**Software Tools for Technology Transfer manuscript No.**
(will be inserted by the editor)

# Querying Parametric Temporal Logic Properties in Model Based Design

**Bardh Hoxha, Adel Dokhanchi, Georgios Fainekos**

Arizona State University
e-mail: {bhoxha,adokhanc,fainekos}@asu.edu

**Abstract.** One of the advantages of adopting the Model Based Development (MBD) process is that it enables testing and verification at early stages of development. However, it is often desirable to not only verify/falsify certain formal system specifications, but also to automatically explore the properties that the system satisfies. In this work, we present a framework that enables property exploration for Cyber-Physical Systems. Namely, given a parametric specification with multiple parameters, our solution can automatically infer the ranges of parameters for which the property holds/does not hold on the system. In this paper, we consider parametric specifications in Metric Temporal Logic (MTL). Using robust semantics for MTL, the parameter estimation problem can be converted into a Pareto optimization problem for which we can provide an approximate solution by utilizing stochastic optimization methods. We include solutions and algorithms for the exploration and visualization of multi-parameter specifications. The framework is demonstrated on an industrial size, high-fidelity engine model and examples from the related literature.

**Key words:** Metric Temporal Logic, Verification, Testing, Robustness, Multiple Parameter Estimation, Cyber-Physical Systems

## 1 Introduction

Testing, verification and validation of Cyber-Physical Systems (CPS) is a challenging problem. Prime examples of such systems are aircraft, cars and medical devices. The complexity in these systems arises mostly from the complex interactions between the numerous components, the controllers, and the physical environment, the plant. Many accidents [1,2] and recalls in the industry have reinforced the need for better methodologies in this area. In addition, general trends indicate that software complexity in CPS is going to increase in the future.

A recent shift in system development, aimed to alleviate some of the challenges, is the Model Based Design (MBD) paradigm. One of the benefits of MBD is that a significant amount of testing and verification of the system can be conducted in various stages of model development. As opposed to the traditional approach, where most of the analysis is conducted on the physical prototype of the system. Due to the importance of the problem, there has been a substantial level of research on testing and verification of models of embedded and hybrid systems (see [3] for an overview).

In [4], the authors propose an approach to support the testing and verification process in MBD. The paper provides a new method for testing embedded and hybrid systems against formal requirements which are defined in Metric Temporal Logic (MTL) [5]. Given a system and MTL specification, the method searches for operating conditions such that the MTL specification is not satisfied or, in other words, falsified. The authors utilize the concept of system robustness of MTL specifications [6] to turn the falsification problem into an optimization problem. The notion of the robustness metric enables system developers to measure by how far a system behavior is from failing to satisfy a requirement. This allows for the development of an automatic test case generator, which uses a stochastic optimization function to find operating conditions that falsify the system in terms of the MTL specifications.

The resulting optimization problem can be both non-linear and non-convex. To solve the problem, in [7,8,4], the authors present stochastic optimization techniques that solve the falsification problem with very good performance in both accuracy and number of simulations required.

In [9], the authors utilize this notion of robustness to explore and determine system properties. In more detail, given a parameterized MTL specification [10], where there is an unknown state or timing parameter, the authors find the range of values for the parameter such that the system is not satisfied.

In this work, we extend and generalize this method to enable multiple parameter estimation and analysis of parametric MTL specifications. We improve the efficiency of the existing algorithm and present a parameter estimation framework for MBD. Such an exploration framework would be of great value to the practitioner. The benefits are twofold. One, it allows for the analysis and development of specifications. In many cases, system requirements are not well formalized by the initial system design stages. Two, it allows for the analysis and exploration of system behavior. If a specification can be falsified, then it is natural to inquire for the range of parameter values that cause falsification.

The extension to multiple parameter estimation of MTL specifications allows practitioners to use this method with more complex specifications. However, as the number of parameters in the specification increases, so does the complexity of the resulting optimization problem. In the case of single parameter estimation, the solution of the problem is a one dimensional range. On the other hand, with multiple parameters, finding a solution to the problem becomes more challenging since, in a sense, the optimization problem is converted to a multi-objective optimization problem where the goal is to determine the Pareto front. To solve this problem, we present a method for effective exploration of the pareto front and provide a visualization method for the analysis of parameters. The algorithms presented in this work are incorporated in the testing a verification toolbox S-TaLiRo [11,12]. For an overview of the toolbox see [13]. Finally, we demonstrate our framework on a challenge problem from the industry on an industrial scale model and present experimental results on several benchmark problems. Even though our examples and case study are from the automotive domain, our results are general and can be applied in any application domain where Model Based Design (MBD) is utilized, e.g., medical devices [14–17]. The choice of examples from the automotive domain is preferable since everyone can relate to these examples.

