Wei et al., 2022; Chen et al., 2022a; Khatuya et al., 2025); (2) knowledge augmentation: Standard RAG, LLM as retrieval judge, LLM-instructed knowledge retrieval (Lewis et al., 2020; Guan et al., 2024b; Tang et al., 2025); (3) ReAct (Yao et al., 2023). All experiments employ GPT-5 (OpenAI, 2025) and Qwen3-14B (Yang et al., 2025) as the backbone models. The LLM generates Python code, which is then executed in the same sandboxed Python environment to eliminate arithmetic errors.

Evaluation metrics. To comprehensively assess both reasoning performance and safety, we employ two key metrics: (1) **Total Accuracy (Total Acc.)**: This measures the overall performance on the entire test set, encompassing both answerable and unanswerable queries. Unlike simple string matching, we employ a normalization-based comparison to handle unit and precision variations. If a query explicitly dictates precision (e.g., "to three decimal places"), we require an exact match at that resolution (e.g., 0.213 does not match 0.214). If precision is unspecified, we round both the executor's output and the ground-truth to the same decimal precision before comparison (e.g., 0.213 matches 0.21 after rounding to two decimal places). (2) **Abstention**

Accuracy (Abst. Acc.): This metric specifically assesses the model's robustness against adversarial inputs. It is defined as the recall rate on the unanswerable subset, calculating the proportion of invalid queries where the model correctly triggers the designated abstention token.

Detailed statistics of datasets, baseline implementations are provided in **Appendix E**.

5.2 Main Result (RQ1)

Table 1 reports the main results on four financial numerical reasoning benchmarks. GBFR achieves the strongest overall performance across both total accuracy and abstention accuracy, outperforming prompting-based, knowledge-augmented, and tool-augmented baselines under both GPT-5 and Qwen3-14B. With GPT-5, GBFR obtains the best total accuracy on all four benchmarks. The largest gain is on FinanceReasoning, where it improves over the strongest baseline by 9.69 points. The same trend is observed with Qwen3-14B. In terms of abstention accuracy, GBFR delivers the best or highly competitive results across benchmarks, showing that it improves both answer generation and reliable abstention. We have conducted significance testing using McNemar's Chi-squared Test on the results. The performance improvements of GBFR over the best-performing baseline are statistically significant on all four benchmarks (all p-values < 0.01).

We obtain three main observations from these results. First, retrieval-based knowledge augmentation does not consistently improve reasoning performance. In several cases, RAG and LLM-based retrieval underperform PoT, suggesting that unstructured retrieval introduces semantically related but constraint-incompatible evidence. Second, although ReAct incorporates tool use, it still relies on step-by-step interaction and lacks an explicit reasoning structure over metric dependencies. Third, GBFR achieves the best results by grounding reasoning in FMKG-guided decomposition, explicit evidence constraints, and verifiable execution paths over all benchmarks.

5.3 Analysis on Abstention Capability (RQ2)

To answer RQ2, we evaluate the ability of different methods to abstain from answering unanswerable financial questions using counterfactual samples. These samples introduce semantically related but constraint-incompatible evidence, requiring the model to distinguish unsupported queries from answerable ones.

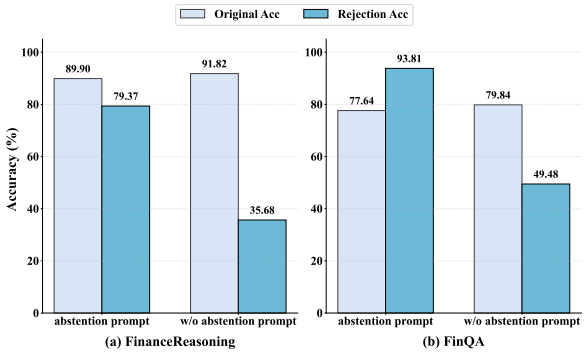

Figure 3: Comparison of model performance under different prompt settings in counterfactual samples.

Retrieval noise undermines abstention. Table 1 shows that knowledge augmentation methods do not reliably improve abstention performance. Instead, retrieval can introduce noisy context that is semantically related to the query but violates entity, temporal, or metric constraints, thereby encouraging unsupported answering rather than rejection. This effect is particularly clear for Qwen3-14B: on FinQA, abstention accuracy drops from 62.89% with PoT to 49.48% with RAG, and on FinanceReasoning, it drops from 51.21% to 38.83%. These results indicate that naive retrieval makes it harder for the model to correctly abstain on unanswerable questions.

Prompt-based abstention is unstable. Figure 3 shows that the abstention behavior of LLMs depends heavily on explicit abstention prompts. Without the abstention prompt, rejection accuracy drops sharply, from 79.37% to 35.68% on FinanceReasoning and from 93.81% to 49.48% on FinQA. This indicates that abstention is not stably grounded in evidence verification. Although adding the prompt improves rejection accuracy, it also lowers accuracy on answerable questions, from 91.82% to 89.90% on FinanceReasoning and from 79.84% to 77.64% on FinQA. This trade-off suggests that simple prompting improves abstention accuracy by steering the model toward refusal, but at the cost of making it overly conservative.

GBFR enables more reliable abstention. In contrast, GBFR makes abstention decisions through evidence and reasoning paths rather than prompt. As shown in Table 1, with Qwen3-14B, GBFR achieves 72.16% abstention accuracy on FinQA and 83.25% on FinanceReasoning. These results show that GBFR provides a more reliable

Table 2: Ablation study on GBFR. Numbers in parentheses denote absolute drops compared to the full model.

