REINS: A Self-Evolving Agent Harness for Real-Time Trajectory Planning

Zhihong Cui[1*] **Hengyu Liu**[2*†] **Haoran Tang**[3] **Shijun Liu**[4]
Amir Taherkordi[1] **Tor Skeie**[1,5]
[1]Department of Informatics, University of Oslo, Norway
[2]Department of Computer Science, Aalborg University, Denmark
[3]Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China
[4]Shandong Key Laboratory of Foundational Software, School of Software, Shandong University, China
[5]Simula Research Laboratory, Norway

Abstract

Agent harnesses—runtime scaffolds that wrap an LLM's reasoning loop with tool dispatch, verification, and persistent memory—have become the standard substrate for deploying LLMs as autonomous agents, as demonstrated by Claude Code and Codex in software engineering. In trajectory planning, however, this paradigm has not taken hold: existing LLM-based trajectory planning methods either keep the LLM in the decision loop, exceeding the millisecond-level control budget, or offload it to offline rules or training-time supervision—leaving the runtime planner non-evolving. Intermediate hybrids still lack the verification-and-consolidation loop that makes harnesses reliable. We ask: *what runtime harness would let an LLM planner meet real-time constraints while self-evolving into a deployable one?* We propose **REINS**, a harness built on three LLM–planner couplings: LLM outputs are (i) **grounded** to a learned variable-level dynamics model, (ii) **verified** via calibrated predicates and forward rollout, and (iii) **consolidated** into an indexed rule memory that serves future scenes in $\mathcal{O}(\log n)$ without re-invoking the LLM. The harness closes a self-evolution loop: familiar scenes resolve by memory lookup; novel scenes invoke the LLM, whose verified outputs enrich the memory. Without modifying LLM weights, REINS attains 98%+ compliance, nearly 50% collision reduction, and 1.9–4.6 ms latency on simulated and real-world benchmarks at the tactical decision layer, with LLM invocation rate monotonically declining as memory matures. Code: `https://github.com/ZhihongCui/REINS`.

1 Introduction

Trajectory planning is the safety-critical core of autonomous driving: each decision must respect traffic rules, anticipate other agents, and complete within a millisecond-level control budget (e.g., 50 ms at 20 Hz) [Zhang et al., 2021a, Falcone et al., 2007, Lenz et al., 2016, Li et al., 2024b, Montgomery, 1983]. Recent work shows that large language models (LLMs) bring rich semantic priors over traffic norms, agent intent, and safety reasoning that classical planners lack [Manas et al., 2025, Fu et al., 2024, Mao et al., 2024, Sha et al., 2023, Cai et al., 2024, Li et al., 2024a], yet integrating their reasoning into the runtime loop is bottlenecked: each LLM query takes seconds—orders of magnitude over the control budget—and naive integration leaves the planner either too slow to act or unable to adapt as deployment proceeds.

This is precisely the kind of deployment problem—running a stochastic LLM inside a low-latency, safety-sensitive loop—that the **agent harness** paradigm has recently addressed in software engineering. An agent harness wraps the LLM with a thin runtime layer of tool dispatch, intermediate-result

*Equal contribution. †Corresponding author: `heli@cs.aau.dk`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

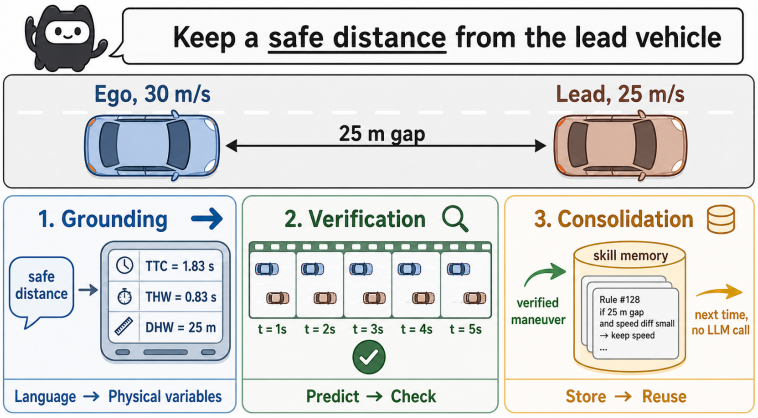


Figure 1: Three couplings an agent harness must enforce on the LLM planner, illustrated on a car-following scene with a 25-m gap. The LLM utterance "keep a safe distance" is (1) **grounded** to physical state variables (TTC, THW, DHW), (2) **verified** by forward rollout over a multi-step horizon, and (3) **consolidated** as an indexed rule keyed by these variables for instant reuse.

verification, persistent memory, and safety enforcement—as exemplified by systems like Claude Code, Codex, and Cursor-Agent. On Terminal-Bench [Merrill et al., 2026], harness design has emerged as a primary driver of ranking, with automatic harness optimization surpassing hand-engineered baselines [Lee et al., 2026]. Trajectory planning, however, is a sharp test of the paradigm: its millisecond-level control budget is much tighter than the seconds-to-minutes scale at which coding harnesses operate, and the safety stakes are higher. In this regime, the harness's generic functions specialize into three couplings between LLM and planner, illustrated in Figure 1: when an LLM suggests "keep a safe distance" in a car-following scene, the runtime must (i) *ground* the suggestion to physical state variables (TTC, THW, DHW), (ii) *verify* the resulting maneuver via forward rollout over a multi-step horizon, and (iii) *consolidate* the verified maneuver into an indexed rule memory.

Existing LLM-based trajectory planning methods span two extremes plus intermediate hybrids, none of which constitutes a harness in the above sense: *runtime invocation* methods keep the LLM in the decision loop but exceed the millisecond-level budget by orders of magnitude [Fu et al., 2024, Mao et al., 2024, Sha et al., 2023]; *offline extraction* methods discard the LLM at runtime and freeze the planner against scenarios the offline corpus did not anticipate [Li et al., 2024a, Tang et al., 2025]; and *hybrid* designs partially orchestrate the LLM at runtime but stop short of a full harness, lacking explicit verification of LLM commitments and consolidation of verified ones into reusable structure [Cui et al., 2026, Zhang et al., 2025]. Across all three, the LLM's reasoning is either too slow to act on or too detached from the loop to evolve with deployment.

This raises our central question: *what runtime harness would let an LLM planner satisfy real-time constraints while self-evolving into a deployable one?* Each of the three couplings is non-trivial under real-time control: grounding requires bridging natural-language commitments to the planner's threshold semantics [Wang et al., 2025, Zhai et al., 2025]; verification must catch stochastic, occasionally unsafe LLM outputs before they reach control [Manas et al., 2025, Zeng et al., 2025]; and consolidation must escape per-call LLM latency [Fu et al., 2024, Mao et al., 2024] via a memory pattern emerging in agentic AI [Wang et al., 2023, Zhao et al., 2024, Ma et al., 2024] but not yet transferred to real-time planning. The three are also jointly necessary: without grounding, verification has nothing to check; without verification, consolidation stores unverified content; without consolidation, every scene re-pays the LLM latency cost. When all three are addressed, a *self-evolution* loop emerges: familiar scenes resolve from the consolidated memory in $\mathcal{O}(\log n)$ time, novel scenes invoke the LLM, whose verified outputs enrich the memory, and reliance on the LLM decreases monotonically as memory matures, with no LLM weight update.

We propose **REINS**, a concrete realization of this harness pattern, instantiated for trajectory planning in driving, that addresses each coupling as follows. **Grounding.** We learn a *world dynamics model* over a factored variable space, fusing LLM-derived semantic priors with data-driven dependency discovery. Group-weighted ridge regression selects each variable's parents, and a per-variable Transformer predicts its evolution under the discovered graph. Every LLM output can then be projected onto this variable space and rolled out under the learned dynamics. **Verification.** We

compile each LLM-proposed maneuver into *calibrated predicates* with thresholds derived from data (z-score, quantile, or absolute). Predicates are evaluated against the rollout produced by the dynamics model, and only commitments whose predicted trajectory satisfies all guards are released to control. Hallucinated or unsafe commitments are filtered before any maneuver is issued. **Consolidation.** We persist verified commitments in an *indexed rule memory* backed by a graph database, keyed by the grounding variables that triggered them. At runtime, parameterized graph queries retrieve scene-relevant rules in $\mathcal{O}(\log n)$ time without any LLM call. Recurring scenes are thus served by memory lookup, and the LLM is invoked only on a miss. Together, the three close the self-evolution loop: at each scene the planner first consults the memory; on a miss the LLM is invoked, its output grounded against the dynamics model, verified by rollout, and—if it passes—committed to memory for the next encounter. We provide an empirical demonstration that, under such a harness, an LLM planner monotonically reduces its own LLM-call rate while preserving safety and compliance. Our contributions can be summarized as:

- We extend the agent-harness paradigm to real-time trajectory planning, identifying *three couplings*—grounding, verification, consolidation—any such harness must enforce, and instantiate it for driving; only the variable schema and the action space are domain-specific (Appendix D.1).
- We design **REINS**, a harness that instantiates the three couplings via a learned world dynamics model, calibrated predicates with rollout verification, and an indexed rule memory.
- We provide the first empirical evidence of self-evolution in real-time LLM trajectory planning: LLM invocation rate decreases monotonically over deployment with no LLM weight update.
- We evaluate REINS on simulated and real-world driving benchmarks at the tactical decision layer, achieving 98%+ compliance, nearly 50% collision reduction, and 1.9–4.6 ms planning latency—well within the real-time control budget.

2 Related Work

LLM-Based Trajectory Planning. Recent LLM-based trajectory planners fall along three lines, each realizing some couplings while leaving others open. *Runtime invocation* methods keep the LLM in the decision loop. DiLu [Fu et al., 2024] pairs LLM reasoning with an experience-replay memory; Agent-Driver [Mao et al., 2024] treats the LLM as an orchestrator over perception and planning tools; LanguageMPC [Sha et al., 2023] feeds chain-of-thought outputs into MPC. These systems gain semantic flexibility, but each LLM query takes seconds, and any output that reaches the planner is implicitly trusted, lacking the calibrated checks proposed for stochastic LLM outputs in safety-critical settings [Manas et al., 2025, Zeng et al., 2025].

Offline extraction methods sit at the opposite end. LLaDA [Li et al., 2024a] extracts location-specific rules from driver handbooks, DriReg [Cai et al., 2024] retrieves regulatory text via RAG, and Plan-R1 [Tang et al., 2025] distills LLM-shaped rewards into a reinforcement-learning policy. Removing the LLM at runtime buys real-time inference, but the planner becomes static, unable to consolidate new commitments against learned driving dynamics [Wang et al., 2025, Zhai et al., 2025] or to adapt to scenarios the offline corpus did not anticipate.

Hybrid designs partially orchestrate the LLM at runtime. C-TRAIL [Cui et al., 2026] runs a closed-loop Recall–Plan–Update cycle with trust-weighted MCTS injection; LeAD [Zhang et al., 2025] pairs an end-to-end planner with a redundant LLM reasoning track. These improve over either extreme but lack explicit verification of LLM commitments and consolidation of verified ones into reusable structure; none grounds, verifies, and consolidates LLM outputs jointly under a real-time harness.

Agent Harnesses for LLM Deployment. Outside trajectory planning, the deployment of LLMs into real-world tasks has converged on the *agent harness* pattern: a thin runtime layer that wraps the LLM with tool dispatch, intermediate-result verification, persistent memory, and safety enforcement. In autonomous coding, systems such as Claude Code and Codex turn general-purpose LLMs into reliable software engineers, and harness design has become a primary driver of ranking on Terminal-Bench [Merrill et al., 2026, Lee et al., 2026]. In open-ended decision-making, Voyager [Wang et al., 2023] accumulates a skill library of executable code through curiosity-driven exploration; Eureka [Ma et al., 2024] iteratively refines reward functions by evolutionary search over LLM proposals; ExpeL [Zhao et al., 2024] distills task-specific experience into a retrievable knowledge base. These harnesses share three structural elements with REINS—grounding into a domain

primitive space, verification of LLM outputs, and persistent reuse of verified ones—but none operate under millisecond real-time budgets nor expose their LLMs to the safety-critical constraints.

3 Preliminaries

We formalize the harness's primitives: a factored variable space (Section 3.1); a world dynamics model for *grounding* (Section 3.2); an executable rule representation for *verification* and *consolidation* (Section 3.3); and the problem definition (Section 3.4). Full variable lists, template specifications, and the macro-group dependency enumeration appear in Appendix A.

3.1 Driving Environment

Definition 3.1 (Factored Variable Structure)**.** A *factored variable structure* is $(\mathcal{X}, \mathcal{H}, \mathcal{E}_{\mathcal{G}}, \mathcal{E}_{\mathcal{H}})$: $\mathcal{X} = \{X_1, \ldots, X_d\}$ are observable variables (e.g., TTC, THW, DHW, ego_speed, lead_speed, road_friction); $\mathcal{H} = \{\textsc{Env}, \textsc{Traffic}, \textsc{Ego}, \textsc{Action}, \textsc{Outcome}\}$ groups them by semantic role [Chen et al., 2025]; $\mathcal{E}_{\mathcal{G}} \subseteq \mathcal{X} \times \mathcal{H}$ assigns each variable to a group; and $\mathcal{E}_{\mathcal{H}} \subseteq \mathcal{H} \times \mathcal{H}$ specifies permissible group-level dependencies (Eq. 10 in Appendix A.1).

Definition 3.2 (State and Action Spaces)**.** The *state* $s_t \in \mathcal{S} = \prod_i \mathrm{dom}(X_i)$ is a complete variable assignment. The *action space* $\mathcal{A} = \{\texttt{LANE_LEFT}, \texttt{LANE_RIGHT}, \texttt{FASTER}, \texttt{SLOWER}, \texttt{IDLE}\}$ is a finite set of discrete meta-actions for the ego vehicle.

The meta-actions are the interface of the tactical (behavior-planning) layer in the standard stack decomposition [Paden et al., 2016]; continuous low-level controllers execute them, and all safety reasoning below (TTC, headway, accelerations) remains continuous.

3.2 World Dynamics Model

Definition 3.3 (Dependency Graph and Dynamics)**.** A *dependency graph* is a weighted DAG $\mathcal{G} = (\mathcal{X}, \mathcal{E}, w)$ with $w : \mathcal{E} \to \mathbb{R}$, where edges $(X_i, X_j) \in \mathcal{E}$ are admissible only if $(h(X_i), h(X_j)) \in \mathcal{E}_{\mathcal{H}}$ [Giamattei et al., 2024]. A *dynamics function set* $\mathcal{F} = \{f_i\}_{i=1}^{d}$ predicts each variable from its parents over a temporal lag L [Peters et al., 2017]:

$$\hat{X}_i^{t+1} = f_i\big(\mathrm{Pa}(X_i)^{t-L+1:t}\big), \quad \mathrm{Pa}(X_i) = \{X_j \mid (X_j, X_i) \in \mathcal{E}\}. \tag{1}$$

An H-step rollout from s_t under $(a_t, \ldots, a_{t+H-1})$ yields $\hat{\xi} = (\hat{s}_{t+1}, \ldots, \hat{s}_{t+H})$.

3.3 Rule Representation

Definition 3.4 (Predicate)**.** A *predicate* is a Boolean function $p(s) = \mathbb{I}[s[X_i] \prec \tau]$ with comparison operator $\prec \in \{<, >, \leq, \geq\}$ and threshold $\tau \in \mathbb{R}$.

TTC and THW guards are the surrogate safety measures of traffic engineering [Gettman and Head, 2003]; their calibrated thresholds act as RSS-style safe-distance bounds [Shalev-Shwartz et al., 2017].

Definition 3.5 (Rule Template)**.** A *rule template* T specifies how predicates compose [Manas et al., 2024]. We use seven templates: `IF-THEN`, `AND-THEN`, `THRESHOLD`, `PRIORITY`, `MULTI-ACTION`, `WEIGHTED-ACTION`, and `TEMPORAL-CHAIN` (specifications and examples in Appendix A.2).

