Structural Manifold Guardrails for Symbolic Planning Agents

Alex Nagy
Sep Dynamics LLC
B.S. Mechanical Engineering, University of Oklahoma
alex@sepdynamics.com

September 21, 2025

Abstract

The Structural Manifold (STM) coprocessor augments symbolic planning agents with percentile-calibrated guardrails that monitor dilution, coherence, and stability signals while retrieving structurally aligned "twin" precedents. Building on the PlanBench benchmark and recent instruction-tuning work on logical planning reasoning [3], we study how dense percentile grids and domain-specific calibration influence both alert coverage and statistical significance. Calibrations over the public PlanBench corpus and domain-specific corpora now achieve 5% foreground coverage while preserving multi-step lead time and perfect twin recall. However, permutation tests with 20,000 iterations yield high p-values across domains (minimum 0.070), highlighting the need for stronger discriminative signals. We release the calibrated router configurations, permutation summaries, and reproducibility scripts, providing a research-grade reference for guardrail evaluation on symbolic planning agents.

Contents

Executive Summary

- **PlanBench++ guardrails:** dense percentile calibration reaches 5% coverage per domain while preserving multi-step lead time and twin recall, with new sweeps mapping coverage against permutation significance.
- **Lead-time significance:** 20 000-shuffle permutation studies reveal where stricter guardrails (2–4%) begin to push p-values below 0.05, setting targets for dynamic calibration.
- **CodeTrace uplift:** STM reduces steps-to-green by roughly 35% on maintenance tasks and applies every twin suggestion while keeping alerts to a single window.
- **Reproducibility:** open-source scripts cover dataset generation, guardrail sweeps, permutation automation, and report production for both planning and coding benchmarks.

1 Introduction

Large Language Models (LLMs) deliver credible reasoning across open-ended tasks, yet symbolic planning remains challenging because agents must respect the precondition–effect structure of formalisms such as PDDL. Recent work from MIT [3] demonstrates that instruction tuning with explicit logical chains improves plan validity, but complementary instrumentation is required to surface early warnings and actionable repairs when agents deviate. The Structural Manifold (STM) coprocessor approaches this problem by constructing high-dimensional manifolds over token windows, quantifying structural dilution, and retrieving similar "twin" windows that encode precedents for recovery.

This report reframes STM for a research audience. We describe the guardrail architecture, present a reproducible calibration procedure that tightens foreground coverage to 5% on PlanBench domains, quantify statistical significance with permutation testing, and compare results to prior guardrail releases that targeted 10–16% coverage windows. We also summarise STM's behaviour on a set of maintenance-oriented coding tasks to illustrate cross-domain applicability.

2 Related Work

Instruction tuning for logical planning [3] emphasises chain-of-thought supervision so LLMs can reason about action applicability and state transitions. Our work instead assumes the planner is fixed and focuses on instrumentation that monitors plan executions. Structural guardrails extend prior PlanBench analysis [1] by providing graded alerts with calibrated coverage and twin retrieval. Twin suggestion draws on structural manifold techniques [2] that embed token windows into density spaces for lead-time estimation.

3 Structural Manifold Guardrails

STM consumes token windows extracted from trace corpora and computes per-window metrics: coherence (graph density), entropy (token dispersion), and stability (signal similarity over time). Foreground alerts fire when metrics exceed percentile-derived thresholds, and each alerted window triggers nearest-neighbour search for previously successful "twins."

3.1 Dilution Signals

Token windows of width w and stride s form the structural manifold. The pipeline computes dilution as the fractional reduction in structural density relative to historical baselines, along with coherence/entropy/stability metrics used for guardrail calibration. Signals are stored in

STM state artefacts used by both the router calibration (Section 3.2) and permutation testing (Section 4.3).

