Policy-Aware LLM Agents for Regulated Customer Service Workflows

Anonymous ACL submission

Abstract

Regulated customer-service agents fail not only because a language model misreads intent, but because business rules, action schemas, channels, and workflow timing keep changing after deployment. We study this as a constrained action-commitment problem and present the *Policy-Graph Constrained Agent* (PGCA). PGCA combines a shared semantic proposer, a trained guard classifier, a rule-based safety-prior calibrator, and a typed action projector that can emit a committed action, handoff, or abstain. To evaluate the action layer directly, we turn repository-native evidence into an executable benchmark with 500 cases: 40 cases map directly to tests, repairs, or de-identified incidents in a production-style workflow platform, and 460 cases are repo-native perturbations spanning five drift families. We also construct a 30-case public-benchmark-inspired transfer slice using the same typed action schema, including task patterns adapted from JourneyBench, Flow-Bench, STATE-Bench, and Tau-Bench. To avoid optimistic overfitting estimates, we evaluate guard prediction under blueprint-grouped cross-validation: naive random folds reach 0.990 micro-F1, while grouped raw selection drops to 0.720; after safety-prior calibration, the deployed selector reaches 0.794 micro-F1 and 0.869 macro-F1, with blueprint leakage quantified as 1.00 under random folds and 0.00 under grouped folds. On the repo-native benchmark, PGCA reaches 0.95 compliant action accuracy (95% Wilson CI [0.93, 0.97]) with 0.00 policy violation rate, compared with 0.60/0.35 for a dynamic-prompt family and 0.20/0.65 for a tool-augmented baseline; bootstrap deltas against Dynamic Prompt are +0.35 compliant accuracy and -0.35 violation rate, and exact McNemar tests yield $p < 10^{-10}$ against all three baselines. Relative to a raw grouped selector, calibration improves compliant accuracy from 0.84 to 0.95 and closes residual violations from 0.10 to 0.00; relative to a dense full-guard baseline, PGCA preserves 0.00 violation while improving compliant accuracy from 0.886 to 0.95. We further add a 50-task end-to-end corroboration slice that runs real committed actions through the native WorkflowEngine plus SMS/transfer/hangup executors: PGCA reaches 1.00 planned action accuracy and 0.50 committed-action accuracy, matching the strongest dynamic baseline overall while improving over static prompting and the tool-augmented family. The scope is intentionally narrow. We do not claim general agent reliability or end-to-end text generation superiority; we study typed action commitment for regulated customer-service workflows under business drift.

1 Introduction

Customer-service LLM agents increasingly do more than write helpful text. They decide whether to continue an opening script, whether to dispatch an SMS, which channel to use, whether to hand a case to a human, and whether the workflow can terminate. In regulated operations such as insurance refund support, these decisions sit behind changing business policies: a second scripted opening turn becomes mandatory, a smart-SMS parameter family is renamed, a delivery channel is split into `reach/marketing/third_party`, or an asynchronous node must be joined before another action is legal.

The practical question is therefore not only "can the model understand the user?" but also "which parts of the workflow should the model *not* control directly?" Recent customer-support and workflow benchmarks increasingly model trajectories and state (Balaji et al., 2026; Shi et al., 2026; Liu and Yadav, 2026; Yao et al.,

2025; Xiao et al., 2024; Liu et al., 2024; Ma et al., 2024; Mialon et al., 2024), yet most still assume that action legality is stable during evaluation. In production deployments, the opposite is often true: the semantics of an action can drift because tool schemas, channel rules, audit requirements, and workflow timing change after release.

This paper studies that action-layer problem. Our empirical starting point is a real customer-service workflow platform with specialized LLM-connected nodes for dialogue, transfer, hangup, and SMS dispatch. Earlier drafts of our work over-centered replay evaluation. This revision recenters the technical core on a policy-aware method. We introduce the *Policy-Graph Constrained Agent* (PGCA), which separates semantic proposal generation from high-risk action commitment. PGCA activates a trained subset of workflow guards, calibrates them with explicit safety priors, and projects the proposed action onto a typed legal set. The resulting architecture is still simple enough for production, but it is no longer only a design principle: it is an executable decision system with a benchmark, quantitative results, overfitting diagnostics, and a diagnostic formal lens on action-layer violations.

We make four contributions:

1. We introduce PGCA, a constrained action-commitment method with four components: a shared semantic proposer, a trained tree-based guard classifier, a rule-based safety-prior calibrator, and a typed action projector.

2. We build an executable policy-aware benchmark with 500 repo-native cases (40 direct + 460 perturbation) across five workflow-drift families, plus a 30-case public-benchmark-inspired external transfer slice, and a 50-task end-to-end corroboration slice executed through the native workflow runtime.

3. We show quantitative gains over three comparison families and four key ablations. On the repo-native benchmark, PGCA improves compliant action accuracy from 0.60 to 0.95 over the strongest prompt baseline while reducing policy violations from 0.35 to 0.00; calibration improves over a raw grouped selector from 0.84/0.10 to 0.95/0.00; and PGCA improves over a dense full-guard baseline from 0.886 to 0.95 at the same 0.00 violation rate. On the end-to-end slice, PGCA preserves 1.00 planned-action accuracy and 0.50 committed-action accuracy under real WorkflowEngine execution, confirming the action-layer story without replacing the main benchmark.

4. We formalize the action layer, state a decomposition bound, and provide a proof sketch: under a required-guard coverage assumption, action violations are upper-bounded by missed-required-guard events plus unsound-guard acceptance events.

The scope is intentionally narrow. We do not claim a universal benchmark or full production validation. Our claim is that, for regulated customer-service workflows, action legality must be modeled as a first-class decision problem alongside LLM semantics.