Variant	Total Acc.	Abst. Acc.
w/o evidence constraints	79.89 (↓5.09)	85.57 (↓13.23)
w/o graph decomposition	77.89 (↓7.09)	89.69 (↓9.11)
w/o program execution	81.48 (↓3.5)	94.85 (↓3.05)
GBFR (full)	**84.98**	**98.8**

mechanism for rejecting unanswerable financial queries.

5.4 Ablation Study (RQ3)

To confirm the contribution of each component in the GBFR framework, we evaluate three variants by replacing key modules on FinQA.

- ***w/o evidence constraints***: Removes explicit constraints ($\mathcal{C}$) and aliases ($\mathcal{A}$), forcing the model to autonomously align evidence based on semantic relevance within the context.
- ***w/o graph decomposition***: Removes recursive decomposition of dependent sub-metrics, providing only the direct formula for the target metric and its dependencies.
- ***w/o program execution***: Removes external Python executor, reverting to end-to-end textual computation in which arithmetic operations are performed directly by the LLMs.

As shown in Table 2, removing any component leads to clear performance drops, confirming that all modules are essential for the framework. The *w/o evidence constraints* variant suffers the most severe decline in abstention accuracy (↓13.23%), which verifies that explicit boundary conditions are important for distinguishing genuine data absence from retrieval failure. Conversely, the *w/o graph decomposition setting* shows the most significant decline in total accuracy (↓7.09%), underscoring the instability of implicit linear reasoning in handling complex, recursive dependencies without structural guidance. The *w/o program execution* variant results in a consistent accuracy drop. This confirms that offloading arithmetic tasks to a code interpreter is necessary to avoid the calculation errors common in direct text generation.

5.5 Model Robustness Analysis (RQ4)

To address data privacy requirements in financial scenarios, we evaluate GBFR on open-source LLMs suitable for local deployment, scaling from 14B (Qwen3) to 671B (DeepSeek-V3.2)(DeepSeek-AI, 2024; GLM et al., 2024; Yang et al., 2025). As shown in Table 3, our framework demonstrates stable performance on the dataset FinQA. Despite a 47× reduction in model size, the lightweight Qwen3-14B retains 97.7% of the performance achieved by the massive DeepSeek-V3.2 (76.73% vs.78.53%). This slight performance degradation confirms that GBFR's symbolic scaffolding effectively compensates for the reduced reasoning capacity of smaller models, making it a viable solution for resource-constrained, privacy-sensitive applications.

Table 3: Performance comparison across open-source models of varying sizes on FinQA.

Model	Size	Total Acc.	Abst. Acc.
Deepseek V3.2	671B	78.53	82.47
GLM 4.6	355B	77.43	76.80
Qwen3-next	80B	77.34	78.35
Qwen3-14B	14B	76.73	72.16

6 Conclusion

In this paper, we introduce GBFR, a neuro-symbolic framework for addressing the challenges of evidence ambiguity and structural complexity in financial numerical reasoning. By integrating the FMKG with a parallel graph-constrained reasoning algorithm, our approach bridges the gap between flexible semantic understanding and rigorous domain logic. Empirical evaluations demonstrate that GBFR significantly outperforms baselines on several complex numerical reasoning benchmarks. Our analysis further shows that strict constraint grounding and cross-path verification effectively reduce hallucinations. These findings highlight the importance of symbolic constraints in high-stakes financial reasoning.

Limitations

Despite the superior performance of GBFR in financial numerical reasoning, we acknowledge two primary limitations. (1) Graph Dependency: The reasoning scope is bounded by FMKG coverage; metrics absent from the FMKG trigger default abstention, requiring continuous updates for niche indicators. (2) Token Overhead: While parallel execution optimizes latency, the multi-path exploration and cross-path verification incur higher token costs compared to PoT.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 72474163) and the Intelligent Computing Center of the National Cybersecurity Talent and Innovation Base, Wuhan.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022a. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, and 1 others. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022b. 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*, pages 6279–6292.

DeepSeek-AI. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Chao Feng, Xinyu Zhang, and Zichu Fei. 2023. Knowledge solver: Teaching llms to search for domain knowledge from knowledge graphs. *arXiv preprint arXiv:2309.03118*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Ryan George, Akshay Govind Srinivasan, Jayden Koshy Joe, Harshith MR, Hrushikesh Kant, Rahul Vimalkanth, Sudharshan Suresh, and 1 others. 2025. Enhancing financial rag with agentic ai and multi-hyde: A novel approach to knowledge retrieval and hallucination reduction. In *Proceedings of The 10th Workshop on Financial Technology and Natural Language Processing*, pages 19–32.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, and 37 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Max Glockner, Xiang Jiang, Leonardo FR Ribeiro, Iryna Gurevych, and Markus Dreyer. 2025. Neoqa: Evidence-based question answering with generated news events. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11842–11926.

Jian Guan, Wei Wu, Zujie Wen, Peng Xu, Hongning Wang, and Minlie Huang. 2024a. Amor: A recipe for building adaptable modular knowledge agents through process feedback. In *Advances in Neural Information Processing Systems*, volume 37, pages 126118–126148. Curran Associates, Inc.

Jian Guan, Wei Wu, Peng Xu, Hongning Wang, Minlie Huang, and 1 others. 2024b. Amor: A recipe for building adaptable modular knowledge agents through process feedback. *Advances in Neural Information Processing Systems*, 37:126118–126148.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and 1 others. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 163–184.

Subhendu Khatuya, Shashwat Naidu, Pawan Goyal, and Niloy Ganguly. 2025. Program of thoughts for financial reasoning: Leveraging dynamic in-context examples and generative retrieval. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 30994–31006.

