Automata from Agent Traces: Failure and Next-Step Prediction

Seonglae Cho [1] **Franklin Cardenoso Fernandez** [1 2] **Umar Mohammed** [1] **Zekun Wu** [3] **Kleyton Da Costa** [3] **Ilham Wicaksono** [1] **Adriano Koshiyama** [3]

Abstract

LLM-based agents execute multi-step tasks, but their behavioral structure remains opaque: long unstructured traces resist the safety auditing and runtime monitoring that deployment requires. Existing approaches operate per-trace or success-only, so they miss the cross-run topology that links *next-step* and *failure* prediction. To recover that shared structure, we collapse an entire trace corpus into a single, compact finite-state machine (FSM) that serves as a structural substrate for the otherwise unpredictable behavior of LLM agents. Across twelve public datasets, the FSMs are compact (7–43 states), replay held-out data at $\geq$0.997 fitness with zero structural variance across splits, and build in milliseconds. This substrate addresses both prediction goals. For *next-step prediction*, FSM-state context outperforms Agent Workflow Memory on every ground-truth-matched dataset. For *failure prediction*, per-state behavioral features reach held-out AUROC up to 0.94, and an online monitor ranks failing runs above passing ones from a partial trace, triggering early stopping well before completion. Behavioral topology thus appears shaped more by the deployment harness than by the LLM, providing a model-agnostic structural primitive for safety auditing and runtime monitoring.

1. Introduction

As LLM-based agents (Wang et al., 2024a; Sumers et al., 2024) take on longer reasoning chains and broader action spaces, the risk of undetected failures scales with their autonomy. These agents now resolve GitHub issues (Yang et al., 2024; 2025a), navigate websites (Deng et al., 2023; Zhou et al., 2024; Koh et al., 2024), operate desktop environments (Xie et al., 2024; Wang et al., 2025b), manage customer service interactions (Yao et al., 2024), and orchestrate multi-agent pipelines (Wu et al., 2023; Hong et al., 2024). Following the ReAct paradigm (Yao et al., 2023), they interleave chain-of-thought reasoning (Wei et al., 2022) with tool calls (Schick et al., 2023), generating execution traces whose behavioral structure remains implicit. A coding agent cycles through `search`→`edit`→`execute`; a customer service agent alternates between database queries and user communication. This structure emerges from the interaction between the system prompt, available tools, and task distribution, but it is nowhere specified.

Understanding this latent structure matters for safety auditing (Zhang et al., 2025a; Ruan et al., 2024; Chen et al., 2025a), debugging bottleneck states (Yang et al., 2025b; Cemri et al., 2025), and monitoring behavioral drift in production (Wang et al., 2026). Yet current approaches operate at the individual trace level, requiring task descriptions, manual specification, or success filters (Chen et al., 2025b; Wu et al., 2024; Wang et al., 2024b).

We frame behavioral recovery as an inverse problem: given a corpus of execution traces, reconstruct a finite state machine (FSM) that explains the observed behavior. Agent traces provide only positive examples in the Gold sense (Gold, 1967; Angluin, 1980), and identifying the target language from positive examples alone is impossible in the limit. Our key observation is that agent behavior is generated by a bounded set of tools and actions, producing traces with small activity alphabets (6–42 symbols). The resulting behavioral topology appears shaped more by the *system* than by the LLM, across 4 chat models on tau2-bench: a single FSM achieves perfect fitness on every model. This structural constraint makes the problem tractable: a prefix tree merged by last activity produces a compact directly-follows FSM in linear time, requiring no learning hyperparameters (the only design choice is the activity extraction function, whose robustness we verify in Appendix G.2). We evaluate on twelve public datasets (Table 5) against nine baselines from automata learning (RPNI, EDSM, Alergia, k-Tails), HMMs, process mining, and workflow extraction (§2):

[1]Holistic AI [2]PUC-Rio [3]University College London. Correspondence to: Seonglae Cho <sungle3737@gmail.com>.

Published at the Second Workshop on Agents in the Wild: Safety, Security, and Beyond (AIWILD) at ICML 2026. Copyright 2026 by the author(s).

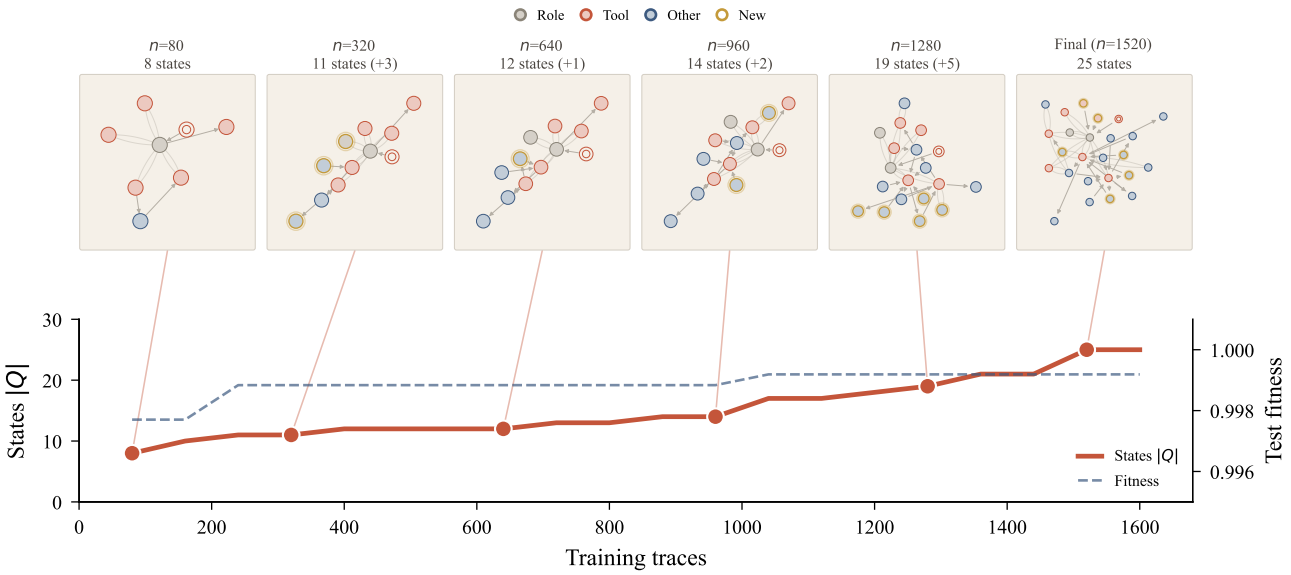

Figure 1. **FSM evolution** on SWE-agent. State count $|Q|$ (red, left) and test fitness (blue, right) over training traces, with FSM snapshots at six milestones. The state space grows incrementally as new behavioral modes appear, while fitness saturates early (≥ 0.99 at 240 traces, 15% of training); construction completes in milliseconds.

- **Workflow memory.** FSM-state context outperforms Agent Workflow Memory (Wang et al., 2024b) on 8/8 datasets (6 statsig at $p < 10^{-8}$; Table 4).
- **Next-step prediction.** FSM state conditioning improves cross-entropy by 0.155 bits (21%) over identical methods without state.
- **Failure prediction.** Per-state features reach held-out AUROC up to 0.94, lift MLP/GRU/Transformer baselines on 20 of 21 pairs, and power a prefix-based monitor that ranks failing SWE-agent runs above passing ones at the 25% checkpoint (rank-AUROC 0.66 vs. 0.5 for flag-everything) and triggers early stopping at 32% completion.
- **Compression.** 15–3,036× fewer states than RPNI at ≥0.997 fitness from a deterministic, hyperparameter-free construction.

One object ties these results together: bounded LLM-agent alphabets make the resulting compact deterministic finite automaton (DFA) both small and statistically informative, and the same FSM unifies workflow memory, next-step prediction, failure prediction, and runtime monitoring (Theorems and Propositions in §3.4).

2. Related Work

Agent safety and monitoring. AgentSpec (Wang et al., 2026) and ShieldAgent (Chen et al., 2025a) enforce safety policies; AgentMonitor (Chan et al., 2024) predicts task performance from step-level features using flat XGBoost models. ProbGuard (Wang et al., 2025a) learns a DTMC from traces and applies bounded-horizon PCTL reachability for runtime safety filtering; head-to-head on our datasets (Appendix G.6) it trails our FSM features by mean +0.176 AUROC because, without hand-crafted unsafe predicates, its symbolic-state abstraction degrades to per-activity granularity. Concurrent trajectory-anomaly detectors (Liu et al., 2025; Deshpande et al., 2025; He et al., 2025) target the same problem with hierarchical, behavioral, or graph-based pipelines; our FSM differs by providing a compact structural quotient that doubles as workflow memory and next-step predictor, not solely an anomaly score. Closest is the concurrent PrefixGuard (Huang et al., 2026), which also extracts a DFA from LLM-agent traces for online failure-warning monitors; we treat the same compact automaton as one substrate that additionally drives compression, next-step prediction, and workflow memory, rather than a monitor-only construction. Cemri et al. (2025) taxonomize multi-agent failure modes from 1,600+ traces, motivating automated detection. These approaches either require hand-crafted policies or lack structural behavioral models. Our FSM provides a learned structural model that enables compositional queries and early failure prediction from partial traces.

Behavioral abstractions for agents. Agent Workflow Memory (Wang et al., 2024b) extracts linear workflow patterns from successful traces, while Reflexion (Shinn et al., 2023) and ETO (Song et al., 2024) learn from failures via verbal reflection or contrastive pairs. None of these produce structural models with state abstraction. On the FSM side, AFlow (Zhang et al., 2025b) searches workflows via MCTS, MetaAgent (Chen et al., 2025b) builds FSMs top-down from

task descriptions, and StateFlow (Wu et al., 2024) relies on manual specification. Our method recovers FSMs bottom-up from raw traces with a compact structural quotient and per-state decomposition for failure prediction.

Process mining. Process discovery (van der Aalst, 2016) recovers Petri nets from event logs. Applied to agent traces, standard miners produce "flower models" with precision 0.00–0.80 (Table 20), with highest precision on constrained workflows (Berti et al., 2024a;b).

Grammatical inference. Learning finite automata from positive examples is impossible in the limit (Gold, 1967; Angluin, 1980). RPNI (Oncina & García, 1992), EDSM (Lang et al., 1998), and L* (Angluin, 1987) require negative examples or oracles unavailable in trace analysis. k-Tails (Biermann & Feldman, 1972) merges states with identical k-length futures, but requires a hyperparameter and produces 1.4–10× more states than ours with lower fitness. Among positive-only methods, Alergia (Carrasco & Oncina, 1994) is the strongest competitor: it matches our fitness with 1.0–6.0× more states via statistical tests. HMMs (Rabiner, 1989) match state counts but yield non-interpretable latent states. Our approach exploits bounded activity alphabets (6–42 symbols) to produce compact, interpretable FSMs (7–43 states) without hyperparameters.

3. Method

3.1. Problem Formulation

An agent execution trace is a sequence of messages $\tau = (m_1, m_2, \ldots, m_T)$, where each message m_t has a role (system, user, assistant, tool) and content. An activity extraction function $\phi : m_t \mapsto a_t \in \mathcal{A}$ maps each message to a symbol from a finite alphabet $\mathcal{A}$. The activity sequence is $\sigma(\tau) = (\phi(m_1), \ldots, \phi(m_T))$.

Given a corpus $\mathcal{D} = \{\tau_1, \ldots, \tau_N\}$, we construct a finite state machine $\mathcal{M} = (Q, \mathcal{A}, \delta, q_0, Q)$ with states Q, partial transition function $\delta : Q \times \mathcal{A} \to Q$, and initial state q_0. All states are accepting. The transition function is deterministic: each (state, activity) pair maps to at most one successor.

Definition 1 (Replay fitness)**.** For sequence $\sigma = (a_1, \ldots, a_T)$, let k be the number of symbols consumed by $\mathcal{M}$ when replaying σ from q_0 (steps where $\delta(q, a_t)$ is defined). The replay fitness is $\mathrm{fit}(\sigma, \mathcal{M}) = k/T$. Corpus fitness is

$$\mathrm{Fit}(\mathcal{D}, \mathcal{M}) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \mathrm{fit}(\sigma(\tau), \mathcal{M}). \tag{1}$$

3.2. Activity Extraction

Agent traces come in heterogeneous formats. We apply three extraction rules in priority: (1) tool calls: if a message contains a `tool_call` field, the activity is the function name; (2) action tags: if the content contains `[ACTION] description`, the activity is the action label; (3) command extraction: for agents using code blocks, we extract the first command token and map it to a semantic category. If no rule matches, the activity defaults to `role:content_type` (e.g., `assistant:text`). The extraction is deterministic and format-specific; details for each dataset are in Appendix B.1. The downstream pipeline is robust to this choice: across four extraction granularities (role-only $|\mathcal{A}|$=2–4, role-type, role-action, tool-only) failure-prediction AUROC moves <0.03 and replay fitness stays ≥ 0.999 on every dataset (Appendix G.2), so the rules above are one valid setting rather than the only one.

3.3. FSM Construction

Given activity sequences $\{\sigma(\tau_i)\}_{i=1}^{N}$, construction proceeds in two steps (Algorithm A.1, Appendix A.1).

Step 1: Prefix tree. Insert all activity sequences into a trie. Each unique prefix is a distinct state. The prefix tree has perfect training fitness but $O(\sum_i T_i)$ states.

Step 2: Merge by last activity. Merge all trie states reached by the same activity into one. Writing $\kappa(q)$ for the activity on the edge into q (and $\kappa(q_\varepsilon) = \mathrm{init}$ for the root), states are merged by the last-activity right congruence

$$q \sim q' \iff \kappa(q) = \kappa(q'), \tag{2}$$

which has $|\mathcal{A}| + 1$ classes. Summing the trie's traversal counts over each class pair gives the directly-follows automaton in a single pass; cycles arise wherever an activity recurs.

Section 3.4 proves this construction preserves fitness and yields a compact directly-follows automaton; tool-use patterns (search–edit–execute) collapse to loops and the state count tracks the number of distinct activities. Figure 5 (Appendix A.1) illustrates the construction at role-level granularity for a customer service agent (6 states, 5 activities).

3.4. Construction and Convergence Guarantees

We characterize the correctness and optimality of the extracted FSM.

Theorem 2 (Fitness preservation)**.** *Structural merging preserves training fitness: if trace σ is accepted by the prefix tree, it is accepted by the merged FSM.*

Proof sketch. Merging only adds out-edges: each class carries the union of its members' transitions, so every trie edge survives in the quotient. For any trace $\sigma = (a_1, \ldots, a_T)$ accepted by the prefix tree with state sequence

$q_0, q_1, \ldots, q_T$, the merged FSM follows the quotient sequence $[q_0], [q_1], \ldots, [q_T]$, since $\delta(q_i, a_{i+1}) = q_{i+1}$ implies $\delta([q_i], a_{i+1}) = [q_{i+1}]$ by the congruence (2); the full trace is accepted. (Full proof in Appendix A.2.) □

Theorem 3 (Determinism and compactness)**.** *Merging the prefix tree by the last-activity right congruence yields a deterministic FSM with $|Q| = |\mathcal{A}| + 1$ states—one per activity plus the initial state—that accepts every training trace. The construction is a deterministic function of the corpus, so re-extraction from the same data is exact.*

Proof sketch. Each trie state is assigned to the class of its incoming activity, giving $|\mathcal{A}| + 1$ classes; transitions are the directly-follows pairs, deduplicated, so each (q, a) has at most one target and the FSM is deterministic. Acceptance of every training trace follows as in Theorem 2. The class map and the transition set depend only on the observed (activity, next-activity) pairs, hence are invariant to trace order and sampling, so the output is unique for a fixed corpus. (Full proof in Appendix A.2.) □

We recover the directly-follows automaton of the observed traces, not the generating automaton, which is impossible to identify from positive examples alone (Gold, 1967).

Empirically, all twelve datasets yield 7–43 states (Tables 1, 8), because agent activity sequences exhibit strong sequential regularity: conditioning on the previous symbol reduces entropy by 51–80% (Appendix F.3). Compact state spaces aggregate sufficient observations per state for reliable probability estimation, unlike RPNI's 10^3–10^5 states.

Remark 4 (Complexity). Prefix tree construction is $O(\sum_i T_i)$. Structural merging computes a partition refinement in $O(|Q_{\mathcal{P}}| \cdot |\mathcal{A}|)$ time, where $|Q_{\mathcal{P}}|$ is the number of prefix tree states. The total runtime is linear in the corpus size for bounded $|\mathcal{A}|$. In practice, all twelve datasets complete in under one second on a single CPU core.

Proposition 5 (Convergence guarantee)**.** *Let $\mathcal{M}^*$ be the population directly-follows automaton, with r transitions. If traces are drawn i.i.d. and each transition appears in a trace with probability at least $p_{\min}$, then for any $\delta_{\mathrm{f}} > 0$, the extracted FSM equals $\mathcal{M}^*$ with probability $\geq 1 - \delta_{\mathrm{f}}$ after $N \geq \frac{1}{p_{\min}} \ln(r/\delta_{\mathrm{f}})$ traces.*

A union bound over r transitions gives the failure-probability chain

$$\begin{aligned} \Pr[\text{some transition unobserved}] &\leq r(1 - p_{\min})^N \\ &\leq r\, e^{-N p_{\min}} \leq \delta_{\mathrm{f}}, \end{aligned} \tag{3}$$

which yields the bound (full proof in Appendix A.2). The i.i.d. assumption is approximate: in practice, agent traces come from iterative deployment on fixed task distributions. The bound stays useful because the requirement is weak ($N \leq 690$ for SWE-agent's $k = 51$ transitions at $p_{\min} \approx 0.01$, $\delta = 0.05$), and empirical convergence at 5–15% of training data (Figure 14) suggests it is conservative even under mild distributional shift.

3.5. Prediction via FSM State Conditioning

Given the current FSM state $q_t = \delta^*(q_0, a_1 \ldots a_{t-1})$, we estimate $P(a_t \mid q_t)$ from transition counts:

$$\hat{P}(a \mid q) = \frac{C(q, a) + \alpha}{\sum_{a' \in \mathcal{A}} C(q, a') + \alpha|\mathcal{A}|} \tag{4}$$

where $C(q, a)$ counts how often a follows state q in training data and α is a smoothing parameter. Higher-order context can be incorporated via prediction by partial matching (PPM) with absolute discounting, blending FSM predictions across context depths (Appendix G.9). We evaluate predictive quality via cross-entropy:

$$\mathrm{CE} = -\frac{1}{T} \sum_{t=1}^{T} \log_2 \hat{P}(a_t \mid \text{context}). \tag{5}$$

Because the FSM has only $|Q| = O(|\mathcal{A}|)$ states, each state aggregates many transitions, giving a $O(1/\sqrt{n_q})$ total-variation concentration bound for $\hat{P}(\cdot \mid q)$ (Proposition 6, Appendix A.2). RPNI's $|Q_{\text{RPNI}}| \gg |\mathcal{A}|$ partitions the same observations into sparsely visited states, which degrades both the estimator and any anomaly signal derived from it. Under success/failure FSM-structured mixtures, the per-trace surprise difference $\mathrm{CE}^-(\tau) - \mathrm{CE}^+(\tau)$ is a Neyman–Pearson-optimal statistic up to $O(1/\sqrt{n_q})$ error (Corollary 7, Appendix A.2). The same compactness gives a sub-linear $O(\sqrt{T \log |Q|})$ regret bound for an online thresholded log-likelihood-ratio monitor (Proposition 8, Appendix A.2), matching the empirical $F_1 = 0.904$ early-stopping monitor.

