Runtime Model Predictive Verification on Embedded Platforms [1]

Pei Zhang, Jianwen Li, Joseph Zambreno, Phillip H. Jones, Kristin Yvonne Rozier

Presenter: Pei Zhang

Iowa State University

peizhang@iastate.edu

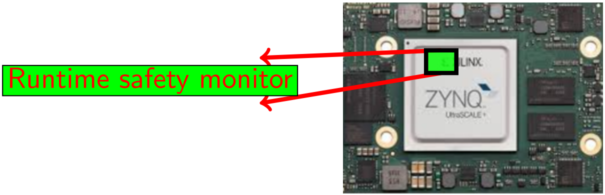
September 28, 2018

[1] Work supported by NASA ECF NNX16AR57G and NSF CAREER Award CNS-1552934.

Overview

Motivation

- Light weight monitor for embedded platform;
- Unobstrusive to a certified safety-critical system;
- Providing timely information;



Overview of Design Architecture

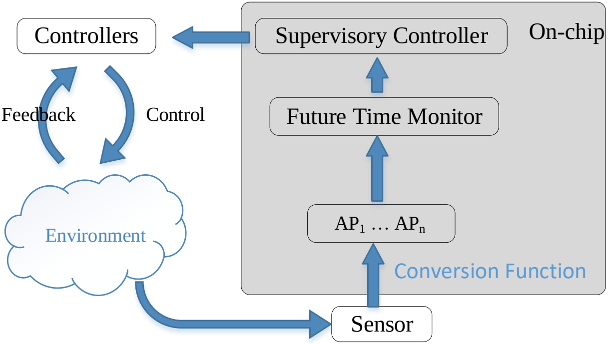


Figure: High level architecture of model predictive runtime verication.

Overview of Design Architecture

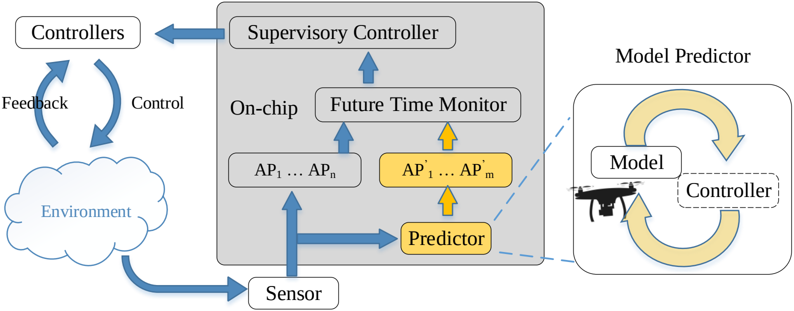


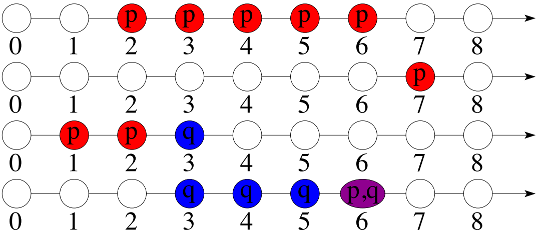
Figure: High level architecture of model predictive runtime verication.

Extending LTL for Safety Properties: MLTL

Mission-Time Linear Temporal Logic (MLTL) reasons about *bounded* timelines:

- finite set of atomic propositions {p q}
- Boolean connectives: $\neg$, $\wedge$, $\vee$, and $\rightarrow$
- temporal connectives *with time bounds*:

Symbol	Operator	Timeline
$\Box_{[2,6]} p$	ALWAYS$_{[2,6]}$	
$\Diamond_{[0,7]} p$	EVENTUALLY$_{[0,7]}$	
$p \mathcal{U}_{[1,5]} q$	UNTIL$_{[1,5]}$	
$p \mathcal{R}_{[3,8]} q$	RELEASE$_{[3,8]}$	



Model Predictive Function $\mathcal{F} : \Sigma \to \Sigma^*$.

