Falsification of Dynamical Systems – An Industrial Perspective

Thomas Heinz([✉])

Robert-Bosch GmbH, Corporate Research,
Robert-Bosch-Campus 1, 71272 Renningen, Germany
thomas.heinz@de.bosch.com

Abstract. Whenever formal verification of dynamical system models is not applicable, e.g., due to the presence of black-box components, simulation-based verification and falsification methods are promising approaches to gain confidence in a system satisfying its specification. With the introduction of robust semantics it is not only possible to answer this question in the Boolean sense but to quantify its truth. We illustrate a number of applications that are interesting from an industrial perspective, and point out how robustness could become even more versatile in the engineering process.

Keywords: Falsification · Conformance · Testing · Robust semantics · Simulation-based verification · Automotive control systems

1 Introduction

Models of dynamical systems play a crucial role in the development of automotive control systems. Here, we focus on models describing the closed-loop interaction between physical processes and controllers. Such models exist at various levels of abstraction, e.g., in order to design a controller, a simplified physical model is used whereas for validation purposes the controller is tested against a detailed physical model. Controller models on the other hand range from abstract continuous-time models to fixed-step implementation models involving precise digital hardware behavior. There exist a variety of different modeling tools specialized in different physics domains based on different formalisms such as PDEs (partial differential equations), ODEs (ordinary differential equations), DAEs (differential algebraic equations), and hybrid or switched versions thereof. The situation is similar for controller models. Modeling tools such as Matlab/Simulink or Modelica can express both physics and high-level continuous- or discrete-time controllers in a single model. During refinement of the controller implementation however, different tools must be used that reflect real-time scheduling [9], and hardware behavior as well. Co-simulation is required

T. Heinz Invited talk. Thanks to my colleagues Jens Oehlerking, Matthias Woehrle, and Christoph Gladisch for valuable discussions and feedback.

to perform closed-loop simulations of such implementation models and physical processes, i.e., the entire system model is represented by different modeling tools and the simulation becomes a distributed process which is typically coordinated by a single tool. Alternatively, tool-specific models are converted into tool-independent models such as FMUs (Functional Mock-up Units) [3]. Given the current state of the art, it is not possible to formally verify functional properties of such models. This is partially due to the lack of formal semantics of some modeling tools and the presence of black-box components in form of libraries. For physical processes which do not admit an appropriate characterization by closed-form ODE/DAE models, such as combustion, the model may involve functions described by large tables of data. In these cases, even if we describe the model in a mathematically unambiguous way, e.g., as a hybrid automaton, formal verification such as reachability analysis is often not practically possible with state of the art methods.

A promising approach to address these complexities is simulation-based verification resp. falsification built on top of robust semantics for temporal property specification logics [7,8]. So far research has mostly focused on variants of linear temporal logic (LTL), in particular metric temporal logic (MTL) and signal temporal logic (STL). The basic idea is to generalize the Boolean semantics of a temporal logic by a metric. In the Boolean semantics, a signal (trace) either satisfies a specification or not. In a robust semantics, a signal is mapped to a real number indicating some measure of distance to the satisfaction "border". A value in $\mathbb{R}_0^+$ indicates that the signal satisfies the specification, a value in $\mathbb{R}^-$ shows that it does not. Such a quantification of truth enables the use of optimization methods to guide the exploration of a model and to find initial conditions, parameters, and possible input signals falsifying the property. If the model indeed satisfies the property – which is in general undecidable – the exploration tries to minimize the degree of satisfaction. Under certain assumptions, e.g., the model being Lipschitz continuous, simulation-based verification is possible, i.e., a finite number of simulation runs may suffice to show a temporal property [6]. In the following, these approaches are summarized by the term *property conformance checking*.