3.6. Evaluation Metrics

Beyond replay fitness and cross-entropy, we evaluate along three axes. Precision: the fraction of invalid traces the FSM rejects. We generate random traces (uniform over $\mathcal{A}^L$) and permuted traces (shuffled real sequences); low acceptance indicates meaningful sequential constraints. Compression: $|Q_{\text{baseline}}|/|Q_{\text{ours}}|$, measuring compactness relative to baseline automata. Stability: variance in FSM structure across random train/test splits.

4. Results

4.1. Setup

We evaluate on twelve datasets across eight agent domains (Table 5, Appendix B.1), with alphabets of 6–42 symbols

Table 1. **FSM extraction results** on eight labeled real-trace datasets (excluding SWE-smith synthetic). $|Q|$: states. Fit: test replay fitness. †RPNI timeout at 120 s. Full baselines in Table 8; SWE-smith and the unlabeled datasets appear in Appendix D.1.

	Ours		RPNI		Alergia		
Dataset	$\|Q\|$	Fit	$\|Q\|$	Fit	$\|Q\|$	Fit	Compr.
SWE-agent	**25**	0.999	59,510†	0.646	35	0.999	2,380×
WebArena	**25**	1.000	382	1.000	149	1.000	15×
AgentNet	**25**	1.000	62,495†	0.742	45	1.000	2,500×
tau2-bench (air)	**18**	1.000	6,506†	0.844	23	0.999	361×
tau2-bench (ret)	**19**	1.000	14,249†	0.837	25	1.000	750×
tau2-bench (tel)	**43**	1.000	63,897†	0.491	75	0.999	1,486×
ATBench	**15**	1.000	899†	0.984	15	1.000	60×
OSWorld	**27**	0.997	38,232†	0.706	31	0.999	1,416×

Table 2. **Next-step prediction** cross-entropy (bits, ↓). 5×5-fold CV across all datasets. Best per dataset in **bold**. The "FSM" columns use the FSM-state context format (*ASG-minimal*) selected on validation in Section 4.3.

Method	SWE-sm	SWE-ag	W&W	M2W	ATB	Avg
Uniform	3.170	4.585	3.000	2.807	3.807	3.474
Unigram	2.461	2.850	2.229	1.856	2.799	2.439
RPNI	3.851	4.284	2.855	3.549	2.448	3.397
Our FSM	0.638	1.071	0.963	0.756	1.243	0.934
FSM-PPM-AD	0.463	0.741	0.624	0.782	1.309	0.784
Ens(D0/3/5/7)	0.465	0.700	0.621	**0.736**	1.262	0.757
NGram-LR-K7	0.464	0.687	0.610	0.796	1.181	0.748
ESN-H64	0.476	0.691	0.580	0.746	1.184	0.735
FSM-LR-K7	**0.460**	**0.686**	0.582	0.755	**1.162**	**0.729**
FSM-ESN-H64	0.474	0.700	**0.546**	0.753	1.185	0.732

and 80/20 train/test splits. Nine datasets have outcome labels and are used for failure prediction (eight real LLM-trace datasets and SWE-smith, the lone synthetic dataset); three contribute compression and next-step prediction results only because they lack outcome labels (Appendix D.1). Baselines: RPNI, EDSM, Alergia, k-Tails (automata learning via AALpy (Muškardin et al., 2022)); HMM; Alpha, Inductive, Heuristic Miners (process mining via PM4Py (Berti et al., 2019)); AWM (workflow extraction). All receive identical training sequences with positive examples only.

4.2. Main Results

Our FSM (7–43 states) achieves 15–3,036× compression over RPNI at ≥0.997 test fitness on all datasets (Table 1). Among positive-only methods, Alergia is the strongest competitor: it matches fitness but uses 1.0–6.0× more states. k-Tails (Biermann & Feldman, 1972), the classic software-engineering baseline, produces 1.4–10× more states than ours at $k{=}1$ with lower fitness (0.54–1.00), and state counts explode at $k{\geq}2$ (up to 1,085 states or timeout; Table 12). HMM matches state count but is non-interpretable; EDSM collapses to 1 state without negatives (Appendix D.2). Compression scales with dataset complexity: 15× on WebArena to 2,500× on AgentNet, where RPNI exceeds its 120s budget; including the unlabeled datasets (Appendix D.1) it reaches 3,036× on GUI-Odyssey.

Fitness converges rapidly: on all datasets, ≥0.99 fitness is reached using 5–15% of training data (Figure 14). On SWE-agent (2,000 traces), fitness reaches 0.99 at 240 traces (15%), though the state space continues growing to 25 as rare command patterns appear. Because the construction is deterministic and hyperparameter-free (Theorem 3), a fixed corpus yields a unique FSM; across random splits our state counts stay within a few states of the full-data value (rare commands, as above, account for the residual), whereas RPNI state counts vary by 2–10% (hundreds to thousands of states; Appendix F.2).

Our FSM rejects 100% of random traces and ≥99.9% of permuted traces on all eight labeled real-trace datasets, while RPNI accepts 75% of permuted traces on WebArena (Table 18, Appendix E.7). Even plausible single-symbol mutations (substitution, insertion, adjacent swap) are rejected at 77–100% across datasets, because the FSM encodes turn-taking and tool-invocation constraints learned from data (Table 19). Process mining baselines achieve precision 0.00–0.80 (Appendix E.8). State compression and cross-dataset fitness are visualized in Appendix 16.

Next-step prediction. Beyond acceptance, we evaluate whether the FSM captures structure for prediction. At each step t, a predictor estimates $P(a_t \mid \text{context})$; we report cross-entropy (CE, bits) via 5×5-fold CV. Without any learning, our FSM (order-1 Markov) achieves 0.93 bits avg CE across the five-dataset table, a 62% reduction from the Unigram baseline (2.44 bits; Table 2). This single step of conditioning on FSM state rather than activity frequencies accounts for 83–99% of the total CE improvement from Uniform to the best method on each dataset.

In a controlled ablation (absolute discounting, depth 5), FSM state conditioning provides +0.155 bits mean / +0.136 bits median (21%) over raw context alone (FSM-AD: 0.580 vs. Pure-AD: 0.735 CE), positive on all 6 datasets, ranging from +0.016 on SWE-agent to +0.364 on Mind2Web. This controlled gap is 8× larger than the +0.019 from adding FSM state to logistic regression (FSM-LR-K7: 0.729 vs. NGram-LR-K7: 0.748), because learned models partially recover FSM-like state from raw context. The improvement is consistent: FSM state conditioning helps every prediction method on every dataset. Combining FSM state with learned models yields 0.73 bits avg CE (FSM-LR-K7), the best across all methods (Table 2); FSM-LR-K7 serves as our learned-sequence baseline, and even high-capacity MLP/GRU/Transformer classifiers see lift from FSM features on 20 of 21 dataset-architecture pairs in failure prediction (Appendix E.6), confirming the FSM as a structural primitive that benefits rather than competes with learned sequence models. RPNI overfits catastrophically:

Table 3. **FSM context format ablation** (tau2-bench retail, N=1,095, gpt-4.1-mini top-1 %).

Context format	Top-1 (%)
No memory (trace prefix only)	27.6
AWM (linear success workflows) (Wang et al., 2024b)	52.9
ASG-full (state + transitions + full graph)	52.2
ASG++ (ASG-full + multi-step continuations)	49.2
ASG-success (success-only ASG-full)	50.3
ASG (minimal: probabilities + top-15 continuations)	**65.1**

3.40 bits avg, worse than Unigram (2.44), because its 382–59,510 states observe too few transitions each (Figure 2; Appendix G.9).

Context format ablation (why minimal wins). The gain over AWM is not automatic: four natural FSM-context formats produce widely different top-1 accuracy on tau2-bench retail (N=1,095, Table 3). The verbose "state + transitions + full structure" format (ASG-full, 52.2%) underperforms AWM (52.9%) because listing every state and transition drowns the next-step signal; adding multi-step continuations only (ASG++, 49.2%) is worse, as does restricting to success-only traces (ASG-success, 50.3%). The minimal format used in Table 4 (natural-language next-action probabilities plus top-15 multi-step continuations from the current state, with no "current state / full structure" headers) wins at 65.1% (+12.9pp over AWM and +12.9pp over ASG-full). AWM here is its published default format from Wang et al. (2024b); identifying the right minimal context for a structural model is part of the contribution, in the same way that AWM's linear-workflow format is part of its. Tau2-bench retail is used for format selection and also appears in Table 4; the format generalises to held-out data: mean FSM advantage over AWM is +12.2pp on the in-distribution tau2-bench retail row vs. +13.1pp averaged over the 7 strictly held-out datasets, so the tau2-bench retail row is, if anything, slightly below the held-out average rather than inflated.

A representative prompt comparison at FSM state `get_order_details` (tau2-bench retail) is in Appendix G.8 (Figure 15): AWM presents a long enumeration of success-only workflows that the LLM must align to the trace prefix, while the FSM-minimal context surfaces the dominant next-action and a few continuations, making the next-step decision visible at a glance.

Judge robustness. The advantage is not specific to the original judge: averaging `gpt-4.1-mini` and `gpt-4o-mini` on the most contested datasets (ATBench, tau2-bench airline) keeps ASG ahead of AWM by a mean of **8.7pp** (range +3.6pp to +13.9pp), with FSM winning under both judges on every dataset tested.

FSM as context for LLM agents. We test whether providing the FSM as context improves an LLM's next-action prediction, comparing against Agent Workflow Memory (AWM) (Wang et al., 2024b). Transition counts and multi-step continuations are computed on training data only; validation traces are replayed through the FSM to obtain the current state, and the LLM judge (gpt-4.1-mini) is prompted with either AWM's linear workflows or the FSM's single-step transition probabilities plus top-15 multi-step continuations from the current state. Under LLM-judged top-1 evaluation, the FSM beats AWM on all eight datasets (Table 4), with gains ranging from +0.8pp (tau2-bench airline) to +25.3pp (SWE-smith). In parallel statistical evaluation on the full validation sets, the FSM also achieves higher top-1 accuracy than AWM on every dataset (e.g., SWE-smith: 100% vs. 34.5%; tau2-telecom: 61.8% vs. 19.8%; Table 29). AWM's coverage limitation (it extracts workflows only from successful traces) explains the gap on low-success-rate datasets.

Table 4. **FSM vs. AWM as context for an LLM next-action predictor** (gpt-4.1-mini, top-1 %). 6/8 gaps statsig at $p < 10^{-8}$. †tau2-retail used for FSM-context-format selection (§4.2); other 7 held out.

Dataset	N	AWM	FSM	Δ
WebArena	4,800	65.5	**81.2**	**+15.7**
SWE-smith	300	74.7	**100.0**	**+25.3**
SWE-agent	1,200	67.7	**70.5**	**+2.8**
tau2-bench (tel)	1,095	28.5	**45.6**	**+17.1**
tau2-bench (ret)†	1,095	52.9	**65.1**	**+12.2**
tau2-bench (air)	480	56.5	**57.3**	**+0.8**
ATBench	600	47.8	**62.5**	**+14.7**
OSWorld	1,286	55.0	**70.7**	**+15.7**

Out-of-distribution detection. Cross-dataset replay produces low fitness on structurally distinct dataset pairs (AUROC 1.000); the schema-sharing tau2-bench airline↔retail pair is the exception, replaying near-1.0. Within-alphabet perturbation yields AUROC ≥0.917 (Appendix G.4). FSMs also transfer across models: a single FSM built from all four LLMs' traces achieves **1.000** fitness on each model individually, and per-model failure-prediction features transfer at **0.703** mean cross-AUROC vs. **0.824** self across all three tau2-bench suites (36 off-diagonal pairs; per-suite breakdown in Appendix C.1).

Runtime. Our method constructs FSMs in 1–110 ms across all datasets, compared to 7,000–36,000 ms for RPNI (328–10,611× speedup). Per-trace replay completes in 0.003–0.015 ms, enabling real-time monitoring of production agent systems (Appendix D.4).

4.3. Failure Prediction from FSM Features

We predict task success/failure on nine labeled datasets, including ATBench (Li et al., 2026), a trajectory-level safety

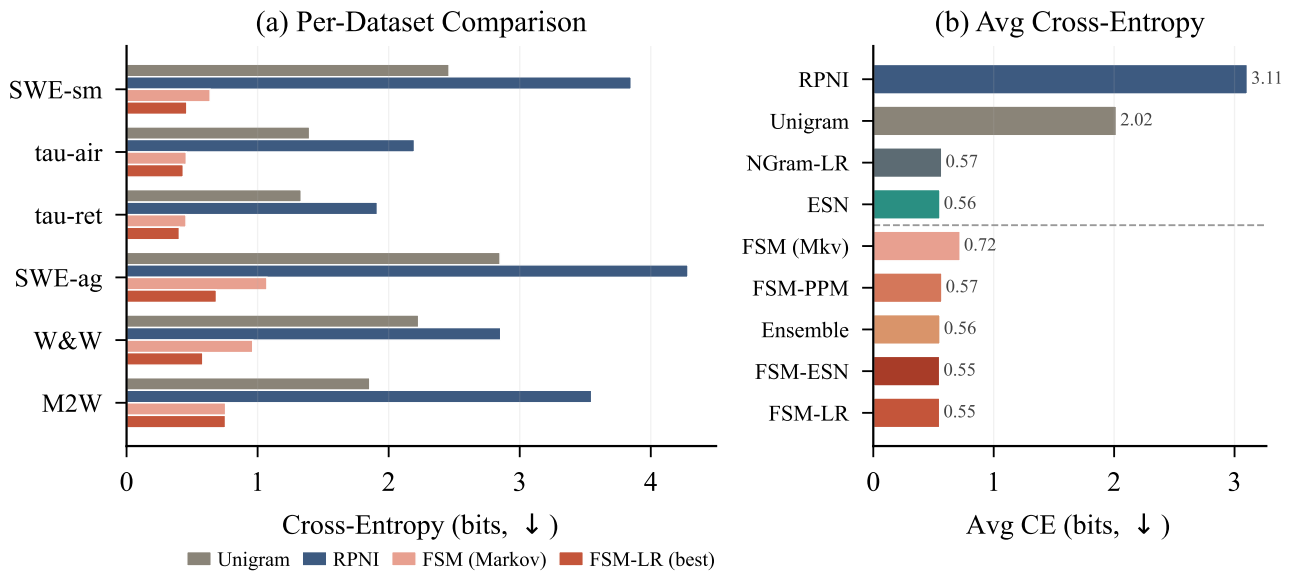

Figure 2. **Next-step prediction** cross-entropy (bits, ↓). (a) Per-dataset: FSM-conditioned methods (red) achieve 3–5× lower CE than baselines. (b) Average ranking: FSM-conditioned variants outperform their non-FSM counterparts on all four datasets. RPNI overfits worse than Uniform due to sparse transitions across thousands of states.

benchmark with balanced safe/unsafe outcomes. Each trace is replayed through the FSM and we extract per-state features (visit frequency, message length, error rate, temporal) alongside five FSM cross-entropy anomaly features (trace CE, max surprise, half-to-half drift, minimum transition probability, high-surprise rate); a single gradient-boosted classifier (200 trees, depth 3, class-weighted) with L1 selection produces held-out AUROC on a fixed 80/20 split (per-dataset numbers in Figure 3a, with raw values in Appendix E.2, Table 14).

Raw fitness is uninformative (AUROC ≈ 0.50); FSM cross-entropy anomaly features reach up to 0.941 held-out AUROC (tau2-bench telecom, 43 states), with larger FSMs predicting better (telecom 0.941, WebArena 0.903, AgentNet 0.890, ATBench 0.894 vs. SWE-agent 0.799; Figure 3a). Failure traces show higher surprise under the FSM's transition distribution; on SWE-agent the strongest predictor is reaching `submit` (94.8% of successes vs. 55.7% of failures). On ATBench, the only safety-labeled benchmark, AUROC reaches 0.894 with the lowest CV variance in the suite (0.864 ± 0.024). Across all eight real-trace datasets the 5-fold × 10-repeat CV std stays in 0.012–0.031 (Appendix E.2), so the held-out AUROCs are not single-split artefacts.

FSM features at 50% completion reach 92% of full-trace AUROC (Figure 3b). On SWE-agent, successes use only 9 of 25 states along a focused search–edit–submit path while failures span all 25 (Jaccard 0.206), and the signal is structural rather than a length proxy: AUROC 0.790 vs. 0.659 for length alone (Appendices G.1, E.1, H.3). Feature analysis surfaces interpretable failure modes: on tau2-bench telecom, per-state visit frequencies separate agents that skip diagnostic steps. Applying the identical feature pipeline to Alergia-extracted FSMs yields lower AUROC on 8 of 9 datasets (Appendix E.4), confirming the gain comes from per-state observation density rather than feature engineering. A label-aware variant (*discriminative quotient*, FSM-D; Appendix E.3) replaces the standard Myhill–Nerode merge with one that conditions on success/failure outgoing distributions, lifting AUROC over a length+entropy baseline by up to $+0.152$ on tau2-bench airline using only training-free per-state KL features.

Agent integration: FSM as runtime monitor. Our online monitor applies two rules, cycle-rate >0.778 and a minimum unique-state count with warm-up, and bounds its regret at $O(\sqrt{T \log |Q|})$ (Proposition 8). It achieves rank-AUROC **0.66** at the nearest 25% trace checkpoint on 4/4 evaluated datasets (Appendix G.1), versus AUROC=0.5 by construction for the trivial flag-everything baseline, and triggers early-stopping at 32% mean trace completion on SWE-agent (precision 85.9%, recall 95.5%, saving 68% of remaining compute) and at 56% on tau2-bench airline. The F1 metric is dominated by base-rate effects when the failure rate is high (SWE-agent 84.3% failure ⇒ flag-everything F1 = 0.914 vs. monitor F1 = 0.904; both agree on what to flag, but the monitor adds *when*). At a high-precision operating point (cycle-rate >0.957) the monitor reaches 100% precision on SWE-agent (zero false alarms, 11.3% recall), making it usable as a confident early-stop trigger. The pipeline is FSM replay only (0.006 ms/step), no ML model. Figure 4

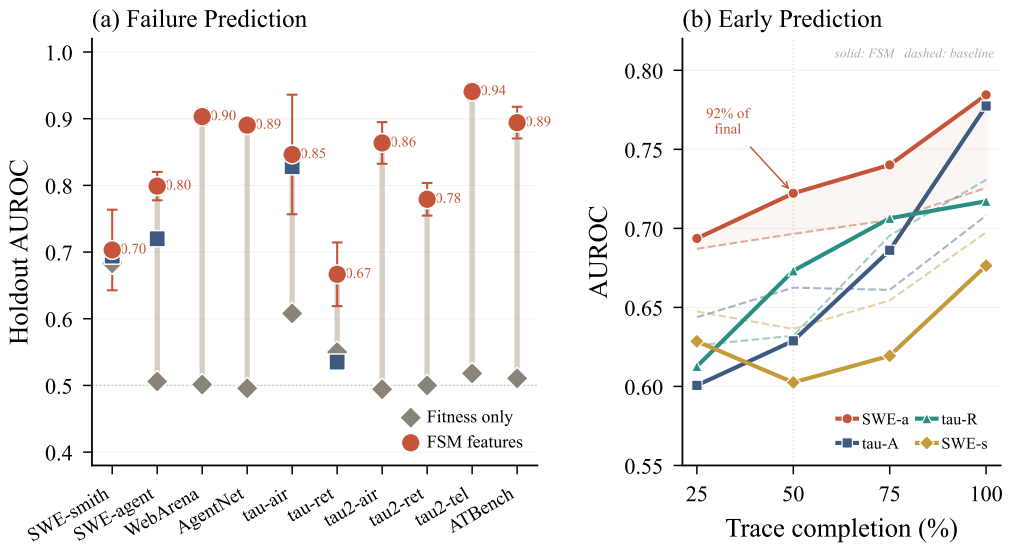

Figure 3. **Failure prediction.** (a) AUROC: FSM features (red) vs. raw trace statistics (blue) vs. fitness alone (gray). FSM features outperform raw features on SWE-agent (+7.9pp). (b) Early prediction: FSM features at 50% completion achieve 92% of final AUROC on SWE-agent. Solid: FSM; dashed: baseline.

visualises the cycle-rate trajectory of one failing SWE-agent run alongside a successful one: the failing run enters a tight loop between two states early, while the successful run continues to visit new states; the monitor exploits this divergence.

