Industrial Examples of Formal Specifications for Test Case Generation

Hendrik Roehm[1], Rainer Gmehlich[1], Thomas Heinz[1],
Jens Oehlerking[1], Matthias Woehrle[1]
Robert Bosch GmbH, Corporate Research, Renningen, Germany

April 8, 2015

Abstract

While requirements engineering has received considerable attention in academia over the past years, formalization of requirements for physically influenced systems is still a difficult task in practice. In this paper, we give formal representations of some typical requirement classes arising in the automotive industry. We divide these patterns into three main classes: those mostly referring to properties of continuous signals, those mostly referring to discrete events and those referring to similarity to a reference signal. We discuss these patterns on concrete examples from automotive embedded systems, where specifications are used for test case generation.
Category: industrial **Difficulty:** medium

1 Context and Origins

Deriving formal specifications for industrial embedded systems is a challenging task. In this paper, we discuss some typical patterns of such specifications as well as challenges when trying to represent the requirements in existing formalisms. One major use case for the derived formal specification is test case generation, i. e., systematically deriving test cases from specifications. In this paper we discuss typical specification patterns occuring in the automotive domain, for test case generation approaches see [3] and [6], for instance.

From an industry perspective, it is important to note that formalized specifications for physically-driven systems will usually form an incomplete picture, as there are requirements which are simply not amenable to suitable formalization at this point. This includes requirements on how driving is supposed to "feel" for the customer. Also, some requirements (like noise levels or vibrations inside the car) may be formalizable, but since there are usually no useful physical models for these effects, the specification cannot be leveraged. However, we believe that formal specifications even of subsets of requirements, together with

[1] firstname.lastname@de.bosch.com

adequate physical models are still very useful, since the development process can be accelerated with help of formal methods.

For the specifications given in this paper, we generally use pre- and post-conditions. Here, a precondition A describes the assumptions made on the environment under which the required behavior, a postcondition B, is supposed to hold. The examples raise the question what are suitable formalisms for A and B to cover a large class of requirements. Usually, A will be tightened for testing purposes. While a specification might require B to hold under a large class A of environment behaviors, if the specification is used to generate tests, A might be replaced by a much more restrictive condition. This is due to the fact that some behaviors might be known a priori as safe, or relatively uninteresting to test. One can view this process analogous to theorem proving techniques: a clause is split into two clauses which are then proved separately. In fact, one may have one resulting clause which can be proven true while the other one cannot, requiring testing based methods. As the examples will show, this tightening of assumptions will often considerably complicate formalization. On the other hand, the challenge we face with respect to B is to capture the radically different types of requirements in formalisms that are both expressive enough to cover a large class of requirements and concise enough to be usable in practice.

Generally, the specification should be kept separate from the model of the system. A common issue is that calculations required solely for checking the specification are included in the model itself, in particular when the specification formalism is not expressive enough. For instance, specifications of assumptions on disturbances (which would be part of A) could theoretically be described as part of the system model itself. In this case, the implicitly modeled assumptions cannot be easily leveraged by model-based testing tools and different modifications of the model might be necessary for different requirements, making this set of models difficult to maintain.

2 Specification classes

Specifications based on properties of continuous signals
Automatically generating test cases from continuous-time specifications of controllers increases efficiency of the development process and facilitates compliance to automotive quality and safety industry standards such as Automotive SPICE and ISO26262. Work related to this topic suggests that such requirements are either captured using ad hoc formalisms as in [7] – where 4 particular control requirements are formulated and subject to be falsified by searching for an appropriate step input – or using temporal logic such MTL (metric temporal logic) or STL (signal temporal logic) as in [5], [8]. Recently, quantitative semantics for temporal logics have been introduced to express a metric on the satisfaction of a temporal logic formula. This development led to falsification tools such as S-TaLiRo [3] and Breach [6] which can be used to generate test cases from temporal logic specifications by maximizing a distance metric that ideally selects only interesting/relevant test cases.

Figure 1 shows an abstract block diagram of the system for which the requirement is established. The purpose of the controller is to drive the current

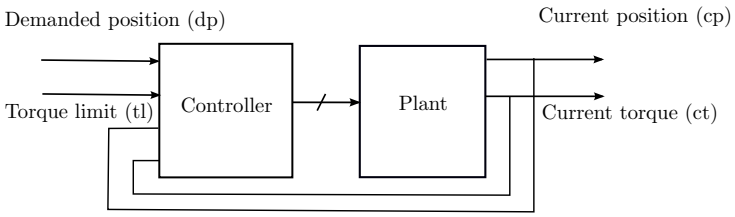


Figure 1: Abstract block diagram of position control system.

