AN ATTENTION MECHANISM FOR PROBABILISTIC ZERO-SHOT MONITORING OF FORMAL LANGUAGES

Anonymous authors
Paper under double-blind review

ABSTRACT

The deployment of autonomous systems raises the question of how safety-relevant properties can be assured at runtime. This is a challenging task for vision-based systems, in which safety-relevant quantities must be inferred from raw sensor data; the need for suitable solutions to this problem is reinforced by the increasing use of end-to-end policies, such as VLAs. We seek a monitor that a user can prompt with any property expressed in a formal language (a machine-interpretable specification language) and that predicts its satisfaction probability from the observation history *zero-shot*, i.e., without training on the specific property. We cast monitoring as *attention over potential futures*: the property acts as a mask over candidate futures in the chosen language, whilst a predictive state representation captures the probabilities of a fixed set of short candidate futures given the observation history. Via a linear decoder, attention over these short futures collapses into one fixed weight vector compiled from the property's syntax. Hence, one learned representation serves any property without property-specific data or training. Whilst the number of potential futures grows exponentially in the time horizon of the property, our monitor only evaluates an inner product at runtime at a cost independent of the horizon. Lastly, we compute calibrated error bounds for learned predictive state representations; a new property needs no new data or model training, only recalibration on stored calibration labels. We validate the zero-shot capabilities of our monitor on driving data spanning agent tracks, rendered bird's-eye views, and raw camera frames. We attain 91 to 93% of the predictive skill of reference monitors which are trained separately for each property, answering each frame as it arrives at 9 to 150 times lower cost than a sampled autoregressive baseline.

1 INTRODUCTION

Autonomous systems such as embodied AI systems operate in open environments, where runtime monitors track safety-relevant quantities such as distances to obstacles (Leucker & Schallhart, 2009; Bartocci et al., 2018). Decisions rely on these monitors, so they must provide guarantees of correctness. Two developments challenge current monitoring approaches. Vision-based systems infer safety-relevant quantities from raw sensor data; e.g., end-to-end vision-language-action (VLA) policies (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024) that need not expose interpretable intermediate estimates. Generalist systems switch between different tasks on request without the ability of task-specific retraining, changing what properties need to be monitored.

Existing monitors target individual properties (Phan et al., 2018; Bortolussi et al., 2019; Cairoli et al., 2021), fixed vision-pipeline requirements (Dokhanchi et al., 2018; Balakrishnan et al., 2021; Hekmatnejad et al., 2024), or fixed policy failures (Sinha et al., 2024; Agia et al., 2024; Gu et al., 2025; Römer et al., 2025); runtime property queries may lack guarantees (Zhou et al., 2025). Predictors for unseen formulas instead estimate marginal probabilities, without conditioning on observation histories (Bortolussi et al., 2022; Saveri et al., 2024). We seek *zero-shot monitoring*: a monitor trained once per system, then prompted with any property of the supported fragment, without property-specific data or training. A zero-shot monitor is naturally at a disadvantage against vision-based monitors trained *for a fixed property*, so we investigate how closely it can match them in accuracy and runtime cost while retaining monitoring guarantees. Further related work appears in Appendix A.

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

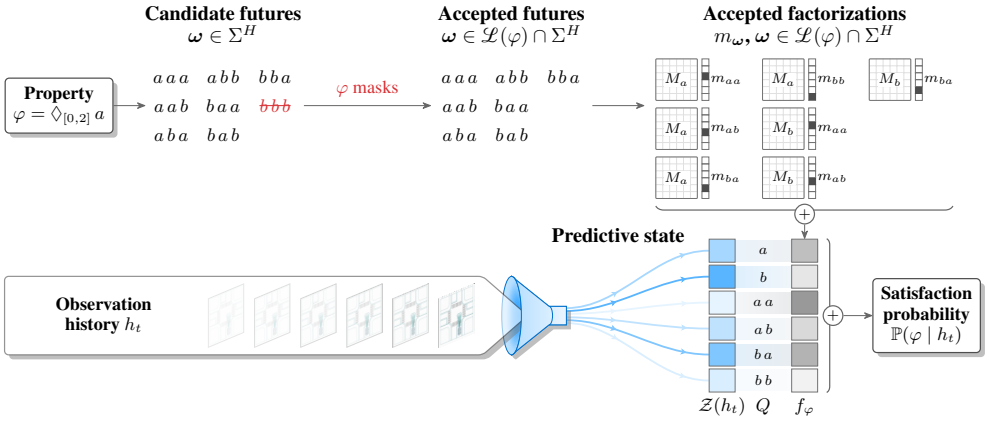


Figure 1: **Monitoring as attention over futures.** A formal property φ over labels a and b with horizon $H := \mathrm{hor}(\varphi) = 3$ accepts a set of futures (top row). Each accepted future factorizes into letter operators M_σ applied to the vector $m_{\boldsymbol{\omega}}$ of a core test $\boldsymbol{\omega} \in Q$, here all words of length at most $L = 2$ (Section 3.1); summing the factorizations gives f_φ. The same core tests index the predictive state $\mathcal{Z}(h_t)$, whose blue fill indicates each core test's probability given the observation history h_t. The property's probability is $\mathbb{P}(\varphi \mid h_t) = \mathcal{Z}(h_t)^\top f_\varphi$.

We cast monitoring of properties expressed in a formal language, such as *signal temporal logic* (STL) (Maler & Nickovic, 2004), as *attention over potential futures* (Figure 1). The *queried* property therein acts as a *mask* over all candidate futures, selecting all label sequences it accepts. A predictive state holds the probabilities of a fixed set of short candidate futures, called *core tests*, given the observation history; with the core tests as *keys* and their probabilities as *values*, attention returns the total probability of all futures that satisfy the property. The decoder is linear so that this attention collapses into one weight vector compiled from the property's syntax by formula progression (Bacchus & Kabanza, 1998). For this reason, runtime monitoring evaluates one inner product at a cost independent of the property's horizon despite exponentially many candidate futures. We introduce *language-induced predictive state representations* (PSRs), whose core tests are these short futures, modeling a partially observable system through predictions of fixed tests (Littman & Sutton, 2001; Singh et al., 2004). Unlike hidden-Markov forward filtering (Stoller et al., 2011), this requires no model of the physical state and carries only a fixed-size summary of the observation history, so its cost does not grow with the observation history and nothing needs learning at runtime. Zero-shot monitoring through this symbolic interface is a paradigm shift relative to existing property-independent monitors, which sample a generative model of the future and evaluate the property on every sample (Cairoli et al., 2025), at a runtime cost that grows with the horizon and sample budget.

In practice, the law of the system is unknown so that the predictive state needs to be learned from labeled trajectories. Nonetheless, a monitor used for safety needs a guarantee on how well its learned predictions are calibrated. This calibration certificate has to be group-conditional and cannot be point-wise, as in existing work. Therefore, we apply existing distribution-free histogram calibration (Gupta et al., 2020) to bound satisfaction probability averaged over histories assigned similar predictions. We develop a general mechanism for applying existing distribution-free calibration methods to zero-shot monitoring of formal-language properties. In this way, any group-conditional calibration method can be adopted in its place. We note that each new property requires recalibration, reusing stored calibration trajectories and representations without new model inference.

On real vision-based driving data, our zero-shot monitor retains 91 to 93 percent of the predictive skill of monitors explicitly trained per property. Compared to an autoregressive baseline, our monitor operates at 9 to 150 times lower monitoring cost per monitored frame. While the sampled generative monitor offers some degree of re-usability, our decoder offers deterministic decoding with a constant cost of one inner product per property per frame independently of the horizon and 0.6 to 1.0 millisecond adaptation to new properties; sampling trades horizon- and sample-dependent runtime for slightly higher accuracy with plentiful data, varying across repeated queries.

Contributions.

1. We propose language-induced PSRs for zero-shot monitoring (Section 3) that provide one representation that can be learned without property awareness, but serve any property in the formal language through syntax-compiled decoders, so a new property requires no new data or training.
2. The syntax-compiled linear decoder (Section 3.2) requires only one inner product per monitored frame, despite exponentially many candidate futures, independently of property horizon and history length; in comparison, generative sampling costs scale with horizon and sample budget.
3. We propose a learning and calibration framework for PSRs (Section 4) via distribution-free methods for calibrated error bounds per property. In our empirical evaluation (Section 5) we retain 91 to 93 percent of the predictive skill of monitors trained separately for each property, answering each frame as it arrives at 9 to 150 times lower cost than a sampled autoregressive baseline.

2 Problem Formulation

2.1 Partially Observable Stochastic System

We consider a stochastic system evolving in discrete time with latent state X_t taking values in $\mathbb{X} \subseteq \mathbb{R}^{d_x}$ and observation O_t taking values in $\mathbb{O}$, described as

$$X_{t+1} = f(X_t, W_t), \qquad O_t = g(X_t, V_t), \tag{1}$$

where f and g are measurable functions and W_t and V_t are i.i.d. process and observation noise sequences. The initial distribution $X_0 \sim \mu_0$ is fixed across training, calibration, and testing; μ denotes the joint process law of the random variables (X_t, O_t). Uppercase letters denote random variables and lowercase letters their realizations. We write $\mathbb{P}_\mu$ and $\mathbb{E}_\mu$ for probabilities and expectations under μ, dropping the subscript whenever the law is clear from context. Note that even if the latent process is Markov, future observations may depend on the full history $h_t = o_{0:t}$.

2.2 Signal Temporal Logic

Predicate functions $\pi_k \colon \mathbb{X} \to \mathbb{R}$ define a finite set of atomic propositions p_k by $\pi_k(x) \geq 0$, a fixed set $\mathrm{AP} := \{p_1, \ldots, p_m\}$, and alphabet $\Sigma := 2^{\mathrm{AP}}$. The predicates induce a labeling map $\ell(x) := \{p_k \in \mathrm{AP} : \pi_k(x) \geq 0\}$ recording the propositions that are true at a state x. A *word* of length H is $\boldsymbol{\omega} = \sigma_1 \cdots \sigma_H \in \Sigma^H$ with letters $\sigma_t := \ell(x_t) \in \Sigma$; ℓ also maps trajectories $\mathbf{x} = (x_s)_{s \in \mathbb{N}_0}$ to words. We also write $x_{s:t}$ to denote the trajectory of $\mathbf{x}$ between s and t. The bounded *signal temporal logic* (STL) (Maler & Nickovic, 2004) fragment $\mathcal{F}(\mathrm{AP})$ has grammar

$$\varphi \ ::= \ p \ \mid \ \neg\varphi \ \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \Box_I \varphi \ \mid \ \Diamond_I \varphi \ \mid \ \varphi_1 \ \mathsf{U}_I \ \varphi_2,$$

for $p \in \mathrm{AP}$ and bounded discrete-time intervals $I \subset \mathbb{N}_0$. The *language* of a formula φ is defined as $\mathcal{L}(\varphi) = \{\boldsymbol{\omega} \in \Sigma^* : \boldsymbol{\omega} \models \varphi\}$ (Maler & Nickovic, 2004), where Σ^* denotes the set of finite sequences over Σ, including the empty one, and Σ^+ the non-empty ones; the *horizon* $H = \mathrm{hor}(\varphi)$ counts the consecutive time steps needed to decide satisfaction (Dokhanchi et al., 2014). Appendix B gives the satisfaction semantics and horizon recursion. We write $\mathbb{P}(\varphi \mid h_t) := \mathbb{P}_\mu((\mathbf{X}, t+1) \models \varphi \mid O_{0:t} = h_t)$ for satisfaction of φ over the uncertain future $X_{t+1:t+H}$ given observations h_t.

2.3 Problem Formulation

We aim to construct one monitor for any property $\varphi \in \mathcal{F}(\mathrm{AP})$ without property-specific retraining, using N labeled trajectories split into training and calibration sets,

$$\mathcal{D} \equiv \{\mathcal{D}_{\mathrm{train}}, \mathcal{D}_{\mathrm{cal}}\} := \left\{o^{(i)}_{0:T_i},\, \ell(x^{(i)}_{0:T_i})\right\}_{i=1}^{N} \overset{\text{i.i.d.}}{\sim} \mu^{\otimes N}, \tag{2}$$

where $N = N_{\mathrm{train}} + N_{\mathrm{cal}}$. Only the labels are used, not the predicate functions or the latent states, so a proposition such as "the lead vehicle is cutting in" can be annotated per frame or read from a ground-truth log. At test time, neither the latent state nor the labeling map is available. Which propositions form AP is the user's choice, constrained by the available ground truth.

Reading the entire history h_t at each step costs time that grows without bound, motivating a *fixed-size* representation, which retains the information needed to answer *every* property in the fragment, with

one decoder per property reading out that property's satisfaction probability. Thus, we seek one encoder $z_t = Z(h_t)$ whose dimension is fixed independently of time and property, and per-property decoders g_φ recovering $\mathbb{P}(\varphi \mid h_t) = g_\varphi(z_t)$. This excludes storing raw history in z_t. The encoder depends on $\mathcal{F}(\mathrm{AP})$ but not on any individual property φ. With known law μ, the encoder is exact (Section 3). From finite data $\mathcal{D}$, both are estimated, equality is approximate, and we certify the mean satisfaction probability among histories carrying similar predictions (Section 4). Appendix C states both problem statements formally, omitted here for brevity.

3 Language-Induced Predictive State Representations

This section assumes the law μ is known and no estimation from data $\mathcal{D}$ is involved. Subsection 3.1 constructs a predictive state that is sufficient for the satisfaction probability of every formula in an STL fragment $\mathcal{F}(\mathrm{AP})$ (Theorem 1), and Subsection 3.2 compiles the decoder of any concrete property $\varphi \in \mathcal{F}(\mathrm{AP})$ from its syntax, so that no property-specific training is needed.

3.1 Predictive State Representations for Formal Languages

PSRs provide sufficient statistics for predicting the future behavior of partially observed dynamical systems by tracking the conditional probabilities of a fixed set of core *tests*; classically, these tests are potential continuations of the observation history h_t, from which every other test is predicted (Littman & Sutton, 2001; Singh et al., 2012); a PSR is called *linear* when every test probability is a linear function of this prediction vector (Definition 3 in Appendix D).

Key Idea: Words as Tests. STL formulas specify events over the evolution of latent propositions, as in the monitoring target $\mathbb{P}(\varphi \mid h_t)$ we strive to predict. To this end, we construct PSRs whose tests are based on the STL alphabet $\Sigma = 2^{\mathrm{AP}}$ and track labels $\ell(X_t)$, so that a test is a *word* $\boldsymbol{\omega} \in \Sigma^+$, with

$$\mathbb{P}(\boldsymbol{\omega} \mid h_t) \;=\; \mathbb{P}_\mu\big(\ell(X_{t+1:t+|\boldsymbol{\omega}|}) = \boldsymbol{\omega} \,\big|\, O_{0:t} = h_t\big). \tag{3}$$

Unlike in the classical PSR definition (Singh et al., 2004), such a test does not extend the observed history h_t, and its *concrete* outcome is never observed at test time. Since the labeling map ℓ acts on the latent state X_t, (3) resolves predictive uncertainty about the future $X_{t+1:t+|\boldsymbol{\omega}|}$ and partial observability of the past $x_{0:t}$ simultaneously.

Constructing a PSR. The core tests follow from the semantics of STL rather than having to be discovered. For a max. *test length* $L \in \mathbb{N}$, let the core tests be the set of all words of length at most L,

$$Q := \textstyle\bigcup_{1 \le n \le L} \Sigma^n, \qquad \text{so that} \qquad \mathcal{Z}(h_t) = [\mathbb{P}(\boldsymbol{\omega} \mid h_t)]_{\boldsymbol{\omega} \in Q} \in [0,1]^q \tag{4}$$

with $q := |Q|$ predicts latent words, excluding the empty word ε. For now, we assume that the probabilities $\{\mathbb{P}(\boldsymbol{\omega} \mid h_t)\}_{\boldsymbol{\omega} \in Q}$ are known from μ.

Factorization. If a property's horizon satisfies $\mathrm{hor}(\varphi) \le L$, each of its accepted words is a core test, i.e., $\mathbb{P}(\boldsymbol{\omega} \mid h_t) = \mathcal{Z}(h_t)^\top e_{\boldsymbol{\omega}}$ with $e_{\boldsymbol{\omega}}$ indexing the appropriate coordinate. In this case, $\mathbb{P}(\varphi \mid h_t)$ is the total probability over those core tests whose words satisfy φ, i.e., the sum of coordinates of the predictive state $\mathcal{Z}(h_t)$. Since the number q of core tests grows exponentially in L, Q cannot cover every horizon, so for $\mathrm{hor}(\varphi) > L$ the accepted words lie outside Q and must be *factorized* into core tests (Figure 1). Under the span assumption of Appendix D (Assumption 1), their probabilities are nevertheless linear in the predictive state, and this linearity extends from a word $\boldsymbol{\omega}' \in \Sigma^+$ to the longer word $\sigma\boldsymbol{\omega}'$ through an operator $M_\sigma \in \mathbb{R}^{q \times q}$ per letter $\sigma \in \Sigma$. Composing operators recursively,

$$\mathbb{P}(\boldsymbol{\omega} \mid h_t) = \mathcal{Z}(h_t)^\top m_{\boldsymbol{\omega}}, \qquad m_{\boldsymbol{\omega}} = \begin{cases} e_{\boldsymbol{\omega}}, & \boldsymbol{\omega} \in Q, \\ M_\sigma\, m_{\boldsymbol{\omega}'}, & \boldsymbol{\omega} = \sigma\boldsymbol{\omega}' \text{ with } |\boldsymbol{\omega}| > L, \end{cases} \tag{5}$$

e.g., $m_{aba} = M_a e_{ba}$ for $L = 2$ (Figure 1). Appendix D constructs the operators (Asm. 1 & Prop. 2). They are the only part of the decoder that depends on the system, and Section 4 estimates them from data $\mathcal{D}_{\mathrm{train}}$. Equation (5) is a *letter-wise factorization*, peeling one letter at a time; for exact operators, any split of $\boldsymbol{\omega}$ into blocks of at most L letters gives the same $m_{\boldsymbol{\omega}}$; Appendix E.2 discusses other factorizations. Based on (5), we establish sufficiency for every formula in the fragment:

Definition 1 (Language-Induced PSR)**.** A *language-induced (linear) PSR of order* L is a linear PSR (formally stated in Definition 3 in Appendix D) whose core tests are the latent words Q of (4), with predictive state $\mathcal{Z}(h_t)$ and operators $\{m_\sigma, M_\sigma\}_{\sigma \in \Sigma}$ as in (5).

