REAL-TIME HEURISTIC LEARNING: COMPILING ONLINE EVIDENCE INTO EXECUTABLE POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-time agents must act promptly, but handling changing states does not establish adaptation to changing rules. We introduce Evolving Freeway and Evolving Snake, extending Real-Time Reasoning Gym with repeated hidden rule changes within continuing episodes. Phase boundaries are public and active rules hidden. Transitions provide evidence, and world state persists. Reactive, Planning, and AgileThinker retain discoveries as prompt evidence, action sequences, or textual guidance rather than installed policy revisions. We propose *Real-Time Heuristic Learning* (RTHL), an evidence-to-code framework compiling transition evidence into reusable executable rules. A deterministic heuristic selects actions. Post-transition diagnosis drives bounded code revisions, screened by program checks and replay, with phase scoping and rollback governing reuse. Across 576 episodes, RTHL leads all six pooled comparisons of mean terminal scores across environments and budgets, gaining 8.5–12.7% over Reactive on Freeway and 43.4–49.4% on Snake. At 1K, updating the same fixed initial controller adds 5.00 Freeway points and 10.25 Snake points. RTHL leads 12 of 14 Hard comparisons across seven budgets (1K–32K), including seven on Snake. RTHL enables reusable adaptation without weight updates.

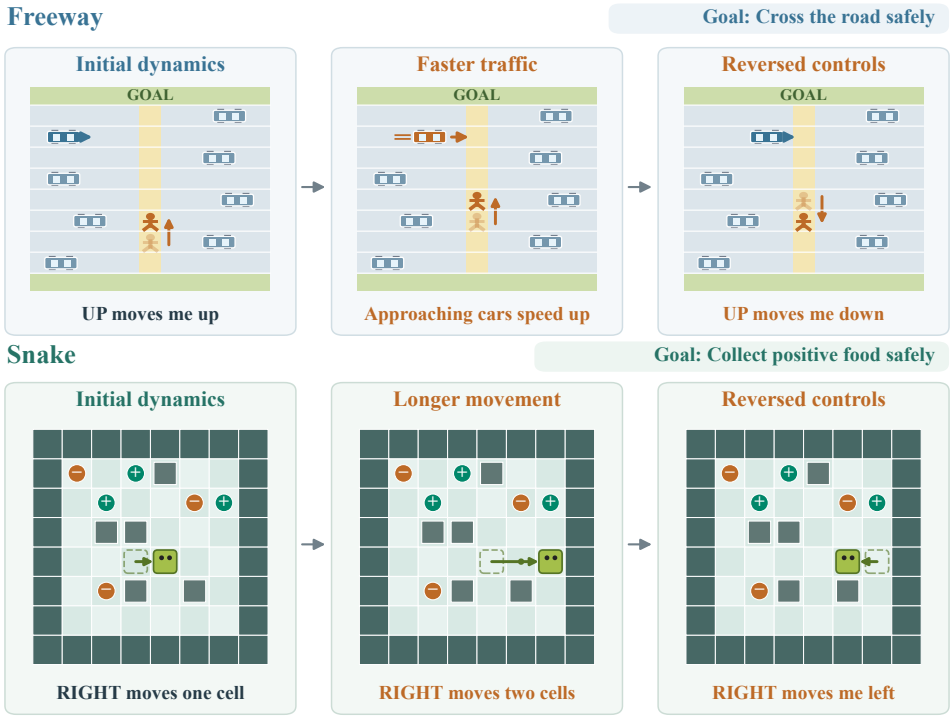


Figure 1: **Changing rules within a continuing episode.** Freeway (top) and Snake (bottom), read left to right, illustrate changes in traffic and movement rules. Active rules are hidden from the agent, and world state persists across phase boundaries. These are task schematics, not recorded trajectories from the experiments reported below for either game.

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.

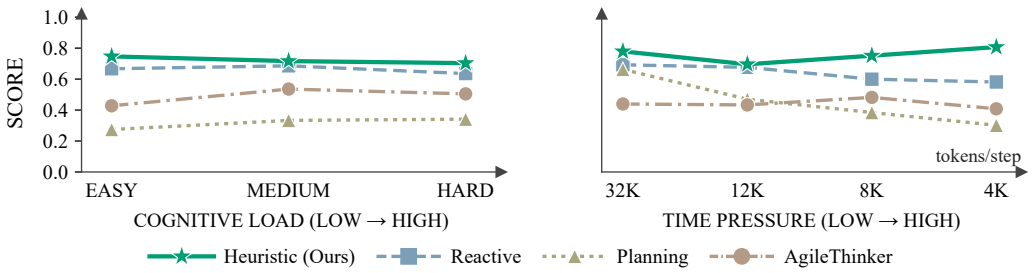


Figure 2: **Normalized performance across difficulty and token settings.** Left: Easy, Medium, and Hard strata. Right: selected Hard-stratum budgets from 32K to 4K. Heuristic (Ours) denotes RTHL. Higher is better. Budget labels are architecture-native, not matched total compute.

1 INTRODUCTION

Language agents interleave reasoning and actions, using interaction feedback to revise plans and guide subsequent decisions (Yao et al., 2022).Evaluating this feedback-driven behavior requires interactive benchmarks that test how agents use observations to sustain progress across successive decisions. AgentBench and WebArena evaluate multi-turn decision making and long-horizon task completion (Liu et al., 2024; Zhou et al., 2024b). HAZARD and Real-Time Reasoning Gym extend this agenda to dynamic hazards and time-sensitive interaction (Zhou et al., 2024a; Wen et al., 2026).As hazards move and opportunities expire, useful reasoning must lead to timely actions, not just correct plans, while relevant opportunities for successful action remain available.

These benchmarks evaluate interaction and adaptation to changing states, but their standard protocols do not directly test repeated, hidden changes to the rules governing a continuing episode. This leaves a distinction between *changing states* and *changing rules* untested: reacting to moving traffic does not establish whether an agent can discover that its movement commands now have opposite effects. Evaluating this ability requires agents to infer new rules from outcomes and reuse them before the next change, without resetting the world at rule boundaries. A discovery must remain useful as positions and hazards change, yet cease to guide behavior when its supporting mechanism no longer applies, as new evidence reveals further changes in the environment.

We introduce a gym comprising *Evolving Freeway* and *Evolving Snake*, extending Real-Time Reasoning Gym with this requirement. Figure 1 illustrates the tasks. Freeway requires crossing eight traffic lanes as vehicles accelerate or UP begins moving the player down. Snake requires collecting positive food before expiry as RIGHT moves two cells or moves left. Other rules change food values and hazards. Each episode contains three hidden-rule phases: boundaries are public, active rules are undisclosed, and world state persists without a boundary reset.Adaptation must therefore use public transition evidence, linking actions to outcomes under the currently active rules.

On this benchmark, reasoning schedules alone do not make discoveries reusable. Reactive reinterprets evidence within bounded action calls. Planning releases sequences tied to earlier snapshots, using defaults until actions become available. AgileThinker combines current-state reactions with slower textual guidance. These methods primarily adapt through textual reasoning and plan updates, leaving the reliable reuse and timely revision of inferred dynamics a challenge when rules change repeatedly. This motivates a different target for inference: *revise the policy that produces later actions*, enabling discovered rules to guide decisions across changing states.

We propose *Real-Time Heuristic Learning* (RTHL), an evidence-to-code framework. A deterministic heuristic supplies each action without a fresh action-generating language-model call. After transitions, a Critic diagnoses prediction discrepancies and a coding agent proposes bounded revisions. Program checks and replay validate candidates before installation. Phase scoping controls reuse, and contradiction monitoring can restore prior code. Thus, evidence changes subsequent action selection without updating model weights. An installed revision is evaluated on the current state at each subsequent turn, rather than replaying an action sequence tied to an earlier observation. This separates the lifetime of a discovery from the reasoning call that produced it, while keeping its use conditional on evidence and phase scope as the environment changes further.

Figure 2 summarizes normalized scores across difficulty and selected Hard-stratum token settings, with RTHL leading all displayed method averages. In the 576-episode main evaluation, RTHL leads all six pooled comparisons across environments and budget settings, improving raw scores over Reactive by 8.5–12.7% on Freeway and 43.4–49.4% on Snake. Online revision at 1K adds 5.00 and 10.25 points, respectively, over the same fixed initial controller. Across the full seven-point 1K–32K Hard diagnostic, RTHL leads in 12 of the 14 combinations of game and budget, including all seven on Snake across the entire range of evaluated token settings.

The paper makes three contributions:

- **A gym for real-time adaptation:** two environments with repeated hidden rule changes, persistent state, and public evidence for adaptation within each episode.
- **An evidence-to-code framework:** deterministic control with validated source revision, phase-scoped reuse, and rollback to turn observed transitions into reusable action rules.
- **An empirical test of deployment-time updating:** four-architecture comparisons and matched Fixed controls, with separate measurements of scores, action availability, and token usage across difficulty strata and architecture-native budget settings.

2 Evaluation In Evolving Environments

2.1 Games With Changing Rules

We build *Evolving Freeway* and *Evolving Snake* on Real-Time Reasoning Gym (Wen et al., 2026), adding within-episode rule changes to action semantics, object behavior, and hazards.

Evolving Freeway And Snake. Freeway requires repeated safe crossings of eight traffic lanes. Collisions reset the player but not traffic. Rules accelerate approaching vehicles, reverse commands, or widen collision regions. Snake requires collecting positive food before expiry while avoiding negative food, obstacles, and hazards. Death resets the player without ending the episode. Rules double movement distance, swap food polarity, or make positive-food columns dangerous.

Changing Rules Within One Episode. Each episode contains three 30-action phases with different hidden rules. Boundaries are public, while rule identities and parameters are hidden. World state persists, requiring agents to infer rules from outcomes and reuse them during play. Table 1 summarizes the tasks. Appendix A details rules and schedules.

2.2 Difficulty And Token Budgets

Task Difficulty. Snake's Easy, Medium, and Hard settings contain 1, 5, and 8 obstacles, respectively, increasing navigation constraints. Freeway uses three fixed seed strata with the same labels. These strata sample traffic and rule sequences rather than impose a monotonic traffic-density scale.

Inference Allocation. Let B denote the architecture-native token setting, with K equal to 1,024 tokens. For Reactive, it bounds per-turn reasoning. For Planning, it controls how much generated plan text becomes available each logical turn. AgileThinker combines fast responses with released planning guidance. RTHL caps each Critic and coding call while its installed program supplies actions. RTHL's adaptation follows the observed transition and completes before the next environment step. Thus we evaluate logical action order and report resource usage separately. Equal budget labels do not imply equal total compute across the four evaluated reasoning architectures.

