Runtime Monitoring Temporal Logic using streamLAB
CSE 293 Formal Methods

David Kooi

UC Santa Cruz

June 29, 2020

Motivation: High Consequence Missions

- $300 million failure
- Cause: Inadequate testing and specifications
- Calculated thrust: lbf-s
- Expected thrust N-s

 [1]

Figure 1: Mars Climate Orbiter(1998)

[1] https://news.cornell.edu/stories/1998/12/two-nasa-spacecraft-launches-one-mars-and-other-research-birth-stars-involve

Introduction

- Temporal logic is a formal method to describe the behavior requirements of Cyber Physical Systems(CPS).
- Can formalize "Safety"(Always) and "Liveness"(Eventually) requirements
- Dynamical requirements such as rise time, overshoot, settling time

Overview

- Background in Signal Temporal Logic
- Runtime monitoring using streamLAB and illustration of (in)correct monitors
- Perspectives on adaptive sample rates

STL Background

- Signal temporal logic(STL) was introduced in 2004 by [Maler and Nickovic, 2004]
  - An extension of MITL(Metric Interval Temporal Logic)[Alur,] for real valued signals
- Features quantatative semantics: A measure of how well a trace satisfies it's specification
- Origins from Metric Temporal Logic developed in 1990. [Koymans, 1990]

Past-Time STL

$$\varphi := p_i \in AP \mid \neg\varphi \mid \varphi_i \wedge \varphi_j \mid \varphi_i \mid \blacklozenge\varphi \mid \blacksquare\varphi$$

STL Semantics

(Past-Time) Boolean Satisfaction Semantics

$$
\begin{aligned}
(s,k) &\models p_i \in AP && \iff f_i(s_k) > 0 \\
(s,k) &\models \neg\varphi_i && \iff (s,k) \not\models \varphi_i \\
(s,k) &\models \varphi_i \wedge \varphi_j && \iff (s,k) \models \varphi_i) \wedge (s,k) \models \varphi_j) \\
(s,k) &\models \blacklozenge\varphi_i && \iff \exists t' \in t_k - \mathcal{I} \text{ s.t } x(t') \models \varphi \\
(s,k) &\models \blacksquare\varphi && \iff \forall t' \in t_k - \mathcal{I}, s(t') \models \varphi
\end{aligned}
$$

(Past-Time) Quantitative Semantics

$$
\begin{aligned}
\rho(\top, s, k) &= +\infty \\
\rho(\varphi_i \in, s, k) &= f_i(s_k) \\
\rho(\neg\varphi_i, s, k) &= -\rho(\varphi_i, s_k) \\
\rho(\varphi_i \wedge \varphi_j, s, k) &= \min(\rho(\varphi_i, s_k), \rho(\varphi_j, s_k)) \\
\rho(\blacklozenge\varphi_i, s, k) &= \max_{\tau_{k'} \in \tau_k -} \rho(\varphi_i, s_{k'}) \\
\rho(\blacksquare\varphi_i, s, k) &= \min_{\tau_{k'} \in \tau_k -} \rho(\varphi_i, s_{\tau'})
\end{aligned}
$$

Example: Satellite Regulation

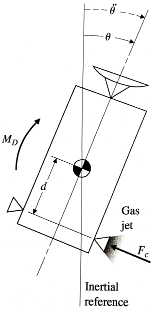

Figure 2: Satellite Model[Franklin et al., 1994]

$$\dot{\vec{\theta}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \vec{\theta} + \begin{bmatrix} 0 \\ \frac{d}{I} \end{bmatrix} u \tag{1}$$

Where d is the length between center of mass and the thruster, $I = \frac{M_D l^2}{12}$ is

Time Domain Specifications

- t_r: time for a state to reach 90% of the set-point
- M_p: over-shoot (maximum peak)
- t_s: settling time