Jean Lee, Nicholas Stevens, and Soyeon Caren Han. 2025. Large language models in finance (finllms). *Neural Computing and Applications*, pages 1–15.

Yang Lei, Jiangtong Li, Dawei Cheng, Zhijun Ding, and Changjun Jiang. 2023. Cfbenchmark: Chinese financial assistant benchmark for large language model. *arXiv preprint arXiv:2311.05812*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *Preprint*, arXiv:2501.05366.

L Luo, YF Li, G Haffari, and S Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *ICLR 2024: The Twelfth International Conference on Learning Representations*. ICLR.

OpenAI. 2024. text-embedding-3-large.

OpenAI. 2025. Gpt-5.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2025. Investigating the factual knowledge boundary of large language models with retrieval augmentation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3697–3715.

Pragya Srivastava, Manuj Malik, Vivek Gupta, Tanuja Ganu, and Dan Roth. 2024. Evaluating llms' mathematical reasoning in financial document question answering. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3853–3878.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.

Zichen Tang, E Haihong, Ziyan Ma, Haoyang He, Jiacheng Liu, Zhongjun Yang, Zihua Rong, Rongjin Li, Kun Ji, Qing Huang, and 1 others. 2025. Financereasoning: Benchmarking financial numerical reasoning more credible, comprehensive and challenging. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15721–15749.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. 2025. Plan*rag: Efficient test-time planning for retrieval augmented generation. *Preprint*, arXiv:2410.20753.

Xinyu Wang, Jijun Chi, Zhenghan Tai, Tung Sum Thomas Kwok, Hailin He, Zhuhong Li, Yuchen Hua, Muzhi Li, Peng Lu, Suyucheng Wang, Yihong Wu, Huang Jerry, Jingrui Tian, Fengran Mo, Yufei Cui, and Ling Zhou. 2025a. Finsage: A multi-aspect rag system for financial filings question answering. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, page 6144–6152, New York, NY, USA. Association for Computing Machinery.

Xinyu Wang, Jijun Chi, Zhenghan Tai, Tung Sum Thomas Kwok, Hailin He, Zhuhong Li, Yuchen Hua, Muzhi Li, Peng Lu, Suyucheng Wang, and 1 others. 2025b. Finsage: A multi-aspect rag system for financial filings question answering. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 6144–6152.

Yuxin Wang, Qingxuan Sun, and Sicheng He. 2023. M3e: Moka massive mixed embedding model.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. 2025. Know your limits: A survey of abstention in large language models. *Transactions of the Association for Computational Linguistics*, 13:529–556.

Frank Xing. 2025. Designing heterogeneous llm agents for financial sentiment analysis. *ACM Transactions on Management Information Systems*, 16(1):1–24.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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*.

Yilun Zhao, Hongjun Liu, Yitao Long, Rui Zhang, Chen Zhao, and Arman Cohan. 2024. Financemath: Knowledge-intensive math reasoning in finance domains. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12841–12858.

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. *arXiv preprint arXiv:2105.07624*.

A The Detail of FMKG

A.1 Graph Construction and Update

To ensure the coverage and accuracy of the Financial Metric Knowledge Graph (FMKG), we implement a multi-stage construction pipeline involving data acquisition, term disambiguation, and human-in-the-loop validation.

Metric Name: Cost of Sales

Aliases: *Cost of Goods Sold, Cost of Products Sold, COGS*

Definition:
Cost of sales refers to the production cost of goods sold or the service cost of services provided, as well as other business costs of sales, including main business costs and other business costs. It is a key item on the income statement used to measure the direct cost expenditures in a company's sales activities.

Formula & Code Implementation:
[Formula 1]

$$\text{Cost of Sales} = \text{Beginning Inventory} + \text{Purchases} - \text{Ending Inventory}$$

[Code 1]

```
def sales_cost_v1(beginning_inventory,
purchase, ending_inventory):
    return beginning_inventory + purchase -
ending_inventory
```

[Formula 2]

$$\text{Cost of Sales} = \text{Net Sales} \times (1 - \text{Gross Profit Margin})$$

[Code 2]

```
def sales_cost_v2(net_sales, gross_profit_
margin):
    return net_sales * (1 - gross_profit_
margin)
```

Figure 4: Example of Structured Metric Node: *Cost of Sales*.

.

Data Acquisition. We adopt a multi-source strategy to aggregate financial terminologies and calculation logic:

- **Function Library:** We incorporate core schemas from existing financial reasoning datasets *FinanceReasoning* (Tang et al., 2025) to establish a foundational taxonomy.
- **Encyclopedia Retrieval:** Following Zhao et al. (2024) and Tang et al. (2025), we crawl and parse entries from authoritative online encyclopedias (e.g., *Wikipedia*, *Investopedia*) to expand the set of terminologies, aliases, and definitions.
- **Unstructured Extraction:** To cover long-tail domain knowledge, we extract entities and calculation rules from unstructured documents, including financial textbooks and professional examination materials (e.g., *CFA/CPA prep books*).

Term Disambiguation. The raw collection contains significant redundancy and ambiguity (e.g., *Net Income* vs. *Net Profit*). To resolve this, we employ embedding-based semantic clustering to group synonymous terms. Subsequently, we leverage LLM-based semantic verification to strictly assess whether terms within each cluster referred to the same financial concept. Validated clusters are then merged into canonical nodes, where the LLM automatically aggregates their associated aliases, calculation formulas, and Python code snippets into a unified, standardized representation.

