Multimodal Model Predictive Runtime Verification for Safety of Autonomous Cyber-Physical Systems*

Alexis Aurandt[0000−0003−2008−673X], Phillip H. Jones[0000−0002−8220−7552], Kristin Yvonne Rozier[0000−0002−6718−2828], and Tichakorn Wongpiromsarn[0000−0002−3977−122X]

Iowa State University, USA {aurandt,phjones,kyrozier,nok}@iastate.edu

Abstract. Autonomous cyber-physical systems must be able to operate safely in a wide range of complex environments. To ensure safety without limiting mitigation options, these systems require detection of safety violations by mitigation trigger deadlines. As a result of these system's complex environments, multimodal prediction is often required. For example, an autonomous vehicle (AV) operates in complex traffic scenes that result in any given vehicle having the ability to exhibit several plausible future behavior modes (e.g., stop, merge, turn, etc.); therefore, to ensure collision avoidance, an AV must be able to predict the possible multimodal behaviors of nearby vehicles. In previous work, model predictive runtime verification (MPRV) successfully detected future violations by a given deadline, but MPRV only considers a single mode of prediction (i.e., unimodal prediction). We design multimodal model predictive runtime verification (MMPRV) to extend MPRV to consider multiple modes of prediction, and we introduce Predictive Mission-Time Linear Temporal Logic (PMLTL) as an extension of MLTL to support the evaluation of probabilistic multimodal predictions. We examine the correctness and real-time feasibility of MMPRV through two AV case studies where MMPRV utilizes (1) a physics-based multimodal predictor on the F1Tenth autonomous racing vehicle and (2) current state-of-the-art deep neural network multimodal predictors trained and evaluated on the Argoverse motion forecasting dataset. We found that the ability to meet real-time requirements was a challenge for the latter, especially when targeting an embedded computing platform.

1 Introduction

Autonomous cyber-physical systems such as autonomous vehicles (AVs), unmanned aerial systems (UAS), and robots are considered safety-critical due to their regular and close interaction with humans. Runtime verification (RV) offers an approach to monitor these systems for safety violations in a real-time online manner [9,24]. On-board RV can both detect safety violations and trigger mitigation actions to ensure safety, but the most effective mitigation strategies could require fault detection of future violations to prevent unsafe states [46,63,65,66]. For example, if it takes an AV three seconds to come to a complete stop, then the AV must apply the brakes three seconds before a complete stop is required to mitigate an impending crash. Due to the complexity of these systems' environments, multiple modes of future behavior are plausible [27,44]. For example, a human driver can display different behaviors given a specific traffic scene (e.g., stop, slow down, swerve, merge, turn, etc.). Therefore, for RV to be effective in such systems, it must be able to support multimodal predictions.

Predictive runtime verification [46,66] employs model predictors to detect future specification violations. In previous work, some utilize the given knowledge of a system to produce a model predictor [16,22,46,65], while others learn a system model by statistical learning [6,7,8,47,63] or machine learning [23,38,42], but all of these works focus solely on unimodal prediction. While the complete set of a system's reachable states is infeasible to directly compute in

* Supported by NSF:CPS Award 2038903. Reproducibility artifacts and additional details available at *http://temporallogic.org/research/MMPRV*.

real-time, some have also looked at variations of reachability analysis that compute the reachability offline or over-approximate the reachability online in real-time [1,3,10,12,15,17,55,64]. Overall, reachability analysis produces over-conservative results, potentially leading to numerous false positives; therefore, multimodal prediction has become increasingly popular as it reduces the complete set to a handful of the most plausible future behaviors, but to the best of our knowledge, multimodality has not been considered in predictive runtime monitors. Therefore, we introduce Multimodal Model Predictive Runtime Verification (MMPRV), which evaluates a safety specification by a deadline d given K finite sequences of future states.

MMPRV is a direct extension of Model Predictive Runtime Verification (MPRV) [65] and leverages MPRV's definition of deadline and unique utilization of maximum observed data and minimum predicted data to make an on-deadline evaluation. The MPRV framework was deployed on the R2U2 (Realizable, Responsive, Unobtrusive Unit) RV engine [29,50,51,53] and was the first predictive RV framework to provide memory and real-time guarantees. We also deploy MMPRV on the R2U2 RV engine as it is one of the few RV engines that can operate in real-time [19]. Additionally, R2U2 has a strong history of being deployed on real-time, resource-constrained, mission-critical systems [5,13,20,26,33] and has recently undergone changes for added user usability and further reduction of memory requirements [29,30].

R2U2 natively encodes specifications expressed in Mission-time Linear Temporal Logic (MLTL), but we introduce Predictive MLTL (PMLTL) as an extension of MLTL with the addition of four important features: (1) semantics that utilize maximum observed data and minimum predicted data to evaluate a specification by a deadline d, (2) ability to reason over K finite sequences of future states (i.e., supports multimodality), (3) supports the evaluation of a *sequence* of probabilistic atomic propositions, and (4) allows user-defined probabilistic inference techniques. No existing logic supports even two of these features. Several extensions of Signal Temporal Logic (STL) [43] reason about probabilistic signals by quantifying the probability of satisfying an atomic predicate (C2TL [28], StTL [34], StSTL [36], STL-U [42], PrSTL [52], and ProbSTL [59]), but they all make strong assumptions on the underlying probabilistic inference. There is a single extension of Metric Temporal Logic (MTL) [4] called P-MTL [58] that allows the probabilistic inference technique to be determined by the user. Additionally, STL-U is the only aforementioned extension that can also reason about a *sequence* of probabilistic atomic predicates.

We design MMPRV to allow for *any* user-defined model predictor, extending its applicability to a wide range of systems. To this extent, we minimize MMPRV's memory requirements for deployability to resource-constrained, real-time systems. We examine the correctness and real-time feasibility of MMPRV through two case studies that employ (1) a physics-based Monte Carlo (MC) multimodal predictor and (2) state-of-the-art (SOTA) deep neural network (DNN) multimodal predictors. We illustrate that MMPRV determines the verdict of a PMLTL specification φ by a deadline d utilizing K finite sequences of future states produced by these predictors. In the first case study, we target an embedded computing platform (i.e., NVIDIA® Jetson Xavier NX) and observe that our implementation is feasible in real-time with a 20 Hz control loop, but we do not achieve SOTA accuracy through this approach. In the second case study, we examine the real-time feasibility of SOTA DNNs, but none of these DNNs meet the real-time requirements of a 10 Hz control loop on the NVIDIA® Jetson Xavier NX and instead require the computing capabilities of a desktop GPU.

Our contributions include **(1)** syntax and semantics of PMLTL (Section 3.1), **(2)** the MMPRV algorithm and proofs of correctness (Section 3.2), **(3)** memory requirements for MMPRV (Section 3.3), and **(4)** application and real-time feasibility of MMPRV utilizing a physics-based MC multimodal predictor on the F1Tenth autonomous racing vehicle (Section 4.1) and **(5)** utilizing SOTA DNN multimodal predictors trained and evaluated on the Argoverse dataset (Section 4.2).

2 Preliminaries

2.1 Mission-Time Linear Temporal Logic (MLTL) [35,50]

MLTL is a variant of LTL over finite traces with temporal intervals that are bounded, closed, and discrete. MLTL expresses the most commonly utilized fragments of MTL [4] and STL [43].

Definition 1. *(MLTL Syntax)* The syntax of an MLTL formula φ over a set of atomic propositions $\mathcal{AP}$ is recursively defined as:

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid p \mid \neg\psi \mid \psi \wedge \xi \mid \psi \vee \xi \mid \Box_I \psi \mid \Diamond_I \psi \mid \psi \,\mathcal{U}_I\, \xi \mid \psi \,\mathcal{R}_I\, \xi$$

where $p \in \mathcal{AP}$ is an atom, ψ and ξ are MLTL formulas, and I is a closed interval $[lb,ub]$ where lb and ub denote the lower and upper bound, respectively, such that $lb \leq ub$ and $lb,ub \in \mathbb{N}_0$.

Definition 2. *(Finite Trace)* A finite trace, denoted by π, is a finite sequence of sets of atomic propositions. The i^{th} set is denoted by $\pi(i)$ and contains the atomic propositions that are satisfied at the i^{th} time step. $|\pi|$ denotes the length of π (where $|\pi| < \infty$), and $\pi[lb,ub]$ denotes the trace segment $\pi(lb),\pi(lb+1),...,\pi(ub)$.

Definition 3. *(MLTL Semantics)* We recursively define $\pi,i \models \varphi$ (finite trace π starting from time index $i \geq 0$ satisfies, or "models" MLTL formula φ) as

- $\pi,i \models \mathsf{true}$
- $\pi,i \models p$ for $p \in \mathcal{AP}$ iff $p \in \pi(i)$
- $\pi,i \models \neg\psi$ iff $\pi,i \not\models \psi$
- $\pi,i \models \psi \wedge \xi$ iff $\pi,i \models \psi$ and $\pi,i \models \xi$
- $\pi,i \models \psi \,\mathcal{U}_{[lb,ub]}\, \xi$ iff $|\pi| \geq i+lb$ and $\exists j \in [i+lb,i+ub]$ such that $\pi,j \models \xi$ and $\forall k < j$ where $k \in [i+lb,i+ub]$ we have $\pi,k \models \psi$

Given two MLTL formulas ψ and ξ, they are semantically equivalent (denoted by $\psi \equiv \xi$) if and only if $\pi \models \psi \Leftrightarrow \pi \models \xi$ for all traces π. To complete the MLTL semantics, we define $\mathsf{false} \equiv \neg\mathsf{true}$, $\psi \vee \xi \equiv \neg(\neg\psi \wedge \neg\xi)$, $\neg(\psi \,\mathcal{U}_I\, \xi) \equiv (\neg\psi \,\mathcal{R}_I\, \neg\xi)$, and $\neg\Diamond_I\psi \equiv \Box_I\neg\psi$. MLTL also keeps the standard operator equivalences from LTL, including $\Diamond_I\psi \equiv (\mathsf{true}\,\mathcal{U}_I\,\psi)$, and $\Box_I\psi \equiv (\mathsf{false}\,\mathcal{R}_I\,\psi)$. Notably, MLTL discards the next ($\mathcal{X}$) operator since $\mathcal{X}\psi \equiv \Box_{[1,1]}\psi$.

2.2 Abstract Syntax Tree Architecture

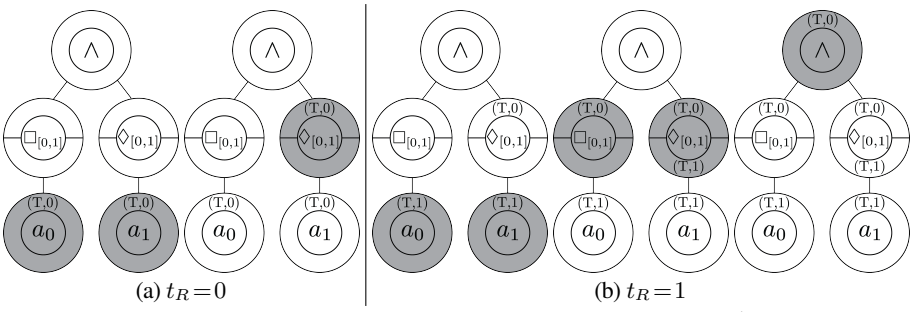


Fig. 1. Abstract syntax tree evaluation of $\varphi = \Box_{[0,1]}a_0 \wedge \Diamond_{[0,1]}a_1$ where $a_0,a_1 \in \mathcal{AP}$. The highlighted nodes are the nodes currently being updated at each step as verdicts are propagated upwards through the tree. Results are shown for the current timestamp $t_R = 0$ and $t_R = 1$.