Definition (Predictive MLTL Semantics)

Let π be a finite trace over Σ^*. The predictive truth value of the MLTL formula φ with respect to π, denoted as $[\pi \models \varphi]_p$, is an element of $\{\text{true}, \text{false}, ?\}$ defined as follows:

$$[\pi \models \varphi]_p = \begin{cases} \text{true} & \text{if } \forall \pi' \in \Sigma^* \cdot (\pi \cdot \mathcal{F}(\pi) \cdot \pi') \models \varphi; \\ \text{false} & \text{if } \forall \pi' \in \Sigma^* \cdot (\pi \cdot \mathcal{F}(\pi) \cdot \pi') \not\models \varphi; \\ ? & \text{(skip) Otherwise.} \end{cases}$$

State Space Model

A discrete state-space model defines what state a system will be in one-time step into the future:

$$x_{k+1} = Ax_k + Bu_k \tag{1}$$

$$y_k = Cx_k + Du_k \tag{2}$$

- x_k represents the state of the system at time k
- u_k represents the input acting on the system at time k
- y_k represents outputs of the system at time k
- A is a matrix that defines the internal dynamics of the system
- B is a matrix that defines how the input acting upon the system impact its state
- C is a matrix that transforms states of the system into outputs (y_k)

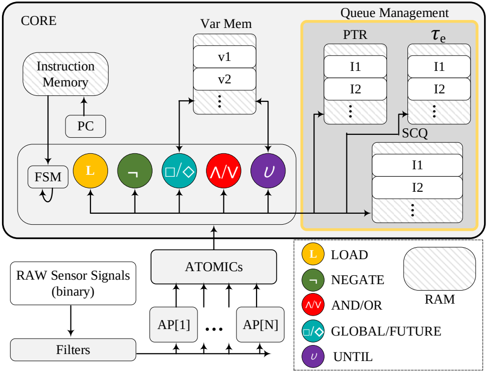
Abstract Syntax Tree (AST)

Q: How can we check MLTL satisfication in hardware?
Compile the MLTL formula into assembly code: e.g. $\Box_{[0,2]}(!a0)$

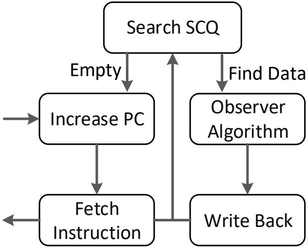
$$
\begin{aligned}
Line\,0: &\quad s0 \leftarrow load\,(a0, time)\\
Line\,1: &\quad s1 \leftarrow \neg\, s0\\
Line\,2: &\quad s2 \leftarrow \Box_{[0,2]}\, s1
\end{aligned}
$$

Each instruction are stored in a data structure called Shared Connection Queue (SCQ).

Computation Core



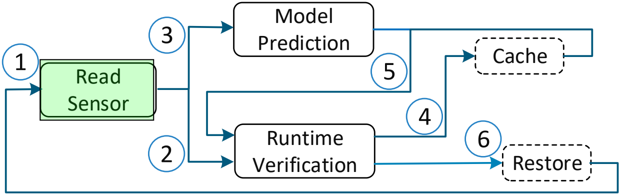
(a) Observer Processing Core.



(b) State machine transitions.

Figure: Hardware design for embedded MLTL observer processor.

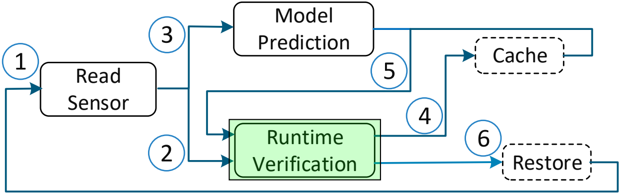
Model Predictive Runtime Verification Processing Flow



Step 1

Convert sensor data into atomic propositions (APs) using predefined atomic conversion functions.

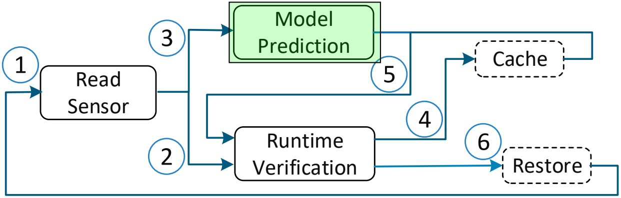
Model Predictive Runtime Verification Processing Flow



Step 2

Observer processing core conducts runtime verification over the newly received APs.

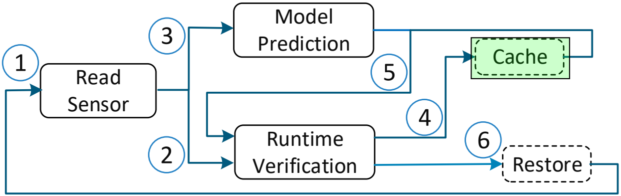
Model Predictive Runtime Verification Processing Flow



Step 3

Model Predictive Control (MPC) for a specified prediction horizon length is executed to estimate future states of the system.