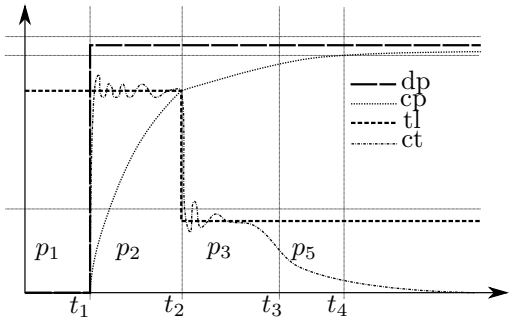


Figure 2: Exemplary simulation run where the requirement is met. Note that $t_3 = t_2 + 10ms$.

position towards the demanded position while respecting the maximum torque limit. The system shall meet the following requirement.

During the position (cp) *regulation after a step input on demand* (dp)*, when the absolute value of the maximum torque limit* (tl) *decreases with a step* (precondition)*, the absolute value of the actuator response in torque* (ct) *must be less than the torque limit plus 10 % in less than 10 ms* (postcondition).

Note that the precondition depends not only on the input (dp, tl) but also on the state (cp) due to the phrase *"during position regulation"*. Figure 2 shows an exemplary simulation run which meets the requirement. After a step of the demanded position at t_1, the current position reaches the demanded position with the required accuracy at t_4. Note that t_4 depends both on the value of the initial torque and the value of the new torque limit at t_2. The requirement states that at $t_3 := t_2 + 10ms$, ct must always be less than tl plus 10 %.

The requirement can be thought of as a special case of a more general requirement that omits the restriction *"during position regulation"*. If the current position has reached the demanded position, the postcondition is trivially satisfied as $ct \approx 0$. In the general requirement, the precondition depends solely on the inputs and not on any state. For the purpose of verification, the general specification without restriction would be selected. However, for the purpose of test case generation, the special requirement which amounts to a tightening of the precondition ensures that only interesting simulation runs are selected.

A very abstract formalization of the specific requirement in LTL is as follows. At any discrete point, each of the 4 signals assume a value from a finite set of

values, i.e. a signal is an infinite word of 4 tuples. The set of values per signal is as follows.

- dp: dpL (low demand position), dpH (high demand position)
- tl: tpL (low torque limit), tpH (high torque limit)
- cp: cpL (below a threshold), cpH (above a threshold), cpF (close to demand position)
- ct: ctL (well below torque limit), ctA (around torque limit), ctH (well above torque limit)

Let the signal alphabet be $\Sigma = \{\text{dpL}, \text{dpH}\} \times \{\text{cpL}, \text{cpH}, \text{cpF}\} \times \{\text{tlL}, \text{tlH}\} \times \{\text{ctL}, \text{ctA}, \text{ctH}\}$. A signal that meets the requirement – henceforth called valid signal – can be divided in 5 phases $p_1, \ldots, p_5$ where

$$p_1 \equiv (\text{dpL}, \text{cpL}, \text{tlH}, \text{ctL}) \qquad p_4 \equiv (\text{dpH}, \text{cpH} \vee \text{cpF}, \text{tlL}, \textit{true})$$
$$p_2 \equiv (\text{dpH}, \textit{true}, \text{tlH}, \textit{true}) \qquad p_5 \equiv (\text{dpH}, \text{cpH} \vee \text{cpF}, \text{tlL}, \text{ctA} \vee \text{ctL})$$
$$p_3 \equiv (\text{dpH}, \text{cpH}, \text{tlL}, \textit{true})$$

Note that p_4, which is missing in Figure 2, admits additional behavior in the transition between p_3 to p_5 in accordance with the requirement. The LTL formula describing all valid signals with arbitrary phase length is as follows. The corresponding Büchi automaton is depicted in Figure 3a.