During model refinement from abstract to implementation models, it is important to preserve desired properties. Property conformance checking is a way to increase confidence in this preservation. Besides satisfying particular properties, it is often desirable that the behavior of a refined system is close to that of the original system, e.g., when the physics model is replaced by a more complex one or when disturbances are introduced. Another example is comparing a continuous-time model with a discrete-time version with delays characterizing timing properties of the distributed execution of the controller such as computation, communication, and scheduling delays. It has been observed that equivalence notions from discrete systems such as bisimilarity are too strong to adequately capture similarity of closed-loop control models. Instead, quantifying the distance of pairs of corresponding signals from both models provides a useful indication of model similarity. Deshmukh et al. [5] introduce an effective method to compute the distance of signals under the Skorokhod metric which considers retimed versions $s(r(t))$ of a signal $s(t)$ with r being a monotonically increasing retiming function. The idea is to capture both distortions in space and time where time distortions can be more general than constant time shifts. Subsequently, this approach is called *model conformance checking*.

2 Industrially Relevant Applications

Automatic testing is important in industrial practice to have high confidence that the product meets its requirements when formal methods are not applicable. Tests capturing property or model conformance on the level of "virtual" models are already useful as indicated above. Testing the real system is inevitable even if the entire chain of model refinements would be formally verified. This is expensive compared to simulations and thus benefits from careful selection of relevant test cases, i.e., those which either falsify the property, or operate the system at the satisfiability limit. In the context of vehicle dynamics, such test cases can be executed from driving robots which control steering, acceleration, and braking [11]. In the realm of automated driving, a test case may be as complex as finding a particular challenging road configuration with potential obstacles, and defining the behavior of dynamic objects other than the automated vehicle itself.

Clearly, it is desirable to have models which admit **formal verification in practice**. A formal model could be used to describe the behavior for a subset of outputs, e.g., a Büchi automaton capturing discrete behavior. Alternatively, it may be a simplified dynamics model such as a system of ODEs or a linear hybrid automaton, which presumably overapproximates the behavior in a subset of the original state space, hence enabling reachability analysis [2,10] and theorem proving [12]. Model conformance checking can provide best effort indication that a formal model is indeed an abstraction of the original system.

Property conformance checking as described above computes a single robustness value for a particular signal, and thus quantifies the satisfaction of the property by that signal. **Online monitoring** is a form of property conformance checking where a robustness signal is computed, i.e., a function mapping each signal prefix to its robustness value [4]. A robustness signal provides valuable feedback for developers in understanding how the satisfaction of a property evolves in time. Moreover, if the monitor can be computed in real-time, its output could be used to modify the system behavior, e.g., in case a system is approaching its satisfiability limit, it could activate some fallback behavior before the property is actually violated.

A black-box model contains inputs, outputs, parameters and states. In practice, it is often not possible to know all states as they might be hidden in libraries that are part of the model. For autonomous models, i.e., ones without input, property or model conformance checking explores a given parameter space. If inputs are present, the exploration problem becomes more involved. A simple approach is to select inputs from a predefined family of input functions. The disadvantage is that the relation of input to state space is unknown. Consider the simple property $\Box(x \geq 2 \Rightarrow \Diamond_{[0,1]} y \leq 3)$ where x is a state, and y an output. If the goal is to select an interesting input, it would have to be such that $x \geq 2$ holds eventually. Selecting inputs from a predefined set works only if the set is carefully chosen with knowledge about the model that must be obtained by other means. Another approach would be to simply set $x_0 \geq 2$ initially but this is not possible when the test is performed on a real system as we can obviously not simply set real physical states in this way. Hence, for this practically relevant class of specifications, test generation involves **control synthesis**. There are numerous publications about this subject such as [13] which constructs a receding horizon controller from an STL specification. However, it is not clear how to deal with hidden states. Moreover, constraints on inputs must be respected to avoid damaging the real system under test.

Scenario-based verification is an important first step in mastering the safety challenge for automated driving. Instead of testing many individual scenarios, formal specification languages may provide an effective way of producing families of non-deterministic driving scenarios and expectations from the automated vehicle therein. STL – while being amenable to efficient online monitoring – might not be convenient or expressive enough because it is not possible to refer to previous signal values at a given point in time. Introducing the so-called freeze operator [4] solves this problem but at the cost of online monitoring becoming more expensive.

