# Verification of Automotive Control Applications using S-TaLiRo

Georgios E. Fainekos, Sriram Sankaranarayanan, Koichi Ueda and Hakan Yazarel

***Abstract*— S-TALIRO is a software toolbox that performs stochastic search for system trajectories that falsify real-time temporal logic specifications. S-TALIRO is founded on the notion of robustness of temporal logic specifications. In this paper, we present a dynamic programming algorithm for computing the robustness of temporal logic specifications with respect to system trajectories. We also demonstrate that typical automotive functional requirements can be captured and falsified using temporal logics and S-TALIRO.**

## I. INTRODUCTION

Models of automotive control systems can be fairly complex. Such models are usually developed in a modeling environment that supports a block diagram graphical user interface for modeling the continuous plant and control components, enhanced with finite state machines. Examples of commonly used modeling and simulation environments include Simulink/Stateflow (TM), Scade (TM), LabVIEW (TM) and Ptolemy [1]. A typical system model may contain thousands of blocks organized in hierarchical subsystems that go several levels deep. The complexity is further exacerbated by discrete switches in the form of switching blocks, lookup tables, communication of subsystems through shared global variables and so on. Therefore, capturing such an overall model in an analytical form amenable to traditional control theoretic analysis (and possibly design) is not a feasible task.

Testing is a commonly used approach to check that the model satisfies the correctness properties stated in the requirements. However, the success of testing depends primarily on the ability to write test cases that exhaustively cover all the possible corner cases under which a property can possibly fail. For the case of control systems, the space of behaviors is (uncountably) infinite. As a result, the process of testing often fails to expose potential failures in the model. This has lead to work on uniform random testing or "guided" random testing, where the choice of inputs is stochastic, based on some guidance criteria [2]–[5].

Yet, the problem of systematically guiding the search for an input that falsifies a given property has not been well-understood. At its heart, the problem lies in defining mathematically sound notions of "how far away" a given execution of the system is from violating a property. If such a notion of distance were available – *and at the same time easy to compute* – it could be used as an objective function in a global optimization setting. The goal would be to choose inputs that minimize the distance between the resulting execution trace and a violation of the property of interest. Recently, we have made progress towards obtaining mathematical notions of *robustness metrics* [6]–[8] that quantify the distance between the execution trace of a given *hybrid system* combining continuous evolution of state variables with discrete mode switches and a property stated in a commonly used logic for real-time trace properties called Metric Temporal Logic (MTL) [9]. The overall framework has been implemented in a Matlab toolbox called S-TALIRO [10]. This framework allows us to use robustness computations over traces guided by a large class of global optimization techniques including genetic algorithms, ant-colony optimization [11] and stochastic optimization techniques using Monte-Carlo simulations [6], [7] and the Cross-Entropy Method [12].

Frameworks such as S-TALIRO can be used to systematically test a given model by searching for an input that falsifies the given property of interest. Even if a falsification cannot be found by this process, the traces with least robustness discovered by the search are often useful to the developers in showing examples where the simulation comes "closest" to violating the property.

The goal of this paper is to demonstrate how S-TALIRO can be applied to problems in the automotive domain. To this end, we identified as one of the major challenges to applying S-TALIRO to industrial applications the computation time of the robustness metric.

*Contributions:* We present an improved algorithm for the computation of the robustness value based on the dynamic programming principle [13]. The new algorithm has linear worst case execution time with respect to the size of the formula, the number of samples of the system trajectory and the bounds of the temporal operators. We compare the new algorithm with the algorithm of our earlier work [8]. Finally, we present two case studies on automotive applications.

*Related Research:* The applicability of metaheuristics for test generation on industrial size problems has been established by Zhao et al. [14]. The authors utilize genetic algorithms to generate tests for Simulink/Stateflow models. The cost function is the number of regions and states that have been covered. The work in [15] utilizes rapidly exploring random trees guided by automata that recognize all the prefixes that violate a syntactically safe Linear Temporal Logic (LTL) formula. A different algorithm for computing the robustness of MTL formulas as defined in [8] is presented

This work was partially supported by a grant from the NSF Industry/University Cooperative Research Center (I/UCRC) on Embedded Systems at Arizona State University and NSF awards CNS-1017074, CNS-1116136 and CNS-1016994.

G. Fainekos is with the School of Computing, Informatics and Decision Systems Engineering at Arizona State University. E-mail: fainekos@asu.edu.

S. Sankaranarayanan is with the Department of Computer Science, University of Colorado, Boulder. E-mail: srirams@colorado.edu

K. Ueda and H. Yazarel are with the Toyota Technical Center. E-mail: {koichi.ueda,hakan.yazarel}@tema.toyota.com

in [16]. In [17], a different notion of robustness for temporal logic specifications is developed, which is also used as a fitness function for optimization problems.