R2U2 is a stream-based RV engine that reevaluates MLTL formulas for each time index i. These MLTL formulas are represented by decomposing them into subformula nodes in an Abstract Syntax Tree (AST). R2U2 determines the evaluation of each subformula node from the bottom-up and propagates the verdict to the parent node(s). Each node of the AST computes and stores verdict-timestamp tuples $T_\psi = (v,\tau)$ for its subformula ψ, where $v \in \{\mathsf{true},\mathsf{false}\}$

and $\tau \in \mathbb{N}_0$. Each node stores the verdict-timestamp tuples in a shared connection queue (SCQ); the SCQ is a circular buffer that overwrites verdict-timestamp tuples in a circular manner. Figure 1 demonstrates an example of how R2U2 evaluates over an AST.

Propogation Delay. To compute the SCQ size of each node in the AST, the propagation delay of each subformula must first be computed.

Definition 4. *(Propagation Delay [33])* The propagation delay of an MLTL formula φ is the time between when a set of propositions $\pi(i)$ arrives and when the verdict of $\pi, i \models \varphi$ is determinable. The best-case propagation delay ($\varphi.bpd$) is its minimum time delay, and the worst-case propagation delay ($\varphi.wpd$) is its maximum time delay.

Definition 5. *(Propagation Delay Semantics [33])* Let ψ and ξ be MLTL subformulas of MLTL formula φ where the best- and worst-case propagation delay for an MLTL formula φ is structurally defined as follows:

- $\varphi \in \mathcal{AP}$: $\begin{cases} \varphi.wpd = 0 \\ \varphi.bpd = 0 \end{cases}$
- $\varphi = \neg\psi$: $\begin{cases} \varphi.wpd = \psi.wpd \\ \varphi.bpd = \psi.bpd \end{cases}$
- $\varphi = \psi \vee \xi$ *or* $\varphi = \psi \wedge \xi$: $\begin{cases} \varphi.wpd = max(\psi.wpd, \xi.wpd) \\ \varphi.bpd = min(\psi.bpd, \xi.bpd) \end{cases}$
- $\varphi = \Box_{[lb,ub]}\psi$ *or* $\varphi = \Diamond_{[lb,ub]}\psi$: $\begin{cases} \varphi.wpd = \psi.wpd + ub \\ \varphi.bpd = \psi.bpd + lb \end{cases}$
- $\varphi = \psi \,\mathcal{U}_{[lb,ub]}\, \xi$ *or* $\varphi = \psi \,\mathcal{R}_{[lb,ub]}\, \xi$: $\begin{cases} \varphi.wpd = max(\psi.wpd, \xi.wpd) + ub \\ \varphi.bpd = min(\psi.bpd, \xi.bpd) + lb \end{cases}$

SCQ Memory Size. To promote deployability to resource-constrained platforms, R2U2 minimizes the size requirement for its SCQs. The minimum SCQ size of an AST node g is determined by the worst-case propagation delay of its sibling nodes and its own best-case propagation delay. A node g must store verdict-timestamp tuples in its SCQ until all of its siblings have the same timestamp τ for these tuples to be consumed by their parent node. Therefore, the size of node g's SCQ corresponds to the maximum timestamp mismatch between node g and its siblings. If we let $\mathbb{S}_g$ be the set of all of g's sibling nodes, then the size of g's SCQ is given by the following [33,65]:

$$SCQ_{size}(g) = max(max\{s.wpd \mid s \in \mathbb{S}_g\} - g.bpd, 0) + 1 \tag{1}$$

2.3 Model Predictive Runtime Verification (MPRV) [65]

MPRV strives to produce the most accurate evaluation of a specification φ possible by a mitigation trigger deadline d to allow for effective mitigation triggering. Since observed data is often more accurate than predicted data, MPRV utilizes maximum observed data (i.e., observed data for time steps up to and including the current timestamp) and minimal predicted data (i.e., predicted data only after the current timestamp) to make an on-deadline evaluation.

Definition 6. *(Deadline)* Given an MLTL formula φ and trace π starting from time index $i \geq 0$, the deadline $d \in \mathbb{Z}$ is the number of time steps measured relative to i by which the verdict of φ must be determined such that $0 \leq i + d \leq M$, where M denotes the timestamp at the end of the mission (i.e., φ cannot be evaluated before the mission begins or after it ends).

Definition 7. *(Finite Trace with Prediction)* Trace $\hat{\pi}$ is a finite trace (following from Definition 2) that has an observed and predicted segment such that the segment $\hat{\pi}[0, |\pi| - 1] = \pi$ where π is derived from observed data and $|\pi| \leq i + d$, and the segment $\hat{\pi}[|\pi|, |\hat{\pi}| - 1]$ is populated using prediction to determine the verdict of a MLTL specification φ for time index i by deadline d.

Definition 8. *(MLTL Semantics with Deadline)* MLTL semantics with deadline d is an extension of the MLTL Semantics in Definition 3. Given a finite trace π, time index i, and deadline d to produce a finite trace with prediction $\hat{\pi}$ (following from Definition 7), we recursively define $\pi,i,d \models \varphi$ (trace π starting from time index $i \geq 0$ satisfies, or "models" MLTL formula φ by deadline d) as

- $\pi,i,d \models \mathsf{true}$
- $\pi,i,d \models p$ for $p \in \mathcal{AP}$ iff $\hat{\pi},i \models p$ such that $p \in \hat{\pi}(i)$
- $\pi,i,d \models \neg\psi$ iff $\pi,i,d \not\models \psi$
- $\pi,i,d \models \psi \wedge \xi$ iff $\pi,i,d \models \psi$ and $\pi,i,d \models \xi$
- $\pi,i,d \models \psi\, \mathcal{U}_{[lb,ub]}\, \xi$ iff $|\hat{\pi}| \geq i+lb$ and $\exists j \in [i+lb, i+ub]$ such that $\pi,j,d \models \xi$ and $\forall k < j$ where $k \in [i+lb, i+ub]$ we have $\pi,k,d \models \psi$

Definition 9. *(Prediction Horizon)* The prediction horizon H_p is the length of the predicted segment of $\hat{\pi}$ (i.e., $H_p = |\hat{\pi}| - |\pi|$). Given an MLTL formula φ, the maximum prediction horizon is denoted by $max(H_p)$ and is bounded such that $max(H_p) = \varphi.wpd - d$.

To prevent overwriting original SCQ data with any predicted data, we determine $max(H_p)$ at design time and add $max(H_p)$ extra entries to each SCQ given by the following equation:

$$SCQ_{size}(g) = max(max\{s.wpd \mid s \in \mathbb{S}_g\} - g.bpd, 0) + max(H_p) + 1 \quad (2)$$

Note that we improve on these memory requirements in Section 3.3.

3 Multimodal Model Predictive Runtime Verification (MMPRV)

MMPRV extends MPRV [65] to support multimodal prediction and to reason over a finite sequence of sets of probabilistic atomic propositions (i.e., atomic propositions with associated probability). To determine the verdict of a PMLTL formula φ by a deadline d (Definition 6) such that $\pi,i,d \models \varphi$, either the verdict must be determinable by the trace $\pi[0, i+d]$ (i.e., observed data only) or prediction must be utilized by populating K finite traces with prediction $\hat{\pi}_0, \hat{\pi}_1, ..., \hat{\pi}_{K-2}, \hat{\pi}_{K-1}$ (Definition 10 below). In other words, if the verdict of $\pi,i,d \models \varphi$ is unknown at the current timestamp $t_R = i+d$, then MMPRV must receive predicted values to determine the verdict of φ by deadline d. If MMPRV reveals that φ does not hold (i.e., the specification was violated), the result can trigger an appropriate mitigation action.

Definition 10. *(K Finite Traces with Prediction)* Let $K \in \mathbb{N}$ be the number of predicted finite traces (following from Definition 2) denoted by $\hat{\pi}_0, \hat{\pi}_1, ..., \hat{\pi}_{K-2}, \hat{\pi}_{K-1}$ where $\hat{\pi}_j$ is the j^{th} predicted trace. Each trace has an observed and a predicted segment. Every trace has the identical observed segment such that $\forall j \in [0, K-1]$ we have $\hat{\pi}_j[0, |\pi|-1] = \pi$ where π is derived from observed data and $|\pi| \leq i+d$. We populate each trace segment $\hat{\pi}_j[|\pi|, |\hat{\pi}|-1]$ (where $j \in [0, K-1]$) using a different mode of prediction to make an evaluation decision by d.

Consider an autonomous vehicle (AV) where the specification violation of $\varphi = \Box_{[0,3]} a$ (where a is an atomic proposition) indicates a collision and the appropriate mitigation action is coming to a complete stop. Let's assume that the mitigation trigger deadline for the AV to brake and come to a complete stop is $d = -4$ (i.e., the verdict must be determined four time steps before time index i). Therefore, for this AV to ensure safety, it must be able to determine the verdict of $\pi,i,-4 \models \varphi$. At the current timestamp $t_R = 4$, as illustrated in Figure 2, MMPRV must determine the verdict

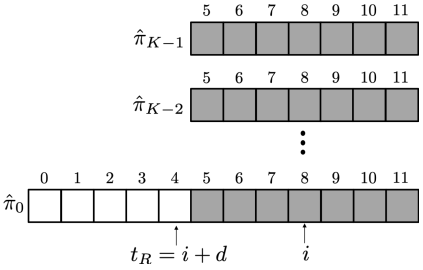


Fig. 2. K finite traces with prediction evaluating $\pi, 8, -4 \models \Box_{[0.3]} a$. White boxes indicate observed data and gray boxes are predicted data.

of $\pi, 8, -4 \models \varphi$ (i.e., $t_R = i + d = 8 + (-4) = 4$). Following from Definition 9, the $max(H_p) = \varphi.wpd - d = 3 - (-4) = 7$; therefore, MMPRV will obtain up to seven predicted values of a for all K traces such that MMPRV will incrementally obtain predicted values for $\hat{\pi}_j(t_R+1),\hat{\pi}_j(t_R+2),...,\hat{\pi}_j(t_R+max(H_p))\ \forall j \in [0,K-1]$ until φ evaluates to true or false. In Figure 2, this means we incrementally populate $\hat{\pi}_j(5),\hat{\pi}_j(6),...,\hat{\pi}_j(11)\ \forall j \in [0,K-1]$ with predicted values of a until the verdict of $\pi,8,-4 \models \varphi$ is known. At the next timestamp ($t_R = 5$), these predictions are no longer relevant; therefore, MMPRV will obtain new predicted values for $\hat{\pi}_j(6),\hat{\pi}_j(7),...,\hat{\pi}_j(12)\ \forall j \in [0,K-1]$ until the verdict of $\pi,9,-4 \models \varphi$ is determinable.

3.1 Predictive Mission-Time Linear Temporal Logic (PMLTL)

PMLTL is an extension of MLTL (Definition 3) that evaluates a specification φ utilizing K finite traces with prediction $\hat{\pi}_0,\hat{\pi}_1,...,\hat{\pi}_{K-2},\hat{\pi}_{K-1}$ (Definition 10) to determine the verdict of φ by a deadline d. (Note that unimodal prediction is still supported within PMLTL when $K = 1$.) Predictions often have an associated probability, and while it is often safe to assume that the observed segment of a trace $\hat{\pi}$ (i.e., π) has a probability of 1.0 because it has been physically observed, observed data can also have an associated uncertainty (e.g., sensor error). As a result, PMLTL introduces the probability operator P_δ, which allows specification and evaluation over sequences of sets of probabilistic atomic propositions. To this extent, a PMLTL specification can quantity the amount of uncertainty that is deemed acceptable by the user. For example, $P_{0.95}(\Box_{[0,3]} a)$ is a PMLTL formula that expresses "the probability of a being globally true from 0 to 3 is greater than or equal to 95%". PMLTL also supports the evaluation of observed and predicted traces without the consideration of probability.

