STAGE: Stateful Translation to Agentic Graph Execution with Policy-Scoped Context and Deterministic Control

Mengxi Luo* **Changjia Chen*** **An Cao** **Zirong Huang** **Wanyi Dai**

Abstract

Policy-governed agents must interpret case evidence while following an authorized procedure. We present STAGE, an executable-graph framework that confines model judgment to policy-scoped nodes while placing procedural control in deterministic code. At each node, the model receives task-relevant policy context and returns a typed result, while the coordinator enforces the reviewed execution contract. We evaluate STAGE on SOP-Bench Referral Abuse, two τ^2-bench domains, and Smart Dispute, a proprietary banking benchmark. Compared with monolithic full-policy execution, STAGE generally improves task success and repeated-run reliability across workflows of varying procedural complexity. The largest gains occur on the deeper Telecom and Smart Dispute workflows, where Pass^3 increases by 7.5–55.0 and 57.2–65.7 percentage points, respectively, depending on the model. These results show that combining policy-scoped context with deterministic procedural control can improve the reliability of policy execution.

1 Introduction

Large language models (LLMs) are increasingly deployed as agents that do more than generate text: they interleave reasoning with actions, coordinate with users, and automate multi-turn tasks. Recent work addresses tool use, workflow orchestration, and reliable task completion (Erdogan et al., 2025; Chen et al., 2025). However, these capabilities are necessary but not sufficient for production use.

In regulated and operational settings, agents must follow heterogeneous instructions, such as policies, rules, and manuals. Real-world policies often define long, branching, and multi-stage procedures. A monolithic agent must handle both semantic reasoning and workflow control while tracking relevant rules and procedural state. As context grows, requirements compete with other instructions and observations (Liu et al., 2024), increasing the risk that steps will be omitted or misrouted and that unvalidated judgments will propagate. The challenge is therefore not only to reason correctly, but also to reliably follow the authorized procedure as context and complexity grow.

We introduce STAGE, a graph-structured framework for the reliable execution of complex policies. STAGE separates local semantic reasoning from execution control through *graph-structured execution*, as illustrated in Figure 1. It represents a policy as explicit action and decision nodes connected by valid transitions. Each node receives only the context, state, and tools required for its local responsibility. In short, *the LLM owns local meaning, while code owns control*. By encoding task focus, procedural state, permitted capabilities, and compliance requirements at each node, the executable graph supports reliable policy execution as context and complexity grow. As a result, STAGE is specifically designed for long, branching, and state-dependent workflows in which procedural correctness cannot be reduced to a single final decision.

Our contributions are threefold. First, we identify procedural complexity, including long dependencies, conditional branching, non-local transitions, and mixed decision and action steps, as a central source of unreliability in policy-following agents. Second, we introduce STAGE, which externalizes policy control into an executable graph with locally scoped nodes and deterministic coordination. Third, we evaluate STAGE against a monolithic full-policy execution baseline and use fixed-graph ablations to isolate the effects of policy scoping and authored node instructions. Our evaluation includes Smart Dispute, an internal industrial workflow in which STAGE is deployed, and three public policy-following benchmarks. STAGE consistently improves execution reliability, with the largest gains on deeper and more conditional

*These authors contributed equally.

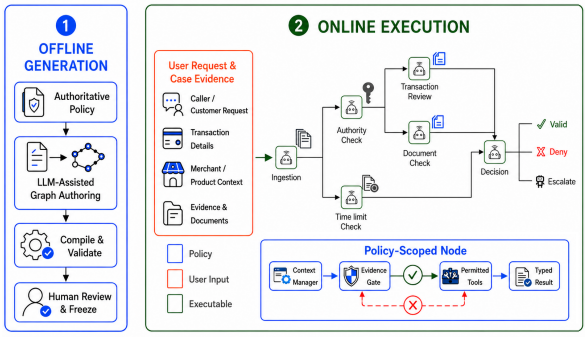


Figure 1: STAGE illustration: STAGE constructs and freezes a human-reviewed execution graph offline (left), then uses policy-scoped nodes to validate evidence, constrain capabilities, and route decisions online (right).

workflows. The same trend appears on the public benchmarks, supporting generalization beyond our deployment.