Definition 1 is the standard PSR of Thon & Jaeger (2015, Definition 29) but for core tests fixed system independently.

Theorem 1 (Sufficiency for the Fragment)**.** *Let $\mathcal{Z}$ be a language-induced PSR of order L. Then, for every $\varphi \in \mathcal{F}(\mathrm{AP})$ with horizon $H := \mathrm{hor}(\varphi)$ and μ-almost every history h_t,*

$$\mathbb{P}(\varphi \mid h_t) = \mathcal{Z}(h_t)^\top f_\varphi, \qquad f_\varphi = \sum_{\boldsymbol{\omega} \in \mathcal{L}(\varphi) \cap \Sigma^H} m_{\boldsymbol{\omega}}, \tag{6}$$

Consequently, (i) one $\mathcal{Z}$ is sufficient for every $\varphi \in \mathcal{F}(\mathrm{AP})$*; (ii) f_φ depends on the system only through $\{m_\sigma, M_\sigma\}_{\sigma \in \Sigma}$ and otherwise only on the syntax of φ, so formulas with $H > L$ need no enlargement of Q and no retraining.*

Equation (6) separates what is learned from what is syntax (Figure 1: the upper row compiles f_φ, the lower row computes $\mathcal{Z}(h_t)$). The system enters only through the operators, the property only through the words it accepts, and the observation history only through $\mathcal{Z}(h_t)$. Hence f_φ is assembled before monitoring begins, and every approximation in the decoder enters through the (learned) operators it composes, which Section 4 estimates and compresses. The proof is in Appendix D.

The decoder $f_\varphi^\top \mathcal{Z}(h_t)$ is an attention-like key–value lookup (Vaswani et al., 2017): the property supplies the query f_φ, the core tests $\boldsymbol{\omega} \in Q$ are the keys with selectors $e_{\boldsymbol{\omega}}$, their probabilities $\mathbb{P}(\boldsymbol{\omega} \mid h_t)$ are the values, and the language $\mathcal{L}(\varphi)$ is the mask over candidate futures. Because query and keys are written in the same alphabet, the weights $f_{\varphi,\boldsymbol{\omega}} = f_\varphi^\top e_{\boldsymbol{\omega}}$ are exact and fixed before the first observation arrives (You et al., 2020; Tay et al., 2021), so runtime attention is one inner product without word matching (Appendix E.1).

3.2 Syntactic Zero-Shot STL Decoding

A property $\varphi \in \mathcal{F}(\mathrm{AP})$ with horizon $H := \mathrm{hor}(\varphi)$ induces the syntactic mask $\mathcal{L}(\varphi) \cap \Sigma^H$, from which (6) assembles the query f_φ. Theorem 1 establishes that f_φ exists but not how to construct it. We show here how to compile it from the syntax of φ by dynamic programming, without enumerating $\mathcal{L}(\varphi)$ and without property-specific data or training.

Automaton Decoder. *Formula progression* (Bacchus & Kabanza, 1998) compiles φ into a layered *deterministic finite automaton* (DFA) with transitions $\delta_\tau \colon \mathcal{R}_\tau \times \Sigma \to \mathcal{R}_{\tau+1}$; evaluating δ_τ follows the progression algorithm of Bacchus & Kabanza (1998, Table 1). Each location $\varrho \in \mathcal{R}_\tau \subset \mathcal{F}(\mathrm{AP})$ is the *residual obligation* after reading τ letters (Figure 2); a word $\boldsymbol{\omega}$ is read against the DFA from left to right. Each residual $\varrho \in \mathcal{R}_\tau$ is a formula of horizon at most $H - \tau$, so Theorem 1 applies at every layer. Its accepted suffix words decompose as $\mathcal{L}(\varrho) = \sqcup_{\sigma \in \Sigma} \{\sigma\} \cdot \mathcal{L}(\delta_\tau(\varrho, \sigma))$, which yields the backward recursion $f_\tau(\varrho) := \sum_{\boldsymbol{\omega} \in \mathcal{L}(\varrho) \cap \Sigma^{H-\tau}} m_{\boldsymbol{\omega}}$, explicitly constructed as:

$$f_\tau(\varrho) = \sum_{\sigma \in \Sigma} M_\sigma \, f_{\tau+1}\big(\delta_\tau(\varrho, \sigma)\big) \quad (\tau < H - 1), \qquad f_{H-1}(\varrho) = \sum_{\sigma \,:\, \delta_{H-1}(\varrho,\sigma) = \top} m_\sigma, \tag{7}$$

and the recursion returns $f_\varphi = f_0(\varrho_0)$. Starting at layer $H-1$ with the vectors m_σ avoids introducing an empty-word weight.

Read against the predictive state, equation (7) is the law of total probability over the next letter, i.e., backward dynamic programming over the DFA layers with $f_\tau(\varrho)$ as value function; δ_τ is known and deterministic, and the unknown, possibly non-Markovian dynamics (1) enter only through M_σ. Hence, we can solve for $f_0(\varrho_0)$ using existing dynamic programming solution approaches.

Zero-Shot Runtime Monitor. Solving (7) offline yields $f_\varphi = f_0(\varrho_0)$, and runtime monitoring evaluates the single inner product $\mathbb{P}(\varphi \mid h_t) = \mathcal{Z}(h_t)^\top f_0(\varrho_0)$ of (6). Compilation thus avoids enumerating the $|\Sigma|^H$ words of $\mathcal{L}(\varphi)$. For exact operators and properties with horizon $H \le L$, the compiled weights equal the mask $\mathbf{1}(\boldsymbol{\omega} \models \varphi)$ over Q; with estimated operators they are dense, signed, and unnormalized. Note that any alternative construction nonlinear in $\mathcal{Z}(h_t)$, such as a softmax over words, does not collapse into a single weight vector offline and would have to be evaluated word by word at runtime; we avoid such string matching. Appendix E.3 explores further decoding strategies.

4 Estimation and Certification

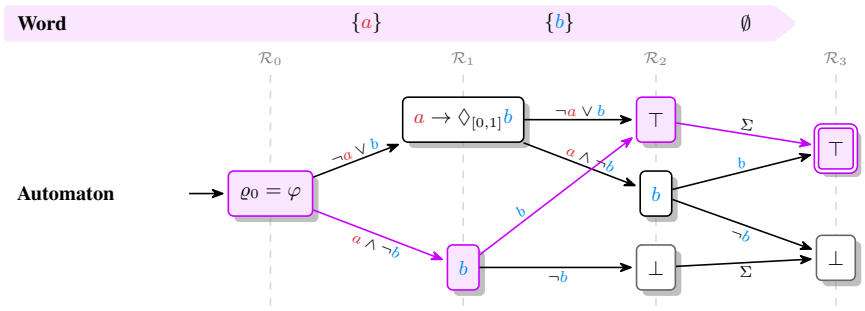


Figure 2: Layered DFA obtained by progressing $\varphi = \Box_{[0,1]}\big(a \to \Diamond_{[0,1]} b\big)$, with $H = 3$. Each location is the *residual obligation* ϱ that the remaining suffix must satisfy, and each edge is labelled by a Boolean test. For example, the word $\{a\}\{b\}\emptyset$ is accepted along the highlighted path. The decoder (7) is a backward pass over this graph.

Subsection 4.1 learns a property-independent representation from data $\mathcal{D}_{\text{train}}$ and compiles property decoders; Subsection 4.2 lower-bounds the mean satisfaction probability within prediction bins on held-out trajectories $\mathcal{D}_{\text{cal}}$ (Appendix C states both problems formally).

4.1 Training the Representation

Figure 5 in Appendix F summarizes the pipeline. We encode the observation history recurrently into a fixed-size history feature,

$$\xi_{t+1} = \text{rnn}_\theta\big(\xi_t,\, \text{enc}(o_{t+1})\big), \qquad \xi_{-1} = 0, \tag{8}$$

where $\xi_t := \xi(h_t) \in \mathbb{R}^d$, enc is a frozen encoder with output dimension d_e, which differs across our benchmarks (Appendix H.1), and $\text{rnn}_\theta \colon \mathbb{R}^d \times \mathbb{R}^{d_e} \to \mathbb{R}^d$ is a recurrent map whose parameters θ are trained on $\mathcal{D}_{\text{train}}$ to predict realized label words of length L through a linear head, independently of any specific monitoring property. Unlike the predictive state $\mathcal{Z}$, the features ξ_t have no probabilistic meaning. The monitor stores only ξ_t between steps, advanced by (8) at a cost independent of time and without retaining raw observations. The recurrent architecture is one choice; the construction requires only a history feature of fixed dimension and the current observation.

We estimate the predictive state by ridge regression of the word indicators $\mathbf{1}(\ell(x_{t+1:t+|\boldsymbol{\omega}|}) = \boldsymbol{\omega})$, $\boldsymbol{\omega} \in Q$, on the feature ξ_t over the labeled training trajectories (Appendix F.1); its query cost $\mathcal{O}(d|Q|)$ does not depend on the training data amount N_{train}. Since an indicator's conditional mean is its probability (3), this estimates $\mathcal{Z}(h_t)$, up to the information about the history h_t that ξ_t discards.

The core tests Q are deliberately overcomplete, so that every training sample enters the estimate, and their probabilities are strongly dependent across histories. Compressing the predictive state to rank $r \ll q$, i.e., $\mathcal{Z}_C := C\mathcal{Z}$ via a compression matrix $C \in \mathbb{R}^{r \times q}$ (each coordinate of $\mathcal{Z}_C$ is a linear combination of core-test probabilities), affords more stable estimates of the letter operators $\{\widehat{m}_\sigma, \widehat{M}_\sigma\}_{\sigma \in \Sigma}$, which now act on $\mathbb{R}^r$ (Thon & Jaeger, 2015). The *characterizer* C is obtained by a weighted singular value decomposition of the estimated predictive states of the training histories, and the letter operators are estimated by ridge regression and composed in these coordinates (App. F.2). Both are trained on observation histories, whereas composition applies the operators to predictive states of *extended* histories (h, u), which append a latent prefix $u \in \Sigma^+$ (Def. 4). The fitted operators are therefore identified only on the span of the observation-history states, which need not contain the extended-history states. We treat compression and learned operator composition as approximations whose errors affect accuracy, not the validity of Prop. 1. The rank r trades representational error against estimation variance and is therefore selected on a validation split. Whilst any factorization of a word into leading letters and a final core test returns the same $m_{\boldsymbol{\omega}}$ for *exact* operators, when operators are estimated from data, factorization matters; our implementation continues the letter-wise factorization (5) down to the final *single* letter, e.g., $m_{aba} = M_a M_b e_a$ (App. E.2).

With $\widehat{\mathcal{Z}}$ the resultant estimate of $\mathcal{Z}$, the compiled decoder reads out

$$\hat{p}_\varphi(h) \;:=\; \big[\, f_\varphi^\top \widehat{\mathcal{Z}}(h) \,\big]_0^1, \tag{9}$$

where $[\cdot]_0^1$ clips to $[0,1]$, and f_φ is compiled by (7) from the estimated compressed operators. Applying the same clipping at calibration and query time preserves the guarantee of Subsection 4.2.

4.2 Certifying the Prediction

The satisfaction probability $\mathbb{P}(\varphi \mid h)$ is never observed in practice; each calibration trajectory in $\mathcal{D}_{\text{cal}}$ reveals only a realization $\text{sat} \in \{0,1\}$ of a Bernoulli random variable with this mean. Without further assumptions, no informative distribution-free bound on $\mathbb{P}(\varphi \mid h)$ for an individual history is possible (Barber, 2020). We therefore use distribution-free histogram binning (Gupta et al., 2020) to certify the *bin-conditional* mean satisfaction probability. Throughout this subsection, the property φ is fixed and suppressed in the notation of $\hat{p}$, β, $\mathcal{H}_b$, and $\hat{p}_b^{\text{lb}}$, which we introduce below.

Fix bin edges $0 = \tau_0 < \cdots < \tau_B = 1$, let $\beta(h) := b$ whenever $\hat{p}(h) \in [\tau_{b-1}, \tau_b)$, closing the last bin, and let $\mathcal{H}_b := \{h \colon \beta(h) = b\}$. At query time t_i, write $h^{(i)} := o^{(i)}_{0:t_i}$, and let $\mathcal{I}_b := \{i \colon \beta(h^{(i)}) = b\}$ index the calibration histories in bin b, with $N_b := |\mathcal{I}_b|$. Each calibration trajectory supplies the outcome $\text{sat}^{(i)} := \mathbf{1}(\ell(x^{(i)}_{t_i+1:t_i+H}) \in \mathscr{L}(\varphi))$ of a Bernoulli random variable with mean $\mathbb{P}(\varphi \mid h^{(i)})$, from which

$$\hat{\pi}_b := \frac{1}{N_b} \sum_{i \in \mathcal{I}_b} \text{sat}^{(i)}, \qquad w_b := \sqrt{\tfrac{\ln(B/\delta)}{2N_b}}, \qquad \hat{p}_b^{\text{lb}} := \max\{0,\, \hat{\pi}_b - w_b\}, \tag{10}$$

where w_b is a one-sided Hoeffding width with Bonferroni correction over B bins; empty bins receive the uninformative valid bound 0. Appendix G.1 collects the calibration notation; Appendix H.3 treats the choice of edges, which must be computed on a data split disjoint from the labels forming $\hat{\pi}_b$.

Proposition 1 (Certified Bin-Conditional Satisfaction Probability)**.** *Fix φ, bin edges $\tau_0, \ldots, \tau_B$, and a measurable predictor $\hat{p}_\varphi \colon \mathbb{O}^* \to [0,1]$ independent of $\mathcal{D}_{\text{cal}}$ in* (2)*. The predictor may be the deployed estimate of* (9)*. For any $\delta \in (0,1)$, the bounds* (10) *satisfy*

$$\mathbb{P}_{\mu^{\otimes N_{\text{cal}}}}\Big(\mathbb{E}_\mu\big[\mathbb{P}(\varphi \mid O_{0:t}) \mid O_{0:t} \in \mathcal{H}_b\big] \geq \hat{p}_b^{\text{lb}} \quad \forall b \in [B] \text{ with } \mathbb{P}_\mu(O_{0:t} \in \mathcal{H}_b) > 0\Big) \geq 1 - \delta.$$

In particular, the simultaneous event applies at the bin $b = \beta(h_t)$ of any query history.

The proof is in Appendix G.2. With fixed edges and calibration-independent predictions, rank-r projection, operator estimation error, and feature lossiness affect the bound's value but not its validity.

Remark 1 (What Prompting a New Property Costs)**.** *We note that only the decoder is zero-shot: Proposition 1 fixes the property φ before calibration and bins the predictions $\hat{p}_\varphi$. A new property $\varphi' \in \mathcal{F}(\text{AP})$ reuses $\mathcal{D}_{\text{cal}}$ without new trajectories, model inference, or retraining, re-evaluating $\text{sat}^{(i)}$ from stored labels and recomputing* (10) *with an $\mathcal{O}(N_{\text{cal}} \log N_{\text{cal}})$ sort.*

5 Experiments

This section tests whether one monitor, learned once per system, predicts the satisfaction probabilities of many properties about as accurately as monitors trained for each property. The monitored system is a vehicle in real driving data. The monitor observes agent tracks, rendered bird's-eye views (BEV), or camera frames, whereas the atomic propositions threshold latent physical quantities, such as headway, time to collision, and clearance, that it never observes directly. A property is an STL formula over these propositions, e.g., $\Diamond_{[0,9]} \neg p_h$, "the headway falls below two seconds within the next ten frames", with horizon $H = \text{hor}(\varphi) = 10$ frames. We address three questions.

Question 1. *Accuracy.* Does one representation learned without reference to any property decode many properties of the fragment about as accurately as a monitor trained per property?

Question 2. *Runtime Cost.* What does answering all properties on every frame, and admitting a new one, cost at runtime?

Question 3. *Certificate.* How informative is the certificate of Proposition 1 at our calibration sizes?

Benchmarks. *Waymo Open Motion Dataset (WOMD)* (Ettinger et al., 2021; Chen et al., 2024) supplies agent tracks, a custom benchmark *Waymo BEV* renders bird's-eye views from WOMD scenarios, and *Waymo Perception* (Sun et al., 2020) supplies raw camera frames through a frozen DINOv2 encoder (Oquab et al., 2024); Perception is two orders of magnitude smaller than the others.

Table 1: **Accuracy and cost of monitoring.** Brier scores are means over three seeds and the included properties of Table 3; skill is measured against the per-property reference. *Inference* answers the properties on every frame, one history per call. *New Property* admits one new property: compiling f_φ for ours, evaluating the property's outcomes on the stored training labels and refitting the ridge regression for the per-property reference, training a new network for the RNN, and re-evaluating stored samples for AR-64. Both are timed on the properties with $H \leq 20$.

