SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Beyond Intent Detection: Multi-Intent Reasoning And Execution In Agentic Observability Systems

Akila Balasubramanian

Independent Researcher, USA.

Abstract—Multi-intent understanding is a fundamental requirement for AI assistants operating in complex, real-world domains such as observability and incident investigation, where user queries routinely encode multiple interdependent tasks within a single utterance. Existing approaches treat multi-intent detection as multi-label classification—identifying intent labels independently without modelling the dependencies, execution ordering constraints, or shared context that govern how compound queries must be resolved. This framing is insufficient for agentic systems that must translate user intent into coordinated tool-driven workflows. This paper introduces an execution-aware multi-intent framework that reframes multi-intent understanding as a planning and orchestration problem. User queries are parsed into structured intent sets with shared entity context, transformed into an intent graph that encodes dependency and ordering constraints as typed directed edges, compiled into an execution plan through topological analysis, and executed through a deterministic orchestration layer that dispatches tool calls, routes results across intents, and recovers from partial failures. Evaluation on a benchmark of 1,240 hybrid synthetic and realistic multi-intent observability workflows demonstrates significant improvements over single-intent, independent multi-intent, and LLM-only implicit orchestration baselines across task completion rate, dependency handling accuracy, and robustness under partial failures. The results establish that modelling execution semantics—not merely intent labels—is the critical architectural requirement for production-grade multi-intent agentic AI systems. The framework and its implications extend beyond observability to any domain where natural language user queries must drive coordinated multi-step execution.

Keywords: Multi-Intent Understanding, Agentic AI, Intent Graph, Execution Planning, Observability, Task Orchestration, Natural Language Understanding.

I. INTRODUCTION

AI assistants capable of understanding and acting on natural language have become a central focus of artificial intelligence research, with applications spanning dialogue systems, code generation, data analysis, and operational automation [1], [2]. In practice, the interactions through which users direct these systems are rarely simple or single-threaded. Real-world queries in complex operational domains are compound: a single utterance from an engineer investigating a production incident may encode multiple tasks—anomaly detection, dependency correlation, log retrieval—that must be executed in a specific order, share intermediate results, and produce a coherent integrated response. The ability to handle such compound queries determines whether an AI assistant can function as a genuine operational partner or is limited to handling artificially simplified single-intent interactions [3].

Existing approaches to multi-intent understanding in NLP and dialogue systems treat the problem as multi-label classification: a model is trained to predict a set of intent labels for a given utterance, treating each label as independent [4], [5]. This framing captures the surface-level identification of multiple intents but fundamentally fails to represent the properties that matter most for execution: the dependency relationships between intents (some intents can only execute after others produce results), the execution ordering constraints (parallel versus sequential execution), and the shared entity context (the same service name or time window is referenced across multiple intents in a compound query). Without these representations, a multi-label classifier produces a flat list of intents that cannot directly drive coordinated system execution—the gap between intent prediction and intent execution is not bridged by label prediction alone [6].

Recent advances in LLM-based agentic systems have demonstrated strong capabilities in multi-step reasoning and tool use, enabling systems to decompose problems, call external APIs, and synthesise results

[7], [8]. However, agentic systems that rely on implicit, prompt-driven control flow for multi-intent handling—allowing the LLM to implicitly decide execution order and manage inter-intent context—exhibit non-deterministic behaviour, fail to enforce necessary ordering constraints, and cannot provide the execution guarantees required in production operational settings. When a compound query requires that anomaly detection precede root cause correlation, and that both precede log retrieval scoped to the identified anomaly window, an implicitly orchestrated system may execute these intents in any order or omit inter-intent context propagation, producing an incoherent or incorrect response [9], [30].

