EPiC Series in Computer Science

Volume 34, 2015, Pages 25–30

ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems

Benchmarks for Temporal Logic Requirements for Automotive Systems

Bardh Hoxha, Houssam Abbas, and Georgios Fainekos

Arizona State University,
Tempe, AZ, USA
{bhoxha, hyabbas, fainekos}@asu.edu

Abstract

We propose to standardize two Matlab/Simulink models of automotive systems as benchmark problems for hybrid system verification. Both models can be simulated quickly, making them ideal for testing-based verification methods that require a significant number of system output trajectories. One of the benchmarks is the Automatic Transmission model, which is deterministic. The other benchmark is the Fault-Tolerant Fuel Control System, which exhibits stochastic behavior. Our benchmark standardization defines a number of Metric Temporal Logic requirements for the models.

Category: academic **Difficulty:** medium

1 Context and Origins

We propose to standardize as benchmarks existing models of hybrid systems that are widely available and documented by Mathworks and at the same time exhibit all the complexities of industrial strength models.

Automatic Transmission We propose a slightly modified version of the Automatic Transmission model provided by Mathworks as a Simulink demo [5]. It is a model of an automatic transmission controller that exhibits both continuous and discrete behavior. The model is deterministic: that is, it does not contain components with stochastic behavior. Our motivation for proposing this model as a benchmark problem is founded on the fact that this model has already been used by multiple research groups.

To the best of our knowledge, this benchmark was first considered in [8] to illustrate a genetic algorithm approach to test input generation for hybrid systems. In [7], the authors used the model for estimating the range of the parameters of Metric Temporal Logic (MTL) specifications such that the system does not satisfy the specification. In [2], the authors use the model to perform MTL falsification, i.e. to find a trajectory that does not satisfy the specification (also known as a *counter example*). In [4], the authors utilize the model to illustrate a method for mining requirements from closed-loop models.

Fault-Tolerant Fuel Control System Fault-Tolerant Fuel Control System is a modified version of the model provided by Mathworks as a Simulink demo [6]. The model detects system failures and as a result modifies its control law to sustain system performance. The arrival of

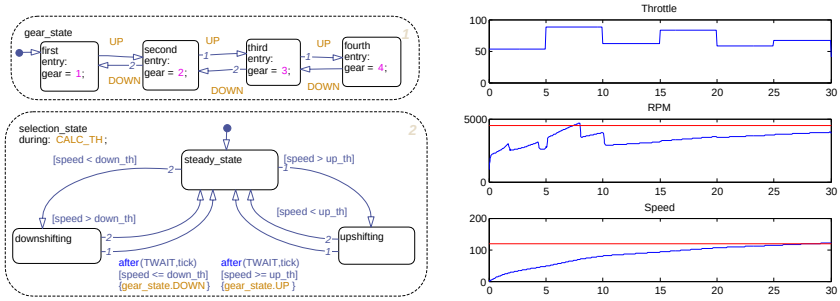

Figure 1: *Left:* The switching logic for the automatic drivetrain; *Right:* An input signal (top) and the corresponding output signals that falsify the specification.

faults is modeled by Poisson stochastic processes with different arrival rates. This benchmark was first considered in [9], where the authors use Bayesian statistical model checking techniques to, among others, estimate the probability of satisfying the specification, and to estimate a corresponding confidence interval.

2 Brief description

Automatic Transmission There are two inputs to the system: the throttle and break. The break input enables the user to model variable load to the engine, e.g., going uphill or downhill. The physical system has two continuous-time state variables which are also its outputs: the speed of the engine ω (RPM) and the speed of the vehicle v (mph). Initially, the vehicle is at rest at time 0, i.e. the speed $v = 0$ and engine speed $\omega = 0$. Therefore, the output trajectories depend only on the input signals u_t and u_b which model the throttle and break inputs. The throttle and break, at each point in time, can take any value between 0 (fully closed) to 100 (fully open). The range for the break depends on the engine load that we would like to model. The system is deterministic, i.e., under the same input u, it will produce the same output y.

The model contains 69 blocks among which there are 2 integrators (i.e., 2 continuous state variables), 3 look-up tables, 3 2D look-up tables and a Stateflow chart. The Stateflow chart (see Fig. 1 for a schematic) contains two concurrently executing Finite State Machines with 4 and 3 states, respectively.

Table 1 presents a number of requirements that should be verified on the automatic transmission model. As an example, consider formula ϕ_2^{AT} in Table 1: this is a simple invariant. The goal of the verification is either to prove the invariant or produce counter examples that demonstrate that the invariant is not true. The verification of the model is challenging for the following reasons. First, the engine and the vehicle components contain nonlinear equations and lookup tables. The latter increases the size of the hybrid state space substantially. Second, the switching conditions of the Stateflow chart depend on both state variables and input signals and are also time dependent. Both reasons make the problem challenging for state of the art reachability analysis tools [3] .

For the invariant in ϕ_2^{AT}, we would like to generate trajectories such that the vehicle speed v and the engine speed ω exceed the values 120 mph and 4500 RPM, respectively. Such a falsifying system trajectory appears in Fig. 1.