Benchmark	Monitor	Reusable	Brier ↓	skill ↑	Inference (GPU / CPU)	New Property (GPU / CPU)
WOMD	ours	✔	0.1291	0.93	107 µs / 658 µs	1.0 ms / 1.0 ms
	per-property reference	✘	0.1209	1.00	46 µs / 604 µs	1.8 ms / 1.8 ms
	RNN	✘	0.1234	0.98	7.6 ms / 7.5 ms	10.5 s / 78 s
	sampled AR-64	✔	0.1075	1.14	1.0 ms / 7.9 ms	0.6 µs / 0.6 µs
Waymo BEV	ours	✔	0.1437	0.91	56 µs / 312 µs	0.8 ms / 0.8 ms
	per-property reference	✘	0.1357	1.00	35 µs / 308 µs	1.7 ms / 1.7 ms
	RNN	✘	0.1353	1.00	9.3 ms / 4.3 ms	9.7 s / 53 s
	sampled AR-64	✔	0.1288	1.07	1.3 ms / 3.4 ms	0.8 µs / 0.9 µs
Perception	ours	✔	0.1020	0.93	23 µs / 22 µs	0.6 ms / 0.6 ms
	per-property reference	✘	0.0976	1.00	4 µs / 4 µs	0.3 ms / 0.3 ms
	RNN	✘	0.1028	0.91	2.0 ms / 1.5 ms	7.2 s / 18 s
	sampled AR-64	✔	0.1098	0.79	1.2 ms / 3.3 ms	0.8 µs / 0.8 µs

Baselines and Evaluation. We compare our monitor against monitors trained for each property separately. We note that these baseline monitors have an "unfair" advantage compared to our zero-shot monitor and merely serve as a reference to compare accuracy and runtime cost against. The *per-property reference* fits the same ridge regression and history feature as our monitor directly to the outcomes of one property, isolating the accuracy lost by decoding all properties through the predictive state. We additionally train one *RNN* for each property. All monitors read the same frozen encoder on each benchmark. Our monitor and the per-property reference also share the history feature ξ, whereas the RNN and AR-64 learn their own recurrent summaries of the encoded frames; see Appendix H.1.

For evaluation, we use the Brier score, averaged over test histories, which is a strictly proper scoring rule to measure prediction accuracy (Gneiting & Raftery, 2007; Waghmare & Ziegel, 2025). To isolate a monitor's accuracy from the observability of the property, we also report the score of the *constant base-rate predictor*, which ignores the history h_t. Based on this base rate, we also compute the *skill* of a monitor, $(\mathrm{Brier}_{\mathrm{const}} - \mathrm{Brier})/(\mathrm{Brier}_{\mathrm{const}} - \mathrm{Brier}_{\mathrm{ref}})$, which is the fraction of the per-property reference's improvement over the constant base-rate predictor, i.e., $\mathrm{Brier}_{\mathrm{const}} - \mathrm{Brier}_{\mathrm{ref}}$, that the monitor attains; skill is 0 for the constant predictor and 1 for the per-property reference. We average over all tested properties. Finally, we include a *sampled autoregressive baseline* (AR-64) akin to Cairoli et al. (2025), which like ours serves every property without retraining. It samples $K = 64$ futures at every frame and reports the fraction that satisfies φ. The appendix gives example frames (Figure 6), dataset sizes, and training and query details.

On some properties, every monitor performs like the constant base-rate predictor. Including these properties in the averaged scores would not add any information and preclude the skill computation. We therefore exclude a candidate if, on a validation split, a regressor trained for this property alone does not improve on the constant predictor by at least 0.01 Brier, if its base rate lies outside $[0.05, 0.95]$, or if its outcomes have absolute correlation exceeding 0.95 with those of an included property. All numbers in this section are means over the remaining 34 of 50 (WOMD), 34 of 51 (Waymo BEV), and 4 of 35 (Perception) candidates at all horizons; Table 3 in Appendix H.2 lists every candidate and the reason for each exclusion.

5.1 Results

Reuse retains near-reference accuracy but degrades at long horizons (Question 1). Our decoder retains 93, 91, and 93% skill on WOMD, Waymo BEV, and Perception, respectively, within 0.004 to 0.008 Brier of the per-property reference and 0.008 of the RNN (Table 1). For properties with long horizons ($H = 40$), skill falls to 0.82 for the nested headway property $\square_{[0,36]}\Diamond_{[0,3]}p_h$ on WOMD and to 0.55 for the corresponding time-to-collision property on Waymo BEV; recall that the decoder composes $H - 1$ learned letter operators in (7). The properties above passed the validation gate.

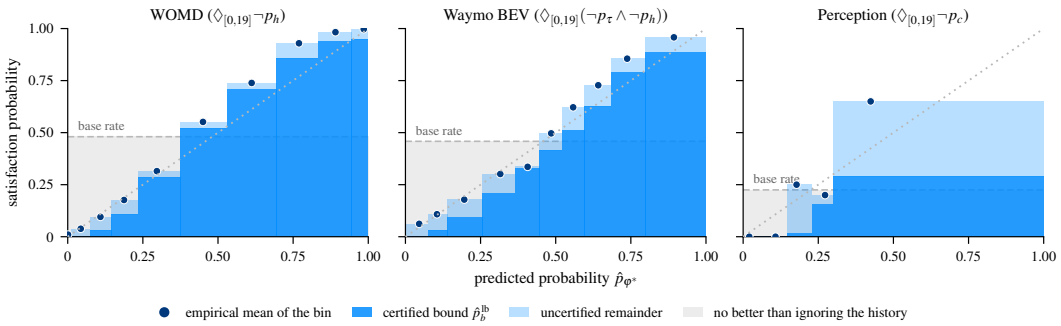


Figure 3: **Certified reliability diagram.** One property per benchmark, one bar per bin of (10). The horizontal axis is the probability the monitor predicts; the vertical axis is the satisfaction probability of the histories it places in that bin, estimated by their frequency of satisfying φ on the test split (dot). The solid part of each bar reaches the certified lower bound, and a bound below the dashed base rate says no more than a constant predictor, which ignores the observation history.

A family of recovery and response properties $\Box_{[0,W-1]}(\neg q \rightarrow \Diamond_I p)$ with outer window W over proposition pairs, compiled without the gate, reaches skill 0.96, 0.92, and 0.81 on WOMD, Waymo BEV, and Perception at the main outer-window settings and 0.92, 0.54, and 0.77 including long instances, over the members whose base rate lies in $[0.05, 0.95]$ and whose per-property reference improves on the constant predictor by at least 0.002 Brier (Appendix H.1).

Sampling improves accuracy with plentiful data but is slower at runtime (Question 2). AR-64 improves Brier by 0.022 on WOMD and 0.015 on Waymo BEV; on Perception, AR-64 is less accurate on average (0.110 against 0.102) and varies strongly across seeds (standard deviation 0.023). Sampling also makes the monitor's output random, whereas our decoder returns a deterministic value. Our monitor runs at 22 to 658 μs per frame across the three benchmarks on both CPU and GPU, against 1.0 to 7.9 ms for AR-64 and 1.5 to 9.3 ms for the per-property RNN, which must run one trained model per property. New properties compile in a median 0.6 to 1.0 ms, and at most 12.6 ms, versus 7.2 to 78 s to train a new RNN; AR-64 re-evaluates stored futures.

Certificate informativeness varies across benchmarks (Question 3). With $\delta = 0.1$, the bounds of a property hold simultaneously over its bins with probability at least 0.9. Upper motion-benchmark bins certify group satisfaction above the base rate; two of Perception's five bins yield the uninformative bound zero (Figure 3).

Limitations. The evidence covers a single application domain. On the raw-camera benchmark, only the clearance properties pass the gate: a regressor trained for each time-to-collision property alone cannot improve on the constant predictor, so the limit lies in the frozen DINOv2 features (Oquab et al., 2024) rather than in our decoder (Appendix H.1). Its only 521 calibration scenarios also make its certificate weak. Intrinsic to our monitor is the long-horizon degradation reported under Question 1. Finally, the alphabet has $2^{|\,\mathrm{AP}\,|}$ letters, so the numbers of core tests and letter operators grow exponentially in $|\,\mathrm{AP}\,|$; our benchmarks use three or four propositions.

6 Conclusion and Future Directions

We proposed a monitor that is prompted with a property in a formal language and predicts its satisfaction probability from the observation history without property-specific training. The property masks candidate futures, a language-induced predictive state captures the probabilities of a fixed set of short futures, and, because decoding is linear, the syntax of the property compiles into one weight vector, so one representation learned per system serves every property of the fragment at a runtime cost independent of the horizon, with a distribution-free certificate per property from a shared calibration set. On driving data from agent tracks to raw camera frames, the monitor retains 91 to 93% of the skill of monitors trained per property at 9 to 150 times lower cost per frame than a sampled autoregressive baseline. Open directions are estimators that limit error accumulation when many learned letter operators are composed, smoothing for properties whose time window has partly elapsed, and a certificate that holds simultaneously for all properties of the fragment.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing, to assist with implementing and running the experiments and with preparing results for the paper, and to review drafts, including checking proofs and reported numbers and proposing corrections. We did not use generative AI tools to generate research ideas; the research questions, the core method, and the main results originate with the authors. All AI-assisted text, results, and proof revisions were reviewed by the authors, and we take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our applications are in safety-critical domains, such as autonomous driving, where the monitor's predictions may be used to inform decisions. Our guarantees are probabilistic in nature, and the monitor's predictions are only as good as the data used to train and calibrate it. We encourage users to consider the limitations of the monitor, including its potential for error, and to use it in conjunction with other safety measures. We also encourage transparency in reporting the monitor's performance and limitations when deploying it in real-world applications.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of our methods, datasets, and experimental setups in the main text and appendix. All assumptions made in our theoretical results are clearly stated, and we provide complete proofs of our claims in the appendix.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Zi-ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *Conference on Robot Learning (CoRL)*, 2024. arXiv:2410.04640.

Masafumi Akahira and Kei Takeuchi. A note on prediction sufficiency (adequacy) and sufficiency. *Australian Journal of Statistics*, 22(3):332–335, 1980.

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2022.

Reza Babaee. *Predictive Runtime Verification of Stochastic Systems*. PhD thesis, University of Waterloo, 2019.

Reza Babaee, Arie Gurfinkel, and Sebastian Fischmeister. $\mathcal{P}$revent: A predictive run-time verification framework using statistical learning. In *International Conference on Software Engineering and Formal Methods (SEFM)*, volume 10886 of *LNCS*, pp. 205–220, 2018.

Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, 22(1):5–27, 1998.

C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008. ISBN 9780262026499.

Anand Balakrishnan, Jyotirmoy Deshmukh, Bardh Hoxha, Tomoya Yamaguchi, and Georgios Fainekos. PerceMon: Online monitoring for perception systems. In *Runtime Verification (RV)*, Lecture Notes in Computer Science, pp. 297–308. Springer, 2021. doi: 10.1007/978-3-030-88494-9_18.

Rina Foygel Barber. Is distribution-free inference possible for binary regression? *Electronic Journal of Statistics*, 14(2):3487–3524, 2020.

Ezio Bartocci, Yliès Falcone, Adrian Francalanza, and Giles Reger. Introduction to runtime verification. In *Lectures on Runtime Verification*, volume 10457 of *Lecture Notes in Computer Science*, pp. 1–33. Springer, 2018. doi: 10.1007/978-3-319-75632-5_1.

Alain Berlinet and Christine Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer, 2004. doi: 10.1007/978-1-4419-9096-9.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control, 2024. arXiv:2410.24164.

Byron Boots, Sajid M. Siddiqi, and Geoffrey J. Gordon. Closing the learning-planning loop with predictive state representations. *IJRR*, 30(7):954–966, 2011.

Byron Boots, Arthur Gretton, and Geoffrey J Gordon. Hilbert space embeddings of predictive state representations. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 92–101, 2013.

Luca Bortolussi, Francesca Cairoli, Nicola Paoletti, Scott A. Smolka, and Scott D. Stoller. Neural predictive monitoring. In *Runtime Verification (RV)*, Lecture Notes in Computer Science, pp. 129–147. Springer, 2019. doi: 10.1007/978-3-030-32079-9_8.

Luca Bortolussi, Giuseppe Maria Gallo, Jan Křetínskỳ, and Laura Nenzi. Learning model checking and the kernel trick for signal temporal logic on stochastic processes. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 281–300. Springer, 2022.

Maxime Bouton, Jana Tumova, and Mykel J. Kochenderfer. Point-based methods for model checking in partially observable Markov decision processes. In *AAAI Conference on Artificial Intelligence*, 2020. arXiv:2001.03809.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control, 2023. arXiv:2307.15818.

Francesca Cairoli, Luca Bortolussi, and Nicola Paoletti. Neural predictive monitoring under partial observability. In *Runtime Verification (RV)*, Lecture Notes in Computer Science, pp. 121–141. Springer, 2021. doi: 10.1007/978-3-030-88494-9_7.

Francesca Cairoli, Nicola Paoletti, and Luca Bortolussi. Conformal quantitative predictive monitoring of STL requirements for stochastic processes. In *ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2023. doi: 10.1145/3575870.3587113.

Francesca Cairoli, Luca Bortolussi, Jyotirmoy V. Deshmukh, Lars Lindemann, and Nicola Paoletti. Conformal predictive monitoring for multi-modal scenarios. In *International Conference on Runtime Verification (RV)*, pp. 336–356. Springer, 2025. doi: 10.1007/978-3-032-05435-7_19.

Kan Chen, Runzhou Ge, Hang Qiu, Rami Ai-Rfou, Charles R. Qi, Xuanyu Zhou, Zoey Yang, Scott Ettinger, Pei Sun, Zhaoqi Leng, Mustafa Baniodeh, Ivan Bogun, Weiyue Wang, Mingxing Tan, and Dragomir Anguelov. Womd-lidar: Raw sensor dataset benchmark for motion forecasting. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2024.

Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. On-line monitoring for temporal logic robustness. In *Runtime Verification (RV)*, volume 8734 of *Lecture Notes in Computer Science*, pp. 231–246. Springer, 2014. doi: 10.1007/978-3-319-11164-3_19.

Adel Dokhanchi, Heni Ben Amor, Jyotirmoy V. Deshmukh, and Georgios Fainekos. Evaluating perception systems for autonomous vehicles using quality temporal logic. In *Runtime Verification (RV)*, Lecture Notes in Computer Science, pp. 409–416. Springer, 2018. doi: 10.1007/978-3-030-03769-7_23.

Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aur'elien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9710–9719, October 2021.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, 2022.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. SAFE: Multitask failure detection for vision-language-action models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.09937.

Chirag Gupta, Aleksandr Podkopaev, and Aaditya Ramdas. Distribution-free binary classification: prediction sets, confidence intervals and calibration. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Mohammad Hekmatnejad, Bardh Hoxha, Jyotirmoy V. Deshmukh, Yezhou Yang, and Georgios Fainekos. Formalizing and evaluating requirements of perception systems for automated vehicles using spatio-temporal perception logic. *The International Journal of Robotics Research*, 43(2): 203–238, 2024. doi: 10.1177/02783649231223546.

Bardh Hoxha, Oliver Schön, Hideki Okamoto, Lars Lindemann, and Georgios Fainekos. Vision-based runtime monitoring under varying specifications using semantic latent representations. In *IEEE Conference on Decision and Control (CDC)*, 2026. arXiv:2605.13923.

Herbert Jaeger. Observable operator processes and conditioned continuation representations. Technical Report Arbeitspapiere der GMD 1043, GMD — Forschungszentrum Informationstechnik, Sankt Augustin, Germany, 1997.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 5156–5165, 2020.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model, 2024. arXiv:2406.09246.

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

Lars Lindemann, Xin Qin, Jyotirmoy V. Deshmukh, and George J. Pappas. Conformal prediction for STL runtime verification. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2023. arXiv:2211.01539.

Michael Littman and Richard S Sutton. Predictive representations of state. *Advances in Neural Information Processing Systems*, 14, 2001.

Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pp. 152–166. Springer, 2004. doi: 10.1007/978-3-540-30206-3_12.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Dung Phan, Nicola Paoletti, Timothy Zhang, Radu Grosu, Scott A. Smolka, and Scott D. Stoller. Neural state classification for hybrid systems. In *Automated Technology for Verification and Analysis (ATVA)*, Lecture Notes in Computer Science, pp. 422–440. Springer, 2018. doi: 10.1007/978-3-030-01090-4_25.

Ralf Römer, Adrian Kobras, Luca Worbis, and Angela P. Schoellig. Failure prediction at runtime for generative robot policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2510.09459.

Matthew Rosencrantz, Geoff Gordon, and Sebastian Thrun. Learning low dimensional predictive representations. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pp. 88, 2004.

Gaia Saveri, Laura Nenzi, Luca Bortolussi, and Jan Křetínský. stl2vec: Semantic and interpretable vector representation of temporal logic. In *European Conference on Artificial Intelligence (ECAI)*, Frontiers in Artificial Intelligence and Applications, 2024. doi: 10.3233/FAIA240638.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 9355–9366, 2021.

Oliver Schön, Licio Romao, and Sadegh Soudjani. Safety certification is classification. *arXiv preprint arXiv:2605.06087*, 2026.

Satinder Singh, Michael R. James, and Matthew R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Conference on Uncertainty in Artificial Intelligence*, pp. 512–519, 2004.

Satinder Singh, Michael James, and Matthew Rudary. Predictive state representations: A new theory for modeling dynamical systems. *arXiv preprint arXiv:1207.4167*, 2012.

Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew Foutter, Edward Schmerling, and Marco Pavone. Real-time anomaly detection and reactive planning with large language models. In *Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.114.

Antonina Skurka, Luko van der Maas, Sebastian Junges, and Hazem Torfah. Learning robust Markov models for safe runtime monitoring. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2026. arXiv:2602.14987.

Scott D. Stoller, Ezio Bartocci, Justin Seyster, Radu Grosu, Klaus Havelund, Scott A. Smolka, and Erez Zadok. Runtime verification with state estimation. In *International Conference on Runtime Verification (RV)*, volume 7186 of *LNCS*, pp. 193–207, 2011.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10183–10192, 2021.