2 Problem Formulation

2.1 Experimental Conditions

Using the reviewed and frozen graph described in Section 3.1, we compare three execution conditions. In the **baseline**, one agent receives the complete policy and controls both semantic reasoning and procedural progression. In **graph with full policy**, the reviewed graph controls execution, but every node receives the complete policy. In **node-scoped STAGE**, the same graph is used, but each node receives only the task-relevant policy context grounded in its mapped source passages. The two graph conditions use the same topology, node objectives, case evidence, capabilities, result interfaces, routing rules, and attempt budgets.

Comparing the baseline with node-scoped STAGE measures the combined effect of bounded node execution, coordinator-owned control, and localized policy context. Comparing the two graph conditions keeps the control structure fixed and isolates the effect of policy localization. Together, these comparisons test whether graph-owned control and policy-scoped context improve execution reliability without withholding information needed for correct judgment. We also examine whether these gains persist as execution chains become longer.

2.2 Research Questions

RQ1: Graph-structured execution. How does STAGE affect task success and repeated-run reliability relative to monolithic full-policy execution?

RQ2: Policy localization under fixed control. Holding the graph fixed, how does node-scoped policy context affect task success, repeated-run reliability, and token consumption relative to presenting the complete policy at every node?

RQ3: Robustness to procedural complexity. How do STAGE's performance advantage and error composition vary with execution-chain length?

3 STAGE

3.1 Executable Policy Graph

We construct the execution graph offline from an authoritative policy. An LLM-assisted procedure decomposes the policy into bounded tasks, maps exact source passages to each task, and authors the corresponding nodes and transitions. The candidate graph is then validated, reviewed and frozen before evaluation. The construction procedure is described in Appendix A.

3.2 Compliance-Carrying Execution Nodes

Each node is an executable contract for one bounded policy judgment. It specifies the local task, permitted skills and tools, result validation, and failure behavior. These constraints define both the model's scope and the results that the runtime may accept.

At runtime, the coordinator resolves the current node and presents its contract together with the available case information, shared memory, and selected policy view. The model returns a typed result containing its judgment, references to supporting case evidence, and an outcome when required. The source-grounded node contract supplies the policy basis for that judgment. The coordinator validates the typed result before routing, retry, termination, and finalization into the auditable runtime record.

3.3 Coordinator-Owned Control

After each attempt, the coordinator validates the result against the current node contract and applies the graph-declared route, retry, escalation, or termination behavior. Because successor selection is outside the model, the model cannot introduce an undeclared transition or bypass the node's attempt limits. The coordinator controls progression through the reviewed procedure.

4 Evaluation

4.1 Benchmarks and Models

We evaluate STAGE on four workflows with different levels of procedural complexity. SOP-Bench Referral Abuse tests policy-guided investigation and decision-making with executable tools (Nandi et al., 2025). Retail and Telecom from τ^2-bench represent shallower and deeper workflows, respectively (Barres et al., 2025). Smart Dispute is based on a production banking policy and provides the deepest workflow. The reconstructed graphs are shown in Appendix B.

We evaluate the public workflows with GPT-5.6 Luna, DeepSeek V4 Flash, Claude Haiku 4.5, and Sonnet 5. Because Smart Dispute contains proprietary banking knowledge, we evaluate it only with Claude Haiku 4.5 and Sonnet 5 in the approved internal environment.

4.2 Protocol and Measures

We construct, review, and freeze one graph for each workflow before evaluation. Within each comparison, we use the same model, task input, environment, and evaluator. Public workflows retain their benchmark tools and success criteria, while Smart Dispute uses its approved internal evaluation specification.