Fault-Tolerant Fuel Control System This system models the fuel controller for a gasoline engine. Its goal is to keep the air-to-fuel ratio close to the "ideal" stoichiometric ratio so that

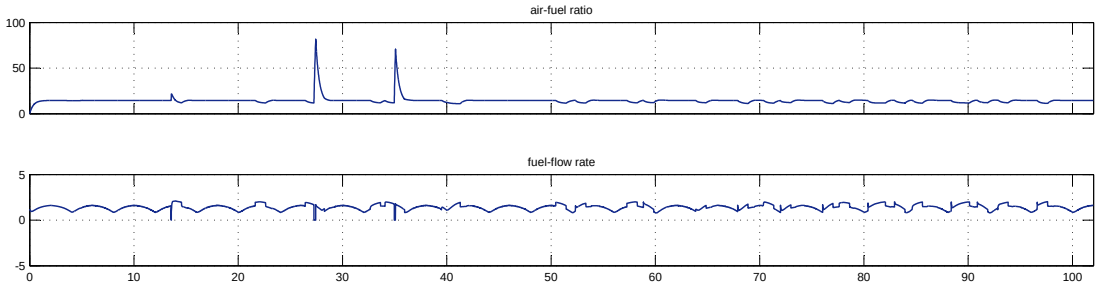

Figure 2: The output trajectories for the Fault-Tolerant Fuel Control System under constant input. Top: air-to-fuel ratio. Bottom: fuel-flow rate.

both the oxygen and the fuel are consumed completely in the process. The outputs of the system are the fuel rate and the air-to-fuel ratio - see fig. 2 for an example output of the system. For correct operation, the system requires sensor information. There is one sensor that provides readings on the amount of residual oxygen present in the exhaust gas, one for the engine speed, one for the throttle, and one for the manifold absolute pressure. The system is designed to detect sensor failures, and the control system changes dynamically to ensure uninterrupted operation. If a single sensor fails, the system compensates. If more than one fail then the system is shut down. The system exhibits discrete and continuous behavior that is described by nonlinear and linear differential equations with a switching condition.

We have extended the modifications to the system implemented in [9] by adding three Poisson processes to model sensor failures with different arrival rates which are inversely correlated with the throttle input signal: the larger the input, the smaller the rate of the Poisson process i.e. more faults on the system. With the modifications made, this becomes an example of a Stochastic Cyber Physical System.

Table 1 lists example specifications that should be satisfied by this system.

3 Formal specifications

Table 1 lists the proposed formal specifications for both systems. It contains both simple properties (e.g. safety property ϕ_1^{AT}) and more complex ones (e.g. ϕ_8^{AT}). It also contains formal requirements with and without real-time constraints. The former is challenging for reachability analysis tools that ignore timing when approximating the states that cross a switching guard.

For the Automatic Transmission model, by varying the thresholds $\bar{\omega}$ and $\bar{v}$, the benchmark problems can vary from proving invariants to falsification problems. To date, the authors are not aware of an algorithm that can *verify* these properties for these systems. Thus they serve as good benchmarks for driving the development of formal verification in that direction. Moreover, some properties have not yet been falsified with testing-based methods. If used as benchmarks for testing-based falsification, the goal would be to compare different methodologies on their speed of detecting counterexamples. All else being equal, properties that involve the discrete gear sequence (like ϕ_{2-5}^{AT}) are generally more challenging than ones that don't use the gear: intuitively, that is because the current continuous state affects the switching guards, and these guards determine the gear sequence. Thus the gear sequence is a *delayed* indicator of the variations in the continuous state. Having to consider the real-valued continuous state and the evolving guards (whose evolution can't be pre-computed or analytically described in terms of the search variables) is problematic for formal methods, and challenging to testing methods

Table 1: Various specifications expressed in natural language and MTL.