## II. The MTL Falsification Problem

In this work, we target directly executable models of automotive control applications. As such, we will assume that the system will be tested for a range of initial conditions, system parameters and input signals. In particular, we will assume that the system under study is modeled in Simulink/Stateflow. However, what we propose here can be readily applied to any other model based design environment such as Ptolemy [1] or LabVIEW.

Formally, we view a system $\Sigma$ as a mapping from initial conditions $\mathcal{X}_0$, system parameters $P$ and input signals $U^R$ to output signals $\mathcal{Y}^R$. Here, $R$ is an abstract time domain, $U$ is the set of input values (input space) and $\mathcal{Y}$ is the set of output values (output space). Thus, a system $\Sigma$ is a function $\Delta_\Sigma : \mathcal{X}_0 \times P \times U^R \to \mathcal{Y}^R$ which takes as input an initial condition $\chi_0 \in \mathcal{X}_0$, a parameter vector $p \in P$ and a signal $u : R \to U$ and produces as output a signal $\eta : R \to \mathcal{Y}$.

Since our analysis is based on performing system simulations, we will assume the existence of a sampling function $\tau : \mathbb{N} \to R$ that returns for each sample $i$ its time stamp $\tau(i)$. In practice, $\tau$ is a partial function $\tau : N \to R$ with $N \subset \mathbb{N}$ and $|N| < \infty$. We will abuse notation and denote by $|\tau|$ the cardinality of the domain of $\tau$, i.e., $|\tau| = |N|$. A *timed state sequence* or *trace* is the pair $\mu = (\eta \circ \tau, \tau)$. We will also denote $\eta \circ \tau$ by $\sigma$.

Our goal – in the line of work that we initiated in [7] – is to infer the correctness of the system $\Sigma$ by observing its response (output signals) to particular input signals, initial conditions and parameter values. In particular, we are interested in finding witnesses, i.e., output signals, which prove that a requirement or specification is not satisfied by the system. The process of discovering such witnesses is usually referred to as *falsification*.

The next question that needs to be answered is how do we formally capture informal specifications regarding the correct or expected behavior of the system. We observe that Metric Temporal Logic (MTL) [9] is an appropriate mathematical formalism that can capture such requirements for automotive control systems. MTL formulas are built over a set of propositions using combinations of the traditional and temporal operators. In our case, the set of atomic propositions $AP$ label subsets of the output space $\mathcal{Y}$. In other words, we define an observation map $\mathcal{O} : AP \to \mathcal{P}(\mathcal{Y})$ such that for each $\pi \in AP$ the corresponding set is $\mathcal{O}(\pi) \subseteq \mathcal{Y}$. Here, $\mathcal{P}(S)$ denotes the powerset of a set $S$. Traditional logic operators are the *conjunction* ($\wedge$), *disjunction* ($\vee$), *negation* ($\neg$), *implication* ($\rightarrow$) and *equivalence* ($\leftrightarrow$). Some of the temporal operators, which we will be using here, are *eventually* ($\Diamond_{\mathcal{I}}$), *always* ($\Box_{\mathcal{I}}$) and *until* ($\mathcal{U}_{\mathcal{I}}$). The subscript $\mathcal{I}$ imposes timing constraints on the temporal operators. The interval $\mathcal{I}$ can be open, half-open or closed, bounded or unbounded, but it must be non-empty ($\mathcal{I} \neq \emptyset$). For example, MTL can capture the requirement that "all the observable trajectories $y(t) \in \mathbb{R}$ attain a value in the set $[10, +\infty)$" ($\Diamond p_1$ with $\mathcal{O}(p_1) = [10, +\infty)$) or that "whenever the value of $y$ drops below 10, then it should go above 10 within 5 sec and remain above 10 for at least 10 sec" ($\Box(\neg p_1 \rightarrow \Diamond_{[0,5]}\Box_{[0,10]}p_1)$).

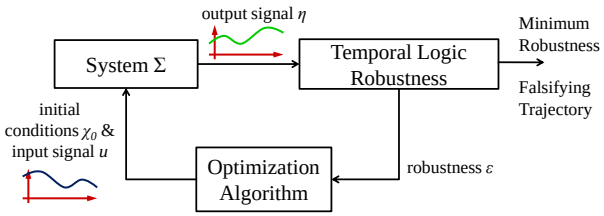


Fig. 1. Overview of the solution to the MTL falsification problem posed as an optimization problem.

The MTL falsification problem can be stated as follows.

*Problem 2.1 (MTL Falsification):* For an MTL specification $\varphi$, the MTL falsification problem consists of finding an output signal $\eta$ of the system $\Sigma$ starting from some valid initial state $\chi_0$ under a parameter vector $p$ and an input signal $u$ such that $\eta$ does not satisfy specification $\varphi$.

The challenges in solving the MTL falsification problem are multiple. The main problem is essentially how to guide the search for such a falsifying trajectory. We remark that the system dynamics of $\Sigma$ are not known to us in some analytical form because most of industrial size models will contain look-up tables and black-box blocks (object code) from various suppliers.

