FRONTIERGUARD: LEARNING EXECUTABILITY FRONTIERS FOR STRUCTURED WEB AND API AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A checkout agent can face the same visible submit control twice: before shipping fields are complete the action is illegal, while one successful form event later it is executable. API agents encounter the analogous transition when an earlier response produces a required identifier. FrontierGuard learns this moving boundary as an *executable-state quotient* over grounded interaction events. Histories are merged only when they preserve action-feasibility profiles and event-conditioned successors; the resulting finite-state controller maps each proposal to `allow`, `block`, or one read-only `query` before any side effect. Across 1,152 WebArena and ToolBench-API tasks, FrontierGuard reduces GPT-4o WebArena illegal actions from 23.4% to 10.7% (54.3% relative), raises success from 18.7% to 24.5%, and cuts dead ends from 4.2% to 1.9%; on ToolBench-API it adds 3.5 success points beyond schema-constrained decoding. At the same 8.4% query rate, the quotient adds 1.3 WebArena success points and removes 1.5 illegal-action points relative to a same-predicate GRU. Pooled leave-one-family-out evaluation retains the strongest listed operating point on both benchmarks, showing that a learned execution boundary can convert local interface evidence into more completed trajectories.

1 INTRODUCTION

Consider the checkout fragment in Figure 1. A `submit` proposal at the incomplete-shipping state is rejected; after `fill_form` executes and the interface reports valid fields, the corresponding proposal is admitted. The control is present in both states and its action syntax is unchanged, but its executability flips with event history. API workflows exhibit the same pattern when a response makes an identifier available to a later call. Such transitions are common in structured web and service interfaces, where form, modal, session, argument, and response state mediate what an agent can execute (Schick et al., 2023; Qin et al., 2024; Nakano et al., 2021; Wang et al., 2024; Zhou et al., 2024; Drouin et al., 2024).

This paper treats *executable state* as the control variable. Histories belong to the same state when they support the same action-feasibility profile after every realizable event continuation, including agreement on which continuations are realizable. The induced *executable-state quotient* compresses a trajectory around one question—which proposed side effects are supported now?—rather than around the agent's full planning state. Explicitly marking unrealizable continuations makes this relation an equivalence and a right congruence, so its action frontier is well defined.

FrontierGuard approximates that quotient from traces of executed events and their resulting DOM, accessibility, schema, or runtime-response predicates. Its action-indexed readout admits supported proposals, rejects contradicted ones with predicate feedback, and spends one read-only probe on information-deficient cases. Only executed side effects and probes advance state; a rejected proposal leaves the monitor state unchanged. Asymmetric false-allow cost and calibrated thresholds turn observation into a budgeted control decision, while the unchanged proposal policy remains responsible for task strategy.

The closest alternatives control different objects. Constrained decoders recognize authored action languages (Polozov et al., 2022; Scholak et al., 2021; Beurer-Kellner et al., 2023); runtime shields and contracts enforce authored temporal, privilege, or precondition specifications (Leucker & Schallhart, 2009; Alshiekh et al., 2018; Shi et al., 2025; Liu et al., 2026); safety guards score harmfulness, authorization, or plan risk (Jia et al., 2025; Yuan et al., 2026; Hossain et al., 2026); and learned

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

affordances or action elimination estimate mostly step-local feasibility (Zahavy et al., 2018; Do et al., 2018; Mandikal & Grauman, 2022). FrontierGuard instead learns dynamic executability, preserves event-conditioned phase memory, and resolves uncertainty before a side effect through an interface observation.

The primary contribution is a finite behavioral quotient specialized to executable actions, together with a trace-induced controller that operationalizes its frontier. The experiments then test the representation rather than relying on the controller package alone: matched feed-forward and recurrent monitors isolate current predicates, generic memory, and the finite quotient; component interventions isolate querying, asymmetric risk, relational grounding, and capacity; family withholding tests whether the state construction survives unseen interface groups. On 812 WebArena and 340 ToolBench-API tasks, FrontierGuard simultaneously improves completion and runtime validity, including a 5.8-point GPT-4o WebArena success gain, a 54.3% relative reduction in illegal actions, and a statistically supported advantage over a same-predicate GRU at equal query rate.

2 Executable-State Quotient

Observable executability. Let s_t be the complete runtime state and x_t the structured state exposed to the controller: DOM and accessibility attributes plus session metadata for web tasks, or schema and runtime-response fields for API tasks. An action's observable legality is

$$\ell_{\mathrm{obs}}(a_t, x_t) = \mathbf{1}[\text{the exposed runtime accepts } a_t \text{ at } x_t]. \tag{1}$$

The extractor ϕ maps x_t to a lossy $f_t \in \{0,1\}^{64}$ over object class, interaction mode, focus, form validity, argument availability, response state, and structural relations. FrontierGuard estimates Eq. (1) from the event–predicate history; this can recover interaction phase that one compressed vector omits without changing the target into a history-defined label. Full execution validity $\ell_{\mathrm{exec}}(a_t, s_t)$ additionally reflects backend permissions, expired sessions, rate limits, and other state absent from x_t. Their disagreement measures the observability gap. Task correctness remains a separate reward property: an executable action can still be strategically poor.

Executable-state quotient. Let $\mathcal{A}$ be the normalized proposal alphabet, $\mathcal{R}$ the disjoint set of read-only probes, and $\mathcal{E} = \mathcal{A} \uplus \mathcal{R}$ the events that can actually be issued to the interface. A realizable history is $h_n = f_0(e_1, f_1)\cdots(e_n, f_n)$ in a prefix-closed set $\mathcal{H} \subseteq \mathcal{F}(\mathcal{E} \times \mathcal{F})^*$. Each f_i is observed after executing e_i; a blocked proposal is not an event and does not extend h_n. Define $v(h) = [P(\ell_{\mathrm{obs}} = 1 \mid h, a)]_{a\in\mathcal{A}} \in \mathcal{V}$, where $\mathcal{V} = [0,1]^{\mathcal{A}}$. For $u \in (\mathcal{E} \times \mathcal{F})^*$, totalize the residual profile with an explicit unrealizable symbol $\dagger \notin \mathcal{V}$:

$$\mathrm{Res}_h(u) = \begin{cases} v(h \circ u), & h \circ u \in \mathcal{H}, \\ \dagger, & h \circ u \notin \mathcal{H}. \end{cases}$$

The ideal quotient is equality of these residual functions:

$$h \equiv_{\mathrm{exec}} h' \iff \forall u \in (\mathcal{E} \times \mathcal{F})^*, \quad \mathrm{Res}_h(u) = \mathrm{Res}_{h'}(u). \tag{2}$$

Equation 2 cannot equate disjoint futures vacuously: equivalent histories have the same continuation language and feasibility profile. Because it is equality of total functions, it is an equivalence; shifting both residuals by the same event–observation symbol proves right congruence. Appendix A.1 gives the proof. Finite traces expose only part of each residual, so FrontierGuard learns an approximate quotient whose merges preserve observed event-conditioned successor determinism and trade profile fidelity against state complexity.