Definition 3.6 (Executable Rule)**.** An *executable rule* is a tuple $r = (T, \mathcal{P}_r, a_r, c_r)$ with template T, guard predicates $\mathcal{P}_r$, recommended action $a_r \in \mathcal{A}$, and confidence $c_r \in [0, 1]$. Activation requires all guards to hold:

$$\mathrm{Exec}(r, s_t) = \bigwedge_{p \in \mathcal{P}_r} p(s_t). \tag{2}$$

3.4 Problem Definition

Definition 3.7 (Self-Evolving Agent Harness for Trajectory Planning)**.** Let $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}$ be a driving dataset and $\mathcal{M}$ a pre-trained LLM. The harness problem is to construct $\langle \hat{\mathcal{G}}, \hat{\mathcal{F}}, \mathcal{R}, \pi \rangle$ where $\hat{\mathcal{G}}, \hat{\mathcal{F}}$ are learned from $\mathcal{D}$ with priors from $\mathcal{M}$ (Def. 3.3), $\mathcal{R}$ is an indexed rule memory compiled offline from $\mathcal{M}$ (Def. 3.6), and $\pi : \mathcal{S} \to \mathcal{A}$ is a policy that, at each s_t, retrieves $\mathcal{R}_t = \mathrm{Retrieve}(s_t, \mathcal{R}) \subseteq \mathcal{R}$ and proceeds as follows:

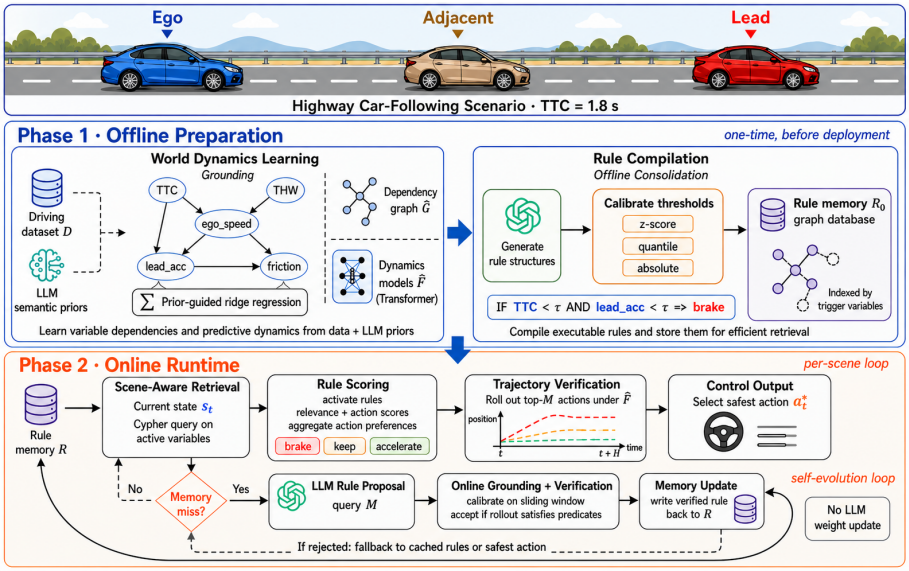


Figure 2: The REINS framework. *Phase 1 (offline)*: learn dynamics $\hat{\mathcal{G}}, \hat{\mathcal{F}}$ and compile rules into initial memory $\mathcal{R}_0$. *Phase 2 (online)*: per-scene retrieval and rollout-based verification produce control; on a miss, the LLM proposes a rule that is grounded, verified, and written back to $\mathcal{R}$, closing the self-evolution loop without any LLM weight update.

- (*hit*) if $\mathcal{R}_t \neq \emptyset$, π scores actions over $\mathcal{R}_t$ and verifies them by rollout under $\hat{\mathcal{F}}$ within a real-time budget;
- (*miss*) otherwise, $\mathcal{M}$ proposes a candidate commitment, which is grounded via $(\hat{\mathcal{G}}, \hat{\mathcal{F}})$ and verified by rollout; if all predicates pass, the commitment is compiled into a rule r and $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$.

The parameters of $\mathcal{M}$ remain unchanged, and $\mathcal{R}$ is monotone non-decreasing; under stationarity of the scene distribution, the per-scene LLM invocation rate $\Pr[\mathcal{R}_t = \emptyset]$ is non-increasing.

4 Methodology

REINS realizes the harness as a two-phase system (Figure 2). *Offline Preparation* (Section 4.1) prepares the world dynamics $\hat{\mathcal{G}}, \hat{\mathcal{F}}$ for *grounding* and an initial rule memory $\mathcal{R}_0$ for offline *consolidation*. *Online Runtime* (Section 4.2) executes a per-scene loop: rule-guided planning performs retrieval, scoring, and *verification* over $\mathcal{R}$, and online rule acquisition handles memory misses by querying $\mathcal{M}$ and writing verified rules back to $\mathcal{R}$, closing the *self-evolution loop*.

4.1 Offline Preparation

4.1.1 World Dynamics Learning

This module realizes *grounding*: we jointly learn dependency graph $\hat{\mathcal{G}}$ and dynamics functions $\hat{\mathcal{F}}$ by combining LLM semantic priors with data-driven structure discovery. The full derivation and the dynamics loss are in Appendix A.4.1.

Dependency Graph. The LLM provides edge confidence $\pi_{ij} \in [0, 1]$ for each candidate edge $X_j \to X_i$ (Appendix A.5); combined with macro-group constraints (Eq. 10), the unified prior is $\tilde{\pi}_{ij} = \alpha\, \pi_{ij} + (1 - \alpha)\, \mathbb{I}\{(h(X_j), h(X_i)) \in \mathcal{E}_{\mathcal{H}}\}$. For each X_i, prior-guided ridge regression [Hoerl and Kennard, 1970] predicts its value from lagged neighbors with adaptive penalty $\lambda_{ij} = \lambda_0(1-\tilde{\pi}_{ij})^{\rho}$, so high-prior edges are weakly regularized:

$$\hat{\mathbf{w}}_i = \arg\min_{\mathbf{w}} \sum_{t=L}^{N} \left(X_i^t - \textstyle\sum_{j \neq i, \ell} w_{j\ell} X_j^{t-\ell} \right)^2 + \textstyle\sum_{j \neq i} \lambda_{ij} \|\mathbf{w}_j\|_2^2. \tag{3}$$

$\mathrm{Pa}(X_i)$ retains the top-K variables by $\sum_\ell |\hat{w}_{j\ell}|$, yielding $\hat{\mathcal{E}}$ and weights $\hat{w}(X_j, X_i) = \sum_\ell |\hat{w}_{j\ell}|$.

Dynamics Functions. Given $\hat{\mathcal{G}}$, each f_i is a multi-head Transformer [Vaswani et al., 2017] trained to predict X_i^{t+1} from $\mathrm{Pa}(X_i)^{t-L+1:t}$ via mean squared error (loss form in Eq. 14). Training curves and hyperparameters are in Appendix C.1.

4.1.2 Rule Compilation

This module realizes offline *consolidation*: compiling LLM knowledge into executable rules. Full equations and per-step prose are in Appendix A.4.2.

Rule Structure Generation. For each edge $(X_j, X_i) \in \hat{\mathcal{E}}$, the LLM generates a rule $r = (T, \mathcal{P}_r, a_r, c_r)$ (Eq. 15) with template T (Def. 3.5), guard predicates $\mathcal{P}_r$ with symbolic thresholds, recommended action $a_r \in \mathcal{A}$, and confidence c_r derived from edge weight $\hat{w}(X_j, X_i)$ (prompts in Appendix A.5).

Threshold Calibration. Each predicate's threshold τ_i is calibrated from $\mathcal{D}$ in three forms [Luo et al., 2024]—z-score $\mu_i + \zeta\sigma_i$, quantile $Q_i(\zeta)$, or absolute constant τ_0—and the comparison operator $\prec$ is selected to maximize separability of safety outcome $y \in \{0, 1\}$:

$$\prec^* = \arg\max_{\prec \in \{<, >\}} \big|\mathbb{E}[y \mid X_i \prec \tau_i] - \mathbb{E}[y \mid X_i \not\prec \tau_i]\big|. \tag{4}$$

Rule activation then follows Eq. 2.

Graph Database Storage. Compiled rules are stored in a Neo4j graph database [Webber, 2012] with rule nodes connected to variable nodes via `TRIGGERS` (input) and `AFFECTS` (output) edges, enabling $O(\log |\mathcal{R}|)$ retrieval. The full schema (Figure 3) is in Appendix A.4.2; database statistics are in Appendix C.7.1.

4.2 Online Runtime

4.2.1 Rule-Guided Planning

This module realizes the runtime planning loop of Def. 3.7—retrieval, scoring, and *verification* (details in Appendix A.4.3).

Scene-Aware Retrieval. Given s_t, active variables $\{X_i\}$ trigger parameterized Cypher queries [Francis et al., 2018] generated by the LLM (read-only, schema-constrained; Appendix A.5). The retrieval map $G : (\{X_i\}, K) \mapsto \mathcal{R}_t \subseteq \mathcal{R}$ (Eq. 17) returns up to K rules ranked by confidence, restricted to triggers whose dependency direction respects the macro-group constraints.

Action Scoring. For each $r \in \mathcal{R}_t$ activated via Eq. 2, a Transformer-based scorer [Liang et al., 2022] produces relevance $S_{\text{rel}}(r, s_t) \in [0, 1]$ and action preference $S_{\text{act}}(r, s_t, a)$, yielding combined score $f(r, a \mid s_t) = S_{\text{rel}} \cdot S_{\text{act}} \cdot c_r$ (Eqs. 18–19; training loss in Eq. 20). Multiple rules recommending action a, denoted $\mathcal{R}_t(a) = \{r \in \mathcal{R}_t \mid a_r = a\}$, are aggregated by noisy-OR:

$$P(a \mid s_t) \;\propto\; 1 - \prod_{r \in \mathcal{R}_t(a)} \big(1 - f(r, a \mid s_t)\big). \tag{5}$$

The full normalized form and argmax decoder are in Appendix A.4.3.

Trajectory Verification. The learned dynamics $\hat{\mathcal{F}}$ verifies safety via forward rollout. For the top-M actions ranked by $P(a \mid s_t)$, we predict H-step trajectories $\hat{\xi}(a) = (\hat{s}_{t+1}, \ldots, \hat{s}_{t+H})$ via $\hat{s}_{t+h+1} = \Phi(\hat{s}_{t+h}, a; \hat{\mathcal{G}}, \hat{\mathcal{F}})$ (Eq. 22), and select $a_t^\star = \arg\max_{a \in \mathcal{A}_M} J(\hat{\xi}(a))$, where J penalizes collisions, rule violations, and control effort (Appendix B.2); the first control is executed in receding-horizon [Zhang et al., 2021b].

4.2.2 Online Rule Acquisition

This module realizes Def. 3.7's miss-handling path through four steps (full equations and the monotonicity argument in Appendix A.4.4).

Miss Detection. We operationalize the miss criterion ($\mathcal{R}_t = \emptyset$) with an additional confidence guard against weakly supported matches:

$$\mathrm{Miss}(s_t) \;=\; \mathbb{I}[\mathcal{R}_t = \emptyset] \;\vee\; \mathbb{I}\left[\max_{r \in \mathcal{R}_t,\, a \in \mathcal{A}} f(r, a \mid s_t) < \eta\right], \tag{6}$$

with tuned threshold $\eta \in (0, 1)$.

Rule Proposal and Online Verification. On a miss, $\mathcal{M}$ proposes $\tilde{r} = (\tilde{T}, \tilde{\mathcal{P}}, \tilde{a}, \tilde{c})$ (Eq. 23) conforming to Defs. 3.4–3.5 (Appendix A.5). Symbolic thresholds in $\tilde{\mathcal{P}}$ are calibrated on a sliding window $\mathcal{D}_t^W = \{s_{t-W+1}, \ldots, s_t\}$ so they track the current distribution. The harness then rolls out $\tilde{a}$ from s_t under $\hat{\mathcal{F}}$ and accepts $\tilde{r}$ iff every predicate holds along the rollout:

$$\mathrm{Accept}(\tilde{r}, s_t) \;=\; \bigwedge_{p \in \tilde{\mathcal{P}}} \bigwedge_{h=1}^{H} p(\hat{s}_{t+h}). \tag{7}$$

On rejection the planner falls back to $\arg\max_a P(a \mid s_t)$ over $\mathcal{R}_t$ if $\mathcal{R}_t \neq \emptyset$; otherwise the harness raises a human-takeover alert and executes the safest action under $\hat{\mathcal{F}}$ by maximizing minimum lookahead TTC (Eq. 24 in Appendix A.4.4).

Memory Update. Accepted $\tilde{r}$ is keyed by content hash $\kappa(\tilde{r}) := \kappa(\tilde{T}, \tilde{\mathcal{P}}, \tilde{a})$ and inserted into Neo4j; on collision, confidences are averaged instead of duplicating the node:

$$\mathcal{R} \;\longleftarrow\; \begin{cases} \mathcal{R} \cup \{\tilde{r}\} & \kappa(\tilde{r}) \notin \mathrm{Keys}(\mathcal{R}), \\ \mathcal{R} \text{ with } c_{r'} \leftarrow \frac{1}{2}(c_{r'} + \tilde{c}) & \exists\, r' \in \mathcal{R} : \kappa(r') = \kappa(\tilde{r}). \end{cases} \tag{8}$$

The collision-branch rationale (averaging vs. duplication, exclusion of $\tilde{c}$ from κ) is in Appendix A.4.4.

Self-Evolution Properties. Eq. 8 structurally enforces $|\mathcal{R}_{t+1}| \geq |\mathcal{R}_t|$ (memory monotonicity) and $\theta_{\mathcal{M}} \equiv \theta_{\mathcal{M}}^{(0)}$ (no LLM update). Under stationarity of the scene distribution, the per-state miss rate is non-increasing as $\mathcal{R}$ accumulates coverage:

$$|\mathcal{R}_{t+1}| \geq |\mathcal{R}_t|, \quad \Pr[\mathrm{Miss}(s_{t+1})] \leq \Pr[\mathrm{Miss}(s_t)], \quad \theta_{\mathcal{M}} \equiv \theta_{\mathcal{M}}^{(0)}, \tag{9}$$

so the per-scene LLM invocation rate decays monotonically with $\Pr[\mathrm{Miss}]$ (full argument in Appendix A.4.4; validated in Section 5.4).

5 Experiment

The main body reports overall comparison (Section 5.2), ablation (Section 5.3), self-evolution dynamics on Highway (Section 5.4), and a summary of the robustness, transfer, and scalability results. Appendices C and D report trajectory accuracy, sensitivity, extended self-evolution traces, a case study, the rule database, and the fairness and robustness studies.

5.1 Experiment Setting

Datasets and Protocols. We evaluate REINS on four `highway-env` [Leurent, 2018] scenarios—Highway, Merge, Roundabout, Intersection (100 steps at 5 Hz, 15 observable vehicles per episode)—and two drone-captured real-world datasets [Krajewski et al., 2018, 2020]: highD (German highway, 4,237 vehicles, 578 lane changes) and rounD (German roundabouts, 2,156 vehicles, 423 yield/merge events). Surrounding vehicles in simulation are *reactive* (IDM car-following [Treiber et al., 2000] with MOBIL lane changes [Kesting et al., 2007]), so they brake, yield, and change lanes in response to the ego. The real-world datasets run in *closed-loop replay*: one recorded vehicle is controlled by the planner while the others follow their recorded trajectories, so these rows reflect human behavior rather than a simulator's; a collision is registered when bounding boxes overlap or the ego leaves the drivable lane. A reactive-replay variant following the nuPlan protocol [Caesar et al., 2021] is in Appendix D.6. All methods observe clean factored variables; perception, tracking, and map errors lie outside the evaluated layer (Section 6).

Evaluation Metrics. Compl. (Rule Compliance) is the fraction of steps at which all active rules hold, measured for each method against its own executable rule set; an audit of all methods against REINS's compiled set as a common reference leaves the ordering unchanged (Appendix D.2). **RE** (Risk-aware Efficiency) is task reward minus a TTC-based risk penalty. **CR** (Collision Rate) is the fraction of *episodes* with at least one contact among the 15 vehicles over 100 steps, an episode-level rate in deliberately stressed traffic. **PE** (Planning Efficiency) is the mean wall-clock ms of the synchronous action-selection computation per step for every method; for REINS it excludes the asynchronous

Dataset	Scenario	Metric	DriveSpeak	LLaDA	MPC	MCTS	PDDL	DriReg	SymAgent	Plan-R1	REINS
Simulation	Hig.	Compl. (↑)	-	-	-	-	0.9123	0.9452	0.9561	0.9717	**0.9956**
		RE (↑)	-6.0360	-5.8537	-5.5613	-5.2311	-4.7868	-4.1282	-3.8662	-3.5623	**0.5228**
		CR (↓,%)	38.28	35.51	14.27	16.76	25.93	21.76	16.52	12.83	**5.67**
		PE (↓, ms)	5.8921	5.2995	2.4742	**2.3591**	4.9521	4.3127	3.9621	3.7391	2.4998
	Mer.	Compl. (↑)	-	-	-	-	0.9011	0.9382	0.9572	0.9641	**0.9924**
		RE (↑)	-5.5730	-5.2621	-5.3891	-5.1890	-4.7129	-4.1067	-3.5622	-3.1023	**-0.0400**
		CR (↓,%)	35.62	22.84	27.91	23.64	21.52	20.56	16.83	14.78	**6.86**
		PE (↓, ms)	5.2120	4.3615	1.7354	**1.6526**	3.8549	2.8122	2.5697	2.2418	1.8928
	Rou.	Compl. (↑)	-	-	-	-	0.8838	0.9129	0.9411	0.9568	**0.9925**
		RE (↑)	-7.6873	-7.1069	-6.7722	-6.5113	-6.4895	-6.3457	-6.2313	-6.0256	**-0.5444**
		CR (↓,%)	39.19	36.63	24.27	21.62	23.84	25.13	24.74	18.57	**9.01**
		PE (↓, ms)	8.2673	7.6541	4.2134	4.0267	5.3793	5.1682	5.0713	4.8621	**3.6243**
	Int.	Compl. (↑)	-	-	-	-	0.8789	0.9153	0.9274	0.9553	**0.9838**
		RE (↑)	-7.9740	-7.5800	-7.1345	-7.0273	-6.9792	-6.7632	-6.3314	-6.0322	**-0.5747**
		CR (↓,%)	44.27	42.22	25.38	23.78	32.12	29.89	25.34	20.37	**9.08**
		PE (↓, ms)	10.9324	9.3549	4.9653	4.7519	6.7682	6.2793	5.9932	5.5733	**4.5807**
Real-world	highD	Compl. (↑)	-	-	-	-	0.9078	0.9289	0.9423	0.9612	**0.9867**
		RE (↑)	-5.8234	-5.5127	-5.2891	-5.0124	-4.6523	-4.2178	-3.7856	-3.4521	**0.3847**
		CR (↓,%)	35.21	32.87	18.34	19.56	23.12	20.89	17.23	12.78	**6.12**
		PE (↓, ms)	6.1234	5.5672	2.6123	**2.4891**	5.1234	4.5623	4.1234	3.8912	2.6234
	rounD	Compl. (↑)	-	-	-	-	0.8956	0.9187	0.9378	0.9534	**0.9856**
		RE (↑)	-7.4521	-6.9834	-6.5123	-6.2478	-6.1234	-5.8912	-5.6234	-5.3178	**-0.4123**
		CR (↓,%)	41.23	38.56	25.34	23.12	26.78	24.23	20.89	18.47	**9.45**
		PE (↓, ms)	8.5621	7.8923	4.3521	4.1234	5.6234	5.2178	4.8912	4.5623	**3.8912**

Table 1: Performance comparison across simulated scenarios and real-world datasets. Methods without explicit rule representations (MPC, MCTS, DriveSpeak, LLaDA) cannot report Compliance against their own rules as they lack executable rule constraints; their compliance against REINS's compiled set as a common reference is given in Table 15.

rule-acquisition pipeline, profiled separately in Appendix D.8. TTC and THW, which underlie the guards and the RE penalty, are the surrogate safety measures of traffic engineering [Gettman and Head, 2003], and Compl. and CR echo the rule-conformance and collision outcomes reported in scenario-based safety assessment [Riedmaier et al., 2020]. Results are averaged over five seeds; formal definitions in Appendix B.2.