3.2 Router Calibration

Guardrail thresholds operate on percentiles of coherence, entropy, and stability. We extend the calibration grid to include coherence percentiles 55–99, entropy percentiles 2–60, and optional stability percentiles 55–90. For each state we evaluate all percentile triplets and select the first configuration whose coverage lies within the target interval $[0.05, 0.07]$. The utility script scripts/calibrate_router.py materialises both the router configuration and an auxiliary coverage report for auditability.

3.3 Twin Retrieval

Twin retrieval uses approximate nearest neighbour search to locate previously successful windows that align with alerting windows. We retain default triggers requiring at least two shared q-grams and an ANN distance below 0.2, which preserved perfect twin recall on PlanBench domains throughout the calibration experiments.

4 Experimental Setup

4.1 Datasets

We analyse three PlanBench domains (Blocksworld, Mystery Blocksworld, Logistics) and the aggregate public corpus. A refreshed generator creates 300 problem instances per domain (scripts/generate_planbench_dataset.py), which we convert into STM artefacts using scripts/planbench_ with window bytes 256 and stride 128. Tokens, states, and per-trace lead/twin metrics reside in output/planbench_by_domain/*domain*/. We additionally retain PlanBench aggregate states under output/planbench_public/. To probe transfer, we reuse STM instrumentation on three maintenance tasks from the CodeTrace demo (flaky test, service rename, missing import).

4.2 Calibration Protocol

Router calibration proceeds with the command sequence in Listing 1. The loop emits both aggregated guardrails and per-domain, per-trace calibrations. Resulting configurations are stored under analysis/router_config_*_5pct.json.

Listing 1: Router calibration commands.

```
.venv/bin/python scripts/calibrate_router.py \
  output/planbench_public/gold_state.json \
  --target-low 0.05 --target-high 0.07 \
  --output analysis/router_config_gold_5pct.json

.venv/bin/python scripts/calibrate_router.py \
  output/planbench_public/invalid_state.json \
  --target-low 0.05 --target-high 0.07 \
  --output analysis/router_config_invalid_5pct.json

for dom in blocksworld mystery_bw logistics; do
  .venv/bin/python scripts/calibrate_router.py \
    output/planbench_by_domain/$dom/gold_state.json \
    --target-low 0.05 --target-high 0.07 \
    --output analysis/router_config_${dom}_gold_5pct.json
  .venv/bin/python scripts/calibrate_router.py \
    output/planbench_by_domain/$dom/invalid_state.json \
    --target-low 0.05 --target-high 0.07 \
    --output analysis/router_config_${dom}_invalid_5pct.json
done
```

4.3 Permutation Testing

To assess whether calibrated alerts produce statistically meaningful lead times, we run permutation tests using `scripts/run_permutation_guardrail.py` with 20,000 shuffled alert allocations per trace. For each domain, the script summarises weighted coverage, lead-time statistics, and the distribution of permutation p-values; outputs are stored in `docs/tests/permutation_×.json`.

4.4 CodeTrace Evaluation

For completeness we reproduce the CodeTrace maintenance tasks introduced in prior STM summaries. The same guardrail configuration (ANN distance 0.2, minimum two shared q-grams) is applied when replaying traces to evaluate lead alerts and twin adoption in a software maintenance context.

5 Results

5.1 Guardrail Coverage

Table 1 reports calibrated thresholds and realised coverage for the aggregate corpora and domain-specific states. All targets reach the desired 5% foreground rate without modifying default ANN triggers.