Frontier readout. For quotient state q_t and proposal a, let $p_t(a) = P(\ell_{\mathrm{obs}} = 1 \mid q_t, a)$. Thresholds with $\theta_{\mathrm{block}} < \theta_{\mathrm{allow}}$ partition the proposal space into

$$\mathcal{A}_t^+ = \{a : p_t(a) \geq \theta_{\mathrm{allow}}\}, \quad \mathcal{A}_t^- = \{a : p_t(a) \leq \theta_{\mathrm{block}}\}, \quad \mathcal{A}_t^? = \mathcal{A} \setminus (\mathcal{A}_t^+ \cup \mathcal{A}_t^-). \tag{3}$$

The ordered partition $(\mathcal{A}_t^+, \mathcal{A}_t^?, \mathcal{A}_t^-)$ is the executability frontier, with $\mathcal{A}_t^?$ its information-deficient band. FrontierGuard realizes the quotient and readout with an action-indexed Mealy monitor $\mathcal{M} = (Q, \mathcal{F}, \mathcal{E}, \mathcal{A}, \Lambda, \iota, \delta, \lambda)$, where $\Lambda = \{\texttt{allow}, \texttt{block}, \texttt{query}\}$, $\iota : \mathcal{F} \to Q$ initializes the state, $\delta : Q \times \mathcal{E} \times \mathcal{F} \to Q$, and $\lambda : Q \times \mathcal{A} \to \Lambda \times [0,1]$ returns a decision and confidence. For $h_n = f_0(e_1, f_1)\cdots(e_n, f_n)$, $q_0 = \iota(f_0)$ and $q_i = \delta(q_{i-1}, e_i, f_i)$. Calling λ never changes state.

Risk objective. The backbone policy π is fixed. Let z_t indicate whether the controller queries and let $\bar{d}_t \in \{\texttt{allow}, \texttt{block}\}$ be its final decision after any probe. FrontierGuard selects its operating point by minimizing

$$\begin{aligned} \min_{\mathcal{M}} \quad & \mathbb{E}\big[w_{\text{fa}}\mathbf{1}[\bar{d}_t = \texttt{allow}, \ell_{\text{obs}} = 0] + w_{\text{fb}}\mathbf{1}[\bar{d}_t = \texttt{block}, \ell_{\text{obs}} = 1] + c_q z_t\big] \\ \text{subject to} \quad & \mathbb{E}[z_t] \le \rho. \end{aligned} \tag{4}$$

Validation selects the operating point under query budget ρ; task success evaluates the downstream consequence rather than entering monitor fitting.

3 FrontierGuard

Predicate grounding and action normalization. The 64 binary slots encode 32 ternary tests as paired value/missingness bits across seven families: object role; visibility and enablement; ARIA and focus state; form/type validity; argument and dependency availability; response/session mode; and structural relations. Appendix Table A1 gives every indexed test and its modality mapping. Web observations populate the tests from the DOM and accessibility tree; API observations use OpenAPI fields and runtime responses. Before readout, the proposal target is resolved by accessibility or DOM identity on the web and by operation identifier plus argument name for APIs. This lookup augments the event descriptor but does not advance monitor state. Actions are normalized for profile estimation as

$$a = (\text{modality}, \text{operation}, \text{target role}, \text{side-effect class}),$$

so surface-specific names share evidence without merging reads, writes, navigation, submission, or argument-bearing calls. Examples are `web:click:submit:write` and `api:call:required-args-ready:write`. A target that cannot be bound sets its target-dependent predicates to unknown rather than silently treating them as false.

State and transition learning. Each executed event–observation prefix is first a node in a prefix tree. For a node set C, let $N_y(C, a)$ count observed label y for normalized action a, and let $m(C, a)$ indicate nonzero support. Its sparse profile is

$$p_C(a) = \frac{N_1(C, a) + 1}{N_0(C, a) + N_1(C, a) + 2} \quad \text{when } m(C, a) = 1.$$

Missing entries remain unknown and map to `query`; they are never imputed as legal. For two nodes with shared action support $\mathcal{A}_{CD}$, profile distance is the support-weighted mean

$$d(C, D) = \frac{\sum_{a \in \mathcal{A}_{CD}} \min\{N(C, a), N(D, a)\} |p_C(a) - p_D(a)|}{\sum_{a \in \mathcal{A}_{CD}} \min\{N(C, a), N(D, a)\}}.$$

Pairs are positive when all shared actions have the same empirical majority label and negative when any shared action has opposite majorities. With $\mathcal{P}^+$ and $\mathcal{P}^-$ denoting these sets, the contrastive term is

$$\mathcal{L}_{\text{contrast}} = \frac{1}{|\mathcal{P}^+|} \sum_{(C,D) \in \mathcal{P}^+} d(C, D)^2 + \frac{1}{|\mathcal{P}^-|} \sum_{(C,D) \in \mathcal{P}^-} [1 - d(C, D)]_+^2.$$

This construction uses only co-observed actions; a pair with no shared support is ineligible to merge. The complete partition score is

$$\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_i w_{\ell_i} \log P(\ell_i \mid q_i, a_i) + \beta \mathcal{L}_{\text{contrast}} + \alpha \frac{|Q| \log |\mathcal{D}|}{|\mathcal{D}|}, \tag{5}$$

with $w_0 : w_1 = 3 : 1$. A merge is deterministic-compatible exactly when, for every executed-event and resulting-signature pair (e, f) observed after both candidate nodes, their successors are already equal or can be merged recursively without joining a negative pair; event identity is never discarded. Among compatible candidates we take the largest decrease in Eq. 5, break ties by prefix-tree creation order, recursively coalesce required successors, and stop when no candidate decreases the score. We retain every partition on each coefficient-induced merge path; validation risk in Eq. 4 selects among these partitions and state caps $\{8, 16, 32, 64\}$, so α and β index the search rather than

Algorithm 1 FrontierGuard pre-execution control

Require: Policy π, monitor $\mathcal{M}$, current observation x, state q
$a \sim \pi(\cdot \mid x, \text{policy transcript})$; $(d, \gamma) \leftarrow \lambda(q, a)$
if $d = \texttt{query}$ **then**
  Select probe r; execute r; observe x^r
  $f^r \leftarrow \phi(x^r)$; $q \leftarrow \delta(q, r, f^r)$; append (r, f^r) to monitor history
  $(d, \gamma) \leftarrow \lambda(q, a)$
  if $d = \texttt{query}$ **then**
    $d \leftarrow \texttt{block}$
  end if
end if
if $d = \texttt{allow}$ **then**
  Execute a; observe x^+; $f^+ \leftarrow \phi(x^+)$
  $q \leftarrow \delta(q, a, f^+)$; append (a, f^+) to monitor history
else
  Return predicate feedback; do not apply δ
end if

survive as deployment parameters. The selected standard operating point fixes $\rho = 0.084$; c_q is the corresponding validation-set Lagrange multiplier.