Hybrid Validation Pipeline. To guarantee the reliability of the graph, we employ a Human-LLM Collaborative verification process: First, we utilize GPT-5 to verify the extracted fields and autocomplete missing attributes, such as standardizing formula expressions and generating executable Python code for derived nodes. Domain experts review the clustered nodes and LLM-generated content to correct errors and resolve edge cases in formula dependencies.

Update Workflow. GBFR is designed to be computationally closed yet semantically extensible. Once a metric node and its formula are instantiated in FMKG, subsequent reasoning can be executed and verified through our four operators. To support this extensibility, FMKG is maintained as an evolving asset, and its update follows a workflow consistent with the construction process above. Specifically, when a new metric is introduced, we first align it to an existing canonical node if it is semantically equivalent; otherwise, we create a new node. We then induce candidate formulas and prerequisite dependencies from authoritative sources, including encyclopedic references, domain materials, and LLM-extracted content. Finally, we cross-validate the induced definitions, formulas, aliases, and code implementations across multiple sources, while uncertain cases are flagged for expert review before incorporation into the graph.

A.2 Rationale for the DAG Structure in FMKG

A key design choice of FMKG is to maintain a DAG structure. This is necessary because financial metrics are often linked by interdependent formulas: a derived metric may be computed from several variables, and the same formula may be algebraically rearranged to solve for one of those variables. For instance, $PV01 = Price \times Modified\ Duration \times c$ can also be rewritten as $Modified\ Duration = PV01/(Price \times c)$. If all such reversible relations were explicitly encoded as dependency edges, the graph would contain directed cycles. This would invalidate topological ordering and make recursive decomposition ill-defined. To avoid this issue, FMKG preserves a one-way dependency hierarchy from foun-

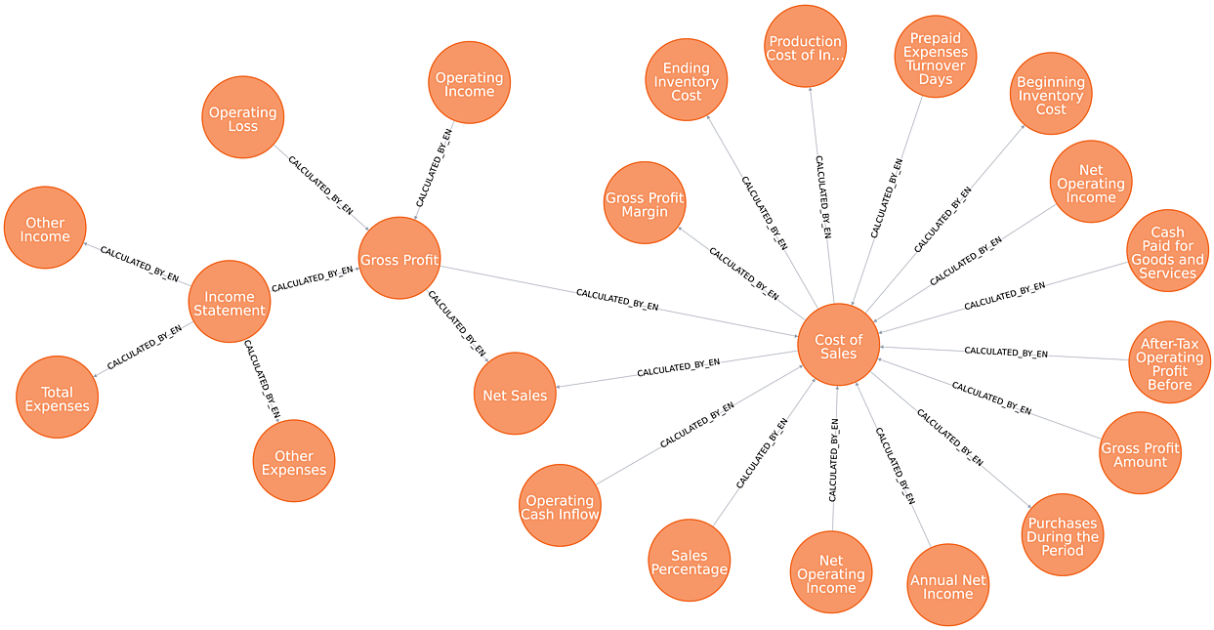

Figure 5: **A visualization of a subgraph extracted from the FMKG.** The graph illustrates the dense hierarchical dependencies among financial indicators, where nodes represent specific metrics (e.g., Cost of Sales, Gross Profit) and directed edges denote the explicit calculation logic ($v_{derived} \xrightarrow{\text{CALCULATED\_BY}} v_{derived}/v_{atomic}$).

dational variables to higher-order derived metrics. Under this design, only the canonical dependency direction is encoded in the graph (e.g., $PV01 \xrightarrow{\text{CALCULATED\_BY}} Modified\ Duration$ in FMKG), while the reverse use of formulas is handled at search and execution time rather than stored as an additional graph edge.

A.3 FMKG Statistics

To assess the quality of FMKG, we examine its scale, formula diversity, structural complexity, and expert-validated correctness. In terms of scale, FMKG contains 7,521 nodes, including 3,318 atomic metrics and 4,203 derived metrics, connected by 20,152 dependency edges. To characterize formula diversity, we measure the number of valid formulas associated with each derived metric. On average, each derived metric has 1.63 formulas, and 40.85% of derived metrics have multiple valid derivation formulas. In terms of structural complexity, FMKG has an average in-degree of 1.6, an average out-degree of 2.3, and a maximum derivation depth of 6, reflecting a complex multi-hop dependency structure. Finally, we conduct expert verification on 10% of the nodes. Two annotators independently assess node correctness, yielding a Krippendorff's α of 0.8231 and an acceptance rate of 90.1% without correction.