2.3 Evaluation Protocol

We compare all four architectures using DeepSeek-V4-Flash (Xu et al., 2026) for every LM role. They share public observations, outcomes, phase boundaries, and bounded evidence, without access to active rules or seed-to-rule mappings. Candidate catalogs match on Snake, Freeway baselines receive seven implemented rule families, while RTHL receives these plus three distractors. Each RTHL seed starts from the same environment-specific controller, and learned edits remain local to that episode without transferring revisions to subsequent seeds or other games.

Table 1: **Two environments test reuse of discovered dynamics within an episode.** Active mechanisms are hidden, and world state persists across public phase changes.

Task	Episode	Hidden changes	Consolidation challenge	Score
Freeway	100 actions, three hidden phases	Vehicle motion, action semantics, responses, and collision geometry	Reuse corrected dynamics across repeated lane crossings	Terminal score: 10× completed crossings
Snake	90 actions, three hidden phases	Movement, food transformation, and hidden hazards	Reuse learned rules while navigating 1/5/8 obstacles	Cumulative reward from food and lethal resets

Scores And Main Comparison. Freeway runs for 100 actions and scores ten points per completed crossing at episode end. Snake runs for 90 actions: positive food contributes one point, while negative food and death each subtract one. We evaluate 1K, 2K, and 4K across E/M/H, using eight seeds for each method at every combination of environment, difficulty, and budget: 576 complete episodes. We report terminal-score means and sample standard deviations. Pooled values combine 24 episodes across strata. Paired intervals match episodes by seed and stratum, treating each episode as the unit of analysis. Higher scores indicate better control. Raw scores are compared within each game under its own scoring function and episode horizon.

Matched Ablations. Three controls cover E/M/H and eight seeds per environment (144 additional episodes). *Fixed Code* retains the initial heuristic and bounded memory but disables LM calls. *No Replay/Rollback* removes historical replay and post-installation rollback, retaining program checks. *No Phase Scoping* removes learned hooks' phase gates, retaining public phase information, replay, and rollback. Online variants use 1K, while Fixed is budget-independent. These controls test updating and its installation safeguards (Appendix C.4) under the same public evidence interface.

Hard-Stratum Budget Diagnostic. We extend B to 1K, 2K, 4K, 8K, 12K, 24K, and 32K on three designated Hard seeds. Planning holds its completion ceiling at 65,536 tokens while varying logical release. AgileThinker holds the same slow-model ceiling and sets both logical release and its reactive budget to B. Reactive and RTHL use the allocations above. The diagnostic reports score, action availability, and recorded completion usage. Reactive/RTHL's 1/2/4K points reuse the designated primary trajectories. Full provenance appears in Appendix C.2.

Instrumented Resource Sample. At 4K, we measure RTHL's provider-reported input/output tokens across all Critic and coding calls, including terminal work, for six episodes: seed index 2 in each game's E/M/H strata. Baseline matching and accounting details appear in Section 4.3 and Appendix C.3, including per-role usage and complete episode totals.

3 Real-Time Heuristic Learning

RTHL separates *executing the current policy* from *revising that policy*. Completed transitions guide revisions for later actions through the logical loop in Figure 3: act, observe, diagnose, propose, validate, and reuse. Updates remain local to each episode.

3.1 Baseline Agents

We compare three reasoning architectures from Real-Time Reasoning Gym (Wen et al., 2026). **Reactive** generates each action from the latest observation and bounded evidence using a fresh LM completion. **Planning** generates a multi-step sequence from a starting snapshot. The evaluated implementation releases action prefixes under the logical token budget and uses a default action when none is available. **AgileThinker** guides a current-state reactive LM with a logically released planning trace, which the reactive branch can override. The slow trace remains grounded in its planning-start observation. All three infer adaptation from public observations and transition evidence, without active rule identities or hidden parameters. They retain prompt evidence, action queues, or textual guidance. RTHL installs validated revisions in its action controller (Table 2).

Table 2: Action generation and retained experience across the evaluated architectures. RTHL carries validated updates in its executable policy for later action selection.

Agent	Current-action source	State grounding	Cross-turn artifact
Reactive	Bounded LM generation	Latest state	Bounded evidence in the next prompt
Planning	Released plan action or default	Planning-start snapshot	Snapshot-conditioned action queue
AgileThinker	Reactive LM guided by slow trace	Latest state plus older trace	Snapshot-conditioned textual trace
RTHL (Ours)	**Installed deterministic heuristic**	**Latest state plus bounded episode memory**	**Validated episode-local program edits**

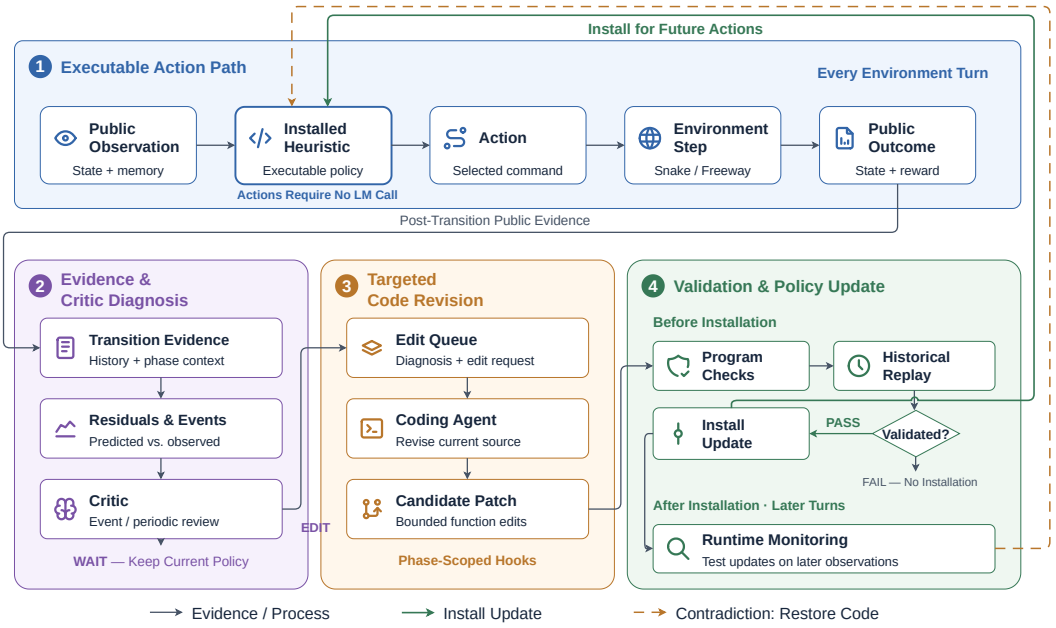


Figure 3: **RTHL places validated experience on the future action path.** The installed heuristic acts on public observations and bounded memory. Post-transition evidence drives Critic diagnosis and validated code revision. Green arrows install updates for later actions, while dashed orange arrows restore prior code after contradictions. Arrows show logical order, not concurrency.

3.2 Executable Action Path

At logical turn t, the installed source ψ_t defines a deterministic policy h_{ψ_t} that maps public observation o_t and bounded outcome memory m_t to the submitted action a_t:

$$a_t = h_{\psi_t}(o_t, m_t). \tag{1}$$

The environment step $\mathcal{E}_t$ in Figure 3 returns the next-turn observation o_{t+1}, reward r_t, and public outcome metadata u_t (e.g., collisions or respawns). Here ψ_t denotes source code, not LM weights. Action selection requires no fresh LM completion.

A Stable Controller With Revisable Dynamics. Each episode starts from a task-specific controller for ordinary mechanics: Freeway uses short-horizon action search, vehicle prediction, and collision checks. Snake searches over positions and time steps, using movement prediction, food forecasts, and safety predicates. Learning revises these assumptions. Fixed Code retains the same initial source and bounded memory, isolating online revision.

One Revision, Many State-Conditioned Uses. Edits target prediction, safety, and action semantics. Correcting Snake's reversed-control predictor, for example, updates successor states in later searches on new boards. Freeway restricts edits to named hooks. Snake allows at most two changed functions, primarily using three modular hooks (Appendix B).

3.3 Evidence-To-Code Adaptation

After each transition, the deterministic, rule-agnostic extractor $\mathcal{R}$ compares outcomes with ordinary-mechanics predictions, producing evidence record d_t and bounded evidence state D_t:

$$d_t = \mathcal{R}(o_t, a_t, o_{t+1}, r_t, u_t), \qquad D_t = \mathrm{Bound}((d_i, p(i))_{i \le t}) \,. \tag{2}$$

Here i indexes completed transitions, and $p(i)$ gives their public phase, not hidden rule identity. Bound retains recent phase-tagged records and aggregates recurring within-phase anomalies. Records contain outcomes and residuals such as unexpected landings, collisions, or food changes, without accessing hidden rules or parameters during extraction.

Anomalies, adverse outcomes, and scheduled five-action reviews trigger Critic opportunities. Given D_t and bounded rolling notes ℓ_t (prior hypotheses and update feedback), the LM-based Critic returns decision q_t for subsequent code revision, rather than current action selection:

$$q_t = \mathrm{Critic}(D_t, \ell_t) \in \{\textbf{WAIT}\} \cup \mathcal{Q}_{\mathrm{edit}}. \tag{3}$$

The union permits either no edit (**WAIT**) or a request from the admissible set $\mathcal{Q}_{\mathrm{edit}}$. Requests specify a falsifiable diagnosis and target behavior, not an action sequence, and enter the pending-request queue Q_t, distinct from decision q_t. At a later service turn $\tau > t$, coding operator G produces $\widetilde{\psi}_\tau = G(\psi_\tau, q_t)$ using the then-installed source ψ_τ. The tilde denotes a proposal, not installed code. The installed controller supplies actions throughout.