In our previous work, we utilized the notion of robustness of temporal logic formulas [8] in order to convert the falsification problem into an optimization problem [6], [7], [10]–[12]. Briefly, temporal logic robustness provides a measure of how robustly a trajectory satisfies a temporal logic specification. Positive robustness implies that the trajectory satisfies the specification and, moreover, that there exists a neighborhood of trajectories (or signals) that also satisfy the specification. Negative robustness implies that the trajectory does not satisfy the specification. Thus, in order to falsify the specification, we can use the temporal logic robustness as a cost function which we attempt to minimize.

The general overview of the solution of the MTL falsification problem as an optimization problem appears in Fig. 1. Based on that principle, we have developed the Matlab toolbox S-TaLiRo [10]. Given a system and its specification, S-TaLiRo searches for a system trajectory that minimizes the robustness value of the specification.

## III. MTL Robustness

In this section, we review the robust semantics of MTL formulas. Details are available in our previous work [6], [8].

*Definition 3.1 (MTL Syntax):* Let $AP$ be the set of atomic propositions and $\mathcal{I}$ be any non-empty interval of $\overline{\mathbb{R}}_+$. The set $MTL$ of all well-formed MTL formulas is inductively defined as $\varphi ::= \mathbf{T} \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathcal{U}_{\mathcal{I}} \varphi$, where $p \in AP$ and $\mathbf{T}$ is *true*. If there are no timing constraints on the operators, then the formula is in LTL.

We provide semantics that maps an MTL formula $\varphi$ and a trace $\mu$ to a value drawn from a partially ordered set $\mathcal{V}$.

The semantics for the atomic propositions evaluated for $\mu(i)$ consists of the distance between $\sigma(i)$ and the set $\mathcal{O}(p)$ labeling atomic proposition $p$. Intuitively, this distance represents how robustly the point $\sigma(i)$ lies within (or is outside) the set $\mathcal{O}(p)$. If this distance is zero, then the smallest perturbation of the point $\sigma(i)$ can affect the outcome of $\sigma(i) \in \mathcal{O}(p)$. We denote the robust valuation of the formula $\varphi$ over the trace $\mu$ at sampling instance $i$ by $[\![\varphi, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i)$. Formally, $[\![\cdot,\cdot]\!]_{\mathbf{d}} : (MTL \times \mathcal{P}(\mathcal{Y})^{AP}) \to (\mathcal{L}(\mathcal{H}) \times N \to \mathcal{V})$, where $N = \tau^{-1}(R) = \{i \in \mathbb{N} \mid \tau(i) \in R\}$.

*Definition 3.2 (Discrete-Time Robust Semantics):* Consider an extended generalized quasi metric space $(\mathcal{Y}, \mathbf{d})$ (see [6] for a definition). Let $\mu$ be a trace of $\Sigma$, $v \in \mathcal{V}$ and $\mathcal{O} \in \mathcal{P}(\mathcal{Y})^{AP}$, then the robust semantics of any formula $\varphi \in MTL$ with respect to $\mu$ is recursively defined as:

$$
\begin{aligned}
[\![\mathbf{T}, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= \bigsqcup \mathcal{V} := \top \\
[\![p, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= \mathbf{Dist}_{\mathbf{d}}(\sigma(i), \mathcal{O}(p)) \\
[\![\neg\varphi_1, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= - [\![\varphi_1, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) \\
[\![\varphi_1 \vee \varphi_2, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= [\![\varphi_1, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) \sqcup [\![\varphi_2, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) \\
[\![\varphi_1 \mathcal{U}_{\mathcal{I}} \varphi_2, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= \bigsqcup_{i' \in \tau^{-1}(\tau(i) +_R \mathcal{I})} ([\![\varphi_2, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i') \sqcap \\
&\qquad \sqcap_{i \le i'' < i'} [\![\varphi_1, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i'')
\end{aligned}
$$

where $\mathbf{Dist}_{\mathbf{d}}(\sigma(i), \mathcal{O}(p))$ is the signed distance of $\sigma(i)$ from $\mathcal{O}(p)$ under the metric $\mathbf{d}$ (see [8] for a definition), $-$ is an unary operator for complement over $\mathcal{V}$, $t +_R \mathcal{I} = \{t'' \in R \mid \exists t' \in \mathcal{I} \,.\, t'' = t + t'\}$ and $\sqcup$ and $\sqcap$ stand for the supremum and infimum, respectively. The semantics of the other operators can be defined using the above basic operators. E.g., $\Diamond_{\mathcal{I}} \phi \equiv \mathbf{T} \mathcal{U}_{\mathcal{I}} \phi$ and $\Box_{\mathcal{I}} \phi \equiv \neg \Diamond_{\mathcal{I}} \neg \phi$.