Baselines. We compare with eight baselines covering the regimes from Section 2 plus classical anchors: (i) *classical* (MPC [Falcone et al., 2007], MCTS [Lenz et al., 2016]); (ii) *runtime invocation* (DriveSpeak [Cui et al., 2024]); (iii) *offline extraction* (LLaDA [Li et al., 2024a], DriReg [Cai et al., 2024], Plan-R1 [Tang et al., 2025]); (iv) *symbolic/hybrid* (PDDL [Guan et al., 2023, Liu et al., 2023], SymAgent [Liu et al., 2025]). All methods share the observation space, action space, episodes, and LLM backbone (GPT-4o, temperature 0); descriptions, the shared and unshared conditions, and a matched re-run of PDDL and DriReg over REINS's vocabulary and calibration pipeline are in Appendices B.3 and D.2.

5.2 Performance Evaluation

Performance evaluation results are reported in Table 1, with trajectory accuracy analysis in Appendix C.2.

Effectiveness Analysis. (i) REINS beats all baselines, improving over Plan-R1 by 2.4%–3.6% Compl., 3.06–5.48 RE, and 49%–56% CR reduction—validating that grounding, verification, and consolidation act jointly. (ii) All methods incur higher CR on complex scenarios (Roundabout, Intersection) than structured ones (Highway, Merge); REINS degrades least, owing to context-aware retrieval. (iii) On highD and rounD, REINS maintains Compl. above 0.98 with 6.6–9.0-point CR reduction over Plan-R1, transferring to naturalistic driving without domain-specific tuning. (iv) Executable rule methods (PDDL, DriReg, SymAgent, Plan-R1) beat language-guided ones (DriveSpeak, LLaDA, with 22%–44% CR for ungrounded rules), yet their static thresholds and

	Metric	w/o World	w/o LLM Prior	w/o Rule Generation	w/o Retrieval	Full
Hig.	Compl. (↑)	0.9521	0.9628	0.9312	0.9845	**0.9956**
	RE (↑)	-1.9234	-1.5390	-2.4156	-0.8921	**0.5228**
	CR (↓,%)	8.47	8.26	11.23	8.12	**5.67**
	PE (↓, ms)	2.9532	2.6831	2.5124	3.8921	**2.4998**
Mer.	Compl. (↑)	0.9687	0.9738	0.9198	0.9812	**0.9924**
	RE (↑)	-1.5621	-1.2619	-2.1893	-0.7234	**-0.0400**
	CR (↓,%)	9.23	9.08	12.87	8.89	**6.86**
	PE (↓, ms)	2.6234	2.1832	1.9021	2.9834	**1.8928**
Rou.	Compl. (↑)	0.9534	0.9611	0.9087	0.9756	**0.9925**
	RE (↑)	-2.3456	-1.1783	-3.1234	-1.2134	**-0.5444**
	CR (↓,%)	12.34	11.84	15.89	11.23	**9.01**
	PE (↓, ms)	4.1245	3.8011	3.6521	5.2134	**3.6243**
Int.	Compl. (↑)	0.9456	0.9582	0.8923	0.9689	**0.9838**
	RE (↑)	-2.5678	-1.9812	-3.4521	-1.3456	**-0.5747**
	CR (↓,%)	13.45	12.23	17.56	11.12	**9.08**
	PE (↓, ms)	4.9521	4.7214	4.6234	6.1234	**4.5807**

Table 2: Ablation study across four driving scenarios.

uniform activation still trail REINS's data-driven calibration and variable-indexed retrieval; a matched re-run over REINS's vocabulary and calibration narrows PDDL's and DriReg's CR by 3–5 points but leaves the gap intact (Appendix D.2). (v) Classical planners (MPC, MCTS) are fast but unsafe (CR 14%–28%); Plan-R1 is the best baseline but plateaus under uniform rule application.

Efficiency Analysis. REINS completes planning in 1.9–4.6 ms per step, up to 58% faster than runtime LLM baselines (DriveSpeak, LLaDA) and 15–33% faster than Plan-R1. Two factors contribute: (i) variable-indexed retrieval activates only 2–3 rules per step via $O(\log |\mathcal{R}|)$ queries, and (ii) learned dynamics enable closed-form rollout without iterative LLM calls. PDDL makes no runtime LLM call either, yet re-grounds every precondition at each step (3.71 of its 4.95 ms on Highway; Appendix D.3), exactly the cost that indexed consolidation removes.

5.3 Ablation Study

We conduct ablation experiments on all four simulated scenarios to evaluate component contributions. Table 2 reports results for four variants: (i) **w/o World Model**: replacing learned dynamics with constant-velocity propagation; (ii) **w/o LLM Prior**: using purely data-driven structure learning without semantic guidance; (iii) **w/o Rule Generation**: substituting LLM-generated rules with hand-crafted ones; (iv) **w/o Retrieval**: applying all rules uniformly without context-aware selection.

Removing the world dynamics model causes the largest safety degradation (CR +49%/+48% on Highway/Intersection), confirming multi-step prediction is essential. Without LLM priors, CR rises 31–46%, showing that semantic guidance regularizes dependency discovery beyond purely data-driven methods. Hand-crafted rules yield the largest compliance drop (0.9956→0.9312 on Highway) and nearly double CR, exposing the limited coverage of manual engineering; even so, they remain safer inside the harness than every matched no-harness baseline (Appendix D.2). Disabling context-aware retrieval raises both CR (22–43%) and PE (34–58%) via conflicting constraints. The four roles are complementary: dynamics enables predictive verification, LLM priors regularize structure, rule generation ensures coverage, and selective retrieval prevents conflicts.

5.4 Self-Evolution Analysis

We deploy REINS for 1000 episodes on Highway from the offline-compiled memory $\mathcal{R}_0$ ($|\mathcal{R}_0|$=88), reporting four checkpoint snapshots alongside a *Frozen* $\mathcal{R}_0$ control that runs the same window with acquisition disabled (Table 3; other scenarios in Appendix C.5). Across deployment, $|\mathcal{R}|$ grows from 88 to 145 (saturating after ep. 500; Eq. 9), the LLM call rate decays from 70.0% to 12.81% (a 5.5× reduction with no weight update), Compl. rises from 0.9885 to 0.9956, and CR drops from 7.05% to 5.67%. Against the Frozen $\mathcal{R}_0$ control over the same horizon (0.9800 Compl., 9.63% CR), online acquisition adds $+0.0156$ Compl. and removes 3.96 CR points—the headline self-evolution gain.

Checkpoint	$\lvert\mathcal{R}\rvert$	LLM Call Rate (%)	Compl. (↑)	CR (%) (↓)
ep. 250	113	70.0	0.9885	7.05
ep. 500	127	48.44	0.9920	6.11
ep. 750	138	25.0	0.9942	5.83
ep. 1000 (final)	**145**	**12.81**	**0.9956**	**5.67**
Frozen $\mathcal{R}_0$ (1000-ep eval)	88	—	0.9800	9.63

Table 3: Self-evolution dynamics on Highway over 1000 episodes; LLM call rate is the miss rate (Eq. 6) in the 250-episode window ending at each checkpoint.

Mapped to the three couplings: monotone $|\mathcal{R}|$ under κ-hash deduplication validates consolidation, the joint Compl. ↑/CR ↓ trajectory validates rollout-based verification, and the 5.5× call-rate drop validates the grounding–consolidation loop, as variable-keyed memory hits replace LLM calls on similar scenes.

Of the roughly 39,000 proposals implied by these call rates, 2.1% failed schema validation, 18.0% were rejected by rollout verification, and the accepted remainder deduplicated to 57 unique rules, 54 of which an audit judged consistent with domain guidelines (Appendix D.5).

Robustness, Transfer, and Scalability. Appendix D stresses these results. Rules regenerated with Claude Sonnet 4.5 or Qwen2.5-72B-Instruct keep Highway CR at 5.89% and 6.24% (vs. 5.67%), and prompt paraphrases span 5.58–5.91%. Under 2.5–10% observation noise and 5% frame drops, REINS CR rises to at most 8.14% while Plan-R1 rises to 19.87%. A memory compiled only from highD/rounD starts at an 86.2% call rate on highway-env and recovers to within 0.5–1.1 CR points of the native memory within 1000 episodes, whereas a frozen transferred memory does not. Inflating the memory to 100,000 rules keeps retrieval under 0.8 ms, and the 1.59 s miss cycle, dominated by the LLM call, runs off the control path behind a 2.6 ms verified fallback.

6 Limitations

Abstraction gap and action interface. The evaluation sits at the tactical decision layer: perception is assumed as clean factored variables, highD and rounD run in replay (reactive-replay variant in Appendix D.6), and occlusion, tracking, and map errors are absent; the five meta-actions are the standard interface of this layer and of all eight baselines, and continuous maneuvers would change only the scorer's output head. The claims are harness results at this layer, not a deployable AV stack, and the episode-level collision rates of 5.7–9.5% in deliberately stressed traffic are not deployment grade; the supported claim is relative (49–56% below the best learned baseline). **Correlational dynamics.** The learned dynamics are prior-constrained but not causally identified; verification is conservative, so misattribution degrades toward rejecting actions rather than accepting unsafe ones, yet its quality depends on dynamics fidelity (constant-velocity propagation raises CR by 48–49%; 3 of the 57 acquired rules were over-specific in audit, Appendix D.5). **Limited zero-shot cross-domain reusability.** Zero-shot cross-domain reusability of the compiled memory is limited: a memory compiled solely from highD/rounD begins at an 86.2% call rate on highway-env, indicating that many target-domain encounters still require online acquisition. REINS is accordingly scoped to the self-evolution loop rather than assuming universal cross-domain memory transfer (Appendix D.7). **Scope.** The monotone miss-rate property assumes a stationary scene distribution, and the implementation is specialized to driving: only the variable schema and the action space must be re-authored for another domain (Appendix D.1).

7 Conclusion

We present REINS, a self-evolving agent harness that adapts the agent-harness paradigm from software engineering to real-time, safety-critical trajectory planning through three LLM–planner couplings: grounding, verification, and consolidation. Familiar scenes resolve from an indexed rule memory in $\mathcal{O}(\log|\mathcal{R}|)$ time, while memory misses trigger LLM-driven rule acquisition that grounds, verifies, and writes back new rules, closing the self-evolution loop without any LLM weight update. At the tactical decision layer, REINS attains 98%+ compliance, nearly 50% collision reduction, and 1.9–4.6 ms latency, with gains that survive matched baselines, observation noise, reactive replay, and cross-domain memory transfer. Future work will couple the harness to perception uncertainty and continuous maneuvers, extend it to multi-agent coordination, and move the learned dynamics toward interventional identification with incremental updates.

Acknowledgments and Disclosure of Funding

This work was supported by the European Union's Horizon Europe research and innovation programme under Marie Skłodowska-Curie grant agreement No. 101126636 (DSTrain) and by the Technical Faculty of IT and Design at Aalborg University. This work was partly supported by the European Union under Grant 101123471 (EDGE-Skills project funded under the Digital Europe Programme). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them. The authors declare no competing interests.

References

Yusra Alkhazraji, Matthias Frorath, Markus Grützner, Malte Helmert, Thomas Liebetraut, Robert Mattmüller, Manuela Ortlieb, Jendrik Seipp, Tobias Springenberg, Philip Stahl, and Jan Wülfing. Pyperplan. Zenodo, `https://doi.org/10.5281/zenodo.3700819`, 2020.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

Tianhui Cai, Yifan Liu, Zewei Zhou, Haoxuan Ma, Seth Z. Zhao, Zhiwen Wu, Xu Han, Zhiyu Huang, and Jiaqi Ma. Driving with regulation: Trustworthy and interpretable decision-making for autonomous driving with retrieval-augmented reasoning. *arXiv preprint arXiv:2410.04759*, 2024.

Haoqiang Chen, Jianxiang Sun, Yadong Liu, Zhiyong Peng, and Dewen Hu. Causal confusion in pedestrian crossing intention prediction for autonomous vehicles: The role of ego-vehicle speed. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, and Ziran Wang. Drive as you speak: Enabling human-like interaction with large language models in autonomous vehicles. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 902–909, 2024.

Zhihong Cui, Haoran Tang, Tianyi Li, Yushuai Li, Peiyuan Guan, Amir Taherkordi, and Tor Skeie. C-TRAIL: A commonsense world framework for trajectory planning in autonomous driving. *IEEE Transactions on Vehicular Technology*, 2026.

Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, 2007.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1433–1445, 2018.

Daocheng Fu, Xin Li, Licheng Wen, Min Dou, Pinlong Cai, Botian Shi, and Yu Qiao. Drive like a human: Rethinking autonomous driving with large language models. In *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, pages 910–919, 2024.

Douglas Gettman and Larry Head. Surrogate safety measures from traffic simulation models. *Transportation Research Record*, 1840(1):104–115, 2003.

Luca Giamattei, Antonio Guerriero, Roberto Pietrantuono, and Stefano Russo. Causality-driven testing of autonomous driving systems. *ACM Transactions on Software Engineering and Methodology*, 33(3):1–35, 2024.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pretrained large language models to construct and utilize world models for model-based task planning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model MOBIL for car-following models. *Transportation Research Record*, 1999(1):86–94, 2007.

Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE, 2018.

Robert Krajewski, Tobias Moers, Julian Bock, Lennart Vater, and Lutz Eckstein. The rounD dataset: A drone dataset of road user trajectories at roundabouts in Germany. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. In *CoRR abs/2603.28052*, 2026.

David Lenz, Tobias Kessler, and Alois Knoll. Tactical cooperative planning for autonomous highway driving using Monte-Carlo tree search. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 447–453. IEEE, 2016.

Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

Boyi Li, Yue Wang, Jiageng Mao, Boris Ivanovic, Sushant Veer, Karen Leung, and Marco Pavone. Driving everywhere with large language model policy adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14948–14957, 2024a.

Yiheng Li, Cunxin Fan, Chongjian Ge, Zhihao Zhao, Chenran Li, Chenfeng Xu, Huaxiu Yao, Masayoshi Tomizuka, Bolei Zhou, Chen Tang, Mingyu Ding, and Wei Zhan. WOMD-Reasoning: A large-scale dataset for interaction reasoning in driving. *arXiv preprint arXiv:2407.04281*, 2024b.

Yuxuan Liang, Kun Ouyang, Yiwei Wang, Xu Liu, Hongyang Chen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. TrajFormer: Efficient trajectory classification with transformers. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1229–1237, 2022.

Ben Liu, Jihai Zhang, Fangquan Lin, Cheng Yang, Min Peng, and Wotao Yin. SymAgent: A neural-symbolic self-learning agent framework for complex reasoning over knowledge graphs. In *Proceedings of the ACM on Web Conference 2025*, pages 98–108, 2025.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuan-Fang Li, Chen Gong, and Shirui Pan. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *arXiv preprint arXiv:2410.13080*, 2024.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *ICLR*, 2024.

Kumar Manas, Stefan Zwicklbauer, and Adrian Paschke. TR2MTL: LLM based framework for metric temporal logic formalization of traffic rules. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 1206–1213. IEEE, 2024.

Kumar Manas, Mert Keser, and Alois Knoll. Integrating legal and logical specifications in perception, prediction, and planning for automated driving: A survey of methods. *arXiv preprint arXiv:2510.25386*, 2025.

Jiageng Mao, Junjie Ye, Yuxi Qian, Marco Pavone, and Yue Wang. A language agent for autonomous driving. In *First Conference on Language Modeling (COLM)*, 2024.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *CoRR abs/2601.11868*, 2026.

Henry Montgomery. Decision rules and the search for a dominance structure: Towards a process model of decision making. In *Advances in Psychology*, volume 14, pages 343–369. Elsevier, 1983.

Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.

Stefan Riedmaier, Thomas Ponn, Dieter Ludwig, Bernhard Schick, and Frank Diermeyer. Survey on scenario-based safety assessment of automated vehicles. *IEEE Access*, 8:87456–87477, 2020.