This paper addresses the gap between intent identification and intent execution by proposing a framework in which multi-intent understanding is treated as a planning and orchestration problem rather than a classification problem. The core contribution is an architectural pipeline that transforms a multi-intent user query into a deterministic execution plan: parsing produces a structured intent set with shared entity context, intent graph construction encodes dependencies and ordering constraints as typed directed edges, topological analysis compiles an execution plan, and a deterministic orchestration layer executes the plan with structured tool dispatch and inter-intent result routing. This framework is evaluated in the domain of observability—a high-complexity real-world setting where multi-intent queries are operationally common and where the consequences of incorrect execution ordering are directly measurable—but its architectural principles are general. The paper is organised as follows: Section II formalises the problem; Section III reviews related work; Section IV describes the framework; Section V presents the evaluation; Section VI provides analysis; Sections VII and VIII discuss implications and conclude.

II. PROBLEM FORMULATION

A multi-intent query Q is a natural language utterance encoding $k \geq 2$ intents $I = \{i_1, i_2, \ldots, i_k\}$, each characterised by an intent type (e.g., anomaly_detection, log_retrieval, dependency_correlation), a set of arguments extracted from the query (e.g., service name, time window, metric name), and a set of context slots that may be filled by outputs from other intents in the same compound query. The shared context C captures entities referenced across multiple intents—a time window identified in one intent may scope the log retrieval in another, and a service identified by anomaly detection may direct subsequent dependency correlation. A multi-intent query differs from a sequence of independent single-intent queries precisely because the intents share context, are ordered by dependencies, and must be resolved as a coordinated unit rather than as isolated requests [3], [6].

Three structural properties of multi-intent queries cannot be captured by multi-label classification. First, intent dependency: the output of one intent is required as an input argument to another (e.g., the anomalous time window identified by intent i_1 is required as the time scope for intent i_2). Second, execution ordering constraints: some pairs of intents may execute in parallel (no dependency between them), while others must execute sequentially in a specified order. Third, conditional dependency: some intents execute only if a prior intent produces a result satisfying a condition (e.g., intent i_3 executes only if intent i_1 identifies an anomaly with confidence above a threshold). A complete multi-intent representation must encode all three dependency types to enable correct execution planning. The intent graph $G = (I, E)$ provides this representation: nodes are intents, and typed directed edges E encode sequential, parallel, and conditional dependencies between pairs of intents.

The execution problem is then: given intent graph G, produce an execution plan P = (stages, routing) such that all sequential ordering constraints are satisfied, all parallel-executable intent sets are identified for concurrent dispatch, all conditional dependencies are evaluated at runtime, and all inter-intent context propagation is explicitly specified. This problem is equivalent to scheduling a directed acyclic graph (DAG) with typed edge semantics and runtime-evaluated conditional branches—a well-defined problem in workflow planning that goes substantially beyond multi-label classification [10], [11]. The observability domain provides an ideal evaluation setting because multi-intent queries are operationally common (e.g., "check latency for service A, correlate with recent deployments, and show related error logs"), the correct dependency ordering is unambiguous, and execution errors produce directly measurable outcomes [28].

III. BACKGROUND AND RELATED WORK

A. Intent Detection and Multi-Label Classification

Intent detection in task-oriented dialogue has been studied extensively, with the field progressing from single-intent classification to multi-label formulations that allow a single utterance to be assigned multiple intent labels [4], [5]. The Multi3NLU++ dataset extended multi-intent evaluation to multilingual, multi-domain settings, establishing that multi-intent understanding is a widespread and challenging problem across operational contexts [5]. However, the multi-label formulation treats intents as a set of independent

labels with no structural relationships. Research on complex task understanding, including TaskLAMA, demonstrates that while LLMs can decompose complex tasks into steps, predicting the temporal dependencies between steps remains a significant challenge even for state-of-the-art models, with dependency prediction accuracy substantially below step identification accuracy [6]. This gap motivates the explicit dependency modelling approach proposed in this paper rather than relying on implicit LLM dependency handling.