$$p_1 \wedge \mathcal{X}\ (p_1\ \mathcal{U}\ (p_2 \wedge \mathcal{X}\ (p_2\ \mathcal{U}\ (p_3 \wedge \mathcal{X}\ (p_3\ \mathcal{U}\ (p_4 \wedge \mathcal{X}\ (p_4\ \mathcal{U}\ \mathcal{G}\ p_5))))))) \tag{1}$$

Expressing the sequence of $p_1, \ldots, p_4$ as a precondition for p_5 leads to the following LTL formula and its corresponding Büchi automaton in Figure 3b.

$$\begin{aligned} p_1 \rightarrow (p_1 \wedge \mathcal{X}\ (p_1\ \mathcal{U}\ ((p_1 \vee p_2) \rightarrow (p_2 \wedge \mathcal{X}\ (p_2\ \mathcal{U}\ ((p_2 \vee p_3) \rightarrow \\ (p_3 \wedge \mathcal{X}\ (p_3\ \mathcal{U}\ ((p_3 \vee p_4) \rightarrow (p_4 \wedge \mathcal{X}\ (p_4\ \mathcal{U}\ \mathcal{G}\ p_5)))))))))) \end{aligned} \tag{2}$$

To express the property over continuous time while keeping the signal values abstract, the LTL formula can be transformed into MTL. For brevity, we use MTL with continuous semantics and remove the next operator $\mathcal{X}$.

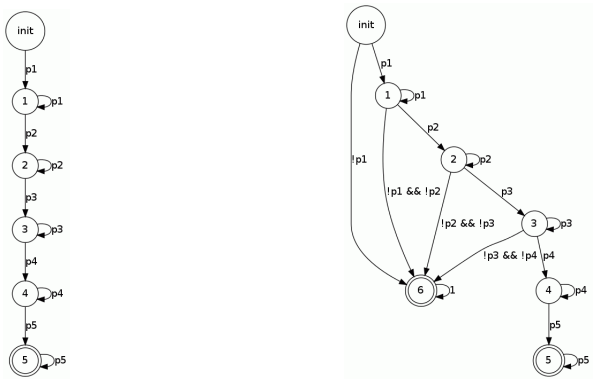
$$p_1\ \mathcal{U}\ (p_2\ \mathcal{U}\ (p_3\ \mathcal{U}\ (p_4\ \mathcal{U}\ \mathcal{G}\ p_5)))$$

The formula can be augmented with time bounds, e.g. to express the $10ms$ constraint as follows. Note that at this level of abstraction it is not possible to pick concrete values for a_i, b_i as these depend on step sizes and thresholds.

$$\begin{aligned} &p_1\ \mathcal{U}_{(a_1,b_1)}\ (p_2\ \mathcal{U}_{(a_2,b_2)}\ (p_3\ \mathcal{U}_{(a_3,b_3)}\ (p_4\ \mathcal{U}_{(a_4,b_4)}\ \mathcal{G}\ p_5))) \\ &\text{where } b_3 + b_4 \leq 0.1 \end{aligned}$$

Similar to LTL, expressing the sequence of $p_1, \ldots, p_4$ as a precondition for p_5 leads to the following MTL formula.

$$\begin{aligned} &p_1 \rightarrow (p_1\ \mathcal{U}_{(a_1,b_1)}\ ((p_1 \vee p_2) \rightarrow (p_2\ \mathcal{U}_{(a_2,b_2)}\ ((p_2 \vee p_3) \rightarrow \\ &\quad (p_3\ \mathcal{U}_{(a_3,b_3)}\ ((p_3 \vee p_4) \rightarrow (p_4\ \mathcal{U}_{(a_4,b_4)}\ \mathcal{G}\ p_5)))))))) \\ &\quad \text{where } b_3 + b_4 \leq 0.1 \end{aligned}$$



(a) (b)

Figure 3: Büchi automata corresponding to LTL formulae: (a) corresponds to (1) and (b) corresponds to (2)

To express the specification over continuous time signals, the above MTL formula can be interpreted as STL formula over continuous time signals $\mathrm{dp}[t]$ (demanded position at time t), $\mathrm{cp}[t]$ (current position at time t), $\mathrm{tl}[t]$ (torque limit at time t) and $\mathrm{ct}[t]$ (current torque at time t) where the abstract values are defined as propositions over continuous time signals.