3.4 Guarded Policy Consolidation

Validate Before Installation. Let $g \in \{\mathrm{Fwy}, \mathrm{Snk}\}$ denote Freeway or Snake. Superscript g labels environment-specific checks. $\Delta_\tau = \mathrm{FnDiff}(\psi_\tau, \widetilde{\psi}_\tau)$ is the set of changed top-level functions returned by source-comparison operator FnDiff, and D^g_τ is the public-history subset selected for replay at service turn τ. The Boolean validation result v^g_τ is one for acceptance and zero for rejection:

$$\begin{aligned} v^g_\tau = {} & V_{\mathrm{parse}}(\widetilde{\psi}_\tau) \wedge V^g_{\mathrm{surface}}(\psi_\tau, \widetilde{\psi}_\tau, \Delta_\tau) \\ & \wedge V_{\mathrm{load}}(\widetilde{\psi}_\tau) \wedge V^g_{\mathrm{contract}}(\widetilde{\psi}_\tau) \\ & \wedge V^g_{\mathrm{history}}(\psi_\tau, \widetilde{\psi}_\tau, D^g_\tau). \end{aligned} \tag{4}$$

The Boolean tests check syntax (V_{parse}), permitted edits (V^g_{surface}), loading (V_{load}), interface and determinism contracts (V^g_{contract}), and hook-specific evidence support and non-regression (V^g_{history}). $\wedge$ requires all to pass. History tests compare candidate and installed source on D^g_τ, without querying hidden outcomes for unobserved actions (Appendix B). The next-turn source $\psi_{\tau+1}$ is

$$\psi_{\tau+1} = \begin{cases} \widetilde{\psi}_\tau, & v^g_\tau = 1 \text{ and installation succeeds,} \\ \psi_\tau, & \text{otherwise.} \end{cases} \tag{5}$$

Only successful installation changes subsequent controller calls.

Scope Reuse And Monitor Contradictions. Phase metadata gates covered hooks to eligible public phases, defining *when* a rule applies without revealing *which* hidden mechanism is active. Subsequent transitions test installed behavior, and covered runtime guards restore prior source after contradictions. These checks keep revisions inspectable and responsive to new evidence.

4 Results

4.1 Executable Adaptation Improves Evolving Control

Across the full primary evaluation, RTHL achieves the highest pooled mean terminal score in all six comparisons across environments and budget settings (Figure 4, upper row), gaining 8.5–12.7% on

Freeway and 43.4–49.4% on Snake over the strongest pooled baseline, Reactive. It exceeds Planning and AgileThinker in all 18 stratum-specific comparisons and leads all nine Snake cells. Pooling the three settings within each Snake stratum (24 episodes per method, Figure 4, lower row), its gains over Reactive increase from 4.17 points with one obstacle to 7.63 with five and 8.63 with eight. The largest margin occurs under the strongest navigation constraints, where corrected movement or hazard rules can guide later decisions. Full stratum-specific results appear in Appendix C.

4.2 Online Updates Add Value Beyond The Initial Controller

Table 3 isolates online revision: Fixed Code retains the same initial hand-written policy and bounded outcome memory but disables the revision stack. Enabling it at 1K adds 10.25 points on Snake and 5.00 on Freeway, with paired 95% confidence intervals of $[6.796, 13.704]$ and $[1.478, 8.522]$, respectively. Full has a higher mean in all six matched comparisons across games and difficulty strata, supporting deployment-time updating beyond the initial controller. Appendix C.4 gives all stratum-level values.

Table 3: Mean terminal scores over 24 matched episodes per game. Full RTHL uses 1K. Fixed makes no LM calls at inference time.

Method	Snake	Freeway
Fixed	11.083	27.083
Full RTHL	**21.333**	**32.083**

Component controls test selective consolidation: replay and rollback govern installed revisions, while phase scoping bounds reuse. Installation counts, returns, and paired intervals appear in Appendix C.4, which separates update acceptance from terminal control scores.

Matched Snake phase profiles and installation traces support reuse within each regime (metric, results, and timing in Appendix C.5). These traces locate reuse within the continuing episode. Each Snake phase is matched to Fixed Code by seed, obstacle stratum, and regime. We compare mean rewards over the first five and last five actions of each phase, with windows defined independently of edit acceptance. This locates behavioral adaptation within a continuing regime rather than treating each phase as a fresh episode with a newly initialized controller.

4.3 Control Quality And Inference Expenditure Across Budgets

The seven-point Hard diagnostic extends evaluation beyond low budgets where Planning rarely releases actions. Across the same three designated seeds at 1K–32K, RTHL leads in mean score in 12 of 14 combinations of game and budget and exceeds AgileThinker in all 14 (Figure 5).

Action Availability And Control Quality. Planning availability (the fraction of turns using parsed plans rather than defaults) rises from 0.00% to 76.30% on Snake and from 0.33% to 91.33% on Freeway between 1K and 32K (Table 10). At 32K, RTHL still leads: 22.667 versus 1.667 on Snake and 36.667 versus 33.333 on Freeway. It leads all seven Snake points, including those with frequent plan-backed actions. AgileThinker's fast branch supplies direct actions on 3,975 of 3,990 turns (99.62%), yet RTHL has higher means at every diagnostic point. Together with the Fixed comparison, these results support online policy revision beyond making textual reasoning available.

Output Tokens Per Complete Episode. At 4K, we instrument six RTHL episodes: seed index 2 in each game's E/M/H strata, with separate 4,096-token caps for Critic and coding calls. Unlike the call or release setting B, usage sums all calls over complete episodes (90 Snake or 100 Freeway actions). Means cover three episodes per game and six overall. These runs do not replace performance trajectories. RTHL averages **12,049 output tokens per Snake episode** and **30,874 per Freeway episode** for diagnosis and source revision. Against episodes matched by game, stratum, and seed in the *4K primary baseline logs*, retaining architecture-native allocations, pooled output is 58.0% below Reactive, 97.7% below Planning, and 96.5% below AgileThinker. RTHL has the lowest mean output in both games (Table 4), consistent with rule reuse. Appendix C.3 details input/output totals, API calls, and role-level accounting for all six instrumented episodes.

Input Context And Output Volume. RTHL averages 345,377 input tokens per episode, including repeated evidence and code: 129,472 cache hits and 215,905 cache misses, each counted once. The

Table 4: Mean recorded output tokens per episode (10^3) on matched 4K task cells: three episodes per game, six pooled. Accounting details appear in Appendix C.3.

Method	Snake	Freeway	Both Games
Reactive	42.223	59.957	51.090
Planning	1229.131	647.163	938.147
AgileThinker	600.319	612.983	606.651
RTHL	**12.049**	**30.874**	**21.462**

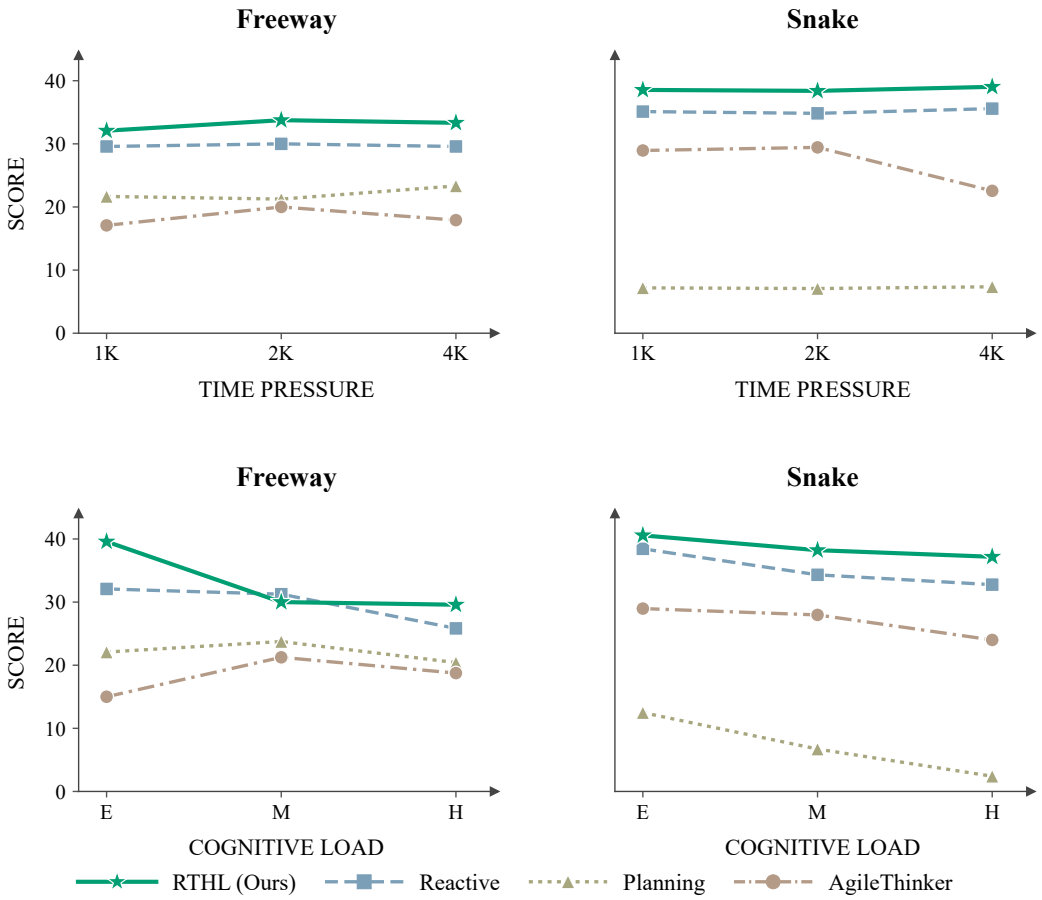


Figure 4: **RTHL leads every pooled comparison across environments and budget settings.** Upper: scores across budget settings. Lower: scores across difficulty strata. Each point averages 24 episodes. Scores are unnormalized. Higher is better. The y-axis ranges differ to show raw scores on their original task scales: Freeway, 0–44, and Snake, -54–32.

58.0–97.7% reductions concern recorded *output*, not total usage. Including input, RTHL averages 366,838 tokens, below even the recorded output of Planning (938,147) and AgileThinker (606,651) on these six cells. Adding baseline input would increase their totals. These are token-volume comparisons, not monetary-cost or local-controller CPU-time measurements.

Separate Accounting For The Wider Grid. Across 42 Hard episodes per method at seven budgets, Planning, AgileThinker, and Reactive output 60.085M, 47.004M, and 2.106M tokens. Episode means are 1,430,595, 1,119,135, and 50,131, respectively (Tables 13 and 12). AgileThinker's slow branch generates 97.4% of its recorded output. The 65,536-token cap applies per slow-model completion. Episode totals include all calls and unreleased text. Together, the grid and 4K sample support inference for reusable policy revisions rather than fresh action generation.