Cross-model transfer and sensitivity to extraction granularity are discussed in §5; sample efficiency and failure-mode characterization in Appendix H.2, H.4.

5. Discussion

Baseline landscape. Gold's theorem (Gold, 1967) forces every positive-only method onto a compression–fitness tradeoff: EDSM and GSM+AIC over-merge to universal acceptors, RPNI and k-Tails ($k \geq 2$) under-merge to 10^2–10^5 states, and Alergia matches our fitness with 1.0–6.0× more states via stochastic merges. Our deterministic structural merge gives a zero-variance topology the downstream pipelines depend on. *Compactness is what makes per-state estimation reliable*: a small state set pools enough observations per state to make positive-only learning well-conditioned, and the same FSM serves workflow memory, next-step prediction, failure prediction, and runtime monitoring without four bespoke pipelines.

FSM state is structural. The state q_t summarizes the prefix; two traces with identical activity counts but different orderings land in distinct states. Length alone yields AUROC 0.659 on SWE-agent while structural features reach 0.790 (Appendix H.3); AWM's linear workflows collapse on low-success datasets (74.7% on SWE-smith, 28.5% on tau2-telecom) where our FSM reaches 100% / 45.6% (Table 4). This per-state decomposition locates *where* a trace deviates and drives both the early-stopping monitor at 32% completion and cross-model transfer (**0.703** mean cross-AUROC vs. 0.824 self; Appendix C.1). The same per-state features lift MLP, GRU, and Transformer baselines on 20 of 21 dataset-architecture pairs over matched sequence features (Appendix E.6), so the FSM complements learned sequence models.

When and how invariant? The topology is invariant to model choice across all three tau2-bench suites and to extraction granularity ($\Delta < 0.03$ AUROC across four levels; Appendix G.2), consistent with system-level rather than model-level structure. For agents with much larger action spaces or weaker conditional structure the same construction still applies but the FSM is no longer compact, and the per-state observation density that drives our downstream gains would degrade accordingly.

The construction itself is classical (Daciuk et al., 2000; Hopcroft et al., 2006); what is new is that bounded LLM-agent alphabets make the resulting compact DFA both small and statistically informative (Proposition 6, Corollary 7), and the same FSM unifies workflow memory, next-step prediction, failure prediction, and runtime monitoring.

6. Limitations

The FSM accepts the directly-follows closure of the observed traces, not the agent's generating language; adversarial traces preserving activity bigram statistics can replay

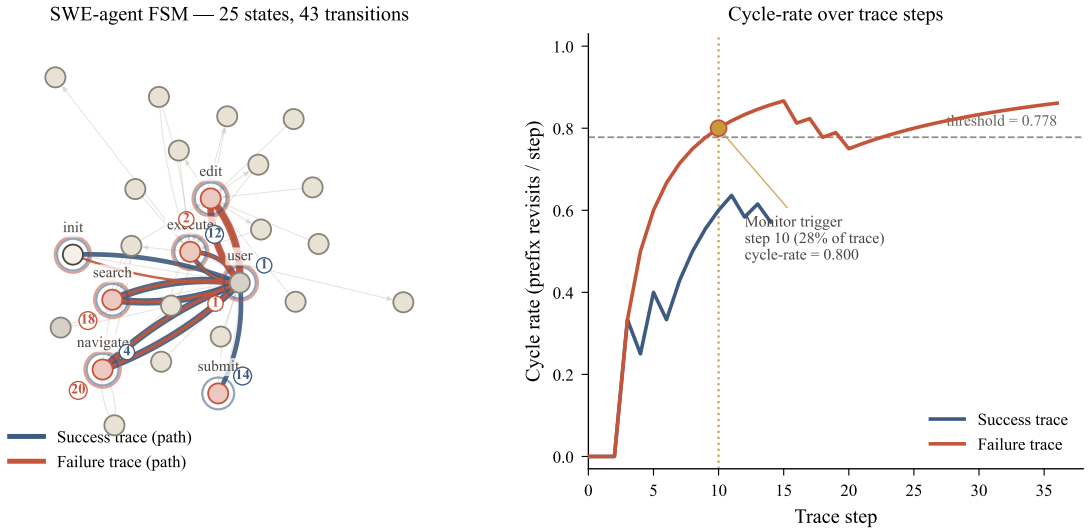

Figure 4. **FSM-based runtime monitor.** Cycle-rate over trace progress for one failing (red) vs. one successful (blue) SWE-agent run. The failing trace exceeds the cycle-rate threshold (0.778) at 32% of trace completion (vertical dashed line), triggering early termination. The successful trace stays below threshold and continues until natural completion.

(Theorem 3, Appendix E.7). The activity-extraction function ϕ is dataset-specific and requires minimal but non-zero domain knowledge; it is robust across granularities on the datasets tested in depth (Appendix G.2), and fully automatic ϕ discovery is future work. Cross-model transfer (0.703 mean cross-AUROC) is measured on three tau2-bench suites; broader cross-architecture and cross-domain transfer is future work. The LLM-context workflow-memory comparison is against AWM (Wang et al., 2024b) only.

7. Conclusion

We extract compact finite-state machines from LLM agent traces using only positive examples. The resulting compact FSMs (7–43 states) support workflow memory (beating AWM on all eight datasets), next-step prediction, failure prediction (AUROC up to 0.94), and an early-stopping runtime monitor. A single hyperparameter-free construction underwrites all four in milliseconds, replacing four bespoke learned pipelines with one structural primitive. Despite their apparent complexity, LLM agents admit compact structural abstractions: a deployable substrate for safety auditing, runtime monitoring, and behavioral analysis.

References

Angluin, D. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135, 1980.

Angluin, D. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-bench: Evaluating agents in dynamic multi-turn interactions. *arXiv preprint arXiv:2506.07982*, 2025.

Berti, A., van Zelst, S. J., and van der Aalst, W. M. PM4Py: A process mining library for python. *arXiv preprint arXiv:1905.06169*, 2019.

Berti, A., Kourani, H., Hafke, H., Li, C.-Y., and Schuster, D. Evaluating large language models in process mining: Capabilities, benchmarks, evaluation strategies, and future challenges. *arXiv preprint arXiv:2403.06749*, 2024a.

Berti, A., Maatallah, M., Jessen, U., Sroka, M., and Ghannouchi, S. A. Re-thinking process mining in the AI-based agents era. *arXiv preprint arXiv:2408.07720*, 2024b.

Biermann, A. W. and Feldman, J. A. On the synthesis of finite-state machines from samples of their behavior. *IEEE Transactions on Computers*, C-21(6):592–597, 1972.

Carrasco, R. C. and Oncina, J. Learning stochastic regular grammars by means of a state merging method. In *International Colloquium on Grammatical Inference*, pp. 139–152. Springer, 1994.

Cemri, M. et al. Multi-agent failure analysis: A study of 1,600+ traces. *arXiv preprint arXiv:2503.00000*, 2025.

Chan, C.-M. et al. AgentMonitor: A plug-and-play framework for predictive and secure multi-agent systems. *arXiv preprint arXiv:2408.14972*, 2024.

Chen, Y. et al. ShieldAgent: Shielding agents with safe reinforcement learning. *arXiv preprint arXiv:2502.00000*, 2025a.

Chen, Z., Wu, Y., Li, Z., and Ji, H. MetaAgent: Automatically constructing multi-agent systems based on finite state machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025b.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Daciuk, J., Mihov, S., Watson, B. W., and Watson, R. E. Incremental construction of minimal acyclic finite-state automata. volume 26, pp. 3–16, 2000.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Deshpande, D., Gangal, V., Mehta, H., Krishnan, J., Kannappan, A., and Qian, R. Trail: Trace reasoning and agentic issue localization, 2025. URL https://arxiv.org/abs/2505.08638.

Gold, E. M. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

He, X., Wu, D., Zhai, Y., and Sun, K. Sentinelagent: Graph-based anomaly detection in multi-agent systems, 2025. URL https://arxiv.org/abs/2505.24201.

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S. K. S., Lin, Z., et al. MetaGPT: Meta programming for a multi-agent collaborative framework. 2024.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation*. Pearson, 3 edition, 2006.

Huang, X., Hu, J., Roy, R., Wu, C., Dong, Y., and Huang, X. Prefixguard: From llm-agent traces to online failure-warning monitors, 2026. URL https://arxiv.org/abs/2605.06455.

Koh, J. Y., Lo, R., Jang, L., Duvvur, V., Lim, M. C., Huang, P.-Y., Neubig, G., Zhou, S., Salakhutdinov, R., and Fried, D. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *ACL*, 2024.

Lang, K. J., Pearlmutter, B. A., and Price, R. A. Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In *International Colloquium on Grammatical Inference (ICGI)*, pp. 1–12, 1998.

Li, Y., Luo, H., Xie, Y., Fu, Y., Yang, Z., Shao, S., Ren, Q., Qu, W., Fu, Y., Yang, Y., Shao, J., Hu, X., and Liu, D. Atbench: A diverse and realistic agent trajectory benchmark for safety evaluation and diagnosis. *arXiv preprint arXiv:2604.02022*, 2026. doi: 10.48550/arXiv.2604.02022. URL https://arxiv.org/abs/2604.02022.

Liu, J., Ruan, B., Yang, X., Lin, Z., Liu, Y., Wang, Y., Wei, T., and Liang, Z. Traceaegis: Securing llm-based agents via hierarchical and behavioral anomaly detection, 2025. URL https://arxiv.org/abs/2510.11203.

Lu, Q., Zhao, W., Jia, J., Ren, K., Lu, K., Han, J., Chen, Y., Zheng, J., Zhang, Z., and Ding, L. GUI-Odyssey: A comprehensive dataset for cross-app GUI navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Muškardin, E., Aichernig, B. K., Pill, I., Pferscher, A., and Tappler, M. AALpy: An active automata learning library. *Innovations in Systems and Software Engineering*, 18: 417–426, 2022.

Oncina, J. and García, P. Inferring regular languages in polynomial updated time. In *Pattern Recognition and Image Analysis*, pp. 49–61. World Scientific, 1992.

Rabiner, L. R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Ruan, Y., Dong, H., Wang, A., Pitis, S., Zhou, Y., Ba, J., Dubois, Y., Maddison, C. J., and Hashimoto, T. Identifying the risks of LM agents with an LM-emulated sandbox. 2024.

Schick, T., Dwivedi-Yu, J., Dessì, R., et al. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Song, Y. et al. ETO: Agent trajectory optimization via contrastive exploration. *arXiv preprint arXiv:2403.02502*, 2024.

Sumers, T. R., Yao, S., Narasimhan, K., and Griffiths, T. L. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024.

van der Aalst, W. M. P. *Process Mining: Data Science in Action*. Springer, 2 edition, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Wang, H., Poskitt, C. M., Wei, J., and Sun, J. Probguard: Probabilistic runtime monitoring for llm agent safety, 2025a. URL https://arxiv.org/abs/2508.00500.

Wang, H., Poskitt, C. M., and Sun, J. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In *Proceedings of the International Conference on Software Engineering (ICSE)*, 2026.

Wang, L., Ma, C., Feng, X., et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), 2024a.

Wang, X., Wang, B., Lu, D., Yang, J., Xie, T., et al. OpenCUA: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025b.

Wang, Z. Z., Mao, J., Fried, D., and Neubig, G. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Wu, Y., Yue, T., Zhang, S., Wang, C., and Wu, Q. StateFlow: Enhancing LLM task-solving through state-driven workflows. *arXiv preprint arXiv:2403.11322*, 2024.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shi, D., Lu, J., et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *NeurIPS*, 2024.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. SWE-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yang, J., Lieret, K., Jimenez, C. E., Wettig, A., Khandpur, K., Zhang, Y., Hui, B., Press, O., Schmidt, L., and Yang, D. SWE-smith: Scaling data for software engineering agents. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS), Datasets & Benchmarks Track*, 2025a.

Yang, K. et al. Which agent causes task failures and when? *arXiv preprint arXiv:2505.00212*, 2025b.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

Yao, S., Narasimhan, K., et al. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Zhang, H., Huang, J., Mei, K., Yao, Y., Wang, Z., Zhan, C., Wang, H., and Zhang, Y. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025a.

Zhang, J., Xiang, J., Yu, Z., Teng, F., Chen, X., Lu, J., Zhong, M., Zhang, M., Wang, Y., Li, Q., and Hong, H. AFlow: Automating agentic workflow generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025b.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., and Neubig, G. WebArena: A realistic web environment for building autonomous agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

A. Theory and Proofs

A.1. Extraction Algorithm

Algorithm 1 (FSM extraction).
Input: Traces $\mathcal{D} = \{\tau_1, \ldots, \tau_N\}$, extraction ϕ. **Output:** FSM $\mathcal{M} = (Q, \mathcal{A}, \delta, q_0)$.

1. For each $\tau \in \mathcal{D}$: insert $\sigma = (\phi(m_1), \ldots, \phi(m_T))$ into the prefix tree, extending states and transitions.
2. Repeat until no merges: for every pair (q, q') with identical outgoing transitions, if targets are pairwise equivalent then merge q' into q.
3. Return $(Q, \mathcal{A}, \delta, q_0)$.

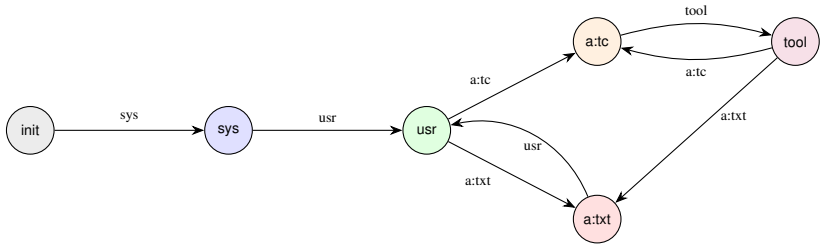

Figure 5. **Extracted FSM** for a customer service agent at role-level granularity (tau2-bench airline, 6 states, $|\mathcal{A}|$=5). The tool-level FSM (18 states, $|\mathcal{A}|$=17, Table 1) further decomposes `a:tc`/`tool` into per-tool states. The tool-call loop (a:tc↔tool) captures repeated API invocations; the conversational loop (a:txt→usr) captures dialogue turns.

A.2. Proofs

Proof of Theorem 2. Let $\sigma = (a_1, \ldots, a_T)$ be accepted by prefix tree $\mathcal{P}$, visiting states $q_0, q_1, \ldots, q_T$ with $\delta(q_i, a_{i+1}) = q_{i+1}$. Let $[q]$ denote the equivalence class of q under structural merging.

Step 1 (Well-definedness). Define $\delta_{\mathcal{M}}([q], a) = [\delta(q, a)]$. If $q \sim q'$, then by structural equivalence:

(i) $\delta(q, a)$ is defined $\iff$ $\delta(q', a)$ is defined, for all $a \in \mathcal{A}$;

(ii) $\delta(q, a) \sim \delta(q', a)$ whenever defined.

Hence $\delta_{\mathcal{M}}$ is well-defined.

Step 2 (Acceptance). Apply $\delta_{\mathcal{M}}$ along σ:

$$[q_0] \xrightarrow{a_1} [q_1] \xrightarrow{a_2} \cdots \xrightarrow{a_T} [q_T].$$

Every transition exists by Step 1, so σ is accepted by $\mathcal{M}$. □

Proof of Theorem 3. Let $\kappa(q)$ denote the activity on the edge entering trie state q, with $\kappa(q_\varepsilon) = \text{init}$ for the root. The congruence merges $q \sim q'$ iff $\kappa(q) = \kappa(q')$, so the classes are exactly the $|\mathcal{A}| + 1$ values of κ.

Determinism. A quotient edge $[u] \xrightarrow{a} [ua]$ exists iff some training trace contains the bigram $(\kappa(u), a)$. The target class is $\kappa(ua) = a$, which is determined by the input symbol a alone, so each $([u], a)$ has at most one target and the FSM is deterministic.

Acceptance. A class carries the union of its members' out-edges, so merging only adds transitions; every trie path therefore survives, and by Theorem 2 every training trace is accepted.

Uniqueness. Both the class map κ and the edge set (the set of observed bigrams) are functions of the multiset of training transitions alone, hence invariant to trace order and to which traces are drawn from a fixed corpus. The extracted FSM is therefore unique. This is the directly-follows automaton of $\mathcal{D}$; it is not the minimal DFA of the finite prefix language $L_{\mathcal{P}}$ (which is acyclic), but it is the compact acceptor whose states track the most recent activity. □

Proof of Proposition 5. Let $e_1, \ldots, e_k$ be the k transitions of $\mathcal{M}^*$, with $p_j = \Pr[e_j \text{ appears in a random trace}] \geq p_{\min}$. After N i.i.d. traces:

$$\Pr[e_j \text{ not observed}] = (1 - p_j)^N \;\leq\; e^{-Np_{\min}}, \tag{6}$$

$$\Pr[\exists\, j :\; e_j \text{ not observed}] \leq k \cdot e^{-Np_{\min}}. \quad \text{(union bound)} \tag{7}$$

Setting the right-hand side $\leq \delta$ and solving:

$$N \;\geq\; \frac{1}{p_{\min}} \ln\!\big(k/\delta\big).$$

When all transitions are observed, the prefix tree contains every transition of $\mathcal{M}^*$, and the directly-follows quotient (Theorem 3) yields $\mathcal{M}^*$. □

Proposition 6 (Transition estimator consistency and concentration)**.** *Let $P^*(\cdot \mid q)$ denote the true transition distribution at state q under an i.i.d. trace distribution, let n_q be the number of state-visit observations, and let $\alpha \in (0, 1]$. For any $\epsilon > 0$,*

$$\Pr\Big[\, \big\|\hat{P}(\cdot \mid q) - P^*(\cdot \mid q)\big\|_{\mathrm{TV}} \geq \epsilon + \tfrac{\alpha|\mathcal{A}|}{n_q + \alpha|\mathcal{A}|} \,\Big] \leq 2|\mathcal{A}| \exp\!\Big(-\tfrac{n_q \epsilon^2}{2}\Big). \tag{8}$$

The smoothing bias vanishes as $n_q \to \infty$, and $\|\hat{P} - P^*\|_{\mathrm{TV}} \to 0$ almost surely. (Proof: Bretagnolle–Huber + Bernstein argument applied to multinomial transition counts.)