Definition 11. *(PMLTL Syntax)* The syntax of PMLTL is an extension of the MLTL syntax defined in Definition 1. The syntax of a PMLTL formula φ over K sets of atomic propositions $\mathcal{AP}$ is recursively defined as:

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid p \mid \neg\psi \mid \psi \wedge \xi \mid \psi \vee \xi \mid \Box_I \psi \mid \Diamond_I \psi \mid \psi\, \mathcal{U}_I\, \xi \mid \psi\, \mathcal{R}_I\, \xi \mid P_\delta \psi$$

where $K \in \mathbb{N}$, $p \in \mathcal{AP}$, ψ and ξ are PMLTL formulas, $\delta \in [0,1]$ is the desired probability, and I is a closed interval $[lb,ub]$ where lb and ub denote the lower and upper bound, respectively, such that $lb \leq ub$ and $lb,ub \in \mathbb{N}_0$.

Definition 12. *(Probability Space of K Finite Traces)* Given a time index i, K finite traces of prediction $\hat{\pi}_0,\hat{\pi}_1,...,\hat{\pi}_{K-2},\hat{\pi}_{K-1}$, and an atomic proposition $p \in \mathcal{AP}$, let the sample space $\Omega_i = \{p \in \hat{\pi}_0(i), p \notin \hat{\pi}_0(i), p \in \hat{\pi}_1(i), p \notin \hat{\pi}_1(i), ..., p \in \hat{\pi}_{K-2}(i), p \notin \hat{\pi}_{K-2}(i), p \in \hat{\pi}_{K-1}(i), p \notin \hat{\pi}_{K-1}(i)\}$. Let the σ-algebra $\mathcal{F}_i = 2^{\Omega_i}$ (i.e., the powerset of Ω_i) be a collection of events. Let the probability measure $\mathbb{P}_i$ assign a probability $\mathbb{P}_i(A)$ to every event A in $\mathcal{F}_i$ such that $\mathbb{P}_i : \mathcal{F}_i \mapsto [0,1]$ where $\mathbb{P}_i(\Omega_i) = 1$ and $\mathbb{P}_i(A) = \sum_{\omega \in A} \mathbb{P}_i(\{\omega\}) \leq 1$ [1]. Note that the complement of an event $A \in \mathcal{F}_i$ is denoted as $A^c = \Omega_i \backslash A$ such that $\mathbb{P}_i(A^c) = 1 - \mathbb{P}_i(A)$. Then, $(\Omega_i, \mathcal{F}_i, \mathbb{P}_i)$ defines the probability space of K finite traces at time index i for $p \in \mathcal{AP}$.

Definition 13. *(PMLTL Semantics)* PMLTL semantics are an extension of the MLTL semantics with deadline in Definition 8. Given a finite trace π, time index i, and deadline d to produce K finite traces with prediction $\hat{\pi}_0,\hat{\pi}_1,...,\hat{\pi}_{K-2},\hat{\pi}_{K-1}$ (following from Definition 10) and a probability space $(\Omega_i, \mathcal{F}_i, \mathbb{P}_i)$ for each $p \in \mathcal{AP}$ (as defined in Definition 12) [2], we recursively define $\pi, i, d \models \varphi$ (trace π starting from time index $i \geq 0$ satisfies, or "models" PMLTL formula φ by deadline d according to K predictions) as

[1] For all $\omega \in \Omega_i$, $\mathbb{P}_i(\{\omega\})$ is defined by the user's choice of probabilistic inference (e.g., Markov chain, Bayesian inference, normal distribution, etc.) but must follow the properties defined in Definition 12.

[2] The probability space $(\Omega_i, \mathcal{F}_i, \mathbb{P}_i)$ is only required if $p \in \mathcal{AP}$ is an operand of P_δ.

- $\pi,i,d \models \mathsf{true}$
- $\pi,i,d \models p$ for $p \in \mathcal{AP}$ iff $\forall j \in [0,K-1]$ we have $p \in \hat{\pi}_j(i)$
- $\pi,i,d \models \neg\psi$ iff $\pi,i,d \not\models \psi$
- $\pi,i,d \models \psi \wedge \xi$ iff $\pi,i,d \models \psi$ and $\pi,i,d \models \xi$
- $\pi,i,d \models \psi \, \mathcal{U}_{[lb,ub]} \, \xi$ iff $\forall j \in [0,K-1]$ we have $|\hat{\pi}_j| \geq i+lb$ and $\exists j \in [i+lb,i+ub]$ such that $\pi,j,d \models \xi$ and $\forall k < j$ where $k \in [i+lb,i+ub]$ we have $\pi,k,d \models \psi$
- $\pi,i,d \models P_\delta \psi$ iff $\Pr(\pi,i,d \models \psi) \geq \delta$ where $\Pr(\pi,i,d \models \psi)$ (the probability of $\pi,i,d \models \psi$) is defined recursively as follows:
  - $\Pr(\pi,i,d \models \mathsf{true}) = 1$
  - $\Pr(\pi,i,d \models p) = \mathbb{P}_i(A)$ for $p \in \mathcal{AP}$, A is an independent event, and $A \in \mathcal{F}_i$ s.t. $A = \bigcup_{j=0}^{K-1} \{\omega \in \Omega_i \mid \omega \equiv p \in \hat{\pi}_j(i)\}$
  - $\Pr(\pi,i,d \models \neg\psi) = 1 - \Pr(\pi,i,d \models \psi)$
  - $\Pr(\pi,i,d \models \psi \wedge \xi) = \Pr(\pi,i,d \models \psi) * \Pr(\pi,i,d \models \xi)$
  - $\Pr(\pi,i,d \models \psi \, \mathcal{U}_{[lb,ub]} \, \xi) = \Pr\Big(\bigvee_{m=i+lb}^{i+ub} \big((\bigwedge_{k=i+lb}^{m-1} \pi,k,d \models \psi) \wedge \pi,m,d \models \xi\big)\Big)$

Given two PMLTL formulas ψ and ξ, they are semantically equivalent (denoted by $\psi \equiv \xi$) if and only if $\pi,i,d \models \psi \Leftrightarrow \pi,i,d_j \models \xi$ for all possible K finite traces with prediction $\hat{\pi}_0,\hat{\pi}_1,...,\hat{\pi}_{K-2},\hat{\pi}_{K-1}$. PMLTL keeps the standard operator equivalences from MLTL with the addition that these equivalences also apply to $P_\delta\psi$ (i.e., $P_\delta(\mathsf{false}) \equiv P_\delta(\neg\mathsf{true})$, $P_\delta(\psi \vee \xi) \equiv P_\delta(\neg(\neg\psi \wedge \neg\xi))$, $P_\delta(\neg(\psi \, \mathcal{U}_I \, \xi)) \equiv P_\delta(\neg\psi \, \mathcal{R}_I \, \neg\xi)$, $P_\delta(\neg\Diamond_I\psi) \equiv P_\delta(\Box_I\neg\psi)$, $P_\delta(\Diamond_I\psi) \equiv P_\delta(\mathsf{true} \, \mathcal{U}_I \, \psi)$, and $P_\delta(\Box_I\psi) \equiv P_\delta(\mathsf{false} \, \mathcal{R}_I \, \psi)$). Figure 3 illustrates a few examples of determining the probability of φ (i.e., $\Pr(\pi,i,d \models \varphi)$).

	Time index i				
	0	1	2	3	4
$a_0 \in \hat{\pi}_0(i)$	true	true	false	true	false
$\mathbb{P}_i(\{a_0 \in \hat{\pi}_0(i)\})$	0.40	0.45	0.90	0.80	0.85
$\mathbb{P}_i(\{a_0 \notin \hat{\pi}_0(i)\})$	0.00	0.00	0.00	0.00	0.00
$a_0 \in \hat{\pi}_1(i)$	false	true	true	false	true
$\mathbb{P}_i(\{a_0 \in \hat{\pi}_1(i)\})$	0.60	0.55	0.10	0.20	0.15
$\mathbb{P}_i(\{a_0 \notin \hat{\pi}_1(i)\})$	0.00	0.00	0.00	0.00	0.00
$\Pr(\pi,i,d \models a_0)$	0.40	1.00	0.10	0.80	0.15

	Time index i				
	0	1	2	3	4
$a_1 \in \hat{\pi}_0(i)$	true	false	false	true	true
$\mathbb{P}_i(\{a_1 \in \hat{\pi}_0(i)\})$	0.95	0.35	0.20	0.90	0.85
$\mathbb{P}_i(\{a_1 \notin \hat{\pi}_0(i)\})$	0.05	0.65	0.80	0.10	0.15
$\Pr(\pi,i,d \models a_1)$	0.95	0.65	0.80	0.90	0.85

	Time index i				
	0	1	2	3	4
$\Pr(\pi,i,d \models a_0 \vee a_1)$	0.97	1.00	0.82	0.98	0.8725
$\Pr(\pi,i,d \models \Box_{[0,1]} a_0)$	0.40	0.10	0.08	0.12	–
$\Pr(\pi,i,d \models \Diamond_{[0,1]} a_1)$	0.9825	0.93	0.98	0.985	–
$\Pr(\pi,i,d \models \Box_{[0,1]} a_0 \wedge \Diamond_{[0,1]} a_1)$	0.393	0.093	0.0784	0.1182	–
$\Pr(a_0 \mathcal{U}_{[0,1]} a_1)$	0.963	0.93	0.818	0.968	–

Fig. 3. Determining the probability of φ (i.e., $\Pr(\pi,i,d \models \varphi)$) where $a_0,a_1 \in \mathcal{AP}$

3.2 MMPRV Algorithm

Algorithm 1 defines the MMPRV algorithm for the R2U2 engine. Offline, the Configuration Compiler for Property Organization (C2PO) [29] compiles PMLTL formula(s) for input into R2U2 by decomposing these formula(s) into an AST (Section 2.2). The AST is a list of nodes in topological order (i.e., child nodes appear before their parent nodes); therefore, evaluating the AST at a specific timestamp means sequentially evaluating each of its nodes (lines 1–2 and 11–12 of Algorithm 1). Algorithm 1 first evaluates the AST based on observed data only

Algorithm 1: MMPRV Algorithm

Input: Current timestamp: t_R; Deadline: d; Prediction modes: K; Finite trace: $\pi[0,t_R]$;
AST representing PMLTL formula φ: φ_{AST}

1 **foreach** *Node* $g \in \varphi_{AST}$ **do** // Update φ_{AST} for current time stamp t_R
2 | $Node\_step([\pi],t_R,g)$; // Algorithm 2
3 **if** $read(\varphi.Queue).\tau < t_R - d$ **then** // Prediction required
4 | **foreach** *Node* $g \in \varphi_{AST}$ **do** // store original AST state
5 | | Store Node g's metadata; // e.g., read/write pointers
6 | **foreach** $j \in [0,K-1]$ **do** $\hat{\pi}_j \leftarrow \pi$; // initialize $\hat{\pi}_j, \forall j \in [0,K-1]$ with π
7 | $t \leftarrow t_R$; // initialize t with current timestamp
8 | **while** $read(\varphi.Queue).\tau < t_R - d$ **do** // if prediction is needed, loop
9 | | $t \leftarrow t+1$; // look into next prediction step
10 | | **foreach** $j \in [0,K-1]$ **do** $\hat{\pi}_j(t) \leftarrow model\_predict(t,j)$; // update $\hat{\pi}_j(t)$
11 | | **foreach** *Node* $g \in \varphi_{AST}$ **do**
12 | | | $Node\_step([\hat{\pi}_0,\hat{\pi}_1,\ldots,\hat{\pi}_{K-1}],t,g)$; // Algorithm 2
13 | **foreach** *Node* $g \in \varphi_{AST}$ **do** // restore original AST state
14 | | Store Node g's metadata; // e.g., read/write pointers