A.4 Instance of FMKG

The constructed FMKG forms a dense network of recursive calculation dependencies, structurally partitioned into *atomic nodes* (irreducible fundamental items) and *derived nodes* (complex metrics). Figure 4 illustrates the schema of a representative metric node along with its associated properties.

To present the complexity of financial reasoning, Figure 5 visualizes a sampled subgraph of the FMKG. The structure is characterized by high connectivity and recursive depth. Pivotal nodes, such as *Cost of Sales*, act as the nexus between foundational atomic nodes (e.g., *Beginning Inventory Cost*, *Purchases During the Period*) and high-level complex indicators (e.g., *Gross Profit*, *Net Operating Income*). This logical polymorphism empirically validates the necessity of our reasoning algorithm. Furthermore, the graph exhibits deep hierarchical dependencies, which confirms that linear PoT is often insufficient, necessitating the explicit structural guidance to accurately traverse these multi-hop calculation paths.

B The Generation Process of Counterfactual Sample

To rigorously benchmark the safe abstention capability, we constructs a high-quality negative dataset derived from existing financial QA benchmarks. Our construction pipeline consists of three main

stages: stratified dampling, LLM-based generation, and human verification.

B.1 Sampling and Generation Pipeline

To ensure the negative samples maintain the distribution characteristics of the original data while controlling the scale, we first perform random stratified sampling. Specifically, we sample 10% of the data instances from each of the four source datasets.

These sampled instances are then evenly divided into three groups to undergo distinct adversarial perturbations:

- **Temporal Shift**: Modifying time constraints to periods absent in the context.
- **Entity Mutation**: Swapping the target financial entity with a distractor.
- **Metric Substitution**: Replacing the requested financial term with a semantically related but unavailable metric.

We utilize LLM to automatically generate the perturbed questions based on specific instructions for each category, ensuring the linguistic fluency of the generated queries.

B.2 Human Verification

Since automated generation may occasionally produce logical loopholes (e.g., the answer remains inferable despite perturbation) or semantic errors, we introduce a human-in-the-loop verification process.

Annotators review the generated samples to ensure:

1. The perturbed question is indeed unanswerable based on the provided context.
2. The modification does not introduce grammatical errors that alter the original intent.
3. The logic of the question remains consistent with financial domain standards.

Samples that fail to meet these criteria are discarded to guarantee the reliability of the evaluation.

B.3 Dataset Statistics

Table 4 presents the statistics of the final dataset. The counterfactual samples account for approximately 10% of the original volume, providing a challenging subset for safety evaluation.

Table 4: Statistics of the datasets before and after counterfactual sample generation. The number of counterfactual samples is controlled at approximately 10% of the original size after sampling and human filtering.

Dataset	Original Samples	Counterfactual Samples	Total
CFBenchmark	140	11	151
FinanceReasoning	2,238	412	2,650
FinQA	1,002	97	1,099
TAT-QA	1,119	151	1,270

C Detailed mechanisms of reasoning types

Financial numerical reasoning involves multiple computation patterns, such as arithmetic composition, cross-entity or cross-time comparison, set-level aggregation, and temporal computation (Srivastava et al., 2024). Different reasoning patterns entail different evidence requirements, subsequent reasoning steps, and execution logic. Motivated by this observation, we divide financial queries into the five reasoning types listed in Table 5. For each type, we provide a symbolic semantics describing its reasoning process and final aggregation logic.

D Complexity and Empirical Cost of Algorithm

We analyze the efficiency of GBFR from both theoretical and empirical perspectives. The theoretical analysis is based on the FMKG structure under the standard work–span model, where *work* measures the total amount of computation and *span* captures the critical-path latency under ideal parallel execution. We then complement this analysis with quantitative measurements of path exploration and end-to-end token cost on the *FinanceReasoning* benchmark.

Work Complexity. Let H denote the maximum depth of the Financial Metric Knowledge Graph (FMKG), B the maximum number of submetrics per derivation, and K the number of candidate derivation formulas for a metric. In the absence of structural constraints, recursively enumerating all derivation paths yields an exponential worst-case complexity of $O((K \cdot B)^H)$. In practice, GBFR operates over an acyclic metric graph and restricts exploration to the query-relevant subgraph. Each derived metric is resolved at most once and evaluates up to K candidate formulas. Under this setting, the total work scales with the number of visited derived nodes V' and is bounded

Table 5: Five reasoning types of financial queries, with their symbolic semantics.

Reasoning Type	Reasoning Process	Final Aggregation Logic
Direct Retrieval	Retrieve a metric instance V that strictly satisfies constraints $\mathcal{C}$ directly from the context.	$Result = V$ (arithmetic prohibited)
Arithmetic Calculation	Extract explicit operands $\{V_1, V_2, \ldots\}$ defined by arithmetic relationships in the query, such as sum, difference, product, and division.	$Result = Op(V_1, V_2, \ldots)$, where $Op \in \{+, -, \times, \div\}$
Comparative Reasoning	Extract distinct operand values V_{target} and V_{base} via dimension shifting, such as time shift (t vs. $t-1$) or entity shift (e_1 vs. e_2).	$Result = \frac{V_{target} - V_{base}}{V_{base}}$ (ratio) or $V_{target} - V_{base}$ (range)
Statistical Aggregation	Perform batch retrieval of a value set $\mathcal{V}$ across a specified dimension, such as all quarters in a fiscal year.	$Result =$ Mean$(\mathcal{V})$, Sum$(\mathcal{V})$, or Count$(\mathcal{V})$
Temporal Reasoning	Define temporal boundaries T_{start} and T_{end} to capture fiscal scope, and extract duration only when explicitly requested.	$Result = \Delta(T_{end}, T_{start})$ (duration) or $V \times \frac{Scale}{\Delta T}$ (normalization)

by: $O\big(|V'| \cdot (1+K)\big)$, where the constant term accounts for evidence alignment and verification, and K corresponds to candidate formula evaluation.