2 Related Work

Customer-service and workflow benchmarks. JourneyBench and SAGE move toward process-sensitive customer-support evaluation (Balaji et al., 2026; Shi et al., 2026), while STATE-Bench adds persistent state and repeated-run reliability to enterprise-style support tasks (Liu and Yadav, 2026). More general agent benchmarks such as τ-bench, FlowBench, AgentBench, AgentBoard, and GAIA broaden the picture to tool use, workflow execution, and broad assistant behavior (Yao et al., 2025; Xiao et al., 2024; Liu et al., 2024; Ma et al., 2024; Mialon et al., 2024). These efforts are directly relevant, but they largely hold policy fixed during evaluation.

Tool-use and workflow execution. ReAct and Toolformer established the modern pattern of interleaving reasoning, acting, and tool invocation (Yao et al., 2023; Schick et al., 2023). API-Bank, Gorilla, and ToolLLM extend that thread toward API-rich evaluation and tool specialization (Li et al., 2023; Patil et al., 2024; Qin et al., 2024). WebShop and ALFWorld provide grounded interaction settings, while Mind2Web, WebArena, VisualWebArena, SWE-bench, and Data Interpreter make the

environments more realistic and action-heavy (Yao et al., 2022; Shridhar et al., 2021; Deng et al., 2023; Zhou et al., 2023; Koh et al., 2024; Jimenez et al., 2024; Hong et al., 2025). Our work shares that trajectory perspective, but it asks a different question: not only whether a tool sequence succeeds, but which actions should remain outside unconstrained model control.

Constraint-aware agent safety. Constraint-aware and self-corrective agent work argues that runtime legality, critique, and recovery should be made explicit rather than left to prompting alone (Kamath et al., 2025; Bai et al., 2022; Gou et al., 2024; Shinn et al., 2023; Madaan et al., 2023). Multi-agent orchestration frameworks such as AutoGen, MetaGPT, and AgentScope make workflow structure explicit, but they do not directly target typed legality under post-deployment business drift (Wu et al., 2023; Hong et al., 2024; Gao et al., 2024). We adapt the constraint lesson to regulated customer-service workflows and go one step further by learning which guards should activate under business drift.

Knowledge and drift. Knowledge-drift work emphasizes post-deployment change in real-world systems (Liu et al., 2026), while evaluation work on LLM judges reminds us that conversation-level scores can hide structured action failures (Zheng et al., 2023). We narrow that perspective to workflow operations: in customer-service agents, drift often changes whether an action is legal, not only whether an answer string is factually current.

3 Policy-Graph Constrained Agent

3.1 System Structure

We model a regulated customer-service agent as a workflow graph $\mathcal{G} = (V, E)$ whose nodes are typed executors and whose edges encode business transitions. In the platform we study, the important node families are:

1. a dialogue node that produces the next customer-facing utterance;
2. a transfer node that decides whether to hand the case to a human;

Node family	Semantic role	Main guard pressure	Typed outputs
Dialogue	continue local reply	opening config / slot repair	reply or scripted continuation
Transfer	infer handoff need	script + audit boundary	handoff flag + script
Hangup	infer terminate need	script + silence boundary	hangup flag + script
SMS	infer send need	channel / schema / join state	typed SMS action

Table 1: Specialized nodes in the policy-aware workflow agent.

3. a hangup node that decides whether the conversation should terminate; and
4. an SMS node that decides whether to dispatch a notification and through which channel.

These nodes are all LLM-connected, but they are not equally free-form. Dialogue can emit broader text; transfer, hangup, and SMS are constrained by typed outputs, omission checks, standard scripts, channel schemas, and execution-state guards.

Figure 1 summarizes the resulting left-to-right control flow: PGCA first reuses a shared semantic proposal bank, then compresses observable workflow state into a required guard set, and finally projects the candidate action onto the typed feasible set before native execution.

3.2 Formalization

Let x_t denote the observable state at turn t, including dialogue history, workflow stage, side-effect counters, visible config deltas, and node-local tool signals. Let c_t denote the current workflow configuration and $\mathcal{A}$ the typed action space. We distinguish a high-risk subset $\mathcal{H} \subset \mathcal{A}$ containing irreversible or regulated actions such as handoff, hangup, and channel-specific SMS dispatch.

The shared semantic proposer emits a ranked action list

$$z_t = (a_1, a_2, \ldots, a_m), \qquad a_i \in \mathcal{A},$$

without committing to legality. PGCA then activates a subset of guards $\hat{G}_t \subseteq \mathcal{K}$ using a trained classifier

$$\hat{G}_t = h_\theta(x_t, c_t),$$

and calibrates that set with explicit safety priors

$$\tilde{G}_t = C(\hat{G}_t, x_t, c_t),$$

where each guard corresponds to an executable legality check such as `compat_guard`, `schema_guard`, `channel_guard`, `join_guard`,

or `script_guard`. Each guard g_k induces a predicate over actions,

$$g_k(a, x_t, c_t) \in \{0, 1\}.$$

The feasible action set is

$$F_t = \{a \in \mathcal{A} \mid g_k(a, x_t, c_t) = 1 \ \forall g_k \in \tilde{G}_t\}.$$

The typed action projector commits the highest-ranked feasible action, or falls back to a boundary action if no feasible action exists:

$$y_t = \arg \max_{a \in F_t \cup \{\texttt{abstain}\}} s(a \mid z_t, x_t).$$

Trained classifier with rule-based safety priors. We instantiate h_θ as a lightweight tree-based multi-label predictor over workflow-state features: node type, risk tier, channel hint, asynchronous pending state, audit availability, confirmation evidence, slot presence, and side-effect counters. The calibrator C then adds non-negotiable high-risk guards when explicit signals make them mandatory (e.g., `join_guard` under asynchronous pending state, `channel_guard` after channel splits, `schema_guard` under parameter-family drift, and `script/audit` guards at transfer or hangup boundaries), while removing guards contradicted by shadow signals such as cached consent or a resolved audit trace. We choose this small model because the training signal comes from bounded workflow artifacts rather than large-scale online logs; the calibrator is the mechanism that injects non-negotiable workflow priors without forcing dense all-guard activation.