For the purposes of the following discussion, let $(\mu, i, \mathcal{O}) \models \varphi$ denote the standard Boolean MTL satisfiability. For clarity in the presentation, we define the satisfiability relation for the base case, i.e., for atomic propositions: $p \in AP$, $(\mu, i, \mathcal{O}) \models \varphi$ if $\sigma(i) \in \mathcal{O}(p)$. It is easy to show that if the signal satisfies the property, then its robustness is non-negative and, similarly, if the signal does not satisfy the property, then its robustness is non-positive.

In our previous work [8], we had implemented the MTL robustness computation algorithm using a forward progression algorithm. The precise complexity of the computation of MTL robustness using formula rewriting procedures is still an open problem [8]. However, the time complexity is at least as hard the time complexity of the Boolean version of the same algorithm [18], which is in the worst case exponential in the size of the formula.

## IV. Dynamic Programming Algorithm for Temporal Logic Robustness

In this section, we present a dynamic programming algorithm for computing the robustness of temporal logic formulas with respect to a timed state sequence.

The basic principle of the dynamic programming algorithm is that it can reuse previously computed results [13]. Here, the results which are going to be reused are the temporal logic robustness of the subformulas of a formula $\phi$ at the current and future points in time. In brief, the algorithm starts by constructing the parse tree of the temporal logic formula $\phi$. Each subformula $\psi_i$ of $\phi$ is uniquely identified starting from the leafs and represents a row $i$ in the dynamic programming table. The columns $j$ of the table represent the different timing instants of the trace. Then, the algorithm starts by filling the values of each $\phi_i$ at the last signal sampled and, then, proceeds backwards until the initial sampling time is reached. The pseudocode for the dynamic programming algorithm appears in Algorithm 1.

**Algorithm 1** Temporal Logic Robustness Computation

**Input**: The MTL formula $\phi$, the trace $\mu = (\sigma, \tau)$, the metric $\mathbf{d}$ and the observation map $\mathcal{O}$
**Output**: Return the value stored in $r[1, 1]$

```
1:  procedure DP-TaLiRo(φ, O, μ, d)
2:      for j ← |τ| to 1; for i ← |φ| to 1 do
3:          if ψ_i = T then r[i, j] = ⊤                    ▷ ⊤ := ⊔V
4:          else if ψ_i = p then r[i, j] ← Dist_d(σ(j), O(p))
5:          else if ψ_i = ¬ψ_k then r[i, j] ← −r[k, j]
6:          else if ψ_i = ψ_{k_1} ∨ ψ_{k_2} then
7:              r[i, j] ← r[k_1, j] ⊔ r[k_2, j]
8:          else if ψ_i = ψ_{k_1} U_I ψ_{k_2} then
9:              if j = |τ| then r[i, j] ← K_∈(0, I) ⊓ r[k_2, j]
10:             else if I = [0, +∞) then
11:                 r[i, j] ← r[k_2, j] ⊔ (r[k_1, j] ⊓ r[i, j + 1])
12:             else
13:                 b_l ← min J(j, I); b_u ← max J(j, I);
14:                 r_min ← ⊓_{j≤j'<b_l} r[k_1, j'];
15:                 r[i, j] ← ⊥;                           ▷ ⊥ := ⊓V
16:                 for j' ← b_l to b_u do
17:                     r[i, j] ← r[i, j] ⊔ (r[k_2, j'] ⊓ r_min);
18:                     r_min ← r_min ⊓ r[k_1, j'];
19:                 end for
20:                 if sup I = +∞ then
21:                     r[i, j] ← r[i, j] ⊔ (r[k_1, j] ⊓ r[i, j + 1])
22:                 end if
23:             end if
24:         end if
25:     end for
26: end procedure
```

where $k, k_1, k_2 > i$; $K_{\in}(a, A) = \top$ if $a \in A$ and $\bot$ otherwise; and $J(j, \mathcal{I}) = \tau^{-1}((\tau(j) +_R \mathcal{I}) \cap (\tau(j+1) +_R \mathcal{I}))$ if $\sup \mathcal{I} = +\infty$ and $J(j, \mathcal{I}) = \tau^{-1}(\tau(j) +_R \mathcal{I})$ otherwise.

*Theorem 4.1:* Given an MTL formula $\phi$, a trace $\mu = (\sigma, \tau)$, a metric $\mathbf{d}$ and an observation map $\mathcal{O}$, then

$$[\![\phi, \mathcal{O}]\!]_{\mathbf{d}}(\mu, 0) = \text{DP-TaLiRo}(\phi, \mathcal{O}, \mu, \mathbf{d})$$

Moreover, Algorithm 1 has worst case running time $O(|\phi||\tau|c)$, where $c = \max_{0 \le j \le |\tau|, \mathcal{I} \in T(\phi)} |[j, \max J(j, \mathcal{I})]|$ and $T(\phi)$ contains all the timing constraints $\mathcal{I}$ of the temporal operators that appear in $\phi$.