After merging,

$$\delta(q, e, f) = \arg\max_{q'} N(q, e, f, q'), \qquad P(\ell_{\text{obs}} = 1 \mid q, a) = \frac{N_1(q, a) + 1}{N_0(q, a) + N_1(q, a) + 2}.$$

For an unseen predicate signature after a supported (q, e) pair, backoff is restricted to signatures previously observed after that same event and minimizes inverse-training-frequency weighted Hamming distance; ties prefer the more supported transition and then creation order. An unseen event has no transition support, and an unseen normalized proposal has no profile support; both are routed to a supported read-only probe rather than imputed as legal. The recurrence $q_i = \delta(q_{i-1}, e_i, f_i)$ therefore realizes an event-conditioned history quotient.

Calibrated intervention. The learned output map applies $\theta_{\text{allow}} = 0.85$ and $\theta_{\text{block}} = 0.15$. For candidate probe r, training traces estimate its possible refreshed signatures and score

$$G(r; q, a) = H(p_q(a)) - \sum_{f'} \widehat{P}(f' \mid q, r)\, H\big(p_{\delta(q, r, f')}(a)\big),$$

with Laplace smoothing and binary entropy H. The controller chooses the available probe with largest G; ties prefer lower measured median latency. The menu is `inspect_element` (∼50 ms), `refresh_state` (∼200 ms), `list_schema` (∼100 ms), and `read_only_probe` (∼300 ms). Appendix Table A2 defines their modality, availability, and returned fields. One probe is permitted per proposal. The probe appends one event–observation pair and updates state once; if the refreshed decision allows the original proposal, its execution appends a second pair. A direct or post-query block appends neither the blocked action nor a state transition.

Incomplete and complete shipping histories occupy different states in Figure 1 because the same submit role has different feasible-action profiles. At payment, a read-only probe refines the state before the pay event is admitted. The graph separates controller outputs from executed events: blocking consumes budget and returns feedback, but cannot create a fictitious interface transition.

4 Experiments

The evaluation asks whether pre-execution control improves complete trajectories, whether the quotient contributes beyond current predicates and generic recurrence, and how that advantage changes under weaker or unseen grounding.

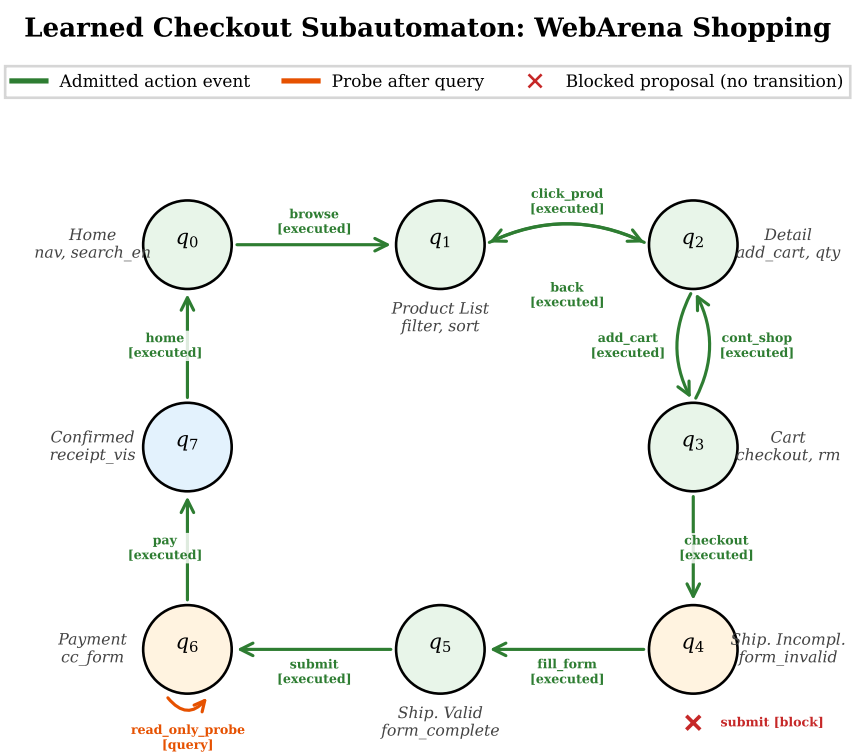

Figure 1: An eight-state checkout subautomaton from FrontierGuard's 28-state WebArena Shopping quotient. Green arrows are executed actions admitted by the readout; the orange self-loop is a read-only probe selected after `query`; red crossed annotations are blocked proposals and produce no state transition.

Benchmarks, models, and protocol. WebArena contributes 812 tasks across Shopping, Forums, CMS, Maps, and GitLab (Zhou et al., 2024); ToolBench-API is the 340-task executable REST slice of the official ToolBench release, spanning 47 categories (Qin et al., 2024). Appendix B defines its inclusion rule, grouping, success evaluator, and response handling. We evaluate GPT-4o and Claude-3.5 Sonnet on WebArena and GPT-4o on ToolBench. Three independent seeds share task instances, prompt templates, model snapshots, action vocabulary, initial states, observation channels, 30-step limits, and 10-minute timeouts. Controller decisions necessarily create different later observations and proposals, so candidate sequences become method-specific after trajectories diverge.

Interaction accounting. A direct `allow` execution or a `block` consumes one task step. A block returns predicate feedback but obtains no free replacement: the next proposal requires another backbone call on the next step. A query consumes one step; if the refreshed decision allows the action, its execution consumes the next step, while a post-query block returns feedback within the query step. Each failed execution and each Post-hoc Filter retry likewise consumes a step. There is one backbone call per proposal and none for a probe. Thus all methods share the same environment-step and timeout ceilings, while proposal and execution counts may differ as a consequence of control. The reported latency is controller-plus-probe time and excludes provider generation; Appendix Table A13 reports both this latency and observed backbone-call counts. Monitor fitting and predicate extraction use eight A100 80GB GPUs with CUDA 12.4; backbone inference uses provider APIs.

Runtime instrumentation labels ℓ_{obs} from exposed acceptance state. To assign ℓ_{exec} to every proposal, including a blocked one, the evaluator executes that candidate in an isolated clone of the same pre-decision benchmark state and discards the clone; no counterfactual observation reaches the monitor or backbone. The live trajectory executes only admitted actions. Oracle Mask alone receives this privileged label before deciding, so it is a ceiling rather than a deployable comparator. The 47K event–

Table 1: WebArena results on 812 tasks. Illegal and B-V are measured against full runtime validity ℓ_{exec}; Dead is a trajectory rate; Ctrl. is controller-plus-probe seconds per proposal.