Violation decomposition bound. Proposition 1. Let $G_t^\star$ be a sound required-guard set such that every illegal action at turn t is rejected by at least one guard in $G_t^\star$, and let $\mathcal{L}(x_t, c_t)$ denote the legal action set. Define the miss event $M_t = \{G_t^\star \not\subseteq \tilde{G}_t\}$ and the unsound-accept event $U_t = \{\exists g_k \in \tilde{G}_t : g_k(a, x_t, c_t) = 1 \text{ for an illegal } a\}$. If the projector commits only actions in F_t (or abstains when $F_t = \emptyset$), then

$$\Pr[y_t \notin \mathcal{L}(x_t, c_t)] \leq \Pr[M_t] + \Pr[U_t].$$

Proof sketch. If neither M_t nor U_t occurs, then every illegal action is blocked by at least one active sound guard, so no illegal action belongs to F_t. The projector either selects inside F_t or abstains, hence it cannot commit an illegal action. Therefore a violation implies $M_t \cup U_t$, and the stated bound follows by the union bound. □

4 Repository-Native Workflow Drift

Our starting point is a repository-level version analysis covering the upgrade from version 1.1.19 to 1.1.23. The analysis spans 42 commits over 24 days, with 24 SMS-related commits accounting for roughly 61% of the reported work volume. The dominant changes were action-layer changes: channel branching, template binding, parameter parsing, workflow timing, and configuration compatibility.

The taxonomy is intentionally narrower than a universal ontology. Its purpose is to define a faithful, executable action space for this platform rather than to invent new failure categories in the abstract.

Concretely, the repo-backed drift space clusters into five families: *prompt/config*, *tool/schema*, *channel/policy*, *workflow-state*, and *action-boundary*. These families cover the dominant repository changes we observed—opening-script compatibility, parameter-family drift, SMS channel splits, asynchronous join timing, and typed transfer/hangup boundaries—and they define the executable action space used throughout the benchmark. We place the full taxonomy table in Appendix 4 so the main text can stay focused on the method and the primary results.

5 Executable Benchmark

5.1 Case Construction

We convert repository evidence into a benchmark schema with three visible fields and one hidden oracle: `observable_context`, `workflow_config`, `meta`, and `hidden_oracle`. The hidden oracle stores the legal action set, the preferred action, the expected boundary requirement, and the gold active guards. Family inputs never receive those oracle fields.

The main benchmark contains 500 repo-native cases: 40 cases map directly to repository tests, reports, or de-identified incidents, and 460 are repo-native perturbations generated from 20 blueprint templates (25 variants per template). Relative to the first an-

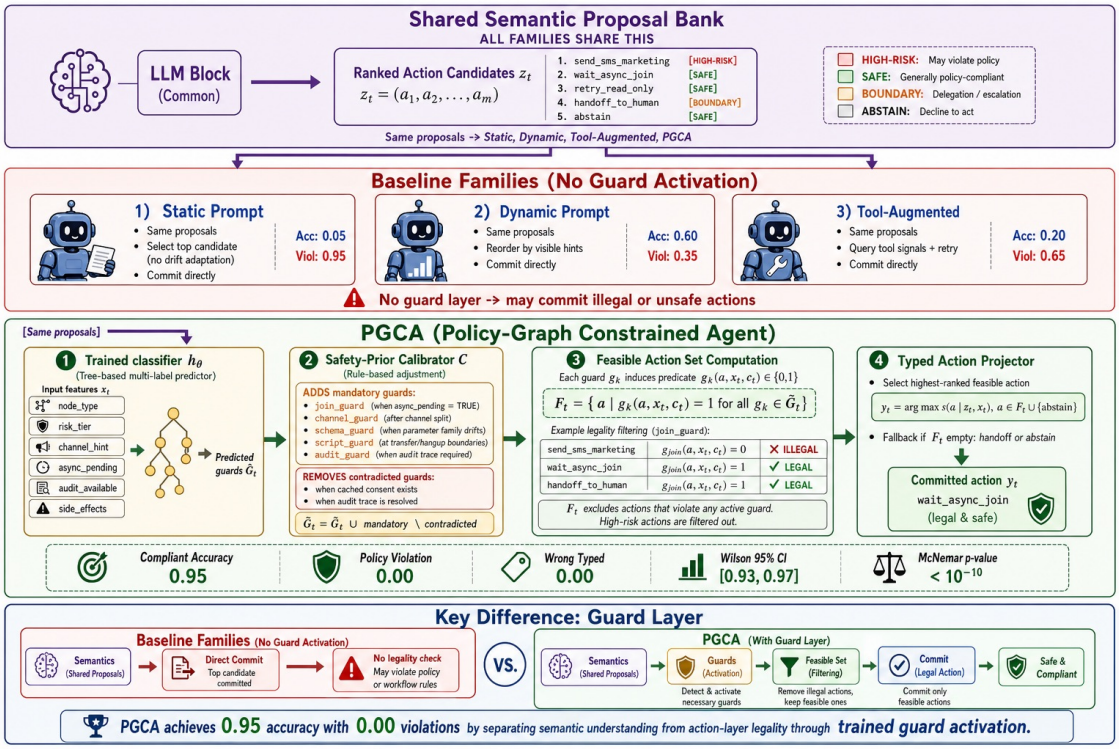

Figure 1: Policy-Graph Constrained Agent. All families share the same semantic proposal bank; PGCA differs by learning guard activation and projecting onto a typed feasible action set.