B. LLM-Based Agentic Systems

LLM-based agentic systems have demonstrated strong capabilities in tool-augmented multi-step reasoning. ReAct introduced the interleaving of reasoning traces and tool actions, enabling agents to ground their reasoning in external evidence [7]. AgentBench established a multi-dimensional benchmark for evaluating LLMs as agents across diverse task environments, revealing substantial performance variation across task types and demonstrating that reasoning capability alone is insufficient for reliable task completion [1]. Recent work on agent planning with world knowledge models at NeurIPS 2024 showed that equipping agents with structured world models improves planning reliability on long-horizon tasks [8]. However, these systems rely on implicit LLM-driven control flow for ordering and coordinating actions, which produces non-deterministic execution sequences and fails to enforce the structured dependency constraints that are necessary for correct multi-intent compound query resolution [9]. The framework proposed in this paper replaces implicit control flow with an explicit intent graph and deterministic orchestration, preserving the reasoning flexibility of LLM-based systems while providing execution determinism.

C. Workflow Planning and Task Dependency Modelling

Dependency-aware task planning using DAG representations has been explored in robotics and multi-robot systems. DAG-Plan demonstrated that LLM-generated DAG-structured task representations enable parallel and adaptive execution of complex robotic tasks, achieving 52.8% higher efficiency compared to sequential single-agent approaches [10]. DART-LLM extended this to multi-robot settings, using DAG-based dependency modelling to coordinate decomposed subtasks across robotic agents with explicit parallel execution support [11]. These systems demonstrate the effectiveness of explicit dependency representation for execution planning, but they are designed for physical robot action sequences rather than natural language intent resolution in AI dialogue systems. The intent graph framework proposed in this paper adapts the DAG-based dependency modelling paradigm to the multi-intent NLU problem, introducing typed edge semantics for sequential, parallel, and conditional dependencies, shared entity context propagation, and integration with LLM-based intent parsing.

IV. MULTI-INTENT FRAMEWORK

The framework comprises four sequential components that transform a natural language multi-intent query into a deterministically executed set of tool-driven workflow results: intent set representation, intent graph construction, execution plan compilation, and deterministic orchestration.

A. Intent Set Representation

The first component parses the input query Q into a structured intent set $I = \{i_1, \ldots, i_k\}$ with shared context C. Each intent i_j is represented as a structured object with three fields: type (the intent category, drawn from a domain-defined ontology of observable action types such as anomaly_detection, metric_query, log_retrieval, dependency_correlation, and deployment_lookup), arguments (the set of extracted entity values directly specified in the query for this intent, such as service="checkout", metric="p99_latency"), and context_slots (the set of argument fields that will be filled by outputs from other intents at runtime, such as time_window=RESULT(i_1).anomaly_window). The shared context C is extracted as a set of entity-value pairs that apply globally across all intents in the compound query, such as the environment identifier or the investigation scope. Intent parsing is performed by an LLM prompted with the domain ontology, with a structured output schema enforcing the typed representation. This representation makes inter-intent dependencies explicit at the data level, enabling the graph construction component to identify dependency edges from context slot references without requiring further LLM inference [3].

B. Intent Graph Construction

The intent graph G = (I, E) is constructed by analysing the context slot references across the parsed intent set. For each context slot in intent i_j that references RESULT(i_n), a directed dependency edge ($i_n \rightarrow i_j$) of type sequential is added to E. For pairs of intents with no context slot dependency between them, a parallel edge is added, indicating they may execute concurrently. Conditional edges are added when the query contains explicit conditionality—"if anomalies are found, then correlate with deployments"—producing an edge ($i_1 \rightarrow_t^{ond} i_2$) that is only activated at runtime if i_1's result satisfies the specified condition. The resulting graph G is validated as a DAG: cycles would indicate circular dependencies that cannot be resolved through sequential execution, and cycle detection is run as a pre-execution consistency check. In practice, well-

formed multi-intent observability queries do not produce cycles because they encode investigation workflows with natural temporal ordering from detection to diagnosis to evidence retrieval [10].

C. Execution Plan Compilation

The execution plan P is compiled from the intent graph G through topological analysis. A topological sort of G partitions the intent set into execution stages: intents with no incoming edges form Stage 1 and may execute in parallel; intents whose all predecessors are in Stage 1 form Stage 2; and so on. Parallel intents within a stage are dispatched concurrently, with the orchestration layer waiting for all intents in a stage to complete before advancing to the next stage. Conditional edges introduce runtime branching: conditional intent i_j is added to the execution plan as a conditional node that is activated only if the result of its predecessor satisfies the specified condition, evaluated by the orchestration layer at the stage transition. The execution plan also specifies the context routing map: for each context slot in each intent, the routing map specifies which prior intent's result field provides the value for that slot. This explicit routing eliminates the need for the LLM to infer context propagation during execution, making the information flow deterministic and auditable [9], [11].