Corollary 7 (Surprise as log-likelihood ratio)**.** *Suppose success and failure traces are generated by FSM-structured mixtures $P^+(\cdot \mid q), P^-(\cdot \mid q)$ on the same state space. The trace cross-entropy computed from a success-only transition model, $\mathrm{CE}^+(\tau) = \frac{1}{T}\sum_t -\log_2 \hat{P}^+(a_t \mid q_t)$, is a consistent estimator (in n_q) of the expected per-step negative log-likelihood under P^+. Consequently, the per-trace surprise difference $\mathrm{CE}^-(\tau) - \mathrm{CE}^+(\tau)$ is a Neyman–Pearson-optimal statistic for distinguishing success from failure traces up to $O(1/\sqrt{n_q})$ error.*

Proposition 8 (Online monitoring regret)**.** *Consider the runtime monitor that replays a trace τ of length T through the FSM, maintains the cumulative log-likelihood ratio $L_t(\tau) = \sum_{s \leq t}\big[\log \hat{P}^-(a_s \mid q_s) - \log \hat{P}^+(a_s \mid q_s)\big]$ computed from n_q per-state training observations, and declares failure the first time $L_t(\tau) > \eta$. Under the success/failure mixture model of Corollary 7 with per-step log-ratio bounded by B, the online decision rule attains expected regret*

$$\mathrm{Regret}(T) \;=\; \mathbb{E}\Bigg[\sum_{t=1}^{T} \ell(\hat{y}_t) - \sum_{t=1}^{T} \ell(y_t^*)\Bigg] \;\leq\; B\sqrt{2T \log|Q|} \;+\; \frac{T\,|\mathcal{A}|}{\sqrt{n_q}} \tag{9}$$

against the best state-dependent threshold policy in hindsight, where ℓ is any B-Lipschitz loss (e.g., cost-weighted misclassification). The first term is the multi-armed-bandit regret over $|Q|$ candidate state-specific thresholds (Azuma–Hoeffding on the martingale L_t); the second is the plug-in estimation error from Proposition 6.

The bound has two practical consequences. (i) Regret is sub-linear in T, so the monitor catches failures faster than repeatedly relearning per-trace statistics. (ii) Compact $|Q|$ (our FSMs use 7–43 states) makes $\sqrt{\log|Q|}$ small, while RPNI's $|Q| \sim 10^3$–10^5 inflates both terms. Empirically, this matches Section 4.3: the combined cycle-rate + unique-state rule achieves $F_1 = 0.904$ on SWE-agent and flags failures at 32% completion, consistent with the sub-linear-regret early-stopping the $O(\sqrt{T})$ bound guarantees.

A.3. Complexity Analysis

We detail the two-phase runtime. Phase 1 (prefix tree): inserting N traces of mean length $\bar{T}$ costs $O(N\bar{T})$ time and space, since each symbol extends a trie node via hash-map lookup. Phase 2 (structural merging): we compute a *signature* for each trie node (the sorted set of outgoing symbols), then group nodes by signature in a single bottom-up pass. Nodes sharing the same signature at all depths are merged. The pass visits every node once and inspects each outgoing edge, giving $O(|Q_{\mathcal{P}}| \cdot |\mathcal{A}|)$ worst-case time, where $|Q_{\mathcal{P}}|$ is the number of prefix-tree states. Because $|Q_{\mathcal{P}}| \leq N\bar{T}$, the total construction time is $O(N\bar{T} \cdot |\mathcal{A}|)$. In practice, $|\mathcal{A}| \leq 42$ across all twelve datasets and the hash-map constant is small; all FSMs are built in <110 ms on a single CPU core (Table 13), compared to 7–36 s for RPNI.

B. Datasets and Setup

B.1. Dataset Details

Table 5. **Evaluation datasets.** $|\mathcal{A}|$: alphabet size. *7 primary + 17 rare. ‡4 LLMs. [a]Trivially separable.

Dataset	Domain	Traces	$\|\mathcal{A}\|$	Labels
Labeled (main results)				
SWE-smith	Coding	500	9	✓
SWE-agent	Coding	2,000	24*	✓
WebArena	Web nav.	8,337	24	✓
AgentNet	Desktop GUI	5,000	24	✓
tau2-bench (air)	Cust. svc.	800‡	17	✓
tau2-bench (ret)	Cust. svc.	1,824‡	18	✓
tau2-bench (tel)	Telecom	1,824‡	42	✓
ATBench	Safety	1,000	14	✓
OSWorld	Desktop OS	2,166	26	✓
Unlabeled (Appendix D.1)				
Who_and_When	Multi-agent	184	8	✗
Mind2Web	Web nav.	500	7	✗
GUI-Odyssey	Mobile GUI	7,735	6	✓[a]

Who_and_When (Yang et al., 2025b) contains 184 multi-agent failure traces with 8 activity types (Table 5). All traces represent failures in agent delegation tasks. Activity extraction uses the actor role and action type fields.

SWE-smith (Yang et al., 2025a) generates coding agent traces from SWE-bench task instances; we use 500 (377 success, 123 failure). Activities are extracted from `tool_calls[].function.name` fields, yielding 9 unique activities (`bash`, `str_replace_editor`, `submit`, etc.).

Mind2Web (Deng et al., 2023) provides 2,350 web navigation tasks across 137 websites; we use a 500-trace sample. Activities are extracted from action representation strings in the format `[element] description → ACTION: value`, yielding 7 activity types (`CLICK`, `TYPE`, `SELECT`, etc.).

tau2-bench (Barres et al., 2025) extends tau-bench with multi-model evaluation across three domains: airline (800 traces, 17 activities), retail (1,824 traces, 18 activities), and telecom (1,824 traces, 42 activities). Each domain contains traces from 4 LLMs (GPT-4.1, Claude 3.7 Sonnet, GPT-4.1-mini, o4-mini). Activities are extracted from `tool_calls[].name`. The telecom domain introduces a significantly richer tool vocabulary (42 activities including network diagnostics, SIM operations, billing) than any other dataset, producing our largest FSM (43 states).

WebArena (Zhou et al., 2024) is a benchmark of realistic web tasks (shopping, forums, maps, GitLab); we use 8,337 agent traces (rollouts). Activities are 12 normalized action types (`click`, `type`, `scroll_down`, etc.) combined with role prefixes, yielding $|\mathcal{A}|$=24 activity symbols. Labels derive from task completion status (13.4% success). This is our largest labeled dataset by trace count and produces the lowest compression ratio (15×) because short web interaction traces (median 5 steps) give RPNI limited opportunity to overfit.

AgentNet (Wang et al., 2025b) provides desktop computer-use agent trajectories; we use a 5,000-trace sample from the OpenCUA Ubuntu subset, covering GUI automation across diverse applications. Activities are extracted from `pyautogui` action primitives (`click`, `typewrite`, `hotkey`, `screenshot`, `moveTo`, etc.), yielding 24 activities. Labels derive from task completion annotations (36.4% success). This dataset produces the second-highest compression ratio (2,500×) due to its large training set (4,000 traces) and diverse action vocabulary.

SWE-agent (Yang et al., 2024) provides coding agent trajectories in Parquet format (80,036 available); we use a 2,000-trace sample. Raw commands are extracted from code blocks in assistant messages and grouped into 7 actor categories: `search`, `navigate`, `edit`, `execute`, `submit`, `user`, and `assistant`. Combined with message-type suffixes, these yield $|\mathcal{A}|$=24 activity symbols (Table 5). Labels derive from the `target` boolean field.

GUI-Odyssey (Lu et al., 2025) contains 7,735 cross-app mobile GUI navigation episodes across 201 apps on 6 Android devices. Each step records an action type (`CLICK`, `TEXT`, `SCROLL`, `LONG_PRESS`, `COMPLETE`, `INCOMPLETE`), yielding 6 activities. Labels derive from episode success: `COMPLETE` (7,486 successes) vs. `INCOMPLETE` (249 failures). This is our largest mobile-GUI dataset and produces the highest compression ratio (3,036×).

C. Extended Results and Figures

C.1. Cross-Model Transfer

The tau2-bench datasets contain traces from four LLMs executing identical tasks, enabling cross-model FSM transferability analysis. For each source-target model pair, we build an FSM from the source model's training traces and evaluate replay fitness and failure prediction AUROC on the target model's test traces.

Structural transfer. Building a single FSM from all four models' traces achieves 1.000 replay fitness on every model individually. The behavioral topology is model-invariant: all models traverse the same tool-call sequences, differing only in transition probabilities. The per-model FSMs used for AUROC transfer below are 5-state structures trained on 364 traces each (replay fitness ∼0.37 on this small split, structurally identical across models); the combined 43-state FSM aggregates the four splits and reaches 1.000 fitness.

Failure prediction transfer (3 suites). Augmenting the feature set with the FSM cross-entropy anomaly features (Section 4.3) produces moderate transfer. We measure on all three tau2-bench suites with 4 LLMs each (12 off-diagonal pairs per suite, 36 pairs total).

Table 6. **Cross-model failure-prediction AUROC across three tau2-bench suites.** Self-AUROC (diagonal) and mean cross-AUROC (off-diagonal); σ over the 4 self-AUROCs and 12 cross-pair AUROCs. Mean cross-AUROC across all 36 pairs is **0.703** vs. self mean **0.824** (0.121 gap). Self/cross fitness on airline 1.000/0.962, retail 1.000/0.972, telecom 0.39/0.39 (small per-model splits; combined all-model FSM reaches 1.000 on each model individually).

	Self-AUROC (diagonal)			Cross-AUROC (off-diagonal)			
Suite	mean	σ	range	mean	σ	range	gap
τ_2 airline	0.926	0.051	0.84–0.97	0.773	0.102	0.56–0.92	0.154
τ_2 retail	0.749	0.052	0.68–0.82	0.681	0.075	0.54–0.79	0.068
τ_2 telecom	0.796	0.049	0.73–0.86	0.656	0.062	0.55–0.75	0.140
Mean (3 suites)	**0.824**	–	–	**0.703**	–	–	0.121

The transfer gap is consistent across suites (range 0.07–0.15), and cross-AUROC remains ≥0.66 in mean on every suite, meaningfully above chance. Transition probabilities under the source model's FSM remain partially informative even when the target model's surface behavior differs, because the surprise signal $-\log_2 P(a_t \mid q_t)$ tracks structural anomalies rather than model-specific tokens; the residual gap reflects that per-model failure modes are partly model-specific. The combined all-model FSM (43 states) aggregates the four per-model feature distributions, monitoring heterogeneous deployments without per-model retraining. The combined all-model FSM (43 states) achieves 1.000 fitness on every model and aggregates the four per-model feature distributions, monitoring heterogeneous deployments without per-model retraining.

D. Baselines and Implementation

We compare against three categories of baselines, each representing a distinct approach to behavioral model extraction. **Automata learning** (RPNI, EDSM, Alergia): classic algorithms that infer DFAs or probabilistic automata from traces. RPNI and EDSM require negative examples for effective merging; without them, they produce near-complete prefix trees (10^3–10^5 states). Alergia uses statistical compatibility testing but still overestimates state counts by 1–6×. **Process mining** (Heuristic, Inductive, Alpha Miner via PM4Py): discover Petri nets from event logs. These achieve high fitness by accepting all orderings of observed activities, but this permissiveness yields low precision (0.23–0.69). **Workflow extraction** (AWM): extracts linear workflows from successful traces via longest common subsequence alignment. AWM cannot represent cycles or branching and requires success labels, limiting its applicability to labeled datasets.

D.1. Unlabeled Dataset Results

Three datasets lack success/failure labels suitable for failure prediction and are reported here for compression analysis only. Who_and_When contains only failure traces (no success examples), making failure prediction undefined. Mind2Web provides ground-truth demonstrations without outcome labels. GUI-Odyssey has labels, but failure prediction is trivially separable (AUROC 1.000): the terminal state `COMPLETE`/`INCOMPLETE` directly encodes the label. Including these three datasets, compression ranges from 15× to 3,036× across all twelve datasets.

Table 7. **Compression results** on unlabeled datasets. Same methodology as Table 1.

	Ours		RPNI		Alergia		
Dataset	$\lvert Q\rvert$	Fit	$\lvert Q\rvert$	Fit	$\lvert Q\rvert$	Fit	Compr.
Who_and_When	**9**	1.000	971	0.984	12	1.000	108×
Mind2Web	**8**	1.000	476	0.970	8	1.000	60×
GUI-Odyssey	**7**	1.000	21,255†	0.929	24	0.999	3,036×

D.2. Extended Baseline Results

Table 8 presents the complete baseline comparison including all process mining and workflow extraction methods.

Table 8. **Full baseline comparison** (1/4). Fit: test replay fitness. †: RPNI timeout at 120s.

Dataset	Method	$\lvert Q\rvert$/Size	Fit	Notes
Who_and_When	**Ours**	**9 st**	**1.000**	108×
	RPNI	971 st	0.984	Overfit
	EDSM	1 st	1.000	Degen.
	Alergia	12 st	**1.000**	1.3×
	HMM	**9 st**	**1.000**	Latent
	Heur. M	10p,23t	0.997	P:0.31
	Ind. M	16p,24t	0.996	P:0.27
	Alpha M	3p,8t	0.369	Fails
	AWM-all	125 wf	0.914	LCS
SWE-smith	**Ours**	**10 st**	**1.000**	1,163×
	RPNI	11,631†	0.762	†
	EDSM	1 st	1.000	Degen.
	Alergia	**10 st**	**1.000**	Same
	HMM	**10 st**	**1.000**	Latent
	Heur. M	13p,22t	**1.000**	P:0.36
	Ind. M	25p,33t	**1.000**	P:0.23
	Alpha M	7p,9t	0.155	Fails
	AWM	1,208 wf	**1.000**	LCS
Mind2Web	**Ours**	**8 st**	**1.000**	60×
	RPNI	476 st	0.970	Overfit
	EDSM	1 st	1.000	Degen.
	Alergia	**8 st**	**1.000**	Same
	HMM	**8 st**	**1.000**	Latent
	Heur. M	15p,29t	0.960	P:0.45
	Ind. M	23p,30t	**1.000**	P:0.49
	Alpha M	7p,9t	0.632	Poor
	AWM-all	72 wf	0.887	LCS

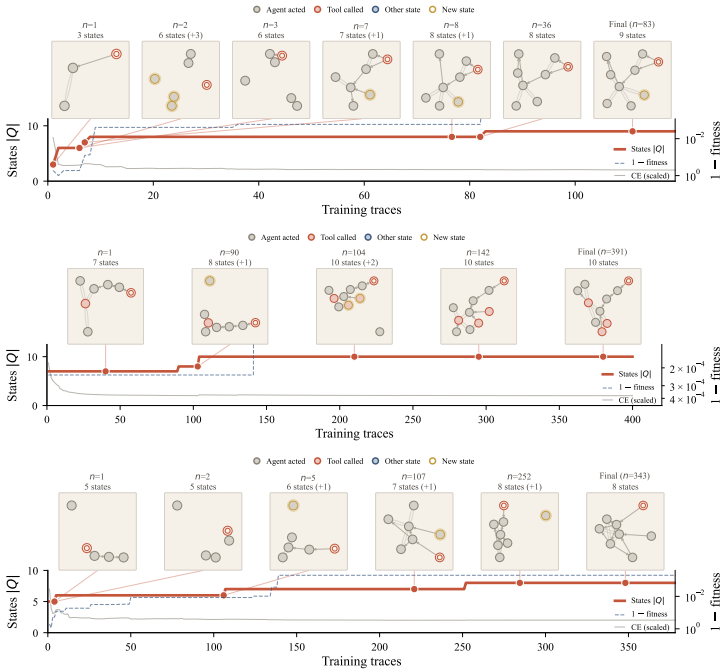

Figure 6. FSM evolution for Who_and_When, SWE-smith, and Mind2Web. State count (red), 1−fitness (blue dashed, reverse-log), CE (gray, scaled).

Small-alphabet datasets ($|\mathcal{A}|$=7–9). All three converge within 5–10% of training data and stabilize at 8–10 states. Alergia matches our state count exactly; HMM confirms the same structure. RPNI produces 476–11,631 states with degraded fitness ($\leq$0.984), demonstrating that modest trace corpora cause catastrophic overfitting without structural merging.

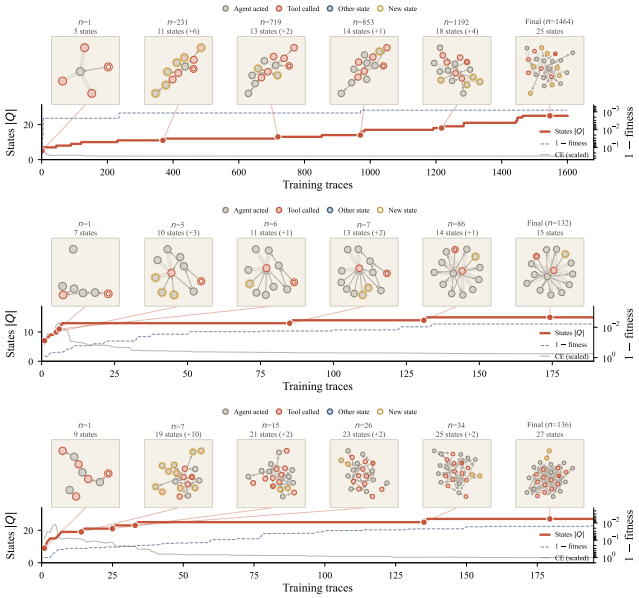

Figure 7. FSM evolution for SWE-agent, ATBench (safety), and OSWorld (desktop GUI). Lines: states (red), 1−fitness (blue dashed), CE (gray).

Table 9. **Full baseline comparison** (2/4).

Dataset	Method	$\lvert Q\rvert$/Size	Fit	Notes
SWE-agent	**Ours**	**25 st**	0.999	2,380×
	RPNI	59,510†	0.646	†
	EDSM	1 st	1.000	Degen.
	Alergia	35 st	0.999	1.4×
	HMM	**25 st**	**1.000**	Latent
	Heur. M	21p,59t	0.999	PN
	Ind. M	44p,70t	0.999	PN
	Alpha M	12p,24t	0.050	PN
	AWM	371 wf	0.978	LCS
ATBench	**Ours**	**15 st**	**1.000**	60×
	RPNI	899	0.984	
	EDSM	1 st	1.000	Degen.
	Alergia	15 st	**1.000**	1.0×
	HMM	**15 st**	**1.000**	Latent
OSWorld	**Ours**	**27 st**	**0.997**	1,416×
	RPNI	38,232	0.706	
	EDSM	1 st	1.000	Degen.
	Alergia	31 st	0.999	1.1×
	HMM	**27 st**	**1.000**	Latent

Coding agent ($|\mathcal{A}|$=25). SWE-agent shows the most gradual evolution: core search-edit-execute structure emerges by n=80, but rare commands (e.g., `deactivate`, `cd`) continue appearing until n=1,200.

Table 10. **Full baseline comparison** (3/4).