chor variant, perturbations change 5.7 structured fields on average (min 4, max 8), and each blueprint yields 25 distinct drift-signal signatures. The split is exactly balanced across the five drift families (100 each), with 300 high-risk, 150 medium-risk, and 50 low-risk cases. We also build a 30-case *external transfer slice* covering telecom, banking, travel, e-commerce, and two Tau-Bench-inspired operational domains adapted to the same typed action schema. The 40 direct anchors include backward-compatibility checks for `message_short`/`message_smart`, opening-script repair regressions, asynchronous join timing failures, typed transfer/hangup boundaries, and de-identified empty-message incidents; perturbations preserve the same node types and legality rules while varying configuration deltas, channel splits, parameter families, and workflow timing. We do not run the official public benchmark harnesses; the external slice adapts public task patterns into our own typed action schema.

All four comparison families receive the same semantic proposal bank. This is a deliberate design choice: the benchmark isolates workflow commitment under drift rather than full free-form generation. In other words, we hold the semantic proposal fixed so that differences in performance come from action-layer control, not from giving one family more oracle information or a stronger text generator. Guard labels are not manually annotated: they are programmatically derived from repo-backed legality rules via the hidden oracle. The resulting training problem should therefore be read as *policy compression* from observable features to required-guard labels, not as independent policy discovery from raw logs.

5.2 Families and Metrics

We compare four families:

1. **Static Prompt**: select the top semantic candidate with no drift adaptation.

2. **Dynamic Prompt**: reorder candidates with visible prompt/config or channel hints, but commit directly.

3. **Tool-Augmented**: query visible tool signals and retry or suppress duplicates, but without typed workflow projection.

4. **PGCA**: dynamic ranking + trained guard classification + rule-based safety-prior calibration + typed action projection.

We evaluate guard prediction in two ways. A naive random 5-fold split produces 0.990 micro-F1 and 0.972 exact-match, but this substantially overestimates generalization because near-duplicate variants from the same blueprint appear in both train and test. Our main protocol therefore uses 5-fold *blueprint-grouped* cross-validation, which drops the raw selector to 0.720 micro-F1, 0.704 macro-F1, and 0.396 exact-match. After safety-prior calibration, the deployed selector reaches 0.794 micro-F1, 0.869 macro-F1, and 0.424 exact-match. For the external slice, we train on the full repo-native benchmark and evaluate zero-shot transfer, obtaining 0.917 micro-F1 and 0.963 macro-F1 after calibration. The grouped protocol also quantifies leakage explicitly: blueprint leakage is 1.00 under random folds and 0.00 under grouped folds, inflating micro-F1 by +0.270 and exact-match by +0.576 if left unchecked.

We report compliant action accuracy, policy violation rate, wrong typed action rate, boundary recall, unnecessary boundary rate, plus normalized latency, tool calls, and token cost. Compliant action accuracy and policy violation rate receive 95% Wilson confidence intervals. Pairwise comparisons use exact McNemar tests, and we additionally report paired bootstrap deltas for the main gains. We intentionally avoid collapsing the main benchmark into a single conversation-level judge score: prior work shows that LLM-as-a-judge can be useful, but also systematically biased when the target signal is open-ended or weakly grounded (Zheng et al., 2023).

End-to-end corroboration slice. To complement the action-layer benchmark without replacing it, we add a bounded 50-task end-to-end corroboration slice executed through the native runtime. The slice contains 10 blueprints with 5 variants each and routes commitment through the real `WorkflowEngine` plus the native `MsAgent`, `TransAgent`, and `HangupAgent` executors. We record committed actions, turns, latency, backend model calls, and a payload-token cost proxy; full breakdowns are reported in the appendix.

6 Results

PGCA is the only family that closes policy violations on the repo-native benchmark while retaining high task correctness. It improves compliant action accuracy from 0.60 to 0.95 over the strongest prompt baseline, with Wilson intervals [0.56, 0.64] and [0.93, 0.97] respectively. Exact McNemar tests show significant gains over all baselines for both compliant accuracy and violation reduction ($p < 10^{-10}$ in every repo-native comparison). Against Dynamic Prompt, paired bootstrap gives a +0.35 compliant-accuracy delta (95% CI [0.31, 0.39]) and a -0.35 violation-rate delta (95% CI [-0.39, -0.31]). Equally important, PGCA improves over the dense full-guard baseline from 0.886 to 0.95 compliant accuracy while preserving the same 0.00 violation rate.

The per-drift picture is equally informative and is tabulated in the appendix. Dynamic prompt repair is already enough for prompt/config drift and channel labeling, but it fails badly on workflow-state drift and remains partially unsafe on action-boundary drift. Tool-augmented behavior helps with some tool/schema cases, yet without typed projection it still commits unsafe actions under asynchronous and boundary conditions. PGCA removes the residual violations across all five families. The ablation view explains why: removing the projector collapses PGCA back to Dynamic Prompt, removing the join guard reintroduces workflow-state violations, and using the raw grouped selector leaves 0.10 residual violation. Dense full-guard activation stays safe but is more conservative, losing 6.4 accuracy points relative to calibrated guard selection.