The proof of Theorem 4.1 is based on rewriting the robustness semantics of Def. 3.2. Similar proofs have appeared in

TABLE I

COMPARISON OF DP-TALIRO VS FW-TALIRO ON THE SPECIFICATIONS OF EXAMPLE 5.2. THE SIMULATION TRAJECTORY HAS 1673 SAMPLES.

| Spec. | DP-TALIRO (sec) | FW-TALIRO (sec) |
|---|---|---|
| $\phi_{e1}$ | 0.0183 | 60 < |
| $\phi_{e2}$ | 0.0171 | 1.0984 |
| $\phi_{e2.1}$ | 0.0175 | 60 < |
| $\phi_{e2.2}$ | 0.0274 | 60 < |
| $\phi_{e3}$ | 0.0127 | 0.0069 |

[8]. The running time of the algorithm for an LTL formulae $\varphi$ is $O(|\varphi||\tau|)$, i.e., it is linear in the size of the formula and the number of samples in the simulation trajectory. This is easy to verify since all the entries $r[i,j]$ of the table in Algorithm 1 require at most 3 $\inf$ or $\sup$ operations. On the other hand, when an MTL formula is considered, then at most $c$ $\inf$ and/or $\sup$ operations are required for the until operator. $c$ is the maximum number of samples that can appear from any sampling point $j$ up to the maximum sampling point allowed $b_u$ by the timing constraints of the operator. In detail, $c$ is the sum of $\sqcap$ operations in line 14 of Alg. 1 plus the number of iterations in the for loop in lines 16-19.

The hidden cost in the above analysis is the running time of the distance computation function. The computational complexity of the distance computations depends on the type of the sets used for modeling the regions of interest in the state-space. Details for the Euclidean distance metric are presented in [8].

### A. Experimental Comparison of two Robustness Computation Algorithms

In [8], we presented an algorithm for computing temporal logic robustness which is based on rewriting techniques. Even though the exact computational complexity is still an open problem, it is as hard as the Boolean version of the algorithm [18], which is at least linear in size of the input trajectory and exponential in the size of the formula and the timing constraints on the operators.

We have implemented both algorithms in ANSI C and both can be executed within the Matlab environment. The dynamic programming version of the algorithm is referred to as DP-TALIRO while the rewriting version of the algorithm is referred to as FW-TALIRO. Both are available at [19].

In order to compare the two algorithms on realistic specifications and signals, we use the formulas and trajectories generated for the examples in Section V. The results are presented in Table IV-A. We observe that the experimental running time of FW-TALIRO is highly dependent on the structure of the formula. On the other hand, DP-TALIRO essentially has constant running time with respect to the length of the simulation trajectory. Thus, DP-TALIRO is better suited for our falsification framework.

## V. S-TALIRO APPLICATION TO AUTOMOTIVE EXAMPLES

In this section, we present the application of S-TALIRO to two automotive applications available in the literature. First, we demonstrate S-TALIRO on the illustrative example from [14]. We establish that S-TALIRO can capture and

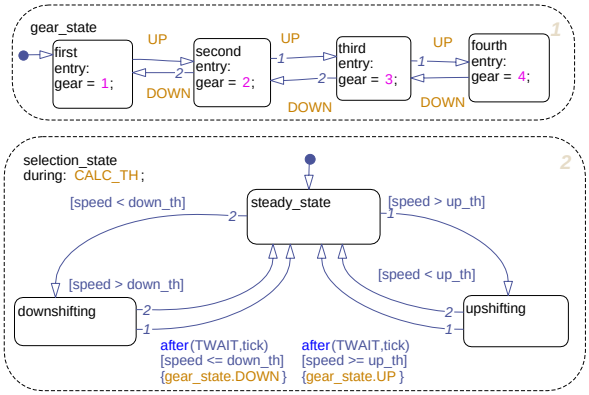


Fig. 2. The switching logic for the automatic drivetrain in Example 5.1.

TABLE II

THE STATE MAPPING OF THE COMPOSITION OF THE TWO FSM.

| | First | Second | Third | Fourth |
|---|---|---|---|---|
| steady state | $q_1$ | $q_2$ | $q_3$ | $q_4$ |
| upshifting | $q_5$ | $q_6$ | $q_7$ | $q_8$ |
| downshifting | $q_9$ | $q_{10}$ | $q_{11}$ | $q_{12}$ |

falsify the requirements posed in [14] on the same problem. Therefore, S-TALIRO can be thought as a generalization of the approach proposed in [14]. Second, we demonstrate that S-TALIRO can not only solve the challenge problem posed in [20], but help the designer easily explore other properties of the system. All the case studies presented here are included with the S-TALIRO distribution [19].