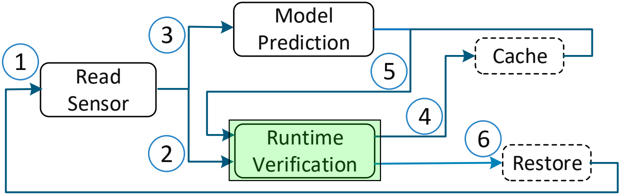
Model Predictive Runtime Verification Processing Flow



Step 4

Contents of the SCQs are cached.

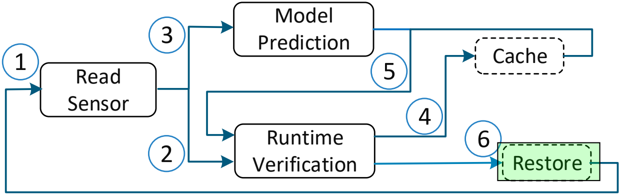
Model Predictive Runtime Verification Processing Flow



Step 5

Observer processing core conducts runtime verification over the generated trace of estimated future system states.

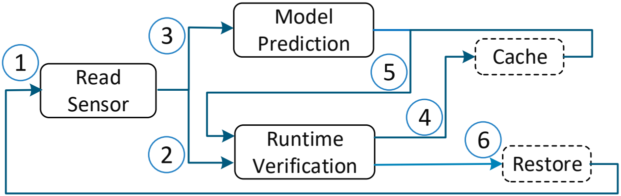
Model Predictive Runtime Verification Processing Flow



Step 6

Restore cached SCQs contents. Thereby, placing the observer processing core back into its original state.

Model Predictive Runtime Verification Processing Flow



Step 7

Return to step 1), once the next sensor sampling period starts.

MPRV on Moving a Point Mass

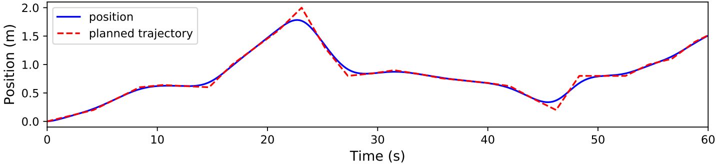


Figure: Model predictive control of the height of a point mass.

Control input force ∈ [-1N, 1N].
Cost weighting: 2 with the error in mass position and 1 with its speed.
Prediction horizon: 100.
Controller actuation update rate to 10 Hz.

- $a0$: absolute speed $< 0.1\text{m/s}$.
- $a1$: absolute value of trajectory error $< 0.08\text{m}$.

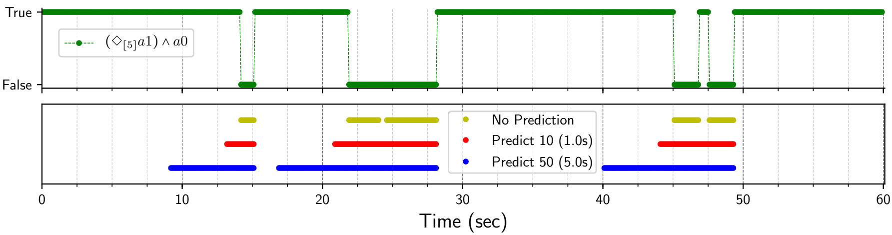
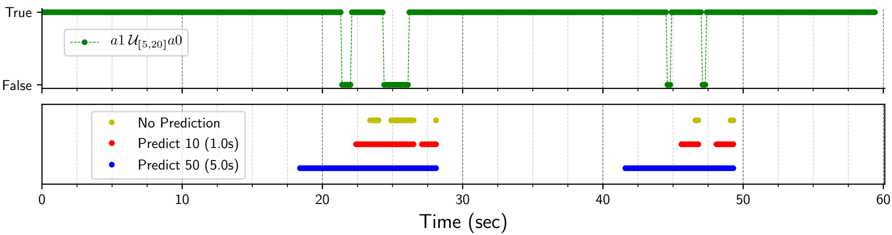


Figure: MPRV responsiveness for different prediction horizons: No prediction, 10 steps (1s), 50 steps (5s).



Disturbance

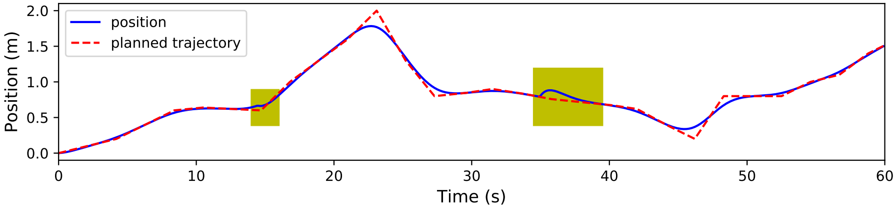


Figure: Unexpected disturbance taken place during control. The disturbance is marked in by the yellow rectangle.

an external disturbance force being applied at time 14.6s and 35.0s.