Property	Natural Language	MTL
Automatic Transmission		
ϕ_1^{AT}	The engine speed never reaches $\bar{\omega}$.	$\Box(\omega < \bar{\omega})$
ϕ_2^{AT}	The engine and the vehicle speed never reach $\bar{\omega}$ and $\bar{v}$, resp.	$\Box((\omega < \bar{\omega}) \wedge (v < \bar{v}))$
ϕ_3^{AT}	There should be no transition from gear two to gear one and back to gear two in less than 2.5 sec.	$\Box((g_2 \wedge Xg_1) \rightarrow \Box_{(0,2.5]} \neg g_2)$
ϕ_4^{AT}	After shifting into gear one, there should be no shift from gear one to any other gear within 2.5 sec.	$\Box((\neg g_1 \wedge Xg_1) \rightarrow \Box_{(0,2.5]} g_1)$
ϕ_5^{AT}	When shifting into any gear, there should be no shift from that gear to any other gear within 2.5sec.	$\wedge_{i=1}^{4} \Box((\neg g_i \wedge Xg_i) \rightarrow \Box_{(0,2.5]} g_i)$
ϕ_6^{AT}	If engine speed is always less than $\bar{\omega}$, then vehicle speed can not exceed $\bar{v}$ in less than T sec.	$\neg(\Diamond_{[0,T]}(v > \bar{v}) \wedge \Box(\omega < \bar{\omega}))$
ϕ_7^{AT}	Within T sec the vehicle speed is above $\bar{v}$ and from that point on the engine speed is always less than $\bar{\omega}$.	$\Diamond_{[0,T]}((v \geq \bar{v}) \wedge \Box(\omega < \bar{\omega}))$
ϕ_8^{AT}	A gear increase from first to fourth in under 10secs, ending in an RPM above $\bar{\omega}$ within 2 seconds of that, should result in a vehicle speed above $\bar{v}$.	$((g_1 \ \mathcal{U} \ g_2 \ \mathcal{U} \ g_3 \ \mathcal{U} \ g_4) \wedge \Diamond_{[0,10]}(g_4 \wedge \Diamond_{[0,2]}(\omega \geq \bar{\omega}))) \rightarrow \Diamond_{[0,10]}(g_4 \rightarrow X(g_4 \ \mathcal{U}_{[0,1]} \ (v \geq \bar{v})))$
Fault-Tolerant Fuel Control System		
ϕ_1^{FCS}	The fuel flow rate should not be 0 for more than 1 sec within the next 100 sec period.	$\neg\Diamond_{[0,100]}\Box_{[0,1]}(FuelFlowRate = 0)$
ϕ_2^{FCS}	Always, if the air-to-fuel ratio output goes out of bounds, then within 1 sec it should settle inside the bounds and stay there for a sec.	$\Box((\lambda \text{ out of bounds}) \rightarrow \Diamond_{[0,1]}\Box_{[0,1]}\neg(\lambda \text{ out of bounds}))$

ω: Engine rotation speed, v: vehicle velocity, g_i : gear i, λ : air-to-fuel ratio.
Recommended values: $\bar{\omega}$: 4500, 5000, 5200, 5500 RPM; $\bar{v}$: 120, 160, 170, 200 mph; T : 4, 8, 10, 20 sec; λ bounds: 0.9 - 1.1.
$\Box$: Always, $\diamond$: Eventually, $\mathcal{U}$: Until

that can only look at the next guard to cross in deciding the next input to try.

4 Outlook

There are several possibilities for future development. For the Automatic Transmission model, noise can be introduced to the system through the sensors and actuators. We can also modify the model to support semiautomatic gear shift instead of fully automatic. This would add the gear setting as part of the input search space. Another possibility is to introduce a hybrid drivetrain. In either case, the temporal logic requirements will become more complicated. For the Fault-Tolerant Fuel Control model, the arrival of faults could be optionally modeled with other stochastic processes. Also, the engine nominal speed and throttle command could be added to the input search space. The current properties for this system are on the output behavior: they describe the permissible changes in the fuel flow rate and the fuel-to-air ratio. It will be interesting and more challenging to examine 'white-box' properties that describe permissible tolerances during fault recovery: i.e., in case of one sensor failure, what transients are allowed? In case of two sensor failures, what is a graceful degradation to engine shutdown?

Acknowledgments. This work was partially funded under NSF awards CNS 1116136, CNS 1319560. We would also like to thank Adel Dokhanchi for his help with the robustness computations.

A Appendix

Both benchmarks are available through our Matlab Toolbox S-TaLiRo [1], available at `https://sites.google.com/a/asu.edu/s-taliro/s-taliro` under the `bechmarks/ARCH2014` subfolder. They can be simulated out-of-the box, and demo programs are provided to illustrate how to define properties for them and how to falsify them.

The following is the list of modifications made to both models: for the automatic transmission, we added inputs for the throttle and brake schedule, and outputs for the vehicle and engine speed and the transmission gear. For the fault-tolerant fuel control system, we added an input for the throttle angle, and three separate Poisson processes to model the arrival of faults, with arrival rates inversely proportional to the throttle angle. We also added outputs for the fuel-flow rate and the air-to-fuel ratio.

References

[1] Yashwanth Singh Rahul Annapureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.

[2] Georgios Fainekos, Sriram Sankaranarayanan, Koichi Ueda, and Hakan Yazarel. Verification of automotive control applications using s-taliro. In *Proceedings of the American Control Conference*, 2012.

[3] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In *Proceedings of the 23d CAV*, 2011.

[4] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy V Deshmukh, and Sanjit A Seshia. Mining requirements from closed-loop control models. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 43–52. ACM, 2013.

[5] Mathworks. http://www.mathworks.com/videos/modeling-an-automatic-transmission-and-controller-68823.html.

[6] Mathworks. http://www.mathworks.com/products/demos/stateflow/fuelsys.html.

[7] Hengyi Yang, Bardh Hoxha, and Georgios Fainekos. Querying parametric temporal logic properties on embedded systems. In *Testing Software and Systems*, pages 136–151. Springer, 2012.

[8] Qianchuan Zhao, Bruce H. Krogh, and Paul Hubbard. Generating test inputs for embedded control systems. *IEEE Control Systems Magazine*, August:49–57, 2003.

[9] Paolo Zuliani, André Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *13th ACM International Conference on Hybrid Systems: Computation and Control*, pages 243–252, 2010.