*Example 5.1:* The illustrative example that is presented in [14] is the Automatic Transmission model provided by Mathworks as a Simulink demo (`http://www.mathworks.com/products/simulink/demos.html`). This is a model of an automatic transmission controller. According to the report generated by `sldiagnostics` (a Matlab function), the model contains 69 blocks out of which there are 2 integrators (i.e., 2 continuous state variables: wheel speed and engine speed (RPM)), 3 look-up tables, 3 look-up 2D tables and a Stateflow chart. The Stateflow chart (see Fig. 2) contains two concurrently executing Finite State Machines (FSM) with 4 and 3 states, respectively, and non-constant switching guard conditions. Even though this is a small size model, it already exhibits all the complexities that prevent formal modeling and analysis.

For comparing our results with [14], we made the same modifications to the model in terms of inputs-outputs. That is, the only input to the system is the throttle schedule, while the break schedule is set simply to 0 for the duration of 30 sec. Also, we modified the model to output the state of the synchronous composition of the two FSM (see Table 5.1).

The method proposed in [14] generates tests such that certain regions of the hybrid state space of the system are visited (coverage requirement). In particular, the requirement is to generate tests such that: (i) the vehicle speed $v$ exceeds 120km/h, (ii) the engine speed $\omega$ exceeds 4500RPM, and, (iii) all states are reached in the switching logic. Assuming that the state vector is $[v\ \omega]^T$ and that the states in the

FSM are $Q = \{q_1, \ldots, q_{12}\}$ (see Table 5.1), then the coverage requirement above can be captured by the LTL formula $\phi_{e0} = \neg(\wedge_{i=1}^{9} p_i)$, where each atomic proposition $p_i$ is mapped to: $\mathcal{O}(p_1) = Q \times [120, +\infty) \times \mathbb{R}$, $\mathcal{O}(p_2) = Q \times \mathbb{R} \times [4500, +\infty)$, and $p_3$ to $p_9$ are mapped to a column or a row in Table 5.1, e.g., $\mathcal{O}(p_3) = \{q_1, q_5, q_9\} \times \mathbb{R}^2$. Note that we add the negation in $\phi_{e0}$ because we are trying to falsify the requirement.

The outcome of the S-TALIRO appears in Fig. 3. The Simulink model was simulated 41 times for this particular test. As evident from the figure, the vehicle indeed reaches the specified thresholds. Running the Simulink toolbox for Model Coverage of Stateflow charts, we can also verify that all the states were indeed visited. △

*Example 5.2:* The second example concerns a more complex model of a powertrain system [20]. The system is modeled in Checkmate [21]. It has 6 continuous state variables and 2 Stateflow charts with 4 and 6 states, respectively. The Stateflow chart for the shift scheduler appears in Fig. 4. The system dynamics and switching conditions are linear. However, some switching conditions depend on the initial conditions of the system. The latter makes the application of standard hybrid system verification tools not a straightforward task.

The system is operating under constant road grade and throttle position, which are the initial parameters for the system. The challenge problem posed in [20] is to find values for the initial parameters such that starting from 0 speed, the gear transitions from second to first to second.

The LTL specification that captures the requirement for switching between gears is: $\phi_{e1} = \neg\Diamond(g_2 \wedge \Diamond(g_1 \wedge \Diamond g_2))$ where $g_1$ and $g_2$ are the atomic propositions indicating that the system operates in first and second gears, respectively. That is, $\mathcal{O}(g_1) = \{1\} \times \mathbb{R}^6$ and $\mathcal{O}(g_2) = \{3\} \times \mathbb{R}^6$. Note that in this example, the atomic propositions do not constrain the continuous state space. However, the information regarding the switching conditions that enable a transition is utilized in the hybrid distance metric for the robustness computation of the specification (see [6]). Again, we are looking for a trajectory that satisfies $\neg\phi_{e1}$. Since S-TALIRO performs falsification, the user must provide $\phi_{e1}$.

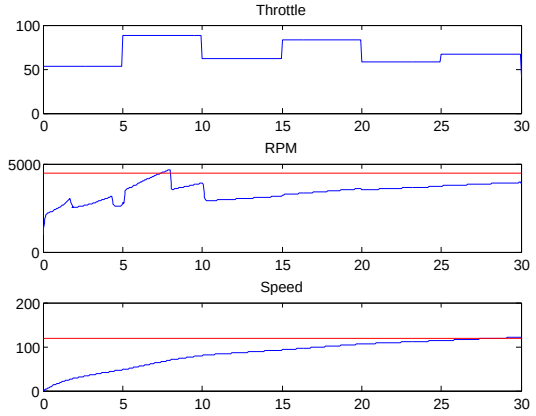


Fig. 3. The falsifying inputs/outputs of Example 5.1.

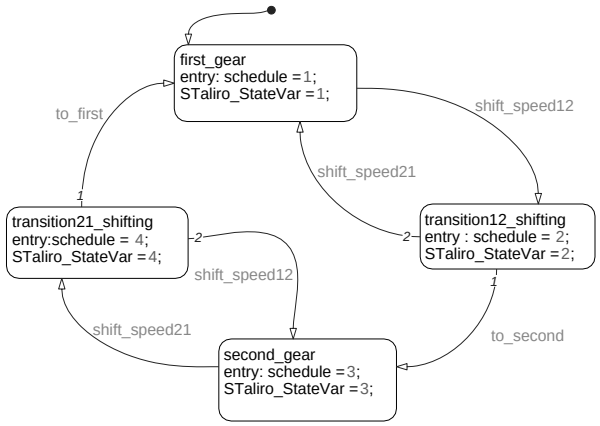


Fig. 4. The shift scheduler of Example 5.2.