All conditions use the same DeepAgent scaffold. RQ1 compares the full-policy baseline with node-scoped STAGE. RQ2 holds the graph fixed and changes only the policy context supplied to each node. RQ3 groups Smart Dispute outcomes by realized execution-chain length.

We report single-run success as Pass^1 and the proportion of cases completed successfully in all three trials as Pass^3. For RQ2, we also report input, output, and total token consumption. For RQ3, we classify each run as accurate, a chain error, or an execution/end error. Correct completions are accurate. A completed run that skips or misorders a required step is a chain error, even when its final result is also wrong. All other unsuccessful runs, including non-completion and process or tool failures, are execution/end errors.

5 Results

Table 1 reports single-run success and repeated-run reliability across workflows and models.

5.1 RQ1: Overall Effectiveness

Public benchmarks. Referral Abuse follows a relatively constrained six-stage sequence, and its stronger baselines are already close to ceiling. STAGE therefore provides its clearest gains for the weaker models: Luna's Pass^3 increases from 88.0% to 95.5%, while Haiku improves from 79.0% to 88.5%. Sonnet and DeepSeek remain near saturation under both conditions, leaving little headroom for further improvement.

Retail contains only five nodes and one decision, yet STAGE improves both Pass^1 and Pass^3 across all four models. The largest gain occurs with DeepSeek, whose Pass^1 increases by 32.5 percentage points, while Pass^3 gains range from 15.0 to 25.0 points across models. Thus, even on a shallow graph, explicit procedural state makes successful execution more reproducible across model families.

The advantage is more pronounced on Telecom, whose 13-node graph requires deeper routing and coordination between agent and user actions. STAGE improves both measures across all four models, with the largest gains among models with lower baseline performance: Luna's Pass^1 increases from 47.5% to 90.0%, while Pass^3 rises from 32.5% to 87.5%. Haiku improves from 55.0% to 72.5% on Pass^1 and from 25.0% to 50.0% on Pass^3. Gains remain visible for Sonnet and DeepSeek.

Smart Dispute. Smart Dispute provides the most demanding evaluation, with 26 nodes, nine decisions, and execution paths of up to 15 steps. STAGE produces large gains for both models. With Haiku, Pass^1 increases from 14.3% to 74.3%, while Pass^3 rises from 11.4% to 68.6%. With Sonnet, Pass^1 increases from 42.9% to 88.6%, and Pass^3 from 14.3% to 80.0%. These results show that graph-structured execution improves not only isolated task completion but also repeated-run reliability on long policy-governed workflows. RQ3 returns to Smart Dispute to analyze this behavior by execution-chain length.

5.2 RQ2: Effect of Policy Localization

RQ2 isolates policy localization using the same reviewed Telecom graph and execution configuration. Only the policy context supplied to each node differs.

Node-scoped context improves both measures for both models. Pass^1 increases by 5.0 percentage

Benchmark	Model	$\mathbf{Pass^1}$ (%) Base	STAGE	Δ	$\mathbf{Pass^3}$ (%) Base	STAGE	Δ
Referral Abuse	GPT-5.6 Luna	94.0	**98.0**	+4.0	88.0	**95.5**	+7.5
	Claude Haiku 4.5	91.0	**98.0**	+7.0	79.0	**88.5**	+9.5
	Claude Sonnet 5	**100.0**	99.5	−0.5	**99.0**	**99.0**	0.0
	DeepSeek V4 Flash	98.5	**99.0**	+0.5	95.5	**97.0**	+1.5
τ^2-bench Telecom	GPT-5.6 Luna	47.5	**90.0**	+42.5	32.5	**87.5**	+55.0
	Claude Haiku 4.5	55.0	**72.5**	+17.5	25.0	**50.0**	+25.0
	Claude Sonnet 5	80.0	**95.0**	+15.0	70.0	**80.0**	+10.0
	DeepSeek V4 Flash	67.5	**82.5**	+15.0	45.0	**52.5**	+7.5
τ^2-bench Retail	GPT-5.6 Luna	57.5	**70.0**	+12.5	37.5	**60.0**	+22.5
	Claude Haiku 4.5	52.5	**60.0**	+7.5	27.5	**45.0**	+17.5
	Claude Sonnet 5	55.7	**65.0**	+9.3	47.5	**62.5**	+15.0
	DeepSeek V4 Flash	40.0	**72.5**	+32.5	10.0	**35.0**	+25.0
Smart Dispute	Claude Haiku 4.5	14.3	**74.3**	+60.0	11.4	**68.6**	+57.2
	Claude Sonnet 5	42.9	**88.6**	+45.7	14.3	**80.0**	+65.7