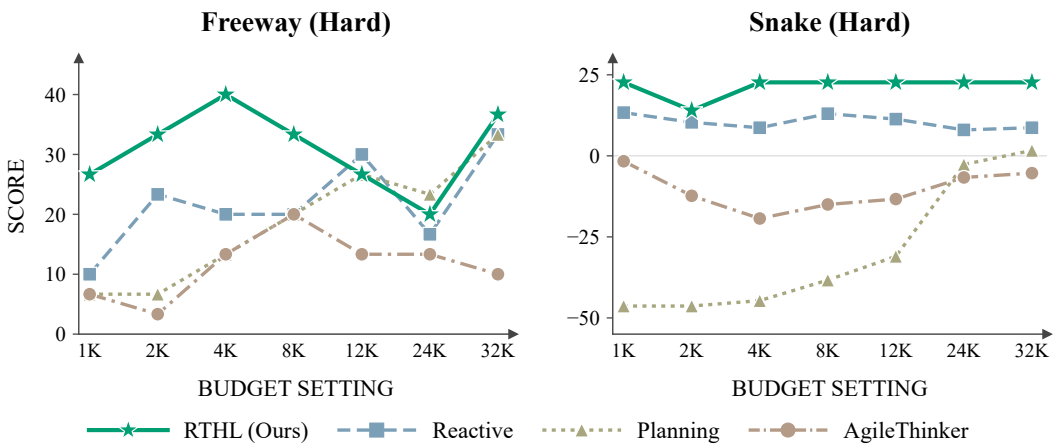


Figure 5: **RTHL leads 12 of 14 Hard-stratum operating points.** Mean terminal scores over the same three seeds per game. Panels use independent y-axis ranges. Settings are architecture-native, not compute-matched. Exact means and sample standard deviations appear in Appendix C.2.

5 Related Work

Games and Interactive Environments for Language Agents: WebArena and KORGym pause during deliberation, leaving delays untested Zhou et al. (2024b); Shi et al. (2026) despite latency-related control and safety concerns (Sinha et al., 2024). HAZARD and Real-Time Reasoning Gym address dynamic, time-sensitive interaction (Zhou et al., 2024a; Wen et al., 2026), but changing scenes alone do not test action-semantics revision. DiscoveryWorld's mechanism discovery and Baba Is AI's explicit-rule manipulation differ from hidden, externally imposed switches Jansen et al. (2024); Cloos et al. (2024). Our games test repeated within-episode adaptation to changing movement, hazard, and food rules. World state persists across these changes.

Inference Budgets, Token Efficiency, and Cost-Aware Agents: Test-time scaling adds reasoning compute, but overthinking need not yield corresponding gains Snell et al. (2024); Chen et al. (2024); Muennighoff et al. (2025). Budget-aware reasoning and thought compression shorten traces, not preserve discoveries for later control (Han et al., 2025; Xia et al., 2025; Hou et al., 2025). FrugalGPT and RouteLLM optimize model selection and query costs rather than executable reuse Chen et al. (2023); Ong et al. (2025). Following joint evaluation of performance and expenditure Kapoor et al. (2024), we measure control scores and per-episode tokens: RTHL reuses evidence-supported controller rules without fresh action-generating completions.

Executable Policies and Latency-Aware Adaptation: Code as Policies and ProgPrompt translate instructions into executable behavior (Liang et al., 2023; Singh et al., 2022). Voyager accumulates reusable skills through execution feedback (Wang et al., 2023). DPT-Agent and ReKep connect language reasoning to responsive control (Zhang et al., 2025; Huang et al., 2024). These goals differ from revising a deployed controller under repeated hidden rule changes. RTHL converts transition evidence into reusable policy updates through code revision, validated installation, phase-scoped reuse, and rollback. Accepted revisions then guide action selection on subsequent observations.

6 Conclusion

RTHL consolidates public transition evidence into validated controller revisions that later actions reuse without updating model weights. It leads all six pooled benchmark comparisons. Matched Fixed controls establish gains beyond the initial heuristic. Across 1K–32K, it leads 12 of 14 operating points and retains its Snake advantage as Planning becomes available. These results support executable policy consolidation as a mechanism for reusing deployment-time discoveries within an evolving episode with changing rules, persistent state, and bounded public evidence.

AI USE STATEMENT

We used Codex to assist with manuscript revision, literature searches, result interpretation, and figure and analysis-script preparation. The authors determined the research plan, implemented and tested the experimental system, verified results against logs, and checked the literature and manuscript.

ETHICS STATEMENT

The reported experiments evaluate software agents in simulated Freeway and Snake environments, without recruiting human participants or collecting personal records. The traffic and hazard scenarios are game abstractions, not interventions in real transportation systems. A central ethical concern for executable adaptation is that generated revisions can alter later behavior. Our implementation bounds the edit surface and uses program checks, history replay, phase scoping, and rollback to govern revisions within the tested interfaces. These controls are specific to the experimental setting. Use in systems affecting people would require independent security review, appropriate access restrictions, and domain-specific safety evaluation. We distinguish illustrative schematics from logged execution cases and report the scope of score and resource comparisons to support transparent interpretation of the evidence. The underlying benchmark is credited, and redistribution of derived code must retain the applicable license notices.

REPRODUCIBILITY STATEMENT

Sections 2 and 3 specify the evaluation design, agent interfaces, and adaptation procedure. Appendix A records game rules, phase schedules, public observations, seeds, trajectory-selection criteria, budget semantics, and statistical procedures. Appendix B details the starting controllers, editable functions, validation checks, and installation behavior. Appendix C provides complete scores, ablations, and adaptation analyses. The wider-budget provenance and instrumented resource accounting are documented separately in Appendices C.2 and C.3. Appendix E connects selected public transitions to recorded revision and installation events. Retained trajectory manifests, source hashes, and offline aggregation scripts support verification against the original logs without rerunning the model calls. This separates recomputation of reported statistics from a new stochastic execution of the evaluated agents.

REFERENCES

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Nathan Cloos, Meagan Jens, Michelangelo Naim, Yen-Ling Kuo, Ignacio Cases, Andrei Barbu, and Christopher J Cueva. Baba is ai: Break the rules to beat the benchmark. *arXiv preprint arXiv:2407.13729*, 2024.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. Discoveryworld: A virtual environment for developing and evaluating automated scientific discovery agents. *Advances in neural information processing systems*, 37:10088–10116, 2024.

Sayash Kapoor, Benedikt Stroebl, Zachary S Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. *arXiv preprint arXiv:2407.01502*, 2024.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

Jiajun Shi, Jian Yang, Jiaheng Liu, Xingyuan Bu, Jiangjie Chen, Junting Zhou, Kaijing Ma, Zhoufutu Wen, Bingli Wang, Yancheng He, et al. Korgym: A dynamic game platform for llm reasoning evaluation. *Advances in Neural Information Processing Systems*, 38:161286–161314, 2026.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew Foutter, Edward Schmerling, and Marco Pavone. Real-time anomaly detection and reactive planning with large language models. *arXiv preprint arXiv:2407.08735*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Yule Wen, Yixin Ye, Yanzhe Zhang, Diyi Yang, and Hao Zhu. Real-time reasoning agents in evolving environments. In *International Conference on Learning Representations*, volume 2026, pp. 126266–126292, 2026.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363, 2025.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shao Zhang, Xihuai Wang, Wenhao Zhang, Chaoran Li, Junru Song, Tingyu Li, Lin Qiu, Xuezhi Cao, Xunliang Cai, Wen Yao, et al. Leveraging dual process theory in language agent framework for real-time simultaneous human-ai collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4081–4108, 2025.

Qinhong Zhou, Sunli Chen, Yisong Wang, Haozhe Xu, Weihua Du, Hongxin Zhang, Yilun Du, Joshua B Tenenbaum, and Chuang Gan. Hazard challenge: Embodied decision making in dynamically changing environments. In *International Conference on Learning Representations*, volume 2024, pp. 50097–50113, 2024a.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024b.

A Evaluation Protocol

Task Specification. Freeway uses actions `U/D/S`. Turns 1–9 follow ordinary dynamics. T10–39, T40–69, and T70–99 each use one hidden mechanism, and T100 is the ordinary terminal step. The initial score is 100, every action costs one point, and a completed crossing earns ten, so the 100-action terminal score is $10\times$ the number of crossings. A collision resets only the player, while traffic state, timers, score, phase, and history continue, with no additional collision penalty. Snake uses actions `U/D/L/R` on an 8×8 grid with a traversable 6×6 interior and a fixed player length of one. T1–30, T31–60, and T61–90 each use one hidden mechanism. The board maintains three positive and three negative foods with an ordinary lifespan of ten turns. Positive food gives $+1$, negative food and lethal transitions give -1, and lethal transitions reset the player without terminating the episode.

Hidden Mechanisms. Each episode samples three distinct implemented mechanisms without replacement. Freeway's seven families alter vehicle displacement, periodic direction reversal, approach-dependent acceleration, player command direction, danger from remaining in one lane, collision width, or periodic two-pass vehicle motion. Snake's six families implement food reflection, food-polarity swapping, outward food drift, positive-food-column hazards, two-cell movement, or reversed movement. Freeway exposes zero-based phase-local time during its hidden phases, while Snake exposes values 1–30. These clocks identify change boundaries, not the active rule.

Information Interface. All methods receive the same observable state, public outcome telemetry, phase boundaries, and rule-agnostic transition evidence. None receives active rule identity, hidden parameters, or the seed-to-rule mapping. On Snake, every method receives the same ten-family candidate catalog: six implemented families and four distractors. On Freeway, the baselines receive the seven implemented families, while the RTHL Critic receives those seven plus three distractors. The RTHL search space is therefore a strict superset rather than a narrower catalog. Evidence retains up to 12 recent transitions, up to 24 phase-local anomalies, ordinary-dynamics residuals, and phase aggregates. The Snake distractors are obstacle changes, lifespan changes, patience rewards, and teleportation. The Freeway Critic's distractors are temporary lane impassability, lane transfer, and vehicle disappearance. Public evidence is derived from observed pre- and post-action state and outcome telemetry. The tracker does not inspect the sampled rule sequence.

Architecture-Native Operating Settings. The 1K, 2K, and 4K settings parameterize each architecture according to its native call pattern. Reactive uses a per-turn reasoning setting. Planning logically releases a completion generated from a planning-start snapshot. AgileThinker combines slow logical release with a reactive call. RTHL applies the cap separately to Critic and coding calls and permits continuation. The reported values are architecture-native operating points. The paper does not interpret them as a controlled total-compute scaling curve.