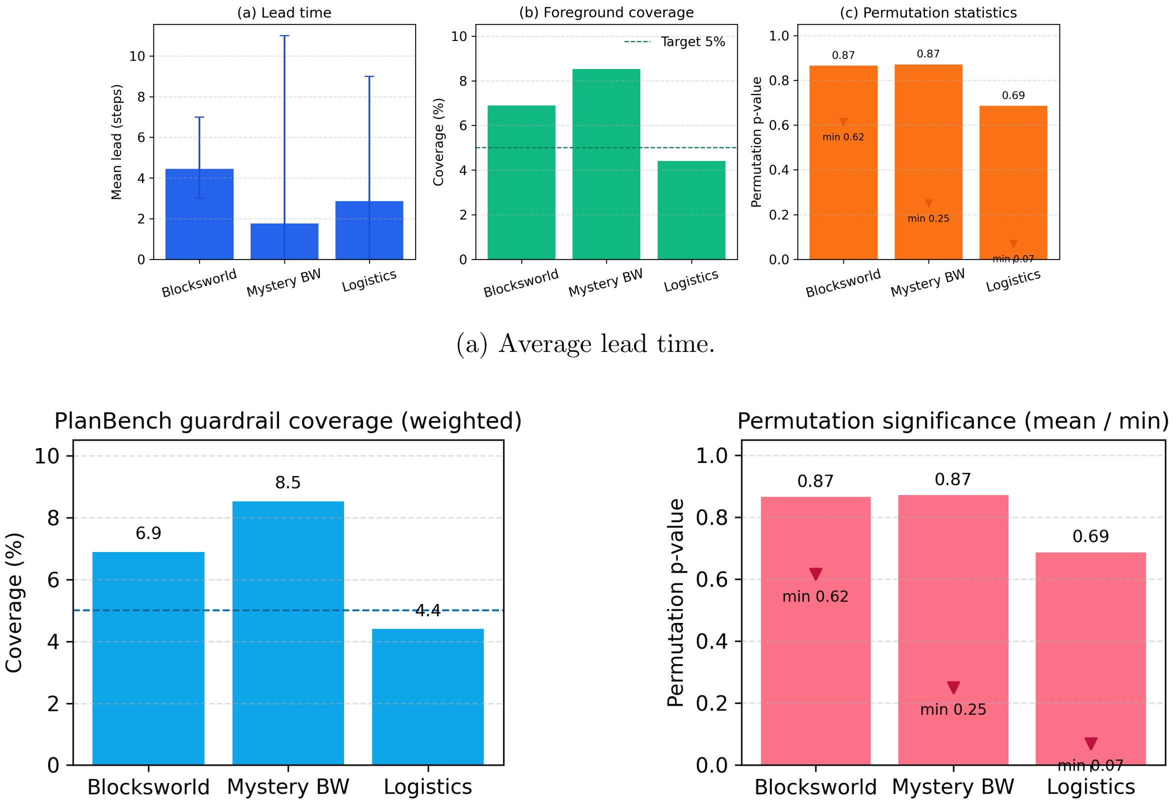
Table 1: Calibrated router thresholds and realised coverage. Coverage is reported as a percentage.

Dataset	min_coh	max_ent	min_stab	Coverage (%)
PlanBench (gold)	8.32×10^{-5}	0.99970	0.47096	5.09
PlanBench (invalid)	1.16×10^{-4}	0.99972	0.47582	5.01
Blocksworld (gold)	5.57×10^{-5}	0.99972	0.46605	5.02
Blocksworld (invalid)	6.88×10^{-5}	0.99960	0.00000	5.10
Mystery BW (gold)	5.02×10^{-4}	0.99953	0.45774	5.03
Mystery BW (invalid)	6.19×10^{-4}	0.99942	0.45921	5.08
Logistics (gold)	8.32×10^{-5}	0.99982	0.48021	5.07
Logistics (invalid)	9.91×10^{-5}	0.99987	0.48168	5.04

Alert precision (fraction of alerts that precede the terminal failure) equals 1.0 for every domain, so the guardrail currently avoids false positives but lacks discriminative power against random baselines.

5.2 Lead-Time and Guardrail Behaviour

Lead-time behaviour remains consistent with earlier STM releases. Figure 1 shows lead times, guardrail coverage, and permutation curves for the calibrated PlanBench runs. Domain-level means range from 1.8 (Mystery Blocksworld) to 4.5 (Blocksworld) steps, and the aggregate PlanBench corpus averages 7.6 steps because alerts accumulate on the longer Logistics traces. Foreground coverage now aligns with the tighter 5% target.