Span Complexity. GBFR executes independent derivation paths in parallel and performs consensus-based verification. As a result, the overall latency is governed by the longest dependency chain. Along a critical path of depth H, each level invokes a constant number of atomic operators—Evidence Seeking, Metric Decomposition, Program Execution, and Verification. The span complexity is therefore: $O\Big(H \cdot (T_{\text{ES}} + T_{\text{MD}} + T_{\text{PE}} + T_{\text{Verify}})\Big)$.

This parallel structure enables GBFR to mitigate the latency growth typically induced by wide branching, making the framework scalable for deep, multi-hop financial reasoning.

Empirical Cost and Path Exploration. We further analyze the practical efficiency of GBFR on the *FinanceReasoning* benchmark. The average number of derivation paths explored per query is 1.53, and only 4.26% of queries require more than 5 paths, indicating that empirical search remains limited. This is due to early pruning by metric constraints, which terminates a path as soon as a required atomic metric cannot be grounded in the context. In terms of end-to-end cost, GBFR consumes 7,281 tokens per query on average, corresponding to $0.0669 per query under GPT-5 pricing. Compared with tool-augmented ReAct baselines, GBFR increases token usage by 78% but yields an absolute accuracy gain of 18%. In high-stakes financial settings, this additional cost may be an acceptable trade-off for improved accuracy.

E Experimental Setup Details

In this section, we provide concise descriptions of the four financial datasets used to evaluate the model's performance across text and table modalities.

E.1 Dataset Descriptions

FinQA (Chen et al., 2021)is a large-scale dataset expert-annotated from financial reports. It contains 8,281 question-answering pairs accompanied by explicit numerical reasoning processes, serving as a benchmark for evaluating complex financial calculations.

TAT-QA (Zhu et al., 2021) addresses discrete reasoning over hybrid financial data. It requires parsing tabular data and associated text simultaneously. The dataset includes diverse question types such as span extraction, arithmetic calculation, and aggregation, testing the model's ability to handle structural heterogeneity.

FinanceReasoning (Tang et al., 2025)comprises 2,238 questions designed to evaluate the numerical reasoning of Large Reasoning Models. Addressing limitations in complexity and accuracy found in prior benchmarks, each sample features a hybrid context and a Python-formatted solution, providing a reliable standard for assessing complex financial calculation tasks.

CFBenchmark (Lei et al., 2023) is a comprehensive Chinese financial evaluation suite designed to assess Large Language Models across practical applications. It covers dimensions including financial natural language processing, numerical calculation, analysis, and compliance.

E.2 Data Selection and Preprocessing

To construct a rigorous benchmark for numerical reasoning, we curate the experimental data through a specific selection and filtering process.

First, regarding data sources, we utilize the standard `test.json` split from FinQA and the `tatqa_dataset_test_gold.json` file from TAT-QA. For FinanceReasoning, the entire dataset is employed. In the case of CFBenchmark, we specifically select the subset related to financial numerical calculations.

Second, to focus exclusively on numerical reasoning capabilities, we exclude samples that require non-numerical answers (e.g., questions where the ground truth is a text span or a string). As a result, only samples involving numerical values and calculation steps are retained for the final evaluation set.

E.3 Baseline Implementation Details

To rigorously evaluate the contribution of our framework, we compare it against the following state-of-the-art baselines. Regarding the embedding models, we employ *text-embedding-3-large* (OpenAI, 2024) for the FinanceReasoning, FinQA, and TAT-QA datasets. Conversely, for the CFBenchmark dataset, we adopt the *M3E-large* (Wang et al., 2023) model to better handle Chinese contexts.

Program-of-Thought (PoT): We prompt the LLM to generate Python programs to solve the query. This baseline tests the model's intrinsic ability to perform reasoning and calculation.

Knowledge Augmentation Strategies. Following Tang et al. (2025), we implement three variants of RAG:

Standard RAG: We utilize the input question to retrieve the description of Top-3 financial metrics based on cosine similarity. This serves as the lower-bound baseline for semantic retrieval.

LLM as Retrieval Judge: Addressing the low precision of raw retrieval, we adopt a re-ranking strategy. We first retrieve a larger candidate pool (Top-30) and then employ the LLM as a semantic discriminator to select the Top-3 most relevant metrics and their knowledge. (Guan et al., 2024a)

LLM-Instructed Knowledge Retrieval: To mitigate the misalignment between raw queries and hybrid contexts, we implement a query rewriting strategy. A LLM summarizes the semantic information from the context to generate de-noised search queries, thereby enhancing retrieval accuracy. (Verma et al., 2025; Li et al., 2025)

F Prompt

The prompts in GBFR are following:

