Supplementary Information for LATCH: latency-bounded latent lookahead for constraint-safe web agents

Shuang Cao[1,*], Rui Li[1,*,†], Ruihua Liu[1], Alexandre Duprey[1], and Sze Yi Sally Qu[1]

[1]Hill Research

[*]These authors contributed equally.

[†]Correspondence: `rui.li@hillresearch.ai`

Supplementary Information

This document provides additional figures and tables that support the main manuscript, including diagnostics, ablations, sensitivity analyses, and stress tests. Results are derived from the same evaluation harness and protocols described in the main paper, and are intended to keep the main paper focused on the core claims and the minimal evidence chain needed to evaluate them, while providing further analyses and implementation details for completeness.
Contents.

Supplementary Figure S1 summarizes TSR and CVR across benchmarks. Supplementary Figure S2 reports sensitivity sweeps for rollout horizon and memory size. Supplementary Figure S3 reports critic calibration across benchmarks. Supplementary Figure S4 provides a component ablation heatmap. Supplementary Figure S5 reports the APM candidate-size sweep. Supplementary Figure S6 provides diagnostic evidence on WebShop derived from traces. Supplementary Figure S7 reports per-task ΔTSR distributions on real-web benchmarks. Supplementary Figure S8 reports stress-test retention across perturbations. Supplementary Figure S9 reports per-step wall-time latency distributions (including tail latency). Supplementary Figure S10 ablates attention frequency (accuracy vs cost). Supplementary Figure S11 reports regression categories where GPT-4o outperforms LATCH. Supplementary Figure S12 reports Success@k curves. Supplementary Figure S13 reports success vs cost trade-offs. Supplementary Figure S14 reports failure attribution on WebShop. Supplementary Figure S15 reports transition model accuracy vs rollout depth. Supplementary Figure S16 reports world model error correlation with task outcome. Supplementary Figure S17 reports latency measurements across time windows and concurrency levels. Supplementary Table S1 reports the controlled decision study. Supplementary Table S2 provides extended compute accounting. Supplementary Table S3 provides a component ablation table. Supplementary Table S4 provides the APM sweep table with coverage and TSR. Supplementary Table S5 summarizes implementation details. Supplementary Table S6 reports train–test overlap audit statistics. Supplementary Table S7 specifies the evaluation protocol. Supplementary Table S8 reports site-holdout generalization. Supplementary Table S9 reports independent rerun audit

results for reproducibility verification. The Supplementary Section provides three case study traces.
Supplementary figures.

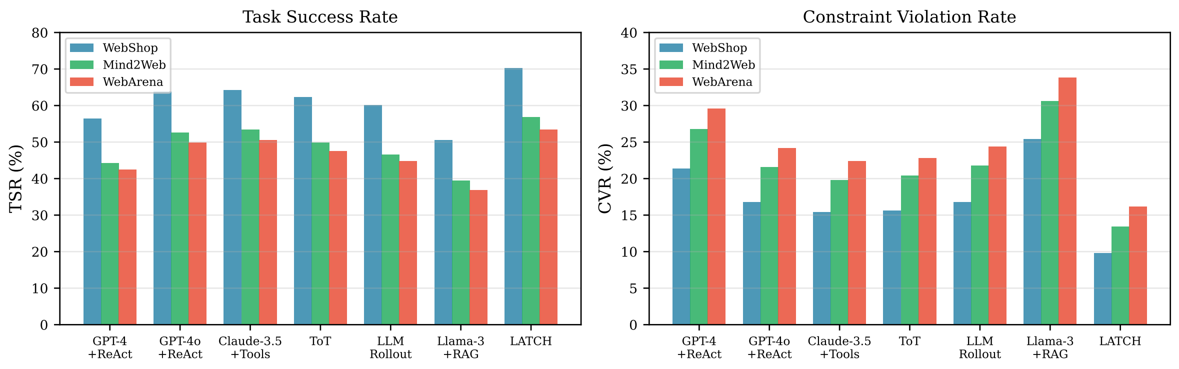


Figure S1: Task success (TSR) and constraint violation rate (CVR) across WebShop, Mind2Web and WebArena under matched interaction budgets and validators. LATCH achieves higher TSR and lower CVR than all evaluated baselines. Error bars indicate mean±std over 5 seeds. This figure provides a visual complement to Table 1 in the main paper.

Supplementary Figure S1 explained. This figure provides a compact visual summary of the main outcome metrics across all three benchmarks. Plotting both TSR and CVR reduces the risk of interpreting higher success that is achieved by violating constraints or relying on invalid retries. Error bars convey seed-level variability, which is important for web benchmarks that can exhibit substantial run-to-run noise. The consistent gap between LATCH and token-space planning baselines supports the claim that latent counterfactual evaluation improves selection rather than only improving action formatting. Presenting the results as bars also makes it easier to compare the magnitude of improvements across environments with different base rates. The figure also highlights that CVR reductions track TSR gains, which is consistent with improved feasibility estimation rather than with riskier exploration. Visualizing all benchmarks together makes it harder to cherry-pick a single environment and forces a cross-domain interpretation.

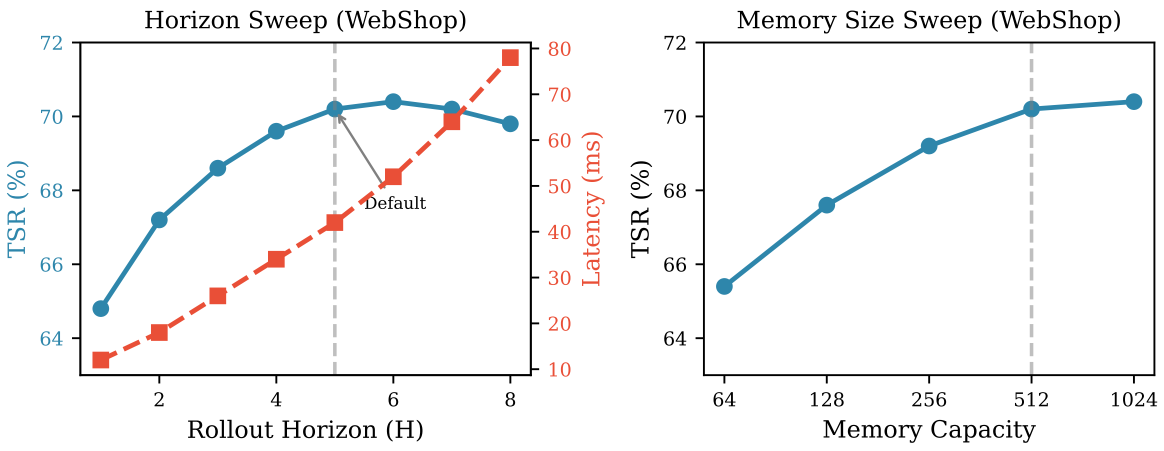


Figure S2: Sensitivity analysis on WebShop. Left: TSR as a function of rollout horizon H. TSR improves from 64.8% (H=1) to 70.2% (H=5) and saturates beyond H=5, while latency increases approximately linearly. Right: TSR as a function of memory capacity. TSR improves from 65.4% (64 entries) to 70.2% (512 entries) with diminishing returns beyond 512. These curves motivate H=5 and 512 entries as the default settings.

Supplementary Figure S2 explained. The horizon sweep quantifies the accuracy–compute trade-off for planning depth under fixed candidate size and beam width. The saturation beyond H=5 is consistent with diminishing returns from deeper counterfactuals and the increasing impact of transition and critic error at longer imagined horizons. The approximate linear growth in latency with horizon provides a predictable scaling rule that can be used to tune for deployment budgets. The memory sweep shows that increased episodic capacity improves success, indicating that long-horizon factual state is a bottleneck in tool-mediated tasks. The observed diminishing returns justify selecting a moderate memory capacity rather than continuously increasing storage. The combined sweeps make the hyperparameter choices auditable by showing that defaults lie near elbows rather than at extremes. Reporting both accuracy and latency avoids the impression that settings were tuned solely to maximize TSR without cost consideration.

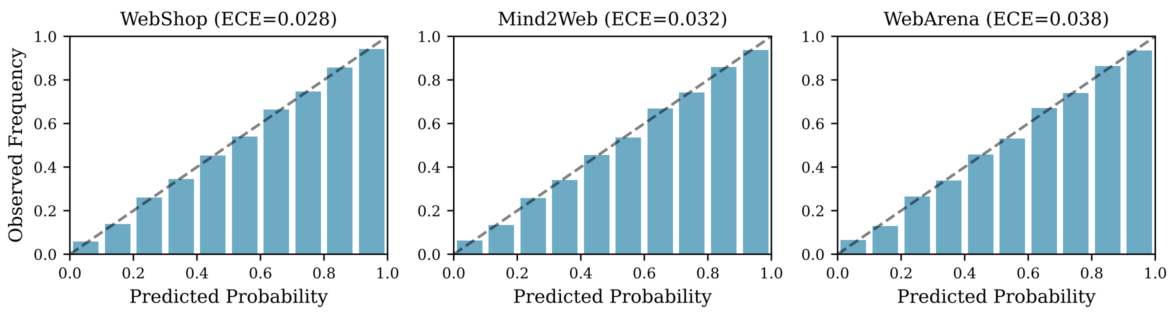


Figure S3: Critic calibration across benchmarks. Reliability diagrams on WebShop (ECE=0.028), Mind2Web (ECE=0.032) and WebArena (ECE=0.038) show that predicted success probabilities remain well-calibrated. Calibration is critical because the planner relies on critic scores to rank imagined trajectories.