Hao Sha, Yao Mu, Yuxuan Jiang, Li Chen, Chenfeng Xu, Ping Luo, Shengbo Eben Li, Masayoshi Tomizuka, Wei Zhan, and Mingyu Ding. LanguageMPC: Large language models as decision makers for autonomous driving. *arXiv preprint arXiv:2310.03026*, 2023.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

Xiaolong Tang, Meina Kan, Shiguang Shan, and Xilin Chen. Plan-R1: Safe and feasible trajectory planning as language modeling. *arXiv preprint arXiv:2505.17659*, 2025.

Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824, 2000.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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. In *CoRR abs/2305.16291*, 2023.

Yucen Wang, Rui Yu, Shenghua Wan, Le Gan, and De-Chuan Zhan. FOUNDER: Grounding foundation models in world models for open-ended embodied decision making. In *Forty-Second International Conference on Machine Learning*, 2025.

Jim Webber. A programmatic introduction to Neo4j. In *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*, pages 217–218, 2012.

Fanzhi Zeng, Siqi Wang, Chuzhao Zhu, and Li Li. ADRD: LLM-driven autonomous driving based on rule-based decision systems. *arXiv preprint arXiv:2506.14299*, 2025.

Mingliang Zhai, Cheng Li, Zengyuan Guo, Ningrui Yang, Xiameng Qin, Sanyuan Zhao, Junyu Han, Ji Tao, Yuwei Wu, and Yunde Jia. World knowledge-enhanced reasoning using instruction-guided interactor in autonomous driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 9842–9850, 2025.

Jiahui Zhang, Zhiqiang Jian, Jiawei Fu, Zhixiong Nan, Jingmin Xin, and Nanning Zheng. Trajectory planning with comfort and safety in dynamic traffic scenarios for autonomous driving. In *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, pages 342–349. IEEE, 2021a.

Lunjun Zhang, Ge Yang, and Bradly C Stadie. World model as a graph: Learning latent landmarks for planning. In *International Conference on Machine Learning*, pages 12611–12620. PMLR, 2021b.

Yuhang Zhang, Jiaqi Liu, Chengkai Xu, Peng Hang, and Jian Sun. LeAD: The LLM enhanced planning system converged with end-to-end autonomous driving. In *CoRR abs/2507.05754*, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *AAAI*, 2024.

A More Details about REINS

A.1 Variable Definitions

We define macro and micro variables shown in Table 4. The macro variables represent high-level semantic categories, while micro variables are the specific measurable quantities used in the world dynamics model and rule system.

The variables are organized into five functional groups: **Environment** variables capture external conditions (weather, visibility, road surface, speed limits) that affect driving but are not directly controllable. **Traffic** variables describe spatial and temporal relationships with surrounding vehicles, including headway distances, relative speeds, TTC, and lane density metrics. **Ego** variables represent the controlled vehicle's kinematic state (position, velocity, acceleration, orientation). **Action** defines the discrete action space: five high-level maneuvers executed by low-level controllers. **Outcome** variables measure action consequences, including safety (collision risk), comfort (jerk-based), and task completion indicators. This hierarchical organization enables structured rule retrieval via variable-group indexing.

The macro-group dependency set $\mathcal{E}_{\mathcal{H}}$ in Def. 3.1 encodes the admissible group-level edges:

$$\begin{aligned}\mathcal{E}_{\mathcal{H}} = \{&(\textsc{Env}, \textsc{Traffic}), (\textsc{Env}, \textsc{Ego}), (\textsc{Env}, \textsc{Outcome}),\\ &(\textsc{Traffic}, \textsc{Traffic}), (\textsc{Traffic}, \textsc{Ego}), (\textsc{Traffic}, \textsc{Outcome}),\\ &(\textsc{Ego}, \textsc{Traffic}), (\textsc{Ego}, \textsc{Ego}), (\textsc{Ego}, \textsc{Outcome}),\\ &(\textsc{Action}, \textsc{Ego}), (\textsc{Action}, \textsc{Outcome})\}.\end{aligned} \tag{10}$$

Environment factors influence Traffic, Ego, and Outcome; Traffic and Ego variables interact bidirectionally and feed into Outcome; Action affects only Ego state and Outcome.

A.2 Rule Templates

Seven rule templates are defined to capture different logical structures for driving knowledge:

- `IF-THEN`: Conditional implication for basic conditional relations.
  Example: IF TTC $<$ 2s THEN decelerate
- `AND-THEN`: Conjunctive trigger requiring multiple conditions.
  Example: IF visibility $<$ 50m AND raining THEN reduce_speed_by_30%
- `THRESHOLD`: Numeric constraint with calibrated boundaries.
  Example: IF speed $>$ speed_limit THEN violation = true
- `PRIORITY`: Decision hierarchy for conflicting actions.
  Example: yield $\succ$ accelerate (yield takes priority)
- `MULTI-ACTION`: Compound actions under one condition.
  Example: IF collision_risk = high THEN brake AND steer_away
- `WEIGHTED-ACTION`: Weighted action selection.
  Example: IF crash_risk THEN {brake: 0.7, steer: 0.3}
- `TEMPORAL-CHAIN`: Temporal constraints between triggers.
  Example: IF obstacle_detected THEN brake within 1.5s

A.3 Rule Conflict Resolution

When multiple rules are activated simultaneously, conflicts may arise if they recommend different actions. REINS resolves conflicts through a three-level arbitration mechanism:

Priority-based ordering. Rules are assigned priority levels based on their functional category: Safety $>$ Compliance $>$ Navigation $>$ Efficiency. When rules from different categories conflict, the higher-priority rule takes precedence. For example, if an Efficiency rule (consider lane change) and a Safety rule (maintain gap) both fire, the Safety rule overrides.

Confidence-weighted aggregation. For rules within the same priority level, REINS aggregates their scores using the noisy-OR formula (Eq. (5)). This allows multiple supporting rules to reinforce an action while remaining robust to individual noisy predictions. The action with the highest aggregated probability is selected.

Macro Variable	Micro Variable	Description
Environment	weather_type	Weather condition (clear/rain/fog/snow)
	visibility_distance	Visibility range (m)
	road_friction	Road surface friction coefficient
	road_type	Scenario type (highway/merge/intersection/roundabout)
	time_of_day	Time period (day/night/dawn/dusk)
	speed_limit	Posted speed limit (km/h)
Traffic	headway_distance	Distance to lead vehicle (m)
	lead_speed	Lead vehicle speed (m/s)
	lead_rel_speed	Relative speed to lead vehicle (m/s)
	lead_ttc	Time-to-collision with lead vehicle (s)
	lead_thw	Time headway to lead vehicle (s)
	rear_distance	Distance to rear vehicle (m)
	rear_speed	Rear vehicle speed (m/s)
	rear_rel_speed	Relative speed to rear vehicle (m/s)
	rear_ttc	Time-to-collision with rear vehicle (s)
	left_front_distance	Distance to left-front vehicle (m)
	left_front_speed	Left-front vehicle speed (m/s)
	left_rear_distance	Distance to left-rear vehicle (m)
	left_rear_speed	Left-rear vehicle speed (m/s)
	right_front_distance	Distance to right-front vehicle (m)
	right_front_speed	Right-front vehicle speed (m/s)
	right_rear_distance	Distance to right-rear vehicle (m)
	right_rear_speed	Right-rear vehicle speed (m/s)
	lane_density_left	Left lane density (vehicles/100m)
	lane_density_right	Right lane density (vehicles/100m)
	available_gap_left	Available gap in left lane (m)
	available_gap_right	Available gap in right lane (m)
	cutin_risk	Cut-in risk indicator (0–1)
Ego	ego_x	Ego vehicle longitudinal position (m)
	ego_y	Ego vehicle lateral position (m)
	ego_vx	Ego vehicle longitudinal velocity (m/s)
	ego_vy	Ego vehicle lateral velocity (m/s)
	ego_speed	Ego vehicle total speed (m/s)
	ego_acceleration	Ego vehicle acceleration (m/s^2)
	yaw_rate	Ego vehicle yaw rate (rad/s)
	jerk	Ego vehicle jerk (m/s^3)
	lane_id	Current lane identifier
	lateral_offset	Lateral offset from lane center (m)
Action	action_type	Discrete action (IDLE/FASTER/SLOWER/LANE_LEFT/LANE_RIGHT)
Outcome	risk_score	Collision probability (0–1)
	comfort_score	Comfort metric (jerk-based)
	energy_consumption	Energy usage metric
	success_flag	Goal completion indicator (boolean)
	violation_flag	Rule violation indicator (boolean)

Table 4: Macro and micro variable definitions used in REINS.

Dynamics-based verification. When the aggregated action prior is below a confidence threshold ($\tau = 0.7$), REINS falls back to dynamics-based verification. The top-M candidate actions are simulated via forward rollout using the world dynamics model, and the action yielding the safest predicted trajectory is selected.

This hierarchical approach ensures that safety-critical rules are never overridden by efficiency considerations, while still allowing flexible action selection in non-critical situations.

A.4 Method Details

This subsection contains the full equations, architectural details, and explanatory prose deferred from the methodology to keep the main body concise. The structure mirrors Section 4.1–4.2.

A.4.1 World Dynamics Learning Details

We learn the dependency graph $\hat{\mathcal{G}} = (\mathcal{X}, \hat{\mathcal{E}}, \hat{w})$ in three steps, then train a per-variable dynamics function on the resulting parent sets.

Step 1: Edge Prior Estimation. We query the LLM with variable dependency prompts (Appendix A.5) to obtain an edge confidence $\pi_{ij} \in [0, 1]$ for each candidate edge $X_j \to X_i$. This is combined with the macro-group constraints (Eq. 10) to form a unified prior:

$$\tilde{\pi}_{ij} = \alpha\, \pi_{ij} + (1 - \alpha)\, \mathbb{I}\{(h(X_j), h(X_i)) \in \mathcal{E}_{\mathcal{H}}\}, \tag{11}$$

where $\alpha \in [0, 1]$ balances LLM confidence against structural constraints. Edges that violate the macro-group constraints carry indicator 0 and are reduced to $\alpha\, \pi_{ij}$, while semantically aligned edges are pushed toward 1.

Step 2: Prior-Guided Regression. For each target variable X_i, we predict its value from the lagged values of other variables via ridge regression [Hoerl and Kennard, 1970] with prior-adaptive penalties $\lambda_{ij} = \lambda_0(1 - \tilde{\pi}_{ij})^\rho$:

$$\hat{\mathbf{w}}_i = \arg\min_{\mathbf{w}} \sum_{t=L}^{N} \left(X_i^t - \textstyle\sum_{j \neq i} \sum_{\ell=1}^{L} w_{j\ell} X_j^{t-\ell} \right)^2 + \sum_{j \neq i} \lambda_{ij} \|\mathbf{w}_j\|_2^2, \tag{12}$$

where $w_{j\ell}$ is the regression coefficient for variable X_j at lag $\ell \in \{1, \ldots, L\}$, $\mathbf{w}_j = (w_{j1}, \ldots, w_{jL})^\top$ groups all lags for X_j, λ_0 is the base penalty, and $\rho > 0$ controls penalty sharpness. High-prior edges ($\tilde{\pi}_{ij} \to 1$) receive weak regularization and survive; low-prior edges are shrunk toward zero.

Step 3: Graph Construction. The parent set $\mathrm{Pa}(X_i)$ retains the top-K variables ranked by aggregated coefficient magnitude $\sum_{\ell=1}^{L} |\hat{w}_{j\ell}|$. The learned edge set and weights are

$$\hat{\mathcal{E}} = \{(X_j, X_i) \mid X_j \in \mathrm{Pa}(X_i)\}, \quad \hat{w}(X_j, X_i) = \textstyle\sum_{\ell=1}^{L} |\hat{w}_{j\ell}|. \tag{13}$$

Dynamics Function Loss. Given $\hat{\mathcal{G}}$, each f_i is implemented as a multi-head Transformer [Vaswani et al., 2017] and trained by mean squared error over its parent trajectory:

$$\mathcal{L}_{\text{dyn}}^{(i)} = \frac{1}{N} \sum_{t} \left(X_i^{t+1} - f_i(\mathrm{Pa}(X_i)^{t-L+1:t}) \right)^2. \tag{14}$$

Optimizer, batch size, and convergence curves are reported in Appendix C.1.

A.4.2 Rule Compilation Details

Rule Structure Generation. For each dependency edge $(X_j, X_i) \in \hat{\mathcal{E}}$, the LLM generates a rule structure

$$r = (T, \mathcal{P}_r, a_r, c_r), \tag{15}$$

where T is the rule template (Def. 3.5), $\mathcal{P}_r$ is a set of guard predicates with symbolic thresholds, $a_r \in \mathcal{A}$ is the recommended action (Def. 3.2), and c_r is the confidence derived from edge weight $\hat{w}(X_j, X_i)$. Prompt details are in Appendix A.5.

Threshold Calibration. Each predicate $p \in \mathcal{P}_r$ (Def. 3.4) requires a concrete threshold τ_i, calibrated from $\mathcal{D}$ in three forms [Luo et al., 2024]:

$$\tau_i = \begin{cases} \mu_i + \zeta\sigma_i & \text{(z-score)}, \\ Q_i(\zeta) & \text{(quantile)}, \\ \tau_0 & \text{(absolute)}, \end{cases} \tag{16}$$

where μ_i, σ_i are empirical mean and standard deviation of X_i, $Q_i(\zeta)$ is the ζ-quantile, and τ_0 is a fixed constant (e.g., speed limits). The comparison operator $\prec$ is selected to maximize safety separability between the two sides of the threshold under safety outcome $y \in \{0, 1\}$ (Section 4.1.2).

Graph Database Storage. Compiled rules are stored in a Neo4j graph database [Webber, 2012] with a variable-indexed schema (Figure 3). Each rule node stores the complete tuple $(T, \mathcal{P}_r, a_r, c_r)$, where T is the template type, $\mathcal{P}_r$ contains calibrated predicates with concrete thresholds τ and operators $\prec$, a_r is the recommended action, and c_r is the confidence. Rule nodes connect to variable nodes via `TRIGGERS` (input variables) and `AFFECTS` (output variables) edges. This enables $O(\log |\mathcal{R}|)$ retrieval via parameterized Cypher queries on variables active in the current state. Database statistics are provided in Appendix C.7.1.

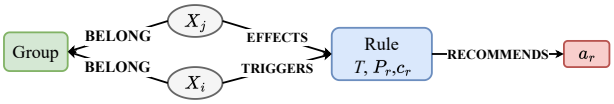


Figure 3: Rule database schema.

A.4.3 Rule-Guided Planning Details

Scene-Aware Rule Retrieval. Given $s_t \in \mathcal{S}$, we identify active variables $\{X_i\}$ and retrieve context-relevant rules via parameterized Cypher queries [Francis et al., 2018] generated by the LLM (read-only and schema-constrained; Appendix A.5). The retrieval map

$$G : (\{X_i\}, K) \longmapsto \mathcal{R}_t \subseteq \mathcal{R} \tag{17}$$

returns up to K rules ranked by confidence c_r. A rule r is retrieved if its trigger variables $\{X_j \mid (X_j, r) \in \texttt{TRIGGERS}\}$ intersect with the active set $\{X_i\}$ and the dependency direction satisfies the macro-group constraints $\mathcal{E}_{\mathcal{H}}$.

Action Scorer Architecture. The Transformer scorer [Liang et al., 2022] takes state encoding $\mathbf{x}_t$ and rule embedding $\mathbf{u}_r$ and emits relevance and action heads:

$$\begin{aligned} \mathbf{h}_{t,r} &= \text{TransformerEnc}([\mathbf{x}_t; \mathbf{u}_r]), \\ S_{\text{rel}}(r, s_t) &= \sigma(\text{Dec}_{\text{rel}}(\mathbf{h}_{t,r})) \in [0, 1], \\ S_{\text{act}}(r, s_t, a) &= \text{softmax}(\text{Dec}_{\text{act}}(\mathbf{h}_{t,r}))_a, \end{aligned} \tag{18}$$

where S_{rel} predicts whether r should influence the current decision, and S_{act} predicts the action preference given r. The combined rule-action score multiplies relevance, action preference, and rule confidence:

$$f(r, a \mid s_t) = S_{\text{rel}}(r, s_t) \cdot S_{\text{act}}(r, s_t, a) \cdot c_r. \tag{19}$$

Scorer Training. The scorer is trained on expert trajectories $\{(s_t, a_t^*)\}$ with a joint relevance + action objective:

$$\mathcal{L} = \sum_{t,r} \left[\text{BCE}(\text{Exec}(r, s_t), S_{\text{rel}}(r, s_t)) - \log S_{\text{act}}(r, s_t, a_t^*)\right], \tag{20}$$

where $\text{Exec}(r, s_t) \in \{0, 1\}$ is the ground-truth rule activation (Eq. 2) and a_t^* is the expert action.

Action Aggregation. The full noisy-OR action distribution corresponding to Eq. 5 is

$$\begin{aligned} P(a \mid s_t) &= \frac{1 - \prod_{r \in \mathcal{R}_t(a)} \left(1 - f(r, a \mid s_t)\right)}{Z}, \\ a^*_{\text{rule}} &= \arg\max_{a \in \mathcal{A}} P(a \mid s_t), \end{aligned} \tag{21}$$

where $\mathcal{R}_t(a) \subseteq \mathcal{R}_t$ denotes the rules with recommended action $a_r = a$ and Z normalizes over $\mathcal{A}$. The formulation favors actions supported by multiple high-confidence rules.