Algorithm 2: *Node_step*: Evaluate a node g in φ_{AST} for one timestamp

1 **function** $Node\_step([\pi_0,\pi_1,\ldots,\pi_{K-1}],i,g)$ **is**
| **Input:** Array of finite traces: $[\pi_0,\pi_1,\ldots,\pi_{K-1}]$; Time index: i; Node: g
2 | **if** *g is a descendant of P_δ operator* **then**
3 | | **if** *g is an $\mathcal{AP}$ operator* **then** // record the value of the atomic proposition
4 | | | $p \leftarrow 0$
5 | | | **for** $j \leftarrow 0$ *to* $K-1$ **do** // evaluate $\Pr(g)$ based on K finite traces
6 | | | | **if** $g \in \pi_j(i)$ **then** $p \leftarrow p + get\_Pr(g \in \pi_j(i))$;
7 | | | | **else** $p \leftarrow p + get\_Pr(g \notin \pi_j(i))$;
8 | | | $g.Queue.write((p,i))$; // write $T_g = (p,\tau)$
9 | | **else**
10 | | | $(p,\tau) \leftarrow$ evaluate Node g; // Algorithms 4, 5, and 6
11 | | | $g.Queue.write((p,\tau))$; // write $T_g = (p,\tau)$
12 | **else**
13 | | **if** *g is an $\mathcal{AP}$ operator* **then** // record the value of the atomic proposition
14 | | | **for** $j \leftarrow 0$ *to* $K-1$ **do** // evaluate g based on K finite traces
15 | | | | **if** $g \in \pi_j(i)$ **then continue**;
16 | | | | **else** $g.Queue.write((\mathsf{false},i))$ **return**; // write $T_g = (v,\tau)$
17 | | | $g.Queue.write((\mathsf{true},i))$; // write $T_g = (v,\tau)$
18 | | **else**
19 | | | $(v,\tau) \leftarrow$ evaluate Node g; // Algorithm 3 and Algorithms 3-6 from [33]
20 | | | $g.Queue.write((v,\tau))$; // write $T_g = (v,\tau)$

(lines 1–2). If the latest time index for a PMLTL formula φ produced by the AST (i.e., the latest $T_\varphi.\tau$ found by reading the root node $\varphi.Queue$) is less than the current timestamp t_R minus the deadline d (i.e., the verdict of $\pi,t_R-d,d \models \varphi$ is unknown), then prediction is required (line 3).

MMPRV provides predictions based on maximum observed data; therefore, to retain observed data in the SCQ that may still be relevant for future evaluations, we size each node according to Equation 3 in Section 3.3. While the observed data is never overwritten, a node's metadata (e.g., its read and write pointers) will change as nodes are evaluated based on predicted data (lines 11-12). Therefore, we store each node's metadata before prediction starts (lines 4–5) and restore it after prediction ends (lines 13–14) to ensure that predicted data is never unintentionally reused at the next execution of Algorithm 1. To support multimodal prediction during the prediction phase (lines 3–14), there are K finite traces with prediction (Definition 10) initialized with observed data (line 6) and populated with predicted data generated by a user-defined $model\_predict$ function (line 10) until $\pi,t_R-d,d \models \varphi$ evaluates to true or false (line 8).

Each node of the AST contains a write pointer to store tuples within its SCQ and read pointer(s) for its children's SCQ(s). R2U2's read and write SCQ operations are defined in [33]. With the addition of MMPRV, the write operation must never write past t_R-d when utilizing prediction to ensure maximum observed data is utilized for all future verdicts. Previously, each node's SCQ stored verdict-timestamp tuples (i.e., $T_\psi = (v,\tau)$ as discussed in Section 2.2), but with the addition of the probability operator P_δ, descendants of the probability operator

(i.e., probabilistic operators) will now store a probability-timestamp tuple $T_\psi = (p,\tau)$ where $p \in [0.0,1.0]$. Additionally, the worst-case propagation delay for probabilistic operators follows Definition 5, but the best-case propagation delay is equivalent to the worse-case as the entire interval $[lb,ub]$ is required for evaluation (i.e., cannot evaluate early based on partial information).

Theorem 1 (Correctness of MMPRV Algorithm). *Given the current timestamp t_R, deadline d, number of prediction modes K, trace $\pi[0,t_R]$, the AST representing the PMLTL formula φ (φ_{AST}), and a model predictor function ($model\_predict$), the MMPRV algorithm (Algorithm 1) utilizes maximum observed data and minimum predicted data to populate K finite traces with prediction in order to evaluate $\pi,i,d \models \varphi$ such that $\forall i\ T_\varphi.v = \mathsf{true}$ iff $\pi,i,d \models \varphi$.*

Proof. MMPRV makes evaluations utilizing all observed data values from π before prediction is even considered (lines 1–2). After this initial evaluation on observed data, if $T_\varphi.\tau \geq t_R - d$, then all deadlines have been met and MMPRV terminates guaranteeing to have determined the verdict of $\pi,i,d \models \varphi$ based on observed data only. But if $T_\varphi.\tau < t_R - d$ (line 3), then MMPRV takes maximum observed data (line 6) augmented incrementally with K modes of minimum prediction data until MMPRV produces the tuple such that $T_\varphi.\tau = t_R - d$ (lines 8–12); therefore, MMPRV only terminates when the verdict of $\pi,t_R-d,d \models \varphi$ is determinable. □

Algorithm 3: Probability Operator: $P_\delta \psi$

1 At each new input T_ψ:
2 **return** $(T_\psi.p >= \delta, T_\psi.\tau)$

Algorithm 4: Probabilistic Negation Operator: $Pr(\pi,i,d \models \neg\psi)$

1 At each new input T_ψ:
2 **return** $(1 - T_\psi.p, T_\psi.\tau)$

Algorithm 5: Probabilistic And Operator: $Pr(\pi,i,d \models \psi \wedge \xi)$

1 At each new input (T_ψ, T_ξ) s.t. $T_\psi.\tau = T_\xi.\tau$:
2 **return** $(T_\psi.p * T_\xi.p, T_\psi.\tau)$

Algorithm 6: Probabilistic Until Operator: $Pr(\pi,i,d \models \psi\, \mathcal{U}_{[lb,ub]}\, \xi)$

1 At each new input (T_ψ, T_ξ) s.t. $T_\psi.\tau = T_\xi.\tau$:
2 **if** $T_\psi.\tau - ub \geq 0$ **then** // check if $i \geq 0$
3 $p_{temp} = T_\xi.p$ // initialize p_{temp} to $\Pr(\pi,i+ub,d \models \xi)$ s.t. $T_\xi.\tau = i+ub$
4 **for** $t \leftarrow 1$ *to* $ub - lb$ **do** // iterate backwards through ψ and ξ's SCQs
5 $p_{temp} = p_{temp} * read(\psi.Queue, \psi.rd\_ptr - t).p$
6 $p_{temp} = (1 - [(1 - read(\xi.Queue, \xi.rd\_ptr - t).p) * (1 - p_{temp})]$
7 **return** $(p_{temp}, T_\psi.\tau - ub)$ // return probability-timestamp tuple

The correctness of Algorithms 2, 3, 4, and 5 follows directly from Definition 13.

Theorem 2 (Correctness of the Probabilistic Until Operator). *Algorithm 6 correctly implements $\varphi = \Pr(\pi,i,d \models \psi\, \mathcal{U}_{[lb,ub]}\, \xi)$ such that for all $i \geq 0$ Algorithm 6 returns the tuple $T_\varphi = \left(\Pr\left(\bigvee_{j=i+lb}^{i+ub} \left(\left(\bigwedge_{k=i+lb}^{j-1} \pi,k,d \models \psi\right) \wedge \pi,j,d \models \xi\right)\right), i\right)$.*

Proof. To evaluate the probability of $\pi,i,d \models \psi\, \mathcal{U}_{[lb,ub]}\, \xi$, the probability values for the children ψ and ξ for the entire interval $[i+lb,i+ub]$ where $lb \leq ub$ are required (i.e., cannot evaluate early based on partial information). When $T_\psi.\tau - ub \geq 0$ (line 2), the probability-timestamp tuple of the Until operator can be calculated for $i = T_\psi.\tau - ub$ such that $i \geq 0$ as the children SCQs are guaranteed to have stored from $[T_\psi.\tau - ub + lb, T_\psi.\tau]$ or $[i+lb,i+ub]$. This guarantee is the result of R2U2's write operation [33] and the SCQ sizing discussed later in Section 3.3. To calculate the probability, the equation $\Pr\left(\bigvee_{j=i+lb}^{i+ub} \left(\left(\bigwedge_{k=i+lb}^{j-1} \pi,k,d \models \psi\right) \wedge \pi,j,d \models \xi\right)\right)$ expands to the following:

$$\begin{aligned}\Pr\big(&\pi,i+lb,d \models \xi \vee (\pi,i+lb,d \models \psi \wedge \pi,i+lb+1,d \models \xi) \vee \\ &(\pi,lb,d \models \psi \wedge \pi,i+lb+1,d \models \psi \wedge \pi,i+lb+2,d \models \xi) \vee ... \\ \vee\ &(\pi,i+lb,d \models \psi \wedge \pi,i+lb+1,d \models \psi \wedge ... \wedge \pi,i+ub-1,d \models \psi \wedge \pi,i+ub,d \models \xi)\big)\end{aligned}$$

which can be rewritten to the following form using the law of distribution:

$$\begin{aligned}\Pr\big(\pi,i+lb,d\models\xi \vee (\pi,i+lb,d\models\psi \wedge (\pi,i+lb+1,d\models\xi \vee \\ (\pi,i+lb+1,d\models\psi \wedge (\pi,i+lb+2,d\models\xi \vee ... \\ (\pi,i+ub-2,d\models\psi \wedge (\pi,i+ub-1,d\models\xi \vee \\ (\pi,i+ub-1,d\models\psi \wedge \pi,i+ub,d\models\xi)))))))...\big)\end{aligned}$$

Utilizing this form, the probability of $\pi,i,d\models\psi\,\mathcal{U}_{[lb,ub]}\,\xi$ is calculated starting from the deepest nested parentheses (i.e., $\pi,i+ub,d\models\xi$) on line 3 and iterating outward on lines 4–6 by iterating backward through the children SCQs from $T_\psi.\tau-ub-1$ to $T_\psi.\tau-ub+lb$ (e.g., next is $\pi,i+ub-1,d\models\xi \vee (\pi,i+ub-1,d\models\psi \wedge$ *previous*$)$). Lastly, the probability-timestamp tuple is returned (line 7) such that $T_\varphi=\big(\Pr\Big(\bigvee_{j=i+lb}^{i+ub}\big((\bigwedge_{k=i+lb}^{j-1}\pi,k,d\models\psi) \wedge \pi,j,d\models\xi\big)\Big),i\big)$. □

3.3 Memory Requirements for MMPRV

MMPRV determines verdicts based on maximum observed data. Additionally, R2U2 utilizes Common Subexpression Elimination (CSE) to reduce memory requirements for sets of PMLTL formulas, where common subexpressions share a singular SCQ node [29,33], but taking advantage of this reduction requires that predicted data doesn't overwrite observed data still relevant to other subexpressions. Therefore, to prevent overwriting any relevant observed data with predicted data, extra slots must be added to the SCQ size (as defined in Equation 1). In [65], the SCQ size increased linearly with $max(H_p)$ as given by Equation 2, but this was an overestimate that can be minimized. On the other hand, the addition of probabilistic operators requires the children of probabilistic temporal operators (e.g., Until) to store results for the entire interval from $[i+lb,i+ub]$ for consumption by the parent (as discussed in Algorithm 6 and Theorem 2). Therefore, the SCQ size is minimized and redefined in Equation 3.