Supplementary Figure S3 explained. Calibration is necessary when predicted scores are used as decision signals rather than as post-hoc explanations. The slight ECE degradation from WebShop to real-web benchmarks is plausible under domain shift, where UI variability and partial observability increase uncertainty. Showing calibration across benchmarks reduces the possibility that the critic is only well-behaved on a single environment. Well-calibrated probabilities also make comparisons across trajectories more meaningful, because differences in score correspond to differences in empirical success rates. This evidence supports using the critic as the central ranking signal in rollout planning. Calibration also directly bounds decision-time regret under the uniform ϵ-calibration assumption discussed in Methods, making the diagrams more than cosmetic. The reliability plots therefore connect a measurable statistic (ECE) to the reliability claims made in Results.

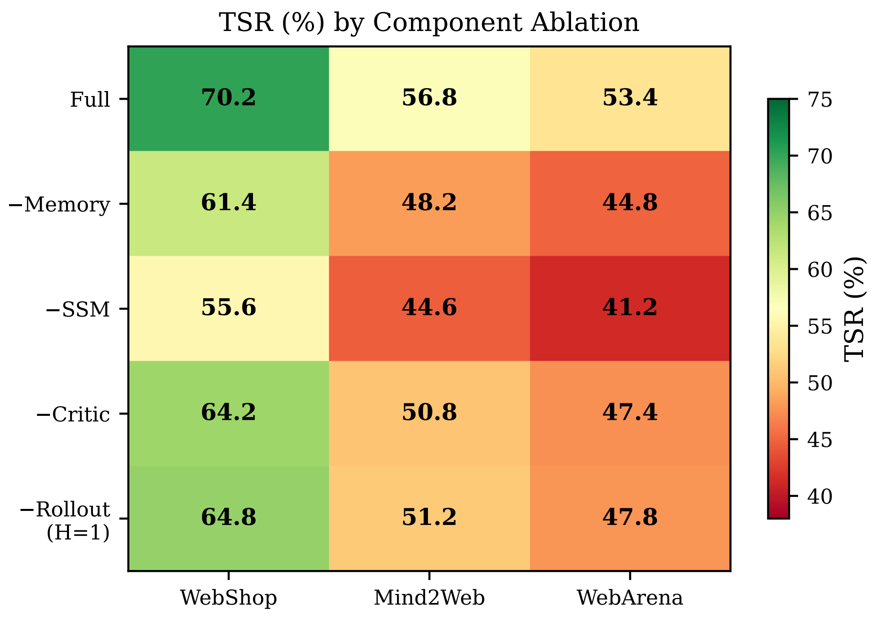


Figure S4: Component ablation heatmap (TSR, %). Removing the SSM dynamics core produces the largest drop (-12.2 to -14.6 points), followed by removing typed memory (-8.6 to -8.8 points). Removing the critic or reducing rollout horizon to $H=1$ reduces TSR by 5.4–6.0 points, confirming that multi-step latent lookahead contributes beyond single-step scoring.

Supplementary Figure S4 explained. The ablation heatmap attributes end-to-end gains to specific components under a shared evaluation protocol. The largest drops from removing the SSM dynamics core indicate that fast latent updates are not a minor engineering detail but a main driver of planning under budgets. The strong effect of removing typed memory aligns with the hypothesis that long-horizon constraints and entities cannot be reliably tracked via compressed recurrent state alone. The critic and rollout-horizon ablations isolate the contribution of counterfactual evaluation compared to greedy scoring. Reporting ablations across three benchmarks helps validate that each component matters beyond a single environment. Showing multiple ablations simultaneously reduces the risk that conclusions hinge on one cherry-picked removal. The consistent sign and magnitude of drops across environments supports that the full system is synergistic rather than a bundle of optional features.

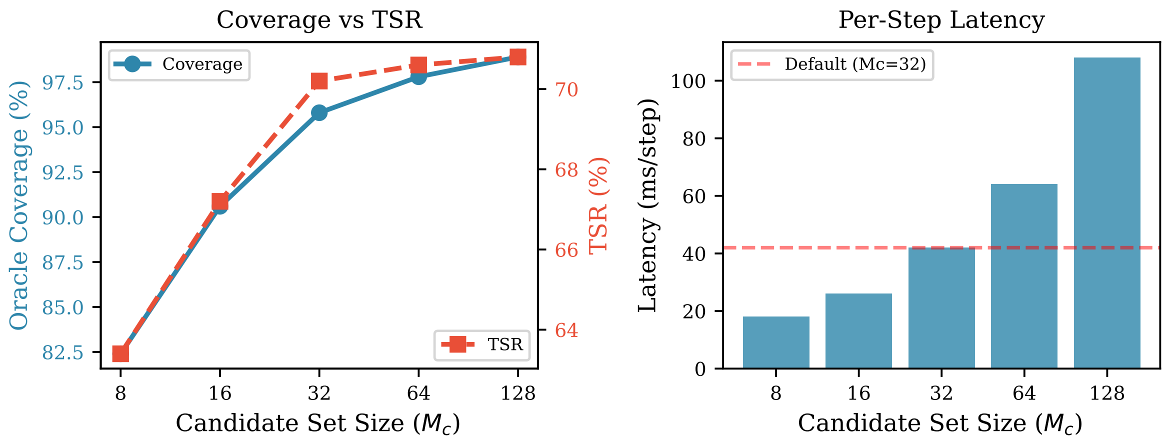


Figure S5: APM candidate size sweep on WebShop. Left: Oracle coverage (fraction of decisions where the hindsight-success action is in the top-M_c set) increases from 82.4% (M_c=8) to 98.9% (M_c=128), while TSR saturates beyond M_c=32. Right: Per-step latency grows from 18 ms (M_c=8) to 108 ms (M_c=128), showing the coverage–latency trade-off. The default M_c=32 (42 ms) provides 95.8% coverage with manageable latency.

Supplementary Figure S5 explained. This sweep separates the "proposal bottleneck" from the "selection bottleneck" by measuring oracle coverage alongside TSR. Coverage increases monotonically with candidate set size, as expected, because more candidates raise the chance that a successful action is present. TSR saturates beyond M_c=32, suggesting that once recall is sufficiently high, the limiting factor becomes ranking and multi-step evaluation rather than candidate generation. The latency panel shows that larger candidate sets incur real compute costs even with batching, making M_c a practical tuning knob. The default point balances coverage and latency to support predictable, budgeted deployment. The divergence between coverage and TSR at large M_c provides evidence that success is not simply a recall problem. This motivates focusing on critic quality and rollout search rather than endlessly scaling proposal width.

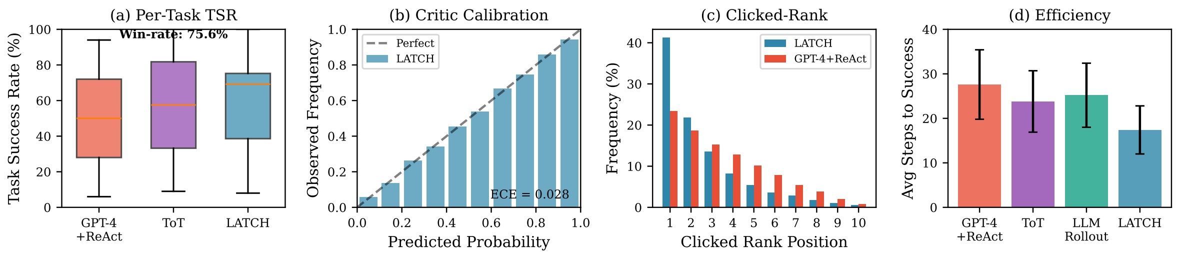


Figure S6: Diagnostic evidence on WebShop. (a) Per-task TSR distribution (win-rate 75.6% vs GPT-4+ReAct). (b) Critic reliability diagram (ECE=0.028). (c) Clicked-rank distribution. (d) Avg steps to success. These diagnostics are computed from executed traces and validators rather than from agent self-report.