Trajectory Verification. For the top-M actions ranked by $P(a \mid s_t)$, we predict H-step future states:

$$\hat{s}_{t+h+1} = \Phi(\hat{s}_{t+h}, a; \hat{\mathcal{G}}, \hat{\mathcal{F}}), \quad h = 0, \ldots, H-1, \tag{22}$$

where Φ applies each $f_i \in \hat{\mathcal{F}}$ to predict X_i from $\mathrm{Pa}(X_i)$ under $\hat{\mathcal{G}}$. Each rollout produces a candidate trajectory $\hat{\xi}(a) = (\hat{s}_{t+1}, \ldots, \hat{s}_{t+H})$, consistent with Def. 3.3. The planner selects $a_t^\star = \arg\max_{a \in \mathcal{A}_M} J(\hat{\xi}(a))$, where $J : \mathcal{S}^H \to \mathbb{R}$ penalizes collisions, rule violations, and control effort (Appendix B.2); the first control of the selected trajectory is executed in receding-horizon [Zhang et al., 2021b].

A.4.4 Online Rule Acquisition Details

LLM-Driven Rule Proposal. On a miss, the harness queries $\mathcal{M}$ with the current state and active variables:

$$\tilde{r} = (\tilde{T}, \tilde{\mathcal{P}}, \tilde{a}, \tilde{c}) \sim \mathcal{M}(s_t), \tag{23}$$

where $\tilde{r}$ conforms to the rule template (Def. 3.5) and uses only variables in $\mathcal{X}$ and operators from the predicate grammar (Def. 3.4). Prompt details are in Appendix A.5.

Safe-Action Selection. When online rule acquisition rejects $\tilde{r}$ and $\mathcal{R}_t = \emptyset$, the harness raises a human-takeover alert and executes the action that maximizes minimum lookahead time-to-collision under $\hat{\mathcal{F}}$:

$$a_t^{\text{safe}} = \arg\max_{a \in \mathcal{A}} \min_{h \in [1,H]} \mathrm{TTC}(\hat{s}_{t+h}(a)) \,. \tag{24}$$

Memory-Update Collision Branch. Eq. 8's collision branch averages confidences instead of creating a duplicate node, so $|\mathcal{R}|$ counts distinct rule structures rather than LLM-noise replicas. The hash κ is computed over the structural triple $(\tilde{T}, \tilde{\mathcal{P}}, \tilde{a})$ but excludes $\tilde{c}$, so semantically identical proposals collide regardless of confidence noise. Collisions are common during early deployment when $\mathcal{R}$ is small, then taper off as coverage saturates.

Self-Evolution Properties. Eq. 8 structurally enforces $|\mathcal{R}_{t+1}| \geq |\mathcal{R}_t|$ (memory monotonicity) and $\theta_{\mathcal{M}} \equiv \theta_{\mathcal{M}}^{(0)}$ (no LLM parameter update). Under stationarity of the scene distribution, the per-state miss rate is non-increasing as $\mathcal{R}$ accumulates coverage, since any state s that triggered a miss and produced an accepted $\tilde{r}$ is now matched by a rule keyed on the variables active at s. Since $\mathcal{M}$ is invoked only on misses, its per-scene invocation rate decays monotonically with $\Pr[\mathrm{Miss}]$. We validate this behavior empirically in Section 5.4 and Appendix C.5.

A.5 Prompts

We provide three prompt templates used in REINS's offline compilation pipeline. Figure 4 shows the **Variable Dependency Prompt**, which instructs the LLM to infer dependencies between driving variables based on domain knowledge; the output serves as semantic priors for structure learning in the world dynamics model. Figure 5 presents the **Rule Extraction Prompt**, which converts dependency edges into executable driving rules with parameterized guards and discrete actions; these rules are stored in the Neo4j graph database. Figure 6 displays the **Parameterized Cypher Query Prompt**, which generates parameterized queries for context-aware rule retrieval during planning. All prompts enforce read-only constraints and structured output formats to ensure safety and parseability.

B Implementation Details

B.1 Experiment Settings

Table 5 summarizes the key hyperparameters and training configurations used in our experiments. The settings are organized into six categories: **Simulation Environment** specifies the `highway-env` simulator settings, including four driving scenarios with varying complexity and a discrete action space of five high-level maneuvers. **Real-world Datasets** describes the highD (highway) and rounD (roundabout) drone datasets used for real-world evaluation, including recording statistics

Variable Dependency Prompt

[**TASK**]
You are **an expert in autonomous driving systems and variable dependency reasoning**.
Your task is to identify a set of **plausible dependency relationships** among driving-related variables observed in an autonomous vehicle simulation environment.
Each relationship should describe how one variable may influence another under realistic driving conditions.
These dependency relations will be used to construct a **dependency graph** for downstream decision-making and trajectory planning.
Each dependency relationship must be:
• **Physically plausible**: reflecting physical laws, driving behavior, or traffic context;
• **Directional**: clearly indicating the influencing variable and the affected variable;
• **Actionable**: relevant to planning, safety, efficiency, or comfort;
• **Concise**: expressed as a simple directed pair without explanation.
[**INPUT**]
You are given a set of driving-related variables extracted from a simulation environment:
• Variable list: `{variable_list}`
[**OUTPUT**]
Please output a list of dependency relationships in the following strict format.
Each relationship must be written as:

`SOURCE` $\rightarrow$ `TARGET`

Return all relationships enclosed in triple quotes, and do not include any explanations or additional text.
’’’
SOURCE $\rightarrow$ TARGET
SOURCE $\rightarrow$ TARGET
SOURCE $\rightarrow$ TARGET
’’’
[**EXAMPLE**]
’’’
ego_speed $\rightarrow$ energy_consumption
road_friction $\rightarrow$ risk_score
headway_distance $\rightarrow$ cutin_risk
’’’
Make sure that your output strictly follows this format. Any relationship that does not adhere to the `SOURCE` $\rightarrow$ `TARGET` structure should be removed.

Figure 4: Variable dependency prompt. Given a list of driving-related variables, the LLM infers directed dependency relationships as structural priors for the world dynamics model.

Rule Extraction Prompt

[**TASK**]
You are an expert in autonomous driving and rule-based reasoning.
Given dependency edges from a learned dependency graph, extract executable driving rules that can be stored in a graph database and used for trajectory planning.
Each rule must be:
• **Grounded**: based on the learned dependency structure;
• **Executable**: containing guards with parameterizable thresholds;
• **Actionable**: mapping to discrete actions (LANE_LEFT, LANE_RIGHT, IDLE, FASTER, SLOWER).
[**INPUT**]
You are given dependency edges with learned weights from the dependency graph:
• Dependency edges: `{dependency_edges}`
• Variable descriptions: `{variable_descriptions}`
[**OUTPUT FORMAT**]
Return a JSON array of rule objects. Each object must include:

```
{"rule_index", "rule_type", "variables", "params", "guards", "action", "confidence"}
```

```
{
 "rule_index": "R_xxx",
  "rule_type": "physical|behavioral|scene",
  "variables": ["cause_var", "effect_var"],
  "params": {"threshold_form": "z|q|abs", "k": <float>},
  "guards": ["cause_var <op> threshold"],
  "action": "SLOWER|FASTER|LANE_LEFT|LANE_RIGHT|IDLE",
  "confidence": <float in [0,1]>
}
```

[**EXAMPLE**]
Input: headway_distance → risk_score (weight=0.85)
Output:

```
{"rule_index": "R_001", "rule_type": "behavioral",
 "variables": ["headway_distance", "risk_score"],
 "params": {"threshold_form": "z", "k": -1.5},
 "guards": ["headway_distance < threshold"],
 "action": "SLOWER", "confidence": 0.85}
```

Make sure each rule follows the format strictly and maps to one of the five discrete actions.

Figure 5: Rule extraction prompt from dependency structure.

and sampling rates. **World Dynamics Model** details the structure learning parameters (LLM prior weight, regularization) and neural network architecture (Transformer layers, attention heads). **Rule Compilation** covers the offline LLM-based rule generation process, including the backbone model, sampling strategy, and threshold calibration. **Training & Evaluation** specifies the data splits, random seeds, and hardware configuration. **Planning & Deployment** reports inference-time parameters and latency measurements demonstrating real-time capability. All hyperparameter configurations follow standard practices unless otherwise stated.

B.2 Evaluation Metrics

Rule Compliance (Compl.) measures the fraction of timesteps where all active rules are satisfied:

$$\text{Compl.} = \frac{1}{T}\sum_{t=1}^{T}\prod_{k=1}^{K}\big(1 - v_{k,t}\big), \tag{25}$$

where $v_{k,t} \in \{0, 1\}$ indicates whether rule k is violated at time t.

Collision Rate (CR) measures the fraction of episodes in which at least one collision occurs:

$$\text{CR} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\text{collision in episode } i), \tag{26}$$

Category	Setting
Simulation Environment	
Simulator	highway-env v1.8.2
Scenarios	Highway, Merge, Roundabout, Intersection
Episode length	100 steps (20s at 5 Hz)
Observation	15 nearest vehicles
Action space	{IDLE, FASTER, SLOWER, LANE_LEFT, LANE_RIGHT}
Weather conditions	Clear, Rain, Fog, Snow
Real-world Datasets	
highD source	German Autobahn (drone)
highD recordings	01, 02, 03, 04
highD vehicles	4,237 (578 lane-change events)
rounD source	German roundabouts
rounD recordings	00, 01, 02
rounD vehicles	2,156 (423 yield/merge events)
Sampling rate	25 Hz
World Dynamics Model	
Structure learning	Ridge regression + LLM prior
LLM prior weight α	0.6
Penalty exponent ρ	2.0
Base regularization λ_0	0.1
Lag length L	6 steps
Top-K parents	5
Transformer layers	2
Attention heads	4
Optimizer	Adam
Learning rate	2×10^{-4}
Batch size	64
Training episodes	50k
Rule Compilation (Offline)	
LLM backbone	GPT-4o (2024-08-06)
LLM temperature	0.0 (deterministic)
Max output tokens	4096
Prompt tokens (avg.)	$\sim$1.2k per query
Total LLM calls	$\sim$50 (one-time offline)
Rule templates	7 types
Neo4j version	5.12
Threshold calibration	Gaussian ($\mu + \zeta\sigma$), ζ=1.5
Confidence τ	0.7
Training & Evaluation	
Training data	50k episodes
Test episodes	1000 per scenario
Random seeds	5 (0, 42, 123, 456, 789)
Training hardware	4$\times$ A100 (40GB)
Training time	$\sim$8 hours
Planning & Deployment	
Retrieval count k	10
Planning horizon H	5 steps
Rollout discount γ	0.99
Safety weight λ_{safe}	1.0
Comfort weight λ_{comf}	0.1
Progress weight λ_{prog}	0.5
Inference hardware	RTX 3090 or above
GPU memory	$\sim$4 GB
Neo4j query latency	$<$1 ms
Planning latency	1.9–4.6 ms per decision
Real-time capability	200–500 Hz

Table 5: Experiment settings and hyperparameters for REINS.

Parameterized Cypher Query Prompt

[**TASK**]
You are an expert in Neo4j graph databases and autonomous driving systems.
Generate a parameterized, **read-only** Cypher query to retrieve relevant driving rules based on the current scene context.
[**INPUT**]
You are given the following information about the current driving scene:
• Scene summary: `{scene_summary}`
• Active cause variables: `{active_causes}`
• Target variable groups: `{target_groups}` (e.g., Outcome, Ego, Action)
• Maximum results: `{k}`
[**GRAPH SCHEMA**]
The rule database contains the following structure:
• Nodes: `(Rule)`, `(Variable)`, `(Category)`, `(Group)`, `(Action)`
• Relations: `[:TRIGGERS]`, `[:CAUSES]`, `[:BELONG_TO]`
[**CONSTRAINTS**]
Your query must satisfy the following safety requirements:
• Query MUST start with: `MATCH`, `WITH`, `RETURN`, `CALL`, or `UNWIND`
• NO write operations: `CREATE`, `MERGE`, `SET`, `DELETE`, `DETACH`, `LOAD`, `DROP`, `ALTER`
• Use parameter placeholders: `$active_causes`, `$target_groups`, `$k`
[**OUTPUT**]
Return only the Cypher query string enclosed in triple quotes. Do not include any explanations.

```
'''
<Cypher query here>
'''
```

[**EXAMPLE**]

```
'''
MATCH (r:Rule)-[:TRIGGERS]->(c:Variable)-[:CAUSES*0..1]->(e:Variable)
WHERE c.name IN $active_causes AND e.group IN $target_groups
RETURN r.rule_id, r.text, collect(e.name) AS effects,
       r.type, r.confidence
ORDER BY r.confidence DESC LIMIT $k
'''
```

Ensure the query is syntactically correct and adheres to all constraints.

Figure 6: Parameterized Cypher query prompt.

where N is the number of test episodes. A lower CR indicates safer planning.

Risk-aware Efficiency (RE) balances task performance against safety:

$$\text{RE} = R_{\text{task}}(\tau) - \lambda_{\text{risk}} R_{\text{risk}}(\tau), \tag{27}$$

$$R_{\text{risk}}(\tau) = \sum_{t=1}^{T} \max(0, \delta_{\text{ttc}} - \text{TTC}_t), \tag{28}$$

where R_{task} is the task reward, R_{risk} penalizes low TTC states, $\lambda_{\text{risk}} = 0.5$ is the risk penalty weight, and $\delta_{\text{ttc}} = 3.0$s is the TTC threshold.

Planning Efficiency (PE) measures average computation time per decision:

$$\text{PE} = \frac{1}{T} \sum_{t=1}^{T} t_{\text{plan}}, \tag{29}$$

where t_{plan} (ms) is the wall-clock time for action selection.

Trajectory Objective $J(\hat{\xi})$. The trajectory-level objective used in dynamics-based verification is defined as:

$$J(\hat{\xi}) = \sum_{h=1}^{H} \gamma^{h-1} \Big[- \lambda_{\text{safe}} \cdot \mathbb{I}(\text{TTC}_{t+h} < \tau_{\text{ttc}}) + \lambda_{\text{prog}} \cdot \Delta d_{t+h} - \lambda_{\text{comf}} \cdot |j_{t+h}| \Big], \tag{30}$$

where TTC_{t+h} is the predicted time-to-collision, Δd_{t+h} is the progress toward the goal, j_{t+h} is the jerk, and $\tau_{\text{ttc}} = 2.0$s is the safety threshold. The weights $\lambda_{\text{safe}} = 1.0$, $\lambda_{\text{prog}} = 0.5$, $\lambda_{\text{comf}} = 0.1$ prioritize safety over progress and comfort.

Real-world Evaluation Protocol. We use closed-loop replay: one recorded vehicle is controlled by the planner while others follow recorded trajectories. Collisions are detected when bounding boxes overlap or the ego deviates beyond lane boundaries. In the simulated scenarios the surrounding vehicles instead react to the ego (IDM with MOBIL); a reactive-replay variant of highD is reported in Appendix D.6.

B.3 Baseline Descriptions

We select baselines from three categories to comprehensively evaluate REINS: (i) *classical planners* (MPC, MCTS) that represent traditional optimization and search-based approaches without rule guidance; (ii) *LLM-based planners* (DriveSpeak, PDDL, LLaDA, DriReg) that leverage language models for driving decisions but differ in how they integrate linguistic knowledge; and (iii) *hybrid methods* (SymAgent, Plan-R1) that combine learning with symbolic reasoning. This selection covers the spectrum from rule-free to rule-guided planning, enabling analysis of how different rule representation and retrieval strategies affect safety and efficiency.

- **MPC** [Falcone et al., 2007]: Classical optimization-based planner solving finite-horizon control under kinematic and safety constraints. We use a 5-step horizon with quadratic cost on acceleration and jerk.
- **MCTS** [Lenz et al., 2016]: Sampling-based planning via tree search and rollout evaluation. We use 1000 simulations per decision with UCB exploration.
- **DriveSpeak** [Cui et al., 2024]: LLM-based method mapping natural-language scene descriptions directly to driving actions. Uses GPT-4o with scene prompts at each timestep.
- **PDDL** [Guan et al., 2023, Liu et al., 2023]: the LLM writes the domain's symbolic preconditions and effects once offline; at runtime a classical STRIPS solver (Pyperplan [Alkhazraji et al., 2020]) is invoked in process at each decision step with no LLM call. Its runtime profile is in Appendix D.3.
- **LLaDA** [Li et al., 2024a]: LLM interpretation of region-specific traffic rules from driver handbooks. Uses pre-extracted rule sets with LLM-based rule matching.
- **DriReg** [Cai et al., 2024]: RAG-based traffic regulation retrieval with LLM reasoning for action legality. Retrieves top-5 regulations per query using embedding similarity.
- **SymAgent** [Liu et al., 2025]: Neural-symbolic agent integrating LLMs with structured knowledge graphs. Uses the same Neo4j backend as REINS for fair comparison.
- **Plan-R1** [Tang et al., 2025]: Two-stage learning-based planner with rule-shaped rewards and reinforcement learning. Trained on the same 50k episodes as REINS's world model.

All baselines use identical observation spaces, action spaces, episodes, and evaluation protocols. For LLM-based methods, we use GPT-4o (2024-08-06) with temperature 0 for deterministic outputs. Hyperparameters for each baseline follow their original papers or are tuned on a held-out validation set. The conditions that are shared with REINS and those that each rule-based baseline keeps by design, together with a matched re-run and a common-reference compliance audit, are documented in Appendix D.2.