Table 1: Single-run task success and repeated-run reliability. Pass^1 denotes single-run success, while Pass^3 requires successful completion in all three runs. Δ is the absolute percentage-point difference between STAGE and the baseline.

Model	Graph condition	$\mathbf{Pass^1}$ (%)	$\mathbf{Pass^3}$ (%)	Avg. output (K tokens/case)	Avg. input (K tokens/case)	Avg. total (K tokens/case)
Claude Haiku 4.5	Full policy	87.50	22.50	11.44	1890.39	1901.83
	Node-scoped STAGE	**92.50**	**50.00**	**10.53**	**1173.36**	**1183.89**
Claude Sonnet 5	Full policy	95.00	57.50	5.81	881.08	886.89
	Node-scoped STAGE	**100.00**	**80.00**	**5.52**	**377.26**	**382.78**

Table 2: Fixed-graph policy-view comparison on the 40 Telecom cases. Both conditions use the same execution graph and coordinator. Only the policy context supplied to each node changes. Average total tokens are the sum of average input and output tokens.

points for Haiku and Sonnet, while Pass^3 improves by 27.5 and 22.5 points, respectively. Average total token consumption simultaneously falls by 37.8% for Haiku and 56.8% for Sonnet, primarily through lower input consumption.

Taken together, the accuracy and token results support policy localization as a component distinct from graph control. Restricting each node to task-relevant policy context reduces competition from unrelated instructions, improves repeated-run reliability, and substantially lowers the amount of policy text processed during execution.

5.3 RQ3: Robustness to Procedural Complexity

To answer RQ3, we stratify Smart Dispute outcomes by realized execution-chain length (Figure 2). STAGE outperforms the baseline for both models in every group, with gains peaking on medium chains ($+73.3$ percentage points for Haiku and $+66.7$ for Sonnet) and remaining substantial on long chains ($+43.3$ and $+46.7$ points, respectively). Accuracy nevertheless declines on long chains, and the error composition clarifies this pattern: STAGE reduces chain-error rates to at most 6.7% for Haiku and 3.3% for Sonnet, leaving execution/end errors as the main source of remaining failures. Within Smart Dispute, these results indicate that explicit state and coordinator-owned routing improve robustness to procedural depth, although long-horizon completion remains challenging.

5.4 Additional Analysis

End-to-end token trade-offs. Table 3 reports average input and output token consumption. The reliability gains involve a model- and workflow-dependent token trade-off. STAGE executes multiple node-scoped calls, which generally increases output-token consumption, while accumulated input consumption depends on path length, model behavior, and context reuse. The increase is substantial in several public-model settings, showing that the current implementation should be understood primarily as a reliability and control mechanism rather than a token-reduction method.