Budget-Diagnostic Provenance. The Hard-stratum diagnostic selects seed indices 0, 1, and 2 and evaluates $B \in \{1, 2, 4, 8, 12, 24, 32\}$K. Planning uses dedicated runs with a fixed 65,536-token completion ceiling and logical release B. AgileThinker likewise uses a fixed slow-model ceiling,

with `time_pressure=internal_budget=`B. Increasing B thus changes its method-native slow-release and fast-branch allocation together. Reactive and RTHL reuse the designated Hard trajectories from the primary evaluation at 1/2/4K and use dedicated runs at 8/12/24/32K. The selected grid contains 168 complete entries, including 36 reused primary trajectories. Together, the 576-entry primary evaluation, 144 ablation episodes, and this diagnostic contain 852 distinct complete episodes. Six additional resource-only RTHL repeats are documented in Appendix C.3. Their scores do not replace these performance records. It is a method-specific nominal-budget diagnostic, not a total-compute-matched comparison or a single globally frozen compute sweep. Scores never enter trajectory selection.

Temporal Execution. Evaluation follows logical environment turns. The heuristic selects the current action before post-transition adaptation. The Critic may wait for more evidence or queue an edit. Queued coding work is normally serviced on a later logical turn. A remaining queue can be drained after the final action for logging, but any source installed then cannot affect terminal score. Language-model calls in this stage block advancement of the Python loop, so the evaluated protocol makes no wall-clock concurrency claim. An accepted edit can affect later actions but never the action whose evidence produced it.

Seeds, Completion, And Statistics. Snake Easy, Medium, and Hard use seeds 1000–1007, 5000–5007, and 8000–8007. Freeway uses Easy seeds {1000,1001,1002,1003,1013,1014,1016,1018}, Medium seeds {1069,1093,1536,1858,1338,2496,1933,1863}, and Hard seeds {1447,2408,2418,2661,1100,1944,1310,2453}. For RTHL and its Fixed control, each seed restores the same environment-specific starting controller. Installed RTHL revisions never transfer across seeds. Each stratum-specific cell contains eight complete episodes, and pooled values combine the three strata (24 episodes). Trajectory selection is deterministic: for each designated seed, the dataset uses the latest complete trajectory produced under the final interface, and scores never enter the selection rule. Standard deviations are sample standard deviations over episodes. The 95% confidence intervals for differences between Full and Fixed use Student's t distribution with 23 degrees of freedom over 24 episode pairs matched by seed and stratum. Here t names the statistical distribution, not an interaction-turn index.

Phase-Aligned Analysis. For the comparison between Full and Fixed on Snake at 1K, every phase is matched by stratum, seed, and phase index. Per-action reward is the change in cumulative score across that action. Equation 6 compares turns 1–5 with turns 26–30 inside each 30-turn phase. The reported confidence interval is a two-sided Student's t interval over 24 matched episode-level differences after averaging the three phase differences within each episode. Installation coverage counts a phase only when its first accepted candidate passes the recorded contract and replay checks, changes source reachable from the action controller, and is installed before that phase's final action, ensuring at least one later in-phase controller call. Action reachability is an analysis criterion for this coverage statistic, not an additional installation gate. It excludes accepted but uncalled helper code and does not assert that each counted edit improves reward.

B RTHL Implementation Contracts

Table 5 summarizes the installation transaction. The following details specify the starting controllers and the environment-specific contracts used by that transaction.

Table 5: **Guarded commit is the action-authority boundary.** Evidence and candidate source remain outside the action path until applicable checks pass and installation succeeds. An action-reachable revision can govern only later controller calls.

Stage	Transaction	Action authority
Evidence	A completed public transition updates bounded state D_t. $\mathrm{Critic}(D_t, \ell_t)$ returns **WAIT** or enqueues $q_t \in \mathcal{Q}_{\mathrm{edit}}$.	**None**
Candidate	At a later service opportunity $\tau > t$, propose $\widetilde{\psi}_\tau = G(\psi_\tau, q_t)$ while installed source remains ψ_τ.	**None**

Continued on next page

Table 5 (continued)

Stage	Transaction	Action authority
Commit	Evaluate v_τ^g in Equation 4 and attempt installation. A failed gate or installation keeps $\psi_{\tau+1} = \psi_\tau$.	**Pending**
Installed	A successful commit sets $\psi_{\tau+1} = \widetilde{\psi}_\tau$. Only action-reachable changes enter subsequent calls to $h_{\psi_{\tau+1}}$.	**Later turns**

Initial Controllers. Snake uses bounded utility search over positions and time steps, with maximum depth 15 and at most 2,000 node expansions per search. Candidate food targets are ranked by net food reward, arrival depth, and destination mobility. The controller first seeks a path avoiding negative food, then relaxes that constraint, with safe wandering as a fallback. Freeway uses collision-aware lookahead of depth four, extended to six when its anomaly score exceeds 0.25, together with safe-entry and escape fallbacks. In both games, ordinary-mechanics predictors and safety functions provide the initial search model. The same source and bounded outcome-memory logic initialize Full and Fixed Code.

Observable-History Validation. Replay compares a candidate with the currently installed program, not always with the initial source. Freeway starts from the latest 60 public transitions, selecting the eligible-phase subset when present. Vehicle prediction checks use observed endpoints at horizons of one to six steps: they preserve correct predictions and require progress on existing misses when present. Lane-safety checks protect observed safe positions and compare a score weighting false positives by five against missed unexplained collisions. Action-map checks protect confirmed command behavior and test the displacement discrepancies motivating the request. The body-direction hook receives contract and smoke checks and phase gating.

Snake uses all completed current-phase transitions for direct-support checks and the latest 12 for comparative replay. Movement updates must not increase prediction misses on survived transitions. Hazard updates must not increase unsafe false positives on observed survived states. New movement and hazard rules ordinarily require two independent supporting transitions, or one when replacing an active hook. Food-forecast checks compare observable positions, lifespans, and polarity while allowing unknown random respawns. Candidate misses may not exceed the larger of the installed program's miss count and one. These are evidence-bounded support and regression tests, rather than a requirement to explain every historical transition.

Phase Scope And Monitoring. Freeway applies dedicated runtime rollback to prediction, action-map, and lane-safety hooks, and its body-direction hook is phase-gated. Snake's three primary hooks receive automatic phase gating, and supported movement, hazard, or food-forecast contradictions can restore the pre-edit program. Phase gates restrict application of learned hooks while preserving the ordinary controller outside their eligible phase. Prior source is retained for restoration on the covered paths.

Review Scheduling. Freeway uses collision, direction-inversion, residual, anomaly, hook-reality, and scheduled review signals. A clean non-boundary command inversion can also queue a narrowly scoped action-map request. Snake uses respawn, anomaly, and scheduled review signals, while hook contradictions feed monitoring and later reviews. Both implementations organize scheduled opportunities around five-action evidence windows.

C Complete Benchmark-Stratified Results And Additional Analyses

C.1 Complete Benchmark-Stratified Performance

Tables 6 and 7 report every stratum-specific comparison. RTHL records a higher mean than Planning and AgileThinker in all 18 cells, achieves the highest mean in all nine Snake cells, and retains the highest pooled mean at all six operating points.

Table 6: Freeway terminal score (mean ± sample standard deviation). Each E/M/H cell contains eight seeds, and Overall pools 24 episodes. Bold marks the highest pooled mean at each operating point.

Setting	Method	Easy	Medium	Hard	Overall
1K	Reactive	35.000 ± 17.728	31.250 ± 22.321	22.500 ± 19.821	29.583 ± 19.886
1K	Planning	20.000 ± 22.039	23.750 ± 16.850	21.250 ± 20.310	21.667 ± 19.035
1K	AgileThinker	17.500 ± 14.880	18.750 ± 13.562	15.000 ± 21.381	17.083 ± 16.280
1K	RTHL	40.000 ± 16.036	30.000 ± 15.119	26.250 ± 14.079	$\mathbf{32.083 \pm 15.598}$
2K	Reactive	31.250 ± 24.165	31.250 ± 16.421	27.500 ± 19.821	30.000 ± 19.560
2K	Planning	21.250 ± 21.002	23.750 ± 16.850	18.750 ± 18.077	21.250 ± 18.013
2K	AgileThinker	15.000 ± 16.036	23.750 ± 10.607	21.250 ± 22.952	20.000 ± 16.940
2K	RTHL	38.750 ± 17.269	31.250 ± 14.577	31.250 ± 14.577	$\mathbf{33.750 \pm 15.269}$
4K	Reactive	30.000 ± 22.039	31.250 ± 18.077	27.500 ± 17.525	29.583 ± 18.528
4K	Planning	25.000 ± 20.702	23.750 ± 16.850	21.250 ± 19.594	23.333 ± 18.337
4K	AgileThinker	12.500 ± 8.864	21.250 ± 12.464	20.000 ± 20.702	17.917 ± 14.738
4K	RTHL	40.000 ± 15.119	28.750 ± 17.269	31.250 ± 16.421	$\mathbf{33.333 \pm 16.330}$

Table 7: Snake terminal score (mean ± sample standard deviation). Each E/M/H cell contains eight seeds, and Overall pools 24 episodes. Bold marks the highest pooled mean at each operating setting.

Setting	Method	Easy	Medium	Hard	Overall
1K	Reactive	20.125 ± 4.970	12.625 ± 8.749	11.125 ± 4.883	14.625 ± 7.365
1K	Planning	-29.500 ± 8.018	-41.000 ± 13.512	-49.375 ± 18.631	-39.958 ± 15.813
1K	AgileThinker	8.375 ± 5.927	5.125 ± 6.402	-5.750 ± 5.701	2.583 ± 8.433
1K	RTHL	24.750 ± 5.600	20.000 ± 4.209	19.250 ± 8.031	$\mathbf{21.333 \pm 6.384}$
2K	Reactive	19.500 ± 8.036	12.875 ± 5.718	9.875 ± 11.457	14.083 ± 9.297
2K	Planning	-30.000 ± 7.616	-41.000 ± 13.395	-49.500 ± 19.420	-40.167 ± 15.923
2K	AgileThinker	6.000 ± 6.633	4.750 ± 6.519	-0.125 ± 10.589	3.542 ± 8.230
2K	RTHL	25.250 ± 5.800	20.875 ± 4.764	17.000 ± 6.234	$\mathbf{21.042 \pm 6.389}$
4K	Reactive	23.750 ± 4.496	13.750 ± 5.339	9.125 ± 8.442	15.542 ± 8.683
4K	Planning	-29.250 ± 7.924	-40.625 ± 14.687	-49.000 ± 18.792	-39.625 ± 16.143
4K	AgileThinker	-6.500 ± 4.598	-7.875 ± 6.490	-15.375 ± 7.230	-9.917 ± 7.144
4K	RTHL	25.875 ± 3.357	21.250 ± 5.007	19.750 ± 6.628	$\mathbf{22.292 \pm 5.614}$

C.2 Hard-Stratum Method-Specific Budget Diagnostic

Tables 8 and 9 report the complete common grid described in Appendix A. Each value summarizes the same three designated Hard seed indices. The diagnostic is descriptive: its role is to test whether the primary result is confined to one narrow operating point, not to support a three-seed significance claim.