Theorem 3 (MMPRV SCQ Size). *Consider an AST representing PMLTL formula(s). Let $\mathbb{S}_g$ be the set of all sibling nodes of g, $\mathbb{TP}_g$ be the set of all probabilistic temporal parent nodes of g, and $max(H_p)$ be the maximum prediction horizon of g's parent formula(s). Then, the minimum size of g's SCQ required for MMPRV is given by the following:*

$$\begin{aligned}SCQ_{size}(g)=max(max\{s.wpd \mid s\in\mathbb{S}_g\}-g.bpd,\,0)+max\{p.ub-p.lb \mid p\in\mathbb{TP}_g\}+ \\ min\Big(max\big(max\{s.wpd \mid s\in\mathbb{S}_g\}-g.bpd,\,0\big)+max\{p.ub-p.lb \mid p\in\mathbb{TP}_g\}, \\ max\big(max(H_p)-1,\,0\big)\Big)+1 \quad (3)\end{aligned}$$

Proof. Without prediction or probabilistic operators, $SCQ_{size}(g)=max(max\{s.wpd \mid s\in\mathbb{S}_g\}-g.bpd,\,0)+1$ (Equation 1) has already been proven in [65]. In Equation 1, $max(max\{s.wpd \mid s\in\mathbb{S}_g\}-g.bpd,\,0)$ represents the maximum timestamp mismatch between g and its sibling nodes; node g may have to buffer this many tuples before they are consumed by the parent node. $+1$ extra SCQ slot is added to the size to account for the implementation requirement that a tuple must be buffered at least one cycle before it is consumed by a parent node(s). With the addition of probabilistic operators, children of probabilistic temporal operators must buffer $max\{p.ub-p.lb \mid p\in\mathbb{TP}_g\}+1$ extra SCQ slots (instead of just $+1$) as required by Algorithm 6. Since observed data is always fully evaluated before predicted data enters the SCQ, this $+1$ extra slot can be reused for predicted data without overwriting relevant observed data (i.e., this slot will never be required after prediction starts). Therefore, to store predicted data without overwriting relevant observed data, at most $max(max(H_p)-1,0)$ extra slots need to be added to g's SCQ as at most $max(H_p)$ predicted verdict-timestamp tuples ever enter the SCQ. On the other hand, at most $max\big(max\{s.wpd \mid s\in\mathbb{S}_g\}-g.bpd,\,0\big)+$

$max\{p.ub-p.lb \mid p \in \mathbb{TP}_g\}$ extra slots are required to be added to g's SCQ; any predicted data that needs to be stored is limited by the timestamp mismatch between node g and its siblings (following from the proof in [65]) and the temporal interval $[i+lb,i+ub]$ of its parent node(s). Therefore, only $min\big(max(max\{s.wpd \mid s \in \mathbb{S}_g\} - g.bpd,\ 0) + max\{p.ub - p.lb \mid p \in \mathbb{TP}_g\}, max(max(H_p)-1, 0)\big)$ extra slots are required for proper storage of predicted data. □

MMPRV Total Memory Size. PMLTL specifications that utilize the probability operator P_δ have larger memory requirements than similarly structured formulas that do not utilize P_δ (as shown in Table 1). The reason for the larger memory requirement is twofold: (i) children of probabilistic temporal operators may be required to store additional tuples for consumption by their parents as defined in Equation 3 and (ii) probabilistic operators have to store probability–timestamp tuples. Let's assume verdicts are single-byte boolean values, probabilities are stored as 8-byte doubles, and the timestamp is stored as a 4-byte integer. Therefore, a verdict-timestamp tuple requires 5 bytes and a probability-timestamp tuple requires 12 bytes, and the memory size in bytes of a single node g's SCQ is given by the following:

$$SCQ_{memory}(g) = \begin{cases} 12 * SCQ_{size}(g), & \text{descendant of } P_\delta \\ 5 * SCQ_{size}(g), & \text{otherwise} \end{cases} \tag{4}$$

Furthermore, the total memory size in bytes of the entire AST is given by the following[3]:

$$AST_{memory} = \sum_{g \in AST} SCQ_{memory}(g) \tag{5}$$

Following Equation 5, Table 1 provides the memory required in bytes for the AST of various PMLTL formulas. Based on the deadline d (which determines $max(H_p)$), there is a minimum and maximum size for each node such that $max(max\{s.wpd \mid s \in \mathbb{S}_g\} - g.bpd,0) + max\{p.ub-p.lb \mid p \in \mathbb{TP}_g\} + 1 \leq\ SCQ_{size}(g)\ \leq 2 * (max(max\{s.wpd \mid s \in \mathbb{S}_g\} - g.bpd,0) + max\{p.ub-p.lb \mid p \in \mathbb{TP}_g\}) + 1$. While probabilistic operators increase the size of the AST, the sizing equation has a fixed upper bound (compared to linearly increasing with $max(H_p)$ as in [65]) such that we can look as far into the future as desired without increasing the SCQ size beyond this upper bound.

Table 1. AST_{memory} (in bytes) of example PMLTL formulas where $a_0,a_1,a_2,a_3 \in \mathcal{AP}$

Example PMLTL formulas	Deadline d						
	-15	-5	0	5	15	30	45
$\Box_{[0,30]} a_0$	10	10	10	10	10	10	10
$P_{0.80}(\Box_{[0,30]} a_0)$	749	749	737	677	557	389	389
$\Box_{[0,10]} a_0 \wedge \Diamond_{[0,20]} a_1$	125	125	125	125	95	75	75
$P_{0.95}(\Box_{[0,10]} a_0 \wedge \Diamond_{[0,20]} a_1)$	1025	1025	1013	953	689	545	545
$((\Box_{[0,5]} a_0) \mathcal{U}_{[0,10]} a_1) \vee ((\Box_{[0,5]} a_2) \mathcal{U}_{[0,10]} a_3)$	445	445	435	385	245	245	245
$P_{0.85}((\Box_{[0,5]} a_0) \mathcal{U}_{[0,10]} a_1) \vee P_{0.98}((\Box_{[0,5]} a_2) \mathcal{U}_{[0,10]} a_3)$	1551	1551	1527	1383	831	831	831

4 Autonomous Vehicle Case Study

Autonomous vehicles (AVs) are common targets of multimodal prediction research due to the safety-critical and multimodal nature of vehicles and other road agents (e.g., pedestrians). Conventionally, multimodal prediction has been produced by purely physics-based approaches (e.g., Monte Carlo [11,60]). Physics-based approaches are known for having low computational cost but are only valid for short prediction horizons (i.e., less than one second). For this reason, deep learning methods have gained recent popularity as they can accurately predict longer prediction horizons (i.e., several seconds); however, deep learning methods experience a higher

[3] This only includes the memory requirement of the SCQs and doesn't consider the node's metadata.

computational cost in terms of memory requirements and latency [27,44]. In this section, we utilize a physics-based and deep learning-based multimodal model predictor.

4.1 F1Tenth Autonomous Racing

Fig. 4. F1Tenth autonomous vehicle.

System Description. The F1Tenth platform is a ROS-based 1/10th scale autonomous racing vehicle equipped with a NVIDIA® Jetson Xavier NX, LiDAR, and vehicle electronic speed controller (VESC) as shown in Figure 4. For our experiments, we utilize the OpenAI Gym simulator provided by the creators of the F1Tenth platform [45]. The simulator models up to two vehicles utilizing the single-track model from [2] along with parameters derived from the physical F1Tenth platform. The VESC and LiDAR for each vehicle are directly simulated within the simulator, and although the LiDAR can be utilized to provide vehicle localization (e.g., the particle filter in [61]), the simulator broadcasts the ground truth odometry for each vehicle.

Implementation. We simulate a multi-agent race on the NVIDIA® Jetson Xavier NX, where an ego-vehicle (i.e., the vehicle operating MMPRV) races against an opponent vehicle. The ego-vehicle monitors the following safety specification utilizing MMPRV:

$$\varphi = a_0 \, \mathcal{U}_{[0,15]} \; P_{0.98}(a_1 \vee a_2) \tag{6}$$

where $a_0, a_1, a_2 \in \mathcal{AP}$ defined in Table 2 such that v_{ego} is the ego-vehicle's velocity, and x_{ego}, y_{ego}, x_{opp}, and y_{opp} are the x- and y-coordinates of the ego-vehicle and opponent vehicle, respectively. The specification φ aims to ensure that "the ego-vehicle decelerates for the next 15 time steps until the probability that either the ego-vehicle comes to a complete stop or the vehicles being greater than 0.58 meters apart is greater than or equal to 98%", where 0.58 meters is the F1Tenth platform's length. Note that φ is reevaluated for each time index, creating an implicit global operator $\square_{[0,M]}\varphi$ (i.e., M is the end of mission-time).

Table 2. Atomic Propositions in Equation 6

Atomic	Atomic Proposition	English Translation
a_0	$v_{ego}[i] - v_{ego}[i-1] < 0.0$	Ego-vehicle is decelerating
a_1	$v_{ego} == 0.0$	Ego-vehicle is stopped
a_2	$\sqrt{(x_{ego}-x_{opp})^2+(y_{ego}-y_{opp})^2} > 0.58$	Ego-vehicle and opponent do not collide

The ego-vehicle utilizes a Model Predictive Control (MPC) controller to minimize its deviation from the reference trajectory (i.e., the track centerline). The overall goal of MPC is to minimize a cost function while also following a series of given constraints (e.g., physical dynamics and limitations of a system) [54]; therefore, the ego-vehicle's objective cost function is as follows [4]:

$$\min_{\mathbf{X},\mathbf{U}} \sum_{k=0}^{N-1} \Big((\mathbf{X}_k - \mathbf{X}_{ref,k})^T \mathbf{Q} (\mathbf{X}_k - \mathbf{X}_{ref,k}) + \mathbf{U}_k^T \mathbf{R} \mathbf{U}_k \Big) + (\mathbf{X}_N - \mathbf{X}_{ref,N})^T \mathbf{Q} (\mathbf{X}_N - \mathbf{X}_{ref,N})$$

$$\text{such that: } \mathbf{X}_0 = \text{current state and } \mathbf{X}_{k+1} = \mathbf{A}\mathbf{X}_k + \mathbf{B}\mathbf{U}_k + \mathbf{C} \qquad \forall k \in 0,1,2,...,N \tag{7}$$

$$0.5\frac{m}{s} \leq V_k \leq 5.0\frac{m}{s} \text{ and } -25^\circ \leq \delta_k \leq 25^\circ \qquad \forall k \in 0,1,2,...,N$$

where N is the prediction horizon, $\mathbf{X}_k^T = [x_k, y_k, \psi_k]^T$, $\mathbf{U}_k^T = [V_k, \delta_k]^T$, $\mathbf{X}_{ref,k}$ is the reference trajectory, x and y are the x- and y-coordinates of the center of gravity of the vehicle in the global frame, ψ is the angle of the vehicle relative to the x-axis, V is the velocity, δ is the steering angle, $\mathbf{Q}$ is a positive semi-definite weight matrix of size 3x3, and $\mathbf{R}$ is a positive