3 One Robustness Does Not Fit All

Property conformance checking as described in Sect. 1 is key to various industrially relevant applications sketched above. The notion of robustness generalizes the classical Boolean semantics by quantifying the satisfaction of a temporal property. Initially, space robustness [8] was proposed to measure the satisfaction degree of a signal with respect to a formula. The robustness value indicates how much the entire signal can be shifted in space and still satisfy the formula. Most publications dealing with robust semantics of temporal logics are based on space robustness. As a generalization, space-time robustness [7] was introduced where time robustness describes how much a signal may be shifted in time and still satisfy the formula. Besides various semantics, an STL extension was proposed which augments the logic by two operators (averaged-until and averaged-released) [1]. The metric associated with these operators allows to express for example expeditiousness, persistence (as long as possible), and soft deadlines (earlier is better).

In practice, the properties of a model can often be written in the form $\Box \bigwedge_{1 \leq i \leq n} (\phi_i \Rightarrow \psi_i)$ where ϕ_i refers to inputs and states and ψ_i refers to inputs and outputs. A single notion of robustness is not sufficient to represent the property adequately. For a test/simulation to be relevant, it is required that some ϕ_i is satisfied since otherwise the formula holds trivially. While it is possible that the precondition benefits from robust semantics, it may as well be sufficient to

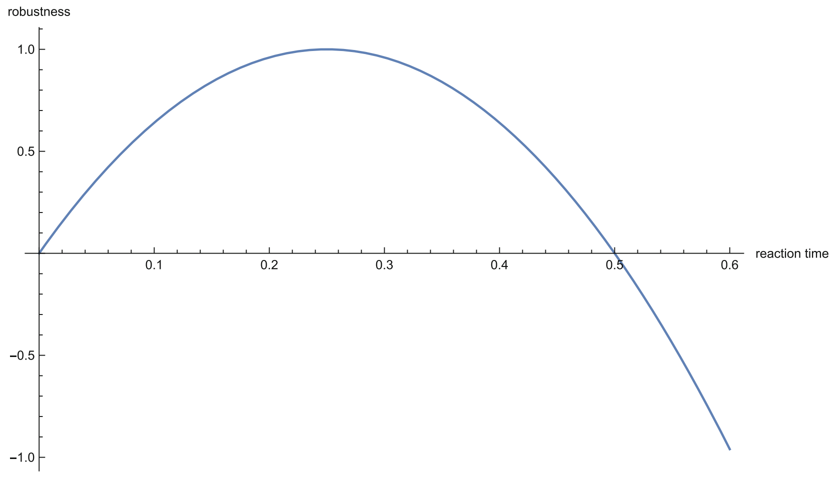

Fig. 1. Robustness for acceleration pedal reaction

evaluate the precondition under the Boolean semantics. For different postconditions though, different robustness metrics might be desirable. Let

$$\begin{aligned}\phi_1 &\equiv \phi_2 \equiv \mathit{input\_step}\\ \psi_1 &\equiv \Box_{[0,10]} x \leq 1.2\\ \psi_2 &\equiv \psi_{2,1} \wedge \psi_{2,2}\\ \psi_{2,1} &\equiv \Box_{[0,13]}\ \mathit{input\_unchanged}\\ \psi_{2,2} &\equiv \Diamond_{[0,3]}\Box_{[0,10]} 0.98 \leq x \leq 1.02\end{aligned}$$

where $\phi_1 \Rightarrow \psi_1$ is an overshoot and $\phi_2 \Rightarrow \psi_2$ is a settling time requirement. It is natural to evaluate ψ_1 under the space robustness semantics (less overshoot is better), $\psi_{2,1}$ under the Boolean semantics (input should not change), and $\psi_{2,2}$ under the time robustness semantics (earlier settling is better). The robustness of the entire requirement can be computed by suitable monitors for the subformulas from which a single robustness value can be calculated. This is possible whenever the formula can be structurally decomposed into parts such that each part can be assigned any of the standard robustness metrics. However, faster/earlier is not always better, neither is slower/later. Consider a simplified requirement for an accelerator pedal.

$$\begin{aligned}\phi &\equiv \mathit{pedal\_pushed}\\ \psi &\equiv \Diamond_{[0,0.5]} a \geq 1\end{aligned}$$