Applying S-TALIRO to the above problem returns several different initial parameters that generate trajectories that falsify $\phi$. Figure 5 displays the shifting schedule for initial conditions throttle $\approx 18.8$ and road grade $\approx 0.0663$. Thus, it is possible indeed to have a non-required change of gears.

However, note that specification $\phi_e$ does not pose any restrictions between the timing of events. A more useful property is that the gear change from second to first to second should not happen within 2.5 sec, for example. The requirement $\phi_{e2.1} = \neg\Diamond(g_2 \wedge \Diamond(g_1 \wedge \Diamond_{[0,2.5]} g_2))$ would not work because we have to measure time since the first time that event $g_1$ happened. The subformula $\Diamond(g_1 \wedge \Diamond_{[0,2.5]} g_2)$ is allowed to measure time from the last time that $g_1$ occurred. If we attempt to use the specification $\phi_{e2.2} = \Box(g_1 \rightarrow \Diamond_{[2.5,+\infty)} g_2)$, then $\phi_{e2.2}$ would also not work since there exist initial parameters that will force the vehicle to go downhill and, thus, never switch to gear 2. That is, the falsification is achieved simply because we do not switch to gear 2. This implies that in terms of falsification we must also require the system to switch to gear 2, i.e., $\phi_{e2.2} = \Box(\neg g_2 \vee \Box(g_1 \rightarrow \Diamond_{[2.5,+\infty)} g_2)$. But, again the property may be falsified simply because the duration of the simulation time is short enough that $g_2$ does not occur for a second time. This requirement might also fail simply because the last occurrence of $g_1$ happens too close to the end of the simulation that even though $g_2$ occurs for the second time the operator $\Diamond_{[2.5,+\infty)}$ trivially evaluates to **F** because the timing bounds are outside the time domain of the simulation.

If we were to restate the requirement that we are trying to impose on the system, then we would specify that "whenever the system enters state `first_gear`, then it should not enter

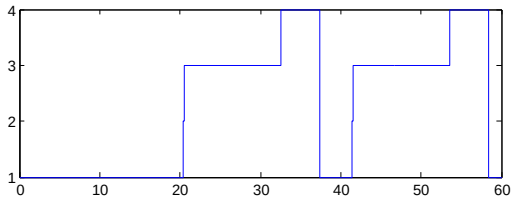


Fig. 5. The shift schedule falsifying requirement $\phi_{e1}$ in Example 5.2.

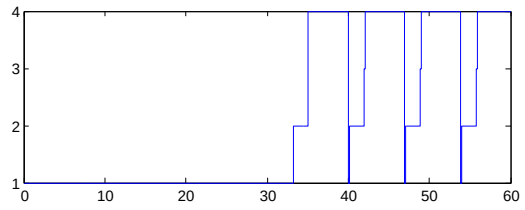


Fig. 6. The shift schedule that falsifies requirement $\phi_{e2}$ of Example 5.2.

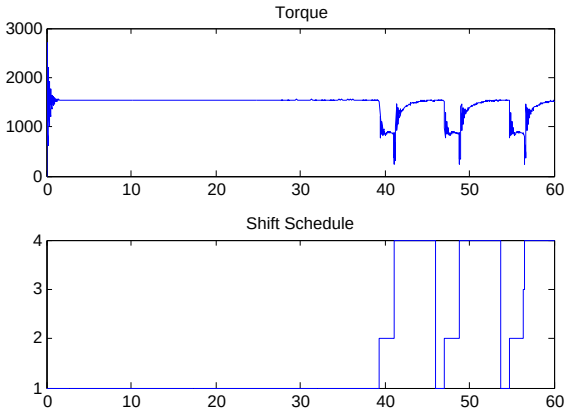


Fig. 7. The torque signal that falsifies requirement $\phi_{e3}$ of Example 5.2.

the state `second_gear` within 2.5 sec". This requirement is formally captured by: $\phi_{e2} = \Box((\neg g_1 \wedge X g_1) \rightarrow \Box_{[0,2.5]} \neg g_2)$. Figure 6 presents a shift schedule that falsifies formula $\phi_{e2}$. For this specification, the initial conditions were throttle $\approx$ 93.9 and road grade $\approx$ 0.2453 and S-TALIRO used 742 simulations. We remark that on an Intel Core Duo at 3.33GHz with 4.00 GB RAM and Windows Vista 64-bit each execution of the model takes about 3 sec and each robustness computation about 0.02 sec.