Michael Thon and Herbert Jaeger. Links between multiplicity automata, observable operator models and predictive state representations: a unified learning framework. *The Journal of Machine Learning Research*, 16(1):103–147, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

Kartik Waghmare and Johanna Ziegel. Proper scoring rules for estimation and forecast evaluation. *Annual Review of Statistics and Its Application*, 13, 2025.

David Wingate, Vishal Soni, Britton Wolfe, and Satinder Singh. Relational knowledge with predictive state representations. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2035–2040, 2007.

Weiqiu You, Simeng Sun, and Mohit Iyyer. Hard-coded Gaussian attention for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 7689–7700, 2020. doi: 10.18653/v1/2020.acl-main.687.

Honghua Zhang, Meihua Dang, Nanyun Peng, and Guy Van den Broeck. Tractable control for autoregressive language generation. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2304.07438.

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck, and Nanyun Peng. Adaptable logical control for large language models. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2406.13892.

Lijun Zhang, Holger Hermanns, and David N. Jansen. Logic and model checking for hidden Markov models. In *International Conference on Formal Techniques for Networked and Distributed Systems (FORTE)*, volume 3731 of *LNCS*, pp. 98–112, 2005.

Enshen Zhou, Qi Su, Cheng Chi, Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and He Wang. Code-as-monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. arXiv:2412.04455.

A Additional Related Work

Abstraction-based model checking verifies temporal logic properties against an explicit model of the dynamics, at a cost that grows with the state space (Baier & Katoen, 2008). Under partial observability, the state becomes a belief over hidden states, and verification remains largely intractable. Bouton et al. (2020) combine a known partially observable model with a property automaton whose labels are not observed. They synthesize a policy that maximizes the satisfaction probability, whereas we predict this probability from data.

A Data-Driven Alternative. We predict satisfaction probabilities from observed trajectories, with formal guarantees and without committing to a model class. Observable operator models (Jaeger, 1997) and *predictive state representations* (PSRs) (Littman & Sutton, 2001; Singh et al., 2004) represent a system through predictions of future events, and Thon & Jaeger (2015) unify these representations with multiplicity and weighted automata. We follow the estimation approach of the Hilbert-space embedding PSRs of Boots et al. (2013) with a linear kernel.

Predicting a Property's Probability from an Observation History. Stoller et al. (2011) estimate the probability that a program run satisfies a temporal property, using a hidden Markov model of the program to fill sampling-induced gaps in the observed trace. Their estimate concerns the execution already completed, whereas ours concerns future behavior.

Skurka et al. (2026) predict the probability of violating a property within a bounded horizon given an observed prefix, assuming a state space and transition structure known from domain knowledge and learning only transition probabilities as intervals from state-labeled paths. We learn from observation traces annotated with propositions, posit no state space, and provide a finite-sample, distribution-free bound.

Reuse of One Fit Across Properties. Cairoli et al. (2025) reuse a property-independent generative trajectory model for monitoring, but require runtime sampling and a recalibration for each new property. Lindemann et al. (2023) also reuse a trajectory predictor, and their indirect method reuses its calibration across properties with the same horizon, which makes its certificate more reusable than ours at the cost of distributing the error budget over the steps of the horizon. Cairoli et al. (2023) train one monitor per property and combine monitors by Boolean connectives without retraining, but a new temporal operator requires a new monitor. None of these methods returns the conditional satisfaction probability given an observation history under partial observability.

Prediction Across Unseen Formulas. Bortolussi et al. (2022) predict satisfaction probabilities for unseen formulas by regression over formula space, and Saveri et al. (2024) refine the formula representation used by this regression. Both predict the marginal satisfaction probability under a model's initial distribution, without observations or a runtime monitor.

Weight Vectors Paired with a State by One Inner Product. Our decoder pairs a property-dependent, history-independent weight vector with a predictive state through one inner product. Zhang et al. (2005) pair a belief with a vector of satisfaction probabilities for a known latent-state model. Babaee et al. (2018); Babaee (2019) learn a hidden Markov model and precompute, for each latent state and property-automaton state, the probability of satisfying the property over the remaining horizon. These methods require an enumerated latent state space, whereas ours uses a predictive state without a state model. In Babaee et al. (2018), the automaton reads observed symbols and is tracked online, whereas ours reads unobserved latent labels and is marginalized offline.

For constrained decoding of language models, Zhang et al. (2023; 2024) compute the probability that a hidden Markov model completes a prefix into an accepted string, under keyword constraints and a property automaton, respectively. Unlike ours, these methods operate over a known latent state space and observed tokens. Wingate et al. (2007) also predict tests from a PSR by one inner product, as we do, but only for a restricted class of tests and without learning the PSR. Boots et al. (2011) evaluate value functions on a learned PSR as a maximum over linear functionals obtained by value iteration, rather than as one functional compiled from the syntax of a property.

Hoxha et al. (2026) construct a representation for predicting the *robustness* of past-time STL formulas that is minimal relative to monotone, 1-Lipschitz decoders and a finite atomic dictionary. We predict future-time satisfaction *probabilities* with a *linear* decoder, which allows the sum over futures to be collapsed offline.

Linear Attention and Recurrence. Our decoder is an attention head in which the property supplies the query, the core tests are the keys, and their probabilities are the values (Table 2). Linear attention replaces the softmax so that attention can be computed recurrently with a fixed-size state (Katharopoulos et al., 2020; Schlag et al., 2021). Without normalization, as in Schlag et al. (2021), the read-out is one inner product with this state, which is the form of our decoder, with the predictive state as the state and the property's weight vector as the query. Our decoder has this form because satisfaction probabilities are linear in the predictive state, not as an approximation of softmax attention. Attention weights fixed before any input is seen also appear in hard-coded attention (You et al., 2020) and in the fixed random variant of Tay et al. (2021), but there one pattern serves every input, whereas our weights are compiled from the syntax of each property. Deep state-space models also summarize the history in a fixed-size state (Gu et al., 2022), but the coordinates of our state are probabilities of future events.

B Satisfaction and Horizon of a Formula

This appendix states two standard definitions used throughout the paper. *Satisfaction* of STL formulas is adapted from Maler & Nickovic (2004) to discrete time, with the conventional until, whose left operand need not hold at the witness. The *horizon* recursion of STL formulas is adapted from Dokhanchi et al. (2014), shifted by one so that it counts letters.

Satisfaction. Let $\boldsymbol{\omega} = \sigma_1\sigma_2\cdots\sigma_n \in \Sigma^n$ be a word and $i \in \{1,\dots,n\}$ a position index. For a property $\varphi \in \mathcal{F}(\mathrm{AP})$ define the satisfaction relation $(\boldsymbol{\omega}, i) \models \varphi$ inductively by

$$
\begin{aligned}
(\boldsymbol{\omega}, i) &\models p && \iff p \in \sigma_i,\\
(\boldsymbol{\omega}, i) &\models \neg\varphi && \iff (\boldsymbol{\omega}, i) \not\models \varphi,\\
(\boldsymbol{\omega}, i) &\models \varphi_1 \wedge \varphi_2 && \iff (\boldsymbol{\omega}, i) \models \varphi_1 \text{ and } (\boldsymbol{\omega}, i) \models \varphi_2,\\
(\boldsymbol{\omega}, i) &\models \Box_I\, \varphi && \iff (\boldsymbol{\omega}, j) \models \varphi \text{ for all } j \in (i + I) \cap \{1,\dots,n\},\\
(\boldsymbol{\omega}, i) &\models \Diamond_I\, \varphi && \iff (\boldsymbol{\omega}, j) \models \varphi \text{ for some } j \in (i + I) \cap \{1,\dots,n\},\\
(\boldsymbol{\omega}, i) &\models \varphi_1\, \mathsf{U}_I\, \varphi_2 && \iff \exists\, j \in (i + I) \cap \{1,\dots,n\} :\ (\boldsymbol{\omega}, j) \models \varphi_2\\
& && \qquad \text{and } (\boldsymbol{\omega}, k) \models \varphi_1 \text{ for all } k \in \{i,\dots,j-1\},
\end{aligned}
$$

where $i + I := \{i + \iota : \iota \in I\}$. We read words from their first letter and write $\boldsymbol{\omega} \models \varphi$ for $(\boldsymbol{\omega}, 1) \models \varphi$, which is the relation defining the language $\mathscr{L}(\varphi)$ of φ in Subsection 2.2. A trajectory is lifted to a word by the labeling map, i.e., $(\mathbf{x}, t) \models \varphi$ iff $\ell(x_{t:t+H-1}) \models \varphi$.

Horizon. The horizon $\mathrm{hor}(\varphi) \in \mathbb{N}$ of a property φ is computed recursively by

$$\begin{aligned} \mathrm{hor}(p) &= 1, & \mathrm{hor}(\neg\varphi) &= \mathrm{hor}(\varphi), \\ \mathrm{hor}(\varphi_1 \wedge \varphi_2) &= \max\{\mathrm{hor}(\varphi_1), \mathrm{hor}(\varphi_2)\}, & \mathrm{hor}(\Box_I \varphi) &= \max I + \mathrm{hor}(\varphi), \\ \mathrm{hor}(\Diamond_I \varphi) &= \max I + \mathrm{hor}(\varphi), & \mathrm{hor}(\varphi_1 \,\mathsf{U}_I\, \varphi_2) &= \max I + \max\{\mathrm{hor}(\varphi_1), \mathrm{hor}(\varphi_2)\}, \end{aligned}$$

which is finite because every interval $I \subset \mathbb{N}_0$ in the grammar of Subsection 2.2 is bounded. It counts the time steps a formula may depend on. The following lemma is used throughout the paper.

Lemma 1 (Bounded Lookahead)**.** *Let $\varphi \in \mathcal{F}(\mathrm{AP})$ and $H = \mathrm{hor}(\varphi)$. For every $n \geq H$ and every word $\boldsymbol{\omega} := \sigma_1, \sigma_2, \ldots, \sigma_n \in \Sigma^n$, satisfaction $\boldsymbol{\omega} \models \varphi$ is determined by the prefix $\sigma_1 \cdots \sigma_H$. In particular, $\mathscr{L}(\varphi) \cap \Sigma^n = \{\boldsymbol{\omega} := \sigma_1, \sigma_2, \ldots, \sigma_n \in \Sigma^n : \sigma_1 \cdots \sigma_H \in \mathscr{L}(\varphi) \cap \Sigma^H\}$.*

Proof. We prove the statement by induction on the structure of φ. Claim: for every i, $(\boldsymbol{\omega}, i) \models \varphi$ depends only on the letter sequence $\sigma_i \cdots \sigma_{i+\mathrm{hor}(\varphi)-1}$. For $\varphi = p$, where $p \in \mathrm{AP}$, this is immediate, as only the letter σ_i is read. Negation and conjunction inherit the claim, using $\max$ for the latter. For $\Box_I \varphi$ the positions inspected are $j \in i + I$, each reading $\sigma_j \cdots \sigma_{j+\mathrm{hor}(\varphi)-1}$, and $j \leq i + \max I$, so no letter beyond index $i + \max I + \mathrm{hor}(\varphi) - 1$ is read; the case of $\Diamond_I \varphi$ is identical. For $\varphi_1 \,\mathsf{U}_I\, \varphi_2$ the witness satisfies $j \leq i + \max I$ and the positions k inspected for φ_1 satisfy $k < j$, so the largest index read is at most $i + \max I + \max\{\mathrm{hor}(\varphi_1), \mathrm{hor}(\varphi_2)\} - 1$. Setting $i = 1$ gives the statement. □

Lemma 1 is what licenses the finite disjoint union in the proof of Theorem 1, where the satisfaction event anchored at $t + 1$ is decomposed over $\mathscr{L}(\varphi) \cap \Sigma^H$. Any upper bound on $\mathrm{hor}(\varphi)$ may be substituted for H without affecting correctness; a loose bound only enlarges the word set Σ^H that the compilation of Subsection 3.2 traverses offline. The recursion above is tight for $\Box_I$ and $\Diamond_I$, and overestimates by one step for U_I when $\mathrm{hor}(\varphi_1) > \mathrm{hor}(\varphi_2)$, since positions inspected for φ_1 are strictly below the witness j.

C Formal Problem Statements

We here provide the two formal problem statements that we address in this paper.

The Constructive Case. We first study the zero-shot monitoring problem in the case that the law μ from Subsection 2.1 is known. In this case, we can construct the monitor explicitly.

Problem 1 (Exact Representation)**.** Let the law μ from Subsection 2.1 be given. Construct an *encoder* $Z \colon \mathbb{O}^* \to \mathbb{R}^{d_z}$, with $z_t := Z(h_t)$ being a representation computed from the history $h_t := o_{0:t} \in \mathbb{O}^{t+1}$, and *decoders* $\{g_\varphi\}_{\varphi \in \mathcal{F}(\mathrm{AP})}$ such that, at any time $t \in \mathbb{N}_0$ and from the observed history h_t alone, we have that

(i) *(sufficiency)* $g_\varphi(z_t)$ equals the conditional satisfaction probability of φ,

$$p_\varphi(h_t) := \mathbb{P}_\mu\big((\mathbf{X}, t+1) \models \varphi \mid O_{0:t} = h_t\big), \tag{11}$$

i.e., $g_\varphi(z_t) \equiv p_\varphi(h_t)$ for every monitored property $\varphi \in \mathcal{F}(\mathrm{AP})$, where Z depends on the fragment $\mathcal{F}(\mathrm{AP})$ but not on any individual φ; and

(ii) *(bounded dimension) the latent's dimensionality d_z is fixed independently of t and φ, so that the size of the representation does not grow with the length of the history.*

Problem 1 asks for a prediction-sufficient statistic of fixed dimension (Definition 2), with the following demands: requirement (ii) rules out trivially recording the raw history in z_t, forcing the representation to discard information no formula in $\mathcal{F}(\mathrm{AP})$ can request; requirement (i) asks one representation to serve the whole fragment rather than one property at a time. Filtering a continuous-space POMDP, for instance by a particle filter, meets neither demand, as its belief over $\mathbb{X}$ is infinite-dimensional and must be recomputed for each φ. In contrast, the predictive state representation of Section 3 can

be compressed to the linear dimension of the system (Appendix D) and decoders compiled from syntax. Its coordinates are further reduced to a rank r chosen by the user (Section 4), so the bounded dimension d_z of requirement (ii) becomes a desired monitoring budget/trade-off set at training time.

The Learning-Based Case. With known law μ, the encoder's conditional-probability coordinates are available exactly, and the decoders recover $p_\varphi(h_t)$ in (11). With only the N trajectories of $\mathcal{D}$ in (2), we estimate those coordinates empirically and construct $\widehat{Z}$ and $\{\hat{g}_\varphi\}$ from these estimates. The equality in (11) is therefore attained only approximately for finite N, and the prediction error is the gap between the monitor's output $\hat{p}_\varphi(h_t)$ and the conditional satisfaction probability $p_\varphi(h_t)$.

A concrete trajectory reveals only one realization of a Bernoulli random variable with mean $p_\varphi(h_t)$, not the probability (11) itself. A distribution-free pointwise bound from a single realization per history is impossible without further assumptions (Barber, 2020). We therefore seek to certify a bin-conditional mean. For any measurable prediction map $\hat{p}_\varphi\colon \mathbb{O}^* \to [0,1]$ for (11) and a fixed partition $0 = \tau_0 < \cdots < \tau_B = 1$ of $[0,1]$, let $\mathcal{H}_b := \{h : \hat{p}_\varphi(h) \in [\tau_{b-1}, \tau_b)\}$ collect the histories predicted into bin b. The certified quantity is the bin-conditional mean satisfaction probability

$$p_{\varphi,b} \;:=\; \mathbb{E}_\mu\big[\,\mathbb{P}(\varphi \mid O_{0:t}) \;\big|\; O_{0:t} \in \mathcal{H}_b\,\big], \tag{12}$$

i.e., the mean satisfaction probability under μ among histories with similar predictions, not the probability at an individual history h_t.

Problem 2 (Estimation and Certification)**.** Let the law μ from Subsection 2.1 be unknown, with only data $\mathcal{D} \equiv \{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{cal}}\}$ in (2) and AP given. Construct from $\mathcal{D}_{\text{train}}$ estimates $\widehat{Z}$ and $\{\hat{g}_\varphi\}_{\varphi \in \mathcal{F}(\text{AP})}$ of the map and decoders in Problem 1, meeting its requirement (ii), and from $\mathcal{D}_{\text{cal}}$ a bound $\hat{p}^{\text{lb}}_{\varphi,\cdot} : \{0,\dots,B\} \to [0,1]$, such that

(i) *(accuracy)* the induced prediction $\hat{p}_\varphi(h_t) := \hat{g}_\varphi(\hat{z}_t)$, with $\hat{z}_t := \widehat{Z}(h_t)$, approximates the target $p_\varphi(h_t)$ of (11);

(ii) *(validity)* for every $\delta \in (0,1)$, the bound $\hat{p}^{\text{lb}}_{\varphi,\cdot}$ holds simultaneously across bins, i.e.,

$$\mathbb{P}_{\mu^{\otimes N_{\text{cal}}}}\Big(\hat{p}^{\text{lb}}_{\varphi,b} \;\le\; p_{\varphi,b} \quad \text{for all } b \in \{0,\dots,B\}\Big) \;\ge\; 1-\delta, \tag{13}$$

with $p_{\varphi,b}$ as in (12) and with no distributional assumption on μ.

Requirements (i) and (ii) are separately necessary. A monitor returning $\hat{p}^{\text{lb}}_\varphi \equiv 0$ satisfies (ii) and is useless, while (i) alone carries no guarantee at all. Section 4 answers Problem 2.

D Language-Induced Predictive State Representation

We first record the notion of sufficiency that PSRs realize, used in Problem 1.