D. Deterministic Orchestration Layer

The orchestration layer executes the compiled plan by dispatching tool calls, routing results, and managing execution state. At each stage, parallel intents are dispatched concurrently as typed tool invocations with arguments populated from their argument fields and context slots resolved from the routing map. Each tool invocation returns a structured result object that is stored in the execution state. At stage transitions, the orchestration layer evaluates conditional edge predicates against the relevant result objects, activates or deactivates conditional intents accordingly, and advances the stage pointer. If a tool invocation fails—due to timeout, API error, or empty result—the orchestration layer applies intent-level failure handling: retry with backoff, fallback to an alternative tool with equivalent capability if available, or graceful degradation that marks the intent as unresolved and propagates a nil result to dependent intents rather than halting execution. This failure-tolerant design ensures that partial failures in compound queries produce the best available partial response rather than complete execution failure [2], [8], [23].

V. EVALUATION

The framework is evaluated on a benchmark of 1,240 multi-intent observability workflows. The benchmark comprises 620 synthetic workflows constructed by programmatically composing intent sequences with known dependency structures and ground-truth execution orders, and 620 realistic workflows derived from real-world incident investigation patterns with expert-annotated intent structures and expected outputs. Workflows are stratified by structural complexity: 310 two-intent workflows with one sequential dependency, 520 three-intent workflows with mixed sequential and parallel dependencies, and 410 four-or-more-intent compound workflows with conditional dependencies. Four systems are compared: a single-intent baseline that processes one intent at a time without multi-intent awareness, an independent multi-intent baseline that identifies all intents through multi-label classification and executes them independently without dependency modelling, an LLM-only implicit orchestration baseline that uses chain-of-thought prompting to guide multi-intent execution without an explicit intent graph, and the proposed execution-aware multi-intent framework.

Table I presents the primary evaluation results. The proposed framework achieves 84.3% overall task completion rate, compared to 47.6% for the single-intent baseline, 63.2% for the independent multi-intent baseline, and 71.8% for the LLM-only implicit orchestration baseline. The performance gap is largest on four-or-more-intent compound workflows with conditional dependencies: 76.4% (proposed) versus 28.3% (single-intent), 41.7% (independent multi-intent), and 58.9% (LLM-only)—confirming that the architectural advantages of explicit dependency modelling and deterministic orchestration are most consequential precisely in the most complex compound queries. Dependency handling accuracy—the proportion of executions in which inter-intent context propagation and ordering constraints are correctly satisfied—is 91.2% for the proposed framework versus 0% for the two non-dependency-aware baselines and 64.7% for LLM-only implicit orchestration. Under a 20% simulated tool failure rate, the proposed framework maintains 73.6% task completion, compared to 31.4% for LLM-only implicit orchestration, confirming the practical importance of intent-level failure recovery.

TABLE I Comparative Evaluation Results Across All Systems and Benchmark Strata.

Metric	Single-Intent	Indep. Multi-Intent	LLM-Only Implicit	Proposed Framework
Overall Task Completion	47.6%	63.2%	71.8%	84.3%
2-Intent Sequential Completion	71.3%	78.4%	82.1%	91.7%
3-Intent Mixed Completion	45.2%	61.8%	70.3%	83.6%
4+ Intent Conditional Completion	28.3%	41.7%	58.9%	76.4%
Dependency Handling Accuracy	N/A	0%	64.7%	91.2%
Task Completion (20% tool failure)	29.1%	38.4%	31.4%	73.6%

Sources: References [1], [7], [9]

VI. ANALYSIS

The evaluation results expose distinct failure modes for each baseline that explain the performance gaps. The single-intent baseline fails on compound queries because it processes one intent at a time without awareness of inter-intent context, requiring the user to manually decompose compound queries and issue multiple requests—a task allocation that eliminates the primary value of natural language interaction. The independent multi-intent baseline identifies all intents but executes them without dependencies, producing results that are individually correct but contextually incoherent: log retrieval is scoped to the full time range rather than the anomaly window identified by the anomaly detection intent, and deployment correlation retrieves all recent deployments rather than those affecting the services flagged by metric analysis. This explains the 0% dependency handling accuracy: when dependencies are not modelled, they are never satisfied by definition.