Table 8: Snake-H terminal score across the wider method-specific budget diagnostic (mean ± sample standard deviation over three seeds). Bold marks the highest mean at each budget.

Setting	Reactive	Planning	AgileThinker	RTHL
1K	13.333 ± 4.041	-46.333 ± 6.028	-1.667 ± 0.577	$\mathbf{22.667 \pm 5.508}$
2K	10.333 ± 17.039	-46.333 ± 6.028	-12.333 ± 5.508	$\mathbf{14.000 \pm 5.568}$
4K	8.667 ± 13.204	-44.667 ± 6.506	-19.333 ± 4.163	$\mathbf{22.667 \pm 5.508}$
8K	13.000 ± 6.000	-38.333 ± 10.693	-15.000 ± 10.392	$\mathbf{22.667 \pm 5.508}$
12K	11.333 ± 7.024	-31.000 ± 2.646	-13.333 ± 11.676	$\mathbf{22.667 \pm 5.508}$
24K	8.000 ± 8.660	-2.667 ± 11.676	-6.667 ± 14.503	$\mathbf{22.667 \pm 5.508}$
32K	8.667 ± 14.572	1.667 ± 7.371	-5.333 ± 14.154	$\mathbf{22.667 \pm 5.508}$

Table 9: Freeway-H terminal score across the wider method-specific budget diagnostic (mean ± sample standard deviation over three seeds). Bold marks the highest mean at each budget.

Setting	Reactive	Planning	AgileThinker	RTHL
1K	10.000 ± 10.000	6.667 ± 5.774	6.667 ± 5.774	$\mathbf{26.667 \pm 5.774}$
2K	23.333 ± 20.817	6.667 ± 5.774	3.333 ± 5.774	$\mathbf{33.333 \pm 5.774}$
4K	20.000 ± 0.000	13.333 ± 15.275	13.333 ± 5.774	$\mathbf{40.000 \pm 10.000}$
8K	20.000 ± 10.000	20.000 ± 10.000	20.000 ± 0.000	$\mathbf{33.333 \pm 5.774}$
12K	$\mathbf{30.000 \pm 10.000}$	26.667 ± 15.275	13.333 ± 5.774	26.667 ± 15.275
24K	16.667 ± 5.774	$\mathbf{23.333 \pm 15.275}$	13.333 ± 11.547	20.000 ± 10.000
32K	33.333 ± 5.774	33.333 ± 5.774	10.000 ± 10.000	$\mathbf{36.667 \pm 5.774}$

Table 10: **Additional logical release progressively exposes Planning's action queue.** Percentage of Hard-stratum actions backed by a parsed plan with `max_tokens`=65,536 fixed. Each entry aggregates three designated seeds (270 Snake or 300 Freeway turns). Remaining turns execute the environment-specific default.

Game	1K	2K	4K	8K	12K	24K	32K
Snake	0.00%	0.00%	2.59%	16.67%	30.74%	66.30%	76.30%
Freeway	0.33%	4.00%	13.33%	38.33%	65.33%	80.33%	91.33%

Table 11: **The four architectures spend inference at different locations.** Historical accounting aggregates the same 42 selected diagnostic episodes per method. Token values are provider-recorded completion tokens retained in the baseline CSVs, excluding prompt tokens. The historical RTHL runs retained Critic/Coding artifacts. Direct provider measurements are reported separately for the six 4K repeats in Appendix C.3. Logged artifacts need not equal HTTP calls when continuations or retries occur. Reactive forcing fields are absent from the reused 1K–4K schema, so its forced-response count is a retained lower bound.

Method	Retained inference records	Tracked completion	Action producer
Reactive	3,990 main-response records plus at least six retained forced-response records	2.106M main-response tokens plus six retained forced tokens	Latest-state LM action or default after forcing
Planning	1,795 slow-plan request records	60.085M slow-plan tokens	Released snapshot plan or default
AgileThinker	3,990 reactive, 1,576 slow-plan, and 45 forced-response records	47.004M total, 97.4% from the slow branch	Latest-state reactive LM action or default after forcing
RTHL	1,330 Critic-response and 351 coding-attempt files	Not instrumented in this grid	Installed deterministic heuristic with no action-generating LM completion

RTHL records the highest mean in 12 of the 14 complete operating points, including every Snake-H point, and exceeds AgileThinker in all 14. Planning's plan-backed rate reaches 76.30% on Snake and 91.33% on Freeway at 32K, while RTHL retains the higher mean in both environments. AgileThinker's current action is directly available on 3,975/3,990 turns, yet most of its tracked completion tokens are spent in the slow textual branch. Together, the score, availability, and accounting views distinguish producing more reasoning from installing an evidence-supported rule for repeated latest-state execution. Table 11 characterizes the auditable resource topology rather than a total-token-normalized performance ranking.

Table 12: **Per-episode resource measurements at each diagnostic setting.** Each entry averages three complete Hard-stratum seeds, in 10^3 tokens. The baseline columns contain retained provider completion usage, excluding input prompts, so they are not total API costs. Table 13 gives pooled baseline means and sample SDs. Table 14 reports the separate instrumented RTHL sample.

Game	B	Reactive	Planning	AgileThinker
Snake	1K	44.562	1538.081	461.541
	2K	42.478	1552.144	666.612
	4K	42.953	1460.066	755.352

Continued on next page

Table 12 (continued)

Game	B	Reactive	Planning	AgileThinker
	8K	45.284	1489.306	981.633
	12K	42.638	1552.749	1158.875
	24K	42.669	1840.660	1775.113
	32K	41.373	2157.728	2123.153
Freeway	1K	55.328	678.538	341.275
	2K	52.915	754.237	524.477
	4K	55.989	763.925	724.046
	8K	54.488	988.572	940.245
	12K	56.095	1143.320	1189.747
	24K	56.295	1907.855	1828.773
	32K	68.768	2201.144	2197.045

Table 13: **Historical Hard-grid completion usage per episode.** Mean $\pm$ sample SD in 10^3 provider-recorded output tokens. Each game pools 21 episodes (seven settings, three seeds), and Both pools 42. These means describe the seven-budget diagnostic, not the six-cell 4K resource comparison in Table 4. Prompt tokens are excluded. Forcing is counted where retained.

Method	Snake	Freeway	Both Games
Reactive	43.137 ± 2.779	57.126 ± 10.707	50.131 ± 10.479
Planning	1655.819 ± 262.276	1205.370 ± 578.994	1430.595 ± 499.047
AgileThinker	1131.754 ± 584.836	1106.515 ± 655.302	1119.135 ± 613.581

C.3 Instrumented Per-Episode API Usage At 4K

Measured RTHL Usage: Diagnosis And Revision, Not Action Generation. Table 14 reports provider-returned input, output, and total tokens. RTHL averages **12,049 output tokens per Snake episode** and **30,874 per Freeway episode**. Including input prompts, the corresponding totals are 318,509 and 415,168 tokens per episode. Across the six runs, the mean is 21,462 output tokens and 366,838 input-plus-output tokens. Thus the output column measures generated tokens, while the total column also includes the evidence and source-code context supplied to the models. Both columns report per-episode averages.

Table 14: **Measured RTHL API usage per complete episode at 4K.** Input, output, and total columns give mean provider-reported token counts in 10^3 tokens, and Calls gives mean API calls. Each game uses one episode from each E/M/H stratum at seed index 2. Here n counts complete episodes pooled equally across strata. All Critic and coding requests, including episode-associated terminal work, are counted. Per-episode and role-level details appear in Tables 15 and 16.

Game	n	Input	Output	Total	Calls
Snake	3	306.460	12.049	318.509	31.67
Freeway	3	384.294	30.874	415.168	47.33
Both games	6	345.377	21.462	366.838	39.50

The sample contains 237 API calls: 191 Critic calls and 46 coding calls, averaging 31.83 and 7.67 per episode, respectively. Their mean output volumes are 18,963 and 2,498 tokens per episode. All 237 calls returned complete usage and ended with a normal stop. The longest completion was 1,769 tokens, below the 4K cap. The measured output volume therefore does not result from truncation at that ceiling. All 570 environment actions in this sample are produced by the installed program, while language-model output is spent on diagnosis and source revision for subsequent decisions.

The resource sample uses the third designated seed (index 2) in each difficulty stratum of each game. Actual seeds are 1002/5002/8002 for Snake E/M/H and 1002/1536/2418 for Freeway E/M/H. RTHL uses DeepSeek-V4-Flash for both roles, a 4,096-token output cap per request, the full validation and scope mechanisms, and the existing continuation setting of three. Each run starts from its environment-specific initial controller. Snake runs for 90 actions and Freeway for 100. All six runs complete these horizons.

For every API response, the logger retains provider-reported `prompt_tokens`, `completion_tokens`, and `total_tokens`, the role, and the finish reason. Summing the 237 response records reproduces the seed-level JSON and CSV summaries exactly: 2,072,261 input tokens plus 128,769 output tokens equal 2,201,030 total tokens across the six episodes. There are no recorded request errors or responses with missing usage. Every response ends with `stop`, so no length-triggered continuation is required in this sample. The logger covers episode-associated terminal requests as well as those issued during play. The accounting does not condition inclusion on whether an edit is accepted. Provider-unreported SDK-internal retry usage is outside this measurement.

Table 15: **Complete per-episode RTHL usage at 4K.** Each row is one complete resource run, not a multi-seed mean. Token columns contain exact provider counts, and Calls includes both Critic and coding. All rows use seed index 2. This sample supplements resource measurement without replacing any historical performance trajectory.

Game	Stratum	Seed	Calls	Input	Output	Total
Snake	E	1002	27	254,197	9,804	264,001
	M	5002	33	318,668	12,761	331,429
	H	8002	35	346,514	13,583	360,097
Freeway	E	1002	51	399,286	29,585	428,871
	M	1536	35	316,886	26,196	343,082
	H	2418	56	436,710	36,840	473,550

Table 16: **Per-episode RTHL usage by inference role.** Mean API calls and tokens (10^3). Each game averages its three E/M/H episodes, and Both averages all six (Table 15).