[4] Additional details available at *https://temporallogic.org/research/MMPRV/MPC.pdf*

definite weight matrix of size 2x2. To define **A**, **B**, and **C**, the kinematic bicycle model derived from [48] is discretized and linearized around a reference point to the following form:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \psi_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -V_{ref} sin(\psi_{ref}+\beta_{ref})dt \\ 0 & 1 & V_{ref} cos(\psi_{ref}+\beta_{ref})dt \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \psi_k \end{bmatrix} + \begin{bmatrix} cos(\psi_{ref}+\beta_{ref})dt & 0 \\ sin(\psi_{ref}+\beta_{ref})dt & 0 \\ \frac{cos(\beta_{ref})}{\ell_f+\ell_r} tan(\delta_{ref})dt & \frac{V_{ref} cos(\beta_{ref})}{(\ell_f+\ell_r)cos^2(\delta_{ref})} dt \end{bmatrix} \begin{bmatrix} V_k \\ \delta_k \end{bmatrix} + \begin{bmatrix} V_{ref}\psi_{ref} sin(\psi_{ref}+\beta_{ref})dt \\ -V_{ref}\psi_{ref} cos(\psi_{ref}+\beta_{ref})dt \\ \frac{-V_{ref}\delta_{ref} cos(\beta_{ref})}{(\ell_f+\ell_r)cos^2(\delta_{ref})} dt \end{bmatrix} \quad (8)$$

where β is the slip angle and ℓ_f and ℓ_r are the distances from the center of gravity to the front and rear axles. The ego-vehicle utilizes the operator splitting quadratic program (OSQP) [56] to solve the cost function (Equation 7), and the output is a sequence of states ($\mathbf{X}_k$) and inputs ($\mathbf{U}_k$) for the next N time steps that minimize the cost function. Only the first inputs are applied to the ego-vehicle as MPC is recalculated for each timestamp (i.e., receding horizon control), but these predicted inputs and states are utilized as the predicted velocity and trajectory for the ego-vehicle (similar to [65]).

The opponent vehicle utilizes Rapidly exploring Random Trees (RRT*) to select the path that avoids obstacles and maximizes the progress along the centerline [32] and the pure pursuit algorithm to follow this path [18]. The opponent vehicle's current position is broadcast to the ego-vehicle, but the ego-vehicle is unaware of the opponent vehicle's control strategy. As a result, the ego-vehicle utilizes a naïve approach where k_{opp} possible future trajectories of the opponent's vehicle are generated based on random behavior modes [11] produced by Monte Carlo (MC) random sampling [25] over the model's input space in Equation 8.

To determine the verdict of $a_2 \in \mathcal{AP}$ in Equation 6, the single predicted trajectory of the ego-vehicle's MPC controller and the k_{opp} predicted opponent vehicle trajectories produce $1 * k_{opp}$ signal tuples (v_{ego}, x_{ego}, y_{ego}, x_{opp}, y_{opp}) for each predicted time step. R2U2's booleanizer [29] produces boolean atomics from these signal values to populate $K = 1 * k_{opp}$ finite traces with prediction until the verdict of φ is determinable. Note that equal likelihood is assumed for this approach; therefore, the probability space $(\Omega_i, \mathcal{F}_i, \mathbb{P}_i)$ is defined according to Definition 12 such that $\forall j \in [0, K-1]\ \mathbb{P}_i(\{a \in \hat{\pi}_j(i)\}) = \frac{1}{K}$ and $\mathbb{P}_i(\{a \notin \hat{\pi}_j(i)\}) = 0$.

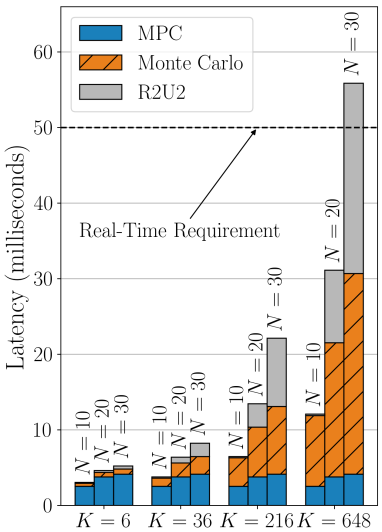
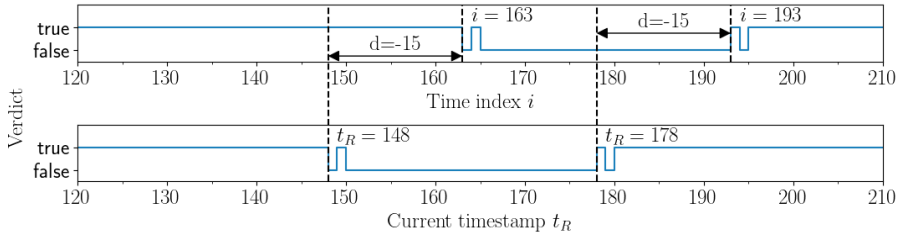


Fig. 5. 90% tail latency for varying values of K and N.

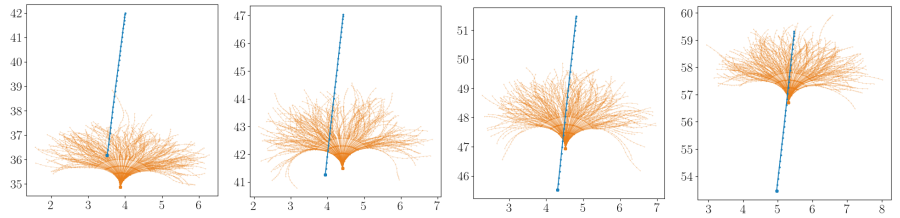
Real-Time Feasibility. To evaluate the real-time feasibility of our implementation, we record the 90% tail latency over 10,000 time steps (i.e., 90% of reported latencies are less than or equal to the given latency) of the MPC controller, MC multimodal predictor, and R2U2 (i.e., the latency of MMPRV not including the prediction time) for varying values of K and N; the latency increases linearly with increasing values of K and N as one would intuitively expect as shown in Figure 5. The F1Tenth vehicle has an update rate of 20 Hz; therefore, we assume the ego-vehicle's control loop must also operate at 20 Hz (i.e., every 50 milliseconds) as shown by the dashed line in Figure 5. Even though we only consider MMPRV and the

associated predictors in our latency analysis, it's important to note that the control loop includes other processes such as sensor processing and localization. Furthermore, although the latencies of the model predictors are specific to our implementation, MMPRV allows for *any* user-defined model predictor to be utilized with *any* values of K and N. While the chosen model predictor(s) must be feasible in real-time, R2U2 must also be able to run in real-time to produce verdicts without delay. Generally, with low values of K and N, R2U2 performs relatively quickly (e.g., if $K=36$ and $N=max(H_p)=30$, R2U2 has a 90% tail latency of 0.759 milliseconds).

MMPRV Results. For simplicity, we will assume that when the ego-vehicle detects violations of φ (Equation 6), the chosen mitigation action is to apply the brakes where a deadline $d=-15$ is required. Figure 6a displays the evaluation of $\pi,i,-15 \models \varphi$ for time index i reported by MMPRV at timestamp t_R. Note that for every timestamp t_R, the verdict of $\pi,i,-15 \models \varphi$ is reported fifteen time steps before i such that $i=t_R-d=t_R-(-15)$. For example, when $t_R=148$, the verdict was reported for $i=163$ such that $163=148-(-15)$. Figures 6b, 6c, 6d, and 6e display trajectory predictions with a $N=max(H_p)=\varphi.wpd-d=15-(-15)=30$ that were utilized to populate $K=216$ finite traces with prediction for evaluation of $\pi,i,-15 \models \varphi$. Note, that $K=216$ and $N=30$ meets the real-time requirement according to Figure 5.



(a) Results for $\pi,i,-15 \models a_0 \, \mathcal{U}_{[0,15]} \, P_{0.98}(a_1 \vee a_2)$



(b) $t_R=130$ (c) $t_R=150$ (d) $t_R=170$ (e) $t_R=200$

Fig. 6. MMPRV results. (a) displays the verdicts for time index i returned at timestamp t_R. (b), (c), (d), and (e) show the trajectory predictions for the ego-vehicle (blue) and opponent vehicle (orange) with $N=30$.

4.2 Argoverse Autonomous Driving

Table 3. Accuracy on Argoverse test set [14] ($k=6$)

Model	minADE	minFDE	MR
LaneGCN[37]	0.870	1.362	16.2%
LAformer[40]	0.772	1.163	12.5%
mmTransformer[41]	0.844	1.338	15.4%
Lane Transformer [62]	0.866	1.316	15.2%
HiVT-64[67]	0.807	1.243	14.0%
HiVT-128[67]	0.774	1.169	12.7%

SOTA Multimodal Models. The Argoverse motion forecasting dataset [14] contains 323,557 traffic scenarios captured in Miami and Pittsburgh. Each scenario captures five seconds at 10 Hz such that each model is expected to predict the future three seconds (i.e., $N=30$) given the past two seconds of observed trajectories. We examine six SOTA open-source deep neural network (DNN) multimodal predictors trained and evaluated on the Argoverse dataset: LaneGCN [37], LAformer [40], mmTransformer [41], Lane Transformer [62], HiVT-64 [67], and HiVT-128 [67]. Each of

these DNNs produces $k=6$ multimodal predictions for each vehicle in the traffic scene, and Table 3 displays the accuracy of these predictions based on standard metrics. The minimum Average Displacement Error (minADE) and minimum Final Displacement Error (minFDE) are the average and endpoint L_2 distance errors, respectively, between the best-predicted and ground truth trajectories, and the Miss Rate (MR) is the percentage where none of the predicted trajectories have an endpoint L_2 distance error within two meters from the ground truth.

Real-Time Feasibility. Deep learning models are known to have large variations in latency based on several factors such as input/output data, model architecture, hardware platform, etc. [39]; therefore, we deploy all six models on four different hardware platforms: a laptop with a 2.8 GHz Quad-Core Intel® Core i7 CPU (Laptop CPU), the NVIDIA® Jetson Xavier NX with its GPU enabled (Jetson GPU), a desktop with a 3.00 GHz 8-core Intel® Xeon® Gold 6354 CPU (Desktop CPU), and that same desktop with the NVIDIA® A40 GPU enabled (Desktop GPU). To simulate predicting the multimodal trajectories of 16 vehicles in a given city traffic scene, each of these models (except HiVT-64 and HiVT-128) are run with a batch size of 16. HiVT-64 and HiVT-128 uniquely compute the multimodal predictions for *all* agents in a given traffic scene within a single forward pass; therefore, they are run with a batch size of one to provide an accurate comparison. The latencies of these models are captured utilizing the official open-source implementations, and the standard 90% tail latency [49] is reported in Figure 7.