The LLM-only implicit orchestration baseline achieves 64.7% dependency handling accuracy despite having no explicit dependency model, because the LLM's reasoning capability enables it to infer correct ordering in a majority of cases from the query semantics alone. However, the 35.3% of cases where it fails are concentrated in compound workflows with non-obvious conditional dependencies and in cases where the query phrasing does not explicitly signal the dependency relationship—precisely the cases that are most common in real operational settings where users phrase requests concisely rather than explicitly. The 31.4% task completion under tool failures for the LLM-only baseline, compared to 73.6% for the proposed framework, reveals a critical practical difference: when a tool invocation fails, implicit orchestration has no recovery mechanism and either stalls or proceeds with missing context, while the proposed framework's intent-level failure handling produces a partial but coherent response.

The comparison with prior work on dependency-aware planning is instructive. DAG-Plan achieved 52.8% efficiency improvement through DAG-based task planning in robotics, and TaskLAMA showed that LLMs struggle specifically with dependency prediction despite strong step identification performance [6], [10]. The proposed framework resolves the dependency prediction challenge identified by TaskLAMA by deriving dependencies structurally from context slot references rather than asking the LLM to predict them, and extends the DAG execution paradigm established by DAG-Plan to the multi-intent NLU setting with typed edge semantics and LLM-parsed intent representations. This combination of LLM-strength parsing with structured deterministic execution planning is the architectural synthesis that produces the observed performance improvements [26].

VII. DISCUSSION

The central contribution of this work to the AI research community is the reframing of multi-intent understanding as an execution problem with architectural implications that extend well beyond the observability domain [23], [24]. The framing demonstrates that multi-label classification, while necessary, is architecturally insufficient for production-grade multi-intent agentic systems: intent label prediction must be accompanied by dependency graph construction, execution plan compilation, and deterministic orchestration to produce reliable compound query resolution [25]. This architectural requirement is not specific to observability—it applies to any domain where natural language user queries encode multiple interdependent tasks that must drive coordinated tool-driven execution. Customer support automation (diagnose issue, check account status, initiate resolution), code generation assistance (understand requirement, search codebase, generate and validate code), and multi-step data analysis pipelines (retrieve, transform, analyse, visualise) all exhibit the same structural requirements [3], [8], [27].

The relationship to prior work on planning and agent systems is direct. The planning-as-search paradigm in classical AI has long recognised that action dependency and ordering constraints are fundamental to plan correctness—the contribution here is adapting this insight to the natural language multi-intent setting by introducing an intent graph formalism that bridges the gap between linguistic intent representation and executable workflow planning [28]. The results also have implications for the evaluation of multi-intent systems: existing benchmarks based on intent label accuracy are insufficient for evaluating production-grade systems, and dependency handling accuracy and compound task completion under partial failures should be included as primary evaluation metrics. Several limitations define the scope of the current contribution: intent graph construction quality depends on the LLM's ability to correctly identify context slot references during parsing; highly ambiguous queries with unclear intent boundaries may produce incorrect graph topologies; and the framework assumes a predefined domain ontology of intent types, which requires domain-specific engineering for new application areas [29], [30].

VIII. CONCLUSION

This paper introduced an execution-aware multi-intent framework that addresses the fundamental gap between intent label prediction and intent execution in agentic AI systems. By representing multi-intent queries as structured intent sets with shared context, constructing intent graphs that encode dependency and ordering constraints, compiling execution plans through topological analysis, and executing them through a deterministic orchestration layer, the framework achieves substantial improvements over multi-label classification and LLM-only implicit orchestration baselines across task completion, dependency handling accuracy, and fault tolerance. The evaluation on 1,240 multi-intent observability workflows demonstrates that explicit dependency modelling is the critical architectural differentiator, with the performance advantage largest on complex conditional compound queries that are most representative of real operational interactions.