Game	Role	Calls	Input	Output	Total
Snake	Critic	26.33	246.610	10.924	257.534
	Coding	5.33	59.850	1.125	60.975
Freeway	Critic	37.33	283.447	27.002	310.449
	Coding	10.00	100.847	3.872	104.719
Both	Critic	31.83	265.029	18.963	283.992
	Coding	7.67	80.348	2.498	82.847

The two role rows sum to the game's episode mean, up to rounding. The aggregate input count comprises 776,832 cache-hit tokens and 1,295,429 cache-miss tokens. The cache-hit fraction is 37.5%. Cache-hit counts are subsets of input tokens, not additional tokens. Counts therefore describe input/output volume rather than monetary charges or local-controller CPU time. The per-game averages give E/M/H equal weight, and the six-run mean also weights the two games equally. There is one observed episode per stratum, so these are descriptive resource-sample means, not repeated-seed uncertainty estimates for each difficulty.

For Table 4, baseline records are selected from the frozen *primary* 4K evaluation using the same six combinations of game, stratum, and seed. They are not the dedicated 65,536-ceiling Planning diagnostic runs. We sum retained fast- and slow-completion fields within each episode before averaging. Those selected historical schemas do not retain prompt or additional forcing usage, so the comparison concerns the recorded completion volume. The RTHL row contains the full observed completion usage of the six instrumented repeats, not re-tokenized saved text. Matching task cells does not impose equal architectural budgets or make the baseline runs contemporaneous with the resource repeats. Selection and source hashes are frozen independently of the scores. None of the resource-repeat scores replace the primary or Hard-diagnostic results.

C.4 Matched Code And Component Ablations

Table 17 reports all three controls against Full RTHL using the same initial controller, bounded memory, and eight seeds per stratum. Fixed Code (Fixed Heuristic) disables online updates and makes no LM calls. The other variants use the 1K Critic and coding caps. No Replay/Rollback disables observable-history replay and post-installation rollback, retaining syntax, interface, purity,

determinism, shape, smoke, and phase-scope checks. No Phase Scoping disables runtime/source gates that restrict edited hooks to their supporting phase. It does not hide phase boundaries from the Critic. Replay and rollback remain enabled in that variant.

Table 17: **Matched ablations distinguish enabling online updates from controlling their installation and scope.** Terminal-score mean ± sample standard deviation. Each stratum uses eight seeds and Overall pools 24. Full and online variants use 1K, while Fixed is setting-independent. No R/R and No Scope abbreviate No Replay/Rollback and No Phase Scoping. Bold identifies the complete RTHL system, not a per-column ranking.

Game	Variant	Easy	Medium	Hard	Overall
Snake	Fixed	11.125 ± 10.869	9.500 ± 10.379	12.625 ± 9.841	11.083 ± 9.995
Snake	No R/R	25.875 ± 4.257	21.375 ± 4.984	19.375 ± 8.467	22.208 ± 6.527
Snake	No Scope	23.250 ± 8.498	20.500 ± 4.342	16.875 ± 7.864	20.208 ± 7.325
Snake	Full RTHL	$\mathbf{24.750 \pm 5.600}$	$\mathbf{20.000 \pm 4.209}$	$\mathbf{19.250 \pm 8.031}$	$\mathbf{21.333 \pm 6.384}$
Freeway	Fixed	32.500 ± 17.525	25.000 ± 16.036	23.750 ± 15.059	27.083 ± 16.011
Freeway	No R/R	41.250 ± 17.269	28.750 ± 22.952	25.000 ± 15.119	31.667 ± 19.262
Freeway	No Scope	32.500 ± 13.887	33.750 ± 24.458	21.250 ± 15.526	29.167 ± 18.631
Freeway	Full RTHL	$\mathbf{40.000 \pm 16.036}$	$\mathbf{30.000 \pm 15.119}$	$\mathbf{26.250 \pm 14.079}$	$\mathbf{32.083 \pm 15.598}$

Paired Interpretation. Full improves over Fixed by $+10.250$ Snake points (95% CI $[6.796, 13.704]$) and $+5.000$ Freeway points ($[1.478, 8.522]$). Subtracting the scores of No Replay/Rollback from those of Full gives differences of -0.875 ($[-1.889, 0.139]$) and $+0.417$ ($[-3.802, 4.635]$), respectively. For No Phase Scoping, the corresponding differences are $+1.125$ ($[-1.885, 4.135]$) and $+2.917$ ($[-2.436, 8.269]$). Intervals use differences from 24 episode pairs matched by seed and stratum. The Fixed comparison establishes the benefit of enabling the online stack. The component controls characterize how updates are governed. Their intervals span zero, so we do not assign either component a separate mean-return improvement.

Update Selectivity. Snake records 71, 92, and 74 accepted edits for Full, No Replay/Rollback, and No Phase Scoping. Freeway records 218/45/10/35, 166/105/0/105, and 218/44/11/33 for attempts/installs/rollbacks/surviving installs, respectively. These are update-event counts rather than episode-return measures. They document the installation and withdrawal decisions made by the guarded adaptation path.

C.5 Within-Phase Reward And Installation Timing

Useful adaptation must arrive before the hidden regime changes again. We examine this requirement through complementary reward and source traces in Snake. At 1K, we match each Full phase to the same Fixed seed, obstacle stratum, and 30-turn regime, then measure the opening-to-closing reward shift:

$$\Delta_{\text{phase}} = \overline{r}_{26:30} - \overline{r}_{1:5}, \tag{6}$$

Here Δ_{phase} is the closing-minus-opening change in mean reward per action within one episode phase. The bar denotes an arithmetic mean of step rewards: $\overline{r}_{26:30}$ averages phase-local turns 26 through 30, and $\overline{r}_{1:5}$ averages turns 1 through 5, inclusive. These subscripts index positions within a phase, not episode-global turns. The comparison windows are fixed independently of when an edit is accepted.

Table 18: **Online revision produces a more positive within-phase reward shift in every Snake stratum.** Each stratum contains 24 matched phase windows (eight episodes × three phases), and Overall summarizes 72 windows. Entries are Δ_{phase} in reward per action, and the final column reports the paired difference obtained by subtracting Fixed from Full. Higher is better.

Stratum	Full RTHL	Fixed	Difference
1 obstacle	$\mathbf{+0.225}$	-0.017	$\mathbf{+0.242}$
5 obstacles	$\mathbf{+0.125}$	-0.050	$\mathbf{+0.175}$
8 obstacles	$\mathbf{+0.100}$	-0.075	$\mathbf{+0.175}$

Continued on next page

Table 18 (continued)

Stratum	Full RTHL	Fixed	Difference
Overall	$+\mathbf{0.150}$	-0.047	$+\mathbf{0.197}$

Full has a more positive opening-to-closing shift in all three obstacle strata (Table 18). Across 24 matched episodes, Full exceeds Fixed by $+0.197$ reward per action (95% CI $[0.114, 0.280]$, using episode-level averages of the three phases). Thus the online stack improves not just terminal score, but the within-regime reward profile relative to the fixed controller.

Source traces establish that the learned artifact becomes executable before the supporting regime ends. Excluding installations without a later in-phase controller call, RTHL installs an accepted action-path revision in 57/72, 59/72, and 59/72 Snake phases at 1K, 2K, and 4K, respectively, totaling 175 of 216 phases (81.0%). Among phases with such an installation, the median first accepted action-path installation occurs within the first nine phase turns at all three settings, leaving most of the 30-turn regime available for reuse. Together, the matched reward profile and installation timing show that action-reachable revisions are installed early enough for within-regime reuse.

C.6 Full Planning Availability

Table 19 reports the complete fraction of actions supplied by a parsed Planning sequence. Other turns execute the environment-specific default action.

Table 19: Percentage of Planning actions backed by a parsed plan under the evaluated logical-release protocol. Each cell aggregates eight episodes (720 Snake or 800 Freeway turns). Other turns use the environment-specific default.

Game	Setting	Easy	Medium	Hard
Snake	1K	0.00%	0.00%	3.33%
Snake	2K	0.69%	0.83%	0.28%
Snake	4K	5.00%	2.22%	1.39%
Freeway	1K	1.50%	2.00%	0.00%
Freeway	2K	5.25%	4.75%	3.88%
Freeway	4K	16.25%	20.12%	13.62%

D Benchmark Coverage Of Evolving Dynamics

Our environments combine two distinct adaptation demands: the world keeps changing, and the rules governing action consequences change within the same episode. Table 20 separates these properties from inference-time constraints. This comparison concerns benchmark design, not agent scores, and uses the settings described in the cited publications.

Table 20: **Benchmark coverage of changing states and changing rules.** ✓: included. ×: not part of the cited benchmark design under the definitions below. †Logical token-budget and plan-release protocol, rather than wall-clock measurement.

Benchmark	World dynamics	Timed inference	Rule changes	Hidden rule switches
WebArena (Zhou et al., 2024b)	×	×	×	×
ScienceWorld (Wang et al., 2022)	✓	×	×	×
DiscoveryWorld (Jansen et al., 2024)	✓	×	×	×
Baba Is AI (Cloos et al., 2024)	×	×	✓	×
HAZARD (Zhou et al., 2024a)	✓	×	×	×
RTR Gym (Wen et al., 2026)	✓	✓	×	×
Evolving Freeway (ours)	✓	✓†	✓	✓
Evolving Snake (ours)	✓	✓†	✓	✓

Column Definitions. *World dynamics* denotes simulated processes such as object motion, heat transfer, growth, or hazard propagation that evolve beyond the immediate effect of the agent's command. It does not require the simulator to run while an LM call is pending. *Timed inference* means that inference allocation affects which action or plan is available at an environment decision boundary, rather than merely limiting the number of actions. *Rule changes* denotes changes to active transition or goal/reward semantics within a task, rather than ordinary state evolution under the same rules. *Hidden rule switches* requires repeated, externally triggered changes of the active mechanism within one continuing episode, without revealing mechanism identity or resetting the world at the change boundary.

What The Comparison Establishes. ScienceWorld's physical processes and DiscoveryWorld's simulated organisms provide evolving state. Discovering an unknown law does not itself make that law change during evaluation. Baba Is AI explicitly supports changing rules through visible word tiles: its original evaluation asks for plans from an initial image, rather than response to autonomous hidden switches. HAZARD supplies spreading disasters, while Real-Time Reasoning Gym (RTR Gym) couples reasoning availability to ongoing game evolution. Our extensions add repeated changes to the mechanisms themselves: three sampled rule regimes alter movement, hazards, or food behavior while episode state persists. Phase boundaries are public, but active rule identities are not. The resulting target is *continued control under changing action consequences*, not simply longer interaction or a more visually dynamic scene.