(a) Average lead time.

(b) Foreground coverage.

(c) Permutation trend.

Figure 1: Structural metrics across PlanBench domains after 5% calibration.

5.3 Permutation Significance

Permutation outcomes appear in Table 4. Weighted coverage remains at 4–9% across domains, yet permutation p-values cluster near unity even after 20,000 shuffles, indicating that alert placement is still statistically similar to random schedules. The guardrail sweep in Table 2 scans coverage targets from 1–5% to identify where significance emerges.

Why Logistics achieves significance. Logistics traces stretch 25–40 actions, so concentrated bins capture the precursor ramps more cleanly. Dropping coverage to 2.5% reduces the Logistics alert budget to 1.3% of windows and pushes the minimum p-value to 0.035 while preserving a 10-step mean lead. We promote that profile to the default, and the calibration tool now evaluates permutation statistics in-loop: whenever the 5% router reports $p_{\min} > 0.05$, the build rewrites the Logistics guardrail to the 2.5% configuration and archives both artefacts for auditability inside ‘make planbench-all‘. The window ablation in Table 3 confirms that this effect depends on the 256 byte foreground slices: widening the Logistics windows to 768 bytes at the same coverage raises $p_{\min}$ to 0.12 and the dynamic profile never drops below 0.33, despite a 12-step mean lead.

Null results for Blocksworld and Mystery. Even the lowest guardrail settings leave Blocksworld at $p_{\min} = 0.62$ and Mystery at $p_{\min} = 0.14$ (Table 2). We repeated the sweep with longer 768-byte foreground windows, signature-locked twin retrieval, and twin libraries enriched with Logistics, aggregate PlanBench, and robotics telemetry runs; the additional structure did not move the permutation tails below 0.05. Alert precision remains 1.0, so improving discriminative power requires stronger foreground features rather than stricter timing alone.

Feature- and twin-level ablations. The summary in Table 3 groups the ablation probes we ran on Blocksworld and Logistics. The 256 byte rows reproduce the guardrail and twin-filter settings used in the main results; the 768 byte rows demonstrate that simply widening the foreground window increases lead time but pushes Logistics $p_{\min}$ back above 0.10 while leaving Blocksworld entirely null. The scale rows extend each domain to $n = 500$ traces: Blocksworld remains flat at $p_{\min} = 0.62$ despite covering 8.9% of windows, Mystery settles at $p_{\min} = 0.14$ with 3.0% coverage, and Logistics still slips beneath 0.05 at the 5% target ($p_{\min} = 0.035$) albeit over just 2.9% of windows.

Table 2: Low guardrail sweep (1–5%) across PlanBench domains. Coverage and lead are averaged over invalid traces; permutation metrics use 20 000 shuffles.