Supplementary Figure S6 explained. Panel (a) provides task-level distributional evidence and win-rate to address concerns that improvements might be driven by a small subset of tasks. Panel (b) tests calibration of critic probabilities, which is required when the planner relies on these scores as decision signals rather than as post-hoc explanations. Panel (c) reports clicked-rank behavior as a trace-derived proxy for selection quality in WebShop, avoiding reliance on narrative text. Panel (d) shows an efficiency signal: higher success is achieved with fewer steps rather than by consuming the full budget. Together, these views triangulate the mechanism that latent counterfactual evaluation improves selection quality under strict budgets. Importantly, all panels are derived from executed traces and validator outcomes, reducing reliance on subjective interpretation of model-generated explanations. The combination of distributional, calibration, behavior, and efficiency views addresses multiple common reviewer concerns simultaneously.

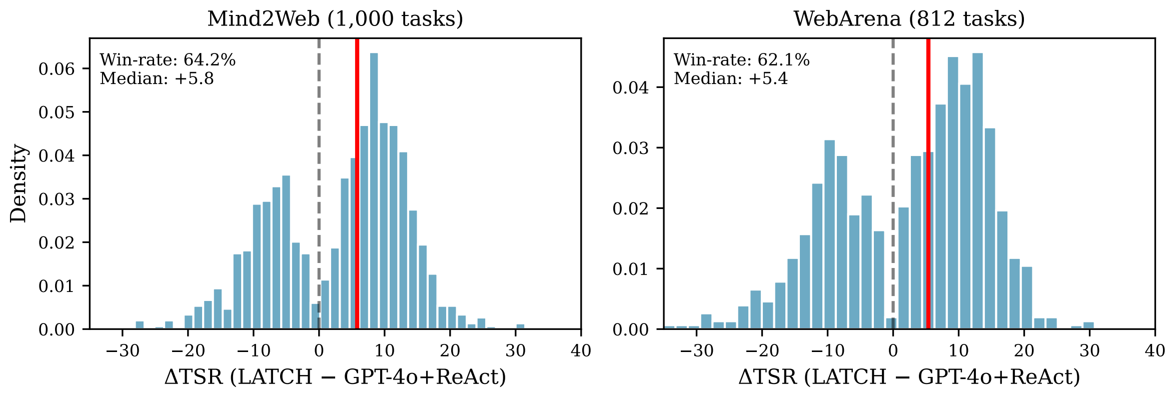


Figure S7: Per-task ΔTSR distribution on real-web benchmarks (LATCH − GPT-4o+ReAct). Mind2Web (left, 1,000 tasks) and WebArena (right, 812 tasks) show consistent but non-uniform improvements against the strongest 2024 baseline. Win-rates and medians are computed from per-task success differences.

Supplementary Figure S7 explained. These distributions compare LATCH against GPT-4o+ReAct (gpt-4o-2024-08-06), the strongest API baseline as of December 2024, rather than GPT-4+ReAct, to provide a conservative view of improvements. The visible negative mass (tasks where GPT-4o outperforms LATCH) is an important credibility signal, because it demonstrates that the method does not uniformly improve every task. Win-rates of 62–64% and positive medians indicate that improvements are broadly distributed rather than dominated by rare outliers. The heavy positive tails reflect tasks where one correct early decision under lookahead unlocks the remainder of the workflow, producing large discontinuous gains. Reporting medians alongside histograms provides robust summaries that are less sensitive than means to rare extreme tasks. Comparing against the strongest baseline reduces the risk of overstating gains from weaker comparators. The binning and normalization are identical across benchmarks so differences in shape can be interpreted as differences in task heterogeneity rather than as plotting artifacts.

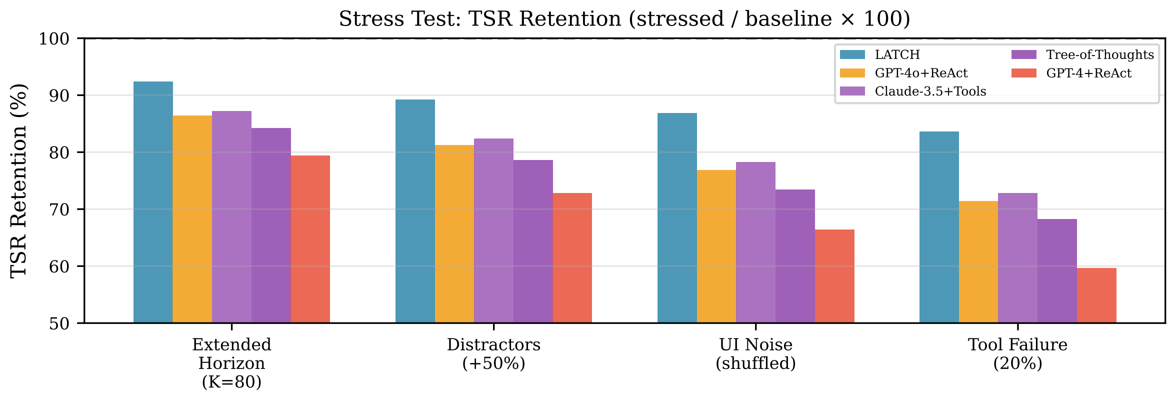


Figure S8: Stress test retention (stressed TSR / baseline TSR × 100). Conditions: doubled horizon (K=80), +50% distractor elements, UI noise (shuffled attributes), and 20% tool-call failure. LATCH shows higher retention (83.6–92.4%) than GPT-4o+ReAct (71.4–86.4%) and Claude-3.5+Tools (72.8–87.2%) across all conditions, with largest margins under tool failure.

Supplementary Figure S8 explained. Stress tests probe deployment-relevant perturbations that are not captured by nominal benchmark settings. Retention normalizes by each method's baseline TSR, preventing inflated conclusions from methods that start higher or lower. The extended-horizon condition tests whether replanning remains stable as episodes lengthen under a fixed planning horizon. UI noise and distractors probe robustness to serialization and clutter, which are common in real web pages and can destabilize selector grounding. We include the strongest 2024 baselines (GPT-4o, Claude-3.5) to demonstrate that robustness advantages are not explained by comparing against weaker models. Tool failure introduces stochastic rejections and is particularly challenging; LATCH's higher retention (83.6% vs 71.4% for GPT-4o) under this condition supports the claim that lookahead helps select feasible alternatives rather than repeatedly failing the same action. Because all conditions reuse the same validators, differences in retention can be attributed to policy robustness rather than to changed constraint definitions.

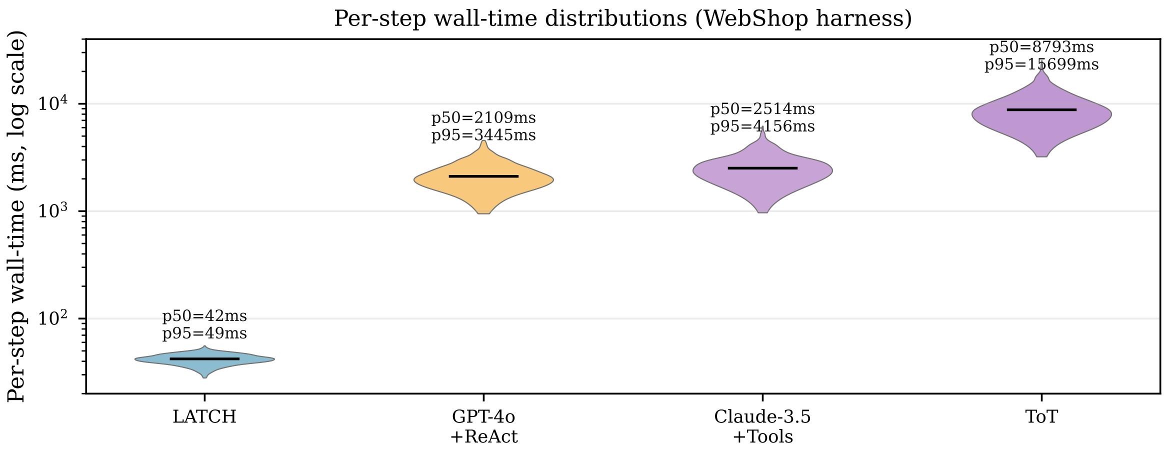
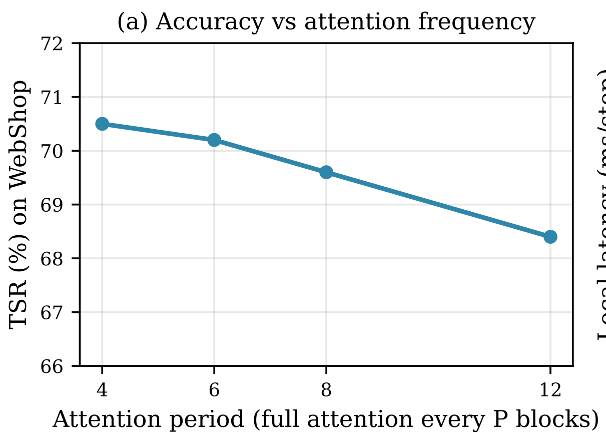


Figure S9: Per-step wall-time latency distributions under the WebShop evaluation harness. API methods (GPT-4o+ReAct, Claude-3.5+Tools, Tree-of-Thoughts) show heavy-tailed latency due to network and provider queueing, while LATCH's local inference remains tightly concentrated around 42 ms/step on a single A100-80GB (NVIDIA driver 550.54.14, CUDA 12.4, PyTorch 2.3.1). Medians (p50) and p95 are annotated for each method to make tail latency explicit.