Definition 2 (Prediction Sufficiency)**.** Let $(O_t, \Lambda_t)_{t\in\mathbb{N}_0}$ be a joint process under a law ν, where $O_t \in \mathbb{O}$ is observed and $\Lambda_t \in \mathcal{A}$ need not be. A measurable statistic $Z\colon \mathbb{O}^* \to \mathbb{R}^{d_z}$ is *prediction sufficient* if, for every $t \in \mathbb{N}_0$ and every event E concerning the sequence $\Lambda_{t+1}, \Lambda_{t+2}, \dots$, there is a measurable map $g_E\colon \mathbb{R}^{d_z} \to [0,1]$ satisfying

$$\mathbb{P}_\nu\big(E \mid O_{0:t}\big) \;=\; g_E\big(Z(O_{0:t})\big) \qquad \nu\text{-almost surely.} \tag{14}$$

Equation (14) states that compressing the history $O_{0:t}$ to $z := Z(O_{0:t})$ loses no information it provides on the unobserved future. It instantiates Akahira & Takeuchi (1980, Definition 2) for the singleton family $\{\nu\}$, stated through the existence of a measurable map on the statistic.[1] It places no assumption on the conditional law of $\Lambda_{t+1}, \Lambda_{t+2}, \dots$ given $O_{0:t}$.

Definition 3 (Predictive State Representation (PSR))**.** A *test* is a word $\tau \in \mathcal{A}^+$ naming the event $\{\Lambda_{t+1:t+|\tau|} = \tau\}$, whose conditional probability given $h_t := O_{0:t}$ is written $\mathbb{P}(\tau \mid h_t)$. Given a set

[1] Conditioning (14) on $Z(O_{0:t})$ gives $\mathbb{P}_\nu(E \mid O_{0:t}) = \mathbb{P}_\nu(E \mid Z(O_{0:t}))$ ν-almost surely, which expresses conditional independence of the future from the history given the statistic. The converse follows from the Doob–Dynkin lemma.

of q *core tests* $Q := \{\tau_i\}_{i=1}^{q}$, define their prediction vector as

$$\mathcal{Z}(h_t)^\top := \big[\,\mathbb{P}\big(\tau_i \mid h_t\big)\big]_{i=1}^{q}. \tag{15}$$

Then, $\mathcal{Z}$ is a ***PSR*** iff it is prediction sufficient for the class of all tests, and a ***linear PSR*** if for every test $\tau \in \mathcal{A}^+$ there exists a vector $f_\tau \in \mathbb{R}^q$ with $\mathbb{P}(\tau \mid h_t) = f_\tau^\top \mathcal{Z}(h_t)$ for every $t \in \mathbb{N}_0$ and μ-almost every history h_t.

Definition 3 posits core tests but does not say how to select them, and finding a minimal set, the *discovery problem*, is generally difficult for continuous-space processes (Thon & Jaeger, 2015). In a language-induced PSR, the core tests are instead fixed by the semantics of STL (Subsection 3.1).

Adapting the *system-dynamics matrix* of Singh et al. (2004), let D have entries $D_{h,\boldsymbol{\omega}} := \mathbb{P}(\boldsymbol{\omega} \mid h)$, with observation histories $h \in \mathbb{O}^*$ as rows and latent words $\boldsymbol{\omega} \in \Sigma^+$ of arbitrary length as columns (Figure 4a). Its columns stay symbolic, so that a property can select coordinates $\boldsymbol{\omega} \in Q$ by its syntax, whereas its rows index histories of arbitrary length that can neither be enumerated nor observed repeatedly (Thon & Jaeger, 2015), which is why Section 4 regresses the row axis on a history feature. The set Q is intentionally non-minimal, as retaining all tests lets more data enter the estimate even when $|Q|$ greatly exceeds $\operatorname{rank}(D)$ (Thon & Jaeger, 2015, Section 4.2); we defer dimensionality reduction to the *characterizer* of Subsection 4.1.

Extended Histories. The rows of D are observation histories, whereas its columns are latent words. Conditioning on a latent letter σ therefore does not return a row of D: the vector $[\mathbb{P}(\boldsymbol{\omega} \mid h, \sigma)]_{\boldsymbol{\omega} \in Q}$ is a prediction given the event $\{O_{0:t} = h,\ \Lambda_{t+1} = \sigma\}$, which no observation history realizes. In the classical case $\mathcal{A} = \mathbb{O}$, this vector is $\mathcal{Z}(h\sigma)$ and D is closed under conditioning. We make the missing rows explicit.

Definition 4 (Extended System-Dynamics Matrix)**.** Consider the joint process $(O_t, \Lambda_t)_{t \in \mathbb{N}_0}$ on $(\mathbb{O} \times \Sigma)^{\mathbb{N}_0}$. An *extended history* is a pair (h, u) of an observation history $h \in \mathbb{O}^{t+1}$ and a latent prefix $u \in \Sigma^k$, $k \in \mathbb{N}_0$, and denotes the cylinder event

$$E_{h,u} := \{O_{0:t} = h,\ \Lambda_{t+1:t+k} = u\} = \big(\{h\} \times \Sigma^{t+1}\big) \times \big(\mathbb{O}^k \times \{u\}\big) \subseteq (\mathbb{O} \times \Sigma)^{t+k+1}. \tag{16}$$

A test $\boldsymbol{\omega} \in \Sigma^+$ denotes the cylinder event $E_{\boldsymbol{\omega}} := \mathbb{O}^{|\boldsymbol{\omega}|} \times \{\boldsymbol{\omega}\}$, appended after $E_{h,u}$. The *extended system-dynamics matrix* $\widetilde{D}$ has rows indexed by extended histories, columns indexed by tests, and entries

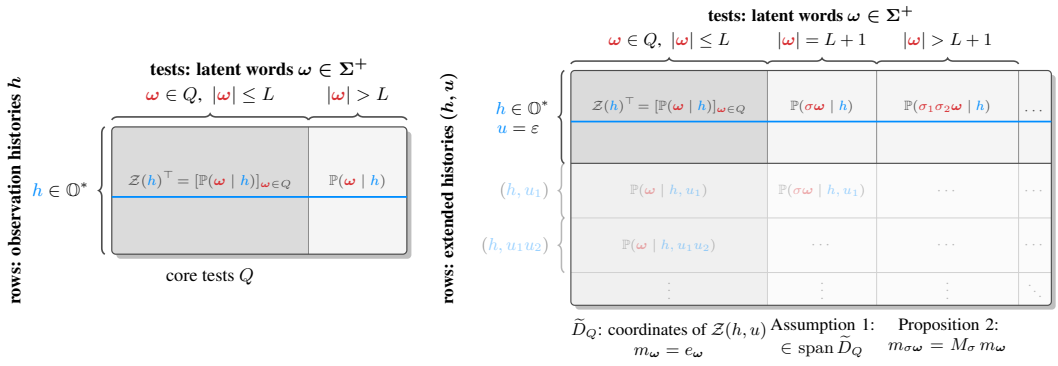
$$\widetilde{D}_{(h,u),\boldsymbol{\omega}} := \mathbb{P}\big(\Lambda_{t+k+1:t+k+|\boldsymbol{\omega}|} = \boldsymbol{\omega} \;\big|\; O_{0:t} = h,\ \Lambda_{t+1:t+k} = u\big) =: \mathbb{P}(\boldsymbol{\omega} \mid h, u), \tag{17}$$

defined for μ-almost every h and every u with $\mathbb{P}(u \mid h) > 0$. Its Q-columns form $\widetilde{D}_Q$, and $\mathcal{Z}(h, u) := [\mathbb{P}(\boldsymbol{\omega} \mid h, u)]_{\boldsymbol{\omega} \in Q}$ is the row (h, u) of $\widetilde{D}_Q$. The rows with $u = \varepsilon$ form D, and $\mathcal{Z}(h, \varepsilon) = \mathcal{Z}(h)$.

Both index sets of $\widetilde{D}$ are marginal events of the product alphabet $\mathbb{O} \times \Sigma$. A row event fixes the observations and leaves the labels free on its first block, then fixes the labels and leaves the observations free on its second; a column event fixes labels only. For $\mathcal{A} = \mathbb{O}$ every row event is again a history, so $\widetilde{D} = D$. Here D is the $u = \varepsilon$ block of $\widetilde{D}$ (Figure 4b), and only this block is indexed by objects available at test time. The remaining rows are indexed by latent prefixes, which the labeled data (2) provide but the runtime observations do not. Weighted transducers and IO-OOMs also split the alphabet into two factors (Thon & Jaeger, 2015), but there both factors are observed; we are not aware of a formulation in which one factor is latent at test time.

Assumption 1 (Characteristic Core Tests)**.** *For every letter $\sigma \in \Sigma$ and every word $\boldsymbol{\omega} \in \Sigma^L$, the column $\sigma\boldsymbol{\omega}$ of $\widetilde{D}$ lies in the span of the columns of $\widetilde{D}_Q$. That is, there exists $m_{\sigma\boldsymbol{\omega}} \in \mathbb{R}^q$ with $\mathbb{P}(\sigma\boldsymbol{\omega} \mid h, u) = \mathcal{Z}(h, u)^\top m_{\sigma\boldsymbol{\omega}}$ for every extended history (h, u).*

Assumption 1 concerns the $|\Sigma|^{L+1}$ words that are one letter longer than the longest core test. Every shorter word is itself a core test, so $m_{\boldsymbol{\omega}} = e_{\boldsymbol{\omega}}$ for $\boldsymbol{\omega} \in Q$. Finite rank is a consequence, $\operatorname{rank} \widetilde{D} \le q$, and $r_* := \operatorname{rank} \widetilde{D}_Q$ is the linear dimension of the system. In the terminology of Thon & Jaeger (2015), Q is a set of *characteristic events*, and $\operatorname{rank} \widetilde{D} = \operatorname{rank} \widetilde{D}_Q$ follows from the assumption and Proposition 2. For $\mathcal{A} = \mathbb{O}$ the assumption is the classical requirement that Q contains a core



(a) System-dynamics matrix D, after Singh et al. (2004). Rows are observation histories $h \in \mathbb{O}^*$, which form a continuum since $\mathbb{O}$ is continuous. Columns are latent words $\omega \in \Sigma^+$. Conditioning arguments are shown in blue and predicted words in red. The row h restricted to the core tests Q is the predictive state $\mathcal{Z}(h)^\top$.

(b) Block structure of the extended system-dynamics matrix $\widetilde{D}$ of Definition 4. Rows are extended histories (h, u), grouped by the length of the latent prefix u; the block $u = \varepsilon$ is the matrix D of (a), whose rows Section 4 regresses on a history feature. Columns are latent words, grouped by length. The Q-columns are the coordinates of $\mathcal{Z}(h, u)$. Assumption 1 concerns the columns of length $L + 1$ only; all longer columns then lie in the same span by Proposition 2. For $\mathcal{A} = \mathbb{O}$ every row is a history and $\widetilde{D} = D$.

Figure 4: The system-dynamics matrix and its extension to latent prefixes.

test set (Littman & Sutton, 2001; Singh et al., 2004). It places no condition on the rows of $\widetilde{D}$ and assumes neither a finite hidden state space nor a finite dimension of the joint process. Words of length $L + 1$ are also the longest that Section 4 fits from data. Whether it holds is decided by the choice of propositions AP and the test length L.

Proposition 2 (Linear Structure). *Under Assumption 1 there exist vectors $m_\omega \in \mathbb{R}^q$, $\omega \in \Sigma^+$, with $m_\omega = e_\omega$ for $\omega \in Q$, such that for every extended history (h, u) of Definition 4:*

(i) every test is decoded linearly,

$$\mathbb{P}(\omega \mid h, u) = \mathcal{Z}(h, u)^\top m_\omega \qquad \text{for all } \omega \in \Sigma^+; \tag{18}$$

(ii) conditioning on a latent letter $\sigma \in \Sigma$ with $\mathbb{P}(\sigma \mid h, u) > 0$ is a normalized linear map,

$$\mathcal{Z}(h, u\sigma)^\top = \frac{\mathcal{Z}(h, u)^\top M_\sigma}{\mathcal{Z}(h, u)^\top m_\sigma}, \qquad M_\sigma := \left[\, m_{\sigma\tau_1} \ \cdots \ m_{\sigma\tau_q} \,\right] \in \mathbb{R}^{q \times q}; \tag{19}$$

(iii) the weight vectors factorize,

$$m_{\sigma\omega} = M_\sigma\, m_\omega \qquad \text{for all } \sigma \in \Sigma,\ \omega \in \Sigma^+. \tag{20}$$

The family $\{m_\omega\}_{\omega \in \Sigma^+}$ is unique up to the null space $\{v \in \mathbb{R}^q : \widetilde{D}_Q v = 0\}$, and unique if $r_* := \operatorname{rank} \widetilde{D}_Q = q$. In particular, restricted to observation histories (where $u = \varepsilon$), $\mathcal{Z}$ is a classical linear PSR in the sense of Definition 3.

Proof. For $\omega \in Q$ set $m_\omega := e_\omega$; for $|\omega| = L + 1$ take m_ω from Assumption 1. This gives (18) for $|\omega| \le L + 1$. For (19), coordinate i of $\mathcal{Z}(h, u\sigma)$ is $\mathbb{P}(\tau_i \mid h, u\sigma) = \mathbb{P}(\sigma\tau_i \mid h, u)/\mathbb{P}(\sigma \mid h, u)$ by the definition of conditional probability, since $\{\Lambda_{t+1:t+k+1} = u\sigma,\ \Lambda_{t+k+2:t+k+1+|\tau_i|} = \tau_i\} = \{\Lambda_{t+1:t+k} = u,\ \Lambda_{t+k+1:t+k+1+|\tau_i|} = \sigma\tau_i\}$; numerator and denominator are $\mathcal{Z}(h, u)^\top m_{\sigma\tau_i}$ and $\mathcal{Z}(h, u)^\top m_\sigma$ because $|\sigma\tau_i| \le L + 1$. We now define $m_{\sigma\omega} := M_\sigma m_\omega$ for $|\omega| \ge L + 1$, which is (20), and prove (18) by induction on $|\omega|$. For $\omega \in Q$, $M_\sigma e_\omega$ is the column $m_{\sigma\omega}$, so (20) holds for these as well. Let $|\omega| \ge L + 1$ and assume (18) for ω at every extended history. If $\mathbb{P}(\sigma \mid h, u) = 0$, then $\mathbb{P}(\sigma\omega \mid h, u) = 0$ and $\mathcal{Z}(h, u)^\top M_\sigma m_\omega = \sum_i (m_\omega)_i\, \mathbb{P}(\sigma\tau_i \mid h, u) = 0$, as

$\mathbb{P}(\sigma\tau_i \mid h, u) \le \mathbb{P}(\sigma \mid h, u)$. Otherwise,

$$\mathbb{P}(\sigma\boldsymbol{\omega} \mid h, u) = \mathbb{P}(\sigma \mid h, u)\,\mathbb{P}(\boldsymbol{\omega} \mid h, u\sigma) = \mathbb{P}(\sigma \mid h, u)\,\mathcal{Z}(h, u\sigma)^\top m_{\boldsymbol{\omega}} = \mathcal{Z}(h, u)^\top M_\sigma\, m_{\boldsymbol{\omega}},$$

by the induction hypothesis at $(h, u\sigma)$ and (19). Uniqueness up to the null space follows since two families satisfying (18) differ by vectors annihilated by every row of $\widetilde{D}_Q$. □

The matrix M_σ conditions on the marginal event $E_\sigma = \mathbb{O} \times \{\sigma\}$, and (19) is Bayes' rule for that event. It cannot be run on observations at test time, since Λ_{t+1} is not observed; Section 4 updates the history feature instead. Iterating estimated operators would in any case compound their error and divide by an estimate that may vanish for a rare letter. The columns $m_{\sigma\tau_i}$ that form M_σ are the length-$(L+1)$ columns of Assumption 1 and are estimated by regression in Subsection 4.1. Unlike POMDP parameters, entries of m_σ and M_σ may be negative, making linear PSRs strictly more general (Singh et al., 2012).

Proof of Theorem 1. Since $\mathrm{hor}(\varphi) = H$, whether $\boldsymbol{\omega} \models \varphi$ is decided by the first H letters of $\boldsymbol{\omega}$, so the satisfaction event anchored at $t+1$ is the finite disjoint union $\sqcup_{\boldsymbol{\omega} \in \mathscr{L}(\varphi) \cap \Sigma^H} \{\ell(X_{t+1:t+H}) = \boldsymbol{\omega}\}$. Conditioning on $O_{0:t} = h_t$ and using finite additivity gives, for μ-almost every h_t, $\mathbb{P}(\varphi \mid h_t) = \sum_{\boldsymbol{\omega} \in \mathscr{L}(\varphi) \cap \Sigma^H} \mathbb{P}(\boldsymbol{\omega} \mid h_t)$. Each summand is a test, so Definition 3 applies through (18), and $m_{\boldsymbol{\omega}}$ factorizes by (20); collecting terms yields (6). □

E Decoding

E.1 Attention Correspondence

Table 2 expands the attention correspondence of Subsection 3.1.

Table 2: **Attention-like decoding.** The decoding in (6) is a key–value lookup on the predictive state. Because query and keys are written in the same language, addressing is exact and the weights are input(/observation)-agnostic.