$$
\begin{aligned}
\mathrm{dpL} &\equiv \mathrm{dp}[t] = l_1 \\
\mathrm{dpH} &\equiv \mathrm{dp}[t] = h_1 \\
\mathrm{cpL} &\equiv \mathrm{cp}[t] < l_1 + p(h_1 - l_1),\ \epsilon_1 \le p \le 1 - \epsilon_1 \\
\mathrm{cpH} &\equiv l_1 + p(h_1 - l_1) \le \mathrm{cp}[t] \le l_1 + (1 - \epsilon_2)(h_1 - l_1),\ \epsilon_1 \le p \le 1 - \epsilon_1 \le 1 - \epsilon_2 \\
\mathrm{cpA} &\equiv l_1 + (1 - \epsilon_2)(h_1 - l_1) < \mathrm{cp}[t] \\
\mathrm{tlL} &\equiv \mathrm{tl}[t] = l_2 \\
\mathrm{tlH} &\equiv \mathrm{tl}[t] = h_2 \\
\mathrm{ctL} &\equiv \mathrm{ct}[t] < \mathrm{tl}[t] - \epsilon_3 \\
\mathrm{ctA} &\equiv \mathrm{tl}[t] - \epsilon_3 \le \mathrm{ct}[t] \le \mathrm{tl}[t] + \epsilon_3 \\
\mathrm{ctH} &\equiv \mathrm{tl}[t] + \epsilon_3 < \mathrm{ct}[t] \\
&\text{where } 0 \le l_i < h_i,\ 0 < \epsilon_i < 1
\end{aligned}
$$

The time bounds a_i, b_i can be expressed as a function of the parameters l_i, h_i, ϵ_i, p. Thus, the specification has the form $\forall l_i, h_i, \epsilon_i, p\ P(l_i, h_i, \epsilon_i, p) \rightarrow \varphi(l_i, h_i, \epsilon_i, p)$ where $P(l_i, h_i, \epsilon_i, p)$ is a predicate identifying valid parameter configurations and $\varphi(l_i, h_i, \epsilon_i, p)$ is the STL formula where time bounds and propositions depend on parameters.

This formulation is a challenge for currently available falsification tools, since a time bound must be either numerical or may consist of a single parameter but does not admit complex functions of parameters or internal state. Moreover, there is no possibility to quantify over a predicate describing admissible param-

Figure 4: Testing of a diagnosis function by using a sensor model as a precondition.

eter configurations. Finally, selecting the input signal in such a way that the precondition, which involves both inputs and states, is met cannot be expressed in the tools.

Specifications based on properties of discrete events

The following example concerns test case generation for sensor diagnosis. The intention of sensor diagnosis functions is the following: "A diagnosis function shall provide a valid diagnosis return value iff the signal coming from the sensor allows the software to compute the diagnosis." Expressing in terms of pre- and postconditions yields: "Given that the sensor signal is of a certain form (A), then the diagnosis function provides a valid return value and signals to other functions that a return value is available (B). In all other cases, the diagnosis function does not signal a valid return value.[1]" Figure 4 exemplifies this approach. Note that defining B in this case is rather simple, yet defining relevant sensor signals in A is a challenge for engineers.

The starting point is often a previous measurement that serves as a reference (see below). As shown in Figure 5, such a reference signal often follows certain sequences of different phases. Here, a signal is by default at a *low* value for a potentially long time. As a result of a physical event in the environment the signal exhibits a *rise*. We are interested in a *stable* value after the rise. However, the signal may have *unstable* intervals due to a noisy underlying physical process. We only want to provide diagnosis return values for stable signals. Hence, if the signal is stable for a long enough time interval, we provide a return value when the signal finally *fall*s to its default value. We annotate the point where a valid return value is signaled in Figure 5; additionally, we show a second trajectory that falls before a diagnosis can be performed.

For testing and verification, we want to generalize from these individual traces to a relevant test input set. Sampling around a tube of reference signals is neither sufficient nor is it easily interpretable, since this does not necessarily result in a signal that can be diagnosed. One possibility is to model a generalized signal using hybrid automata. Figure 6 shows such an input generation automaton on the left, where each mode of the automaton represent a characteristic phase of the reference signals and we use state variables to describe timing and signal characteristics. Additionally, Figure 6 displays the corresponding observer automaton on the right. The observer synchronizes to the input generation automaton by a label `stable` that indicates that a stable measurement should have

[1] In these cases, there is no requirement on the return value.