C Additional Results

C.1 Training Convergence

We train the world dynamics model for 300 epochs using the Adam optimizer with a learning rate of 2×10^{-4} and batch size 64. Each variable-specific dynamics function f_i is implemented as a two-layer Transformer with four attention heads. Training is conducted on 50k episodes collected

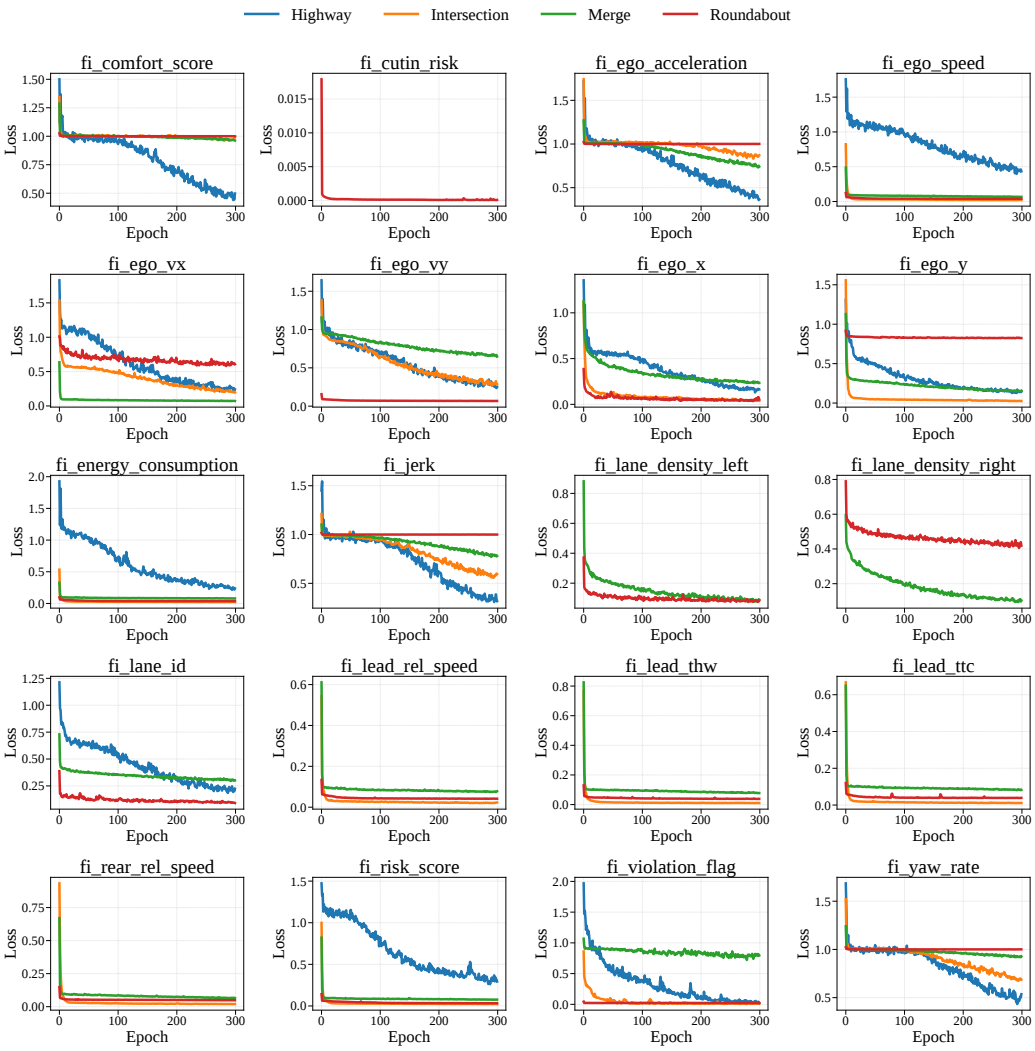


Figure 7: Training loss convergence for the World Dynamics Learning. Each subplot shows one variable's dynamics learning across four scenarios (Highway, Intersection, Merge, Roundabout). All variables converge within 150 epochs, with scenario-dependent convergence rates reflecting inherent complexity differences.

from four scenarios (Highway, Intersection, Merge, and Roundabout) using NVIDIA A100 GPUs. The model learns 20 variable-specific dynamics functions, including ego-state variables (`ego_speed`, `ego_acceleration`, `ego_vx`, `ego_vy`, `ego_x`, `ego_y`, `yaw_rate`, `jerk`, `lane_id`), traffic-related variables (`lead_ttc`, `lead_thw`, `lead_rel_speed`, `rear_rel_speed`, `lane_density_left`, `lane_density_right`, `cutin_risk`), and outcome variables (`risk_score`, `comfort_score`, `violation_flag`, `energy_consumption`). Figure 7 shows the training loss curves for all dynamics functions across the four scenarios.

The results suggest that the learned world dynamics are scenario-adaptive not only in difficulty but also in structural composition: different scenarios activate and learn different subsets of variable-specific dynamics functions. Interaction-heavy variables such as `cutin_risk`, `lane_density_left/right`, and `rear_rel_speed` are prominent in Merge and Intersection, where lane changes and merging behaviors are frequent, but are weakly activated or effectively absent in Highway, where traffic flow is comparatively stable. Conversely, lateral-motion variables (e.g., `ego_vy`, `yaw_rate`) become substantially more relevant in Roundabout due to persistent curvature, while contributing marginally in straight-road scenarios. This indicates that the model does

not impose a fixed global mechanism set across environments, but instead learns scenario-specific mechanism structures, supporting structural adaptability to heterogeneous traffic contexts.

Beyond these structural differences, the loss curves exhibit consistent convergence patterns. Most variables converge within 100–150 epochs. Simple kinematic variables (e.g., `ego_x`, `ego_y`, `lead_thw`) converge within 50 epochs, whereas interaction-intensive outcome variables (e.g., `risk_score`, `violation_flag`) require longer training. Highway achieves the lowest final loss due to simpler lane-following dynamics. Roundabout exhibits higher loss for lateral variables because of curved trajectories, while Intersection shows higher variance in `violation_flag` due to diverse right-of-way interactions. Safety-related variables (e.g., `lead_ttc`, `cutin_risk`) converge rapidly in early training when relevant interactions are present, indicating strong dependency signals. In terms of final loss magnitude, continuous variables (e.g., `ego_speed`, `ego_acceleration`) achieve losses below 0.5, whereas discrete or binary variables (e.g., `violation_flag`, `lane_id`) stabilize at higher loss due to their classification nature.

C.2 Trajectory Accuracy

We evaluate trajectory accuracy using Average Displacement Error (ADE), Final Displacement Error (FDE), and cumulative episode reward (Reward) across four driving scenarios: Highway, Merge, Roundabout, and Intersection. ADE and FDE measure the mean and final Euclidean distance between predicted and executed trajectories, respectively, while Reward reflects overall driving efficiency and safety. All methods are evaluated under identical environment settings with the same observation space, action space, and rollout horizon.

Table 6 reports the mean performance over all evaluation episodes. REINS consistently achieves the best performance across all scenarios in terms of both trajectory accuracy and cumulative reward. In Highway, REINS reduces ADE and FDE to 15.9 and 20.5, respectively, outperforming the strongest baseline (SymAgent) by 12.2% in ADE and 16.3% in FDE, while also achieving the highest reward. Similar improvements are observed in Merge and Roundabout, where REINS yields substantial gains in both short-horizon and long-horizon prediction accuracy, indicating more stable multi-step trajectory generation.

The performance gap becomes more pronounced in complex interaction scenarios. In Intersection, REINS achieves an ADE of 5.6 and FDE of 9.7, representing relative reductions of 30.0% and 13.4% compared to SymAgent, and 41.7% (ADE) and 28.7% (FDE) compared to MPC. This suggests that integrating rule-guided action selection with world-model-based forward rollout is particularly effective when decision-making requires anticipating multi-agent interactions and right-of-way constraints.

Across all scenarios, higher trajectory accuracy strongly correlates with higher episode reward, confirming that more accurate future state prediction leads to safer and more efficient planning. Methods relying primarily on language-based reasoning (LLaDA) exhibit the largest errors, especially in high-speed or curved-road settings, highlighting the limitations of planning without explicit dynamics modeling. MPC improves performance by incorporating learned motion predictors, while SymAgent further benefits from symbolic constraints. REINS combines symbolic rule guidance with learned world dynamics, enabling both structured action filtering and predictive verification, which explains its consistent superiority across heterogeneous traffic scenarios.

C.3 Sensitivity to Retrieval Count k and Planning Horizon H

We analyze sensitivity to two runtime hyperparameters: rule retrieval count k (the maximum number of rules returned per scene by the Cypher query) and planning horizon H (the rollout depth used by trajectory verification). Both are evaluated on simulated (Highway, Roundabout) and real-world (highD, rounD) scenarios.

Rule Retrieval Count k. Table 9a reports results for varying k. Small values ($k \leq 3$) miss critical rules, increasing CR by 3.5–4.7 points; large values ($k \geq 15$) introduce redundancy, raising PE by 60–71% at k=20 without safety gains. The optimal k=10 generalizes across domains, with real-world and roundabout scenarios showing higher sensitivity than structured highway settings.

Planning Horizon H. Table 9b reports results for varying H. Short horizons ($H \leq 3$) cause myopic decisions with 2.9–5.1 point CR increase; long horizons raise PE by 35–44% at H=7 and roughly

Scen.	Metric	LLaDA	MPC	SymAgent	REINS
Hig.	ADE↓	26.5	20.1	18.1	**15.9**
	FDE↓	29.5	25.1	24.5	**20.5**
	Reward↑	14.7	21.6	23.1	**26.3**
Mer.	ADE↓	17.6	13.3	12.7	**9.9**
	FDE↓	21.8	20.0	18.5	**15.2**
	Reward↑	25.2	50.7	54.5	**58.4**
Rou.	ADE↓	19.7	16.5	14.3	**10.1**
	FDE↓	23.2	21.7	19.4	**14.9**
	Reward↑	12.3	22.9	23.6	**27.8**
Int.	ADE↓	21.6	9.6	8.0	**5.6**
	FDE↓	25.4	13.6	11.2	**9.7**
	Reward↑	35.3	54.2	56.8	**60.5**

Table 6: Trajectory accuracy comparison (mean values).

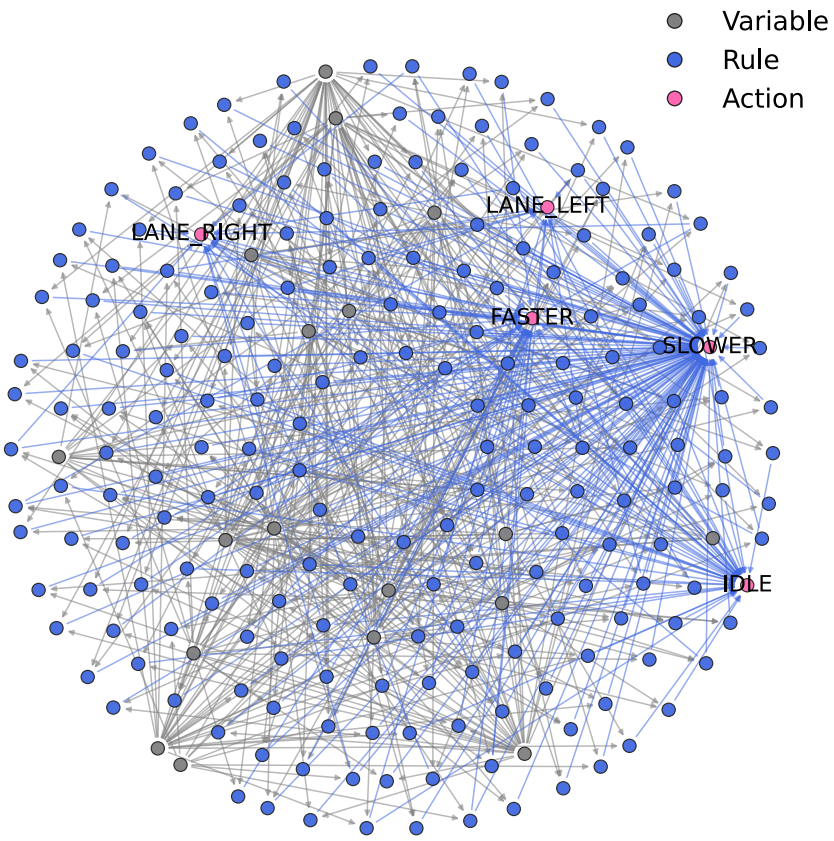


Figure 8: Graph structure of 200 randomly sampled rules.

Dataset	Scen.	Metric	L=2	L=4	L=**6**	L=8	L=10
Sim.	Hig.	Compl.	0.9934	0.9948	**0.9956**	0.9952	0.9947
		RE	-0.42	0.31	**0.52**	0.38	0.15
		CR (%)	9.34	6.89	**5.67**	6.72	7.98
		PE (ms)	**2.21**	2.35	2.50	2.68	2.89
	Rou.	Compl.	0.9898	0.9915	**0.9925**	0.9918	0.9908
		RE	-1.23	-0.78	**-0.54**	-0.72	-0.95
		CR (%)	14.12	10.78	**9.01**	10.34	11.89
		PE (ms)	**3.18**	3.41	3.62	3.89	4.21
Real	highD	Compl.	0.9845	0.9858	**0.9867**	0.9861	0.9852
		RE	-0.35	0.18	**0.38**	0.25	0.08
		CR (%)	10.12	7.34	**6.12**	7.12	8.56
		PE (ms)	**2.38**	2.52	2.62	2.87	3.12
	rounD	Compl.	0.9823	0.9842	**0.9856**	0.9848	0.9834
		RE	-1.12	-0.65	**-0.41**	-0.58	-0.82
		CR (%)	15.23	11.34	**9.45**	10.78	12.45
		PE (ms)	**3.52**	3.76	3.89	4.25	4.63

Table 7: Sensitivity to temporal lag length L (default: 6).

Template	*Simulation*				*Real-world*		**Total**
	Hig.	Mer.	Rou.	Int.	highD	rounD	
`IF-THEN`	42	48	50	55	38	43	276
`AND-THEN`	38	45	47	52	32	37	251
`THRESHOLD`	32	35	36	40	30	32	205
`PRIORITY`	12	20	22	28	15	21	118
`MULTI-ACTION`	10	13	14	17	11	12	77
`WEIGHTED-ACTION`	6	8	9	11	7	10	51
`TEMPORAL-CHAIN`	5	6	7	7	5	15	45
Total	145	175	185	210	138	170	**1,023**

Table 8: Post-deployment distribution of rules by template type across driving scenarios (after 1000-episode self-evolution per scenario).

double it at H=10, with marginal safety benefits. The optimal H=5 balances safety and efficiency, though complex scenarios (Roundabout, rounD) require longer horizons to anticipate multi-agent interactions.

C.4 Sensitivity to Temporal Lag Length

The temporal lag length L determines how many historical timesteps the world dynamics model uses when predicting future states. Table 7 reports results on both simulated and real-world scenarios.

When using small lag lengths ($L \leq 2$), the model fails to capture cumulative temporal patterns such as sustained tailgating or gradual lane drift, leading to substantially higher collision rates. On the simulated Highway scenario, the collision rate increases from 5.67% at $L = 6$ to 9.34%, corresponding to a 65% relative increase. A similar degradation is observed on the real-world highD dataset, where the collision rate rises from 6.12% to 10.12%. These results indicate that short-term observations are insufficient to infer latent interaction states, such as risk accumulation and driver intent, which are critical for safe planning. In contrast, large lag lengths ($L \geq 10$) introduce excessive historical noise and dilute recent dependency signals, reducing the effective signal-to-noise ratio for short-horizon prediction. Consequently, collision rates increase by approximately 40% on simulated scenarios compared to the optimal setting. This suggests that overly long histories may impair the model's ability to focus on immediately relevant dynamics, particularly in rapidly evolving traffic situations.

Overall, $L = 6$ achieves the best and most consistent performance across both simulated and real-world domains, corresponding to approximately 1.2 seconds of historical context at a sampling rate of 5 Hz. This time window appears sufficient to capture short-term interaction dynamics while avoiding

Dataset	Scen.	Metric	k=3	k=5	k=**10**	k=15	k=20
Sim.	Hig.	Compl.	0.9712	0.9856	**0.9956**	0.9948	0.9951
		RE	0.18	0.41	**0.52**	0.35	0.21
		CR (%)	9.21	7.08	**5.67**	6.02	6.48
		PE (ms)	**1.82**	2.13	2.50	3.21	4.15
	Rou.	Compl.	0.9589	0.9778	**0.9925**	0.9912	0.9908
		RE	-1.12	-0.78	**-0.54**	-0.72	-0.89
		CR (%)	13.34	10.45	**9.01**	9.62	10.28
		PE (ms)	**2.67**	3.12	3.62	4.56	5.78
Real	highD	Compl.	0.9689	0.9798	**0.9867**	0.9856	0.9852
		RE	0.05	0.25	**0.38**	0.22	0.09
		CR (%)	10.01	7.63	**6.12**	6.58	7.08
		PE (ms)	**2.05**	2.38	2.62	3.52	4.48
	rounD	Compl.	0.9534	0.9712	**0.9856**	0.9842	0.9836
		RE	-1.02	-0.65	**-0.41**	-0.58	-0.76
		CR (%)	14.12	11.08	**9.45**	10.02	10.75
		PE (ms)	**2.98**	3.48	3.89	5.02	6.28

(a) Rule retrieval count k.