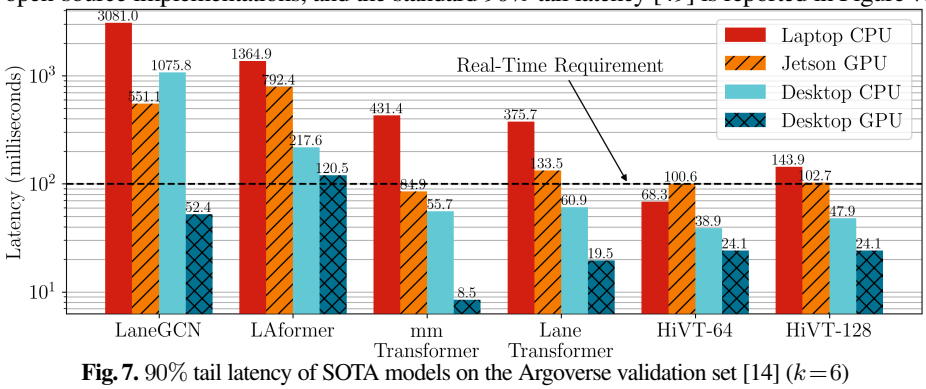


Fig. 7. 90% tail latency of SOTA models on the Argoverse validation set [14] ($k=6$)

Since the Argoverse dataset was captured at 10 Hz, we assume the control loop must also operate at 10 Hz (i.e., every 100 milliseconds) as indicated by the dashed line in Figure 7. While we only analyze the latency of the multimodal predictor, the control loop also includes tasks such as localization, object detection, lane detection, planning, etc. [39]. Consequently, these SOTA multimodal predictors must operate with a latency much less than 100 milliseconds to allow these other tasks enough time to execute; therefore, according to Figure 7, there are only a few cases that meet this real-time requirement. When these SOTA DNNs target the NVIDIA® Jetson Xavier NX, they all fall short of this real-time requirement; instead, these models generally require the computing capabilities of the Desktop GPU to be feasible in real-time.

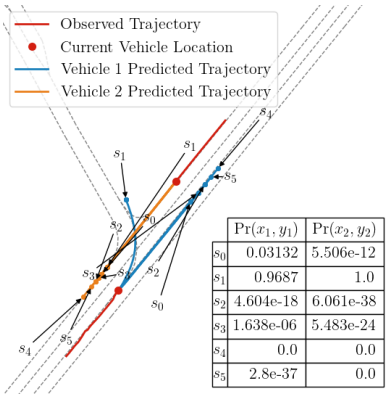
To compare our physics-based predictor in Section 4.1 and the SOTA DNNs, we reran the DNNs on the NVIDIA® Jetson Xavier NX to simulate predicting the multimodal trajectories of a single opponent vehicle (i.e., batch size of one), and mmTransformer had the lowest latency with 37.8 milliseconds. Therefore, in order to achieve SOTA accuracy, we must experience a 55x slowdown from our Monte Carlo approach (i.e., Figure 5: $K=6$, $N=30 \mapsto 0.69$ milliseconds).

MMPRV Results. MMPRV determines the verdict of the following safety specification by deadline $d=0$ (with $max(H_p)=30$):

$$\varphi = P_{0.90}(\Box_{[0,30]} a), \text{ where } a = \sqrt{(x_1-x_2)^2+(y_1-y_2)^2} > 1.7$$

where a is an atomic proposition and x_1, y_1, x_2, and y_2 are the x- and y-coordinates of Vehicle 1 and Vehicle 2, respectively. The specification φ aims to ensure that "the probability of the distance between two vehicles being greater than 1.7 meters for the next 30 time steps (i.e., next 3 seconds) is greater than or equal to 90%", where 1.7 meters is the average width of a vehicle.

The mmTransformer is utilized to predict $k_{veh1} = 6$ and $k_{veh2} = 6$ multimodal trajectories for two nearby vehicles labeled Vehicle 1 and Vehicle 2, respectively. This produces $k_{veh1} * k_{veh2} = 36$ signal tuples (x_1, y_1, x_2, y_2) which are input as raw float values into R2U2's booleanizer [29] to populate $K = 36$ finite traces with prediction. Each multimodal trajectory produced by mmTransformer also has an associated probability; therefore, the probability space $(\Omega_i, \mathcal{F}_i, \mathbb{P}_i)$ is defined according to Definition 12 such that $\forall j \in [0, K-1]\ \mathbb{P}_i(\{a \in \hat{\pi}_j(i)\}) = \Pr(x_1, y_1) * \Pr(x_2, y_2)$ and $\mathbb{P}_i(\{a \notin \hat{\pi}_j(i)\}) = 0$ where $\Pr(x_1, y_1)$ and $\Pr(x_2, y_2)$ are the probabilities of (x_1, y_1) and (x_2, y_2) being the ground truth.



	$\Pr(x_1, y_1)$	$\Pr(x_2, y_2)$
s_0	0.03132	5.506e-12
s_1	0.9687	1.0
s_2	4.604e-18	6.061e-38
s_3	1.638e-06	5.483e-24
s_4	0.0	0.0
s_5	2.8e-37	0.0

Fig. 8. Multimodal trajectory prediction for Vehicle 1 and 2. s_j is the trajectory produced by the j^{th} multimodal prediction.

Figure 8 illustrates an interaction between Vehicle 1 and 2 at an intersection that results in $\pi, i, d \not\models \varphi$. According to the predictions of mmTransformer, Vehicle 1 might make a left turn (with $\Pr(x_1, y_1) = 0.9687$) that will result in a collision with Vehicle 2 whose predicted trajectories are straight. Note that the verdict is still a predicted result that is only as accurate as the prediction.

5 Conclusion and Future Work

MMPRV allows for *any* user-defined model predictor, including both unimodal and multimodal model predictors, while guaranteeing a verdict for a given PMLTL specification by a given deadline. The additional support of multimodal prediction allows for applicability to complex systems where multiple future behavior modes are plausible. As shown through our case studies, SOTA DNN multimodal predictors struggle to meet real-time requirements, especially when targeting embedded computing platforms. This motivates future work to continue investigating computationally lighter methods (e.g., Interacting Multiple Model Kalman Filters [31] or maneuver-based recurrent neural networks [21]) and investigate potential avenues for acceleration of these SOTA DNNs through techniques such as pruning or precision reduction [57]. Additionally, while we primarily focused on quantifying probabilistic multimodal predictions, future work includes quantifying distributed signal values within PMLTL specifications through techniques such as chance constraints [28,34,36] and conformal prediction [38].

References

1. Althoff, M., Dolan, J.M.: Online verification of automated road vehicles using reachability analysis. IEEE Transactions on Robotics **30**(4), 903–918 (2014). https://doi.org/https://doi.org/10.1109/TRO.2014.2312453
2. Althoff, M., Koschi, M., Manzinger, S.: Commonroad: Composable benchmarks for motion planning on roads. In: 2017 IEEE Intelligent Vehicles Symposium (IV). pp. 719–726. IEEE (2017). https://doi.org/https://doi.org/10.1109/IVS.2017.7995802
3. Althoff, M., Stursberg, O., Buss, M.: Model-based probabilistic collision detection in autonomous driving. IEEE Transactions on Intelligent Transportation Systems **10**(2), 299–310 (2009). https://doi.org/https://doi.org/10.1109/TITS.2009.2018966

4. Alur, R., Henzinger, T.A.: Real-time Logics: Complexity and Expressiveness. In: LICS. pp. 390–401. IEEE (1990)
5. Aurandt, A., Jones, P.H., Rozier, K.Y.: Runtime verification triggers real-time, autonomous fault recovery on the cysat-i. In: NASA Formal Methods Symposium. pp. 816–825. Springer (2022), *https://temporallogic.org/research/CySat-NFM22/CySat-NFM22.pdf*
6. Babaee, R., Ganesh, V., Sedwards, S.: Accelerated learning of predictive runtime monitors for rare failure. In: International Conference on Runtime Verification. pp. 111–128. Springer (2019). https://doi.org/https://doi.org/10.1007/978-3-030-32079-9_7
7. Babaee, R., Gurfinkel, A., Fischmeister, S.: $\mathcal{P}$revent: A predictive run-time verification framework using statistical learning. In: SEFM. pp. 205–220. Springer (2018). https://doi.org/https://doi.org/10.1007/978-3-030-03769-7_11
8. Babaee, R., Gurfinkel, A., Fischmeister, S.: Predictive run-time verification of discrete-time reachability properties in black-box systems using trace-level abstraction and statistical learning. In: RV. pp. 187–204. Springer (2018). https://doi.org/https://doi.org/10.1007/978-3-030-03769-7_11
9. Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Ničković, D., Sankaranarayanan, S.: Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. Lectures on Runtime Verification: Introductory and Advanced Topics pp. 135–175 (2018). https://doi.org/https://doi.org/10.1007/978-3-319-75632-5_5
10. Bortolussi, L., Cairoli, F., Paoletti, N., Smolka, S.A., Stoller, S.D.: Neural predictive monitoring. In: Runtime Verification: 19th International Conference, RV 2019, Porto, Portugal, October 8–11, 2019, Proceedings 19. pp. 129–147. Springer (2019). https://doi.org/https://doi.org/10.1007/978-3-030-32079-9_8
11. Broadhurst, A., Baker, S., Kanade, T.: Monte carlo road safety reasoning. In: IEEE Proceedings. Intelligent Vehicles Symposium, 2005. pp. 319–324. IEEE (2005), *https://doi.org/10.1109/IVS.2005.1505122*
12. Cairoli, F., Bortolussi, L., Paoletti, N.: Neural predictive monitoring under partial observability. In: Runtime Verification: 21st International Conference, RV 2021, Virtual Event, October 11–14, 2021, Proceedings 21. pp. 121–141. Springer (2021). https://doi.org/https://doi.org/10.1007/978-3-030-88494-9_7
13. Cauwels, M., Hammer, A., Hertz, B., Jones, P., Rozier, K.Y.: Integrating Runtime Verification into an Automated UAS Traffic Management System. In: DETECT. Springer, L'Aquila, Italy (September 2020), *https://r2u2.temporallogic.org/wp-content/uploads/2020/12/CHHJR20.pdf*
14. Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al.: Argoverse: 3d tracking and forecasting with rich maps. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8748–8757 (2019), *https://doi.org/10.1109/CVPR.2019.00895*
15. Chou, Y., Yoon, H., Sankaranarayanan, S.: Predictive runtime monitoring of vehicle models using bayesian estimation and reachability analysis. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2111–2118. IEEE (2020). https://doi.org/https://doi.org/10.1109/IROS45743.2020.9340755
16. Cimatti, A., Tian, C., Tonetta, S.: Assumption-based runtime verification with partial observability and resets. In: International Conference on Runtime Verification. pp. 165–184. Springer (2019). https://doi.org/https://doi.org/10.1007/978-3-030-32079-9_10
17. Cleaveland, M., Sokolsky, O., Lee, I., Ruchkin, I.: Conservative safety monitors of stochastic dynamical systems. In: NASA Formal Methods Symposium. pp. 140–156. Springer (2023). https://doi.org/https://doi.org/10.1007/978-3-031-33170-1_9
18. Coulter, R.C.: Implementation of the pure pursuit path tracking algorithm. Carnegie Mellon University, The Robotics Institute (1992)
19. Dabney, J.B., Badger, J.M., Rajagopal, P.: Adding a verification view for an autonomous real-time system architecture. In: Proceedings of SciTech Forum. p. Online. 2021-0566, AIAA (January 2021). https://doi.org/https://doi.org/10.2514/6.2021-0566
20. Dabney, J.B., Badger, J.M., Rajagopal, P.: Trustworthy autonomy for gateway vehicle system manager. In: 2023 IEEE Space Computing Conference (SCC). pp. 57–62. IEEE (2023). https://doi.org/https://doi.org/10.1109/SCC57168.2023.00018