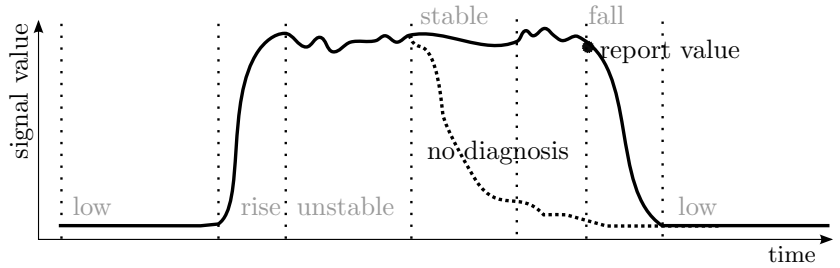


Figure 5: Example of a reference trajectory from sensor measurements.

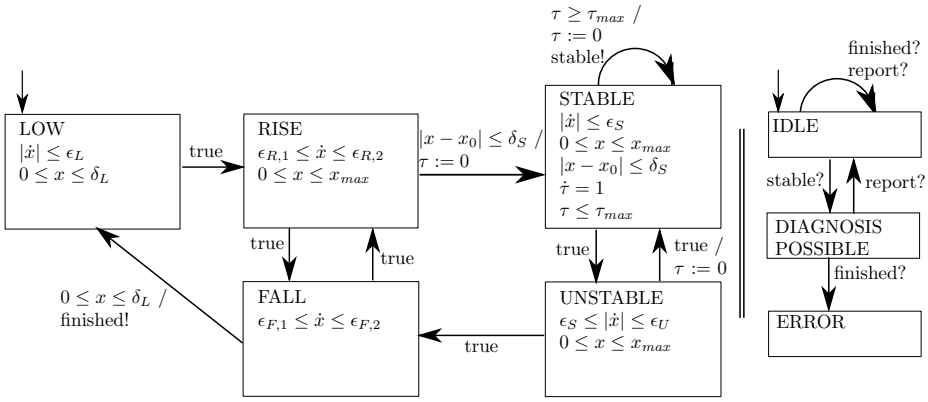


Figure 6: Example of a sensor model using a hybrid automaton.

been possible and a label `falling` to indicate when a report shall be provided by the implementation (checked by label `report`).

When using hybrid automata for test case generation, automata sampling is a major challenge, i. e., how to select relevant traces. A quantitative semantics similar to the one defined for STL might be helpful. Moreover, ease of specification for practitioners is crucial to enable adoption of specification formalisms. High-level specification languages that allow engineers to express properties at a suitable level of abstraction are needed. In the given example, it would be helpful to specify the stable/unstable property as a property of the frequency spectrum. While it is possible to define a filter in form of a differential equation and cast the frequency property as a property over the filtered signal, it is not very convenient.

Note that a formulation in a temporal logic (over phases) would have resulted in a blowup that is impractical for industrial applications. This is in contrast to other work that has proposed to use STL to describe test inputs [8].

Specifications based on comparison to a reference

In an initial development stage, a specification of the system may not be available. Instead engineers may rely on a reference trace, measured or simulated

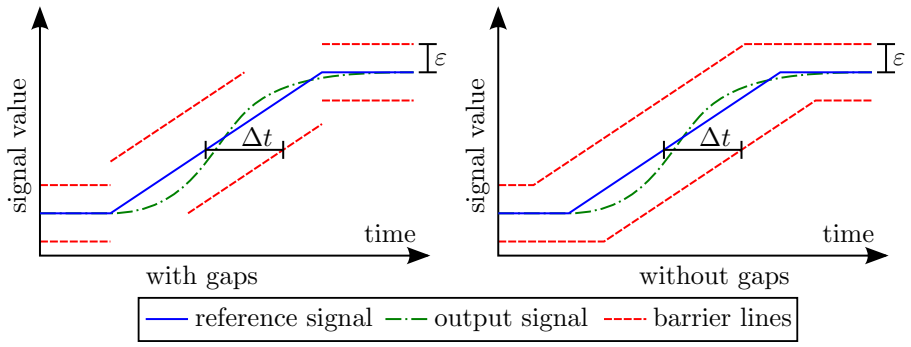


Figure 7: Example of a requirement defined by the similarity to a reference signal. Left: original requirement. Right: (τ, ε)-closeness