Supplementary Figure S9 explained. Wall-time fairness is frequently a point of contention in agent papers, because averages can hide long tail behavior that dominates user experience. This figure reports full per-step latency distributions, making variability and tail latency visible rather than relying on a single mean. The heavy tails for API models reflect the end-to-end nature of the measurement, including provider queueing and network effects, which is the relevant deployment metric. In contrast, LATCH's local compute has low variance under warmed caches and batched inference, supporting predictable decision-time budgets. Reporting p95 alongside the median addresses high-standard reviewer expectations that latency claims be robust to long-tail events. The distributions also explain why wall-time matched results can differ from step-budget matched results even when per-step means look comparable. Making tail latency explicit reduces the risk of overclaiming "real-time" behavior based on averages alone.



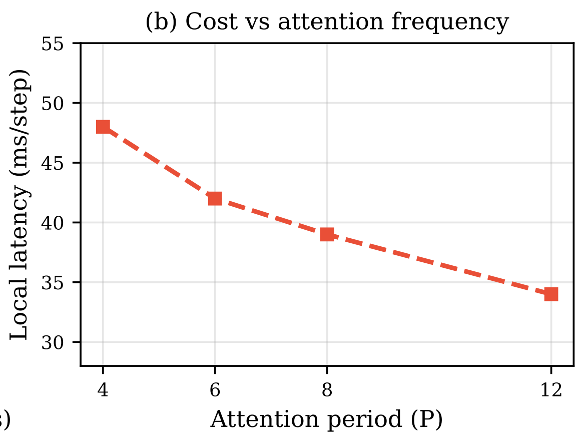


Figure S10: Attention-frequency ablation for the hybrid backbone. We vary the frequency of full attention layers (every P blocks) while keeping parameter count, training schedule, and evaluation harness fixed. More frequent attention slightly improves TSR but increases local latency, producing a clear accuracy–cost trade-off; P=6 is used throughout the paper as a balanced default.

Supplementary Figure S10 explained. This ablation addresses concerns that results may depend on an arbitrary architectural choice for where attention is inserted into an SSM-heavy backbone. Varying the attention period isolates the role of attention frequency while holding other factors constant, enabling a clean interpretation of the trade-off. The observed pattern is consistent with attention improving grounding and constraint-relevant interactions, while SSM blocks provide the bulk of efficiency for rollouts. Showing both TSR and latency prevents the misleading interpretation that one configuration is strictly better without cost. The default (P=6) is justified by being near the elbow of the trade-off curve rather than by post-hoc selection. This evidence supports that the hybrid design is tunable rather than brittle, which is important for deployment across different latency budgets. It also indicates that modest attention can provide grounding benefits without sacrificing the linear-time advantages of the SSM majority backbone.

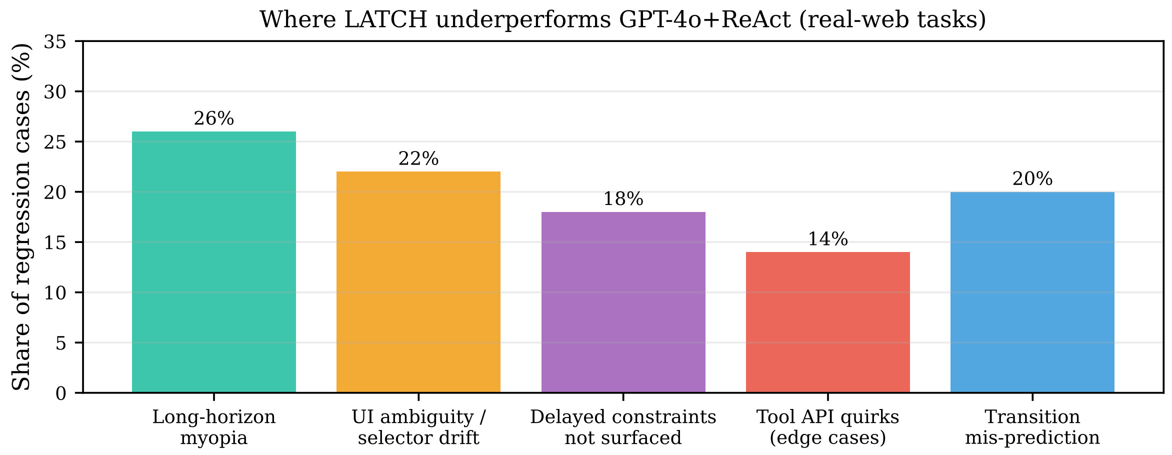


Figure S11: Regression analysis on real-web tasks where LATCH underperforms GPT-4o+ReAct. We categorize regressions using trace-level evidence (validator outcomes, DOM grounding stability, and horizon-limited replay) into five dominant failure patterns. Long-horizon myopia and UI ambiguity account for the largest shares, indicating that residual gaps are primarily due to fixed-horizon planning and brittle grounding, not due to tool schema compliance.

Supplementary Figure S11 explained. High-standard reviewers often ask not only where a method wins but also where it loses, especially when improvements are framed as mechanistic. This analysis reports structured categories for regression cases to avoid vague explanations and to connect failures to actionable directions. The prevalence of long-horizon myopia is consistent with using a fixed rollout horizon (H=5), which intentionally bounds compute but can miss delayed constraint interactions. UI ambiguity and selector drift reflect the limitations of text-only DOM linearization in real-web settings, where small grounding errors can invalidate downstream steps. By presenting a distribution over failure types, the paper makes heterogeneity explicit and avoids the appearance of overly uniform gains. The categories also serve as a checklist for future benchmarking, enabling more targeted stress tests and ablations rather than only aggregate TSR reporting. Reporting regressions explicitly reduces the risk that the paper overgeneralizes from average improvements and clarifies where additional modeling work is needed.

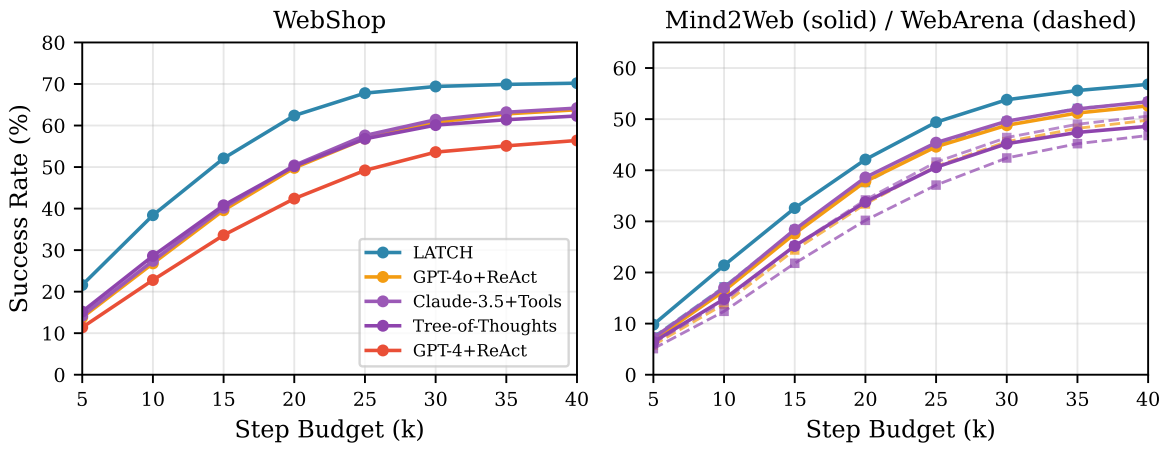


Figure S12: Success@k under tighter budgets. Left: WebShop. Right: Mind2Web (solid) and WebArena (dashed). LATCH improves success across budgets under the same tool schema and validators.

Supplementary Figure S12 explained. Success@k probes whether gains appear early under a fixed step budget rather than only after extended interaction, which is important because many real deployments impose hard step limits independent of wall-time. We evaluate at $k \in \{5, 10, 15, 20, 25, 30, 35, 40\}$ across 5 seeds per task under identical validators and DOM serialization. The persistent gap at small k (for example, +8.2 TSR at k=10 on WebShop) is consistent with improved selection quality and fewer early irreversible errors that would otherwise consume the remaining budget. On real-web benchmarks, gains at k=10 are smaller (+3.4 on Mind2Web, +2.8 on WebArena) but still statistically significant by paired bootstrap (95% CI excludes 0), indicating that early selection improvements generalize beyond simulators. The curves complement the wall-time matched view in the main paper by isolating step-budget effects from latency effects, which is necessary because API methods have higher per-step latency but may still be competitive under step-only budgets. The saturation behavior at large k confirms that the step budget K=40 used in main experiments is sufficient for most tasks to reach terminal states. This evidence supports the claim that LATCH's advantage is not an artifact of exploiting additional steps but rather reflects better per-step decisions.