The trade-off is not universal. On Telecom, Luna's average input consumption decreases by 17.3%, from 253.15K to 209.47K tokens per case, while its Pass^3 increases by 55 percentage points. On Smart Dispute with Sonnet, input consump-

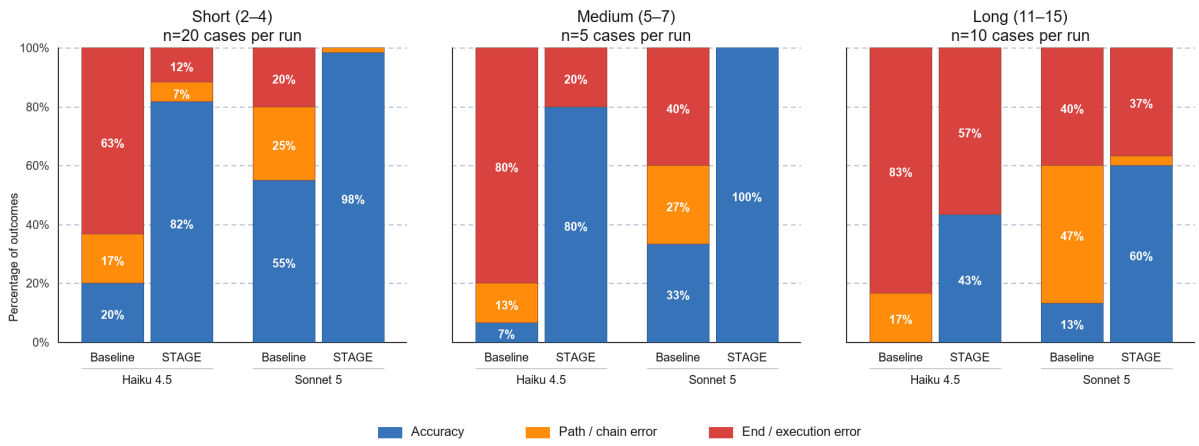


Figure 2: Smart Dispute accuracy and mutually exclusive error composition by model, method, and realized execution-chain length, aggregated over three runs. Path/chain errors denote completed runs that violate the required path or order and take priority over terminal correctness. Execution/end errors comprise non-completion, process or tool failure, or an incorrect terminal result without a chain error.

		Output tokens		Input tokens	
Benchmark	**Model**	**Base**	**STAGE**	**Base**	**STAGE**
Referral Abuse	GPT-5.6 Luna	0.68	2.70	35.39	152.21
	Claude Haiku 4.5	1.61	15.99	9.86	81.92
	Claude Sonnet 5	1.78	3.74	14.07	70.48
	DeepSeek V4 Flash	2.68	11.29	14.15	97.36
τ^2-bench Telecom	GPT-5.6 Luna	2.40	5.57	253.15	**209.47**
	Claude Haiku 4.5	2.89	10.53	254.78	1173.36
	Claude Sonnet 5	1.85	5.52	283.09	377.26
	DeepSeek V4 Flash	3.29	10.51	193.94	1344.41
τ^2-bench Retail	GPT-5.6 Luna	2.03	7.64	85.74	202.27
	Claude Haiku 4.5	1.90	3.58	130.72	311.15
	Claude Sonnet 5	1.58	3.22	146.80	216.67
	DeepSeek V4 Flash	2.61	6.11	127.19	407.07
Smart Dispute	Claude Haiku 4.5	3.61	3.67	68.73	78.91
	Claude Sonnet 5	1.81	4.65	153.23	**96.76**

Table 3: Average input and output tokens per case, reported in thousands of tokens. Bold input-token values identify settings in which STAGE consumes fewer input tokens than the corresponding baseline. Token consumption is reported as a runtime characteristic rather than the primary optimization objective.

tion decreases by 36.9%, from 153.23K to 96.76K, alongside an increase in Pass^3 from 14.3% to 80.0%. These cases show that node-scoped context can offset coordination overhead when it prevents long or repeated reasoning over the full policy. Improving cache reuse and consolidating simple nodes remain opportunities for further efficiency gains.