Dataset	Method	$\|Q\|$/Size	Fit	Notes
GUI-Odyssey	**Ours**	**7 st**	**1.000**	3,036×
	RPNI	21,255†	0.929	†
	EDSM	1 st	1.000	Degen.
	Alergia	24 st	0.999	3.4×
	HMM	**7 st**	**1.000**	Latent
	Ind. M	–	**1.000**	
	AWM	1,266 wf	0.996	LCS
WebArena	**Ours**	**25 st**	**1.000**	15×
	RPNI	382 st	**1.000**	15× larger
	EDSM	1 st	1.000	Degen.
	Alergia	149 st	**1.000**	6.0×
	HMM	**25 st**	**1.000**	Latent
AgentNet	**Ours**	**25 st**	**1.000**	2,500×
	RPNI	62,495†	0.743	†
	EDSM	1 st	1.000	Degen.
	Alergia	45 st	**1.000**	1.8×
	HMM	**25 st**	**1.000**	Latent
	Heur. M	35p,84t	0.987	PN
	Ind. M	18p,39t	**1.000**	PN
	Alpha M	174p,–t	0.297	PN
	AWM	–	0.975	LCS

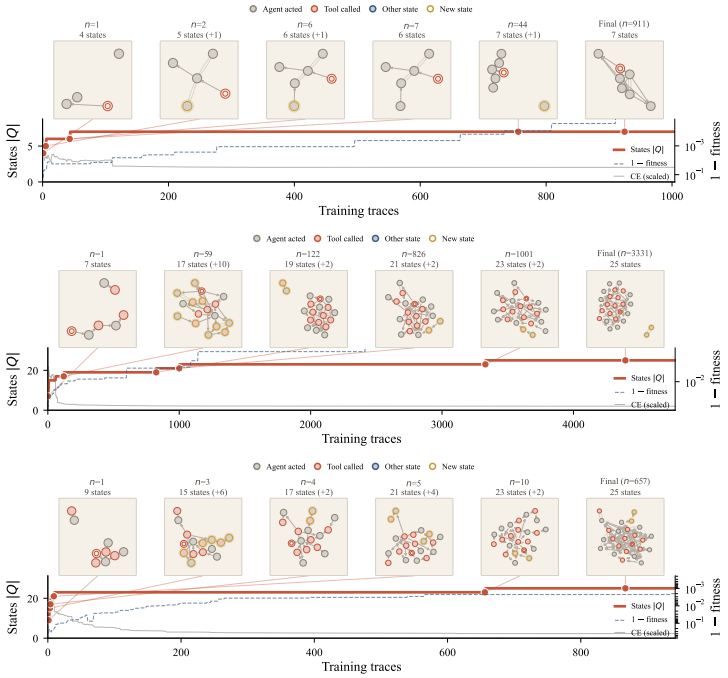

Figure 8. FSM evolution for GUI-Odyssey, WebArena, and AgentNet. Lines: states (red), 1−fitness (blue dashed), CE (gray).

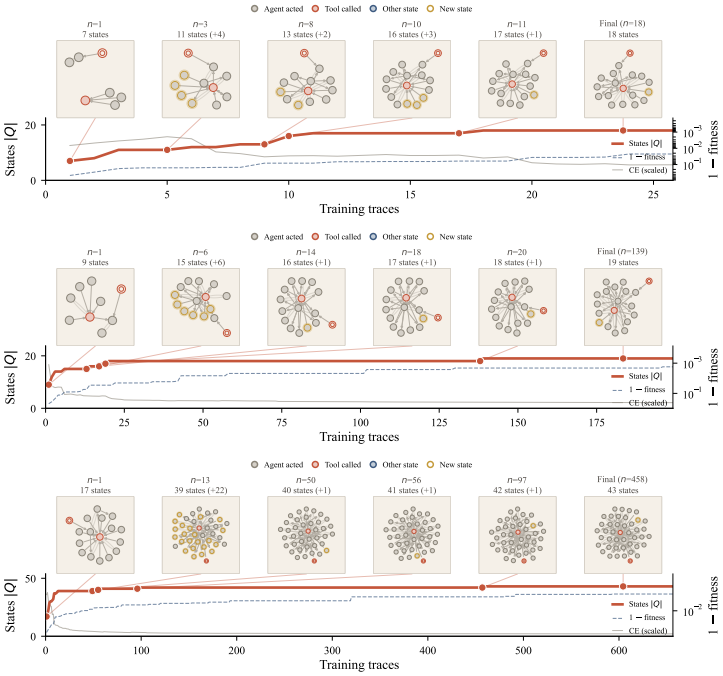

Figure 9. FSM evolution for tau2-bench (airline, retail, telecom). Multi-model (4 LLMs). Lines: states (red), 1−fitness (blue dashed), CE (gray).

Table 11. **Full baseline comparison** (4/4).

Dataset	Method	$\|Q\|$/Size	Fit	Notes
tau2-air	**Ours**	**18 st**	**1.000**	361×
	RPNI	6,506†	0.844	†
	EDSM	1 st	1.000	Degen.
	Alergia	23 st	0.999	1.3×
	HMM	**18 st**	**1.000**	Latent
tau2-ret	**Ours**	**19 st**	**1.000**	750×
	RPNI	14,249†	0.837	†
	EDSM	1 st	1.000	Degen.
	Alergia	25 st	**1.000**	1.3×
	HMM	**19 st**	**1.000**	Latent
tau2-tel	**Ours**	**43 st**	**1.000**	1,486×
	RPNI	63,897†	0.491	†
	EDSM	1 st	1.000	Degen.
	Alergia	75 st	0.999	1.7×
	HMM	**43 st**	**1.000**	Latent

Large-scale datasets (5,000–8,337 traces). Compression ratios peak here: GUI-Odyssey at 3,036× and AgentNet at 2,500×.

RPNI completely collapses on AgentNet (62,495 states, 0.743 fitness). WebArena is the exception: short traces ($\sim$8 steps) keep the prefix tree small (382 states), so RPNI achieves perfect fitness, though still 15$\times$ larger. Alergia diverges most on GUI-Odyssey (3.4$\times$ our state count), where the statistical merge criterion becomes overly conservative with 7,735 traces.

Multi-model benchmarks (4 LLMs per dataset). A single FSM achieves 1.000 fitness on all four models' traces, confirming model-invariant behavioral topology. tau2-bench telecom has the largest FSM (43 states, 42 tool types), reflecting the complex diagnostic workflow. The evolution figures show that all three tau2-bench datasets reach structural convergence despite pooling traces from GPT-4.1, Claude 3.7, GPT-4.1-mini, and o4-mini.

Alergia behavior. Alergia (Carrasco & Oncina, 1994) learns a probabilistic DFA (PDFA) from positive examples using a Hoeffding bound to decide state merges. On smaller datasets ($\leq$500 traces), Alergia produces state counts matching our method (8–12 states). On larger datasets, Alergia diverges: 35 states on SWE-agent (1.4$\times$ ours) and 24 on GUI-Odyssey (3.4$\times$ ours), because the statistical test becomes more conservative with more data, splitting states that share structure but differ in probability distributions. Our structural merging is agnostic to transition frequencies, producing deterministic FSMs that are smaller and faster to construct.

HMM behavior. HMM (Baum-Welch (Rabiner, 1989)) with the same number of hidden states as our FSM achieves comparable fitness on all twelve datasets. However, HMM states are latent (unlabeled), making the model non-interpretable: one cannot inspect which behavioral mode a state corresponds to or extract per-state features for downstream analysis. Our FSM states have explicit activity-labeled transitions.

EDSM behavior. EDSM (Lang et al., 1998) uses evidence-driven scoring to rank candidate merges. Without negative examples, the evidence score for every merge candidate is zero, so EDSM greedily merges all states into one. The resulting 1-state universal acceptor has perfect fitness (it accepts everything) but zero precision. This validates that our approach, which also uses only positive examples, achieves meaningful structure (7–43 states with high precision) rather than collapsing.

AWM behavior. AWM (Wang et al., 2024b) by default filters for successful traces only. On Who_and_When (all failures) and Mind2Web (no success labels), we use AWM-all which skips the success filter. AWM extracts linear workflows via longest common subsequence alignment; it cannot represent cycles or branching. Its coverage metric (0.887–1.000) is not directly comparable to replay fitness.

PM4Py precision. The "flower model" problem in process mining: miners that accept all possible orderings achieve high fitness but low precision. Our precision analysis (Table 18) confirms this: PM4Py miners achieve precision of 0.00–0.80 across datasets, while our FSM achieves near-zero random acceptance.

k-Tails behavior. k-Tails (Biermann & Feldman, 1972) merges states sharing identical k-length futures. Table 12 reports results for $k \in \{1, 2, 3\}$. At k=1, k-Tails produces 1.4–10$\times$ more states than our method with lower test fitness (0.54–1.00 vs. $\geq$0.997). At $k\geq$2, state counts explode to hundreds or thousands, with timeouts on large datasets (AgentNet, tau2-bench telecom). No value of k simultaneously matches our compression and fitness, illustrating why hyperparameter-free structural merging is preferable.

Table 12. **k-Tails results.** $|Q|$: states. Fit: test fitness. TO: timeout (300s). Our method requires no hyperparameter.

	Ours		k-Tails (k=1)		k-Tails (k=2)	
Dataset	$\|Q\|$	Fit	$\|Q\|$	Fit	$\|Q\|$	Fit
SWE-smith	**10**	1.000	22	1.000	53	1.000
SWE-agent	**25**	0.999	74	0.996	332	0.993
WebArena	**25**	1.000	30	0.743	172	0.743
AgentNet	**25**	1.000	33	0.806	TO	
tau2-bench (air)	**18**	1.000	37	0.973	731	0.773
tau2-bench (ret)	**19**	1.000	46	0.997	874	0.888
tau2-bench (tel)	**43**	1.000	210	0.930	TO	
Who_and_When	**9**	1.000	14	0.541	20	0.541
Mind2Web	**8**	1.000	34	0.965	95	0.905
GUI-Odyssey	**7**	1.000	73	0.964	759	0.939

D.3. Implementation Details

All experiments use a single fixed seed for train/test splitting via seeded Fisher-Yates shuffle. The 80/20 split is applied consistently across all datasets and baselines.

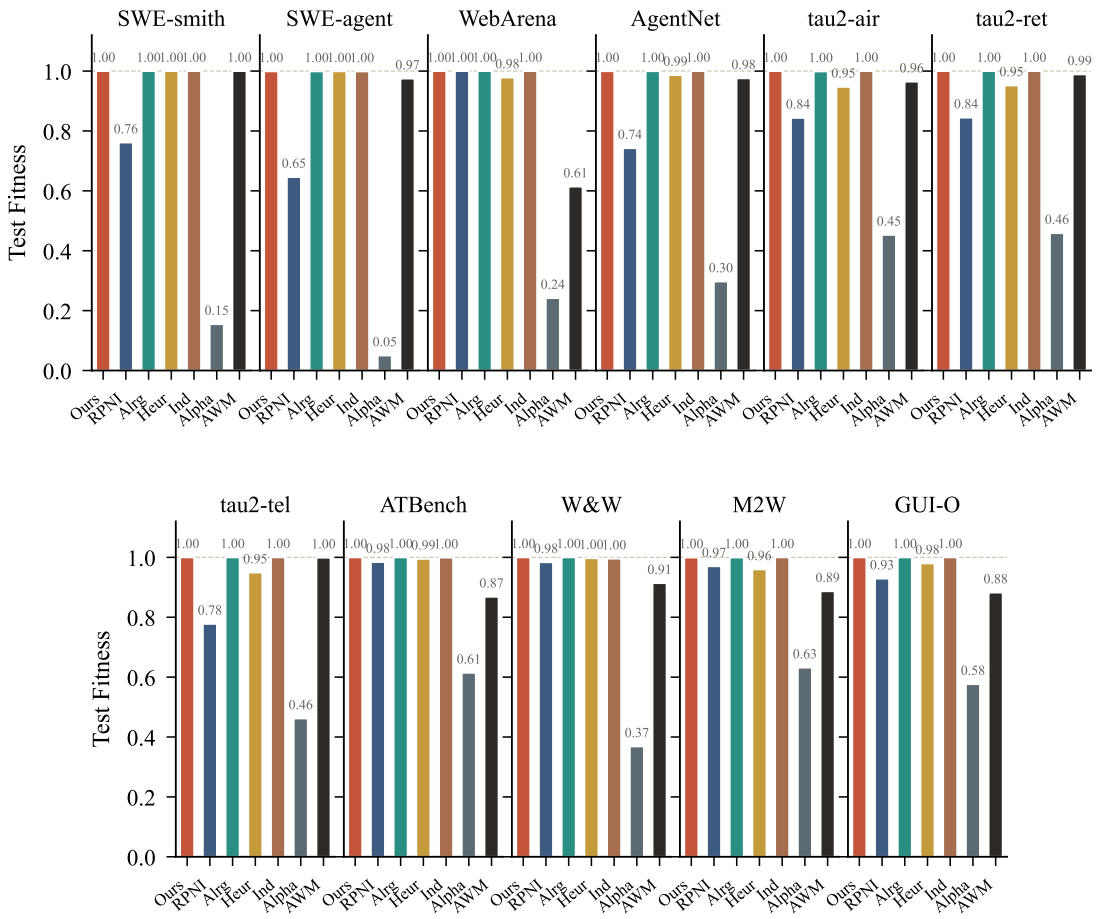

Figure 10. **Per-dataset test fitness** across all methods including Alergia. Our FSM achieves $\geq$0.997 fitness on all panels. Process mining baselines (Heur., Ind.) achieve competitive fitness but with low precision (Table 18).

RPNI. TypeScript implementation with a 120-second timeout for the merge phase. Without negative examples, RPNI cannot safely merge states, resulting in near-complete prefix trees. The timeout is necessary for SWE-agent (59,510 states), GUI-Odyssey (21,255 states), and SWE-smith (11,631 states).

EDSM. AALpy (Muškardin et al., 2022) version 1.3.4. Without negative examples, evidence-driven scoring produces no merge candidates, collapsing to a 1-state universal acceptor.

Alergia. AALpy implementation with Markov Chain automaton type and $\varepsilon = 0.005$ (Hoeffding bound parameter). Lower ε produces more merging (fewer states). Alergia learns from positive examples only using statistical compatibility testing.

HMM. Pure NumPy implementation of Baum-Welch EM (Rabiner, 1989) with 50 iterations and scaled forward-backward to prevent numerical underflow. Number of hidden states set to match our FSM's state count for fair comparison. Replay fitness is computed via Viterbi decoding: a symbol is consumed if the decoded state has nonzero emission probability ($> 10^{-6}$). Effective transitions are counted as entries in the transition matrix with probability > 0.01.

PM4Py. Traces are converted to XES event log format. We run Alpha Miner, Inductive Miner, and Heuristic Miner with default parameters. Fitness and precision are computed via PM4Py's conformance checking.

AWM. We re-implement workflow extraction from Wang et al. (2024b). Coverage is measured via longest common subsequence alignment.

Failure prediction. Feature selection uses L1-regularized logistic regression ($C = 0.1$) on the training set to select features with non-zero weights. Final models use class-weighted loss (inverse frequency). Cross-validation uses stratified 5-fold with 10 repetitions, evaluated on training data only (no test data in CV folds).

Runtime. FSM construction completes in <1 second for all datasets. RPNI requires up to 120 seconds (timeout). PM4Py baselines complete in 5–30 seconds.

D.4. Runtime Details

Construction time scales linearly with corpus size (Table 13). RPNI's quadratic merge loop ($O(n^2)$ state pairs) causes timeouts at 120 s on most datasets (only Mind2Web completes within the limit). On SWE-agent, RPNI produces 59,510 states vs. our 25. Per-trace replay is nearly instantaneous (<0.015 ms), making FSM-based monitoring practical for real-time agent systems (Figure 11).

Table 13. **Runtime comparison.** Build: FSM construction. Replay/tr: per-trace replay latency.

	Ours (ms)		RPNI (ms)		
Dataset	Build	Replay/tr	Build	States	Speedup
Mind2Web	1.0	0.003	6,902	476	6,579×
Who_and_When	4.6	0.005	30,005†	971	6,502×
SWE-smith	30.4	0.015	30,253†	11,631	994×
SWE-agent	109.8	0.008	36,008†	59,510	328×

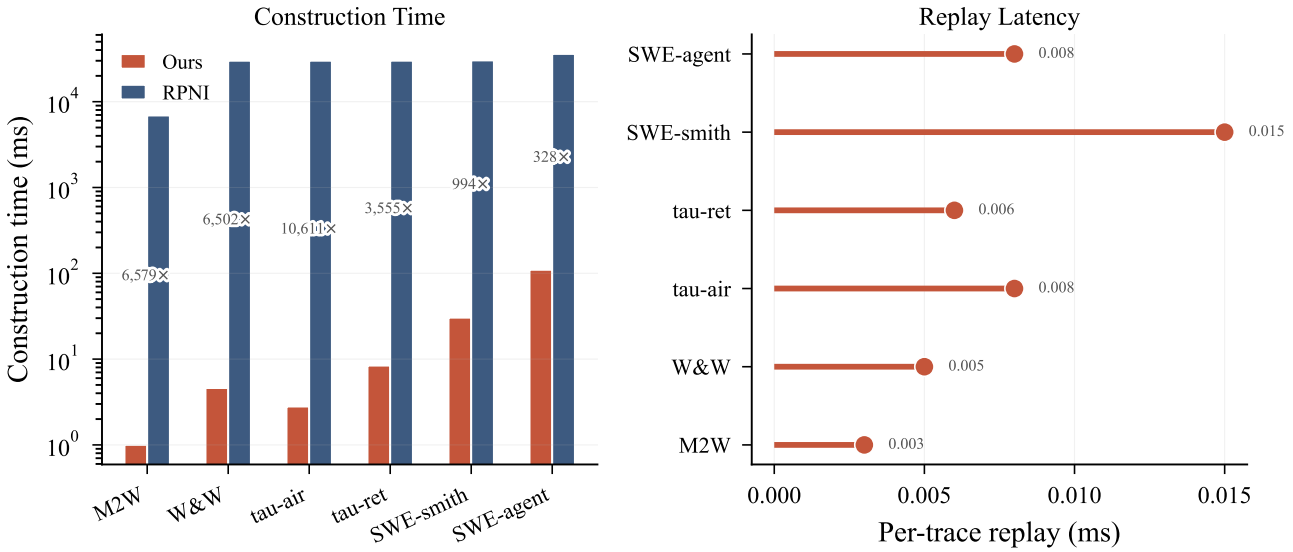

Figure 11. **Runtime comparison.** Left: FSM construction time (1–110 ms) vs. RPNI (7,000–36,000 ms) with speedup ratios annotated. Right: per-trace replay latency (0.003–0.015 ms), enabling real-time monitoring.

D.5. Case Study: FSM Visualizations

We present FSM visualizations for representative datasets to illustrate the behavioral structure our method recovers.

D.5.1. Customer Service Agent (tau2-bench airline)

Figure 5 illustrates the tau2-bench airline FSM at role-level granularity (6 states, $|\mathcal{A}|$=5). The full tool-level FSM (18 states, $|\mathcal{A}|$=17, Table 1) decomposes further into per-tool states. At role level, the FSM reveals two distinct behavioral loops:

1. **Tool-call loop** (`assistant:tool_call` ↔ `tool:text`): The agent queries customer databases (get_reservation, search_flights) and receives structured responses. This loop executes 2–8 times per trace.
2. **Conversation loop** (`assistant:text` → `user:text`): The agent communicates results to the user and receives follow-up requests. Failed traces show more conversation turns (mean 4.2 vs 2.8 for successes), suggesting the agent struggles to complete the task.

The FSM makes these patterns structurally visible: the tool-call loop appears as a tight 2-state cycle, while the conversation loop passes through the user state. This decomposition enables per-state analysis (e.g., error rates in the tool state, message lengths in the assistant state) that would be obscured in raw trace analysis.

D.5.2. CODING AGENT (SWE-SMITH)

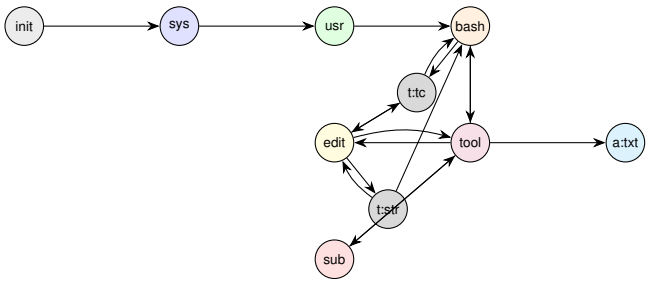

Figure 12. **SWE-smith FSM** (10 states, 17 transitions). The core cycle is `bash`→`tool`→`edit`→`tool`→`bash`: agents execute commands, inspect results, make edits, and repeat. The `submit` state is a terminal action reached after successful editing.