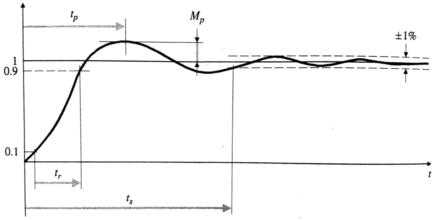

Figure 3: Time Domain Specifications[Franklin et al., 1994]

Formal Requirements

Step-response requirements: $\phi = \Box\left(\varphi_{RT} \land \varphi_{OS}\right)$

$$\varphi_{RT} = \Diamond_{[0,1.5s]} \; |\theta - SP| < 0.1 * SP \tag{2}$$

$$\varphi_{OS} = \Box \; \frac{M_p}{SP} < 25\% \tag{3}$$

Note: These requirements are written in future-time logic.

The problem of Causality

- Future logic is intuitive, but not causal

Future Time	$(s,k) \models \Box_{[a,b]}\psi \iff \forall t' \in [t_k + a, t_k + b], x(t') \models \psi$
Past Time	$(s,k) \models \blacksquare_{[a,b]}\psi \iff \forall t' \in [t_k - a, t_k - b], x(t') \models \psi$

Table 1: Example of (non)causality

How to handle?

- Restrict the temporal logic to it's past time fragment.[Ulus, 2019]
- Delay evaluation until a future time interval has passed
- Return indeterminate outputs until a determination can be made[Deshmukh et al., 2015][Reinbacher et al., 2014]

Translation to Past Time Logic

Top Level Requirement

$$\phi = \Box\,(\varphi_{RT} \wedge \varphi_{OS})$$

to

$$\phi = \blacksquare\,(\varphi_{RT} \wedge \varphi_{OS})$$

Rise Time Requirement

$$\varphi_{RT} = \Diamond_{[0,r_t]}\ |\theta - SP| < 0.1$$

to

$$\varphi_{RT} = \blacklozenge_{[0,r_t]}(t == 0 \ \wedge \ |\theta - SP| < 0.1)$$

Overshoot Requirement

$$\varphi_{OS} = \Box\ \frac{M_p}{SP} < 25\%$$

to

$$\varphi_{OS} = \blacksquare\ \frac{M_p}{SP} < 25\%$$

Monitor Implementation

- Runtime monitoring using streamLAB[Faymonville et al., 2019a]: real-time monitoring engine
- Uses rtLOLA[Faymonville et al., 2019b] as temporal specifications
- Takes CSV formated input in real-time

```
input x: Float64
output x_max @10Hz := x.aggregate(over:10s, using:max)
trigger x_max > 10
```

Figure 4: Example rtLOLA Specification

Quantifying the Requirements

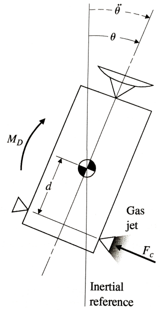

Figure 5: Satellite Model[Franklin et al., 1994]

- System states: $t, u, \theta, \dot{\theta}$.

```
input time:     Float64
input u:        Float64
input theta:    Float64
input d_theta:  Float64
```

Figure 6: Monitor Inputs

Quantifying The Requirements

Top Level Requirements

$$\phi = \blacksquare\ (\varphi_{RT} \wedge \varphi_{OS})$$
$$\rho(\phi) = \min(\rho(\varphi_R), \rho(\varphi_{OT}))$$

Quantifying the Requirements

Overshoot Requirements

$$\varphi_{OS} = \blacksquare \; \frac{M_p}{SP} < 25\%$$
$$= \blacksquare \; \varphi_3$$

$$\rho(\varphi_{OS}) = \min(\rho(\varphi_3))$$
$$\rho(\varphi_3) = 0.25 - \frac{M_p}{SP}$$