**Summary of Contributions:**

- We extend and generalize the parameter estimation problem presented in [9].
- We provide an efficient solution to the problem of multiple parameter estimation.
- We present an algorithm to explore the Pareto front or parametric MTL with multiple parameters.
- We illustrate our method with an industrial size case study of a high-fidelity engine model.
- The algorithms presented in this work are publicly available through our software toolbox S-TaLiRo.

## 2 Problem Formulation

### *2.1 Preliminaries*

In the rest of the paper, we take a general approach to modeling real-time embedded systems that interact with physical systems that have non-trivial dynamics. A major source of complexity in the analysis of these systems arises from the interaction between the embedded system and the physical world.

We fix $N \subseteq \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers, to be a finite set of indexes for the finite representation of a system behavior. In the following, given two sets $A$ and $B$, $B^A$ denotes the set of all functions from $A$ to $B$. That is, for any $f \in B^A$ we have $f : A \to B$.

We consider a system $\Sigma$ as a mapping from a compact set of *initial operating conditions* $X_0$ and *input signals* $\mathbf{U} \subseteq U^N$ to *output signals* $Y^N$ and *timing* (or *sampling*) functions $\mathfrak{T} \subseteq \mathbb{R}_+^N$. Here, $U$ is a compact set of possible input values at each point in time (input space), $Y$ is the set of output values (output space), $\mathbb{R}$ is the set of real numbers and $\mathbb{R}_+$ the set of positive reals.

We impose three assumptions/restrictions on the systems that we consider:

1. The input signals (if any) must be parameterizable using a finite number of parameters. That is, there exists a function $\mathfrak{U}$ such that for any $u \in \mathbf{U}$, there exist two parameter vectors $\boldsymbol{\lambda} = [\lambda_1 \ \dots \ \lambda_m]^T \in \Lambda$, where $\Lambda$ is a compact set, and $t = [t_1 \ \dots \ t_m]^T \in \mathbb{R}_+^m$ such that $m << \max N$ and for all $i \in N$, $u(i) = \mathfrak{U}(\lambda, t)(i)$.
2. The output space $Y$ must be equipped with a generalized metric $\mathbf{d}$ which contains a subspace $Z$ equipped with a metric $d$.
3. For a specific initial condition $x_0$ and input signal $u$, there must exist a unique output signal $\mathbf{y}$ defined over the time domain $R$. That is, the system $\Sigma$ is deterministic.

Further details on the necessity and implications of the aforementioned assumptions can be found in [18].

Under Assumption 3, a system $\Sigma$ can be viewed as a function $\Delta_\Sigma : X_0 \times \mathbf{U} \to Y^N \times \mathfrak{T}$ which takes as an input an initial condition $x_0 \in X_0$ and an input signal $u \in \mathbf{U}$ and it produces as output a signal $\mathbf{y} : N \to Y$ (also referred to as *trajectory*) and a timing function $\tau : N \to \mathbb{R}_+$. The only restriction on the timing function $\tau$ is that it must be a monotonic function, i.e., $\tau(i) < \tau(j)$ for $i < j$. The pair $\mu = (\mathbf{y}, \tau)$ is usually referred to as a *timed state sequence*, which is a widely accepted model for reasoning about real time systems [19]. A timed state sequence can represent a computer simulated trajectory of a CPS or the sampling process that takes place when we digitally monitor physical systems. We remark that a timed state sequence can represent both the internal state of the software/hardware

(usually through an abstraction) and the state of the physical system. The set of all timed state sequences of a system $\Sigma$ will be denoted by $\mathcal{L}(\Sigma)$. That is,

$$\mathcal{L}(\Sigma) = \{(\mathbf{y}, \tau) \mid \exists x_0 \in X_0 \,.\, \exists u \in \mathbf{U} \,.\, (\mathbf{y}, \tau) = \Delta_\Sigma(x_0, u)\}.$$

Our high-level goal is to explore and infer properties that the system $\Sigma$ satisfies. We do so by observing system response (output signals) to particular input signals and initial conditions. We assume that the system designer has partial understanding about the properties that the system satisfies (or does not satisfy) and would like to be able to precisely determine these properties. In particular, we assume that the system developer can formalize the system properties in Metric Temporal Logic (MTL) [5], where some parameters are unknown. Such parameters could be unknown threshold values for the continuous state variables of the hybrid system or some unknown real time constraints.