On the 30-case external transfer slice, PGCA reaches 0.933 compliant accuracy and 0.00 violation rate, compared with 0.60/0.33 for Dynamic Prompt and 0.20/0.63 for the Tool-Augmented baseline. The dense full-guard ablation also stays at 0.00 violation, but drops to 0.833 compliant accuracy, showing that the selector is not only a safety mechanism but also a way to avoid unnecessary over-constraint under cross-domain drift. The bounded 50-task end-to-end corroboration slice and the decomposition / stress-test diagnostics are reported in the appendix so that the main text can stay focused on the primary action-layer benchmark.

What the gains mean. The strongest result is not that PGCA always chooses a more conservative action. If that were true, boundary

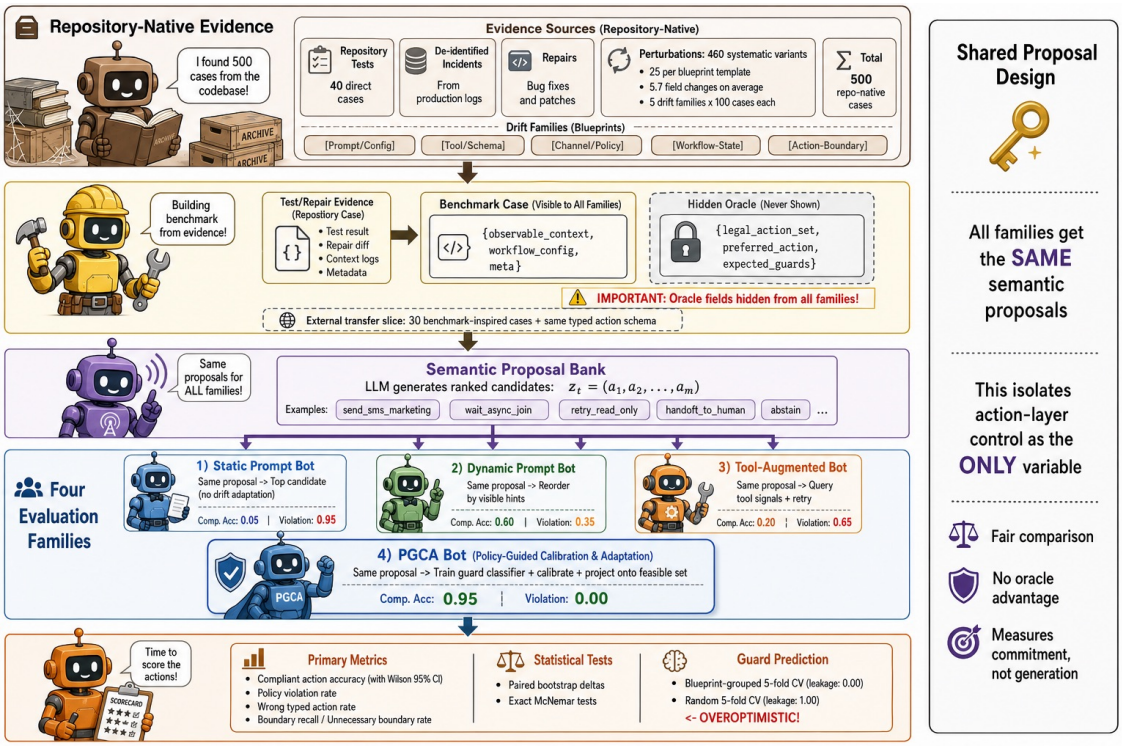

Figure 2: Executable evaluation pipeline. Repository-native evidence is converted into a benchmark with hidden legality oracles, while all families share the same visible semantic proposal bank.

Family	Comp. Acc.	Violation	Wrong Typed	Boundary Recall	Unnec. Boundary	Latency	Tool Calls	Token Cost
Static Prompt	0.05	0.95	0.45	0.75	0.00	1.05	0.00	110.0
Dynamic Prompt	0.60	0.35	0.10	0.50	0.00	1.20	0.00	135.0
Tool-Augmented	0.20	0.65	0.35	0.75	0.00	1.79	1.25	192.5
PGCA	**0.95**	**0.00**	**0.00**	**0.75**	**0.00**	1.40	**0.00**	145.0
PGCA (dense full guard)	0.886	**0.00**	**0.00**	**0.75**	**0.00**	1.43	**0.00**	147.0

Table 2: Overall results on the 500-case repo-native benchmark. Lower is better for violation, wrong typed action, latency, tool calls, and token cost.

Variant	Comp. Acc.	Violation	Wrong Typed	Boundary Recall
PGCA	**0.95**	**0.00**	**0.00**	**0.75**
Raw grouped selector	0.84	0.10	0.00	**0.75**
No projector	0.60	0.35	0.10	0.50
− join guard	0.85	0.10	0.00	0.75
Dense full guard	0.886	**0.00**	**0.00**	**0.75**

Table 3: Ablations on the repo-native benchmark.

recall would rise at the cost of compliant action accuracy. Instead, PGCA improves both. It removes wrong typed actions (0.00 vs. 0.10 for Dynamic Prompt) and keeps normalized latency closer to prompt baselines than to the heavier tool-augmented family.