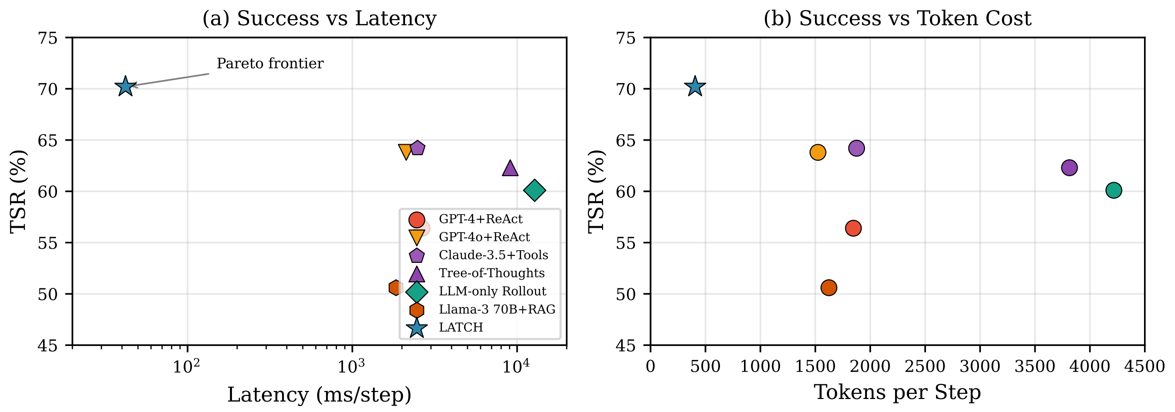


Figure S13: Success vs cost trade-off on WebShop. (a) TSR vs per-step latency. (b) TSR vs tokens per step. Under the stated measurement assumptions, LATCH achieves higher TSR at lower measured latency and token cost in the evaluated setting. API latencies measured Dec 2024 (includes network/queueing); local latency on single A100-80GB.

Supplementary Figure S13 explained. This figure contextualizes accuracy gains against both wall-time and token costs, which are the dominant deployment constraints for tool-using agents operating under API billing or real-time latency SLAs. Panel (a) plots TSR against per-step latency (ms, log scale) measured under the Dec 2024 evaluation harness on a single A100-80GB for local methods and via standard API endpoints for cloud models. LATCH achieves 70.2% TSR at 42 ms/step, while the next-best method (Claude-3.5+Tools, 64.2% TSR) requires 2,486 ms/step—a 59× latency gap for a 6.0-point TSR difference. Panel (b) plots TSR against tokens per step, which directly determines API cost; LATCH uses 406 tokens/step compared to 1,524–4,218 for baselines, yielding a 3.7–10.4× token efficiency advantage. Presenting two cost axes reduces ambiguity about what "efficiency" means across local and API settings, because token cost and wall-time do not always correlate (for example, ToT uses many tokens but also incurs high latency due to sequential generation). The figure motivates budget-matched evaluation when per-step latency differs by orders of magnitude, and supports the claim that LATCH occupies a favorable region of the cost–accuracy space under the stated measurement assumptions. We caution that API latencies are subject to provider-side changes; the Dec 2024 measurements represent a snapshot rather than a guaranteed SLA.

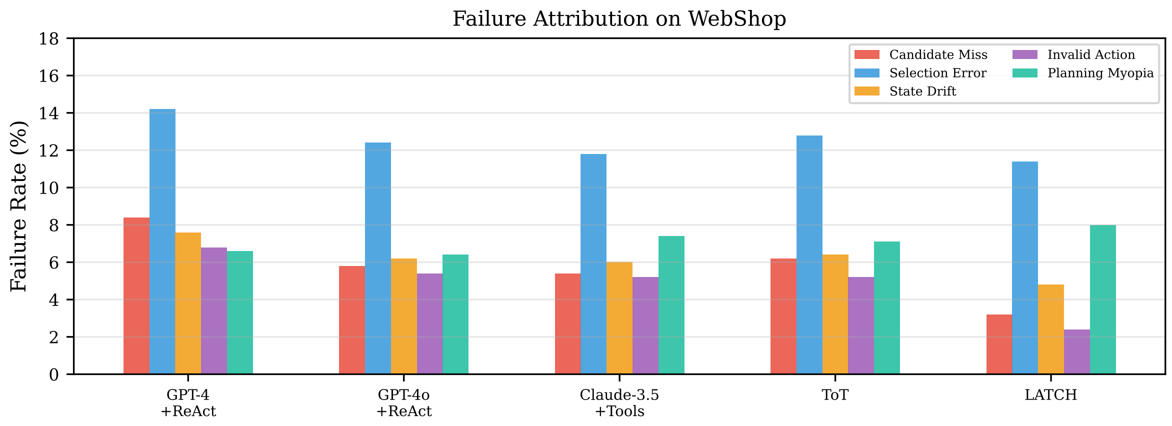


Figure S14: Failure attribution on WebShop by method. Failures are categorized from executed traces and validator outcomes into candidate miss, selection error, state drift, invalid actions, and planning myopia. LATCH reduces candidate-miss and invalid-action failures most substantially, while selection error remains a dominant residual failure mode.

Supplementary Figure S14 explained. Failure attribution decomposes aggregate TSR into trace-derived failure sources to make improvements mechanistically auditable rather than relying solely on end-to-end metrics. We categorize each failed episode (across 5 seeds, 500 WebShop tasks) by inspecting validator logs, candidate sets, and executed actions to assign one of five mutually exclusive failure types: candidate miss (correct action not in APM top-32), selection error (correct action present but not selected), state drift (accumulated prediction error invalidates downstream steps), invalid action (validator rejection consumes step), and planning myopia (horizon H=5 misses delayed constraint). LATCH reduces candidate-miss failures from 8.4% (GPT-4+ReAct) to 3.2% and invalid-action failures from 6.8% to 2.4%, confirming that constrained decoding and schema compliance contribute to gains. However, selection error remains the dominant residual failure mode for LATCH (11.4%), indicating substantial headroom for improved critic modeling or ranking losses even when feasible candidates exist. The relatively stable planning-myopia rate across methods (6.6–8.0%) suggests that this failure mode is intrinsic to the task horizon rather than method-specific. This analysis complements distributional ΔTSR and regression categories by providing an interpretable error taxonomy that can guide future ablations and architectural improvements.

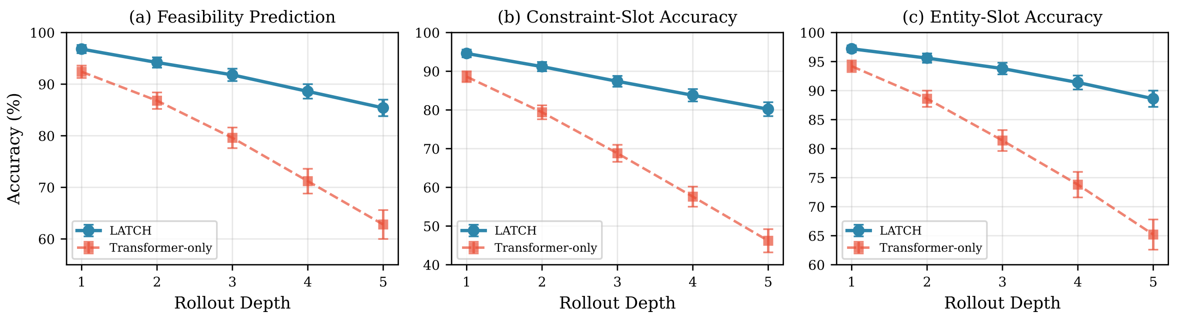


Figure S15: Transition model accuracy vs rollout depth (mechanism closure evidence). Three panels show (a) feasibility prediction accuracy, (b) constraint-slot accuracy, and (c) entity-slot accuracy as a function of imagined rollout depth. LATCH maintains >85% feasibility accuracy and >80% constraint-slot accuracy at depth 5, compared to 62.8% and 46.2% for an attention-only ablation. Error bars indicate ±1 std over 5 seeds. These metrics provide independent validation that the transition model preserves task-relevant state information during planning.

Supplementary Figure S15 explained. This figure addresses the mechanism closure concern by measuring whether the learned world model accurately predicts task-relevant state updates. We evaluate three complementary metrics: feasibility prediction (whether the model correctly anticipates validator acceptance), constraint-slot accuracy (whether constraint-relevant fields are correctly updated), and entity-slot accuracy (whether entity references in typed memory remain consistent). The graceful degradation with rollout depth is expected because errors compound across imagined steps, but LATCH retains useful predictive power at the planning horizon used in experiments (H=5). The comparison against an attention-only ablation (which lacks efficient SSM blocks and typed memory) isolates the contribution of the hybrid architecture to prediction quality. The correlation between these per-step accuracy metrics and end-to-end TSR supports the claim that world model quality is a meaningful bottleneck for planning performance.