Post-generation refinement. The Smart Dispute graph used for the main results was evaluated exactly as generated, without post-generation manual adjustment. In a separate diagnostic experiment, we used its node-localized execution traces and error messages to perform one targeted revision round, after which task accuracy increased to 94%. This refined result is not included in the main results, but it illustrates an additional benefit of the structured design: failures are attached to explicit nodes, transitions, and typed outputs, making them easier to diagnose and correct than failures embedded in a monolithic trajectory.

6 Related Work

Prior work on policy-constrained agents primarily checks whether behavior selected by an agent is permissible. ShieldAgent and AgentLTL verify actions or traces against temporal rules (Chen et al., 2025; Elkoussy and Perez, 2026), while ToolGuards, solver-aided checking, and AgentSpec gate tool calls using generated, reviewed, or authored

constraints (Zwerdling et al., 2025; Winston et al., 2026; Wang et al., 2025). Although their policy representations differ, these systems share a control boundary: a persistent agent selects the next task-level behavior, and an enforcement layer judges whether it may proceed. STAGE instead places policy control before action selection by making the reviewed procedure determine which policy judgment is active and what execution may follow.

Workflow-oriented systems are the closest comparisons. JourneyBench executes reviewed SOP DAGs (Balaji et al., 2026); Compile, Then Page compiles SOP constraints into executable, paged programs (Yu et al., 2026); Declarative Skills studies phase-based orchestration (Lim et al., 2026); and COVENANT compiles natural-language instructions into controller-interpreted WCFGs (Wang et al., 2026). These works establish explicit workflow state and controller-mediated execution. Building on this direction, STAGE centers the compliance-carrying execution node: a reviewed contract that binds source policy passages, visible case evidence, permitted capabilities, admissible outcomes, result schemas, and failure and transition behavior. The graph is therefore not only a representation of control flow, but the authorized interface around every policy judgment. The model resolves the bounded semantic question within that interface, while the coordinator enforces which judgment is active and how the reviewed procedure may advance. This design targets long, branching, and state-dependent policies, where compliance depends on the provenance and ordering of intermediate judgments rather than only on the permissibility of a final action.

7 Conclusion

We studied whether policy-governed agents benefit from separating local model judgment from graph-owned execution control. Across the public benchmarks and Smart Dispute, STAGE substantially improved task success and repeated-run reliability over monolithic full-policy execution. The Smart Dispute chain-length analysis further shows that these gains persist on long workflows and that STAGE removes most errors caused by skipped or misordered procedural steps. The remaining failures on the longest chains are concentrated in execution and terminal completion.

The fixed-graph comparison further shows that policy representation matters independently of control structure. Node-scoped policy context improves success and repeated-run reliability while reducing the total context processed relative to full-policy exposure. These findings support source-grounded node contracts as a practical way to concentrate model reasoning without surrendering procedural control. A conforming trace still does not guarantee a correct semantic judgment, but it makes the execution path bounded, reviewable, and auditable. Overall, the results show that reliable policy execution depends not only on stronger models but also on how policy context and control authority are structured around them.

References

Sumanth Balaji, Piyush Mishra, Aashraya Sachdeva, and Suraj Agrawal. 2026. Beyond IVR: Benchmarking customer support LLM agents for business-adherence. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 193–208. Association for Computational Linguistics.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Zhaorun Chen, Mintong Kang, and Bo Li. 2025. ShieldAgent: Shielding agents via verifiable safety policy reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 8313–8344. PMLR.

Laïla Elkoussy and Julien Perez. 2026. AgentLTL: A trace-verification framework for measuring, enforcing, and training procedural compliance in tool-using LLM agents. *arXiv preprint arXiv:2607.02599*.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 15419–15462. PMLR.

M. Danish Lim, I. Danial Bin Sharudin, Wen Han Chen, Cedric Lim, and Laura Wynter. 2026. Declarative skills for AI agents in knowledge-grounded tool-use workflows. *arXiv preprint arXiv:2606.06923*.