7 Discussion

The benchmark reveals a sharper story than the earlier artifact-only draft. Prompt repair helps where drift still looks like text or configuration munging, but once drift changes the legal action set—wrong SMS channel, wrong parameter family, missing join, missing audit trace, or untyped transfer boundary—prompt-centric families stop being enough. Tool-centric behavior helps with read-time recovery, yet it

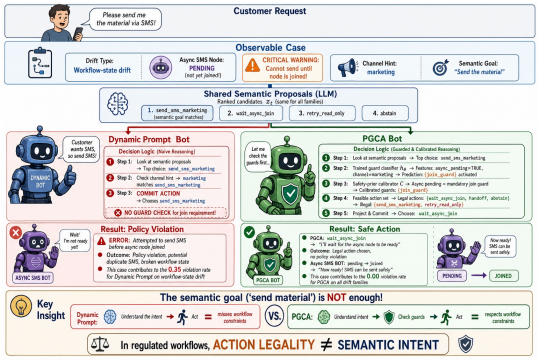

Figure 3: Representative case from the workflow-state drift family. The semantic goal is "send the material," but immediate send is illegal until the asynchronous SMS node is joined.

still lacks a first-class representation of typed action legality. PGCA closes that gap by activating guards only when they are needed and by committing only legal typed actions.

This also clarifies the engineering trade-off. PGCA adds a classifier, a calibrator, and a projector, but the measured overhead remains modest relative to a tool-heavy baseline

while the violation reduction is decisive. The grouped-vs-random CV gap is also methodologically important: repository-derived agent tasks can look solved if blueprint leakage is left unchecked. We report the end-to-end deployment interpretation and the detailed leakage diagnostics in the appendix so the main paper can stay focused on the central action-layer story.

8 Limitations

This paper still has important boundaries. The main benchmark is an action-layer study, not a live end-to-end text-generation benchmark: all families share the same semantic proposal bank so that action commitment can be isolated cleanly. The evidence is also anchored in a single workflow platform, with only 40 direct anchors and 460 systematic perturbations. Moreover, the classifier is trained on oracle-derived required-guard labels and the calibrator is explicitly rule-based, so the contribution is policy compression from bounded workflow evidence rather than end-to-end policy induction from raw supervision. The end-to-end corroboration slice, proposal-noise stress tests, and external transfer slice partially probe those boundaries, but they do not replace a live deployment study on a second native platform or an official run on public benchmark harnesses.

9 Conclusion

For regulated customer-service workflows, business drift is an action-legality problem as much as a semantic-understanding problem. The Policy-Graph Constrained Agent addresses that gap with a trained guard classifier, a rule-based safety-prior calibrator, and a typed action projector. On a 500-case repo-native benchmark and a 30-case external transfer slice, PGCA substantially outperforms prompt-only and tool-only families, and it improves over dense full-guard activation while maintaining zero observed policy violations. A bounded 50-task end-to-end corroboration slice further shows that these gains survive real typed action commitment through the native workflow runtime, while also revealing the remaining transfer/hangup boundary gap that our action-layer benchmark cannot erase. This does not solve general agent reliability, but it does make one claim substantially stronger: high-risk actions in regulated customer-service workflows should remain policy-governed, calibrated, and typed all the way to commitment.

10 Ethics Statement

The repo-native evidence uses de-identified incidents, tests, and repairs. We do not reveal personal information, raw production secrets, or private identifiers. Our action design intentionally prefers typed boundary actions and abstention over unconstrained autonomy in regulated settings.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 32 others. 2022. Constitutional ai: Harmlessness from ai feedback. Anthropic technical report.

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, Rabat, Morocco. Association for Computational Linguistics.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems 36*. Datasets and Benchmarks Track.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, Liuyi Yao, Hongyi Peng, Zeyu Zhang, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. Agentscope: A flexible yet robust multi-agent platform. *Preprint*, arXiv:2402.14034.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. Critic: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*. Poster.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang,

Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Robert Tang, Xiangtao Lu, and 9 others. 2025. Data interpreter: An LLM agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821, Vienna, Austria. Association for Computational Linguistics.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*. Oral.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*. Oral.

Adharsh Kamath, Sishen Zhang, Calvin Xu, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. 2025. Enforcing temporal constraints for llm agents. *Preprint*, arXiv:2512.23738.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905, Bangkok, Thailand. Association for Computational Linguistics.

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

Hanbing Liu, Lang Cao, and Yang Li. 2026. Rag or learning? understanding the limits of llm adaptation under continuous knowledge drift in the real world. *Preprint*, arXiv:2604.05096.

Lewis Liu and Nishant Yadav. 2026. Introducing state-bench: A benchmark for ai agent memory. https://opensource.microsoft.com/blog/2026/05/19/introducing-state-bench-a-benchmark-for-ai-agent-memory/. Official source.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations*. Poster.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. Workshop paper at LLM Agents @ ICLR 2024 (poster).

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36*. Main Conference Track.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*. Poster.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. Published at NeurIPS 2024.

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

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36*. Main Conference Track.

Ling Shi, Yuqin Dai, Ziyin Wang, Ning Gao, Wei Zhang, Chaozheng Wang, Yujie Wang, Wei He, Jinpeng Wang, and Deiyi Xiong. 2026. Sage: A service agent graph-guided evaluation benchmark. *Preprint*, arXiv:2604.09285.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36*. Main Conference Track.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *The Ninth International Conference on Learning Representations*. Poster.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. Workshop paper at LLM Agents @ ICLR 2024 (oral).

Ruixuan Xiao, Wentao Ma, Ke Wang, Yuchuan Wu, Junbo Zhao, Haobo Wang, Fei Huang, and Yongbin Li. 2024. Flowbench: Revisiting and benchmarking workflow-guided planning for llm-based agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10883–10900, Miami, Florida, USA. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35*. Main Conference Track.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2025. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*. Published at ICLR 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36*. Datasets and Benchmarks Track.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2023. Webarena: A realistic web environment for building autonomous agents. Workshop paper at FMDM@NeurIPS 2023.