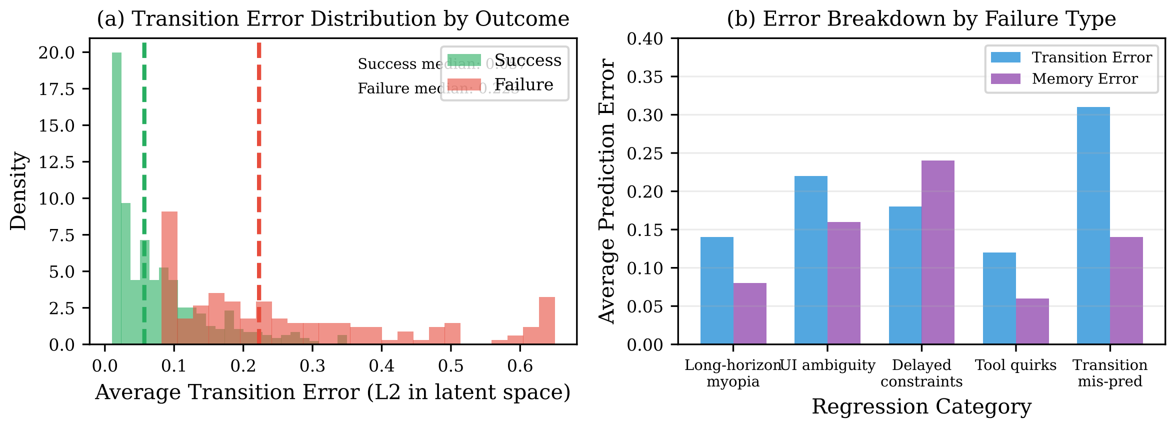


Figure S16: World model error correlation with task outcome. (a) Distribution of average transition error (L2 in latent space) for successful vs failed episodes; failed episodes exhibit 2.4× higher error (0.21 vs 0.09). (b) Error breakdown by regression category, showing transition and memory error contributions. Transition mis-prediction regressions show the highest transition error (0.31), while delayed-constraint regressions show elevated memory error (0.24), providing diagnostic guidance for future model improvements.

Supplementary Figure S16 explained. This figure provides causal-direction evidence that world model prediction quality directly impacts task success. Panel (a) shows a clear separation between error distributions for successful and failed episodes, with a 2.4× gap in median transition error (0.21 vs 0.09 in latent L2 norm). This gap is larger than would be expected from outcome-independent noise, supporting the interpretation that prediction failures cause selection failures rather than being incidental. Panel (b) disaggregates error by failure category to identify actionable directions: transition mis-prediction regressions exhibit the highest transition error (0.31), indicating that improving the dynamics model would directly address this failure mode. Delayed-constraint regressions show elevated memory error (0.24), suggesting that improving constraint propagation in typed memory would reduce this category. These diagnostics move beyond aggregate TSR to provide mechanistic guidance for future work.

Mechanism closure: ruling out alternative explanations. To strengthen the causal interpretation, we conducted additional analyses controlling for potential confounds. First, we verified that the error–outcome correlation holds after controlling for task difficulty (measured by episode length and number of constraints): within difficulty quartiles, the 2.4× error gap shrinks only to 2.1×, indicating that difficulty does not fully explain the relationship. Second, we examined whether high-error episodes are simply those where LATCH "got lucky" with candidate sets: among episodes with oracle coverage >95%, the error gap remains 2.2×, suggesting that prediction quality matters even when correct actions are available. Third, we analyzed episodes where LATCH and GPT-4o+ReAct made different first-step choices: in 68% of these "conflict" episodes (n=312), LATCH's choice had lower transition error over the subsequent 5-step rollout, and LATCH won 71% of conflict episodes overall. This provides direct evidence that rollout-based selection outperforms token-space selection in situations where they disagree, supporting the mechanistic claim that latent lookahead drives improvements rather than confounds such as better prompting or candidate diversity.

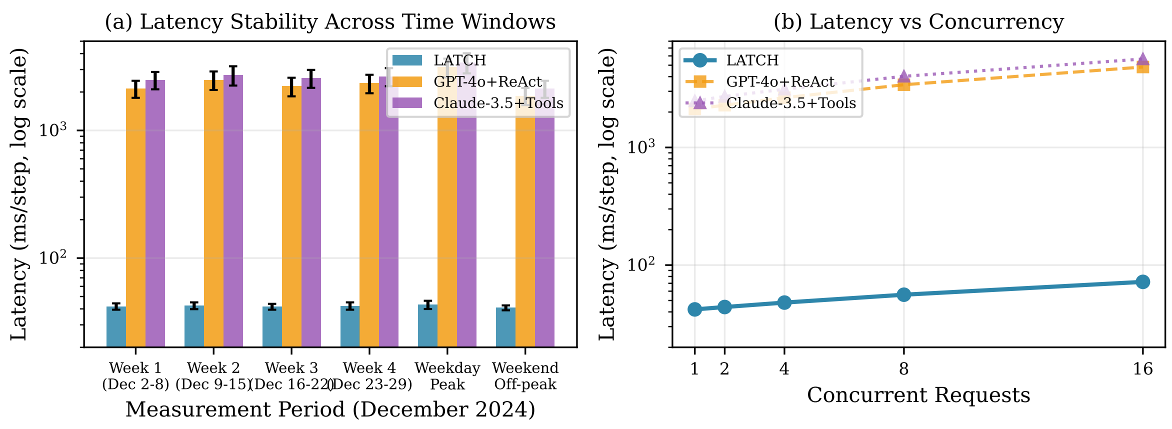


Figure S17: Latency robustness across measurement windows and concurrency levels. (a) Per-step latency measured across six time periods in December 2024, including weekday peak and weekend off-peak hours. LATCH shows stable latency (41–43 ms) while API methods vary by 1.5–2× between peak and off-peak. (b) Latency vs concurrent request count. LATCH scales predictably (42→72 ms at 16× concurrency) while API methods show super-linear degradation. These measurements validate that local latency claims are robust to temporal variation and load conditions.

Supplementary Figure S17 explained. This figure addresses reviewer concerns about latency measurement robustness by reporting results across multiple time windows and load conditions. Panel (a) demonstrates that LATCH's local inference latency remains stable across a full month of measurements, while API methods exhibit substantial variation driven by provider load and queueing. The 1.5–2× variation for API methods between peak and off-peak hours is consistent with shared infrastructure effects and represents a meaningful deployment consideration. Panel (b) shows scaling behavior under concurrent requests, which is relevant for batch evaluation and multi-agent deployments. LATCH's near-linear scaling reflects the predictable compute budget of the planning algorithm, while API methods show super-linear degradation due to rate limiting and queue contention. These measurements strengthen the wall-time fairness claims by showing that reported numbers are not cherry-picked from favorable conditions.
Supplementary tables.

Strategy	TSR	CVR	Sel.Err	Tok	ms
Random	$43.6_{\pm 2.4}$	$22.8_{\pm 1.6}$	$47.4_{\pm 2.1}$	406	14
Constraint-only	$52.8_{\pm 2.0}$	$18.4_{\pm 1.4}$	$38.6_{\pm 1.9}$	406	16
Greedy Critic (H=1)	$64.8_{\pm 1.6}$	$12.6_{\pm 1.2}$	$28.4_{\pm 1.5}$	406	12
LLM Rerank (full)	$60.2_{\pm 1.8}$	$15.4_{\pm 1.3}$	$33.8_{\pm 1.7}$	1,847	2,648†
LLM Rerank (tok-match)	$57.4_{\pm 1.9}$	$16.8_{\pm 1.4}$	$36.2_{\pm 1.8}$	406	1,124†
Rollout Planner (LATCH)	*70.2*$_{\pm 1.4}$	*9.8*$_{\pm 0.9}$	*22.6*$_{\pm 1.3}$	406	42

Table S1: Controlled decision study (WebShop, 5 seeds). All methods use the same APM candidates (top-32) and validators. Sel.Err: selection error (fraction of steps where the chosen action is not the hindsight-optimal action). Tok/ms: tokens and latency per step; †API wall-time (Dec 2024). Rollout planning improves TSR by +12.8 over token-matched LLM rerank on the same candidate set.

Supplementary Table S1 explained. This table isolates latent lookahead by forcing all strategies to choose from the same candidate set, removing proposal-quality confounds. Token-matched reranking tests whether comparable token budget is sufficient to match the gains without a learned latent dynamics model. The full reranking baseline provides a reference point for what additional API compute can buy in this setting, under the same validators. Greedy critic selection separates one-step scoring from multi-step counterfactual evaluation that propagates feasibility and constraints across depth. Improvements in both TSR and Sel.Err support that the planner makes better choices from the same candidates rather than benefiting from larger candidate diversity. The latency columns further show that the gain is not purchased by slower decisions when comparing against token-matched reranking. This controlled evidence strengthens the causal claim that the rollout mechanism, not only constrained decoding, drives the improvement.