Backbone	Method	Succ. (%)	Illegal (%)↓	B-V (%)↓	Ctrl. (s)	Dead (%)↓
GPT-4o	Unconstrained	18.7 ± 0.8	23.4 ± 1.2	—	—	4.2
	Schema-Constrained	19.2 ± 0.9	16.8 ± 1.0	4.1 ± 0.5	0.03	3.8
	Post-hoc Filter	19.8 ± 0.7	14.7 ± 0.9	2.3 ± 0.4	0.41	3.1
	Hierarchical Ctrl.	20.1 ± 0.8	13.9 ± 1.1	3.7 ± 0.6	0.08	3.5
	FrontierGuard	24.5 ± 0.6	10.7 ± 0.7	1.8 ± 0.3	0.12	1.9
	Oracle Mask	27.3 ± 0.5	0.0	0.0	—	0.8
Claude-3.5	Unconstrained	21.3 ± 0.9	19.8 ± 1.1	—	—	3.6
	Schema-Constrained	21.8 ± 0.8	14.2 ± 0.9	3.5 ± 0.4	0.03	3.2
	Post-hoc Filter	22.4 ± 0.7	11.9 ± 0.8	1.9 ± 0.3	0.38	2.7
	Hierarchical Ctrl.	22.7 ± 0.8	11.2 ± 1.0	3.2 ± 0.5	0.07	2.9
	FrontierGuard	26.4 ± 0.5	8.3 ± 0.6	1.4 ± 0.2	0.11	1.5
	Oracle Mask	29.8 ± 0.4	0.0	0.0	—	0.6

predicate–proposal tuples from 3.2K exploration episodes are stratified by task identity, domain, and seed into 2,240/480/480 fitting/validation/test episodes and 32.9K/7.1K/7.0K tuples. Fitting estimates profiles, event-conditioned transitions, and probe outcomes; validation selects the partition, state cap, and operating point; the untouched 480-episode test partition supplies the calibration results in Table A13. Downstream evaluation tasks are isolated from all three monitor-development partitions. Matched-family experiments fit one monitor per WebArena application and one joint ToolBench monitor. In each leave-one-family-out fold, vocabulary construction, fitting, and selection use only the remaining families; all choices are frozen before evaluation, and every task appears in exactly one withheld fold.

Metrics and uncertainty. Success is the fraction of completed task trajectories. Runtime-illegal rate is the fraction of executed actions with $\ell_{\text{exec}} = 0$; runtime false allow is the fraction of candidates with $\ell_{\text{exec}} = 0$ that are allowed; blocked-valid/false-block is the fraction with $\ell_{\text{exec}} = 1$ that are blocked; query rate is the fraction of proposals probed; dead-end rate is the fraction of task trajectories entering an unrecoverable state; and control latency is wall-clock monitor-plus-probe time per proposal. Fitting loss and decision calibration instead use ℓ_{obs}; comparing these labels with ℓ_{exec} exposes missing or stale state. Headline benchmark estimates are task-weighted within each seed and then averaged across three seeds; each displayed $\pm$ term is the standard error across seeds. Paired differences use 10,000 task-level bootstrap resamples stratified by interface family, preserving task weights, with percentile 95% intervals and a two-sided null test.

Comparators and aggregation. Unconstrained executes each proposal directly. Schema-Constrained restricts decoding to statically valid actions (Polozov et al., 2022). Post-hoc Filter executes, observes, and retries (Yao et al., 2023); both attempts consume budget. Hierarchical Controller exposes benchmark-authored subsets through phase options. Matched FF reads the current predicates; Matched GRU reads the same executed (e_i, f_i) sequence as FrontierGuard, and neither model updates on a blocked proposal. Their learned-module counts differ by less than 1%, while FrontierGuard stores count tables whose size is reported as states. Appendix Table A4 gives the architectures, counts, training, and selection rules. The mechanism comparison shares predicates, tuples, normalized actions, asymmetric loss, thresholds, and query band. All headline and held-out tables are task-weighted.

4.1 From Local Decisions to Completed Trajectories

The important coupling in Table 1 is local validity to trajectory outcome. With GPT-4o, success rises 5.8 points while runtime-illegal actions fall 12.7 points (54.3% relative) and dead ends fall from 4.2% to 1.9%. The paired 95% intervals are $[4.7, 6.9]$ for success and $[-14.2, -11.2]$ for illegal-action rate

Table 2: ToolBench-API results on 340 GPT-4o tasks. Illegal is measured against ℓ_{exec}.

Method	Success (%)	Illegal (%)↓	Query (%)
Unconstrained	42.1 ± 1.3	18.6 ± 1.4	—
Schema-Constrained	44.7 ± 1.1	8.3 ± 0.9	—
Post-hoc Filter	43.8 ± 1.2	10.1 ± 1.0	—
FrontierGuard	48.2 ± 0.9	5.7 ± 0.6	8.4
Oracle Mask	54.8 ± 0.7	0.0	—

Table 3: Representation test on 812 WebArena GPT-4o tasks. Inputs, supervision, risk, thresholds, and query budget are shared. Illegal and B-V use ℓ_{exec}; Ctrl. is seconds per proposal.

Method	Success (%)	Illegal (%)↓	B-V (%)↓	Query (%)	Ctrl. (s)
Matched FF	22.4 ± 0.7	13.8 ± 0.8	2.4 ± 0.3	8.5 ± 0.4	0.08
Matched GRU	23.2 ± 0.7	12.2 ± 0.7	2.1 ± 0.3	8.4 ± 0.4	0.15
FrontierGuard	24.5 ± 0.6	10.7 ± 0.7	1.8 ± 0.3	8.4 ± 0.4	0.12

(both $p < 0.001$). Claude-3.5 preserves the ranking: FrontierGuard reaches 26.4% success and 8.3% illegal actions, so the effect is not tied to one proposal model.

ToolBench-API isolates the same gap after schemas have already removed malformed calls. FrontierGuard adds 3.5 success points over Schema-Constrained (paired 95% CI $[2.1, 4.9]$, $p < 0.001$) and removes 2.6 runtime-illegal points. Argument availability and response dependencies therefore leave a dynamic frontier after syntax is fixed; the corresponding monitors average 23.7 states.

4.2 Behavior When Grounding Weakens

Selector renaming leaves FrontierGuard at 21.1% success versus 13.7% for Schema-Constrained (Appendix Table A11). More revealingly, delayed observations make query use rise before runtime false allows: at 5 seconds the pair is 14.2% queries / 7.3% false allows, and at 10 seconds it is 19.6% / 11.8% (Appendix Table A12). The controller thus reacts to missing evidence through its designated information channel instead of only increasing blocks. Under the hardest reported stale and hidden-permission settings, its success/false-allow pair also remains ahead of the static reference and both same-predicate learned controls (Appendix Table A5).

4.3 What the Finite Quotient Contributes

Current predicates account for most of the control gain: Matched FF reaches 22.4% success and 13.8% illegal actions. Recurrence improves those values to 23.2% and 12.2%. At the same 8.4% query rate, replacing generic recurrent memory with the explicit quotient adds another 1.3 success points (paired 95% CI $[0.6, 2.0]$) and removes 1.5 illegal-action points ($[-2.4, -0.6]$). Component interventions agree with that attribution: removing query costs 1.7 success points, removing relational predicates adds 2.5 query points, and capacity improves through 16–32 states before flattening (Appendix Tables A6–A7).