Object	Standard attention (Vaswani et al., 2017)	This paper
Property	instruction the user provides	$\varphi \in \mathcal{F}(\mathrm{AP})$; the monitored property
Tokens	chunks of the input sequence	core tests $\boldsymbol{\omega} \in Q$; the coordinates of $\mathcal{Z}$
Keys	learned projection of a token	the core tests $\boldsymbol{\omega} \in Q$; coordinate selectors $e_{\boldsymbol{\omega}} \in \mathbb{R}^q$
Values	learned projection of a token	$\mathbb{P}(\boldsymbol{\omega} \mid h_t)$, $\boldsymbol{\omega} \in Q$; the entries of $\mathcal{Z}(h_t)$
Query	learned projection of a token	$f_\varphi \in \mathbb{R}^q$, compiled from the syntax of φ via (7)
Score	query–key inner product	$f_\varphi^\top e_{\boldsymbol{\omega}} = f_{\varphi,\boldsymbol{\omega}}$; exact since query & keys share Σ
Mask	causal or padding	$\mathscr{L}(\varphi) \cap \Sigma^H$; fixed by formal syntax
Output	values summed with normalized scores	$\mathbb{P}(\varphi \mid h_t) = \sum_{\boldsymbol{\omega} \in Q} f_{\varphi,\boldsymbol{\omega}}\, \mathbb{P}(\boldsymbol{\omega} \mid h_t) = f_\varphi^\top \mathcal{Z}(h_t)$

E.2 Compiled vs. Soft Addressing

Factorization. Any partition of a word into blocks of length at most L returns the same $m_{\boldsymbol{\omega}}$ for exact operators, whereas least-squares estimates yield different numerical decoders across partitions. Longer blocks require fewer operator applications but are estimated from rarer observations and therefore have higher estimation variance. We fix the *letter-wise factorization*, which peels single letters down to the final letter and is therefore unique; this choice performed well in our empirical experiments, and the remainder of this subsection explores more advanced factorizations.

Compilation selects how each accepted word is factored into stored blocks. For $\boldsymbol{\omega} \in \mathscr{L}(\varphi)$, let $\Pi(\boldsymbol{\omega})$ contain its factorizations $\pi = (u_1, \ldots, u_k)$ into blocks $u_j \in Q$, each yielding $m_\pi := M_{u_1} \cdots M_{u_{k-1}} m_{u_k}$. The general compiled query is

$$f_\varphi(h) \;:=\; \sum_{\boldsymbol{\omega} \in \mathscr{L}(\varphi) \cap \Sigma^H} \;\sum_{\pi \in \Pi(\boldsymbol{\omega})} \alpha_\pi(h)\, m_\pi, \qquad \alpha_\pi(h) \ge 0, \qquad \sum_{\pi \in \Pi(\boldsymbol{\omega})} \alpha_\pi(h) = 1. \tag{21}$$

For exact operators, every factorization m_π equals $m_{\boldsymbol{\omega}}$ by (20), so (21) holds independent of the choice of α. For estimated operators, the m_π are computed based on varying amounts of training data, and α provides soft addressing over factorizations according to their reliability. If α depends on h, addressing is computed at runtime, making (21) an attention head in the usual sense. This paper uses the indicator of the letter-wise factorization, i.e., $\alpha_\pi(h) = \mathbf{1}_{\pi=(\sigma_1 \dots \sigma_H)}$, where $\boldsymbol{\omega} = \sigma_1 \cdots \sigma_H$, independent of h, so (21) reduces to (6) and compiles offline. This choice is appropriate because the labels in (2) are assumed to be measured exactly, and letters are the most frequently observed blocks. Soft addressing also applies when the property and the predictive state no longer share an exact language, e.g., when propositions carry perception-module confidence scores instead of ground-truth labels, or the property contains fuzzy or learned propositions. The keys then become soft, and history-dependent α restores runtime matching. Proposition 1 is unaffected, as it requires only a $[0, 1]$-valued prediction map that is measurable and independent of the calibration data.

E.3 More on Decoders

Windows Anchored in the Past. The decoder (6) is exact because φ is specified entirely on the future $\ell(X_{t+1:t+H})$. A formula whose window has partly elapsed additionally requires the distribution of the residual $\varrho_\tau \in \mathcal{R}_\tau$ reached in the meantime. This depends on the *past* latent labels $\ell(X_{s:t})$ and is therefore a smoothing rather than a prediction problem. An off-the-shelf smoother over $\mathcal{R}_\tau$ provides this distribution, for which the residual structure of Subsection 3.2 supplies the state space. We do not pursue this extension, since its guarantees inherit those of the smoother. Whether a similar attention mechanism can be developed for smoothing remains open.

General Decoders. The decoding in (6) can be generalized beyond satisfaction probabilities. Write $W := \ell(X_{t+1:t+H}) \in \Sigma^H$ for the future label word. For *any* $g\colon \Sigma^H \to \mathbb{R}$,

$$\mathbb{E}\big[g(W) \mid h_t\big] \;=\; \mathcal{Z}(h_t)^\top f_g, \qquad f_g \;:=\; \sum_{\boldsymbol{\omega} \in \Sigma^H} g(\boldsymbol{\omega})\, m_{\boldsymbol{\omega}}, \tag{22}$$

and (6) is the case where $g = \mathbf{1}(\boldsymbol{\omega} \models \varphi)$. Given our assumed data (2), the only requirement on g is that it be a function of W; no further regularity is needed, as Σ^H is finite and the sum is therefore finite. Regularity instead governs the cost of *compiling* f_g. Note that our calibration does not apply the same way as presented since (27) is then no longer a Bernoulli realization. Proposition 1 nevertheless extends to any target bounded in $[0, s]$, with $\hat{\pi}_b$ its within-bin sample mean and w_b in (10) scaled by s, because Hoeffding's inequality requires only boundedness and no property of binary outcomes. We do not pursue these instantiations here.

F Estimating the Representation

F.1 Predictive-State Regression

Let $\xi\colon \mathbb{O}^* \to \mathbb{R}^d$ be a *history feature* map and $k\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a positive semi-definite kernel (Berlinet & Thomas-Agnan, 2004). We train the PSR on the training split of N_{train} labeled trajectories in (2), for which only ground truth labels are required and no access to the latent state is assumed. Since ξ_t is a deterministic function of the observations, the state carries no accumulated estimation error along the trajectory (Schön et al., 2026), the regressor being applied once at decoding and the decoder f_φ composing $H - 1$ estimated operators once, offline. The feature map ξ is generally not injective on $\mathbb{O}^*$, so conditioning on $\xi(h_t)$ need not equal conditioning on h_t, an approximation analogous to the finite observation window of HSE-PSRs (Boots et al., 2013) that the certified bound of Subsection 4.2 absorbs. Let $h_i := o^{(i)}_{0:t_i}$ be the observed history of the i-th training sample, and let $\boldsymbol{\omega}^{(i)} := \ell(x^{(i)}_{t_i+1:t_i+L})$ be the corresponding latent word. Set $K_{ij} := k(\xi(h_i), \xi(h_j))$, $Y_{i\boldsymbol{\omega}} := \mathbf{1}\big(\boldsymbol{\omega}^{(i)}_{1:|\boldsymbol{\omega}|} = \boldsymbol{\omega}\big)$, and $k_\xi(h) := [k(\xi(h), \xi(h_i))]_{i=1}^{N_{\text{train}}}$. Kernel ridge regression can be used to compute estimates of the predictive state $\mathcal{Z}(h)$ as

$$\widehat{\mathcal{Z}}(h)^\top = k_\xi(h)^\top (K + \lambda N_{\text{train}} I)^{-1}\, Y, \tag{23}$$

following the estimation route of HSE-PSRs (Boots et al., 2013, Eqs. 28–30). A characteristic kernel permits raw sensor readings or vision embeddings without discretizing $\mathbb{O}$. For the linear kernel,

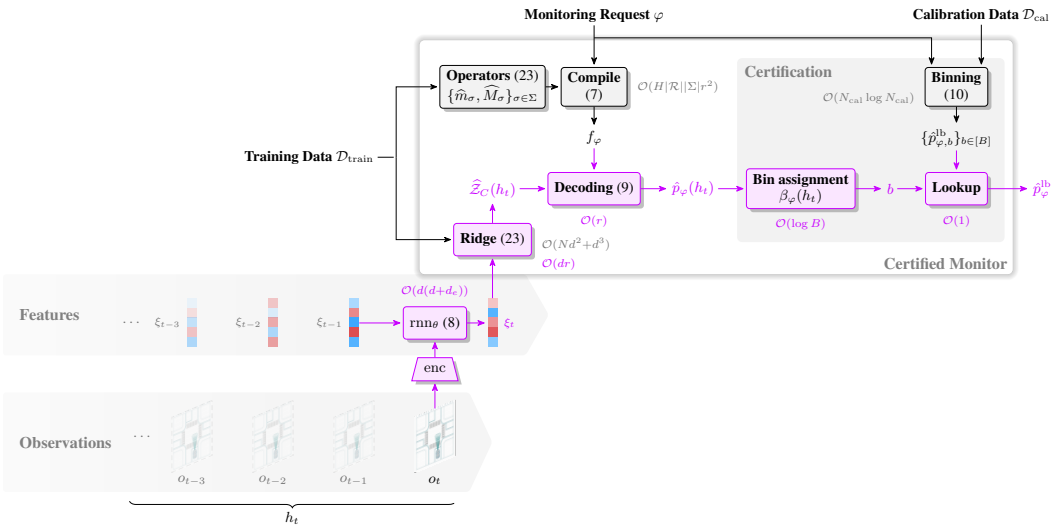


Figure 5: **Architecture overview.** Online components are purple; offline ones are grey. A recurrent history feature feeds the compressed predictive state and compiled readout; binning (10) certifies the bin-conditional mean. The regression equation shown is in Appendix F.1. Costs are those of the deployed regressor; its query cost does not depend on N.

offline fitting costs $\mathcal{O}(Nd^2 + d^3)$ instead of $\mathcal{O}(N^3)$, and compressed online regression costs $\mathcal{O}(dr)$ instead of $\mathcal{O}(N(d+r))$. Least-squares estimation can place the raw inner product in (9) outside $[0, 1]$. Applying the same clipping at calibration and query time preserves the guarantee of Subsection 4.2; clipping the estimate to the range of its probability target cannot increase its absolute error.

F.2 The Weighted Spectral Characterizer

The columns of $\widehat{D}_Q := [\,\widehat{\mathcal{Z}}(h_i)^\top]_{i=1}^{N_{\text{train}}} \in \mathbb{R}^{N_{\text{train}}\times|Q|}$ are strongly dependent, as a word and its prefixes are estimated from the same continuations, and fitting full-rank operators amplifies this noise.

This subsection gives the weighted decomposition behind the characterizer C of Subsection 4.1 and derives its diagonal weights.

The estimation variance of $\widehat{D}_Q$ varies across histories and words, so an unweighted SVD need not retain the most informative directions (Thon & Jaeger, 2015, Section 4.4.2). We therefore use the row- and column-weighted characterizer of (Thon & Jaeger, 2015, Algorithm 6). We derive the diagonal row and column weight matrices W_H and W_Q below, from an approximately factorized variance model for $\widehat{\mathcal{Z}}$. With these, let

$$W_H\,\widehat{D}_Q\,W_Q \;=\; \widetilde{\Theta}\,\widetilde{S}\,\widetilde{V}^\top, \qquad C \;:=\; \widetilde{V}_r^\top W_Q \;\in\; \mathbb{R}^{r\times q}, \qquad C^+ = W_Q^{-1}\widetilde{V}_r, \tag{24}$$

where $\widetilde{V}_r$ contains the leading r right singular vectors.[2] Equation (24) is (Thon & Jaeger, 2015, Algorithm 6) transposed to our orientation and reduces to the plain spectral characterizer when W_H and W_Q are proportional to the identity.

F.2.1 Deriving the Weights

Let $a(h) := (K + \lambda N_{\text{train}} I)^{-1} k_\xi(h)$. Approximating the conditional variance of each Bernoulli random variable $Y_{i\boldsymbol{\omega}}$ by its word-level average gives

$$\operatorname{Var}\big[\widehat{\mathcal{Z}}(h_i)_{\boldsymbol{\omega}}\big] \;\approx\; \underbrace{\|a(h_i)\|_2^2}_{=:\,\nu_i} \cdot \underbrace{\bar{\mathcal{Z}}_{\boldsymbol{\omega}}\big(1-\bar{\mathcal{Z}}_{\boldsymbol{\omega}}\big)}_{=:\,\varsigma_{\boldsymbol{\omega}}}, \qquad \bar{\mathcal{Z}}_{\boldsymbol{\omega}} := \tfrac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \widehat{\mathcal{Z}}(h_i)_{\boldsymbol{\omega}}, \tag{25}$$

[2] C^+ is a *right* inverse, $CC^+ = \widetilde{V}_r^\top \widetilde{V}_r = I_r$, and not the Moore–Penrose pseudoinverse of C unless $W_Q \propto I$; only $CC^+ = I_r$ is used below.

which factorizes variance across histories and words as (Thon & Jaeger, 2015, Algorithm 6) requires.

Both factors vanish on degenerate coordinates. A word never realized in the training sample has an identically zero column in Y, hence $\varsigma_{\boldsymbol{\omega}} = 0$. Before inversion, we floor the factors as

$$\varsigma_{\boldsymbol{\omega}} \leftarrow \max\{\varsigma_{\boldsymbol{\omega}}, 1/N_{\text{train}}\}, \qquad \nu_i \leftarrow \max\{\nu_i, 10^{-12}\}.$$

The former limits amplification of unobserved words, while the latter handles degenerate history weights. Thus,

$$W_H := [\operatorname{diag}(\nu_i)_{i=1}^{N_{\text{train}}}]^{-1/2}, \qquad W_Q := [\operatorname{diag}(\varsigma_{\boldsymbol{\omega}})_{\boldsymbol{\omega}\in Q}]^{-1/2}$$

are well defined and invertible.

F.2.2 Compressed Coordinates

We use the transformed PSR coordinates $\mathcal{Z}_C := C\mathcal{Z}$ (Rosencrantz et al., 2004; Boots et al., 2011) with the compressed operators

$$m_\sigma^C := (C^+)^\top m_\sigma, \qquad M_\sigma^C := (C^+)^\top M_\sigma\, C^\top, \qquad \sigma \in \Sigma, \tag{26}$$

which are estimated from (23) by ridge regression, with the targets of (23) extended to $\{\sigma\boldsymbol{\omega} : \sigma \in \Sigma,\ \boldsymbol{\omega} \in Q\}$, since (19) builds M_σ from the columns $m_{\sigma\boldsymbol{\omega}}$. The labeled words in (2) therefore reach $L + 1$ steps.

G Certification

G.1 Calibration Construction

Split conformal prediction (Vovk et al., 2005; Angelopoulos & Bates, 2022) is not suitable for our target, as a nonconformity-score quantile yields a prediction set for the binary *outcome* rather than an interval for the *probability* in (11), and for binary labels such prediction sets degenerate to the uninformative set $\{0, 1\}$ whenever the satisfaction probability is bounded away from 0 and 1 (Gupta et al., 2020, Section 2.2).

For a fixed property φ, suppress its subscript in $\hat{p}$, β, $\mathcal{H}_b$, and $\hat{p}_b^{\text{lb}}$. Let $\mathcal{D}_{\text{cal}} := \big\{h^{(i)} := o_{0:t_i}^{(i)},\, \ell(x_{0:T_i}^{(i)})\big\}_{i=1}^{N_{\text{cal}}} \sim \mu^{\otimes N_{\text{cal}}}$ be a calibration set of i.i.d. trajectories from (2), disjoint from the training set. The target probability $p^{(i)} := \mathbb{P}(\varphi \mid h^{(i)})$ is unobserved, and each trajectory instead provides

$$\text{sat}^{(i)} := \mathbf{1}\big(\ell(x_{t_i+1:t_i+H}^{(i)}) \in \mathscr{L}(\varphi)\big) \in \{0, 1\}, \qquad \text{with} \qquad \mathbb{E}\big[\text{sat}^{(i)} \,\big|\, h^{(i)}\big] = p^{(i)}, \tag{27}$$

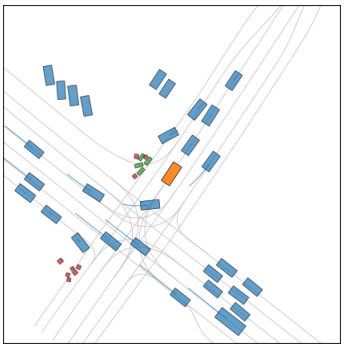
a realization of a Bernoulli random variable with mean $p^{(i)}$, against which the monitor's prediction $\hat{p}^{(i)} := \hat{p}_\varphi(h^{(i)})$ of (9) is calibrated. The bin edges $\tau_0, \ldots, \tau_B$, the assignment β, the sets $\mathcal{H}_b$ and $\mathcal{I}_b$, the counts N_b, and the estimator $\hat{\pi}_b$, width w_b, and bound $\hat{p}_b^{\text{lb}}$ of (10) are as defined in Subsection 4.2, with the last bin closed. Appendix H.3 treats the choice of edges, which must be computed on a split disjoint from the labels forming $\hat{\pi}_b$.

G.2 Proof of Proposition 1

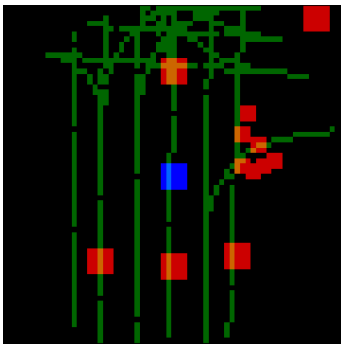
Proof. Write $\pi_b := \mathbb{E}_\mu[\,\text{sat} \mid O_{0:t} \in \mathcal{H}_b\,] = \mathbb{E}_\mu[\,\mathbb{P}(\varphi \mid O_{0:t}) \mid O_{0:t} \in \mathcal{H}_b\,]$ for the bin-conditional mean (12), the two expressions agreeing by the tower property because $\mathcal{H}_b$ is fixed once $\hat{p}$ and the bin edges are. Condition on the vector of bin memberships $(\beta(h^{(1)}), \ldots, \beta(h^{(N_{\text{cal}})}))$. This fixes $\mathcal{I}_b$ and N_b. Given $h^{(i)} \in \mathcal{H}_b$, the history $h^{(i)}$ is distributed as μ restricted to $\mathcal{H}_b$, so $\text{sat}^{(i)}$ is Bernoulli with mean π_b; independence across i is inherited from (2). Hence $\{\text{sat}^{(i)}\}_{i\in\mathcal{I}_b}$ are conditionally i.i.d. Bernoulli(π_b) with sample mean $\hat{\pi}_b$, for every nonempty bin. Bins with $\mathbb{P}_\mu(O_{0:t} \in \mathcal{H}_b) = 0$ are empty almost surely.