Figure 1 illustrates a robustness definition which favors neither fast nor slow reaction time but defines the optimal reaction time to be at 0.25 s with robustness 1. Recall that a non-negative value indicates satisfaction, and a negative value indicates violation of the property. Deviations from the optimal reaction time are penalized symmetrically, i.e., $0 \leq r(0.25+\Delta) = r(0.25-\Delta) \leq 1$, $\Delta \in [0, 0.25]$ where $r(t)$ denotes the robustness of reaction time t. Reaction times greater than 0.5 s violate the property and its negative robustness grows quadratically with increasing reaction time. Such a robustness is not conveniently expressible as time robustness in the sense mentioned above. It would be possible to approximate this notion arbitrarily by dividing the interval [0, 0.5] into subintervals each associated with time robustness, and then compute the overall robustness from the robustness values associated with each time interval. However, such a specification is neither convenient nor precise.

The notion of robust semantics for temporal logics is very powerful. MTL/STL offer great flexibility in specifying non-trivial properties. A similar degree of freedom for specifying metrics would be of great value. Ideally, a developer would be able to define both the property and a suitable metric. The metric differentiates between good and bad signals among all those which satisfy the specification in the Boolean sense. This allows an intuitive understanding of the robustness signal and enables meaningful computation involving different robustness values. Besides assisting engineers in assessing the quality of their models, metric diversity may be useful in steering the optimization process into different regions of the state space.

4 Conclusion

Property and model conformance checking are versatile approaches to assess correctness of industrial models that cannot be handled by current state of the art formal methods due black-box components and other complexities. By quantifying the degree to which a property is satisfied based on recently introduced robust semantics of linear temporal logics, it is possible to effectively explore models and discover property violating behavior, or behavior which is close to the satisfaction "border". In practice however, one particular robust semantics cannot adequately capture all desired quantifications of truth. Thus, we encourage to develop a generalized robust semantics which enables user-defined metrics, thus leveraging flexible and intuitive composition of multiple properties.

References

1. Akazaki, T., Hasuo, I.: Time robustness in MTL and expressivity in hybrid system falsification. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9207, pp. 356–374. Springer, Cham (2015). doi:10.1007/978-3-319-21668-3_21
2. Althoff, M.: An introduction to CORA 2015. In: Proceedings of the Workshop on Applied Verification for Continuous and Hybrid Systems (2015)
3. Bastian, J., Clauß, C., Wolf, S., Schneider, P.: Master for co-simulation using FMI. In: 8th International Modelica Conference, Dresden. Citeseer (2011)
4. Deshmukh, J.V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., Seshia, S.A.: Robust online monitoring of signal temporal logic. In: Bartocci, E., Majumdar, R. (eds.) RV 2015. LNCS, vol. 9333, pp. 55–70. Springer, Cham (2015). doi:10.1007/978-3-319-23820-3_4

5. Deshmukh, J.V., Majumdar, R., Prabhu, V.S.: Quantifying conformance using the Skorokhod metric. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9207, pp. 234–250. Springer, Cham (2015). doi:10.1007/978-3-319-21668-3_14
6. Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 174–189. Springer, Heidelberg (2007). doi:10.1007/978-3-540-71493-4_16
7. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 92–106. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15297-9_9
8. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theoret. Comput. Sci. **410**(42), 4262–4291 (2009)
9. Frehse, G., Hamann, A., Quinton, S., Woehrle, M.: Formal analysis of timing effects on closed-loop properties of control software. In: 2014 IEEE on Real-Time Systems Symposium (RTSS), pp. 53–62. IEEE (2014)
10. Immler, F.: Verified reachability analysis of continuous systems. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 37–51. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46681-0_3
11. Mikesell, D.R.: Portable automated driver for universal road vehicle dynamics testing. Ph.D. thesis, The Ohio State University (2008)
12. Platzer, A.: Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics. Springer, Heidelberg (2010)
13. Raman, V., Donzé, A., Maasoumy, M., Murray, R.M., Sangiovanni-Vincentelli, A., Seshia, S.A.: Model predictive control with signal temporal logic specifications. In: 2014 IEEE 53rd Annual Conference on Decision and Control (CDC), pp. 81–87. IEEE (2014)