With every fitting choice frozen before the withheld fold, FrontierGuard retains the best pooled success and runtime-illegal estimates in both benchmarks (Table 4). Its pooled success advantage over Matched GRU is 1.4 points on each benchmark, with paired 95% intervals $[0.5, 2.3]$ on WebArena and $[0.2, 2.7]$ on ToolBench-API. The family-withheld aggregate therefore preserves the representation advantage under unseen interface groups. Predicate density helps explain variation in the matched domains: GitLab closes 78.7% of the oracle gap, whereas asynchronous Maps closes 51.2% (Appendix Table A10).

Table 4: Leave-one-family-out transfer. Each task appears in one withheld family; Illegal uses ℓ_{exec}. Entries are task-weighted mean $\pm$ standard error over three seeds.

	WebArena		ToolBench-API	
Method	Success	Illegal↓	Success	Illegal↓
Schema-Constrained	18.0 ± 0.9	17.4 ± 1.1	42.9 ± 1.2	9.2 ± 1.0
Post-hoc Filter	18.6 ± 0.8	15.3 ± 1.0	41.8 ± 1.3	11.4 ± 1.1
Matched FF	20.3 ± 0.8	15.0 ± 0.9	43.5 ± 1.1	8.4 ± 0.8
Matched GRU	21.2 ± 0.8	13.9 ± 0.8	44.2 ± 1.1	7.6 ± 0.7
FrontierGuard	22.6 ± 0.7	12.8 ± 0.8	45.6 ± 1.0	6.9 ± 0.7

4.4 Decision Reliability and Cost

On the untouched monitor-test partition, block decisions attain ECE 0.031 and AUPRC 0.913; the selected threshold gives 96.1% precision at 82.4% recall (Appendix Table A13). Calibration uses ℓ_{obs}, whereas all Illegal, B-V, and stress-test FA values use ℓ_{exec}; the latter therefore include hidden-state disagreement. Mean controller-plus-probe latency is 0.12 seconds per proposal and p95 is 0.34 seconds. Raw proposal denominators imply 6.34 backbone calls per GPT-4o task versus 6.30 for Unconstrained, an increase of 0.04 calls per task; Table A13 separates this call inflation from probe latency.

5 Related Work and Positioning

FrontierGuard draws its state-learning machinery from behavioral abstraction. Classical stochastic automata inference merges compatible prefix-tree states (Carrasco & Oncina, 1994); predictive-state representations identify histories by action-conditioned predictions of future observations (Littman et al., 2001); and bisimulation metrics aggregate states according to reward and transition behavior (Ferns et al., 2004). Our quotient instead preserves an action-indexed executability profile and event-conditioned predicate successors, without modeling the environment's full observation distribution or task value. Selective classification contributes the reject-option view (Geifman & El-Yaniv, 2017); FrontierGuard turns rejection into an interface operation by choosing a read-only probe, updating state from its result, and deciding before a side effect. The novelty lies in this executable-history target and its composition with grounded transitions, asymmetric intervention, and active observation, rather than state merging alone.

Structured web and tool benchmarks provide the machine-readable interaction setting (Zhou et al., 2024; Koh et al., 2024; Deng et al., 2023; He et al., 2024; Drouin et al., 2024; Qin et al., 2024; Li et al., 2023; Lu et al., 2025). Constrained decoding validates membership in authored languages (Polozov et al., 2022; Scholak et al., 2021; Beurer-Kellner et al., 2023). Runtime verification and shielding enforce temporal specifications (Leucker & Schallhart, 2009; Alshiekh et al., 2018); ToolGate tracks symbolic state for authored pre/postconditions (Liu et al., 2026); Progent and AEGIS enforce privilege or safety policies (Shi et al., 2025; Yuan et al., 2026); and Task Shield and NEXUS score task alignment or plan risk (Jia et al., 2025; Hossain et al., 2026). Learned action elimination and perceptual affordances estimate feasibility (Zahavy et al., 2018; Do et al., 2018), while options and finite-state controllers encode planning state (Parr & Russell, 1997; Sutton et al., 1999; Meuleau et al., 1999). Table 5 locates the different intervention contracts. The same-predicate FF and GRU are the direct empirical comparators because they hold the learned target, observations, risk, and query interface fixed while changing the state representation.

The categorical distinction in Table 5 becomes measurable in Table 3: at the same query frequency, the explicit quotient improves both task success and runtime-illegal rate over generic recurrence. The ablations then assign separate effects to information acquisition, asymmetric loss, relational grounding, and state capacity.

Table 5: Closest control families. Src.: legality source; Dyn.: live interface state; Hist.: history-dependent state; Pre: intervention before execution; Q: explicit query/abstention; SE: side-effect-aware evaluation; W/A: web/API-agent evaluation.

Family	Src.	Dyn.	Hist.	Pre	Q	SE	W/A
Constrained decoding	authored	no	no	yes	no	no	partial
Contracts/shields/policies	authored	yes	partial	yes	partial	yes	partial
Learned affordances	learned	yes	no	yes	no	no	no
Action elimination	learned	yes	partial	yes	no	yes	no
Post-hoc repair	mixed	yes	yes	no	no	yes	yes
Options/finite-state control	learned	yes	yes	yes	no	no	no
FrontierGuard	learned	yes	yes	yes	yes	yes	yes

6 Discussion

The 12.7-point drop in GPT-4o runtime-illegal actions coincides with 2.3 fewer dead-end trajectories per hundred tasks and 5.8 more completions, linking the monitor's local target to the unchanged proposal policy's final outcome. The representation ladder localizes the gain: current predicates provide most of it, event history adds more, and the explicit quotient contributes a further 1.3 success points while reducing illegal actions at equal query rate. Querying is not interchangeable with conservative blocking: removing the branch costs 1.7 success points, while stale evidence first increases observation use. Pooled held-out intervals preserve the quotient advantage when complete interface families are excluded from fitting.

The evaluated regime also gives a compact boundary on the claim. Dense structured predicates and inexpensive refreshes produce the strongest effects, including 78.7% oracle-gap closure on GitLab and a 3.5-point gain beyond API schema constraints. Backend-only permissions and long delays increase disagreement between ℓ_{obs} and ℓ_{exec}; provider changes can shift proposals; and finite state trades unbounded memory for inspectable transitions. The stress grid, call accounting, capacity sweep, and held-out folds quantify these effects without making them the paper's primary takeaway.

7 Conclusion

FrontierGuard turns grounded event histories into deterministic executable states and resolves each proposed side effect through `allow`, `block`, or one read-only `query`. On GPT-4o WebArena it cuts illegal actions by 54.3% relative, raises success by 5.8 points, and reduces dead ends from 4.2% to 1.9%. The quotient contributes 1.3 success points and 1.5 illegal-action points beyond a same-predicate GRU at equal query rate, with positive pooled intervals after interface-family withholding. Across 1,152 web and API tasks, learning the boundary between proposal and execution makes structured agents both more valid locally and more effective end to end.

Reproducibility Statement

Sections 2–4 specify the target, monitor, optimization, splits, baselines, budgets, metrics, and aggregation rules. The appendix proves the quotient properties and details predicates, probes, comparator architectures, evaluation labels, state structure, ablations, raw counts, stress slices, calibration, and cost.