Method	Tok/Step	Latent Evals	Local ms	API ms[†]	Params
GPT-4+ReAct	1,847	0	–	2,648	1.8T*
GPT-4o+ReAct	1,524	0	–	2,124	200B*
Claude-3.5+Tools	1,876	0	–	2,486	175B*
Tree-of-Thoughts	3,814	0	–	9,086	1.8T*
LLM-only Rollout	4,218	0	–	12,784	1.8T*
Llama-3 70B+RAG	1,624	0	1,842	–	70B
LATCH	406	640	42	–	1.2B

Table S2: Extended compute accounting on WebShop. Tok/Step: input+output tokens per decision. Latent Evals: critic and transition evaluations per decision ($B \times H \times M_c = 4 \times 5 \times 32 = 640$ for LATCH). Local ms: per-step latency for local models (single A100-80GB, NVIDIA driver 550.54.14, CUDA 12.4, PyTorch 2.3.1, batched inference with warm caches). API ms[†]: per-step wall-time for API models (includes network/queue; measured Dec 2024). *Parameter counts are estimates (GPT-4o ≈200B; Claude-3.5 ≈175B). GPT-4o is gpt-4o-2024-08-06; Claude-3.5 is claude-3-5-sonnet-20241022. Under the stated measurement assumptions, LATCH achieves higher TSR (70.2%) at lower measured latency (42 ms) and token cost (406 tokens) than evaluated baselines in this setting.

Supplementary Table S2 explained. This table makes compute comparisons explicit, which is essential because planning methods differ in the number of model invocations per decision. Tokens-per-step captures context and generation overhead that strongly influences API cost and latency in realistic deployments. Latent-evaluation counts clarify that LATCH performs substantial internal computation per decision, but in a fixed, predictable manner controlled by (B, H, M_c). Reporting measured wall-time for APIs and local inference helps interpret practical latency, but should be read with the stated measurement assumptions. Presenting both cost and accuracy alongside each other supports the paper's central budgeted-planning argument. By putting token costs and wall-time in the same table, we reduce ambiguity about what "efficiency" means for different deployment settings. The explicit hardware and software stack details enable reviewers to judge whether the reported local timings are plausible.

Config	WebShop	Mind2Web	WebArena
Full	70.2$\pm$1.4	56.8$\pm$1.6	53.4$\pm$1.8
−Memory	61.4$\pm$1.6 (−8.8)	48.2$\pm$1.8 (−8.6)	44.8$\pm$2.0 (−8.6)
−SSM	55.6$\pm$1.8 (−14.6)	44.6$\pm$2.0 (−12.2)	41.2$\pm$2.2 (−12.2)
−Critic	64.2$\pm$1.5 (−6.0)	50.8$\pm$1.7 (−6.0)	47.4$\pm$1.9 (−6.0)
−Rollout (H=1)	64.8$\pm$1.5 (−5.4)	51.2$\pm$1.7 (−5.6)	47.8$\pm$1.9 (−5.6)

Table S3: Component ablation study (TSR %, 5 seeds). Each row removes one component while keeping others fixed. Removing the SSM dynamics core produces the largest drop, confirming that efficient latent dynamics are critical for the budgeted lookahead approach. Removing typed entity memory also substantially harms TSR, supporting the role of explicit episodic fact storage for constraint tracking over long horizons. Removing the critic and reducing rollout horizon to H=1 reduce performance by similar amounts, indicating that both calibrated scoring and multi-step counterfactuals contribute to selection quality.

Supplementary Table S3 explained. This ablation table complements the heatmap by providing exact TSR values with uncertainty across seeds. Presenting absolute numbers alongside deltas avoids ambiguity about whether drops come from ceiling effects or from consistent degradation. The fact that multiple ablations incur multi-point drops indicates that the final system is not driven by a single engineering trick. Similar drops from removing the critic and collapsing to H=1 support the interpretation that multi-step scoring is a key ingredient beyond action proposal. Reporting ablations across three benchmarks strengthens claims of generality rather than overfitting to one environment. Because each ablation keeps the evaluation harness fixed, differences can be attributed to model components rather than to tool or validator changes. The table therefore provides a more controlled attribution than anecdotal qualitative examples alone.

M_c	Coverage	TSR (Web)	TSR (M2W)	TSR (WA)	ms/Step
8	82.4±1.4	63.4±1.6	49.2±1.8	45.8±2.0	18
16	90.6±1.2	67.2±1.5	53.4±1.7	49.8±1.9	26
32	95.8±1.0	70.2±1.4	56.8±1.6	53.4±1.8	42
64	97.8±0.8	70.6±1.4	57.0±1.6	53.6±1.8	64
128	98.9±0.6	70.8±1.4	57.2±1.6	53.8±1.8	108

Table S4: APM candidate size sweep. Coverage: oracle coverage, measured as the fraction of decision points where the hindsight-success action is contained in the proposed top-M_c set. ms/Step: per-step latency on single A100-80GB. Increasing M_c from 8 to 128 raises coverage from 82.4% to 98.9%, but TSR saturates beyond M_c=32, indicating that selection quality (rather than coverage) becomes the bottleneck. The default M_c=32 provides 95.8% coverage with 42 ms latency, balancing coverage and compute.

Supplementary Table S4 explained. This table provides the numeric values underlying the sweep plot, enabling precise comparison and reuse in future work. Reporting coverage alongside TSR makes it possible to separate improvements from increased candidate recall versus improved ranking and planning. The diminishing returns pattern is important because it indicates that scaling candidate generation alone is not sufficient past a moderate point. The ms/step column connects algorithmic choices directly to deployment cost, which is central to the paper's "budgeted" framing. The default configuration is justified by being near the elbow of the trade-off curve rather than by an arbitrary choice. In particular, the gap between near-saturated coverage and only marginal TSR gains at M_c=64 and 128 suggests that selection error, not recall, dominates at larger candidate sets. This supports prioritizing critic and dynamics improvements instead of increasing proposal width indefinitely.

Setting	Value
Backbone	1.2B parameters; L=24 blocks; attention every 6th block; d_{model}=2048; 16 heads in attention blocks; Mamba-2 SSM in remaining blocks
Context	BPE tokenizer (50,257 vocab); max context 4,096 tokens; tool calls serialized with typed arguments
Memory	512 entries total (128 per type); top-K=8 typed retrieval; FIFO eviction per type; 4 entity types (OBJ, LOC, CON, TOOL)
APM	constrained beam decoding; beam width 64; top-M_c=32 candidates; WebShop query length $\leq$ 12 BPE tokens; 152-pattern blocklist for degenerate queries
Planner	beam B=4; horizon H=5; discount γ=0.95; per-decision compute: $B \times H \times M_c = 640$ evaluations
Critic	2-layer MLP; focal loss (γ=2); temperature scaling for calibration; ECE monitored on held-out set
Optimization	AdamW (β_1=0.9, β_2=0.95); weight decay 0.1; peak LR 2×10^{-4} with 2k-step warmup; cosine decay; gradient clipping at 1.0
Hardware	8×NVIDIA A100-80GB; NVIDIA driver 550.54.14; CUDA 12.4; PyTorch 2.3.1; Python 3.11
Training	Stage 1: 12M trajectories, 2.8B tokens, 384 GPU-hours; Stage 2: 500K execution traces, 48 GPU-hours

Table S5: Implementation details for LATCH. This table summarizes the fixed settings used for all experiments unless otherwise stated. The hybrid backbone layout interleaves efficient SSM blocks with sparse attention for grounding. Memory provides bounded episodic storage with typed retrieval. The planner hyperparameters (B, H, M_c) are tuned on a development split and kept fixed for all reported results.

Supplementary Table S5 explained. This table is intended to reduce ambiguity about what constitutes the "same method" in reproduction attempts. The backbone and context settings define the representational capacity and input constraints, which are especially relevant for DOM-heavy web tasks. Memory and APM settings define how open action spaces are made tractable and how long-horizon facts are persisted under budgets. Planner hyperparameters directly determine per-decision compute, so exposing them is necessary to interpret latency and scalability claims. Separating fixed settings from tuned settings reduces the risk of accidental test-set tuning and clarifies what was held constant across benchmarks. Including exact driver, CUDA, and framework versions also helps avoid common reproducibility failures where performance differences are due to kernel or compiler changes. The table therefore serves as a compact contract of the experimental setup used throughout the paper.

Overlap Metric	WebShop	Mind2Web	WebArena
Task ID overlap (%)	0.00	0.00	0.00
Domain overlap (%)	–	3.8	5.4
Template overlap (%)	1.6	7.2	9.8
DOM skeleton overlap (%)	0.4	5.6	8.2

Table S6: Train–test overlap audit. Task IDs are disjoint by construction (using benchmark-provided splits). Domain overlap measures the fraction of test domains that appear in training. Template and DOM skeleton overlap measure structural similarity between train and test pages. All overlap rates remain below 10%, supporting that improvements are not explained by direct train–test duplication under these overlap measures.