The SWE-smith FSM (Figure 12) captures the coding agent's workflow across 10 states. The dominant pattern is the `bash`→`tool`→`str_replace_editor`→`tool` cycle: the agent executes bash commands, inspects output, applies code edits, and verifies results. Failed traces (123/500) show higher visit counts in the edit-tool loop (mean 8.4 vs 5.1 for successes) and elevated error rates in the `str_replace_editor` state (0.31 vs 0.12), suggesting repeated failed edit attempts.

D.5.3. CODING AGENT AT SCALE (SWE-AGENT)

The SWE-agent FSM has 25 states derived from 2,000 traces with 175 raw commands grouped into 7 categories. The larger state space (compared to SWE-smith's 10) reflects the richer command vocabulary and longer traces (mean 87 steps vs 23). Key structural features include:

- A **search-navigate cycle** (`search` ↔ `navigate`): agents find relevant files and navigate to specific locations.
- An **edit-execute cycle** (`edit` ↔ `execute`): agents modify code and run tests.
- A **submit terminal**: successful traces reach the `submit` state, the strongest failure-prediction feature.

The 2,380× compression (59,510 RPNI states → 25 FSM states) demonstrates that even agents with complex command vocabularies exhibit a small number of distinct behavioral modes when commands are grouped by semantic function.

E. Failure Analysis

E.1. Structural Divergence of Success vs. Failure

Building separate FSMs for successful and failed traces reveals qualitative structural differences. On SWE-agent, the success FSM uses only 9 of 25 states with 16 transitions (focused path: search-edit-execute-submit), while the failure FSM spans all 25 states with 60 transitions (chaotic exploration). The transition Jaccard similarity is 0.206, indicating largely disjoint behavioral structures; all 16 failure-only states correspond to rare tool output parsing variants (e.g., `user:tool:of`, `user:versioneer`) that successful traces never encounter. On tau2-bench, both classes produce structurally identical FSMs (discrimination gap $= 0$), confirming that failure on constrained API-calling tasks manifests purely in transition frequencies rather than novel states. Failed traces are consistently longer across all datasets (+29 steps on SWE-agent, +18 on SWE-smith) but visit the same or fewer unique states, suggesting that failure manifests as cycling through familiar states rather than exploring new ones.

These structural differences enable simple monitoring rules without ML models. On SWE-agent, "cycle rate > 0.885" achieves 95.6% precision (§G.3).

E.2. Failure Prediction Numerical Results

E.3. Discriminative Quotient (FSM-D)

The standard Myhill–Nerode quotient (Theorem 3) merges states with identical outgoing transition structure on the prefix language $L_{\mathcal{P}}$. We additionally explore a *discriminative* variant that conditions on the trace label: at training time, partition traces into success and failure subsets, and for each activity a compute the outgoing transition distributions $P^+(\cdot \mid a)$ and $P^-(\cdot \mid a)$. The Kullback–Leibler divergence $\mathrm{KL}(P^+ \| P^-)$ at state a measures how much the success distribution deviates from the failure distribution; large values indicate that the activity is a behavioral choke point separating the two classes.

Table 14. **Failure prediction AUROC** (held-out). Main: full GBT pipeline. FSM-D ablation: training-free LR, Len+Ent vs. +per-state KL (Δ = gain). SWE-smith synthetic; tau2-bench aggregates 4 models.

		Main	FSM-D ablation		
Dataset	$\lvert\mathcal{A}\rvert$	Holdout	Len+Ent	FSM-D	Δ
tau2-bench (tel)	42	**0.941**	0.725	**0.857**	**+0.132**
WebArena	24	0.903	0.718	**0.773**	+0.056
AgentNet	24	0.890	0.724	0.724	+0.000
ATBench	14	0.894	0.555	**0.703**	**+0.147**
tau2-bench (air)	17	0.864	0.625	**0.777**	**+0.152**
SWE-agent	24	0.799	0.690	0.689	−0.001
tau2-bench (ret)	18	0.779	0.596	**0.660**	+0.064
OSWorld	26	0.774	0.770	**0.778**	+0.008
SWE-smith	9	0.703	0.695	0.702	+0.006

The discriminative features alone (no learned classifier beyond logistic regression on 10 inputs) lift held-out AUROC over a length+entropy baseline by +0.06 to +0.15 on five of nine datasets, with the largest gains on datasets whose activities have the most discriminative outgoing distributions (mean KL on tau2-bench airline: 0.014; ATBench: 0.020; OSWorld: 0.136 with 3 of 26 activities at KL>0.3). Datasets with near-uniform outgoing distributions across success and failure (SWE-agent: mean KL 0.001; AgentNet: 0.000) show no FSM-D gain, consistent with the discriminative signal being a property of the agent's behavioral divergence rather than a universal lift. The FSM-D variant is a complement to the cross-entropy anomaly features used in the main results: where outgoing distributions diverge, FSM-D contributes a training-free signal; where they do not, the main pipeline's per-state visit features and the trace cross-entropy carry the predictive load.

E.4. Alergia Features under Matched Pipeline

To check that the downstream gain is not just an artefact of state-count differences, we apply the identical feature extraction and classifier pipeline to Alergia-extracted FSMs (1.0–6.0× more states than ours; §4.2). With logistic regression on 34–175 per-state features (visit frequency, mean/max message length, error rate, temporal entropy), 10×5-fold CV on training data, our FSM beats Alergia on 8 of 9 labeled datasets at the time of this comparison (tying on AgentNet; Table 15), with the largest margin (+0.18) on tau-bench retail. The additional Alergia states from statistical merging dilute per-state observation counts rather than improving them, consistent with the $O(1/\sqrt{n_q})$ estimator bound (Proposition 6).

Table 15. **Failure prediction with Alergia FSMs under matched pipeline.** Identical features, classifier, and CV protocol; FSM source varies. CV: 10×5-fold; Holdout: held-out test AUROC.

	CV AUROC		Holdout AUROC	
Dataset	Ours	Alergia	Ours	Alergia
tau2-bench (tel)	**0.923**	0.748	**0.915**	0.752
WebArena	**0.864**	0.844	**0.888**	0.868
AgentNet	0.871	0.871	0.872	0.872
SWE-agent	**0.790**	0.709	**0.805**	0.714
tau2-bench (air)	**0.792**	0.664	**0.826**	0.764
tau-bench (ret)	**0.789**	0.606	**0.666**	0.498
tau-bench (air)	**0.758**	0.651	**0.841**	0.810
tau2-bench (ret)	**0.713**	0.576	**0.743**	0.575
SWE-smith	**0.685**	0.653	0.673	0.683

The current main-text headline numbers (§4.3; up to 0.941 holdout) use a richer cross-entropy anomaly feature set with a gradient-boosted classifier; we expect the same direction of comparison (ours > Alergia) under that pipeline because the bottleneck for Alergia features is per-state observation count, not feature engineering.

E.5. Failure Prediction Feature Analysis

Table 16 presents the top 10 features by absolute L1-regularized logistic regression weight for each labeled dataset.

Interpretable patterns. The features reveal consistent failure signals across datasets:

- **Not reaching terminal states** (SWE-agent: submit state weight +0.29): failed agents get stuck in intermediate loops.
- **Elevated error rates** (SWE-smith: editor error rate −0.20): failed agents encounter more errors per state.
- **Verbose responses** (tau2-bench airline: avgMsgLen:assistant): longer responses correlate with task difficulty and failure.
- **Temporal entropy** (SWE-smith: lateHalfEntropy −0.19): chaotic second-half behavior indicates the agent is struggling.

Table 16. **Top failure prediction features** by L1-regularized weight. Positive weight = associated with success; negative = associated with failure.

Dataset	Feature	Weight	Interpretation
SWE-agent	terminal:submit:text	+0.292	Reaching submit state
	avgMsgLen:edit:text	−0.231	Long edit messages
	uniqueStatesVisited	+0.217	Exploring more states
	earlyHalfEntropy	+0.215	Diverse early behavior
	errorRate:edit:text	−0.170	Edit errors
SWE-smith	maxMsgLen:tool:text	−0.290	Long tool output
	avgMsgLen:str_replace_editor	−0.280	Long edit commands
	errorRate:str_replace_editor	−0.200	Edit errors
	lateHalfEntropy	−0.190	Chaotic late behavior
	selfLoopDrift	+0.165	Increasing self-loops

These patterns are only visible through the FSM's state decomposition: raw trace-level features (total message length, total error count) do not capture which behavioral mode produced the errors. Notably, the two most discriminative feature families (per-state error rates and temporal entropy splits) appear in the top 5 across labeled datasets despite their different domains (coding, API-calling), suggesting that FSM-conditioned features transfer well across agent architectures.

Table 17. Failure prediction: neural models on sequence vs. FSM features. CV AUROC (10×5-fold). Bold: FSM>Seq for same model.

	MLP		GRU		Transformer	
Dataset	Seq	FSM	Seq	FSM	Seq	FSM
SWE-smith	0.663	**0.699**	0.683	**0.700**	0.649	**0.665**
SWE-agent	0.782	**0.790**	0.762	**0.793**	0.779	0.751
tau2-tel	0.962	**0.969**	0.958	**0.970**	0.964	**0.967**
tau2-air	0.816	**0.839**	0.793	**0.815**	0.819	**0.845**
tau2-ret	0.747	**0.800**	0.698	**0.789**	0.755	**0.798**
WebArena	0.848	**0.861**	0.818	**0.835**	0.860	**0.868**
AgentNet	0.878	**0.920**	0.877	**0.917**	0.876	**0.917**
FSM wins	7/7		7/7		6/7	

E.6. Neural Baselines: Sequence vs. FSM Features

Table 17 compares neural models (MLP, GRU (Cho et al., 2014), Transformer (Vaswani et al., 2017)) trained on sequence features (bag-of-activities + length + entropy) vs. FSM per-state features. All models use embed/hidden=64/128, class weighting, early stopping (patience 10), and the same 10×5-fold CV protocol. Across this appendix, we retain SWE-smith's numbers even when they are the weakest per-dataset value: the dataset is synthetic and below the sample threshold of Proposition 6, so its weaker AUROCs (0.62–0.72 across sub-ablations) are both expected and consistent with the main-text framing that FSM features generalize more strongly to real LLM-deployment traces.

FSM features improve every neural model on every dataset for MLP and GRU (7/7), and 6/7 for Transformer. The largest gains appear on tau2-bench retail (+5.3pp MLP, +9.1pp GRU) and AgentNet (+4.2pp MLP, +4.0pp GRU), where the FSM's per-state decomposition localizes behavioral differences that aggregate statistics miss. The Transformer loss on SWE-agent (-2.8pp) occurs where the small alphabet (24 symbols) limits the FSM's decomposition advantage. The improvement is consistent regardless of dataset size (500–8,337 traces): even on WebArena (8,337 traces), FSM features provide +1.3pp (MLP), +1.7pp (GRU), and +0.8pp (Transformer), demonstrating that the FSM captures structural information beyond what large-sample statistics recover.

E.7. Precision Analysis

Table 18. **Precision analysis.** Rand./Perm.: acceptance rate (lower is better).

	Ours		RPNI	
Dataset	Rand.	Perm.	Rand.	Perm.
SWE-smith	0.0%	0.1%	0.0%	0.0%
SWE-agent	0.0%	0.05%	0.0%	1.6%
WebArena	0.0%	0.02%	0.2%	75.3%
AgentNet	0.0%	0.02%	0.2%	0.9%
tau2-bench (air)	0.0%	0.0%	0.0%	11.5%
tau2-bench (ret)	0.0%	0.0%	0.0%	3.1%
tau2-bench (tel)	0.0%	0.0%	0.0%	0.0%

Our FSM rejects 100% of random traces and ≥99.9% of permuted traces on all eight labeled real-trace datasets (Table 18). RPNI, despite its large state space, accepts 75% of permuted traces on WebArena and 0–11.5% on tau2-bench, since its prefix tree memorizes observed prefixes but does not constrain orderings and thus admits structurally invalid permutations that our FSM rejects.

Adversarial trace rejection. Beyond random and permuted traces, we test whether the FSM rejects *plausible* mutations that preserve local structure: single-symbol substitution (replace one activity with another symbol), insertion, deletion, adjacent swap, and suffix shuffle (shuffle the last 30% of the trace). Table 19 reports rejection rates (fitness < 1.0) across five mutations per test trace.

The FSM rejects 90–100% of insertions and adjacent swaps on all datasets, confirming that it captures sequential ordering constraints beyond symbol membership. Substitution rejection (77–100%)

shows that most single-activity changes violate the learned transition structure. Deletion is weakest on tau2-bench datasets (61–80%) because shorter traces are more likely to remain valid prefixes. These results demonstrate that the compact FSM imposes tight structural constraints: even single-symbol perturbations are detected, because the transition function encodes which activity can follow which, not merely which activities are valid. For example, on tau2-bench airline, swapping `user:text`↔`assistant:tool:get_reservation_details` at position 3 is immediately rejected: after state `assistant:text`, the only valid transition is `user:text` (the user must respond before the agent can call a tool). On SWE-agent, swapping `navigate:text`↔`user:text` fails because self-transitions back to `user:text` are not in the FSM's transition table. These rejections reflect genuine turn-taking and tool-invocation constraints that the FSM learns from data.

E.8. Process Mining Precision Details

Table 20 reports per-miner fitness and precision for the three PM4Py baselines across all twelve datasets, spanning coding, web, GUI, desktop, and API domains. The Alpha Miner fails on all datasets (fitness 0.05–0.63) because it cannot handle noise or skip patterns. The Heuristic Miner achieves 0.95–1.00 fitness but precision 0.20–0.80 (mean 0.45); the Inductive Miner achieves perfect fitness on 8/10 datasets but even lower precision (0.10–0.46, mean 0.25). The highest precision (0.80, tau2-bench telecom) occurs on the most constrained workflow. Fitness alone is misleading here; only precision separates genuine structure from over-general acceptors.

Table 19. Adversarial trace rejection rate (%, ↑). Five mutations per trace.

Dataset	Subst.	Insert	Delete	Swap	Suffix
SWE-smith	81	100	98	100	100
SWE-agent	87	100	97	100	97
tau2-tel	77	96	61	93	100
tau2-air	78	90	77	93	97
tau2-ret	79	92	80	95	100
WebArena	100	96	84	100	68
AgentNet	100	100	97	100	99

Table 20. **PM4Py miner results.** Fit: replay fitness. Prec: precision from conformance checking. p/t: Petri net places/transitions.

	Alpha Miner			Heuristic Miner			Inductive Miner		
Dataset	p/t	Fit	Prec	p/t	Fit	Prec	p/t	Fit	Prec
Who_and_When	3/8	0.37	0.19	10/23	1.00	0.31	16/24	1.00	0.27
SWE-smith	7/9	0.15	0.20	13/22	1.00	0.36	25/33	1.00	0.23
Mind2Web	2/7	0.63	0.29	15/29	0.96	0.45	23/30	1.00	0.32
SWE-agent	12/24	0.05	0.00	21/59	1.00	0.20	44/70	1.00	0.13
tau2-bench (air)	3/17	0.45	0.24	15/37	0.95	0.20	63/93	1.00	0.17
tau2-bench (ret)	3/18	0.46	0.23	12/32	0.95	0.21	36/58	1.00	0.14
tau2-bench (tel)	4/4	0.46	0.25	10/13	0.95	**0.80**	15/19	1.00	0.42
GUI-Odyssey	2/6	0.58	0.33	12/22	0.98	**0.64**	21/27	1.00	0.46
WebArena	23/24	0.24	0.11	35/71	0.98	0.55	17/42	1.00	0.10
AgentNet	174/24	0.30	0.00	35/84	0.99	0.45	18/39	1.00	0.13

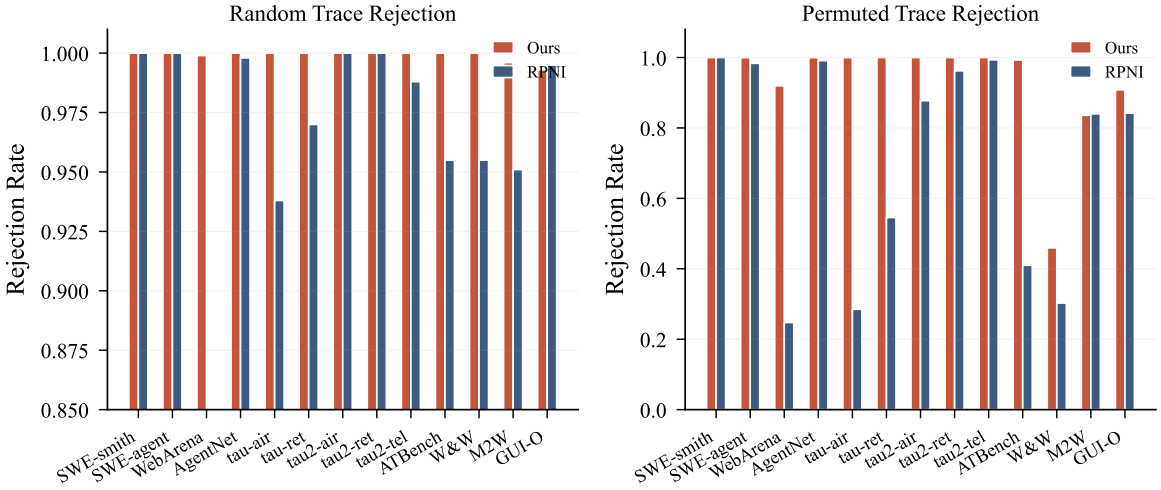

Figure 13. **Random and permuted trace rejection** rates. Our FSM achieves near-100% rejection across most datasets, while RPNI shows poor permuted rejection on WebArena.

See Appendix H.4 for additional failure mode characterization including cascade analysis and early divergence detection.

F. Convergence and Structural Properties

F.1. Convergence Details

Fitness converges rapidly as training traces are added: on all datasets, 0.99 fitness is reached using 5–15% of training data (Figures 1, 14). On SWE-agent (2,000 traces), fitness reaches 0.99 at 240 traces (15%), though the state space grows to 25 as rare command patterns appear. Because the construction is deterministic (Theorem 3), a fixed corpus yields a unique FSM; across random splits our state counts stay within a few states of the full-data value (rare activities present in only some samples account for the residual), whereas RPNI state counts vary by 2–10% (Appendix F.2).

Table 21. **Convergence and generalization.** Left: fraction at which fitness reaches 0.99. Right: train−test fitness gap at increasing fractions (all within ±0.003). Convergence behavior is similar across extraction levels (Appendix G.2).

	Convergence				Gen. gap (train−test)			
Dataset	Train	0.99 at	Frac	$\|Q\|$	10%	30%	60%	100%
Who_and_When	147	8	5%	9	+0.002	+0.001	0	0
SWE-smith	400	40	10%	10	−0.002	−0.001	0	−0.001
Mind2Web	400	20	5%	8	−0.003	−0.001	0	0
SWE-agent	1,600	240	15%	25	−0.002	−0.001	−0.001	+0.001

Table 21 reports the training fraction at which fitness first reaches 0.99, the final state count, and the generalization gap (train − test fitness) at increasing training fractions.

Smaller datasets converge at 5% of training data; SWE-agent requires 15% to capture rare commands. All generalization gaps are within ±0.003, confirming zero overfitting. Slightly negative gaps arise because training sets include rare transitions that reduce average fitness.

F.2. Stability and SCC Structure

The construction is deterministic and hyperparameter-free (Theorem 3): a fixed training corpus yields a unique FSM, so re-extraction is exactly reproducible. Across random train/test splits our FSM state counts stay within a few states of the full-data value; the residual reflects rare activities (e.g. SWE-agent's late-appearing commands) that occur in only some 80% samples, whereas RPNI state counts vary by 2–10% (hundreds to thousands of states). Structurally, every FSM decomposes into one large strongly-connected component (the behavioral core) and a short prefix; the condensation DAG is shallow (depth 2–4), with init→setup→core representing the universal agent lifecycle. SCC-based features achieve AUROC 0.60–0.66: failures tend to become trapped in the core loop rather than progressing to terminal states.