The broader contribution to the AI research community is the establishment of execution semantics—not merely intent labels—as the central design requirement for multi-intent agentic AI. The intent graph formalism introduced here provides a principled bridge between natural language understanding and structured execution planning, applicable across any domain where users direct AI agents through compound natural language queries. Future work will address intent graph construction from ambiguous or underspecified queries, adaptive ontology learning for new application domains, and integration with multi-modal intent signals that combine natural language with contextual system state.

REFERENCES

[1] X. Liu, H. Yu, H. Zhang et al., "AgentBench: evaluating LLMs as agents," in Proc. Int. Conf. Learning Representations (ICLR 2024), 2024. [Online]. Available: https://arxiv.org/abs/2308.03688

[2] T. Masterman, S. Besen, M. Sawtell, and A. Chao, "The landscape of emerging AI agent architectures for reasoning, planning, and tool calling: a survey," arXiv preprint arXiv:2404.11584, 2024. [Online]. Available: https://arxiv.org/abs/2404.11584
[3] N. Moghe, E. Razumovskaia, L. Guillou, I. Vulić, A. Korhonen, and A. Birch, "Multi3NLU++: a multilingual, multi-intent, multi-domain dataset for natural language understanding in task-oriented dialogue," in Findings of the Association for Computational Linguistics: ACL 2023, 2023. [Online]. Available: https://aclanthology.org/2023.findings-acl.230
[4] J. Liu, Z. Wu, J. Wang, J. Su, T. Che, and Q. Fang, "A segment augmentation and prediction consistency framework for multi-label unknown intent detection," ACM Transactions on Knowledge Discovery from Data, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3680286
[5] Y. Cheng et al., "Intent-aware dialogue generation and multi-task contrastive learning for multi-turn intent classification," arXiv preprint arXiv:2411.14252, 2024. [Online]. Available: https://arxiv.org/abs/2411.14252
[6] Q. Yuan, M. Kazemi, X. Xu, I. Noble, V. Imbrasaite, and D. Ramachandran, "TaskLAMA: probing the complex task understanding of language models," in Proc. AAAI Conf. Artificial Intelligence, vol. 38, 2024. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29918
[7] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: synergizing reasoning and acting in language models," arXiv preprint arXiv:2210.03629, 2023. [Online]. Available: https://arxiv.org/abs/2210.03629
[8] M. Yin et al., "Agent planning with world knowledge model," in Advances in Neural Information Processing Systems 37 (NeurIPS 2024), 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/d032263772946dd5026e7f3cd22bce5b-Paper-Conference.pdf
[9] X. Liu et al., "AgentBoard: an analytical evaluation board of multi-turn LLM agents," in Advances in Neural Information Processing Systems 37 (NeurIPS 2024), 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/877b40688e330a0e2a3fc24084208dfa-Paper-Datasets_and_Benchmarks_Track.pdf
[10] Z. Gao et al., "DAG-Plan: generating directed acyclic dependency graphs for dual-arm cooperative planning," arXiv preprint arXiv:2406.09953, 2024. [Online]. Available: https://arxiv.org/abs/2406.09953
[11] Y. Wang, R. Xiao, J. Kasahara et al., "DART-LLM: dependency-aware multi-robot task decomposition and execution using large language models," arXiv preprint arXiv:2411.09022, 2024. [Online]. Available: https://arxiv.org/abs/2411.09022
[12] H. Chen et al., "A survey of AIOps in the era of large language models," ACM Computing Surveys, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3746635
[13] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: language agents with verbal reinforcement learning," in Advances in Neural Information Processing Systems 36 (NeurIPS 2023), 2023. [Online]. Available: https://arxiv.org/abs/2303.11366
[14] X. Li, S. Wang, S. Zeng, Y. Wu, and Y. Yang, "A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges," Vicinagearth, vol. 1, no. 1, art. 9, 2024. [Online]. Available: https://doi.org/10.1007/s44336-024-00009-2
[15] Q. Ma et al., "Planning with multi-constraints via collaborative language agents," in Proc. COLING 2025, 2025. [Online]. Available: https://aclanthology.org/2025.coling-main.672
[16] M. Tan et al., "Balancing accuracy and efficiency in multi-turn intent classification for LLM-powered dialog systems in production," arXiv preprint arXiv:2411.12307, 2024. [Online]. Available: https://arxiv.org/abs/2411.12307
[17] J. Wang et al., "Aflow: automating agentic workflow generation," arXiv preprint arXiv:2410.10762, 2024. [Online]. Available: https://arxiv.org/abs/2410.10762
[18] Y. Liu et al., "L4: diagnosing large-scale LLM training failures via automated log analysis," in Proc. 33rd ACM Int. Conf. Foundations of Software Engineering, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3696630.3728531
[19] X. Zhuge et al., "GPTSwarm: language agents as optimizable graphs," in Proc. Int. Conf. Machine Learning (ICML 2024), 2024. [Online]. Available: https://arxiv.org/abs/2402.16823
[20] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: language models can teach themselves to use tools," in Advances in Neural Information Processing Systems 36 (NeurIPS 2023), 2023. [Online]. Available: https://arxiv.org/abs/2302.04761
[21] E. Karpas, O. Abend et al., "MRKL systems: a modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning," arXiv preprint arXiv:2205.00445, 2022. [Online]. Available: https://arxiv.org/abs/2205.00445