from a previous or similar system. Hence, we want to check that our new implementation behaves similarly to the given reference. While similarity is hard to define generally, engineers typically have some notion of what is needed. However, similarity may depend on the context of the system, such as concrete inputs. In our general framework with precondition A and postcondition B, a single reference signal can be seen as one element b in B and the context in this case would be one element a in A for a single input and context. The corresponding postcondition $B' \subseteq B$ is formed by the signals similar to b. This partial specification is very useful in practice for falsification. However, in industry this process is often ad hoc, performed by visual inspection, as formalizing similarity is tedious and sometimes not even possible. As an example, Figure 7 shows a reference signal with the following required similarity: "(1) When the reference signal is constant, the difference to the output signal should be less then ε. (2) When the reference signal changes, the output signal should reach the same value within a time delta of at most Δt." Note that this requirement does not talk about the window between the parts (1) and (2); this results in gaps in the specification as shown in the left hand side of Figure 7. The value of the signal is unconstrained from above or below for some time instants. Practitioners typically fill these gaps using their intuition and domain knowledge. The right hand side figure shows how these gaps might be filled satisfying the designer's intentions.

While individual references are very valuable in early development stages, we would like to generalize to a "complete" specification over all relevant contexts A. For instance, in this case, this could involve different slopes s for the interval where the reference signal is rising. This necessitates the generalization of our similarity notion from individual references to a set of contexts. One challenge is to find a metric which handles the possible time shift in the comparison of two signals. Classical formalisms like MTL are not appropriate, since we cannot directly formulate shifts in signals[2].

[2]For an industrial approach see [2].

With the notion of (τ, ε)-closeness from [1], the right hand side specification in Figure 7 can be described through $(\Delta t - \frac{\varepsilon}{s}, \varepsilon)$-closeness. However, the slope s is explicitly used in this formalization. Therefore, the closeness notion is parameterized in a property of the reference signal, even though the original requirement did not depend on the parameter s. This illustrates a drawback of using (τ, ε)-closeness for requirements divided into time segments: due to time shifts on signals defined on time domains bounded from above or below, (τ, ε)-closeness is often counterintuitive.

As discussed in [4], it is not yet sufficiently clear which classes of specifications can be effectively mapped on one single metric, e. g. neither (τ, ε)-closeness nor other notions like the Skorokhod metric handle all possible specifications with time shifts. Another important question from the practical point of view is how such metrics can be leveraged for test generation.

3 Outlook

In this paper, we have presented specification patterns that are typically encountered in automotive embedded systems. We have discussed challenges in formalization, esp. when writing specifications for test case generation. This benchmark underlines the need for (i) specification formalizations that can be easily used by practitioners and (ii) support for these specification patterns in test-case generation tools, especially w. r. t. input restriction guidance.

Acknowledgements: The authors gratefully acknowledge financial support by the European Commission project UnCoVerCPS under grant number 643921.

References

[1] H. Abbas, H. Mittelmann, and G. Fainekos. Formal property verification in a conformance testing framework. In *Formal Methods and Models for Codesign (MEMOCODE), 2014 Twelfth ACM/IEEE International Conference on*, pages 155–164. IEEE, 2014.

[2] M. Conrad, S. Sadeghipour, and H.-W. Wiesbrock. Automatic evaluation of ECU software tests. Technical report, SAE Technical Paper, 2005.

[3] G. E. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel. Verification of automotive control applications using S-TaLiRo. In *American Control Conference (ACC), 2012*, pages 3567–3572. IEEE, 2012.

[4] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts. Benchmarks for model transformations and conformance checking. In *1st International Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH)*, 2014.

[5] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts. Powertrain control verification benchmark. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 253–262. ACM, 2014.

[6] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia. Mining requirements from closed-loop control models. In C. Belta and F. Ivancic, editors, *HSCC*, pages 43–52. ACM, 2013.

[7] R. Matinnejad, S. Nejati, L. Briand, T. Bruckmann, and C. Poull. Automated model-in-the-loop testing of continuous controllers using search. In *Search Based Software Engineering*, volume 8084 of *LNCS*, pages 141–157. Springer Berlin Heidelberg, 2013.

[8] B. Wilmes and A. Windisch. Considering signal constraints in search-based testing of continuous systems. In *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*, pages 202–211. IEEE, 2010.