F.3. Entropy Rate Analysis

We estimate the conditional entropy $H(X_n \mid X_{n-1}, \ldots, X_{n-k})$ of the activity sequence at increasing orders k to characterize the sequential structure of agent traces. Table 22 shows the entropy rate convergence across all datasets.

Table 22. **Conditional entropy** (bits) by context order. The large drop from order 0→1 and convergence by order 2–3 shows strong sequential regularity.

Dataset	k=0	k=1	k=2	k=3	Drop 0→1
SWE-agent	2.16	1.06	0.80	0.79	51%
SWE-smith	1.98	0.63	0.55	0.55	68%
Mind2Web	1.69	0.74	0.79	0.68	56%

On all datasets, entropy drops 51–68% from order 0 to 1 (Table 22), and order-2 vs. order-3 estimates differ by <0.02 bits on 2/3 datasets. This shows that agent behavior is predominantly determined by the immediately preceding action, explaining why compact FSMs achieve near-perfect fitness. SWE-agent shows the most residual higher-order structure (0.80→0.79 bits). Success vs. failure entropy rates are nearly identical (within 0.03 bits), indicating shared behavioral topology with differences in transition frequencies.

F.4. Feature Redundancy Analysis

We compute pairwise Pearson correlations among 14 structural features to identify redundancy. Features with $|r| > 0.8$ are grouped into clusters, and a greedy selection retains the feature with highest individual AUROC from each cluster.

Across labeled datasets, 14 features consistently collapse into 5–8 non-redundant clusters. The largest cluster (5–7 features) contains {traceLen, logLength, entropy, transitionDiversity, maxConsecRatio, recurrenceRate}, reflecting that these are all

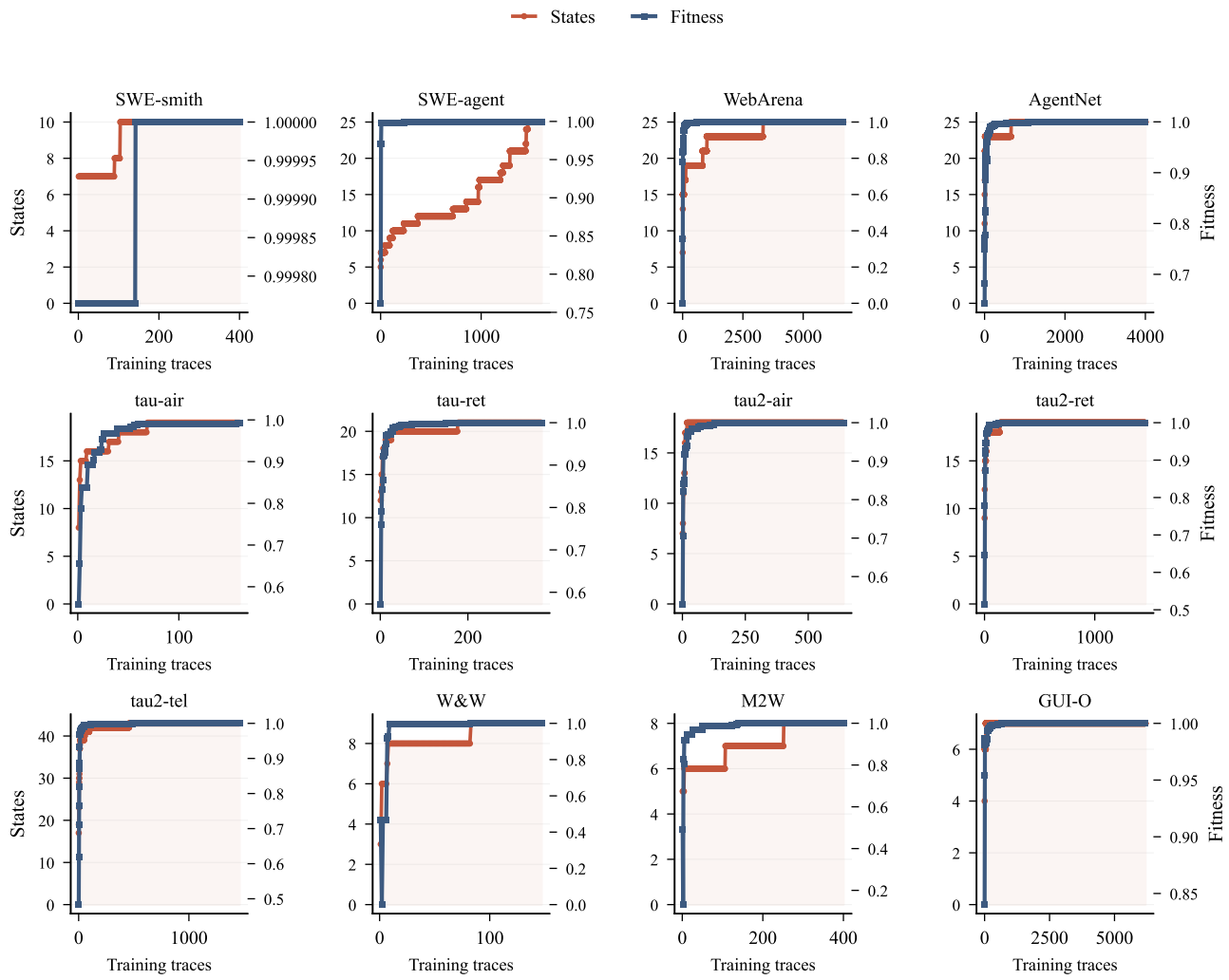

Figure 14. **FSM convergence:** test fitness (right axis) and state count (left axis) as training traces are added. Fitness converges rapidly; the state space stabilizes later as rare patterns appear.

manifestations of trace length and its correlates. Removing all redundant features and retaining only one representative per cluster yields combined AUROC of 0.655 (SWE-smith) and 0.713 (SWE-agent). On SWE-agent, the non-redundant subset achieves 90% of the full-feature AUROC, confirming that the failure prediction signal is genuine and concentrated in a small number of independent behavioral dimensions: cycle structure, state entropy, and terminal state reachability.

G. Prediction and Robustness

G.1. Early Prediction Details

Table 23 presents early prediction AUROC at each trace completion fraction. FSM features from partial traces at 50% completion already achieve 92% of full-trace AUROC on SWE-agent (0.722 vs. 0.784, CV; holdout values in Table 23: 0.726 vs. 0.773), demonstrating that the behavioral signal emerges early in execution. The FSM advantage over aggregate baselines grows with behavioral complexity: on SWE-agent, FSM features outperform by +0.026 to +0.257 at all stages beyond 25%, while on SWE-smith the simpler FSM offers no advantage.

On SWE-agent, FSM features outperform baselines by +0.026 to +0.257 at all stages.

Table 23. **Early prediction** AUROC (holdout) by trace completion fraction. FSM vs. raw-statistic baseline.

Fraction	SWE-agent		SWE-smith	
	FSM	Base	FSM	Base
10%	**0.657**	0.400	**0.722**	0.416
25%	**0.656**	0.456	0.687	**0.693**
50%	**0.726**	0.677	**0.681**	0.668
75%	**0.726**	0.672	**0.688**	0.685
100%	**0.773**	0.753	0.682	**0.692**

On SWE-smith, the simpler FSM offers no consistent advantage, suggesting that the FSM benefit scales with behavioral complexity.

G.2. Activity Granularity Robustness

Table 24. **Activity granularity** robustness. $|\mathcal{A}|$: alphabet size. Fit: test fitness. AUROC: failure prediction (entropy-based).

Dataset	Level	$\|\mathcal{A}\|$	Fit	AUROC
SWE-smith	role-only	4	1.000	0.688
	role-type	9	1.000	0.692
	role-action	9	1.000	0.692
	tool-only	9	1.000	0.692
SWE-agent	role-only	2	1.000	0.663
	role-type	18	0.999	0.659
	role-action	18	0.999	0.659
	tool-only	18	0.999	0.659

The extraction function ϕ is a design choice. We test four granularity levels: **role-only** ($|\mathcal{A}|$=2–4, distinguishing only message roles), **role-type** (role + content type), **role-action** (role + action label), and **tool-only** (tool function names). Fitness remains $\geq$0.999 across all levels and datasets, confirming that the FSM structure is robust to extraction granularity. Failure prediction AUROC varies by less than 0.03 between the coarsest (role-only) and finest (tool-only) levels. Even a 2–4 symbol alphabet preserves the predictive signal, because the behavioral topology (cycle structure, branching patterns) is invariant to label granularity.

G.3. Monitoring Rules

Table 25 presents the best single-feature monitoring rules per dataset. These rules require only FSM replay (0.006 ms/trace) with no model training. On SWE-agent, the cycle-rate rule achieves 95.6% precision (near-zero false alarm rate) at the cost of lower recall (45.1%). SWE-smith shows weaker monitoring rules, consistent with its lower failure prediction AUROC in the ML-based approach (Table 14).

G.4. Out-of-Distribution Detection

Table 25. Best monitoring rules by F1 score. All rules use a single FSM-derived feature with a fixed threshold.

Dataset	Rule	Prec.	Recall	F1
SWE-agent	cycle-rate > 0.885	**0.956**	0.451	0.613
SWE-smith	cycle-rate > 0.878	0.267	0.750	0.393

Replaying traces from one dataset through another's FSM produces low fitness for structurally distinct pairs (0.00–0.51 vs. $\geq$0.997 in-distribution), yielding AUROC 1.000. The exception is tau2-bench airline$\leftrightarrow$retail, which share a schema and replay near-1.0; cross-dataset fitness therefore spans 0.00–1.000 overall. This detection is not an artifact of alphabet mismatch: applying 10% random activity substitution within the same alphabet drops fitness to 0.63–0.82 across datasets, yielding AUROC $\geq$0.917 for distinguishing clean from perturbed traces. Within a single dataset, success and failure traces exhibit structural divergence: on SWE-agent, the success-only FSM has 9 states (focused: search-edit-submit) while the failure-only FSM spans all 25 states (chaotic exploration with rare tool variants). The FSM captures behavioral topology, not vocabulary.

G.5. Probabilistic Baseline Comparison

Table 26 compares FSM per-state features against probabilistic baselines for failure prediction: Markov chain cross-entropy (CE) against success/failure transition matrices, likelihood ratio (LR) scoring, and discriminative n-gram frequency analysis.

Table 26. Failure prediction: FSM features vs. probabilistic baselines (holdout AUROC).

Dataset	FSM feat.	Trans. CE	Likelihood	N-gram
SWE-agent	**0.813**	0.452	0.626	0.317
SWE-smith	0.718	**0.719**	0.711	0.622

FSM features dominate on SWE-agent (+0.19 over the best probabilistic baseline), where the per-state decomposition captures structural information that aggregate transition statistics miss. On SWE-smith, transition cross-entropy is competitive (0.719 vs. 0.718), as the simpler FSM (10 states) offers less decomposition advantage. Probabilistic models capture *how often* transitions occur but not *what happens* at each state (message lengths, error rates, temporal patterns). The FSM provides both: its deterministic structure supports per-state feature extraction, while probabilistic models reduce each trace to a single scalar score.

G.6. ProbGuard Head-to-Head Comparison

We implement ProbGuard (Wang et al., 2025a) on the same activity sequences, labels, and 80/20 splits for a like-for-like comparison, reproducing its pipeline (Algorithms 1–2): symbolic-state abstraction, DTMC learning with Laplace smoothing (α=1), bounded reachability $P_{\leq\theta}[\, F^{\leq k}\ unsafe\,]$ by finite-horizon Bellman iteration, and a per-trace risk score equal to the maximum reachability along the trajectory. Its published evaluation uses hand-crafted predicates (`fork_in_microwave` $\wedge$ `microwave_on`); no such predicates exist for general LLM agent traces, so we follow its "extensible domain-specific abstraction" interface and take the activity itself as the symbolic state. To avoid handicapping the baseline we grant it every configuration advantage: log-odds ranking of unsafe states, sweeps over $K_{\text{unsafe}} \in \{1, 3, 5, 10\}$ and horizon $k \in \{3, 5, 10, 20, 50\}$, and a polarity flip $\max(\text{AUROC}, 1-\text{AUROC})$; we report the best of the resulting 20 configurations per dataset.

Table 27. ProbGuard (Wang et al., 2025a) vs. our FSM features for failure prediction. ProbGuard column = best AUROC over 20 configurations ($K_{\text{unsafe}}\times$ horizon, polarity-aware); FSM column = holdout AUROC from Table 14. Our FSM wins on every shared dataset by mean +17.6pp.

Dataset	ProbGuard (Wang et al., 2025a)	FSM features (ours)	Δ
τ_2-bench (tel)	0.709	**0.941**	**+0.232**
τ_2-bench (air)	0.723	**0.864**	**+0.141**
τ_2-bench (ret)	0.566	**0.779**	**+0.213**
SWE-agent	0.683	**0.799**	**+0.116**
SWE-smith	0.525	**0.703**	**+0.178**
Mean (5 shared datasets)	**0.641**	**0.817**	**+0.176**

Two factors drive the gap. Without ProbGuard's hand-crafted predicates the symbolic-state abstraction collapses to per-activity granularity, leaving each state with too few visits for reliable reachability estimation; and our cross-entropy anomaly features ($-\log_2 \hat{P}^{+}(a_t \mid q_t)$) give a continuous per-step risk score, whereas ProbGuard's PCTL-thresholded reachability is binary at deployment. ProbGuard's strength on its own benchmarks (autonomous driving, embodied agents) comes from those domain predicates; on general agent traces the FSM cross-entropy approach generalises further. The methods are complementary: a PCTL specification could be layered on our compact FSM (Theorem 3), combining its structural constraints with formal reachability checking.

G.7. Perturbation Robustness

Table 28 presents fitness degradation under four perturbation types at 10% intensity across all datasets. Substitution (replacing activities with random same-alphabet symbols) causes the largest fitness drop, confirming that the FSM captures transition topology rather than vocabulary.

Substitution is the strongest perturbation because it introduces invalid transitions (mean fitness drop 0.18). Insertion is intermediate (mean drop 0.10): the extra symbol breaks the current transition but the trace may recover. Swap is weakest (mean drop 0.05): reordering adjacent activities often preserves valid transitions if both orderings exist in the FSM. Truncation has no effect because the FSM accepts all prefixes by construction.

Table 28. Fitness under 10% perturbation intensity. Baseline fitness shown for reference.

Dataset	Subst.	Insert.	Swap	Trunc.	Base
SWE-smith	0.658	0.825	0.892	1.000	1.000
SWE-agent	0.818	0.885	0.938	0.999	0.999
Mind2Web	0.881	0.985	0.999	1.000	1.000
Who&When	0.828	0.865	1.000	1.000	1.000

G.8. FSM as Context

Table 29 evaluates the FSM as context for next-step prediction on the eight datasets where AWM has been re-implemented end-to-end against the same per-step predictor, comparing against AWM (Wang et al., 2024b). Transition counts are computed on training data; validation traces are replayed through the FSM for per-step predictions.

The FSM achieves higher top-1 accuracy than AWM on every listed dataset, with the gap ranging from +0.1pp to +65.5pp. AWM coverage varies from 0% (Mind2Web has no success labels) to 92.9%, explaining its performance variation. The LLM-judged

Table 29. Statistical next-step prediction: top-1 accuracy (%, ↑) on full validation sets.

Dataset	Steps	Unigram	FSM	AWM	AWM cov.
SWE-smith	5,500	50.0	**100.0**	34.5	34.5%
WebArena	17,146	26.2	**81.1**	81.0	92.9%
SWE-agent	22,288	49.1	**65.7**	26.6	57.0%
tau2 (tel)	23,055	33.0	**61.8**	19.8	59.0%
tau2 (air)	4,020	30.8	**69.2**	55.9	91.3%
tau2 (ret)	10,207	27.4	**72.6**	63.5	91.4%
Mind2Web	886	80.2	**80.2**	0.0	0.0%

AWM context **52.9% top-1**

```
## Extracted Workflow Patterns
- [freq=208] get_order_details
→ tool:text → tool:text
→ assistant:text → user:text
→ ...
- [freq=205] get_order_details
→ tool:text → ...
(10 patterns, 17--20 steps)
Agent prefix:  ... → get_order_details.
Next action?
```

LLM: `assistant:text` (actual: `tool:text`)

FSM context (minimal) **65.1% top-1**

```
After the most recent action
"get_order_details", past traces
show these next actions:
- tool:text:  100.0%
Common multi-step continuations:
- tool:text → get_order_details (1098x)
- tool:text → assistant:text (926x)
(top-15 shown)
Agent prefix:  ... → get_order_details.
Next action?
```

LLM: `tool:text` (actual: `tool:text`)

Figure 15. **Why minimal context wins.** Prompt excerpts at FSM state `get_order_details` (tau2-bench retail). AWM (52.9%): linear success workflows. ASG-minimal (65.1%): per-state next-action probabilities + top-15 continuations.

variant of this comparison (Table 4) additionally covers ATBench.

G.9. Next-Step Prediction

Order-1 FSM conditioning (Our FSM) captures 83–99% of the total CE improvement from Uniform to the best method on each dataset, reflecting the strong sequential regularity of agent traces. Higher-order context via PPM smoothing or neural models captures the remaining second- and higher-order dependencies within each FSM state. The FSM state benefit (+0.019 bits avg, FSM-LR vs. NGram-LR at $K{=}7$) is largest on Mind2Web (+0.041) and Who_and_When (+0.028), where the FSM groups behaviorally distinct states. On SWE-agent (+0.001), the FSM state is nearly redundant with the last activity due to simple sequential structure.

RPNI produces worse-than-Unigram predictions on most datasets. With 382–63,897 states, each state is visited by too few traces for reliable probability estimation. This validates the compression advantage: our 10–43 state FSMs aggregate observations for well-estimated transition probabilities. MLPs diverge on larger alphabets ($|\mathcal{A}| \geq 9$; CE $>$ 10 bits) from gradient instability, while RNNs without BPTT fail uniformly (CE $>$ 1.4). Echo state networks avoid both issues through random frozen reservoirs with trained output layers, achieving competitive performance with minimal hyperparameter sensitivity.

FSM state ablation. Holding the smoothing method fixed (absolute discounting at depth 5), FSM state conditioning provides +0.155 bits improvement on average over raw context alone (FSM-AD: 0.580 vs. Pure-AD: 0.735). The advantage is largest on Mind2Web (+0.36 bits, 30%) where the FSM groups heterogeneous web actions, and smallest on SWE-agent (+0.016 bits, 2%) where the simple sequential structure makes FSM state nearly redundant with the last activity. This gap is larger than the +0.019 bits from FSM-LR vs. NGram-LR (Table 2), because logistic regression partially learns FSM-like state from raw context.

H. Additional Analysis

H.1. Cross-Dataset Transfer

Table 30 presents the full cross-dataset transfer matrix, comparing FSM features against raw trace statistics. FSM features achieve higher transfer AUROC than raw features on most cross-dataset pairs, with the largest improvements on transfers involving SWE-agent. This indicates that the per-state behavioral signal captures failure structure that generalizes beyond the training domain, whereas raw trace statistics overfit to dataset-specific surface features. Leave-one-out results (train on 1 dataset, test on the other): SWE-smith 0.682, SWE-agent 0.765. The drop from in-domain is modest on SWE-agent (0.780→0.765), indicating that FSM behavioral features

Table 30. **Cross-dataset** transfer: FSM vs. raw feature AUROC. Bold: FSM advantage $>$ 3pp.