For each nonempty bin, the one-sided Hoeffding inequality gives

$$\mathbb{P}\big(\hat{\pi}_b - \pi_b \geq w_b \mid \beta(h^{(1)}), \ldots, \beta(h^{(N_{\text{cal}})})\big) \leq \exp(-2N_b w_b^2) = \tfrac{\delta}{B}.$$



(a) WOMD: agent tracks



(b) Waymo BEV: rendered scene



(c) Perception: camera frame

Figure 6: One example observation from each benchmark, at the three levels of the perception stack the monitor is evaluated on; the first two show the same WOMD scenario and time step. In all three cases, the atomic propositions are latent thresholds on physical quantities that the monitor never observes directly and must infer from the encoded observation history.

A conditional union bound over the at most B nonempty bins bounds the probability of any failure by δ. Taking the expectation over the memberships gives the same unconditional bound. On the complementary event, $\pi_b \geq \hat{\pi}_b - w_b$ for every nonempty bin, and since $\pi_b \geq 0$ also $\pi_b \geq \hat{p}_b^{\mathrm{lb}}$; for an empty bin $\hat{p}_b^{\mathrm{lb}} = 0 \leq \pi_b$. This is the statement in Proposition 1. Since the statement is simultaneous in b, it holds at the bin of any test history without an assumption on the test draw. □

H Experiments

H.1 Case-Study Protocol

This subsection details the training and evaluation protocol for Section 5. Every Brier value and skill in Table 1 is a mean over three seeds; the sample standard deviations over seeds of the benchmark means are 0.0016 on WOMD and 0.0038 on Waymo BEV, and 0.0052 for the monitor and 0.023 for the sampled baseline on Perception.

Observation and horizon details. WOMD provides agent tracks at 10 Hz over $70{,}541$ scenarios of 20 s, $39{,}135$ of them for training, with headway, time-to-collision, clearance, and pedestrian-clearance propositions ($|\Sigma| = 16$). Waymo BEV renders 64×64 views of agents, lane centerlines, and the ego vehicle from $29{,}066$ WOMD scenarios of 9 s, $20{,}000$ of them for training, encoded by a network trained separately from the monitor ($|\Sigma| = 8$). Perception provides camera frames at 5 Hz, with 319 training, 521 calibration, and 80 test scenarios ($|\Sigma| = 16$). The calibration split is the 319 segments of the frozen partition together with the 202 segments of the dataset's validation partition, which enter no fit. Property horizons range from 5 to 40 frames and are short relative to a scenario; shared query times make horizons comparable.

What the benchmarks do and do not cover. Referenced from the limitations of Section 5. On Perception the property selection keeps only three clearance reachability properties and one nested clearance property, so no Boolean or until property is evaluated on that benchmark; the ungated family of recovery and response properties below checks the Boolean and nested shapes but contains no until operator either. Every time-to-collision candidate is dropped. No history feature in our ablation lifts a candidate involving time to collision, the pedestrian, or the gap proposition above the gate, while every clearance property passes under all of them.

Splits. Each dataset is partitioned by scenario into training, validation, calibration, and test splits. WOMD uses hashed scenario identifiers; Perception is partitioned by drive so that each run remains in one split. The history feature is trained on training scenarios with early stopping on the validation split. Ridge penalty, characterizer rank, and the benchmark gate are selected on the validation split. Bin edges are placed on an edge share of the calibration scenarios whose outcomes are never read

(Subsection H.3); no training stage reads calibration or test data. The calibration split contains $13{,}975$ scenarios on WOMD, $5{,}000$ on Waymo BEV, and 521 on Perception, disjoint from its 319 training scenarios; a quarter of each is reserved for the bin edges. Every split uses one query time per scenario, drawn uniformly among the frames with at least 16 frames of history (8 on Perception) and 40 frames of future (20 on Perception) inside the scenario; the query times are shared by the monitor and all baselines and are the same for every horizon, so a formula reported at two horizons is scored on identical query times.

Encoder. The frozen encoder enc in (8) differs across the three benchmarks, which observe the scene at different levels of the perception stack (Figure 6). On each benchmark, enc ends in a projection onto principal components fitted on the training split, and its outputs, the rows, are standardized using training-split statistics.

- *WOMD* uses no neural network. Each frame is summarized by a hand-built 189-dimensional scene row computed from the agent tracks: ego speed, acceleration, and heading rate; the relative state, size, and type of the 16 nearest agents in the ego frame; and the state of and distance to the nearest traffic signal. enc returns 64 principal components of this row.
- *Waymo BEV* uses a learned network. The 64×64 bird's-eye views are encoded by a convolutional encoder, with temporal fusion over the last four frames, taken from Hoxha et al. (2026), which is trained to regress past-time robustness values and is kept frozen here. enc returns 32 principal components of its latent representation.
- *Perception* uses a pre-trained foundation model. The images of the five cameras are encoded by a frozen DINOv2-L (Oquab et al., 2024) at 378 pixels. enc returns 32 principal components of its tokens.

All monitors of Section 5, i.e., ours, the per-property reference, the RNN, and AR-64, read the same frozen encoder on each benchmark. Ours and the per-property reference also share the history feature ξ, whereas the RNN and AR-64 learn their own recurrent summaries of the encoded frames.

Propositions. The propositions threshold signed per-frame margins and hold when the margin is positive: on WOMD, time to collision to the agents ahead minus 2 s (p_τ), front gap minus two seconds of ego speed (p_h), clearance to any agent minus 1 m (p_c), and distance to the nearest pedestrian minus 5 m (p_p); on Waymo BEV, p_τ and p_h as on WOMD and the lane speed limit minus ego speed (p_s); on Perception, clearance to a labeled box minus 1 m (p_c), closing-speed time to collision to the lead vehicle minus 3 s (p_τ), pedestrian clearance minus 3 m (p_p), and longitudinal gap to the lead vehicle minus 5 m (p_g).

History feature. On the motion datasets, rnn_θ consists of two 128-unit gated recurrent units operating on the frozen rows, one over a 20-frame window and one over the full episode prefix. Their states are concatenated to $d = 256$ and trained on all training scenarios to predict realized length-two label words through a linear head, with the sampled baseline's schedule: at most 12 epochs of $300{,}000$ query times each, early stopping with patience 2 on the validation split. The feature at a query time uses only preceding frames; labels are used only during fitting. On Perception, the recurrent feature overfits at 319 segments, so we use the geometric recursion $\xi_{t+1} = 0.7\,\xi_t + \mathrm{enc}(o_{t+1})$, an instance of (8) without trained parameters. Two lookback scales are concatenated because on WOMD the headway property keeps improving as the state is extended from 20 frames to the whole episode, while time to collision saturates by 12 to 20 frames, so no single fixed-width summary serves both.

Regressor. We use $L = 2$. The reported regressor is ridge regression from $[\xi, 1]$ to the word indicators, the linear kernel in (23), with the penalty $\lambda \in \{10^{-3}, \ldots, 3\}$ and the characterizer rank $r \in \{4, 8, 16, 32\}$ selected jointly by the validation-split Brier score for the word indicators after projection; the ridge fit uses at most $3{,}000$ training query times.

Baselines. Section 5 compares against the *per-property reference*, the *RNN*, and the sampled autoregressive baseline *AR-64*, described in turn below. The per-property reference estimates $\mathbb{P}(\varphi \mid h)$ within the same architecture's accuracy limits. It uses (23) with one property's outcome sat replacing the word indicators, and the corresponding monitor's feature, penalty, and query times, restricted to query times with enough remaining frames to cover that property's horizon. The RNN is one gated

recurrent layer over the last W frames of the same standardized rows with a linear logit head, trained per property on $60{,}000$ training query times with the binary cross-entropy, Adam at learning rate $2 \cdot 10^{-3}$ and batch size 256, at most 40 epochs with early stopping at patience 3 on the validation split, and its hidden width and window selected per property on the validation split among $(64, 10)$, $(64, 20)$, and $(128, 20)$. The sampled autoregressive baseline uses a gated recurrent encoder over the last 20 rows, a $\tanh$ bridge to a 128-dimensional state, and a gated recurrent decoder emitting the next 40 letters (20 on Perception). It is trained with teacher forcing and cross-entropy on realized label words from all training scenarios, at most 12 epochs of $300{,}000$ query times with early stopping at patience 2 on the validation split, the schedule our history feature is trained with. At query time, it samples $K = 64$ futures per query time and reuses the samples for every property at every horizon up to the sampled one. The costs in Table 1 are measured one history per call, the regime a deployed monitor runs in.

Hardware and timing. WOMD and Perception were run on a node with two AMD EPYC 9355 processors and NVIDIA H200 GPUs, Waymo BEV on an AMD Ryzen Threadripper PRO 7975WX with NVIDIA RTX 6000 Ada GPUs. Costs on a CPU use NumPy and SciPy for the monitor and JAX restricted to the CPU for the baseline; costs on a GPU use JAX for the history feature and the baseline's encoder and sampler, batched over scenarios, while compilation, the linear regression, and the STL evaluation on samples run on the CPU. The per-frame costs of Table 1 are for one history per call with the history state carried from frame to frame, medians of repeated runs after a warm-up call on an otherwise idle node. The sampled baseline is timed with one sample at the longest horizon shared across horizons, the cheaper of the two readings.

Averaging. Table 1 averages over the included properties of Table 3 at all horizons and over three seeds; its inference timings use the properties with $H \leq 20$.

Ungated recovery and response properties. The recovery family $\Box_{[0,W-1]}(\neg p \to \Diamond_{[0,n]} p)$, which is equivalent to $\Box_{[0,W-1]}\Diamond_{[0,n]} p$, and its cross-proposition response form $\Box_{[0,W-1]}(\neg q \to \Diamond_{[1,n]} p)$ were compiled for every proposition pair, outer windows $W \in \{5, 10, 20\}$ and $n \in \{3, 5\}$, together with long instances at $W = 40$ with n up to 30 (compiled horizon up to 70), after the representation was fixed and without the property gate: 100 formulas on WOMD, 26 on Waymo BEV, and 104 on Perception. Rows whose validation base rate leaves $[0.02, 0.98]$ are flagged as degenerate, and rows whose realized outcome vector repeats one already listed as duplicates; none is dropped. Section 5 reports skill against the per-property reference on the scored rows inside $[0.05, 0.95]$ whose per-property reference improves on the constant predictor by at least 0.002 Brier. On Perception the sampled baseline is behind the decoder on every informative row.

Property gate. The three validation-split conditions of Section 5 are checked in a fixed candidate order. The gain condition is evaluated with a per-property reference fitted to the candidate's own realized outcomes, and not with our decoder. Candidates are gated at the main horizons only; the rows at $H \in \{15, 30, 40\}$ re-instantiate the syntactic patterns retained at $H = 20$.

H.2 The Properties

Table 3 lists every candidate property of the three benchmarks, including those the gate of Appendix H.1 excludes; we display the excluded ones for full transparency. The candidates are atomic reachability $\Diamond_{[0,H-1]}\neg p$, Boolean combinations under $\Diamond$ or $\Box$, until, and nested $\Box_{[0,H-4]}\Diamond_{[0,3]} p$, over the propositions p_h (headway), p_τ (time to collision), p_c (clearance), p_p (pedestrian clearance), p_g (gap), and p_s (speed). Each property is listed once, with its horizon $H = \mathrm{hor}(\varphi)$ in frames. All properties are scored on the same query times, each followed by at least 40 frames (20 on Perception). Every number reported in Section 5 is a mean over the included properties.

Table 3: **All candidate properties.** A property is included (✔) if, on the validation split, its base rate lies in $[0.05, 0.95]$, a regressor trained for this property alone improves on the constant base-rate predictor by at least 0.01 Brier, and its outcomes do not duplicate those of an included property (absolute correlation below 0.95). Where the constant predictor already performs as well as a property-specific regressor, the denominator of skill is near zero and the property would add noise, not information, to the skill estimate, so it is excluded (✘) for every monitor alike; the reason is given in its row. Brier scores and certificate columns are seed means on the test split, with $B = 10$ bins on the motion benchmarks and $B = 5$ on Perception; cert. LB is the mean over test histories of the certified lower bound $\hat{p}_b^{\mathrm{lb}}$ of the bin each history falls in, and agreement is the fraction of populated bins whose test-set frequency meets that bound, a diagnostic rather than a test of Proposition 1, which bounds the population bin mean.

Property	H	incl.	base rate	ours	reference	RNN	AR-64	cert. LB	agreement
WOMD (34 included of 50 candidates)									
$\Diamond_{[0,2]}\neg p_\tau$	3	✘	*correlation* 0.955 *with* $\Diamond_{[0,4]}\neg p_\tau$						
$\Diamond_{[0,2]}\neg p_c$	3	✘	*reference gain* $0.0062 < 0.01$						
$\Diamond_{[0,4]}\neg p_\tau$	5	✔	0.293	0.1188	0.1175	0.1202	0.1035	0.255	0.97
$\Diamond_{[0,4]}\neg p_h$	5	✔	0.399	0.0853	0.0846	0.0904	0.0766	0.369	1.00
$\Diamond_{[0,4]}(\neg p_\tau \wedge \neg p_h)$	5	✔	0.218	0.0814	0.0808	0.0843	0.0704	0.195	1.00
$\Box_{[0,4]}p_h \wedge \Box_{[0,4]}p_\tau$	5	✔	0.527	0.1083	0.1074	0.1113	0.0960	0.486	1.00
$p_h \,\mathsf{U}_{[0,4]}\, \neg p_\tau$	5	✔	0.267	0.1148	0.1143	0.1168	0.0999	0.232	1.00
$\Diamond_{[0,4]}\neg p_c$	5	✘	*reference gain* $0.0062 < 0.01$						
$\Diamond_{[0,4]}\neg p_p$	5	✘	*base rate* $0.016 \notin [0.05, 0.95]$						
$\Diamond_{[0,4]}\neg p_p \wedge \Box_{[0,4]}p_c$	5	✘	*base rate* $0.009 \notin [0.05, 0.95]$						
$p_c \,\mathsf{U}_{[0,4]}\, \neg p_p$	5	✘	*base rate* $0.016 \notin [0.05, 0.95]$						
$\Diamond_{[0,9]}\neg p_\tau$	10	✔	0.334	0.1342	0.1309	0.1360	0.1159	0.293	1.00
$\Diamond_{[0,9]}\neg p_h$	10	✔	0.428	0.0925	0.0898	0.0965	0.0822	0.396	0.93
$\Diamond_{[0,9]}(\neg p_\tau \wedge \neg p_h)$	10	✔	0.252	0.0931	0.0912	0.0963	0.0810	0.226	0.96
$\Box_{[0,9]}p_h \wedge \Box_{[0,9]}p_\tau$	10	✔	0.493	0.1169	0.1144	0.1216	0.1035	0.451	1.00
$p_h \,\mathsf{U}_{[0,9]}\, \neg p_\tau$	10	✔	0.284	0.1256	0.1241	0.1262	0.1088	0.246	1.00
$\Box_{[0,6]}\Diamond_{[0,3]}p_\tau$	10	✔	0.723	0.1186	0.1170	0.1204	0.1042	0.671	1.00
$\Box_{[0,6]}\Diamond_{[0,3]}p_h$	10	✔	0.613	0.0870	0.0855	0.0916	0.0770	0.567	1.00
$\Diamond_{[0,9]}\neg p_c$	10	✘	*reference gain* $0.0063 < 0.01$						
$\Diamond_{[0,9]}\neg p_p$	10	✘	*base rate* $0.018 \notin [0.05, 0.95]$						
$\Diamond_{[0,9]}\neg p_p \wedge \Box_{[0,9]}p_c$	10	✘	*base rate* $0.010 \notin [0.05, 0.95]$						
$p_c \,\mathsf{U}_{[0,9]}\, \neg p_p$	10	✘	*base rate* $0.018 \notin [0.05, 0.95]$						
$\Diamond_{[0,6]}\Box_{[0,3]}\neg p_c$	10	✘	*reference gain* $0.0056 < 0.01$						
$\Diamond_{[0,14]}\neg p_\tau$	15	✔	0.372	0.1465	0.1408	0.1451	0.1255	0.327	1.00
$\Diamond_{[0,19]}\neg p_\tau$	20	✔	0.402	0.1563	0.1485	0.1522	0.1330	0.356	0.97
$\Diamond_{[0,19]}\neg p_h$	20	✔	0.480	0.1076	0.0967	0.1006	0.0865	0.437	0.97
$\Diamond_{[0,19]}(\neg p_\tau \wedge \neg p_h)$	20	✔	0.306	0.1103	0.1059	0.1080	0.0946	0.275	0.96
$\Box_{[0,19]}p_h \wedge \Box_{[0,19]}p_\tau$	20	✔	0.432	0.1294	0.1212	0.1276	0.1088	0.395	1.00
$p_h \,\mathsf{U}_{[0,19]}\, \neg p_\tau$	20	✔	0.313	0.1441	0.1399	0.1427	0.1233	0.273	0.97
$\Box_{[0,16]}\Diamond_{[0,3]}p_\tau$	20	✔	0.648	0.1449	0.1405	0.1420	0.1260	0.598	1.00
$\Box_{[0,16]}\Diamond_{[0,3]}p_h$	20	✔	0.557	0.1018	0.0936	0.0974	0.0839	0.521	0.97
$\Diamond_{[0,19]}\neg p_c$	20	✘	*reference gain* $0.0067 < 0.01$						
$\Diamond_{[0,19]}\neg p_p$	20	✘	*base rate* $0.022 \notin [0.05, 0.95]$						
$\Diamond_{[0,19]}\neg p_p \wedge \Box_{[0,19]}p_c$	20	✘	*base rate* $0.012 \notin [0.05, 0.95]$						
$p_c \,\mathsf{U}_{[0,19]}\, \neg p_p$	20	✘	*base rate* $0.021 \notin [0.05, 0.95]$						
$\Diamond_{[0,16]}\Box_{[0,3]}\neg p_c$	20	✘	*reference gain* $0.0062 < 0.01$						
$\Diamond_{[0,29]}\neg p_\tau$	30	✔	0.460	0.1737	0.1615	0.1623	0.1458	0.413	1.00
$\Diamond_{[0,29]}\neg p_h$	30	✔	0.521	0.1209	0.1005	0.0998	0.0867	0.473	1.00
$\Diamond_{[0,29]}(\neg p_\tau \wedge \neg p_h)$	30	✔	0.353	0.1260	0.1174	0.1168	0.1048	0.314	1.00
$\Box_{[0,29]}p_h \wedge \Box_{[0,29]}p_\tau$	30	✔	0.383	0.1384	0.1258	0.1270	0.1115	0.349	1.00
$p_h \,\mathsf{U}_{[0,29]}\, \neg p_\tau$	30	✔	0.338	0.1600	0.1541	0.1578	0.1367	0.297	1.00
$\Box_{[0,26]}\Diamond_{[0,3]}p_\tau$	30	✔	0.589	0.1629	0.1550	0.1552	0.1399	0.540	1.00
$\Box_{[0,26]}\Diamond_{[0,3]}p_h$	30	✔	0.512	0.1158	0.0972	0.0971	0.0847	0.480	1.00
$\Diamond_{[0,39]}\neg p_\tau$	40	✔	0.511	0.1841	0.1671	0.1680	0.1514	0.458	1.00
$\Diamond_{[0,39]}\neg p_h$	40	✔	0.550	0.1329	0.1052	0.1023	0.0889	0.501	1.00
$\Diamond_{[0,39]}(\neg p_\tau \wedge \neg p_h)$	40	✔	0.393	0.1388	0.1245	0.1216	0.1100	0.344	1.00
$\Box_{[0,39]}p_h \wedge \Box_{[0,39]}p_\tau$	40	✔	0.347	0.1433	0.1287	0.1289	0.1138	0.311	1.00