Throughout the paper, we will consider two running examples. The first example consists of an automatic transmission model, and the second, consists of a hybrid non-linear time varying system.

**Example 1 (AT)** *We consider a slightly modified version of the Automatic Transmission model provided by Mathworks as a Simulink demo*[1]*. Further details on this example can be found in [20,18]. The only input u to the system is the throttle schedule, while the brake schedule is set simply to 0 for the duration of the simulation which is* $T = 30\,\text{sec}$*. The physical system has two continuous-time state variables which are also its outputs: the speed of the engine* $\omega$ *(RPM) and the speed of the vehicle* $v$*, i.e.,* $Y = \mathbb{R}^2$ *and* $\mathbf{y}(t) = [\omega(t)\ v(t)]^T$ *for all* $t \in [0, 30]$*. Initially, the vehicle is at rest at time 0, i.e.,* $X_0 = \{[0\ 0]^T\}$ *and* $x_0 = \mathbf{y}(0) = [0\ 0]^T$*. Therefore, the output trajectories depend only on the input signal u which models the throttle, i.e.,* $(\mathbf{y}, \tau) = \Delta_\Sigma(u)$*. The throttle at each point in time can take any value between 0 (fully closed) to 100 (fully open). Namely,* $u(i) \in U = [0, 100]$ *for each* $i \in N$*. The model also contains a Stateflow chart with two concurrently executing Finite State Machines (FSM) with 4 and 3 states, respectively. The FSM models the logic that controls the switching between the gears in the transmission system. We remark that the system is deterministic, i.e., under the same input u, we will observe the same output* $\mathbf{y}$*. In our previous work [18, 11, 7], on such models, we demonstrated how to falsify requirements like: "The vehicle speed v is always under 120km/h or the engine speed* $\omega$ *is always below 4500RPM." A falsifying system trajectory appears in Fig. 1 (Left).* ▲

**Example 2 (HS)** *We consider the following hybid time varying non-linear system. Let* $\mathbf{y}(t) = [\mathbf{y}_1(t)\ \mathbf{y}_2(t)]^T$ *we define the system as follows:*

***If*** $x_0 \in [-1, 1]^2 \backslash [0.85, 0.95]^2$ ***then***

$$\frac{d\mathbf{y}(t)}{dt} = \begin{bmatrix} \frac{d\mathbf{y}_1(t)}{dt} \\ \frac{d\mathbf{y}_2(t)}{dt} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1(t) - \mathbf{y}_2(t) + 0.1t \\ \mathbf{y}_2(t)\cos(2\pi\mathbf{y}_2(t)) - \mathbf{y}_1(t)\sin(2\pi\mathbf{y}_1(t)) + 0.1t \end{bmatrix}$$

***else***

$$\frac{d\mathbf{y}(t)}{dt} = \begin{bmatrix} \mathbf{y}_1(t) \\ -\mathbf{y}_1(t) + \mathbf{y}_2(t) \end{bmatrix}$$

*with initial condition* $\mathbf{y}(0) = x_0 \in X_0 = [-1, 1] \times [-1, 1]$*. Interesting requirements on this system would be "A trajectory of the system should never pass set* $[-1.6, -1.4]^2$ *or set* $[3.4, 3.6] \times [-1.6, -1.4]$*". A falsifying system trajectory appears in Fig. 1 (Right).* ▲

### *2.2 Parameter Estimation*

In this work, we provide answers to queries like "What is the shortest time that $\omega$ can exceed 3250 RPM" or "For how long can $\omega$ be below 4500 RPM". We can also answer queries to the relationships between parameters with regard to system falsification. For example, for the specification "Always the vehicle speed $v$ and engine speed $\omega$ need to be less than parameters $\lambda_1, \lambda_2$, respectively" then we could ask "If I increase/decrease $\lambda_1$ by a specific amount, how much do I have to increase/decrease $\lambda_2$ so that I satisfy the specification?".

Formally, we extend and generalize the problem presented in [9] to the following.