AI Use Statement

Generative AI tools assisted with language editing, formal-consistency checks, and manuscript-preparation review of formatting and reference consistency. The authors verified every technical statement, derivation, result, citation, and interpretation and take responsibility for the manuscript.

REFERENCES

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 2669–2678, 2018. doi: 10.1609/aaai.v32i1.11797. URL https://doi.org/10.1609/aaai.v32i1.11797.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7 (PLDI), 2023. doi: 10.1145/3591300. URL https://doi.org/10.1145/3591300.

Rafael C. Carrasco and José Oncina. Learning stochastic regular grammars by means of a state merging method. In *Grammatical Inference and Applications*, volume 862 of *Lecture Notes in Computer Science*, pp. 139–152. Springer, 1994. doi: 10.1007/3-540-58473-0_144.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Thanh-Toan Do, Anh Nguyen, and Ian Reid. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *IEEE International Conference on Robotics and Automation*, pp. 5882–5889. IEEE, 2018. doi: 10.1109/ICRA.2018.8460902. URL https://doi.org/10.1109/ICRA.2018.8460902.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pp. 162–169. AUAI Press, 2004. URL https://mlanthology.org/uai/2004/ferns2004uai-metrics/.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6864–6890. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.371. URL https://aclanthology.org/2024.acl-long.371/.

Elias Hossain, Md Mehedi Hasan Nipu, Tasfia Nuzhat Ornee, Rajib Rana, and Niloofar Yousefi. NEXUS: Structured runtime safety for tool-using llm agents, 2026. URL https://arxiv.org/abs/2607.19356.

Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. The task shield: Enforcing task alignment to defend against indirect prompt injection in llm agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 29680–29697. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1435. URL https://aclanthology.org/2025.acl-long.1435/.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 881–905, 2024. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009. doi: 10.1016/j.jlap.2008.08.004. URL https://doi.org/10.1016/j.jlap.2008.08.004.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Michael L. Littman, Richard S. Sutton, and Satinder P. Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems 14*, pp. 1555–1561, 2001. URL https://proceedings.neurips.cc/paper/2001/hash/1e4d36177d71bbb3558e43af9577d70e-Abstract.html.

Yanming Liu, Xinyue Peng, Jiannan Cao, Xinyi Wang, Songhang Deng, Jintao Chen, Jianwei Yin, and Xuhong Zhang. Toolgate: Contract-grounded and verified tool execution for llms. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9653–9684. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.470. URL https://aclanthology.org/2026.findings-acl.470/.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Priyanka Mandikal and Kristen Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In *Proceedings of the 5th Conference on Robot Learning*, volume 164, pp. 651–661. PMLR, 2022. URL https://proceedings.mlr.press/v164/mandikal22a.html.

Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R Cassandra. Solving pomdps by searching the space of finite policies. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 417–426, 1999. URL https://mlanthology.org/uai/1999/meuleau1999uai-solving/.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. URL https://arxiv.org/abs/2112.09332.

Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems 10 (NIPS 1997)*, volume 10, 1997. URL https://proceedings.neurips.cc/paper/1997/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

Oleksandr Polozov, Gabriel Poesia, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KmtVD97J43e.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.779. URL https://aclanthology.org/2021.emnlp-main.779/.

Tianneng Shi, Jingxuan He, Zhun Wang, Linyu Wu, Hongwei Li, Wenbo Guo, and Dawn Song. Progent: Programmable privilege control for llm agents, 2025. URL https://arxiv.org/abs/2504.11703.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 50208–50232. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24h.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Aojie Yuan, Zhiyuan Su, and Yue Zhao. AEGIS: No tool call left unchecked—a pre-execution firewall and audit layer for ai agents, 2026. URL https://arxiv.org/abs/2603.12621.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/645098b086d2f9e1e0e939c27f9f2d6f-Abstract.html.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rmiwIL98uQ.

A Grounding Schema and Quotient Structure

Table A1 makes the 64-bit grounding interface explicit. For logical test P_j, bits (f_{2j-1}, f_{2j}) encode (value, missing): false is $(0, 0)$, true is $(1, 0)$, and unavailable or inapplicable is $(0, 1)$; $(1, 1)$ is invalid. This separates absent evidence from a negative predicate. The same index carries the analogous web or API test, and the modality in the normalized action disambiguates the interpretation.

At readout time, target binding chooses the object or operation to which target-dependent tests apply. After an admitted action or probe returns, the same binding rule constructs the resulting f_i for the event-conditioned transition. A failed lookup sets the corresponding missingness bits and enters the query band; it never creates a negative observation by default.

A.1 Quotient Properties

Proposition. The relation $\equiv_{\text{exec}}$ in Eq. 2 is an equivalence and a right congruence over realizable event–observation extensions; consequently $v(h)$ is well defined on each quotient class.

Table A1: Complete indexed predicate schema. Each P_j contributes a value bit and a missingness bit, giving 32 logical tests and 64 binary slots.

Family	IDs	Web test	API test
Target role	$P_{1:5}$	target bound; button; input/select; link/navigation; form/dialog/submit	operation bound; operation; argument; resource/read call; request/write call
Visibility / enablement	$P_{6:10}$	exposed in DOM; visible; enabled; read-only; side-effecting	exposed in schema; endpoint available; callable; read-only; side-effecting
ARIA / focus	$P_{11:14}$	focused; selected/checked; expanded; modal active	inapplicable (missing bit set)
Form / type validity	$P_{15:19}$	required fields complete; type valid; constraint valid; value nonempty; form locally valid	required arguments complete; type valid; enum/format valid; body nonempty; request locally valid
Arguments / dependencies	$P_{20:23}$	required input available; upstream value available; dependency satisfied; unique target binding	required argument available; result identifier available; dependency satisfied; unique operation binding
Response / session	$P_{24:28}$	navigation complete; success state; validation error; server/rate error; session/auth valid	call complete; success response; client error; server/rate error; session/auth valid
Relations	$P_{29:32}$	in active form; in active modal; sibling/label relation; parent/control relation	argument-of; response-of; dependency-of; result-of
Normalized event	$(\text{modality}, \text{operation}, \text{target role}, \text{side-effect class})$		

Table A2: Read-only probe contracts. Every successful probe returns structured fields, appends one (r, f') event, and permits one new decision for the original proposal.