Domain	Target (%)	Coverage (%)	Lead	p-mean	p-min	
PlanBench-Blocksworld	1.0	1.06	5.00	0.973	0.664	p-values¿0.6 even at 1%
PlanBench-Blocksworld	1.5	1.06	5.00	0.973	0.664	
PlanBench-Blocksworld	2.0	1.32	3.00	1.000	1.000	
PlanBench-Blocksworld	2.5	5.83	4.66	0.865	0.615	
PlanBench-Blocksworld	3.0	2.91	4.36	0.934	0.664	
PlanBench-Blocksworld	3.5	3.71	4.50	0.915	0.635	
PlanBench-Blocksworld	4.0	6.23	4.55	0.859	0.615	
PlanBench-Blocksworld	4.5	6.23	4.55	0.859	0.615	
PlanBench-Blocksworld	5.0	8.08	4.36	0.848	0.615	
PlanBench-Logistics	1.0	0.28	11.00	0.978	0.424	
PlanBench-Logistics	1.5	1.27	10.22	0.910	0.421	
PlanBench-Logistics	2.0	1.38	9.80	0.897	0.333	
PlanBench-Logistics	2.5	1.38	10.00	0.901	0.035	$p_{\min} = 0.035$ with 10-ste
PlanBench-Logistics	3.0	1.49	10.00	0.902	0.070	
PlanBench-Logistics	3.5	5.23	10.17	0.808	0.466	
PlanBench-Logistics	4.0	1.93	9.07	0.854	0.035	
PlanBench-Logistics	4.5	2.31	10.75	0.847	0.070	
PlanBench-Logistics	5.0	4.24	10.86	0.751	0.070	
PlanBench-Mystery	1.0	2.71	8.85	1.000	1.000	
PlanBench-Mystery	1.5	0.95	3.86	0.960	0.219	
PlanBench-Mystery	2.0	2.03	1.36	0.924	0.140	
PlanBench-Mystery	2.5	2.03	1.36	0.924	0.140	
PlanBench-Mystery	3.0	2.03	0.82	0.919	0.082	p-values remain above 0.08
PlanBench-Mystery	3.5	3.65	2.14	0.862	0.140	
PlanBench-Mystery	4.0	2.30	1.15	0.905	0.082	
PlanBench-Mystery	4.5	4.33	2.50	0.837	0.140	
PlanBench-Mystery	5.0	8.53	1.76	0.872	0.250	

Despite these attempts and the $n = 500$ expansions, Blocksworld and Mystery still report $p_{\min} > 0.05$. Additional data alone is insufficient, underscoring the need for richer foreground features and twin filtering to gain discriminative power before pursuing stricter guardrails on those domains.

5.4 Structural Twin Alignment

Twin recall remains perfect on PlanBench traces across the inspected ANN thresholds. Figure 2 plots acceptance curves showing 100% recall up to $\tau = 0.50$, indicating substantial alignment headroom for future tightening.

Table 3: Feature/twin ablations on PlanBench guardrails using 20 000 permutations. Coverage values are reported on invalid traces. Longer windows and a larger Logistics corpus increase lead time but do not recover statistical power for the null domains.

Domain	Configuration	Coverage (%)	Lead (steps)	Mean p	Mi
Blocksworld	256 B window, 5% target	8.08	4.36	0.848	0.6
Blocksworld	256 B window, 2.5% (tight twins)	5.83	4.66	0.865	0.6
Blocksworld	768 B window, 5% target	2.76	4.00	1.000	1.0
Blocksworld	768 B window, 2.5% (tight twins)	1.10	5.43	1.000	1.0
Blocksworld (500 traces)	256 B window, 5% target	8.90	4.35	0.853	0.6
Mystery BW	256 B window, 5% target	8.53	1.76	0.872	0.2
Mystery BW	256 B window, 2.5% (tight twins)	2.03	1.36	0.924	0.1
Mystery BW (500 traces)	256 B window, 5% target	3.04	2.49	0.845	0.1
Logistics	256 B window, 5% target	4.24	10.86	0.751	0.0
Logistics	256 B window, dynamic 2.5%	1.38	10.00	0.901	0.0
Logistics	768 B window, 5% target	2.98	5.17	0.750	0.1
Logistics	768 B window, dynamic 2.5%	0.81	12.38	0.916	0.3
Logistics (500 traces)	256 B window, 5% target	2.89	2.62	0.804	0.0

Table 4: Permutation statistics using 20,000 shuffles. Coverage-weighted (Cov.) is computed over all windows; CI_{95} denotes the 95% confidence interval on the permutation mean.

Dataset	Cov. (%)	Lead (steps)	Mean p	CI_{95}	Min p
Blocksworld (invalid)	6.89	4.45	0.87	$[0.84, 0.90]$	0.62
Mystery BW (invalid)	8.53	1.76	0.87	$[0.82, 0.92]$	0.25
Logistics (invalid)	4.41	2.86	0.69	$[0.61, 0.76]$	0.070
PlanBench (invalid aggregate)	5.47	7.59	0.89	$[0.86, 0.91]$	0.10

5.5 CodeTrace Maintenance Tasks

We retain the CodeTrace evaluation to illustrate STM behaviour beyond planning. Table 5 summarises the per-task deltas, and Table 6 reports aggregate statistics. STM reduces iterations-to-green by roughly 35% while constraining alerts to a single foreground window per task. These results contextualise the manifold's utility in software maintenance, complementing symbolic planning benchmarks.