[22] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in Advances in Neural Information Processing Systems 35 (NeurIPS 2022), 2022. [Online]. Available: https://arxiv.org/abs/2201.11903
[23] Y. Suthari, Z. Thakker, S. Govindarajan, N. Krishnan, V. Raju, and S. K. Rao Katta, "Cost optimization techniques for efficient resource allocation in cloud computing environments," in Proc. 2025 3rd Int. Conf. Intelligent Cyber Physical Systems and Internet of Things (ICoICI 2025), pp. 793–797, 2025. [Online]. Available: https://doi.org/10.1109/ICoICI65217.2025.11253514
[24] R. Gollapudi, "Autonomous multi-zone replication for zero-loss settlement systems," International Journal of Computational and Experimental Science and Engineering, vol. 12, no. 1, 2026. [Online]. Available: https://doi.org/10.22399/ijcesen.4817
[25] R. Gollapudi, "Risk-controlled near-zero-downtime Oracle database migration using GoldenGate," International Journal of Computational and Experimental Science and Engineering, vol. 8, no. 3, 2022. [Online]. Available: https://doi.org/10.22399/ijcesen.5382
[26] Y. Suthari and K. B. Manam, "Behavioral profiling and AI-powered security for IoT networks in smart city environments," in Proc. 2025 Int. Conf. Artificial Intelligence's Future Implementations (ICAIFI 2025), pp. 41–46, 2025. [Online]. Available: https://doi.org/10.1109/ICAIFI66942.2025.11325861
[27] F. A. Quintero, "Growth-oriented culture within VFX teams: Implications for high-quality effects and simulation innovation," International Journal of Computational and Experimental Science and Engineering, vol. 9, no. 4, 2023. [Online]. Available: https://doi.org/10.22399/ijcesen.5043
[28] P. Sunil Kumar, "Index-state uncertainty for trustworthy enterprise retrieval-augmented generation (RAG)," Journal of Information Systems Engineering and Management, vol. 10, no. 36s, pp. 1258–1271, 2025. [Online]. Available: https://doi.org/10.52783/jisem.v10i36s.15043
[29] D. Bajaj, V. Shah, and S. Kanaparthi, "A practical framework for migrating large manual test suites to automation pipelines: An industrial case study," Journal of Information Systems Engineering and Management, vol. 9, no. 3, pp. 1–8, 2024.
[30] F. A. Clottey, "The role of strategic leadership in strengthening team performance and supply chain resilience for business growth," IPHO-Journal of Advance Research in Business Management and Accounting, vol. 3, no. 7, pp. 16–23, 2025. [Online]. Available: https://doi.org/10.5281/zenodo.19605201