Another property of interest for powertrain systems is to verify that the jitter is within acceptable limits. This specification can be captured by requiring that whenever the system is in transition from gear 2 to gear 1, then the derivative of the torque is within certain bounds, or formally, $\phi_{e3} = \Box(g_{21} \rightarrow b)$ where $\mathcal{O}(g_{21}) = \{4\} \times \mathbb{R}^7$ and $\mathcal{O}(b) = \{x \in \mathbb{R}^7 \mid x_7 \leq 450\}$. In this case, the approximation of the derivative is outputted from the Simulink model and it is appended to the output signals. A falsifying trajectory that corresponds to initial parameters 91.86 and 0.2478 is presented in Fig. 7 and it was derived after 245 tests. $\triangle$

## VI. CONCLUSIONS

In this paper, we presented how S-TALIRO [10] – a tool for the falsification of temporal logic properties of hybrid systems – can be utilized for the verification of automotive applications. S-TALIRO can be used for analyzing Simulink/Stateflow models and it is publicly available [19]. We also demonstrated how improvements in the techniques used to compute robustness using dynamic programming techniques are key towards enhancing the performance of the falsification technique, as a whole. We believe that approaches along the lines of S-TALIRO offer a good trade-off between the exhaustiveness of model-checking and the scalability of techniques based on simulations.

*Acknowledgments:* The authors would like to thank Hengyi Yang for his help with implementing DP-TALIRO.

## REFERENCES

[1] J. Eker, J. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Sachs, and Y. Xiong, "Taming heterogeneity - the ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–144, Jan. 2003.

[2] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg, "On systematic simulation of open continuous systems." in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 2623. Springer, 2003, pp. 283–297.

[3] M. Branicky, M. Curtiss, J. Levine, and S. Morgan, "Sampling-based planning, control and verification of hybrid systems," *IEE Proc.-Control Theory Appl.*, vol. 153, no. 5, pp. 575–590, 2006.

[4] A. Bhatia and E. Frazzoli, "Incremental search methods for reachability analysis of continuous and hybrid systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 2993. Springer, 2004, pp. 142–156.

[5] T. Nahhal and T. Dang, "Test coverage for continuous and hybrid systems," in *CAV*, ser. LNCS, vol. 4590. Springer, 2007, pp. 449–462.

[6] H. Abbas, G. E. Fainekos, S. Sankaranarayanan, F. Ivancic, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems*, vol. (In Press), 2011.

[7] T. Nghiem, S. Sankaranarayanan, G. E. Fainekos, F. Ivancic, A. Gupta, and G. J. Pappas, "Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems," in *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*. ACM Press, 2010, pp. 211–220.

[8] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[9] R. Koymans, "Specifying real-time properties with metric temporal logic." *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990.

[10] Y. S. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *Tools and algorithms for the construction and analysis of systems*, ser. LNCS, vol. 6605. Springer, 2011, pp. 254–257.

[11] Y. S. R. Annapureddy and G. E. Fainekos, "Ant colonies for temporal logic falsification of hybrid systems," in *Proceedings of the 36th Annual Conference of IEEE Industrial Electronics*, 2010, pp. 91–96.

[12] S. Sankaranarayanan and G. Fainekos, "Falsification of temporal properties of hybrid systems using the cross-entropy method," in *ACM International Conference on Hybrid Systems: Computation and Control*, 2012.

[13] G. Rosu and K. Havelund, "Synthesizing dynamic programming algorithms from linear temporal logic formulae," RIACS, Tech. Rep., 2001.

[14] Q. Zhao, B. H. Krogh, and P. Hubbard, "Generating test inputs for embedded control systems," *IEEE Control Systems Magazine*, vol. Aug., pp. 49–57, 2003.

[15] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Falsification of ltl safety properties in hybrid systems," in *Proc. of the Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. LNCS, vol. 5505. Springer, 2009, pp. 368 – 382.

[16] A. Donze and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modelling and Analysis of Timed Systems*, ser. LNCS, vol. 6246. Springer, 2010.

[17] A. Rizk, G. Batt, F. Fages, and S. Soliman, "On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology," in *International Conference on Computational Methods in Systems Biology*, ser. LNCS, vol. 5307. Springer, 2008, pp. 251–268.

[18] P. Thati and G. Rosu, "Monitoring algorithms for metric temporal logic specifications," in *Runtime Verification*, ser. ENTCS, vol. 113. Elsevier, 2005, pp. 145–162.

[19] TaLiRo Tools. [Online]. Available: https://sites.google.com/a/asu.edu/s-taliro/

[20] A. Chutinan and K. R. Butts, "Dynamic analysis of hybrid system models for design validation," Ford Motor Company, Tech. Rep., 2002.

[21] B. I. Silva and B. H. Krogh, "Formal verification of hybrid systems using CheckMate: a case study," in *Proceedings of the American Control Conference*, vol. 3, Jun. 2000, pp. 1679 – 1683.