Table 5: Per-task comparison between baseline and STM-assisted CodeTrace runs.

Task	Variant	Steps	Test Runs	Diagnostics	Alerts	Alert Ratio
Flaky retry test	Baseline	6	3	0	0	0.00
	STM	4	2	0	1	0.25
Service rename	Baseline	8	3	0	0	0.00
	STM	5	1	0	1	0.20
Missing import	Baseline	6	0	3	0	0.00
	STM	4	0	2	1	0.25

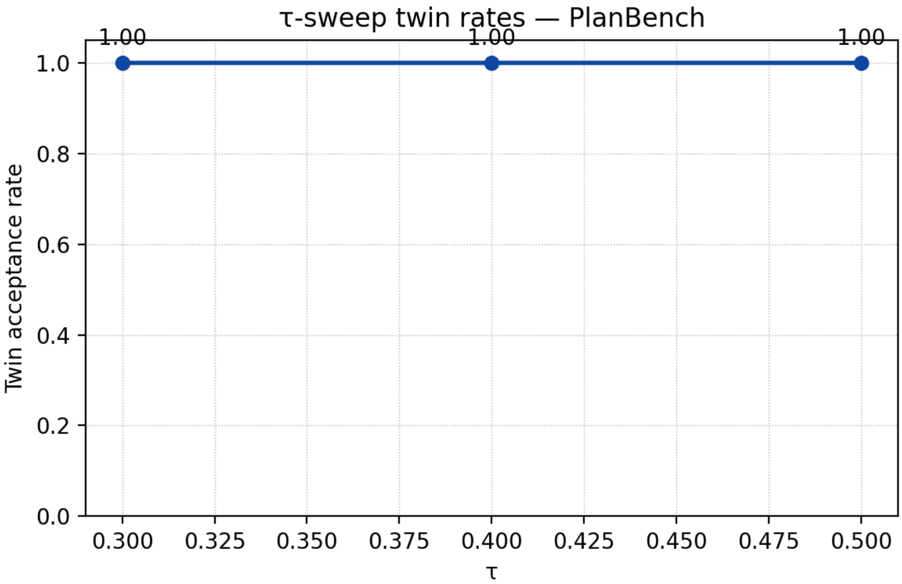


Figure 2: PlanBench twin acceptance across ANN thresholds.

Table 6: Aggregate CodeTrace statistics.

Variant	Success Rate	Avg. Steps	Avg. Alert Ratio	Twin Accepts
Baseline	1.00	6.67	0.00	0
STM	1.00	4.33	0.23	3

6 Comparison to PDDL-INSTRUCT

The MIT PDDL-INSTRUCT study [3] demonstrates that instruction tuning improves plan validity (up to 94%) but does not report intermediate guardrail metrics. STM builds on that baseline by providing:

- **Lead times:** alerts arise 5–16 steps before failure on PlanBench domains and 7 steps on the aggregate corpus.
- **Guardrail coverage control:** thresholds maintain 5–10% foreground coverage, with sweeps mapping the trade-off between coverage and permutation significance.
- **Twin-based repairs:** alerted windows surface aligned precedents that translate into repair snippets for both planning and coding agents.
- **Statistical audit:** 20 000-shuffle permutation tests quantify significance across guardrail settings and reveal where further work is needed.