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

Subhrangshu Nandi, Arghya Datta, Rohith Nama, Udita Patel, Nikhil Vichare, Indranil Bhattacharya, Prince Grover, Shivam Asija, Giuseppe Carenini, Wei Zhang, Arushi Gupta, Sreyoshi Bhaduri, Jing Xu, Huzefa Raja, Shayan Ray, Aaron Chan, Esther Xu Fei, Gaoyuan Du, Zuhaib Akhtar, and 5 others. 2025. SOP-Bench: Complex industrial SOPs for evaluating LLM agents. *arXiv preprint arXiv:2506.08119*.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. 2025. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. *arXiv preprint arXiv:2503.18666*.

Jincheng Wang, Min Zheng, and Tao Wei. 2026. COVENANT: Natural-language workflow compilation for aligned agent execution. *arXiv preprint arXiv:2607.25400*.

Cailin Winston, Claris Winston, and René Just. 2026. Solver-aided verification of policy compliance in tool-augmented LLM agents. *arXiv preprint arXiv:2603.20449*.

Chenglin Yu, Li Yin, Qingxin Fan, Ying Yu, RunyangRay Zhong, and Ming Li. 2026. Compile, then page: Executable SOP programs and a capability-gated runtime for procedural LLM agents. *arXiv preprint arXiv:2607.11346*.

Naama Zwerdling, David Boaz, Ella Rabinovich, Guy Uziel, David Amid, and Ateret Anaby Tavor. 2025. Towards enforcing company policy adherence in agentic workflows. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 595–606. Association for Computational Linguistics.

A Policy-to-graph Construction

Figure 3 summarizes the offline construction workflow. The LLM-assisted stage segments the policy and jointly extracts procedural tasks while authoring nodes, routing, and shared memory. Candidate fan-in, fan-out, or convergence is reorganized through policy-faithful absorption, serialization, or duplication and then rechecked. The resulting plan is deterministically compiled and validated before human review and freezing. If no faithful reorganization is possible, generation stops for review rather than emitting an invented rule.

B Reconstructed Benchmark Graphs

Figures 4 and 5 show the reviewed execution graphs used in the evaluation. SOP-Bench Referral Abuse contains six sequential stages for risk calculation, violation classification, severity assessment, and enforcement. Retail contains five nodes and one decision, providing a compact routing structure. Telecom contains 13 nodes and two decisions, adding wider request routing and a deeper technical-support branch. Smart Dispute contains 26 nodes and nine decisions, producing the longest and most conditional execution structure.

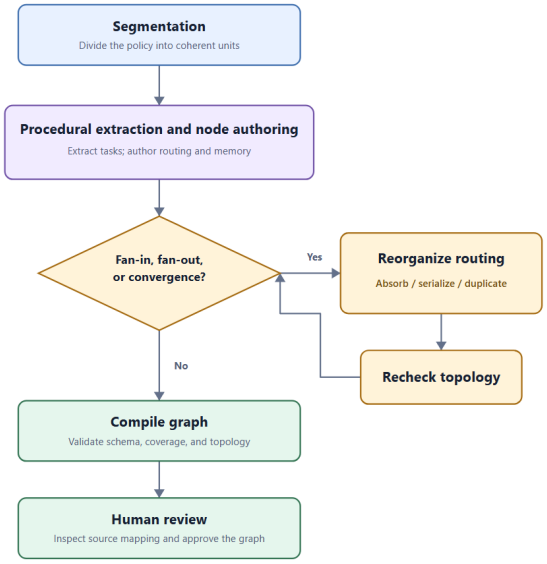


Figure 3: Offline policy-to-graph construction. Semantic authoring produces a source-grounded procedural plan, while topology checks, compilation, and validation constrain the plan before human review.