- $a0$: absolute speed < 0.5m/s.
- $a1$: absolute value of trajectory error < 0.04m.

Disturbance

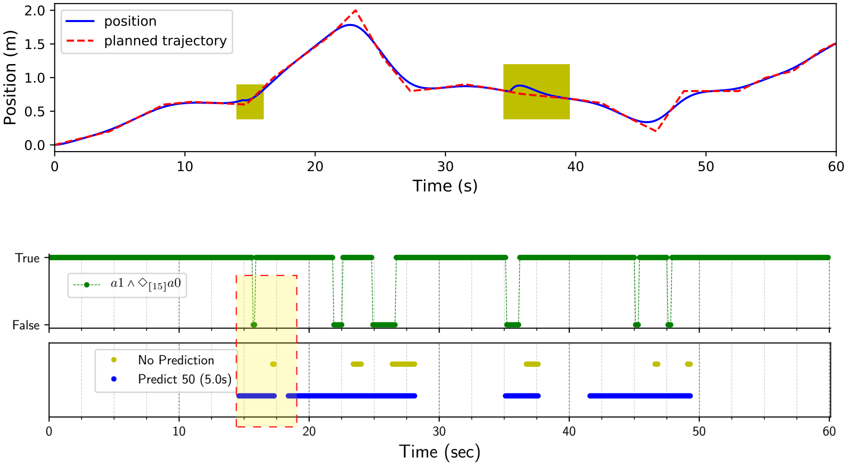


Figure: Comparasion between MPRV and normal RV with disturbance.

Disturbance

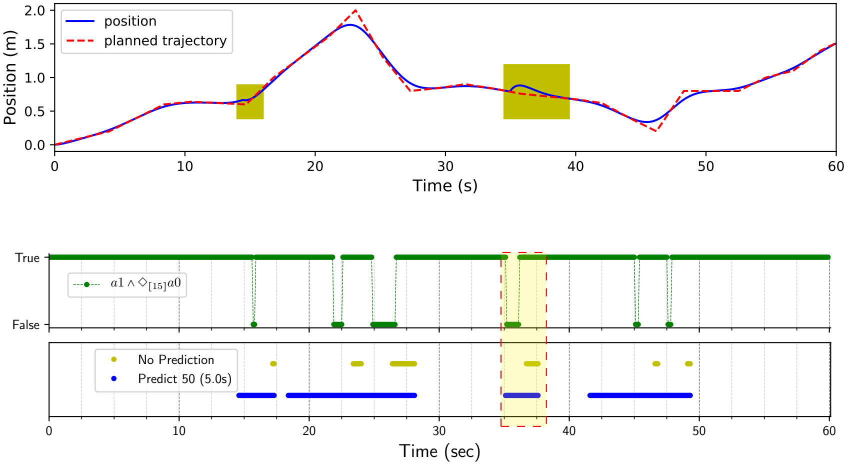
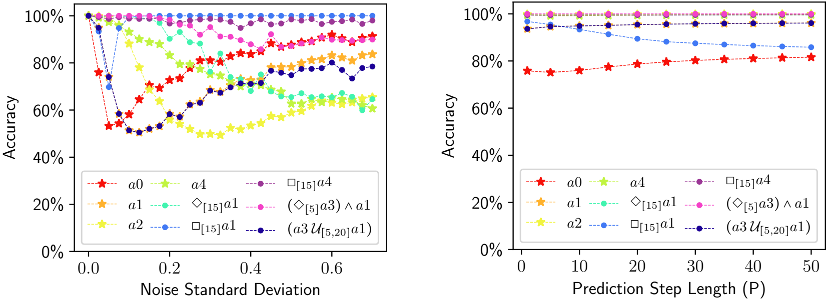


Figure: Comparasion between MPRV and normal RV with disturbance.

Utilize the MPRV Predictions under Disturbance

1. **Case 1: Disturbance instantly breaks** MLTL **rule.**
2. **Case 2: Disturbance does not instantly break the** MLTL **rule.**
3. **Case 3: Disturbance adverts the system from breaking the** MLTL **in the future.**

Sensor Noise and Prediction Horizon Length



(a) Sensor noise impact on MPRV accuracy. Prediction horizon length is 10 (1s)

(b) Prediction horizon length impact on MPRV accuracy. Sensor noise standard deviation is 0.025.