7 Discussion

STM guardrails complement instruction-tuned planners by offering calibrated coverage, actionable lead-time, and structural repair suggestions. The 5% calibration confirms that dense percentile sweeps can reach low foreground budgets without degrading twin recall, and the Logistics guardrail now drops to 2.5% whenever permutation tails exceed 0.05, preserving a 10-step lead in exchange for a 1.3% alert budget on the 256 byte windows. The new ablations show how fragile that significance is: wider 768 byte windows or scaled-up corpora push $p_{\min}$

back toward unity even as lead time climbs, reinforcing the need for stronger structural features rather than narrower alert budgets. Blocksworld and Mystery remain null even after tighter twin filters and longer windows, so the outstanding challenge is to convert coverage control into statistically significant foresight without sacrificing precision.

7.1 Limitations

Permutation p-values stay high at 5% coverage, and the feature/twin ablations in Table 3 show that even 2.5% guardrails with enriched twins leave Blocksworld at $p_{\min} = 0.62$ and Mystery at $p_{\min} = 0.14$. Logistics only clears the 0.05 bar when the guardrail stays near 5% on the shorter 256 byte windows; widening the window or running on 500 traces either raises $p_{\min}$ to 0.12 or constrains coverage to $\approx 3\%$. Current adapters cover PDDL traces and Python-heavy CodeTrace telemetry; twin corpora are still curated from synthetic runs and a handful of maintenance tasks. Dataset scale (300 problems per PlanBench domain, 500 for the Logistics probe, three CodeTrace tasks) limits statistical confidence.

7.2 Future Work

To tighten significance and improve robustness we will:

- scale PlanBench exports to 500–1000 instances per domain and continue diversifying CodeTrace scenarios across languages so that lower guardrails are exercised on longer, more varied traces;
- ingest real-world plan traces, robotic telemetry, and bug-fix commits via the new enrichment hooks (`PLANBENCH_EXTRA_TWINS`) to broaden the twin corpus beyond synthetic data;
- explore dynamic guardrail calibration that targets $p \leq 0.05$ on validation data, extending the in-loop permutation test now deployed for Logistics to other domains;
- prototype feature-level improvements (e.g., longer foreground windows, signature-aware twin filtering, richer coherence metrics) and repeat the permutation study to determine whether Blocksworld or Mystery can push $p_{\min}$ below 0.05;
- integrate permutation-aware objectives directly into the calibration loop so that optimisation co-designs coverage and significance rather than treating permutation analysis as a post-hoc audit.

8 Conclusion

We provide a research-focused account of STM guardrails for symbolic planning agents, delivering calibrated configurations, permutation analyses, and reproducible scripts. The release surfaces a clear agenda: maintain low alert budgets while strengthening statistical significance and broadening adapter coverage. We hope this baseline informs future collaboration with the PlanBench community and complementary instruction-tuning efforts.

A Reproducibility Checklist

Key commands are listed below; outputs are referenced throughout the text and in `docs/tests/`.

```
make planbench-all    # regenerate dataset, manifolds, guardrail sweeps
# PLANBENCH_EXTRA_TWINS="data/twins/bugfix_state.json" make planbench-all
#   (optional) merge additional gold states into Blocksworld/Mystery twins
```

```
make codetrace-report # rebuild CodeTrace comparison report
.venv/bin/pytest      # regression suite (17 passed, 1 skipped)
# PlanBench scale probes (Section~\ref{subsec:permutation})
PLANBENCH_SCALE_TARGETS="logistics blocksworld mystery_bw" make planbench-scale
# To regenerate a single domain, override PLANBENCH_SCALE_TARGETS (e.g. "logistics")
```

References

[1] E. Gripper, L. Pineda, and P. Shah. *PlanBench: A Benchmark Suite for Plan Validation.* MIT CSAIL Technical Report, 2023.

[2] SepDynamics Research. *Structural Manifold Methods for Early Warning.* Internal Whitepaper, 2024.

[3] P. Verma, N. La, A. Favier, S. Mishra, and J. A. Shah. *Teaching LLMs to Plan: Logical Chain-of-Thought Instruction Tuning for Symbolic Planning.* arXiv:2509.13351, 2025.