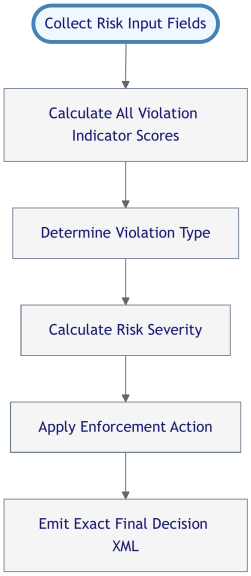
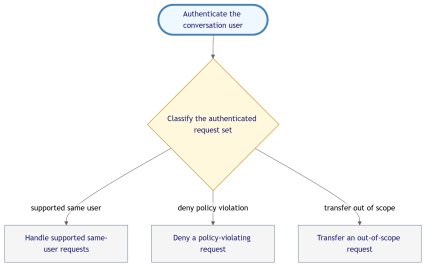
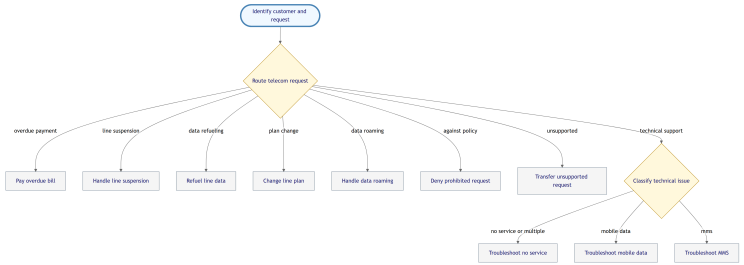


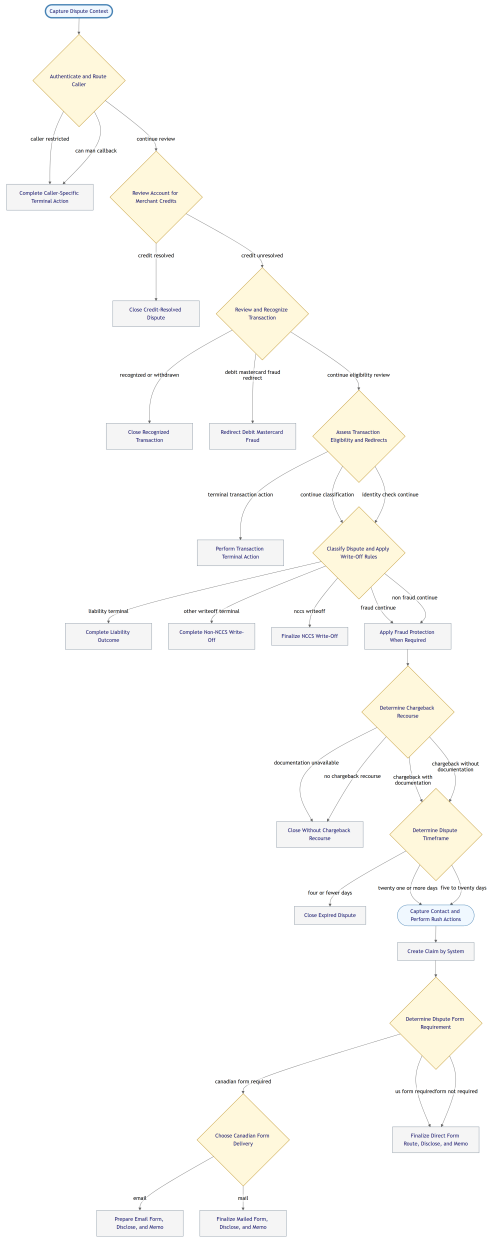
Figure 4: Reconstructed SOP-Bench Referral Abuse execution graph.



(a) Retail



(b) Telecom



(c) Smart Dispute

Figure 5: Reconstructed benchmark execution graphs. Retail provides the shallowest routing structure, Telecom adds broader request routing and a nested technical-support decision, and Smart Dispute contains the deepest sequence and most conditional structure. The panels include only non-sensitive node titles and outcome labels.