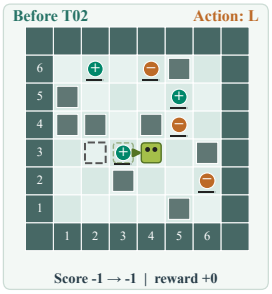
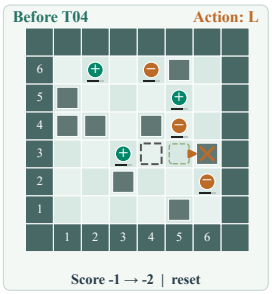
E ACTUAL EXECUTION CASES

Two recorded RTHL episodes illustrate evidence-to-code adaptation. Both use the main benchmark's 4K setting, Hard stratum, and seed index 0, selected without score ranking. Each case shows eight selected turns reconstructed from logs, not captured screenshots or new rollouts.

E.1 SNAKE: REVISING MOVEMENT AND HAZARD CHECKS

Actual seed 8000, with 90 actions and terminal score 19. The sequence shows a movement correction, its phase-scoped use, and a later hazard-check installation within the same episode.

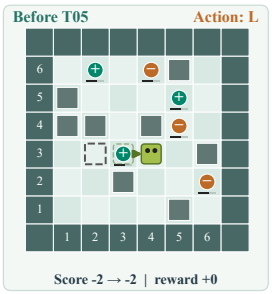
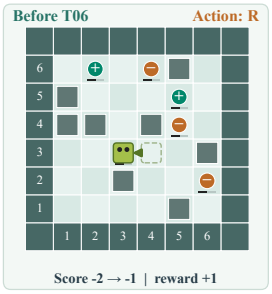
Table 21: **Phase-scoped adaptation in a logged Snake episode.**

Turn	Logged State And Action	Observation And Policy Update
T02		**Movement mismatch** L moves the player from (3,3) to (4,3), although the current predictor expects (2,3). The observed displacement is opposite to the command.
T04		**Evidence triggers EDIT** L from (5,3) reaches obstacle cell (6,3), followed by a reset to (3,3). After this transition, the Critic combines T1–T4 evidence and requests a reversed-movement hook.

Each board shows the pre-action state. Faded/dashed players mark starting positions, while solid players mark observed landings. Crosses mark failed landings followed by reset, not the respawn location. Dashed squares mark differing predicted cells. Solid gray blocks are obstacles. The $+/-$ symbols mark food values and short bars show remaining food life. Coordinates increase rightward and upward.

Snake Execution Case (Continued, 2/3)

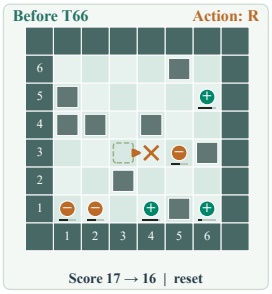
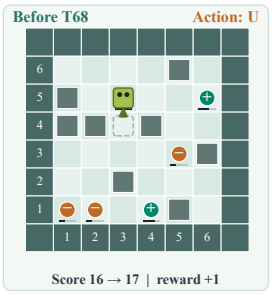
Table 21 (continued): Hard, 4K, seed index 0.

Turn	Logged State And Action	Observation And Policy Update
T05		**Validate and install** This turn still uses the old predictor. After the transition, the coding candidate passes contract and replay checks. The saved source adds a reversed-movement hook, scoped by installer metadata to the first phase.
T06		**Use the revised policy** R now moves left, from (4,3) to (3,3), exactly as the revised predictor expects. Positive food is collected (+1). The accepted rule is now used in action selection.
T31		**Reuse respects phase** At the next phase, the installed reversal hook is disabled by its phase metadata. D moves from (2,2) to (2,1), matching ordinary movement. Position and score continue from T30 without a reset.

Each board shows the pre-action state. Faded/dashed players mark starting positions, while solid players mark observed landings. Crosses mark failed landings followed by reset, not the respawn location. Dashed squares mark differing predicted cells. Solid gray blocks are obstacles. The $+/-$ symbols mark food values and short bars show remaining food life. Coordinates increase rightward and upward.

Snake Execution Case (Continued, 3/3)

Table 21 (continued): Hard, 4K, seed index 0.

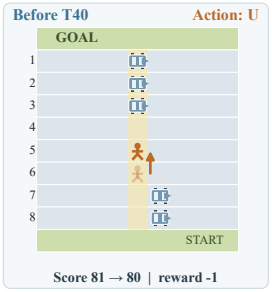
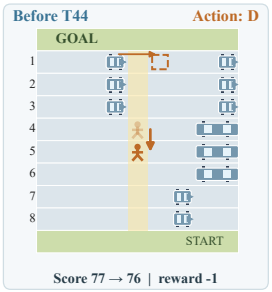
Turn	Logged State And Action	Observation And Policy Update
T61		**A new hazard appears** R reaches (4,1), a cell without an obstacle, but a rule-triggered reset returns the player to (3,3). The hazard-time food record includes positive food in column 4. This is new evidence, not an incorrect movement endpoint.
T66		**Install a hazard predicate** After another hazard reset at T65, the Critic requests a column-based safety check. T66 still resets before installation. Its post-transition candidate passes contract and replay checks and is scoped to the third phase.
T68		**Continue after revision** With the new safety predicate installed, U moves from (3,4) to (3,5) and collects positive food (+1), without a reset. The episode later finishes all 90 turns with score 19.

Each board shows the pre-action state. Faded/dashed players mark starting positions, while solid players mark observed landings. Crosses mark failed landings followed by reset, not the respawn location. Dashed squares mark differing predicted cells. Solid gray blocks are obstacles. The $+/-$ symbols mark food values and short bars show remaining food life. Coordinates increase rightward and upward.

E.2 Freeway: Learning A Reusable Traffic Predictor

Actual seed 1447, with 100 actions, five crossings, and terminal score 50. The excerpt spans T40–T70 and follows a periodic-motion diagnosis through validation, installation, and repeated use of the same source.

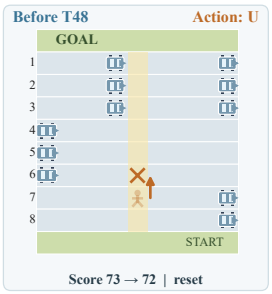
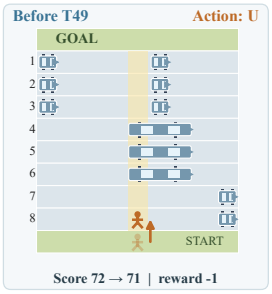
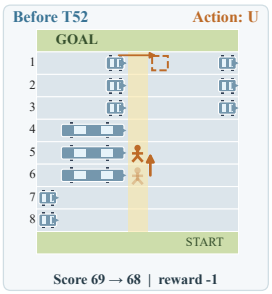
Table 22: **Evidence-supported prediction reuse in a logged Freeway episode.**

Turn	Logged State And Action	Observation And Policy Update
T40		**A continuing phase change** The second hidden-rule window begins without a world reset. U moves the player from lane 6 to lane 5. The case follows how later public vehicle transitions lead to a reusable prediction update.
T44		**Motion exceeds prediction** Vehicle 0 moves from column 3 to 5, while the ordinary one-pass prediction is column 4. The player moves from lane 4 to 5 and survives. Public records across lanes expose the additional motion pass.

Each board shows the pre-action state. Faded/dashed players mark starting positions, while solid players mark observed landings. Crosses mark failed landings followed by reset, not the respawn location. Vehicles show pre-action body occupancy and direction. The orange dashed box, when present, marks vehicle 0's observed next head, while other vehicles remain at their pre-action positions.

Freeway Execution Case (Continued, 2/3)

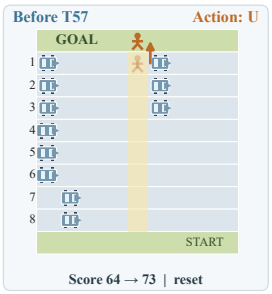
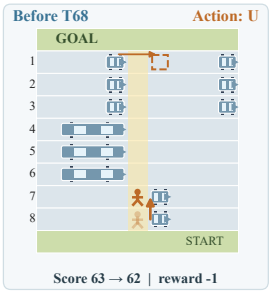
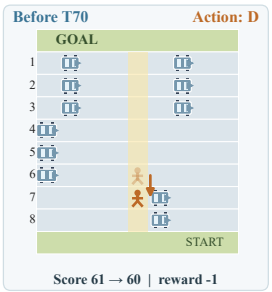
Table 22 (continued): Hard, 4K, seed index 0.

Turn	Logged State And Action	Observation And Policy Update
T48		**Repeated evidence triggers EDIT** Another synchronized motion deviation is recorded. U attempts entry into lane 6, collides, and resets to lane 9. After this transition, the Critic requests a periodic double-pass predictor using T44 and T48 evidence.
T49		**Validate and install** U first moves from lane 9 to 8 under the old source. The subsequent coding candidate passes contract and observable-history replay checks. It adds a phase-gated vehicle predictor, and the new code hash first appears at T50.
T52		**Prediction matches the next trigger** For vehicle 0, ordinary prediction is column 4, while the installed hook predicts 5 and the observed head is 5. The same source is used for the current U action, and the player moves from lane 6 to 5 without collision.

Each board shows the pre-action state. Faded/dashed players mark starting positions, while solid players mark observed landings. Crosses mark failed landings followed by reset, not the respawn location. Vehicles show pre-action body occupancy and direction. The orange dashed box, when present, marks vehicle 0's observed next head, while other vehicles remain at their pre-action positions.

Freeway Execution Case (Continued, 3/3)

Table 22 (continued): Hard, 4K, seed index 0.

Turn	Logged State And Action	Observation And Policy Update
T57		**Complete a crossing** U reaches the goal from lane 1. The environment awards crossing reward and returns the player to lane 9. The vehicle-prediction source remains unchanged. This is a logged outcome, not a counterfactual estimate of the edit's reward gain.
T68		**Reuse at a later trigger** The same code hash remains active. Vehicle 0 again moves from column 3 to 5: the installed hook predicts 5, versus the ordinary prediction of 4. The player safely advances from lane 8 to 7.
T70		**The next rule window begins** Traffic and player state continue into the third hidden-rule window. D moves from lane 6 to 7. The previous predictor's turn gate excludes this window. The complete 100-turn episode ends with five crossings and score 50.

Each board shows the pre-action state. Faded/dashed players mark starting positions, while solid players mark observed landings. Crosses mark failed landings followed by reset, not the respawn location. Vehicles show pre-action body occupancy and direction. The orange dashed box, when present, marks vehicle 0's observed next head, while other vehicles remain at their pre-action positions.