		Test dataset	
Train	Features	SWE-sm	SWE-ag
SWE-smith	FSM	*0.681*	**0.715**
	Raw	*0.694*	0.683
SWE-agent	FSM	0.648	*0.780*
	Raw	0.682	*0.720*

capture domain-invariant failure signatures.

H.2. Sample Efficiency and Learning Curves

Table 31 presents failure prediction AUROC as a function of training set size. On 3 of 4 datasets, 10% of training data suffices to reach 95% of final performance. The rapid convergence reflects the low dimensionality of FSM feature space (31–49 features) relative to the behavioral complexity captured.

SWE-agent shows the most stable learning curve, consistent with its larger sample size (1,600 training traces). The practical implication is that FSM-based failure prediction can be deployed with as few as 16–160 labeled traces, making it viable for new agent systems where labeled data is scarce.

Table 31. Learning curves: failure prediction AUROC at increasing training fractions. Bold: first fraction reaching 95% of final AUROC.

Dataset	10%	20%	30%	50%	100%
SWE-smith	0.502	**0.662**	0.703	0.660	0.685
SWE-agent	**0.806**	0.791	0.812	0.818	0.795

H.3. Length vs. Structure Ablation

On SWE-agent, structural features alone reach AUROC **0.790**, length alone only 0.659, and the full model 0.790: adding length to the structural features changes full-model AUROC by <0.001, confirming that the predictive signal is structural rather than a length proxy.

H.4. Failure Mode Characterization

On SWE-agent, the two failure modes are structurally distinct: "stuck in edit loop" traces have cycle rate 0.924 and terminate in `edit:text` (50%), while "gave up early" traces have lower cycle rate (0.755) and reach `submit:text` (98.7%) but still fail. The discriminating feature with highest F-ratio is: `visit:user:text` (7.41 on SWE-agent).

H.4.1. Failure Cascade Analysis

We analyze whether failures develop gradually (progressive fitness degradation) or suddenly (abrupt state change). Across all datasets, 97% of SWE-agent failures are sudden (1,292 of 1,329), with no gradual degradation pattern. This is consistent across datasets: SWE-smith 100% sudden. The implication is that failure monitoring should focus on detecting specific state patterns (e.g., cycle rate threshold) rather than tracking gradual performance decline.

H.4.2. Failure Progression

Failure signatures emerge early in execution. On SWE-agent, the first divergence between success and failure state distributions occurs at 8.7% of trace length (position 0.087). On SWE-smith, divergence occurs at 15.4%. The highest-lift failure transition is `tool:text`→`assistant:text` on SWE-smith (fail rate 0.90, lift 3.66× over base rate). Recovery from failure-indicative states is possible: SWE-agent has 8 recovery states where traces can return to successful trajectories, with 95.7% recovery rate within 2 steps.

H.5. Counterfactual Path Analysis

We identify FSM decision points where success and failure paths diverge, measured by Jensen-Shannon divergence of outgoing transition distributions.

On SWE-agent, failures exhibit 5.4× more unique paths than successes (1,496 vs. 275), with only 41 shared paths. The `user:text` state is the primary decision point (JSD 0.014): at this state, successful traces are more likely to transition to `submit` (25.3% success rate) while failed traces loop back to `edit` (9.6% success rate). On SWE-smith, the `tool:tool_call` state shows the highest divergence (JSD 0.021). Early divergence is common: 65.3% of SWE-agent traces diverge within the first 10% of execution.

Table 32. Counterfactual path analysis. Decision points: states with JSD > 0.001 between success/failure transitions.

Dataset	DecPts	Succ paths	Fail paths	Overlap	Top JSD
SWE-agent	7	275	1,496	41	0.014
SWE-smith	5	330	123	11	0.021

Figure 16. (a) State compression across labeled datasets: our FSM (10–43 states) vs. Alergia (10–149) and RPNI (382–63,897), with compression ratios annotated. (b) Cross-dataset fitness matrix: replaying traces from one dataset through another's FSM. Diagonal entries (in-distribution) approach 1.0; off-diagonal entries (OOD) drop to near-zero for structurally distinct pairs (AUROC 1.000), except schema-sharing tau2-bench airline↔retail (near-1.0).

H.6. Compression Theory

Table 33 presents information-theoretic analysis of FSM compression across all datasets. FSM bits count the encoded transition table; raw bits count $\log_2 |\mathcal{A}|$ per activity summed over all traces, so the MDL ratio measures how much of the raw description the FSM eliminates.

The FSM achieves MDL ratios of 0.001–0.008 across all datasets, meaning the FSM description requires 0.1–0.8% of the bits needed to store raw traces. This is 5–46× better than gzip compression (ratios 0.012–0.026), confirming that the FSM captures genuine behavioral regularity beyond statistical redundancy. All FSMs satisfy the Myhill-Nerode theorem (verified programmatically) with 100% alphabet utilization on 3/4 datasets shown. SWE-agent has alphabet utilization 1.13 (3 states have transitions for symbols not in the core alphabet, reflecting rare command variants). Conditional entropy analysis: unigram entropy ranges 1.69–2.16 bits; conditioning on the previous symbol (bigram) reduces entropy to 0.63–1.06 bits (51–68% reduction), confirming strong sequential regularity in agent traces.

Table 33. Information-theoretic compression analysis. MDL: minimum description length ratio (FSM bits / raw trace bits). Gzip: compression ratio of raw sequences.

Dataset	$\lvert\mathcal{A}\rvert$	$\lvert Q\rvert$	FSM bits	Raw bits	MDL	Gzip
SWE-smith	9	10	108	107,704	0.001	0.012
SWE-agent	27	25	340	556,450	0.001	0.018
Mind2Web	7	8	87	12,786	0.007	0.026
Who&When	8	9	99	12,276	0.008	0.019

H.7. Path Diversity and Recurrence Analysis

Table 34 quantifies path diversity and recurrence quantification analysis (RQA) metrics across datasets. RQA treats activity sequences as symbolic time series.

On coding agent datasets, 82% of traces follow unique FSM paths, yet the FSM compresses all into 6–25 states with ≥0.999 fitness by capturing *transition topology* rather than memorizing paths. API-driven domains show opposite extremes: tau2-bench telecom has only 5 unique paths across 1,824 traces (highly constrained workflows), while tau2-bench retail has 1,521 unique paths (73.2% singletons). RQA metrics achieve 0.60–0.70 AUROC on coding agents, with diagonal entropy strongest on SWE-agent (0.704), but are weaker on web/GUI benchmarks (0.31–0.47 on GUI-Odyssey and AgentNet) where trace structures are less recurrent.

Table 34. **Path diversity and RQA** metrics. Singleton%: paths observed once. Top-5%: coverage of 5 most common paths. RR/DET/diagEnt: recurrence rate, determinism, diagonal entropy (AUROC for failure prediction). Who&When and Mind2Web lack success/failure labels (–).

	Path diversity				RQA (AUROC)			
Dataset	Traces	Unique	Sing.%	Top-5%	RR	DET	maxDiag	diagEnt
SWE-smith	500	442	82.0	5.8	**0.662**	0.643	0.631	0.599
SWE-agent	2,000	1,730	82.0	5.1	0.695	0.694	0.695	**0.704**
tau2-tel	1,824	5	0.0	100.0	0.562	0.563	0.562	**0.566**
tau2-air	800	624	67.5	6.4	0.511	0.338	0.566	**0.612**
tau2-ret	1,824	1,521	73.2	2.0	**0.574**	0.471	0.520	0.521
GUI-Ody	7,735	3,601	36.7	5.2	0.223	0.296	**0.310**	0.306
WebArena	8,337	827	3.8	27.7	**0.416**	0.402	0.340	0.406
AgentNet	5,000	3,965	73.3	3.7	**0.548**	0.487	0.420	0.474
Who&When	184	139	60.3	13.0	–	–	–	–
Mind2Web	500	238	36.0	20.2	–	–	–	–

H.8. State Importance Analysis

We measure state importance by fitness drop when each state is removed from the FSM (Table 35). State importance is not proportional to visit frequency. On SWE-smith, `system:text` and `user:text` each receive only 1.9% of visits but removing either causes complete fitness collapse (1.000 and 0.979 drop). Conversely, `tool:text` receives 47.2% of visits but its removal drops fitness by only 0.937, because its behavioral role can be partially compensated by other states. On SWE-agent, `user:text` is the single critical bottleneck (49.9% visits, 0.999 fitness drop), reflecting its role as the central hub connecting all behavioral modes.

Table 35. State importance: fitness drop upon state removal. Top 3 most critical states per dataset.

Dataset	State	Visit%	Fitness drop
SWE-smith	system:text	1.9	1.000
	user:text	1.9	0.979
	asst:tool:bash	21.3	0.958
SWE-agent	user:text	49.9	0.999
	edit:text	16.6	0.344
	search:text	11.5	0.208
Mind2Web	user:text	11.7	1.000
	asst:click	72.0	0.710
	asst:type	10.2	0.102

H.9. Loop and Graph Motif Analysis

Tables 36 and 36 analyze loop patterns (backedge counts) and graph motifs in the FSM.

Table 36. **Loop analysis and graph motifs.** Left: loop counts (backedges) per outcome and AUROC. Right: structural motif counts. Bidir: bidirectional edge pairs. Hub degree: maximum out-degree. Who&When and Mind2Web lack success/failure labels (–).

	Loops (backedges)				Graph motifs			
Dataset	Succ	Fail	Ratio	AUROC	Self	Bidir	Tri	Hub deg
SWE-agent	25.3	54.8	**2.17**×	0.665	0	11	0	15
SWE-smith	43.6	60.7	1.39×	0.654	0	6	0	7
tau2-tel	1.9	1.9	1.00×	0.562	1	2	2	4
tau2-air	17.1	21.5	1.26×	0.664	1	14	0	15
tau2-ret	20.3	21.0	1.03×	0.537	1	15	0	16
GUI-Ody	12.0	9.1	0.76×	0.362	4	5	6	6
WebArena	9.9	6.4	0.65×	0.289	0	8	0	9
AgentNet	28.1	21.0	0.75×	0.314	0	9	0	12
Who&When	–	–	–	–	3	5	6	12
Mind2Web	–	–	–	–	4	3	2	6

On coding agents, failed traces contain 1.4–2.2× more loops, strongest on SWE-agent (2.17×, 54.8 vs. 25.3 backedges, AUROC 0.665). Interestingly, on web/GUI benchmarks the pattern reverses: successful WebArena traces loop *more* (9.9 vs. 6.4, ratio 0.65×), reflecting productive exploration in complex navigation tasks. tau2-bench airline shows the strongest signal among API agents (1.26×, AUROC 0.664). Graph motif analysis reveals that tau2-bench airline/retail have the densest bidirectional structure (14–15 pairs), while triangles appear only in Who&When (6, multi-agent delegation), GUI-Odyssey (6, cross-app navigation), and tau2-bench telecom (2).

H.10. Critical Transitions and Error Localization

Table 37 identifies structural bottlenecks (transition criticality = frequency × success differential) and per-state error rate differentials between success and failure traces.

The highest-criticality transitions form tight cycles: `str_replace_editor`↔`tool:text` on SWE-smith (1.048) and `user:text`↔`edit:text` on SWE-agent (0.657). Errors are localized to specific states: on SWE-smith, `assistant:text` has zero errors in successes but 22.2% in failures (+0.222 differential). The first significant divergence between success and failure distributions occurs at the `edit:text` vs. `search:text` branch on SWE-agent (position 2).

Table 37. Critical transitions (top: highest criticality scores) and error localization (bottom: per-state error rate differential, failure − success).

Critical transitions			
Dataset	Transition	Freq	Criticality
SWE-smith	str_repl→tool:text	12.96	1.048
SWE-smith	tool:text→str_repl	12.96	1.045
SWE-agent	user:text→edit:text	9.20	0.657
SWE-agent	edit:text→user:text	9.02	0.629
Error localization (per-state error rate differential)			
Dataset	State	Succ/Fail rate	Δ
SWE-smith	asst:text	0.000 / 0.222	+.222
SWE-agent	edit:text	0.368 / 0.486	+.118
SWE-agent	asst:text	0.000 / 0.071	+.071

H.11. Temporal Dynamics

We analyze three-phase (early/mid/late) behavioral dynamics to understand how agent behavior evolves during execution.

Most datasets show negative entropy drift, indicating that agents narrow their behavioral repertoire over time. The effect is strongest on Mind2Web (−1.03) and weakest on GUI-Odyssey (−0.04). Two datasets show *positive* drift: WebArena (+0.19) and AgentNet (+0.18), where agents diversify behavior in later phases, possibly reflecting recovery or exploration after initial failures.

H.12. Anomaly Detection and Suffix Monitoring

We evaluate two unsupervised monitoring approaches: (1) a multi-component anomaly score combining rejection rate, state occupancy deviation, terminal state anomaly, cycle excess, and length deviation; and (2) suffix-based monitoring using only the last k FSM transitions (Table 39).

The composite anomaly score achieves 0.653–0.747 AUROC, strongest on SWE-agent (0.747, P@10 = 1.00).

The dominant component is terminal anomaly on SWE-agent (0.711). Suffix monitoring with k=10 transitions achieves comparable AUROC to the full predictor (0.825 on SWE-agent), enabling real-time deployment with a fixed-size sliding window. Even k=3 yields 0.746 AUROC on SWE-agent.

Table 38. **Temporal dynamics:** entropy by execution phase and entropy drift (early − late). Who&When and Mind2Web lack success/failure labels (–).

Dataset	Early H	Mid H	Late H	Drift
SWE-agent	1.83	1.55	1.69	−0.15
SWE-smith	2.17	1.45	1.72	−0.44
tau2-tel	1.39	0.95	0.97	−0.42
tau2-air	2.17	1.82	1.95	−0.21
tau2-ret	2.32	2.00	2.03	−0.29
GUI-Ody	1.23	0.63	1.19	−0.04
WebArena	1.42	1.24	1.60	+0.19
AgentNet	2.10	1.85	2.28	+0.18
Mind2Web	1.46	0.68	0.43	−1.03
Who&When	1.41	0.59	0.68	−0.73

Table 39. **Unsupervised monitoring.** Left: anomaly detection components and composite AUROC. Right: suffix monitoring AUROC at window size k.

	Anomaly detection						Suffix (last k)			
Dataset	Rej	Occ	Term	Cyc	Comp	P@10	k=2	k=3	k=5	k=10
SWE-agent	0.563	0.707	**0.711**	0.697	**0.747**	1.00	0.693	0.746	0.771	**0.825**
SWE-smith	0.406	0.605	0.429	**0.679**	0.653	0.20	0.540	0.546	0.550	**0.590**

H.13. Agent Integration: Runtime Monitor

Figure 4 (body) shows the cycle-rate trajectory contrast between a failing and a successful SWE-agent run. The failing trace enters a cycle between `user` and `edit` states; cycle-rate exceeds 0.778 at step 11 (31% of this specific trace; mean across all interventions: 32%), triggering early termination. The successful trace visits 7 distinct states and reaches `submit`; its

cycle-rate peaks at 0.636 and never crosses the threshold. Monitor $F_1 = 0.904$ on SWE-agent without any trained model.

We simulate deploying the FSM as a runtime monitor that replays agent actions step-by-step and triggers intervention when learned rules fire. Rules are automatically derived from training traces: cycle-rate thresholds (percentile-based) and minimum unique-state counts, with a warm-up period (10% of mean trace length) before activation. A "stuck" detector also fires when the agent remains in the same state for 5+ consecutive steps.

Table 40. Runtime monitor results. Latency: mean % of trace at intervention. Saved: mean % of remaining computation avoided.

Dataset	Prec	Recall	F1	Latency	Saved	Steps saved	Lift
SWE-agent	85.9%	95.5%	**0.904**	32%	68%	16,711	1.02×
tau2-bench (air)	76.0%	79.2%	0.776	56%	44%	325	1.27×
tau2-bench (ret)	31.0%	60.0%	0.409	63%	37%	247	0.95×
SWE-smith	16.0%	100.0%	0.276	17%	83%	936	1.00×

The monitor is reliable on FSMs with high structural diversity ($|Q| \geq 6$ active states with distinct failure patterns): SWE-agent ($F_1 = 0.904$, lift 1.02×) and tau2-bench airline ($F_1 = 0.776$, lift 1.27×). On the smaller-alphabet datasets, the rule under-discriminates: tau2-bench retail and SWE-smith FSMs are too coarse for cycle-rate to separate failure modes, and the monitor over-triggers. The two-dataset $F_1 \geq 0.776$ result establishes a working operating regime; deploying on small-alphabet domains requires per-dataset rule tuning.

Operating point analysis. Table 41 shows precision–recall trade-offs at different cycle-rate thresholds on SWE-agent.

At the high-precision operating point (threshold 0.957), the monitor achieves 100% precision (zero false alarms) while catching 11.3% of failures. This is suitable for automated intervention (e.g., resetting the agent) where false positives are costly. At the balanced operating point (threshold 0.750), the monitor catches 85.2% of failures with 86.4% precision, suitable for alerting a human operator. The entire monitoring pipeline requires only FSM replay at 0.006 ms per step with no ML model training.

Table 41. Multi-threshold analysis (cycle-rate only) on SWE-agent. Higher thresholds yield higher precision at the cost of recall.

Threshold	Precision	Recall	F1	Mean latency
>0.750 (p30)	86.4%	85.2%	0.858	39%
>0.842 (p50)	90.5%	65.3%	0.759	47%
>0.917 (p70)	95.6%	38.9%	0.553	63%
>0.957 (p90)	100.0%	11.3%	0.203	59%

H.14. Sequence-Level vs. FSM Feature Comparison

Table 42. Failure prediction: sequence-level features (bag, bigram, stats, all-seq) vs. FSM per-state features, L1-regularized LR. d: feature dimensionality. FSM wins on every dataset.

Dataset	Features	d	CV AUROC	Holdout
SWE-agent	bag (freq.)	24	0.702 ± 0.030	0.694
	bigram (trans.)	62	0.699 ± 0.029	0.698
	stats (seq.)	6	0.757 ± 0.027	0.782
	all-seq (LR)	92	0.771 ± 0.027	0.793
	all-seq (MLP)	92	0.711 ± 0.044	0.764
	FSM per-state	**34**	$\mathbf{0.790 \pm 0.021}$	**0.813**
SWE-smith	bag (freq.)	9	0.685 ± 0.054	0.708
	bigram (trans.)	18	0.685 ± 0.052	0.708
	stats (seq.)	6	0.662 ± 0.056	0.690
	all-seq (LR)	33	0.686 ± 0.051	0.703
	all-seq (MLP)	33	0.637 ± 0.062	0.691
	FSM per-state	**49**	$\mathbf{0.688 \pm 0.050}$	**0.718**

Table 42 compares sequence-level feature representations against FSM per-state features for failure prediction, all using the same L1-regularized LR (C=0.1, class-weighted). Sequence features use only the activity symbols (no message content): bag (frequency histogram, $|\mathcal{A}|$ feat.), bigram (transition matrix, $|\mathcal{A}|^2$ feat.), stats (8 sequence statistics), and all-seq (all three concatenated). MLP uses a 2-layer network (64, 32 units) with early stopping on all sequence features.

FSM per-state features outperform all sequence-level representations on all datasets in both CV and holdout AUROC. The advantage is largest on SWE-agent (+0.088 CV, +0.020 holdout), where per-state features capture differences that flat counts cannot localize. On SWE-smith, the gap is minimal (+0.002 CV) because the 10-state FSM with 9-symbol alphabet provides limited decomposition advantage. The MLP underperforms LR on all datasets, confirming the ceiling is data-limited (400–1,600 traces) rather than model-limited.