Property	H	incl.	base rate	ours	reference	RNN	AR-64	cert. LB	agreement
$p_h \mathsf{U}_{[0,39]} \neg p_\tau$	40	✔	0.357	0.1717	0.1640	0.1690	0.1461	0.317	1.00
$\Box_{[0,36]}\Diamond_{[0,3]} p_\tau$	40	✔	0.538	0.1753	0.1638	0.1622	0.1482	0.493	0.97
$\Box_{[0,36]}\Diamond_{[0,3]} p_h$	40	✔	0.482	0.1295	0.1028	0.1004	0.0871	0.447	1.00
Waymo BEV (34 included of 51 candidates)									
$\Diamond_{[0,2]} \neg p_\tau$	3	✘	*correlation* 0.957 *with* $\Diamond_{[0,4]} \neg p_\tau$						
$\Diamond_{[0,2]} \neg p_h$	3	✘	*correlation* 0.973 *with* $\Diamond_{[0,4]} \neg p_h$						
$\Diamond_{[0,4]} \neg p_\tau$	5	✔	0.405	0.1452	0.1438	0.1414	0.1356	0.325	1.00
$\Diamond_{[0,4]} \neg p_h$	5	✔	0.603	0.0887	0.0878	0.0877	0.0815	0.539	1.00
$\Diamond_{[0,4]} (\neg p_\tau \wedge \neg p_h)$	5	✔	0.343	0.1277	0.1268	0.1251	0.1194	0.279	1.00
$p_h \mathsf{U}_{[0,4]} \neg p_\tau$	5	✔	0.369	0.1394	0.1388	0.1367	0.1302	0.283	1.00
$\Diamond_{[0,4]} \neg p_s$	5	✘	*base rate* $0.042 \notin [0.05, 0.95]$						
$\Diamond_{[0,4]} \neg p_\tau \wedge \Box_{[0,4]} p_h$	5	✘	*base rate* $0.047 \notin [0.05, 0.95]$						
$\Box_{[0,4]} p_h \wedge \Box_{[0,4]} p_s$	5	✘	*correlation* 0.994 *with* $\Diamond_{[0,4]} \neg p_h$						
$\Diamond_{[0,4]} \neg p_h \wedge \Diamond_{[0,4]} \neg p_\tau$	5	✘	*correlation* 1.000 *with* $\Diamond_{[0,4]} (\neg p_\tau \wedge \neg p_h)$						
$p_\tau \mathsf{U}_{[0,4]} \neg p_h$	5	✘	*correlation* 0.969 *with* $\Diamond_{[0,4]} \neg p_h$						
$\Diamond_{[0,9]} \neg p_\tau$	10	✔	0.450	0.1584	0.1556	0.1540	0.1486	0.367	1.00
$\Diamond_{[0,9]} \neg p_h$	10	✔	0.644	0.0976	0.0938	0.0933	0.0870	0.566	1.00
$\Diamond_{[0,9]} (\neg p_\tau \wedge \neg p_h)$	10	✔	0.387	0.1413	0.1390	0.1391	0.1330	0.316	1.00
$p_h \mathsf{U}_{[0,9]} \neg p_\tau$	10	✔	0.382	0.1482	0.1473	0.1476	0.1396	0.292	1.00
$p_\tau \mathsf{U}_{[0,9]} \neg p_h$	10	✔	0.613	0.0991	0.0965	0.0951	0.0883	0.542	1.00
$\Box_{[0,6]}\Diamond_{[0,3]} p_\tau$	10	✔	0.613	0.1596	0.1563	0.1593	0.1516	0.552	1.00
$\Box_{[0,6]}\Diamond_{[0,3]} p_h$	10	✔	0.417	0.1004	0.0990	0.1012	0.0932	0.355	1.00
$\Diamond_{[0,9]} \neg p_s$	10	✘	*base rate* $0.049 \notin [0.05, 0.95]$						
$\Diamond_{[0,9]} \neg p_\tau \wedge \Box_{[0,9]} p_h$	10	✘	*base rate* $0.048 \notin [0.05, 0.95]$						
$\Box_{[0,9]} p_h \wedge \Box_{[0,9]} p_s$	10	✘	*correlation* 0.994 *with* $\Diamond_{[0,9]} \neg p_h$						
$\Diamond_{[0,9]} \neg p_h \wedge \Diamond_{[0,9]} \neg p_\tau$	10	✘	*correlation* 0.994 *with* $\Diamond_{[0,9]} (\neg p_\tau \wedge \neg p_h)$						
$\Diamond_{[0,6]}\Box_{[0,3]} \neg p_\tau$	10	✘	*correlation* 1.000 *with* $\Box_{[0,6]}\Diamond_{[0,3]} p_\tau$						
$\Diamond_{[0,14]} \neg p_\tau$	15	✔	0.490	0.1677	0.1635	0.1623	0.1579	0.406	1.00
$\Diamond_{[0,14]} \neg p_h$	15	✔	0.674	0.1036	0.0967	0.0970	0.0902	0.593	1.00
$\Diamond_{[0,19]} \neg p_\tau$	20	✔	0.522	0.1732	0.1682	0.1671	0.1613	0.442	1.00
$\Diamond_{[0,19]} \neg p_h$	20	✔	0.693	0.1085	0.0989	0.0980	0.0924	0.614	1.00
$\Diamond_{[0,19]} (\neg p_\tau \wedge \neg p_h)$	20	✔	0.458	0.1586	0.1542	0.1514	0.1476	0.384	1.00
$p_h \mathsf{U}_{[0,19]} \neg p_\tau$	20	✔	0.400	0.1597	0.1579	0.1577	0.1501	0.309	1.00
$p_\tau \mathsf{U}_{[0,19]} \neg p_h$	20	✔	0.641	0.1105	0.1035	0.1014	0.0959	0.569	1.00
$\Box_{[0,16]}\Diamond_{[0,3]} p_\tau$	20	✔	0.528	0.1836	0.1720	0.1724	0.1667	0.469	0.97
$\Box_{[0,16]}\Diamond_{[0,3]} p_h$	20	✔	0.357	0.1137	0.1057	0.1068	0.1004	0.295	1.00
$\Diamond_{[0,19]} \neg p_s$	20	✘	*reference gain* $0.0089 < 0.01$						
$\Diamond_{[0,19]} \neg p_\tau \wedge \Box_{[0,19]} p_h$	20	✘	*base rate* $0.044 \notin [0.05, 0.95]$						
$\Box_{[0,19]} p_h \wedge \Box_{[0,19]} p_s$	20	✘	*correlation* 0.989 *with* $\Diamond_{[0,19]} \neg p_h$						
$\Diamond_{[0,19]} \neg p_h \wedge \Diamond_{[0,19]} \neg p_\tau$	20	✘	*correlation* 0.985 *with* $\Diamond_{[0,19]} (\neg p_\tau \wedge \neg p_h)$						
$\Diamond_{[0,16]}\Box_{[0,3]} \neg p_\tau$	20	✘	*correlation* 1.000 *with* $\Box_{[0,16]}\Diamond_{[0,3]} p_\tau$						
$\Diamond_{[0,29]} \neg p_\tau$	30	✔	0.578	0.1809	0.1734	0.1728	0.1666	0.494	1.00
$\Diamond_{[0,29]} \neg p_h$	30	✔	0.724	0.1163	0.1005	0.1002	0.0941	0.651	1.00
$\Diamond_{[0,29]} (\neg p_\tau \wedge \neg p_h)$	30	✔	0.507	0.1686	0.1613	0.1590	0.1536	0.429	0.97
$p_h \mathsf{U}_{[0,29]} \neg p_\tau$	30	✔	0.415	0.1698	0.1668	0.1688	0.1583	0.321	1.00
$p_\tau \mathsf{U}_{[0,29]} \neg p_h$	30	✔	0.661	0.1202	0.1079	0.1074	0.1010	0.591	1.00
$\Box_{[0,26]}\Diamond_{[0,3]} p_\tau$	30	✔	0.472	0.2005	0.1804	0.1807	0.1737	0.406	1.00
$\Box_{[0,26]}\Diamond_{[0,3]} p_h$	30	✔	0.320	0.1227	0.1087	0.1083	0.1036	0.255	1.00
$\Diamond_{[0,39]} \neg p_\tau$	40	✔	0.620	0.1818	0.1718	0.1697	0.1647	0.538	1.00
$\Diamond_{[0,39]} \neg p_h$	40	✔	0.748	0.1240	0.1022	0.1010	0.0947	0.679	1.00
$\Diamond_{[0,39]} (\neg p_\tau \wedge \neg p_h)$	40	✔	0.551	0.1746	0.1632	0.1638	0.1547	0.471	0.97
$p_h \mathsf{U}_{[0,39]} \neg p_\tau$	40	✔	0.424	0.1769	0.1710	0.1738	0.1643	0.332	1.00
$p_\tau \mathsf{U}_{[0,39]} \neg p_h$	40	✔	0.672	0.1288	0.1122	0.1109	0.1044	0.610	1.00
$\Box_{[0,36]}\Diamond_{[0,3]} p_\tau$	40	✔	0.420	0.2081	0.1790	0.1783	0.1722	0.363	0.93
$\Box_{[0,36]}\Diamond_{[0,3]} p_h$	40	✔	0.290	0.1285	0.1106	0.1098	0.1044	0.225	1.00
Waymo Perception (4 included of 35 candidates)									
$\Diamond_{[0,2]} \neg p_c$	3	✘	*correlation* 1.000 *with* $\Diamond_{[0,4]} \neg p_c$						
$\Diamond_{[0,2]} \neg p_\tau$	3	✘	*reference gain* $-0.0025 < 0.01$						
$\Diamond_{[0,4]} \neg p_c$	5	✔	0.175	0.0895	0.0885	0.0910	0.1045	0.060	1.00
$\Diamond_{[0,4]} \neg p_\tau$	5	✘	*reference gain* $-0.0031 < 0.01$						

Property	H	incl.	base rate	ours	reference	RNN	AR-64	cert. LB	agreement
$\Diamond_{[0,4]}\neg p_p$	5	✘	*reference gain* $0.0027 < 0.01$						
$\Diamond_{[0,4]}\neg p_g$	5	✘	*base rate* $0.037 \notin [0.05, 0.95]$						
$\Diamond_{[0,4]}(\neg p_c \wedge \neg p_\tau)$	5	✘	*base rate* $0.025 \notin [0.05, 0.95]$						
$\Diamond_{[0,4]}\neg p_p \wedge \Box_{[0,4]} p_c$	5	✘	*base rate* $0.013 \notin [0.05, 0.95]$						
$\Diamond_{[0,4]}\neg p_c \wedge \Diamond_{[0,4]}\neg p_\tau$	5	✘	*base rate* $0.025 \notin [0.05, 0.95]$						
$p_c \,\mathsf{U}_{[0,4]}\, \neg p_\tau$	5	✘	*reference gain* $-0.0034 < 0.01$						
$p_\tau \,\mathsf{U}_{[0,4]}\, \neg p_g$	5	✘	*base rate* $0.037 \notin [0.05, 0.95]$						
$\Diamond_{[0,9]}\neg p_c$	10	✔	0.188	0.1012	0.0974	0.0981	0.1104	0.074	0.93
$\Diamond_{[0,9]}\neg p_\tau$	10	✘	*reference gain* $-0.0052 < 0.01$						
$\Diamond_{[0,9]}\neg p_p$	10	✘	*reference gain* $0.0038 < 0.01$						
$\Diamond_{[0,9]}\neg p_g$	10	✘	*base rate* $0.037 \notin [0.05, 0.95]$						
$\Diamond_{[0,9]}(\neg p_c \wedge \neg p_\tau)$	10	✘	*reference gain* $0.0031 < 0.01$						
$\Diamond_{[0,9]}\neg p_p \wedge \Box_{[0,9]} p_c$	10	✘	*base rate* $0.013 \notin [0.05, 0.95]$						
$\Diamond_{[0,9]}\neg p_c \wedge \Diamond_{[0,9]}\neg p_\tau$	10	✘	*reference gain* $0.0029 < 0.01$						
$p_c \,\mathsf{U}_{[0,9]}\, \neg p_\tau$	10	✘	*reference gain* $-0.0064 < 0.01$						
$p_\tau \,\mathsf{U}_{[0,9]}\, \neg p_g$	10	✘	*base rate* $0.037 \notin [0.05, 0.95]$						
$\Box_{[0,6]}\Diamond_{[0,3]} p_c$	10	✘	*correlation* 0.954 *with* $\Diamond_{[0,4]}\neg p_c$						
$\Box_{[0,6]}\Diamond_{[0,3]} p_\tau$	10	✘	*reference gain* $-0.0026 < 0.01$						
$\Diamond_{[0,6]}\Box_{[0,3]}\neg p_c$	10	✘	*correlation* 0.954 *with* $\Diamond_{[0,4]}\neg p_c$						
$\Diamond_{[0,19]}\neg p_c$	20	✔	0.225	0.1194	0.1099	0.1228	0.1185	0.095	1.00
$\Box_{[0,16]}\Diamond_{[0,3]} p_c$	20	✔	0.812	0.0979	0.0945	0.0993	0.1060	0.659	0.87
$\Diamond_{[0,19]}\neg p_\tau$	20	✘	*reference gain* $-0.0045 < 0.01$						
$\Diamond_{[0,19]}\neg p_p$	20	✘	*reference gain* $0.0023 < 0.01$						
$\Diamond_{[0,19]}\neg p_g$	20	✘	*base rate* $0.037 \notin [0.05, 0.95]$						
$\Diamond_{[0,19]}(\neg p_c \wedge \neg p_\tau)$	20	✘	*reference gain* $0.0047 < 0.01$						
$\Diamond_{[0,19]}\neg p_p \wedge \Box_{[0,19]} p_c$	20	✘	*base rate* $0.025 \notin [0.05, 0.95]$						
$\Diamond_{[0,19]}\neg p_c \wedge \Diamond_{[0,19]}\neg p_\tau$	20	✘	*reference gain* $0.0049 < 0.01$						
$p_c \,\mathsf{U}_{[0,19]}\, \neg p_\tau$	20	✘	*reference gain* $-0.0046 < 0.01$						
$p_\tau \,\mathsf{U}_{[0,19]}\, \neg p_g$	20	✘	*base rate* $0.037 \notin [0.05, 0.95]$						
$\Box_{[0,16]}\Diamond_{[0,3]} p_\tau$	20	✘	*reference gain* $-0.0072 < 0.01$						
$\Diamond_{[0,16]}\Box_{[0,3]}\neg p_c$	20	✘	*correlation* 1.000 *with* $\Box_{[0,16]}\Diamond_{[0,3]} p_c$						

H.3 Certificate Details on the Driving Benchmarks

Referenced from Section 5. Certificates use $\delta = 0.1$ per property with $B = 10$ bins on the motion benchmarks and $B = 5$ on Perception, whose calibration split affords fewer bins. Bin edges are placed on a quarter of the calibration scenarios, whose outcomes are never read, and test histories are only ever looked up in the stored table. Averaged over the included properties and three seeds, test-split frequencies meet their bounds in 99, 100, and 95% of populated bins on WOMD, Waymo BEV, and Perception, respectively. This agreement is a diagnostic, since test frequencies only estimate the population bin means bounded by Proposition 1.