Dataset	Scen.	Metric	H=3	H=**5**	H=7	H=10
Sim.	Hig.	Compl.	0.9948	**0.9956**	0.9953	0.9949
		RE	-0.12	**0.52**	0.48	0.41
		CR (%)	8.56	**5.67**	5.91	5.78
		PE (ms)	**1.87**	2.50	3.42	5.18
	Rou.	Compl.	0.9908	**0.9925**	0.9919	0.9912
		RE	-1.02	**-0.54**	-0.62	-0.71
		CR (%)	12.78	**9.01**	9.62	9.38
		PE (ms)	**2.78**	3.62	4.89	7.34
Real	highD	Compl.	0.9856	**0.9867**	0.9862	0.9854
		RE	-0.08	**0.38**	0.34	0.28
		CR (%)	9.78	**6.12**	6.48	6.32
		PE (ms)	**2.12**	2.62	3.76	5.62
	rounD	Compl.	0.9838	**0.9856**	0.9848	0.9841
		RE	-0.89	**-0.41**	-0.52	-0.62
		CR (%)	14.56	**9.45**	10.06	9.78
		PE (ms)	**3.15**	3.89	5.52	8.18

(b) Planning horizon H.

Table 9: Sensitivity to retrieval count k (left) and planning horizon H (right).

the accumulation of outdated or irrelevant information, thereby supporting stable multi-step rollout and reliable safety verification in downstream planning.

C.5 Self-Evolution Across Scenarios

This appendix reports self-evolution dynamics on the remaining simulated scenarios (Merge, Roundabout, Intersection) and the real-world datasets (highD, rounD), complementing the Highway results in Section 5.4. For each scenario, we deploy REINS for 1000 episodes from the offline-compiled memory $\mathcal{R}_0$ and report $|\mathcal{R}|$, LLM call rate, Compl., and CR at four deployment checkpoints (ep. 250, 500, 750, 1000), with a Frozen $\mathcal{R}_0$ row showing the same 1000-episode window run with acquisition disabled; this row exposes $|\mathcal{R}_0|$ and isolates the contribution of online rule writing from any aggregation effect of running more episodes. The per-scenario $|\mathcal{R}|$ at ep. 1000 matches the post-deployment totals in Table 8.

Three patterns emerge across the five extended scenarios, mirroring the Highway dynamics. First, $|\mathcal{R}|$ exhibits saturating growth: approximately 44% of the rule additions (counted from $|\mathcal{R}_0|$ in the Frozen row to $|\mathcal{R}|$ at ep. 1000) occur in the first 250 episodes, with diminishing returns thereafter, so the LLM is invoked predominantly for previously unseen scene clusters. Second, the LLM call rate decays monotonically in all scenarios, from 68–79% in the first 250-episode window to 12–19% by ep. 1000 (a 4.2–5.5$\times$ reduction); scene complexity raises the absolute size of $\mathcal{R}$ and the residual call rate (highest on Intersection) but not the shape of the decay. Third, the Frozen $\mathcal{R}_0$ control consistently underperforms the evolving memory at ep. 1000 (mean Compl. gap $\approx$0.017, mean CR gap $\approx$4.2 points), confirming that the gains attributable to online acquisition are not artifacts of statistical aggregation over a longer evaluation window. Intersection benefits most (Compl. $+0.0218$, CR -4.87 points from Frozen to ep. 1000), reflecting larger room for online acquisition where the offline $\mathcal{R}_0$ is most incomplete relative to right-of-way reasoning; the other four scenarios cluster in a tight range (Compl. $+0.0154$ to $+0.0164$, CR -3.79 to -4.40 points). highD mirrors Highway and rounD mirrors Roundabout in both $|\mathcal{R}|$ growth shape and Compl./CR convergence, indicating that the harness operates on factored variable signatures rather than scenario-specific surface features. Cross-domain transfer of the compiled memory itself is studied separately in Appendix D.7.

C.6 Case Study

Figure 9 compares trajectories of REINS, Plan-R1, LLaDA, and MPC on two scenarios, with activated rules and actions shown at each timestep. Rule definitions are provided in Appendix C.7.2.

Scenario	Checkpoint	$\lvert\mathcal{R}\rvert$	LLM Call (%)	Compl. (↑)	CR (%) (↓)
Merge	ep. 250	136	71.36	0.9858	8.20
	ep. 500	153	49.12	0.9893	7.35
	ep. 750	167	26.48	0.9912	7.05
	ep. 1000 (final)	**175**	**13.92**	**0.9924**	**6.86**
	Frozen $\mathcal{R}_0$ (1000-ep eval)	105	—	0.9770	10.65
Roundabout	ep. 250	144	73.84	0.9852	10.55
	ep. 500	162	52.20	0.9888	9.85
	ep. 750	176	28.36	0.9910	9.40
	ep. 1000 (final)	**185**	**15.24**	**0.9925**	**9.01**
	Frozen $\mathcal{R}_0$ (1000-ep eval)	112	—	0.9770	13.40
Intersection	ep. 250	164	78.52	0.9742	10.85
	ep. 500	184	57.64	0.9789	9.95
	ep. 750	200	33.08	0.9817	9.45
	ep. 1000 (final)	**210**	**18.76**	**0.9838**	**9.08**
	Frozen $\mathcal{R}_0$ (1000-ep eval)	127	—	0.9620	13.95
highD	ep. 250	107	68.24	0.9803	7.45
	ep. 500	121	47.56	0.9838	6.65
	ep. 750	131	24.12	0.9856	6.32
	ep. 1000 (final)	**138**	**12.36**	**0.9867**	**6.12**
	Frozen $\mathcal{R}_0$ (1000-ep eval)	84	—	0.9712	9.95
rounD	ep. 250	132	74.68	0.9785	10.95
	ep. 500	149	53.40	0.9821	10.20
	ep. 750	162	29.72	0.9842	9.78
	ep. 1000 (final)	**170**	**16.08**	**0.9856**	**9.45**
	Frozen $\mathcal{R}_0$ (1000-ep eval)	103	—	0.9692	13.85

Table 10: Self-evolution dynamics across the remaining evaluation scenarios. Same metrics and checkpoints as Table 3; the Frozen row runs the same 1000-episode window with acquisition disabled and exposes $|\mathcal{R}_0|$.

Simulated Highway. REINS activates R03 and R07 at t=0 to monitor the lead vehicle and adjacent lanes, triggers R12 at t=0.5s when a safe gap appears, executes a lane change at t=1.0s, and accelerates via R15 at t=1.5s, producing a smooth action sequence (IDLE→IDLE→RIGHT→FASTER). Plan-R1 applies all 15 rules simultaneously, causing conflicting constraints and repeated deceleration (SLOWER×3) before a delayed maneuver.

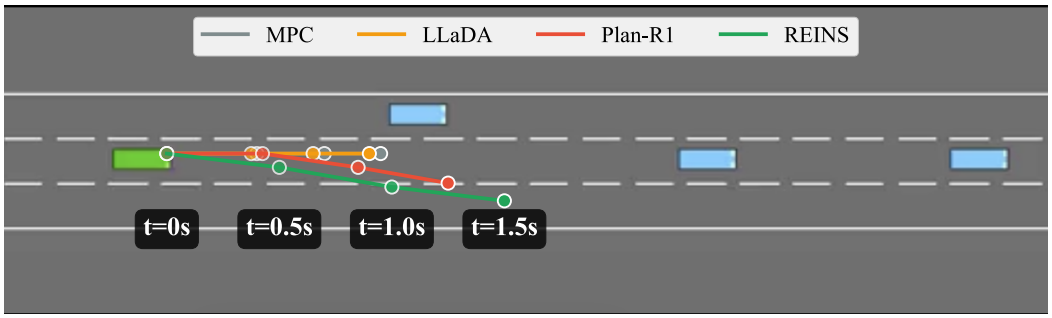
Real-World highD. In the cut-in scenario, REINS detects lateral intrusion at t=0 via R05, performs preemptive deceleration at t=0.5s using R07, and maintains safe distance until t=1.5s (IDLE→SLOWER→SLOWER→IDLE). Plan-R1 responds later with emergency braking at t=1.0s due to uniform rule execution.

C.7 Compiled Rule Database

C.7.1 Rule Database Statistics

Table 8 shows the distribution of variables by macro-group and rules by template type in REINS's Neo4j graph database. The post-deployment database contains 1,023 rules covering six driving scenarios (four simulated: Highway, Merge, Roundabout, Intersection; two real-world: highD, rounD). IF-THEN and AND-THEN templates dominate due to the prevalence of conditional and multi-condition reasoning in driving. Intersection scenarios contain the most rules due to complex right-of-way interactions, while Highway scenarios require fewer given simpler lane-following dynamics. This scale is comparable to industrial rule-based systems while remaining tractable for $O(\log n)$ graph database retrieval (<1 ms per query).

Figure 8 visualizes the graph structure of 200 randomly sampled rules from the database. Gray nodes are variables, blue nodes are rules, and pink nodes are the five discrete actions. Edge colors match source node colors: gray edges represent Variable→Rule, while blue edges represent Rule→Action.



Time	t=0	t=0.5s	t=1.0s	t=1.5s
Methods	Rules | Action	Rules | Action	Rules | Action	Rules | Action
REINS	R03,R07 | IDLE	R07,R12 | IDLE	R12 | RIGHT	R15 | FASTER
Plan-R1	All(15) | SLOWER	All(15) | SLOWER	All(15) | SLOWER	All(15) | RIGHT
LLaDA	"safe" | SLOWER	"safe" | SLOWER	"safe" | IDLE	"safe" | IDLE
MPC	— | IDLE	— | SLOWER	— | SLOWER	— | SLOWER

(a) Simulation: Highway



Time	t=0	t=0.5s	t=1.0s	t=1.5s
Methods	Rules | Action	Rules | Action	Rules | Action	Rules | Action
REINS	R03,R05 | IDLE	R05,R07 | SLOWER	R07 | SLOWER	R03 | IDLE
Plan-R1	All(15) | IDLE	All(15) | IDLE	All(15) | BRAKE	All(15) | BRAKE
LLaDA	"safe" | IDLE	"safe" | IDLE	"safe" | SLOWER	"safe" | SLOWER
MPC	— | IDLE	— | IDLE	— | BRAKE	— | BRAKE

(b) Real-world: highD

Figure 9: Trajectory comparison across four methods on (a) simulated highway and (b) real-world highD scenarios.

ID	**Guard Condition** $\mathcal{P}(r)$	**Action** a_r	**Template**
R01	TTC < 2.0s	emergency brake	`THRESHOLD`
R02	$v_{ego} > v_{limit} + 5$km/h	DEC to limit	`THRESHOLD`
R03	THW < 1.8s $\vee$ DHW < 25m	DEC / maintain gap	`IF-THEN`
R04	TTC < 3.5s $\wedge$ DHW < 40m	DEC (moderate)	`AND-THEN`
R05	$v_{adj}^{lat} > 0.3$m/s $\wedge\ \Delta y < 2$m	DEC (preemptive)	`AND-THEN`
R06	visibility < 100m	reduce speed 20%	`THRESHOLD`
R07	$\text{TTC}_{adj} < 4$s $\vee$ cut-in	increase headway	`IF-THEN`
R08	lane_end < 200m	prepare merge	`TEMPORAL-CHAIN`
R09	at_yield $\wedge$ traffic_present	yield / stop	`PRIORITY`
R10	pedestrian < 30m	DEC / prepare stop	`MULTI-ACTION`
R11	$v_{lead} < v_{ego} - 10$km/h $\wedge\ \text{gap}_{adj} > 35$m	consider lane change	`AND-THEN`
R12	$\text{gap}_{target} > 30$m $\wedge\ \text{TTC}_{target} > 5$s	execute lane change	`AND-THEN`
R13	in_lane_change $\wedge\ \text{TTC}_{new} < 3$s	abort lane change	`AND-THEN`
R14	merge_complete $\wedge$ DHW < 20m	DEC post-merge	`AND-THEN`
R15	post_merge $\wedge\ v_{ego} < v_{flow} - 5$km/h	ACC to flow	`AND-THEN`

Table 11: Representative rules from the compiled rule database.

Several structural patterns emerge from this visualization: (1) **Variable sharing**: A small set of variables (e.g., TTC, THW, speed) triggers a large number of rules, indicating that safety-critical variables serve as common preconditions across diverse driving scenarios. (2) **Action concentration**: The majority of rules converge to `SLOWER` and `IDLE`, which exhibit the highest in-degree among all five actions. This reflects the safety-first design principle where conservative actions are recommended under uncertain or risky conditions. In contrast, `LANE_LEFT`, `LANE_RIGHT`, and `FASTER` receive fewer rule connections, as these actions require more stringent preconditions. (3) **Rule diversity**: Despite action concentration, rules maintain diverse trigger conditions from different variables, enabling fine-grained context-aware activation rather than uniform rule application. These patterns support REINS's retrieval strategy: variable-indexed queries efficiently activate relevant rule subsets, while action aggregation via noisy-OR naturally prioritizes safety-critical recommendations.

C.7.2 Representative Rules

Table 11 lists 15 representative rules from REINS's compiled rule database. The full post-deployment Neo4j database contains 1,023 rules across six driving scenarios, from which context-relevant subsets are retrieved at runtime via parameterized Cypher queries. These rules are automatically compiled from LLM-generated linguistic descriptions and calibrated using the threshold calibration mechanism described in Section 4.1.2. For fair comparison, Plan-R1 applies the same 15 representative rules uniformly at each step, while REINS selectively activates 2–3 context-relevant rules via the retrieval mechanism.

The 15 rules cover six of the seven template types defined in Definition 3.5: **(1)** `THRESHOLD` rules (R01, R02, R06) enforce numeric boundaries on physical variables such as TTC and speed; **(2)** `IF-THEN` rules (R03, R07) implement basic conditional logic with disjunctive guards; **(3)** `AND-THEN` rules (R04, R05, R11–R15) require multiple conditions to be satisfied simultaneously; **(4)** `PRIORITY` rules (R09) establish decision hierarchies for right-of-way scenarios; **(5)** `MULTI-ACTION` rules (R10) trigger compound actions under critical conditions; **(6)** `TEMPORAL-CHAIN` rules (R08) encode time-sensitive constraints for upcoming events. The complete definitions of all seven rule template types are provided in Appendix A.2, and the conflict resolution mechanism is described in Appendix A.3.

D Fairness Protocol, Robustness, Transfer, and Scalability

This appendix collects the studies that scope the harness claim and stress the main results: the domain-specific versus domain-agnostic components of the harness (D.1), the baseline fairness protocol with a matched re-run and a common-reference compliance audit (D.2), the runtime profile of the PDDL baseline (D.3), robustness to the LLM backbone and to prompt variation (D.4), what rollout verification filters (D.5), reactive replay and observation noise (D.6), cross-domain memory transfer (D.7), the latency of the online-miss cycle and its scheduling (D.8), and retrieval at deployment scale (D.9). Unless stated otherwise, all studies use the Highway scenario, five seeds, and the protocol of Section 5.1.

D.1 Scope of the Harness Claim

The three couplings are presented as a harness pattern, while the implementation is specialized to driving. Table 12 classifies each component of REINS and sketches the instantiation for a different trajectory-planning domain, robot-arm manipulation, whose action space is continuous and high-dimensional and whose safety notions (contact force, grasp stability) differ from driving. Two components are domain schemas that any instantiation must author; the remaining five transfer unchanged. The safety notion enters only through the variable schema and the terms of the objective J: calibrated predicates over contact-force variables are structurally identical to TTC guards, and the calibration procedure and the verification protocol do not change. The harness contribution lives in the five unchanged components; the two redesigned ones constitute the domain description that any planner for that domain must author anyway.

D.2 Baseline Fairness Protocol

Shared and unshared conditions. PDDL, DriReg, and LLaDA were conceived for symbolic planning domains, regulatory text, and driver handbooks respectively, none with a factored driving

Component	Driving (this paper)	Manipulation (sketch)	Transfer
Variable space and groups	44 variables in Env, Traffic, Ego, Action, Outcome	joint angles and torques, end-effector pose, contact forces, object poses	redesign
Action space	5 driving meta-actions	skill primitives (reach, grasp, lift, place, retreat)	redesign
Predicate grammar and thresholds	TTC and THW guards	force-margin, slip, and grasp-stability guards	grammar unchanged, thresholds recalibrated
Dynamics learning	LLM prior, group-weighted ridge, per-variable Transformer	same, retrained on domain data	unchanged
Verification	collision and violation terms in J	contact-force and stability terms in J	protocol unchanged
Indexed memory and retrieval	variable-keyed graph schema	same	unchanged
Acquisition loop	miss detection, proposal, verification, write-back	same	unchanged

Table 12: Domain-specific and domain-agnostic components of the harness.

state, so each was re-implemented faithfully to its original design for closed-loop highway-env. Table 13 lists what is shared with REINS and what each method keeps by design. The shared conditions fix the observation space, action space, episodes, LLM backbone, and decoding; the unshared ones leave each method its own knowledge representation and thresholds, which leaves the rule-generation procedure as a confound. Three instruments below isolate the harness regardless.

Condition	PDDL	DriReg	LLaDA
Observation space, action space, episodes	shared	shared	shared
GPT-4o backbone (2024-08-06), temperature 0 for rule generation	shared	shared	shared
44-variable vocabulary (Appendix A.1) in rule-generation prompts	own vocabulary	own vocabulary	n/a (linguistic rules)
Threshold calibration	uncalibrated	uncalibrated	n/a (no thresholds)

Table 13: Fairness conditions for the rule-based baselines as published. "Own vocabulary" and "uncalibrated" follow each method's original design.