Rise-Time Requirements

$$
\begin{aligned}
\varphi_{RT} &= \blacklozenge_{[0,r_t]}(t == 0 \quad \wedge \quad |\theta - SP| < 0.1) \\
&= \blacklozenge_{[0,r_t]}(\varphi_1 \quad \wedge \quad \varphi_2) \\
&= \blacklozenge_{[0,r_t]}\varphi_0
\end{aligned}
$$

$$\rho(\varphi_{RT}) = \max_{t'_k \in t_k - [0,r_t]} \rho(\varphi_0)$$

$$\rho(\varphi_0) = \min(\rho(\varphi_1), \rho(\varphi_2))$$

$$\rho(\varphi_1) = \begin{cases} \infty & \text{if } t == 0 \\ -\infty & \text{if } t \neq 0 \end{cases}$$

$$\rho(\varphi_2) = 0.1 - |\theta - SP|$$

```
/* Phi Rise Time */
output rho_0 := min(rho_1, rho_2) // 3
output phi_0 := rho_0 > 0.0 // 4

output rho_1: Float64 := if time < 1500.0 then 999.0 else -999.0 // 5
output phi_1 := rho_1 > 0.0 // 6

output rho_2 := 0.1 - abs_err // 7
output phi_2 := rho_2 > 0.0   // 8
```