21. Deo, N., Trivedi, M.M.: Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In: 2018 IEEE intelligent vehicles symposium (IV). pp. 1179–1184. IEEE (2018), *https://doi.org/10.1109/IVS.2018.8500493*
22. Ferrando, A., Cardoso, R.C., Farrell, M., Luckcuck, M., Papacchini, F., Fisher, M., Mascardi, V.: Bridging the gap between single-and multi-model predictive runtime verification. Formal Methods in System Design pp. 1–33 (2022). https://doi.org/https://doi.org/10.1007/s10703-022-00395-7
23. Ferrando, A., Delzanno, G.: Incrementally predictive runtime verification. In: CILC. pp. 92–106 (2021)
24. Fisher, M., Mascardi, V., Rozier, K.Y., Schlingloff, B.H., Winikoff, M., Yorke-Smith, N.: Towards a framework for certification of reliable autonomous systems. Autonomous Agents and Multi-Agent Systems **35**, 1–65 (2021). https://doi.org/https://doi.org/10.1007/s10458-020-09487-2
25. Hammersley, J.M.: Monte carlo methods for solving multivariable problems. Annals of the New York Academy of Sciences **86**(3), 844–874 (1960)
26. Hertz, B., Luppen, Z., Rozier, K.Y.: Integrating runtime verification into a sounding rocket control system. In: NASA Formal Methods Symposium. pp. 151–159. Springer (2021). https://doi.org/https://doi.org/10.1007/978-3-030-76384-8_10
27. Huang, Y., Du, J., Yang, Z., Zhou, Z., Zhang, L., Chen, H.: A survey on trajectory-prediction methods for autonomous driving. IEEE Transactions on Intelligent Vehicles **7**(3), 652–674 (2022). https://doi.org/https://doi.org/10.1109/TIV.2022.3167103
28. Jha, S., Raman, V., Sadigh, D., Seshia, S.A.: Safe autonomy under perception uncertainty using chance-constrained temporal logic. Journal of Automated Reasoning **60**, 43–62 (2018), *https://doi.org/10.1007/s10817-017-9413-9*
29. Johannsen, C., Jones, P., Kempa, B., Rozier, K.Y., Zhang, P.: R2u2 version 3.0: Re-imagining a toolchain for specification, resource estimation, and optimized observer generation for runtime verification in hardware and software. In: International Conference on Computer Aided Verification. pp. 483–497. Springer (2023), *https://research.temporallogic.org/papers/JJKRZ23.pdf*
30. Johannsen, C., Kempa, B., Jones, P.H., Rozier, K.Y., Wongpiromsarn, T.: Impossible made possible: Encoding intractable specifications via implied domain constraints. In: International Conference on Formal Methods for Industrial Critical Systems. pp. 151–169. Springer (2023), *https://research.temporallogic.org/papers/JKJRW23.pdf*
31. Kaempchen, N., Weiss, K., Schaefer, M., Dietmayer, K.C.: Imm object tracking for high dynamic driving maneuvers. In: IEEE Intelligent Vehicles Symposium, 2004. pp. 825–830. IEEE (2004), *https://doi.org/10.1109/IVS.2004.1336491*
32. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The international journal of robotics research **30**(7), 846–894 (2011). https://doi.org/https://doi.org/10.1177/0278364911406761
33. Kempa, B., Zhang, P., Jones, P.H., Zambreno, J., Rozier, K.Y.: Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2. In: Proceedings of the 18th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS). pp. 196–214. Lecture Notes in Computer Science (LNCS), Springer, Vienna, Austria (September 2020), *http://research.temporallogic.org/papers/KZJZR20.pdf*
34. Kyriakis, P., Deshmukh, J.V., Bogdan, P.: Specification mining and robust design under uncertainty: A stochastic temporal logic approach. ACM Transactions on Embedded Computing Systems (TECS) **18**(5s), 1–21 (2019), *https://doi.org/10.1145/3358231*
35. Li, J., Vardi, M.Y., Rozier, K.Y.: Satisfiability checking for mission-time LTL. In: International Conference on Computer Aided Verification. pp. 3–22. Springer (2019). https://doi.org/https://doi.org/10.1007/978-3-030-25543-5_1
36. Li, J., Nuzzo, P., Sangiovanni-Vincentelli, A., Xi, Y., Li, D.: Stochastic contracts for cyber-physical system design under probabilistic requirements. In: Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design. pp. 5–14 (2017), *https://doi.org/10.1145/3127041.3127045*

37. Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R.: Learning lane graph representations for motion forecasting. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. pp. 541–556. Springer (2020), *https://doi.org/10.1007/978-3-030-58536-5_32*
38. Lindemann, L., Qin, X., Deshmukh, J.V., Pappas, G.J.: Conformal prediction for stl runtime verification. In: Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023). pp. 142–153 (2023), *https://doi.org/10.1145/3576841.3585927*
39. Liu, L., Wang, Y., Shi, W.: Understanding time variations of dnn inference in autonomous driving. arXiv preprint arXiv:2209.05487 (2022), *https://doi.org/10.48550/arXiv.2209.05487*
40. Liu, M., Cheng, H., Chen, L., Broszio, H., Li, J., Zhao, R., Sester, M., Yang, M.Y.: Laformer: Trajectory prediction for autonomous driving with lane-aware scene constraints. arXiv preprint arXiv:2302.13933 (2023)
41. Liu, Y., Zhang, J., Fang, L., Jiang, Q., Zhou, B.: Multimodal motion prediction with stacked transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7577–7586 (2021), *https://doi.org/10.1109/CVPR46437.2021.00749*
42. Ma, M., Stankovic, J., Bartocci, E., Feng, L.: Predictive monitoring with logic-calibrated uncertainty for cyber-physical systems. ACM Transactions on Embedded Computing Systems (TECS) **20**(5s), 1–25 (2021), *https://doi.org/10.1145/3477032*
43. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, pp. 152–166. Springer (2004), *https://doi.org/10.1007/978-3-540-30206-3_12*
44. Mozaffari, S., Al-Jarrah, O.Y., Dianati, M., Jennings, P., Mouzakitis, A.: Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. IEEE Transactions on Intelligent Transportation Systems **23**(1), 33–47 (2020). https://doi.org/https://doi.org/10.1109/TITS.2020.3012034
45. O'Kelly, M., Zheng, H., Karthik, D., Mangharam, R.: F1tenth: An open-source evaluation environment for continuous control and reinforcement learning. In: NeurIPS 2019 Competition and Demonstration Track. pp. 77–89. PMLR (2020)
46. Pinisetty, S., Jéron, T., Tripakis, S., Falcone, Y., Marchand, H., Preoteasa, V.: Predictive runtime verification of timed properties. Journal of Systems and Software **132**, 353–365 (2017). https://doi.org/https://doi.org/10.1016/j.jss.2017.06.060
47. Qin, X., Deshmukh, J.V.: Clairvoyant monitoring for signal temporal logic. In: Formal Modeling and Analysis of Timed Systems: 18th International Conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings 18. pp. 178–195. Springer (2020). https://doi.org/https://doi.org/10.1007/978-3-030-57628-8_11
48. Rajamani, R.: Vehicle Dynamics and Control. Springer (2011)
49. Reddi, V.J., Cheng, C., Kanter, D., Mattson, P., Schmuelling, G., Wu, C.J., Anderson, B., Breughe, M., Charlebois, M., Chou, W., et al.: Mlperf inference benchmark. In: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). pp. 446–459. IEEE (2020), *https://doi.org/10.1109/ISCA45697.2020.00045*
50. Reinbacher, T., Rozier, K.Y., Schumann, J.: Temporal-logic based runtime observer pairs for system health management of real-time systems. In: Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science (LNCS), vol. 8413, pp. 357–372. Springer-Verlag (April 2014), *https://doi.org/10.1007/978-3-642-54862-8_24*
51. Rozier, K.Y., Schumann, J.: R2U2: Tool Overview. In: Proceedings of International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CUBES). vol. 3, pp. 138–156. Kalpa Publications, Seattle, WA, USA (September 2017), *https://research.temporallogic.org/papers/RS2017_RV.pdf*
52. Sadigh, D., Kapoor, A.: Safe control under uncertainty with probabilistic signal temporal logic. In: Proceedings of Robotics: Science and Systems XII (2016), *https://doi.org/10.15607/RSS.2016.XII.017*
53. Schumann, J., Rozier, K.Y., Reinbacher, T., Mengshoel, O.J., Mbaya, T., Ippolito, C.: Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems. In: PHM. pp. 381–401 (October 2013), *https://research.temporallogic.org/papers/SRRMMI15.pdf*

54. Schwenzer, M., Ay, M., Bergs, T., Abel, D.: Review on model predictive control: An engineering perspective. The International Journal of Advanced Manufacturing Technology **117**(5-6), 1327–1349 (2021). https://doi.org/https://doi.org/10.1007/s00170-021-07682-3
55. Stahl, T., Diermeyer, F.: Online verification enabling approval of driving functions—implementation for a planner of an autonomous race vehicle. IEEE Open Journal of Intelligent Transportation Systems **2**, 97–110 (2021). https://doi.org/https://doi.org/10.1109/OJITS.2021.3078121
56. Stellato, B., Banjac, G., Goulart, P., Bemporad, A., Boyd, S.: Osqp: An operator splitting solver for quadratic programs. Mathematical Programming Computation **12**(4), 637–672 (2020). https://doi.org/https://doi.org/10.1007/s12532-020-00179-2
57. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE **105**(12), 2295–2329 (2017), *https://doi.org/10.1109/JPROC.2017.2761740*
58. Tiger, M., Heintz, F.: Stream reasoning using temporal logic and predictive probabilistic state models. In: 2016 23rd International Symposium on Temporal Representation and Reasoning (TIME). pp. 196–205. IEEE (2016), *https://doi.org/10.1109/TIME.2016.28*
59. Tiger, M., Heintz, F.: Incremental reasoning in probabilistic signal temporal logic. International Journal of Approximate Reasoning **119**, 325–352 (2020), *https://doi.org/10.1016/j.ijar.2020.01.009*
60. Tran, Q., Firl, J.: Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression. In: 2014 ieee intelligent vehicles symposium proceedings. pp. 918–923. IEEE (2014). https://doi.org/https://doi.org/10.1109/IVS.2014.6856480
61. Walsh, C.H., Karaman, S.: Cddt: Fast approximate 2d ray casting for accelerated localization. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 3677–3684. IEEE (2018). https://doi.org/https://doi.org/10.1109/ICRA.2018.8460743
62. Wang, Z., Guo, J., Hu, Z., Zhang, H., Zhang, J., Pu, J.: Lane transformer: A high-efficiency trajectory prediction model. IEEE Open Journal of Intelligent Transportation Systems **4**, 2–13 (2023), *https://doi.org/10.1109/OJITS.2023.3233952*
63. Yoon, H., Chou, Y., Chen, X., Frew, E., Sankaranarayanan, S.: Predictive runtime monitoring for linear stochastic systems and applications to geofence enforcement for uavs. In: RV. pp. 349–367. Springer (2019). https://doi.org/https://doi.org/10.1007/978-3-030-32079-9_20
64. Yu, X., Dong, W., Li, S., Yin, X.: Model predictive monitoring of dynamical systems for signal temporal logic specifications. Automatica **160**, 111445 (2024), *https://doi.org/10.1016/j.automatica.2023.111445*
65. Zhang, P., Aurandt, A., Dureja, R., Jones, P.H., Rozier, K.Y.: Model predictive runtime verification for cyber-physical systems with real-time deadlines. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 158–180. Springer (2023), *https://research.temporallogic.org/papers/ZADJR23.pdf*
66. Zhang, X., Leucker, M., Dong, W.: Runtime verification with predictive semantics. In: NFM. pp. 418–432. Springer (2012). https://doi.org/https://doi.org/10.1007/978-3-642-28891-3_37
67. Zhou, Z., Ye, L., Wang, J., Wu, K., Lu, K.: Hivt: Hierarchical vector transformer for multi-agent motion prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8823–8833 (2022), *https://doi.org/10.1109/CVPR52688.2022.00862*