Figure: Impact of sensor noise and prediction horizon length on MPRV accuracy.

$a0$: absolute value of trajectory error < 0.04m
$a2$: absolute value of trajectory error < 0.20m
$a4$: position > 1.0 m/s

$a1$: absolute value of trajectory error < 0.08m
$a3$: absolute speed > 0.6 m/s

Worst Case Execution Time (WCET) Analysis

$$\mathcal{N}.t = t_{basic} + t_{loop} * \mathcal{N}.\mathcal{X} \leq C * \mathcal{N}.\mathcal{X} \tag{3}$$

where,

$$\mathcal{N}.\mathcal{X} = \begin{cases} \sum(\mathcal{N}.iSCQ) & \mathcal{N} \textit{ is binary operator} \\ P+1 & \mathcal{N} \textit{ is unary operator} \end{cases} \tag{4}$$

t_{basic} is the time for 'Fetch Instruction' and 'Increase PC' etc. in Fig. 3(b)
t_{loop} is the time for 'Observer Algorithm'
C is a constant associated with the hardware computation core pipeline.
In our design, the execution time is bounded by $C = 24e^{-8}$(unit: second)[2].

[2] Based on our hardware running at a clock frequency of 100 MHz.

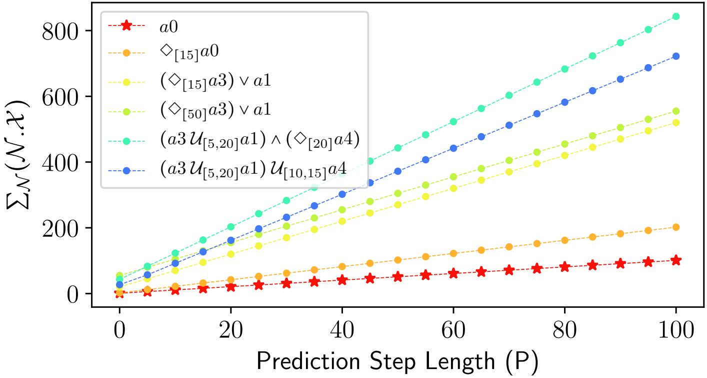


Figure: Relationship between $\mathcal{N}.\mathcal{X}$ and prediction horizon length for MLTL formulas of varying complexity.

MPRV computational complexity: $\mathcal{O}(max(\mathcal{S}, n * P))$,
$\mathcal{S}$ is the total SCQ memory usage,
n is the total number of operators,
$\mathcal{P}$ is the prediction step length.

Summary of Work

The primary contribution of this work is providing predictive runtime verificaiton based on system model:

- extension to an existing state-of-the-art RV tool, R2U2;
- better mitigation of faults by enabling future-time requirements to be evaluated;
- hardware realiable by bounding resource usage;

The End

References I

- Ebru Aydin Gol, Mircea Lazar, and Calin Belta, *Temporal logic model predictive control*, Automatica **56** (2015), 78–85.
- Hong Lu and Alessandro Forin, *The design and implementation of p2v, an architecture for zero-overhead online verification of software programs*, Tech. Report MSR-TR-2007-99, Microsoft Research, August 2007.
- Gary Nutt, *Tutorial: Computer system monitors*, Computer **8** (1975), no. 11, 51–61.
- R. Pellizzoni, P. Meredith, M. Caccamo, and G. Rosu, *Hardware runtime monitoring for dependable cots-based real-time embedded systems*, 2008 Real-Time Systems Symposium, Nov 2008, pp. 481–491.

References II

- Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia, *Model predictive control with signal temporal logic specifications*, Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on, IEEE, 2014, pp. 81–87.
- Thomas Reinbacher, Kristin Yvonne Rozier, and Johann Schumann, *Temporal-logic based runtime observer pairs for system health management of real-time systems*, International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2014, pp. 357–372.

Hardware Monitor for Temporal Logic

Related Hardware Monitor:

- 1975 as Nutt [Nut75] proposed using hardware to monitor computer systems.
- An FPGA-based hardware monitor, called BusMOP [PMCR08].
- Hong created an automated tool, called P2V [LF07].
- R2U2: soft-coded hardware monitor [RRS14].

Predictive Runtime Verification

Interdisciplinary work between **RV** and **control**.

- Model Predictive Control with Signal Temporal Logic Specifications [RDM+14].
- Temporal logic model predictive control [GLB15]