A Appendix: Benchmark Composition and Extended Results

Coverage and slice interpretation. The 20 repo-native blueprints do not exhaust the full space of post-deployment business drift, but they do cover the five dominant repository-derived families that repeatedly triggered workflow repairs in this platform. Given the balanced 500-case construction, the repo-native split effectively supplies four blueprints per drift family and 25 variants per blueprint. Tables 5 and 6 also show that the main PGCA ranking is not carried only by perturbation-heavy slices: the same ordering holds on direct anchors, perturbations, and the external transfer slice, even though the latter adapts public task patterns into our own typed action schema rather than reusing an official public harness.

B Appendix: Drift-Family Plots and End-to-End Corroboration

Drift and risk diagnostics. Figure 4 makes the largest-sample appendix story visual. The repo-native benchmark contributes 100 cases per drift family, so the plot avoids the "tiny-bucket" look of the smaller corroboration slices. It also sharpens the main claim: prompt repair is already enough for prompt/config drift and often enough for channel naming, but it is not enough for workflow-state drift or typed action-boundary drift. Table 7 complements that view by showing that the highest-risk slice is exactly where PGCA's legality advantage matters most.

End-to-end corroboration slice. The 50-task end-to-end slice is intentionally smaller and stricter about what it claims. Because the slice contains only 10 blueprints with 5 variants each, we report the results above as raw counts plus percentages. PGCA reaches 50/50 planned-action hits and 25/50 committed-action hits, compared with 10/50 and 10/50 for Static Prompt, 30/50 and 25/50 for Dynamic Prompt, and 25/50 and 20/50 for Tool-Augmented. Mean runtime remains low at 0.0039s per task with 2.2 turns, 0.42 backend model calls, and 0.84 payload-token cost units on average. The slice therefore validates that our families can commit real typed ac-

Drift family	Representative change	Main code locus	Failure if prompt-only	Guarded repair
Prompt/config	`prompt2` disappears during partial opening-config conversion	canvas manager + workflow prep	workflow continues only the first opening turn	compatibility guard + slot repair before commitment
Tool/schema	`smartSmsParams` and ordinary SMS params drift apart	SMS agent + template binding	agent decides "send", but payload binds the wrong fields	schema guard + typed parameter projection
Channel/policy	`reach`, `marketing`, and `third_party` diverge	node factory + SMS agent	generic "send SMS" action chooses the wrong channel	channel guard + typed channel action
Workflow-state	asynchronous SMS node is read before join completion	workflow engine + SMS gate	correct intent yields stale or empty downstream action	join guard + wait/handoff projection
Action-boundary	transfer / hangup require scripts and audit fields	transfer / hangup agents	agent infers boundary intent but commits malformed action	script guard + audit guard + typed boundary action

Table 4: Workflow drift taxonomy derived from repository-backed artifacts.

Slice	Cases	Direct anchors	Blueprints	Variants / blueprint	Field diff	Risk counts (L/M/H)
Repo-native	500	40	20	25.0	5.7 (4–8)	50 / 150 / 300
External transfer	30	0	20	1.5	5.6 (5–7)	3 / 8 / 19

Table 5: Composition and coverage of the released benchmark slices. The `Field diff` column reports average structured-field changes from each blueprint anchor, with min/max in parentheses.

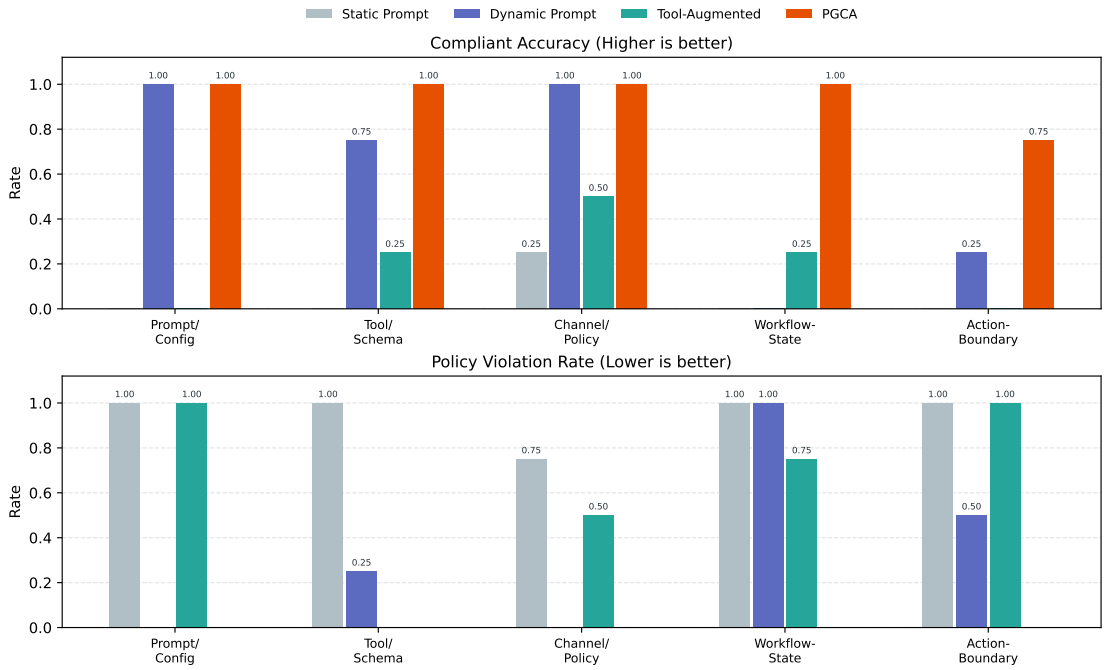

Figure 4: Per-drift results on the 500-case repo-native benchmark (100 cases per drift family). Top: compliant accuracy. Bottom: policy violation rate. Numeric labels show each non-zero bar value.

tions through the native workflow runtime, while also exposing the remaining gap between action-layer legality and end-to-end boundary execution. The strongest corroboration appears on executable SMS actions: PGCA commits the correct action on 15 of 20 SMS tasks, versus 0 of 20 for Static Prompt, 15 of 20 for Dynamic Prompt, and 10 of 20 for Tool-Augmented. Transfer and hangup tasks are still bottlenecked by typed script/boundary behavior in the native executors, where PGCA ties the strongest baseline rather than exceeding it. This is exactly why the 500-case action-layer benchmark remains our primary evidence anchor, and why we present the end-to-end slice as corroboration rather than as a replacement leaderboard.