Probe	Availability	Returned fields
`inspect_element`	Web; bound DOM/accessibility target	role, visibility, enablement, ARIA/focus, local validity, parent/modal relations
`refresh_state`	Web or API; refresh supported	current DOM/accessibility snapshot or latest structured response/session fields
`list_schema`	API; operation identifier bound	operation, required arguments, types/formats, mutability, dependencies
`read_only_probe`	API; endpoint declared non-mutating	response class, returned identifiers, dependency and session fields

Proof. Every history h induces a total function $\mathrm{Res}_h : (\mathcal{E} \times \mathcal{F})^* \to \mathcal{V} \cup \{\dagger\}$. Equality of functions is reflexive, symmetric, and transitive. Now suppose $h \equiv_{\mathrm{exec}} h'$ and let $\sigma = (e, f)$. If $h \circ \sigma$ is realizable, then $\mathrm{Res}_h(\sigma) \neq \dagger$; residual equality makes $h' \circ \sigma$ realizable as well. For every continuation u,

$$\mathrm{Res}_{h\circ\sigma}(u) = \mathrm{Res}_h(\sigma u) = \mathrm{Res}_{h'}(\sigma u) = \mathrm{Res}_{h'\circ\sigma}(u).$$

Thus $h \circ \sigma \equiv_{\mathrm{exec}} h' \circ \sigma$, with the reverse realizability implication following symmetrically. Taking u to be the empty string gives $v(h) = v(h')$. □

The main-text example in Figure 1 exposes phase-sensitive control: shipping separates the same submit role across incomplete and valid forms, while payment shows a probe resolving ambiguity before the pay event. Across domains, the fitted structures remain compact, with no standard-condition domain exceeding 31 states and query mass staying between 8.3% and 9.4% (Table A3).

Table A3: Predicate-grounded automaton characteristics by domain. Active predicates are non-zero features out of 64.

Domain	$\lvert Q\rvert$	Active pred.	Sparse (%)	Allow (%)	Block (%)	Query (%)
Shopping	28	41 / 64	35.9	68.2	23.4	8.4
Forums	22	36 / 64	43.8	71.5	19.7	8.8
CMS	31	47 / 64	26.6	64.8	26.1	9.1
Maps	19	29 / 64	54.7	73.4	17.2	9.4
GitLab	26	44 / 64	31.3	66.9	24.8	8.3
ToolBench avg.	24	38 / 64	40.6	70.1	21.5	8.4

B Evaluation Protocol and Comparator Specifications

ToolBench-API construction. We use the official ICLR 2024 ToolBench release and call the evaluated slice ToolBench-API. Starting from its test queries, we retain every task whose complete reference route uses REST operations with a machine-readable schema, whose required service is available in the frozen benchmark runtime, and whose terminal condition has an executable checker; duplicate task identifiers are resolved before evaluation. This outcome-independent rule yields 340 tasks in 47 original categories. The leave-one-family-out grouping maps weather and geolocation services to Weather/Geo, communication and social-network services to Social Media, shopping and business services to E-commerce, banking and market services to Finance, and developer, data, and utility services to Developer Tools. The same mapping is fixed for fitting, bootstrap stratification, and reporting.

Success is the benchmark checker's terminal-state decision after normalized comparison of required response fields; controller confidence and model-generated text never enter that judgment. Each method–seed run begins from the same service snapshot and fixture stream. Responses are replayed from that seed's frozen stream, so method order cannot change a later method's state; a transport failure is replayed once from the pre-call snapshot, and a repeated failure counts as an unsuccessful call. The API family assignment, evaluator, and response stream are fixed before any controller is fitted.

Counterfactual validity labels. At each proposal the evaluator forks the current benchmark state. The candidate runs once in the fork to produce ℓ_{exec}, after which the fork is discarded. An admitted candidate also runs on the live branch; a blocked candidate does not. The controller receives only exposed predicates and its own live-branch outcomes. Oracle Mask is the sole method allowed to consult the fork label before acting. This procedure supplies false-block denominators without turning rejected actions into live side effects.

Matched FF and GRU optimize the weighted Bernoulli loss with AdamW, batch size 256, learning rate selected from $\{3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$, weight decay from $\{0, 10^{-4}\}$, and early stopping on validation risk with a 100-epoch limit. Calibration, thresholds, and query-budget selection then use the same validation units as FrontierGuard. The GRU consumes only executed side effects and probes; a block repeats its recurrent state. Hierarchical Controller derives page and endpoint phases

Table A4: Comparator implementation and selection. Parameter counts exclude the shared 16-dimensional normalized-action embedding table; including it preserves the FF/GRU difference because the vocabulary is identical.

Comparator	State, readout, and tuning	Learned parameters
Matched FF	Current 64-bit predicates plus action embedding; affine–ReLU–affine head of width 95	7,791
Matched GRU	64-bit predicates plus executed-event embedding; width-20 GRU; recurrent state plus proposal embedding; width-44 affine–ReLU–affine head	7,793
FrontierGuard	Deterministic event-conditioned state; Laplace-smoothed action counts; MDL/risk selection over state caps	count table
Hierarchical Ctrl.	Authored phase option from page/endpoint class and interaction mode; option exposes its authored action subset; no query branch	0

from benchmark-authored structure and applies the same one-step charge and feedback protocol as FrontierGuard. These controls separate representation from observations, labels, action keys, risk weighting, and operating-point selection.

C Mechanism and Component Detail

The stress comparison applies the same learned representations from Table 3 after perturbing the currency or visibility of their inputs. It therefore tests whether finite state remains useful when the current predicate vector is least reliable.

Table A5: Mechanism-matched stress evaluation. Each cell is success / runtime false-allow rate against ℓ_{exec} (%). Matched controls report three-seed means; FrontierGuard also reports standard error.

Method	Delay 10s	Stale 80%	Hidden permissions
Matched FF	16.5 / 17.0	14.9 / 19.3	16.7 / 15.9
Matched GRU	18.1 / 14.1	16.3 / 16.8	18.1 / 13.9
FrontierGuard	19.7 ± 0.7 / 11.8 ± 0.9	17.8 ± 0.8 / 14.7 ± 1.0	19.2 ± 0.7 / 12.1 ± 0.9

Table A6: FrontierGuard component ablations on WebArena with GPT-4o. Illegal and B-V use ℓ_{exec}; values are mean $\pm$ standard error over three seeds.

Variant	Success (%)	Illegal (%)↓	B-V (%)↓	Query (%)
Full FrontierGuard	24.5 ± 0.6	10.7 ± 0.7	1.8 ± 0.3	8.4
No query	22.8 ± 0.7	13.4 ± 0.8	3.2 ± 0.4	0
Symmetric loss	23.5 ± 0.7	12.4 ± 0.7	3.0 ± 0.4	7.7 ± 0.4
No relation predicates	23.4 ± 0.7	12.7 ± 0.8	2.2 ± 0.3	10.9 ± 0.5

Removing query costs 1.7 success points and raises blocked-valid rate by 1.4 points, while removing relational predicates increases query use by 2.5 points. Table A7 shows a credible capacity profile: performance improves into the 16–32-state regime and then flattens as MDL suppresses unnecessary states.

Table A7: State-capacity sweep. Each result reports success (%) / runtime-illegal rate (%) / mean MDL-selected states.

State cap	Success / illegal / selected states
8	22.9 ± 0.8 / 13.0 ± 0.8 / 8.0
16	24.1 ± 0.7 / 11.4 ± 0.7 / 15.2
32	24.5 ± 0.6 / 10.7 ± 0.7 / 25.2
64	24.3 ± 0.7 / 10.9 ± 0.7 / 31.4

D Raw Counts and Paired Statistics

The task numerators in Table A8 reproduce the reported three-seed success means. Table A9 uses a separate denominator for each estimand: executed actions for runtime illegality, class-conditional candidate opportunities for false allow and false block, proposals for queries, and tasks for dead ends. The class-conditional candidate denominators therefore need not sum to proposal counts.