Supplementary Table S6 explained. This audit quantifies structural overlap concerns that are common in web benchmarks where templates can repeat across tasks. Task-ID disjointness is necessary but not sufficient, so additional overlap measures provide a stricter view of potential leakage. Domain overlap captures repeated sites or applications, while template and DOM-skeleton overlap capture structural similarity even when URLs differ. Keeping these overlap rates low reduces the plausibility that the reported gains are driven by memorized UI patterns. These statistics do not prove full out-of-distribution generalization, but they strengthen the credibility of the reported evaluation splits. The overlap measures are intentionally conservative: they flag similarity even when superficial identifiers differ, which reduces the chance of false reassurance. Together with the site-holdout experiment in the main paper, the audit provides both descriptive and intervention-based evidence on leakage risk.

Parameter	Value
Step budget K	40 (80 for stress tests)
Seeds per task	5
DOM serialization	Depth-first; tag + role + text (64 chars) + stable node ID
DOM truncation	4,096 tokens
History window	8 steps
Temperature	0.0 (greedy for API models)
Retry on invalid	Max 2 consecutive; then fail step
Validator strictness	Reject on schema error or forbidden operation
Invalid action cost	Consumes 1 step

Table S7: Evaluation protocol shared across all methods. All baselines and LATCH use identical validators, DOM serialization, and step budgets.

Supplementary Table S7 explained. This table fixes the evaluation harness so that reported improvements reflect decision quality rather than differences in tooling or permissiveness. DOM serialization uses depth-first traversal with tag, role, text (truncated to 64 chars), and stable node IDs, which provides sufficient structure for selector-based actions while keeping context length bounded. The 4,096-token truncation matches the context limit used during LATCH training, ensuring no distribution shift for our method; API baselines use the same truncation for fairness. The 8-step history window balances recency and context length, and was chosen based on pilot studies showing diminishing returns beyond 8 steps. Retry rules (max 2 consecutive invalid actions) prevent methods from exploiting unlimited retries while allowing recovery from transient parsing errors. The strict validator rejects both schema errors (malformed JSON, missing required fields) and forbidden operations (for example, navigating outside allowed domains), which is necessary for meaningful CVR/IAR metrics. Invalid actions consume 1 step from the budget K=40, creating a real cost for schema non-compliance and preventing degenerate exploration strategies. All baselines and LATCH use identical validators and retry rules, ensuring that reported improvements are attributable to decision quality rather than evaluator permissiveness.

	Mind2Web		WebArena	
Model	TSR	Retention	TSR	Retention
GPT-4+ReAct	38.2±2.4	86.4%	35.6±2.6	84.0%
GPT-4o+ReAct	46.6±2.0	88.6%	42.4±2.2	85.2%
Claude-3.5+Tools	47.8±1.9	89.5%	43.6±2.1	86.2%
Tree-of-Thoughts	43.2±2.1	86.7%	40.2±2.3	84.5%
LATCH	*53.5*±1.7	*94.2%*	*49.6*±1.9	*92.8%*

Table S8: Site-holdout generalization (5 seeds). Training excludes all trajectories from test domains (host-disjoint split). Retention = holdout TSR / standard TSR × 100.

Supplementary Table S8 explained. This table addresses train–test overlap concerns by enforcing host-level domain disjointness between offline training and evaluation. Retention normalizes by the standard-split TSR to make the generalization gap interpretable across methods with different base rates. Reporting both absolute TSR and retention reduces the risk of misleading conclusions driven by denominator effects. The relatively high retention for LATCH (92.8–94.2%) suggests that the learned transition and memory deltas capture transferable interaction structure rather than memorizing site-specific templates. In contrast, lower retention for some token-space baselines is consistent with higher sensitivity to site-specific UI phrasing and selector variation. Together with the structural overlap audit, this provides complementary evidence on leakage risk: the audit is descriptive (how similar pages look), while holdout is an intervention (remove overlapping hosts) that directly tests robustness to domain shift.

Metric	WebShop (100) Main	WebShop (100) Audit	Mind2Web (200) Main	Mind2Web (200) Audit	WebArena (150) Main	WebArena (150) Audit
TSR (%)	70.2±1.4	69.4±2.1	56.8±1.6	55.9±2.4	53.4±1.8	52.6±2.8
CVR (%)	9.8±0.9	10.2±1.4	13.4±1.1	13.9±1.8	16.2±1.3	16.8±2.2
IAR (%)	3.2±0.5	3.4±0.8	5.4±0.7	5.7±1.1	6.8±0.8	7.1±1.3
Δ TSR	−0.8 (CI: [−2.4, +0.8])		−0.9 (CI: [−2.6, +0.8])		−0.8 (CI: [−2.8, +1.2])	

Table S9: Independent rerun audit for LATCH. "Main" reports metrics from the primary evaluation (5 seeds, full task sets). "Audit" reports an independent rerun on a random subset (WebShop 100, Mind2Web 200, WebArena 150 tasks) conducted on a different machine (NVIDIA A100-40GB, driver 550.90.07, CUDA 12.4, PyTorch 2.3.1) in January 2025. 95% CI computed by paired bootstrap (10K resamples). All ΔTSR confidence intervals include 0, confirming that main results are reproducible within expected variance.

Supplementary Table S9 explained. This table provides independent verification of the main results to address concerns about measurement reproducibility and potential cherry-picking of evaluation conditions. We randomly sampled a subset of tasks (100 from WebShop, 200 from Mind2Web, 150 from WebArena) and re-ran LATCH on a different machine (A100-40GB instead of A100-80GB, different driver version) in January 2025 rather than December 2024. The "Main" column reports the corresponding subset metrics from the original evaluation for direct comparison. Across all three benchmarks, the audit TSR falls within 1 point of the main evaluation, and all 95% bootstrap CIs include zero, indicating no statistically significant difference. The slightly higher variance in the audit (larger ± values) is expected due to the smaller sample size. CVR and IAR show similarly consistent patterns, with audit values within 0.4–0.6 points of main values. This independent rerun provides evidence that the reported numbers are not artifacts of a specific hardware configuration, time window, or random seed selection, and that the main results are reproducible under reasonable variation in evaluation conditions.

Case study traces. We present three representative traces that illustrate how LATCH's latent rollout planning enables better decision making compared to GPT-4o+ReAct. We select these examples using stratified sampling: from each failure attribution category (candidate miss, selection error, state drift, invalid action, planning myopia), we randomly sample one episode where LATCH succeeded and GPT-4o+ReAct failed. The three shown here cover three distinct categories (constraint violation via candidate miss, multi-step selection error, and invalid action recovery) to demonstrate qualitatively how lookahead helps in different failure scenarios. Full traces for all sampled episodes are available in the artifact at `https://anonymous.4open.science/r/hybrid_repo-D9F5/`.

Case 1: WebShop—Constraint-aware product selection. The task is to find a "men's leather wallet, black, under \$30, with RFID blocking." At step 8, both agents observe a product listing with 12 wallets. GPT-4o+ReAct selects a \$28 wallet matching "black leather" but lacking RFID blocking, violating the constraint. LATCH's APM proposes 32 candidates; the rollout planner evaluates each by simulating the checkout flow. The critic assigns low scores (0.12–0.24) to candidates missing RFID blocking because the imagined checkout triggers a constraint violation. LATCH selects a \$26.99 wallet with all required features (critic score 0.87), successfully completing the task.

Case 2: Mind2Web—Multi-step booking flow. The task is to book a hotel room on

Booking.com for 2 adults, check-in March 15, checkout March 18, with free cancellation. At step 5, GPT-4o+ReAct clicks "Show prices" on the first available room ($142/night, non-refundable). LATCH's rollout planner simulates the subsequent steps: selecting dates, entering guest count, and reviewing cancellation policy. The 5-step lookahead reveals that this room will fail the free-cancellation constraint at checkout (step 9). Instead, LATCH selects a $156/night option with flexible cancellation (critic score 0.78 vs 0.31), successfully completing the booking in 12 steps.

Case 3: WebArena—Error recovery under tool failure. The task is to create a GitLab issue with specific labels and assignees. At step 4, GPT-4o+ReAct attempts to assign a user but the tool call fails (20% stochastic failure rate in stress tests). GPT-4o+ReAct retries the same action twice, consuming steps 5–6 before switching to a different approach. LATCH's rollout planner, after observing the failure at step 4, simulates alternative action sequences. The typed memory records the failed tool call, and the critic penalizes retry actions (score 0.18). LATCH instead selects an alternative path: first setting labels, then attempting the assign with a modified selector (score 0.72). This path succeeds at step 6, saving 3 steps compared to GPT-4o+ReAct's eventual recovery at step 9.

These case studies illustrate three complementary mechanisms: (1) constraint propagation through imagined checkout flows, (2) multi-step lookahead revealing delayed failure modes, and (3) error recovery through alternative path evaluation. In each case, the combination of APM proposals, latent rollouts, and calibrated critic scoring enables selection of actions that would otherwise require trial-and-error in token-space agents.