Question Analysis Prompt

```
You are a "Question Analyst" in a financial reasoning assistant.
Your job is to read a user question together with its provided context (text/tables),and turn
    the question into a clean structured plan for later computation.
You must:
- Use BOTH the question and the context to decide what needs to be computed.
- Extract only the required operand metrics as subgoals. Do not compute the final answer.
- Fill constraints using evidence in the context whenever possible (e.g., company name, dates,
    currency symbol, scale like "million", unit like "percentage").
- If something is truly missing from both question and context, use "unknown" or null (do not
    guess).
Output MUST be a single valid JSON object with exactly these keys:
{
  "reasoning_type": ...,
  "operator": ...,
  "subgoals": [...],
  "precision": ...
}
No markdown, no extra keys, no explanations.
=== Reasoning Type (choose ONE) ===
1) DirectRetrieval
   - The answer is a single value that can be directly retrieved from context, no arithmetic.
2) ArithmeticCalculation
   - Basic arithmetic such as sum / difference / product / division.
3) ComparativeReasoning
   - Comparison logic across entities or periods such as ratio / change ratio / range / compare.
4) StatisticalAggregation
   - Aggregation over a set such as average / counting (including weighted average when implied)
     .
5) TemporalReasoning
   - Time-span reasoning or temporal normalization (e.g., annualized return over a period).
   - For TemporalReasoning, represent time using operands temporal_scope (start/end), rather
     than adding a separate "duration" subgoal unless the question explicitly asks for duration
     itself.
=== Operator (choose ONE from this set) ===
[sum, difference, product, division, ratio, change ratio, range, compare, average, time,
    counting]
=== Subgoals (very important) ===
- subgoals ONLY include the numerical metrics/values that need to be retrieved (or later
    computed by other operators).
- Do NOT add helper items that are not required operands (e.g., do not add "duration" unless the
     question asks for duration).
- Each subgoal must be (metric, constraints).
- constraints MUST include:
  - temporal_scope: year/quarter/date-range if any
  - entity_scope: company/group/subsidiary if any
  - meta: {currency, scale, unit} (use context to fill; otherwise unknown)
=== Precision ===
- If the question specifies formatting, output it in "precision" (e.g., "nearest integer", "
    percentage to two decimal places").
- If not specified, use "unknown".
=== Few-shot Examples ===
{example}

Query: {query}
Context:
{context}

Return JSON only.
```

Evidence Seeking Prompt

```
You are an "Evidence Finder" for financial/numerical questions.
You will receive a context (text/tables) and a list of variables to resolve. For each variable,
    you must find the best evidence span in the given context and extract the numeric value.
=== CRITICAL RULE: EXTRACTION ONLY, NO CALCULATION ===
You are ONLY allowed to EXTRACT values that are EXPLICITLY WRITTEN in the context.
You are ABSOLUTELY FORBIDDEN from:
- Calculating or computing any value (e.g., subtracting, adding, multiplying numbers)
- Deriving a value from other numbers in the context
- Inferring a value that is not directly stated
If a metric is asked but its value is NOT directly written in the context, you MUST put it in
    missing_vars, even if you could theoretically calculate it from other available data.
=== What counts as "directly written"? ===
VALID (can extract):
- "Gross Profit: $350,000": Gross Profit is directly written
- "Net Sales ... $900,000": Net Sales is directly written
- A table cell with header "Revenue" and value "1,000,000": Revenue is directly written
INVALID (must mark as missing):
- Context has "Revenue: $1,000,000" and "Cost: $600,000", but asks for "Gross Profit"
  - Gross Profit is NOT directly written, even though you could calculate 1,000,000 - 600,000
  - Put "Gross Profit" in missing_vars
- Context has component values but not the aggregate
  - The aggregate must go in missing_vars
=== Strict Evidence-Bounded Rules ===
- Use the context as the only source of truth.
- If the context is in Chinese, you need to translate the terms into Chinese before searching.
- The extracted value MUST appear with a label/header that matches (or is a clear alias of) the
    requested metric.
- You may use aliases/synonyms to search, but the extracted value MUST satisfy the given
    constraints: temporal_scope, entity_scope, and meta attributes (currency/scale/unit) when
    specified.
- If you cannot find a value that matches the constraints, or the context is ambiguous/
    conflicting, you must abstain: do NOT guess; put the variable name into missing_vars, and
    do not include it in var_bindings.
=== Confidence Definition ===
- confidence is your subjective probability (0.0~1.0) that the extracted value is the correct
    grounding for this variable under the constraints.
=== Value Rule ===
- Return the numeric value as written (do NOT apply scaling conversions like multiplying "
    million").
- Copy an evidence snippet from the context that contains the number and the minimal surrounding
     label (include table row/column headers if needed).
- The evidence snippet MUST show the metric name/label alongside the number.
=== Notes Rule ===
- If a metric is found using an alias (i.e., the context uses a different name than the metric
    label), you MUST explain the alias mapping in notes.
  Example: "Resolved Revenue via alias Net sales".
- If any metric cannot be found or must be abstained, you MUST explain why in notes (e.g., "
    Gross Profit not directly stated in context, only component values available").
- If ALL extracted values are directly and explicitly matched (no alias needed) AND no variable
    is missing, then notes MUST be null.
=== Output Format ===
Output MUST be valid JSON only, matching:
{format}

Before outputting, ask yourself:
1. Does the metric have its OWN LABEL in the context?
2. Is my "context" field showing a single "[Label]: [Value]" pair?
3. Does the numeric value actually appear in the context?

Context:

{context}
Variables to resolve:
{var_list}
Return JSON only. No markdown. No extra keys. ABSOLUTELY NO CALCULATIONS.
```