Table A8: Success counts by seed. Every denominator is the task count for that benchmark.

Setting and method	Seed 1, seed 2, seed 3
WebArena GPT-4o Unconstrained	141/812, 152/812, 163/812
WebArena GPT-4o FrontierGuard	191/812, 199/812, 207/812
WebArena Claude-3.5 Unconstrained	160/812, 173/812, 186/812
WebArena Claude-3.5 FrontierGuard	207/812, 214/812, 222/812
ToolBench Schema-Constrained	145/340, 153/340, 158/340
ToolBench FrontierGuard	158/340, 165/340, 169/340

Table A9: Decision and trajectory counts by seed against ℓ_{exec}. Ill.: illegal/executed actions; B-V/FB: false blocks/runtime-valid candidates; FA: false allows/runtime-invalid candidates; QR: queries/proposals; Dead: dead ends/tasks.

Setting and method	Seed-level numerator/denominator triples
WA GPT-4o Unconstrained	Ill. 1108/5200, 1193/5100, 1288/5050; Dead 30/812, 34/812, 38/812
WA GPT-4o FrontierGuard	Ill. 456/4800, 510/4770, 564/4740; B-V/FB 53/4100, 73/4050, 92/4000; FA 46/1300, 54/1280, 62/1260; QR 400/5200, 433/5150, 464/5100; Dead 14/812, 15/812, 17/812
WA Claude-3.5 Unconstrained	Ill. 931/5200, 1020/5150, 1107/5100; Dead 25/812, 29/812, 34/812
WA Claude-3.5 FrontierGuard	Ill. 358/4900, 403/4850, 446/4800; B-V/FB 44/4200, 58/4150, 72/4100; Dead 10/812, 12/812, 14/812
ToolBench Schema-Constrained	Ill. 163/2400, 195/2350, 225/2300
ToolBench FrontierGuard	Ill. 108/2300, 129/2260, 149/2220; QR 193/2500, 206/2450, 218/2400

E Domain and Robustness Slices

Domain slices connect the aggregate effect to predicate coverage, while the drift and dynamic-stress interventions separate surface change from stale or hidden runtime state. The former preserves the ranking broadly; the latter increases the observability gap and elicits more queries.

Table A10: Per-domain GPT-4o results. Illegal uses ℓ_{exec}; gap closure is $(\text{FrontierGuard} - \text{Unconstrained})/(\text{Oracle} - \text{Unconstrained})$.

Domain	Unconstr. (%)	Schema (%)	FrontierGuard (%)	Oracle (%)	Illegal (%)↓	Gap closure (%)
WA-Shopping	20.3	20.6	26.8	28.8	8.4	76.5
WA-Forums	17.1	17.8	22.1	25.6	12.3	58.8
WA-CMS	19.4	19.9	25.3	27.7	9.7	71.1
WA-Maps	16.2	17.1	20.4	24.4	14.8	51.2
WA-GitLab	20.5	20.6	27.9	29.9	8.3	78.7
TB-Weather	45.8	48.3	53.7	57.1	3.2	69.9
TB-Social	39.7	42.9	46.1	51.8	6.8	52.9
TB-Commerce	43.2	45.8	49.8	53.6	4.5	63.5
TB-Finance	38.4	41.4	44.2	49.7	7.9	51.3
TB-DevTools	43.4	45.1	51.2	55.6	4.1	63.9

Table A11: WebArena GPT-4o success under interface drift; parentheses show change from each method's standard condition.

Method	CSS relayout	Selector rename	Widget replace	Partial observability
Unconstrained	17.9 (−0.8)	16.1 (−2.6)	15.4 (−3.3)	14.8 (−3.9)
Schema-Constrained	18.1 (−1.1)	13.7 (−5.5)	12.9 (−6.3)	13.2 (−6.0)
Post-hoc Filter	18.9 (−0.9)	15.2 (−4.6)	14.1 (−5.7)	14.7 (−5.1)
FrontierGuard	23.8 (−0.7)	21.1 (−3.4)	19.8 (−4.7)	20.3 (−4.2)

Table A12: Dynamic WebArena stress with GPT-4o. FA and FB use ℓ_{exec}; Schema-Constrained is the static-state reference because each intervention perturbs dynamic observability. All entries are percentages.

	FrontierGuard				Schema-Constrained			
Condition	FA↓	FB↓	QR	Succ.	FA↓	FB↓	QR	Succ.
Standard	4.2	1.8	8.4	24.5	6.8	4.1	—	19.2
Delayed 2s	5.1	2.0	10.7	23.8	9.4	5.6	—	17.3
Delayed 5s	7.3	2.4	14.2	22.1	14.2	7.8	—	14.8
Delayed 10s	11.8	3.1	19.6	19.7	21.7	10.3	—	11.2
Stale obs. 50%	8.4	2.7	16.3	21.4	16.3	8.2	—	13.9
Stale obs. 80%	14.7	4.2	23.8	17.8	24.8	11.7	—	9.6
Hidden permissions	12.1	2.3	18.9	19.2	19.4	7.1	—	12.4
Modal race	9.6	3.8	15.7	20.9	17.8	9.3	—	13.1

F CONTROL BOUNDARY AND SYSTEMS COST

The protected object is a side-effecting web or API proposal before execution (Algorithm 1); the backbone remains free to emit any action in the exposed language. FrontierGuard trusts only the predicate extractor, current structured view, and named read-only probes for observable legality. The Matched GRU comparison isolates memory and representation, while delayed updates, hidden permissions, and held-out families test this boundary under degraded or unseen inputs (Tables 3, A12, and 4).

Table A13 joins calibration and cost at the selected operating point. Latency excludes provider generation, while the call rows expose the re-proposal cost that controller-only timing would miss. Probe shares pool both benchmarks because `list_schema` and `read_only_probe` are API-only contracts; the backbone-call rows are WebArena-specific.

Table A13: Decision reliability, controller cost, and probe mix. Calibration uses the untouched monitor-test split; probe shares pool both benchmarks; calls/task are WebArena GPT-4o values.

Quantity	Primary value	Secondary / p95
All proposals	0.12 s mean	0.34 s p95
`inspect_element`	42.1%	0.07
`refresh_state`	31.8%	0.24
`list_schema`	15.7%	0.13
`read_only_probe`	10.4%	0.41
Allow calibration	ECE 0.023	AUPRC 0.927
Block calibration	ECE 0.031	AUPRC 0.913
Selected block point	96.1% precision	82.4% recall
Unconstrained backbone calls/task	6.30	—
FrontierGuard backbone calls/task	6.34	—

The calibration statistics complement the trajectory results: the controller's low false-block rate is achieved by routing uncertain proposals to explicit observation rather than by suppressing a broad region of candidate actions.