Decomposition and stress tests. The empirical diagnostics also line up with Proposition 1. For calibrated PGCA, the observed violation rate is 0.00, the missed-required-guard rate is 0.09, and the unsound-accept rate is

Source slice (n)	Static Prompt	Dynamic Prompt	Tool-Augmented	PGCA
Repo-direct (40)	0.05 / 0.95	0.60 / 0.35	0.20 / 0.65	**0.95 / 0.00**
Repo-perturbed (460)	0.05 / 0.95	0.60 / 0.35	0.20 / 0.65	**0.95 / 0.00**
External transfer (30)	0.07 / 0.93	0.60 / 0.33	0.20 / 0.63	**0.93 / 0.00**

Table 6: Source-slice robustness, reported as compliant accuracy / violation rate. The external slice uses the same typed action schema but adapts public task patterns rather than running official benchmark harnesses.

Risk tier (n)	Static Prompt	Dynamic Prompt	Tool-Augmented	PGCA
Low (50)	0.00 / 1.00	1.00 / 0.00	0.00 / 1.00	**1.00 / 0.00**
Medium (150)	0.17 / 0.83	0.67 / 0.33	0.33 / 0.67	**1.00 / 0.00**
High (300)	0.00 / 1.00	0.50 / 0.42	0.17 / 0.58	**0.92 / 0.00**

Table 7: Risk-tier breakdown on the 500-case repo-native benchmark, reported as compliant accuracy / violation rate.

Protocol	Leakage	Micro-F1	Macro-F1	Exact Match
Random 5-fold	1.00	0.990	0.993	0.972
Grouped raw selector	0.00	0.720	0.704	0.396
Grouped calibrated selector	0.00	**0.794**	**0.869**	**0.424**

Table 8: Guard-prediction diagnostics. Random folds leak blueprint variants across train/test and inflate grouped performance by +0.270 micro-F1 and +0.576 exact-match.

Panel A: Policy violation by drift family.

Family	Prompt/Config	Tool/Schema	Channel/Policy	Workflow-State	Action-Boundary
Static Prompt	1.00	1.00	0.75	1.00	1.00
Dynamic Prompt	0.00	0.25	0.00	1.00	0.50
Tool-Augmented	1.00	0.00	0.50	0.75	1.00
PGCA	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

Panel B: End-to-end corroboration raw counts.

Family	Planned (overall)	Committed (overall)	SMS committed	Transfer committed	Hangup committed
Static Prompt	10/50 (20%)	10/50 (20%)	0/20 (0%)	5/15 (33%)	5/15 (33%)
Dynamic Prompt	30/50 (60%)	25/50 (50%)	15/20 (75%)	5/15 (33%)	5/15 (33%)
Tool-Augmented	25/50 (50%)	20/50 (40%)	10/20 (50%)	5/15 (33%)	5/15 (33%)
PGCA	**50/50 (100%)**	**25/50 (50%)**	**15/20 (75%)**	5/15 (33%)	5/15 (33%)

Table 9: Additional appendix raw-result tables. Panel A reports policy violation by drift family on the repo-native benchmark (100 cases per family). Panel B reports the end-to-end corroboration slice as raw counts plus percentages; the slice is intentionally bounded to 10 blueprints × 5 variants, so we show discrete hit counts directly rather than only smoothed decimals.

0.00, yielding an empirical union-bound upper estimate of 0.09. For the raw grouped selector, the corresponding values are 0.10 observed violation, 0.30 missed-required-guard rate, and 0.00 unsound accept rate. In other words, the dominant remaining failure mode is not "projector chooses an illegal action anyway," but "the wrong guards never activate." The bound is therefore directionally useful but loose rather than tight: it identifies where errors enter, but it does not predict the realized violation rate sharply. Proposal-noise stress tests make the same point from a different angle: under medium noise PGCA remains at 0.95 compliant accuracy and 0.00 violation, while under hard noise it drops to 0.60 compliant accuracy but still maintains 0.00 violation. Safety survives degraded proposals; task completion does not, which is exactly the action-layer boundary we intend to claim.

C Appendix: Reproducibility

The artifact bundle contains the executable benchmark generator, the evaluation harness, the repo-native and external-transfer JSONL files, the 50-task end-to-end corroboration suite, the guard selector, the statistical summary, and the per-family traces used to populate the paper.

D Appendix: Additional Deployment Discussion

The end-to-end corroboration slice sharpens the same point from the deployment side. Once we route decisions through the native runtime, the main gain is no longer "PGCA wins every metric everywhere." Instead, the picture is more useful: PGCA preserves the action-layer advantage on executable SMS commitments, matches the strongest prompt baseline overall on committed accuracy, and exposes where transfer/hangup boundary scripts still dominate the final outcome. That is a healthier result than a uniformly inflated end-to-end claim because it shows exactly which part of the stack our method improves and which part still belongs to executor design.