Program Generation Prompt for indicator calculation

```
You are a financial calculation code generation expert. Generate correct Python calculation code
     based on the given financial metric formula.

Code Requirements:
1. Must define a `compute(vars)` function as the entry point
2. `vars` is a dictionary containing all required variable values, with metric names as keys
3. The function must return a dictionary containing "value" (calculation result) and "trace" (
    intermediate variables)
4. Do not use import statements
5. Return None when handling division by zero errors

Example:
Input formula: Working Capital = Current Assets - Current Liabilities

Output code:
```python
def compute(vars):
    current_assets = vars["Current Assets"]
    current_liabilities = vars["Current Liabilities"]
    working_capital = current_assets - current_liabilities
    return {
        "value": working_capital,
        "trace": {"Current Assets": current_assets, "Current Liabilities": current_liabilities}
    }
```

Please generate calculation code for the following financial indicator:

Indicator name: {indicator_name}
Calculation formula: {formula_expr}
Required variables: {required_vars}

Please generate the complete compute(vars) function:
```

Program Generation Prompt for solve problem

```
You are a financial calculation code generation expert. Generate correct Python calculation code
     for solving the given query and context based on the given information.

You will receive the following information:
- the query
- the context (text/tables)
- the reasoning type of query
- evidences (each evidence includes metric name, constraints, value, and context snippet)
- an optional concept explanation (what the metric means / how it is described in documents)
- an optional metric formula
- an optional output precision requirement

Code Requirements:
1. Must define a `compute(vars)` function as the entry point
2. `vars` is a dictionary containing all required variable values, with metric names as keys
3. The function must return a dictionary containing "value" (calculation result) and "trace" (
    intermediate variables)
4. No `import` statements. No file/network access, no eval/exec, no classes, no external
    libraries.
5. Return None when handling division by zero errors

=== Reasoning Type Description ===
1) DirectRetrieval
   - The answer is a single value that can be directly retrieved from context, no arithmetic.
2) ArithmeticCalculation
   - Basic arithmetic such as sum / difference / product / division.
3) ComparativeReasoning
   - Comparison logic across entities or periods such as ratio / change ratio / range / compare.
4) StatisticalAggregation
   - Aggregation over a set such as average / counting (including weighted average when implied)
    .
5) TemporalReasoning
   - Time-span reasoning or temporal normalization (e.g., annualized return over a period).
   - For TemporalReasoning, represent time using operands temporal_scope (start/end), rather
    than adding a separate "duration" subgoal unless the question explicitly asks for duration
    itself.

Please generate calculation code according to following information:
Question: {query}
Context: {context}
Reasoning type: {reasoning_type}
Evidences (IMPORTANT - use these exact variable names as keys in vars dict):
{evidences}

CRITICAL: The vars dict will contain keys exactly as shown in the evidences above (e.g., "v1", "
    v2", etc.).
You MUST use these exact keys to access values. For example: vars["v1"], vars["v2"], etc.
DO NOT use metric names as keys - use the var_name (v1, v2, ...) as keys.

Please generate the complete compute(vars) function:
```

Path Validation Instruction

```
You are a financial reasoning assistant.
Your job is to verify multiple candidate reasoning paths produced by upstream operators. Each
    path contains: (i) subgoals with constraints, (ii) evidence bindings/snippets, (iii)
    computation trace, and (iv) a proposed final numeric result with meta (currency/scale/unit).


GOAL
- Decide whether the system can output a single final answer or must safely abstain.
- This decision MUST be evidence-bounded: only accept an answer if it is verifiably supported by
     the given evidence and consistent across valid paths.

HARD RULES (do not violate)
1) Do NOT introduce new evidence, new numbers, new assumptions, or new computation paths.
2) Do NOT "fix" a path by guessing missing operands, units, time ranges, entities, aliases, or
    scales.
3) Do NOT average conflicting results. Do NOT pick a result simply because it "looks reasonable
    ".
4) If no single answer can be verified after exhausting the provided paths, output None.
5) Treat "unknown" / null in constraints or meta as missing information, not something to guess.

INPUT
- question: the user question
- context: the provided text/tables (may be partial)
- candidate_paths: a list of objects, each containing:
  - subgoals: list of (metric, constraints)
  - evidence: variable bindings -> evidence snippets/spans
  - computation: formulas/program + intermediate values

VERIFICATION CHECKLIST (per path)
A) Evidence grounding
- Every operand used in computation must be traceable to an evidence snippet in this path.
- Evidence must match constraints: temporal_scope, entity_scope, and meta (currency/scale/unit)
    whenever specified.
- If the path uses an alias mapping, it must be explicitly supported by evidence/metadata in the
     path (not assumed).
B) Constraint consistency
- Reject the path if it violates temporal_scope (wrong year/quarter/date-range), entity_scope (
    wrong company/group/subsidiary), or incompatible meta (currency/scale/unit).
- Reject the path if it silently mixes scales (e.g., millions vs. thousands) or currencies
    without an explicit conversion supported by evidence.
C) Computation correctness
- Check that intermediate steps follow the stated formulas/program.
- If intermediate values do not follow from prior values, mark ARITHMETIC_ERROR.
- If the computation requires a missing operand, mark MISSING_OPERAND (do not fill it).

PATH VALIDITY
- A path is VALID only if it passes A+B+C with no critical errors.
- A path is INVALID if it has any of: MISSING_OPERAND, EVIDENCE_NOT_FOUND, CONSTRAINT_MISMATCH,
    UNIT_MISMATCH, CURRENCY_MISMATCH, SCALE_MISMATCH, ARITHMETIC_ERROR, UNSUPPORTED_ALIAS.

OUTPUT FORMAT
{format}

Question: {query}
Context:
{context}
Candidate paths:
{candidate_paths}
```