Figure 7: Risetime Requirements in rtLOLA

Incorrect Monitor: Sample Rate 2Hz

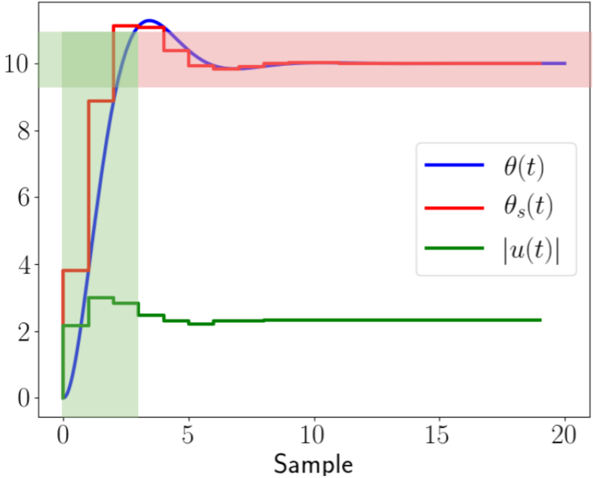

Figure 8: Monitoring at 2Hz

Incorrect Monitor: Sample Rate 2Hz

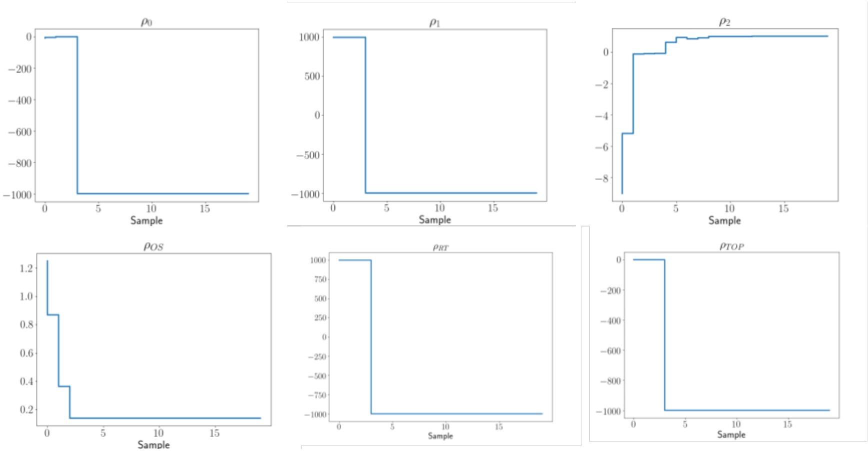

Figure 9: STL Robustness Metrics at 2Hz

Correct Monitor: Sample Rate 4Hz

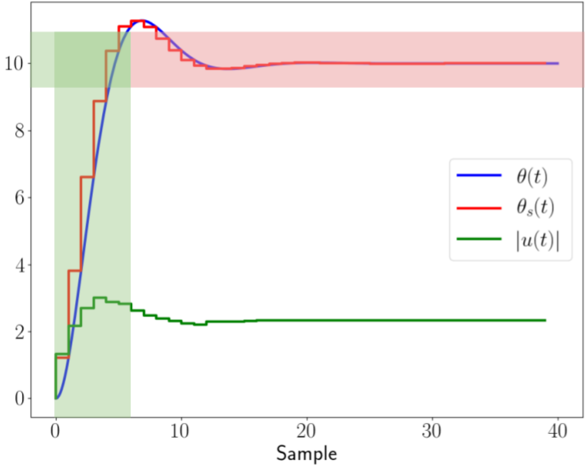

Figure 10: Monitoring at 4Hz

Correct Monitor: Sample Rate 4Hz

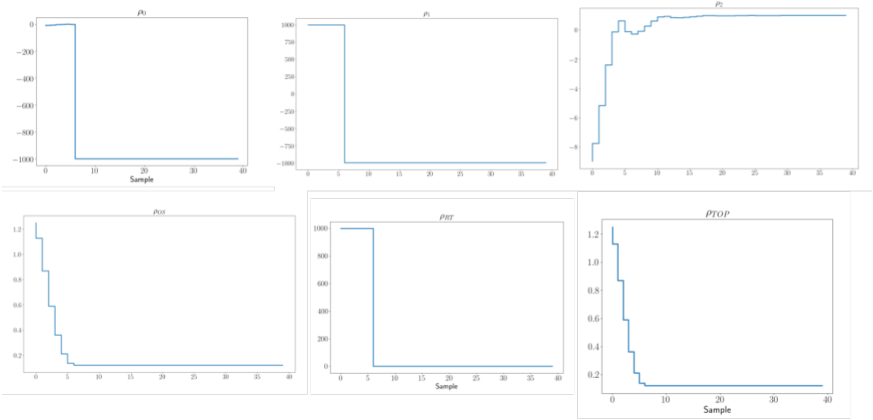

Figure 11: STL Robustness Metrics at 4Hz

Sample Rate Selection

- Badly chosen sample rate can cause incorrect monitor conclusions
- How to choose sample rate?
  - 40 times system bandwidth [Franklin et al., 1994]

However, monitors can be computationally and memory intensive. Can a monitor have an adaptive sample rate that

1. Ensures the monitor is correct
2. Minimizes the number of executions

Adaptive Sample Rates

Using Bounded Velocity Assumptions[Fainekos and Pappas, 2009]
The sample period T_k at state s_k can be chosen adaptively as

$$T_k < \frac{\rho(\phi, s, k)}{V} \tag{4}$$

Using the direction of dynamics (Current research)

$$T_k \leq \frac{\rho(\phi, s, k)}{M_s(s_k)} \tag{5}$$

with

$$M_s(s_k) = \sup_{s' \in R(\bar{T}, s_k)} \langle \nabla \rho(x'), f(x') \rangle \tag{6}$$

Conclusion and Future Work

Conclusion

- Used streamLAB for run-time monitoring of a dynamical simulation
- Manual translation of formal specifications into rtLOLA
- Illustration how sample rate affects monitor correctness

Future Work

- Comparison between streamLAB and Lustre based monitor
- How to ease the translation from formal specifications into software?
- Continuing adaptive sample rate work; developing examples

References

Alur, R.
The Benefits of Relaxing Punctuality.

Deshmukh, J. V., Donzé, A., Ghosh, S. a., Jin, X., Juniwal, G., and Seshia, S. A. (2015).
Robust Online Monitoring of Signal Temporal Logic.
arXiv:1506.08234 [cs].

Fainekos, G. E. and Pappas, G. J. (2009).
Robustness of temporal logic specifications for continuous-time signals.
Theoretical Computer Science, 410(42).

Faymonville, P., Finkbeiner, B., Schledjewski, M. t., Schwenger, M., Stenger, M., Tentrup, L., and Torfah, H. (2019a).
StreamLAB: Stream-based Monitoring of Cyber-Physical Systems.
In Dillig, I. and Tasiran, S., editors, *Computer Aided Verification*, volume 11561, pages 421–431. Springer International Publishing, Cham.