Matched re-run. We regenerated the PDDL and DriReg rules over the same 44-variable vocabulary, passed their thresholds through REINS's calibration pipeline, and re-evaluated (Table 14). Input normalization improves both baselines by 3–5 CR points and explains a minority of the gap; the majority survives it.

CR (%)	PDDL	PDDL matched	DriReg	DriReg matched	REINS
Highway	25.93	21.37	21.76	18.92	**5.67**
Intersection	32.12	27.64	29.89	25.81	**9.08**

Table 14: Matched re-run: PDDL and DriReg rules regenerated over REINS's variable vocabulary with calibrated thresholds.

Bracketing with the ablations. Two ablations from Table 2 bracket the remaining gap with the paper's own numbers. Hand-crafted rules inside the full harness still reach CR 11.23% on Highway, better than every matched no-harness variant in Table 14 and close to the best baseline Plan-R1 at 12.83%. Conversely, applying REINS's rules uniformly, the way the baselines use rules, raises CR by 22–43% and PE by 34–58%. Worse inputs with the harness keep most of the gain; the same inputs without the harness lose it.

Common-reference compliance. Compliance in Table 1 is measured against each method's own rule set. Table 15 re-scores every method against REINS's compiled rule set as a common reference.

Compliance is a property of the executed trajectory rather than of the planner: at each step, the reference rules whose guards hold in the logged state become active, and a violation is counted when the logged behavior fails an active rule (Appendix B.2). Each method therefore plans exactly as before and its trajectories are audited afterward, which is also how the four methods without native rule sets receive values. Because the reference set equals REINS's planning set, REINS's own value is unchanged by construction; the informative content is that the baseline ordering and the gaps persist.

Method	Own rules		Common reference (REINS set)	
	Highway	Intersection	Highway	Intersection
MPC	n/a	n/a	0.8712	0.8423
MCTS	n/a	n/a	0.8831	0.8517
DriveSpeak	n/a	n/a	0.8246	0.7912
LLaDA	n/a	n/a	0.8412	0.8104
PDDL	0.9123	0.8789	0.8867	0.8621
DriReg	0.9452	0.9153	0.9083	0.8874
SymAgent	0.9561	0.9274	0.9312	0.9067
Plan-R1	0.9717	0.9553	0.9498	0.9231
REINS	**0.9956**	**0.9838**	**0.9956**	**0.9838**

Table 15: Compliance against each method's own rule set and against REINS's compiled rule set as a common reference.

D.3 Runtime Profile of the PDDL Baseline

The PDDL baseline follows the pipeline in which the LLM writes the domain's preconditions and effects offline and a classical solver executes at runtime [Guan et al., 2023, Liu et al., 2023]. The solver is Pyperplan [Alkhazraji et al., 2020], a Python STRIPS planner invoked in process at each decision step; there is no per-step LLM call. Its 4.95 ms per step on Highway (Table 1) decomposes into grounding 3.71 ms, search 0.94 ms, and plan extraction 0.30 ms. The solver re-instantiates every precondition against the current state at every step with no indexing, so grounding takes roughly three quarters of the budget. REINS avoids exactly this cost: variable-indexed retrieval touches 2–3 rules in under 1 ms, followed by a fixed-size rollout at $H{=}5$. This is the architectural payoff of consolidation with indexing.

D.4 Robustness to the LLM Backbone and to Prompt Variation

The LLM produces the dependency priors, the rule structures, and the Cypher queries, so we regenerate the whole offline pipeline with two further backbones and with three paraphrases of the dependency and rule-extraction prompts (Table 16). Rule structure varies mildly across backbones and paraphrases, while the calibrated thresholds and the final behavior barely move. The LLM picks variables, templates, and actions; calibration re-anchors every numeric threshold to data, so backbone differences wash out at the threshold level. Malformed or unsafe proposals never reach memory, because rollout verification rejects them regardless of which backbone or prompt produced them.

Variation (Highway)	Rule overlap with GPT-4o	Threshold shift after calibration	Compl.	CR (%)
GPT-4o (paper)	n/a	n/a	0.9956	5.67
Claude Sonnet 4.5	87.3%	3.2%	0.9948	5.89
Qwen2.5-72B-Instruct	81.6%	3.8%	0.9931	6.24
Three prompt paraphrases, GPT-4o	91.2–94.6%	—	—	5.58–5.91

Table 16: Robustness of the compiled rules and of closed-loop behavior to the LLM backbone and to prompt paraphrases.

D.5 What Rollout Verification Filters

On the 1000-episode Highway deployment of Section 5.4, applying the windowed call rates of Table 3 to the 100,000 deployment steps yields roughly 39,000 logged proposals (an arithmetic

consequence of the call rates, not an independent count). Of these, 2.1% failed schema validation, 18.0% were rejected by rollout verification, and 79.9% were accepted. Acceptances overwhelmingly hash-collide with rules already in memory (Eq. 8) and only update confidences; they deduplicate to the 57 unique rules that grow the memory from 88 to 145. A manual audit of the 57 unique rules judged 54 consistent with domain guidelines; the three flagged rules were over-specific and never activated again. Episodes in which at least one acquired rule fired show no elevated collision rate against episodes in which none fired (5.4% against 5.8%).

Three properties of the design bound what a plausible but incorrect proposal can do. Verification errs toward rejection: a rule is accepted only if every calibrated predicate holds along the predicted horizon (Eq. 7), so dynamics error costs availability rather than safety in the common case. Accepted rules never bypass the runtime loop: every executed action is re-verified by rollout at decision time (Section 4.2.1), so a wrongly accepted rule must also survive the per-step check before it can affect control. Finally, the residual dependence on dynamics fidelity is stated in Section 6.

D.6 Reactive Replay and Observation Noise

Reactive replay. In the four simulated scenarios the surrounding vehicles already react to the ego through IDM car-following with MOBIL lane changes. The real-world rows keep highD and rounD as recorded so that they reflect human behavior rather than a simulator's. To measure the effect of this choice, a reactive-replay variant on highD switches recorded vehicles within 50 m of the ego to IDM once interaction occurs, following the reactive closed-loop protocol of nuPlan [Caesar et al., 2021]. REINS CR moves from 6.12% to 6.94% while Plan-R1 moves from 12.78% to 15.36%, so the gap to the strongest baseline widens.

Observation noise. We corrupt the observed variables with zero-mean Gaussian noise scaled to each variable's standard deviation and, separately, drop 5% of frames (Table 17). Calibrated thresholds carry a data-derived margin and every action is vetted by a multi-step rollout rather than by a single reading, so REINS degrades more slowly than the strongest baseline at every level, with Compl. staying at or above 0.987. Full-stack perception errors (occlusion, tracking identity switches, map errors) remain out of scope.

CR (%) on Highway	no noise	2.5% noise	5% noise	10% noise	5% frame drops
REINS	5.67	5.92	6.51	8.14	6.88
Plan-R1	12.83	13.71	15.42	19.87	16.50

Table 17: Collision rate under σ-scaled Gaussian observation noise and under frame drops.

D.7 Cross-Domain Memory Transfer

We test how the memory behaves when the offline data are domain-shifted, using the checkpoint protocol of Table 3. Setting A compiles $\mathcal{R}_0$ solely from highD and rounD (whose memories total 187 rules before merging and content-hash deduplication) and deploys on highway-env; Setting B compiles it solely from the four simulated scenarios (432 rules before deduplication) and deploys on highD and rounD. Table 18 reports the ep. 250 and ep. 1000 endpoints of each setting, together with a frozen-transfer control on Highway; the native reference rows come from Table 3 and Table 10.

A domain-shifted memory starts degraded: at ep. 250 the call rate runs 86.2% on Highway against the native 70.0% of Table 3, and CR runs several points above the native value at the same checkpoint (11.84% against 7.05%). Acquisition recovers most of the gap within 1000 episodes: final CR lands within 0.5–1.1 points of native and Compl. within 0.003–0.006. On Highway, freezing the transferred memory prevents recovery: at ep. 1000 the frozen transfer stays at CR 11.20% against 6.21% with acquisition enabled, so the recovery comes from verification-gated acquisition rather than from the transferred rules themselves, mirroring the native pattern of Table 3, where frozen $\mathcal{R}_0$ ends at CR 9.63% against the evolved 5.67%. Sim-to-real shows slightly smaller residual gaps and lower final call rates than real-to-sim; one plausible factor is its larger source memory (432 against 187 rules before deduplication).

The initial call rate of 86.2% also shows that most of the source-domain memory is not functionally reusable zero-shot in a new but still driving context. This is consistent with the safety-oriented design:

Scenario	Checkpoint	$\lvert\mathcal{R}\rvert$	LLM Call (%)	Compl.	CR (%)	PE (ms)
Setting A: $\mathcal{R}_0$ from highD and rounD, deployed on highway-env						
Highway	ep. 250	265	86.2	0.9612	11.84	2.71
	ep. 1000	349	14.9	0.9914	6.21	2.54
	ep. 1000, acquisition frozen	—	—	—	11.20	—
	native, ep. 1000	145	12.81	0.9956	5.67	2.50
Intersection	ep. 250	287	91.3	0.9487	15.62	4.87
	ep. 1000	391	17.2	0.9782	10.13	4.66
	native, ep. 1000	210	12.81	0.9838	9.08	4.58
Setting B: $\mathcal{R}_0$ from the four simulated scenarios, deployed on highD and rounD						
highD	ep. 250	471	71.4	0.9702	9.87	2.79
	ep. 1000	508	13.6	0.9838	6.79	2.68
	native, ep. 1000	138	12.81	0.9867	6.12	2.62
rounD	ep. 250	478	79.8	0.9645	14.21	4.12
	ep. 1000	517	15.1	0.9821	10.24	3.97
	native, ep. 1000	170	12.81	0.9856	9.45	3.89

Table 18: Cross-domain memory transfer under the checkpoint protocol of Table 3. The frozen-transfer row disables acquisition after transfer; only its CR was recorded.

thresholds are calibrated to the source-domain distribution, so under shift the harness re-invokes and re-verifies rather than reusing rules whose calibration may no longer hold. We therefore do not frame these results as evidence of strong zero-shot generalization; the limitation is stated in Section 6.

D.8 Latency of the Online-Miss Cycle and Its Scheduling

What PE counts. For every method, PE measures the synchronous action-selection computation per step. In the evaluated closed-loop setting, a miss completes proposal, verification, and write-back within the step while simulation time is paused, so the reported PE excludes the acquisition pipeline, namely the LLM call together with the subsequent parsing, calibration, verification, and write-back. Table 19 profiles each stage.

Stage	Mean	p95	Deployment scheduling
Fallback action for the current step	2.6 ms	3.9 ms	control path, within budget
Prompt assembly and state serialization	0.9 ms	1.4 ms	background
LLM rule proposal (GPT-4o API)	1.58 s	2.94 s	background
Parsing and schema validation	2.1 ms	3.0 ms	background
Threshold calibration on the sliding window	1.3 ms	1.9 ms	background
Rollout verification at $H=5$	1.6 ms	2.2 ms	background
Neo4j insert and index update	4.7 ms	6.8 ms	background
Full cycle, miss to usable rule	1.59 s	2.96 s	background

Table 19: Latency of the online-miss cycle on Highway. The fallback row sits on the control path and is not part of the full-cycle latency; the acquisition stages below it sum to the full-cycle row.

Deployment scheduling. The cycle takes 1.59 s mean, dominated by the LLM call; the acquisition stages other than the LLM call total about 11 ms. In deployment the cycle moves off the critical path by construction, because the planner has a defined behavior without the acquired rule: the best action over the retrieved rules or, with empty retrieval, the safe action that maximizes minimum lookahead TTC (Eq. 24) together with a human-takeover alert. The current step executes that fallback while the cycle completes in the background, and the acquired rule serves later encounters. At 5 Hz the acquisition window spans about eight control cycles, during which the planner behaves as a verified conservative planner. Figure 10 contrasts the evaluated schedule with the deployment schedule.

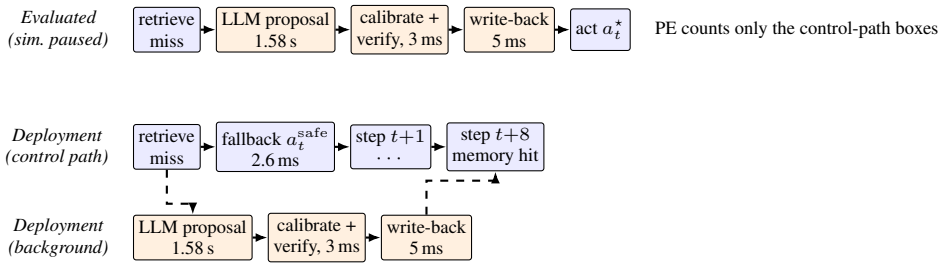


Figure 10: Dataflow of the online-miss cycle. Top: in the evaluated closed-loop setting the cycle completes within the step while simulation time is paused, and PE counts only the synchronous control-path computation. Bottom: in deployment the current step executes the verified fallback while the cycle runs in the background (about eight control cycles at 5 Hz), and the acquired rule serves later encounters.

D.9 Retrieval and Aggregation at Deployment Scale

As memory grows, retrieval and multi-rule aggregation must stay within the control window. The stress test runs on Highway and varies only the database size, inflating it with synthetic rules generated as variable and threshold perturbations of compiled rules that preserve the schema and the trigger-degree distribution (Table 20). Retrieval stays sub-millisecond and grows slowly with memory size, from 0.41 ms at the paper's 1,023 rules to 0.79 ms at one hundred thousand, consistent with indexed lookup rather than linear scanning. Conflict aggregation does not grow with the database: retrieval returns at most $k{=}10$ rules regardless of memory size, so noisy-OR aggregation (Eq. 5) and the arbitration of Appendix A.3 cost $O(k)$. On the online planning path, the database-size-dependent cost therefore concentrates in retrieval; index maintenance and write costs sit on the acquisition path, off the control loop.

Memory size	**1,023 (paper)**	**10,000**	**50,000**	**100,000**
Cypher retrieval (ms)	0.41	0.55	0.68	0.79
End-to-end PE (ms)	2.50	2.64	2.77	2.88

Table 20: Retrieval latency and end-to-end PE on Highway as the rule memory is inflated with schema-preserving synthetic rules.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 list the paper's contributions (extending the agent-harness paradigm to real-time trajectory planning, designing REINS to realize three couplings, and evaluating on simulated and real-world benchmarks); the headline metrics (98%+ compliance, nearly 50% collision reduction, 1.9–4.6 ms planning latency) match the results in Section 5, and both the abstract and the introduction scope the claims to the tactical decision layer, with the limitations stated in Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated Limitations section (Section 6) covers the abstraction gap of the evaluated decision layer, the discrete action interface, the non-deployment-grade absolute collision rates, the replay protocol of the real-world datasets, the correlational nature of the learned dynamics, the limited zero-shot cross-domain reusability of the compiled memory, and the stationarity assumption underlying the monotone miss-rate property; the supporting measurements are in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theorems requiring proofs; the definitions in Section 3 are operational specifications rather than theoretical claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 fully describes REINS's four modules with all equations; experimental setup is detailed in Section 5 and Appendix B.1; datasets (highD, rounD, highway-env) are publicly available and cited; the code is released at the URL in the abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code with reproducibility instructions (README) is released at `https://github.com/ZhihongCui/REINS` (cited in the abstract); all evaluation datasets (highD, rounD, highway-env) are publicly available research datasets.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All hyperparameters (confidence threshold, retrieval count k, planning horizon H, discount γ, objective weights), training configuration (50k episodes, 5 seeds, optimizer), and evaluation protocol are documented in Appendix B.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Results are averaged over 5 random seeds (0, 42, 123, 456, 789; Section 5), but the tables report only the means; standard deviations and confidence intervals across seeds are not reported.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Training hardware (4×A100 40 GB, ∼8 hours), inference hardware (RTX 3090, ∼4 GB GPU memory), and per-decision planning latency (1.9–4.6 ms) are reported in Appendix B.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. We use only public driving datasets, do not involve human subjects, and the safety-critical nature of trajectory planning is addressed via the human-takeover alert mechanism in Section 4.2.2.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Positive impacts (safety improvements via collision reduction and traffic-rule compliance) are discussed in the abstract and quantified in our experiments. Negative impacts—edge-case brittleness when scenarios fall outside the rule memory, and biases potentially inherited from LLM-generated rules—are mitigated by REINS's rollout verification, rejection fallback, and human-takeover alert (Section 4.2.2); real-world deployment would still require additional validation and human oversight beyond the scope of this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release new high-risk pre-trained models, image generators, or scraped datasets. The released code is a research framework that wraps existing public datasets and a frozen pre-trained LLM (whose weights are never updated).

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (highD, rounD, highway-env) and baseline methods (MPC, MCTS, DriveSpeak, LLaDA, PDDL, DriReg, SymAgent, Plan-R1) are properly cited; their licenses follow the original publications and are publicly available research assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released REINS code is the only new asset; documentation and reproducibility instructions (README) are included alongside the code at the URL provided in the abstract.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or research with human subjects; all evaluation uses public driving datasets and simulators.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects and therefore does not require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are central to REINS: they (i) infer variable dependencies as priors for the world dynamics model (Section 4.1.1), (ii) generate executable rule structures from dependency edges (Section 4.1.2), (iii) generate parameterized Cypher queries for runtime retrieval (Section 4.2.1), and (iv) propose new rules on memory misses (Section 4.2.2). The specific LLM backbone (GPT-4o, 2024-08-06) and decoding settings are documented in Appendix B.1; prompt templates are provided in Appendix A.5.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.