**Problem 1 (MTL Parameter Estimation)** *Given an MTL formula* $\phi[\boldsymbol{\theta}]$ *with a vector of unknown parameters* $\boldsymbol{\theta} \in \Theta = [\underline{\theta}, \overline{\theta}]$*, a system* $\Sigma$*, find the set* $\Psi = \{\theta^* \in \Theta \mid \Sigma \not\models \phi[\theta^*]\}$ *such that for any parameter* $\theta^*$ *in* $\Psi$ *the specification* $\phi[\theta^*]$ *does not hold on system* $\Sigma$*.*

In the rest of the paper, we refer to $\Psi$ as the parameter falsification domain. An approximate solution for Problem 1 was presented in [9] for the case where $\theta$ is a scalar. Here we extend the results to specifications with multiple parameters. In [9], the solution to the problem returned a parameter with which the falsifying set can be inferred since the parameter range is one dimensional. In the multiple parameter setting, we have a set of possible solutions which we need to explore. Ideally, by solving Problem 1, we would also like to have the property that for any $\boldsymbol{\zeta} \in \Theta - \Psi$, $\phi[\boldsymbol{\zeta}]$ holds on $\Sigma$, i.e., $\Sigma \models \phi[\boldsymbol{\zeta}]$. However, even for a given $\boldsymbol{\zeta}$, the problem of algorithmically computing whether $\Sigma \models \phi[\zeta]$ is undecidable for the classes of systems that we consider in this work [21].

An overview of our proposed solution to Problem 1 appears in Fig. 2. Given a model and PMTL specification, the sampler produces a point $x_0$ from the set of initial conditions, input signal $u$ and vector of estimated parameters $\boldsymbol{\lambda}$ for the PMTL specification. The

[1] Available at: `http://www.mathworks.com/help/simulink/examples/modeling-an-automatic-transmission-controller.html`

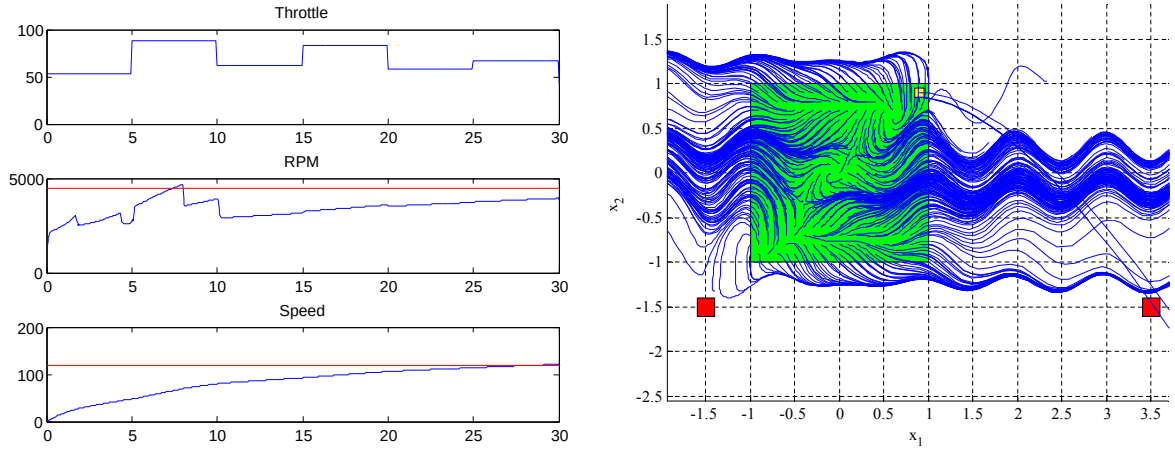


**Fig. 1.** Left: Example 1 (AT): A piecewise constant input signal $u$ parameterized with $\Lambda \in [0, 100]^6$ and $t = [0, 5, 10, 15, 20, 25]$ and the corresponding output signals that falsify the specification "The vehicle speed v is always under 120km/h or the engine speed is always below 4500RPM." Right: Example 2 (HS): Simulated trajectories of the hybrid system containing a trajectory that falisifes the specification "A trajectory should never pass set $[-1.6, -1.4]^2$ or set $[3.4, 3.6] \times [-1.6, -1.4]$". The green square indicates the set of possible initial conditions and the red squares indicate the bad regions which the system should not enter. The yellow region indicates the set of initial conditions where the location on the hybrid system changes.

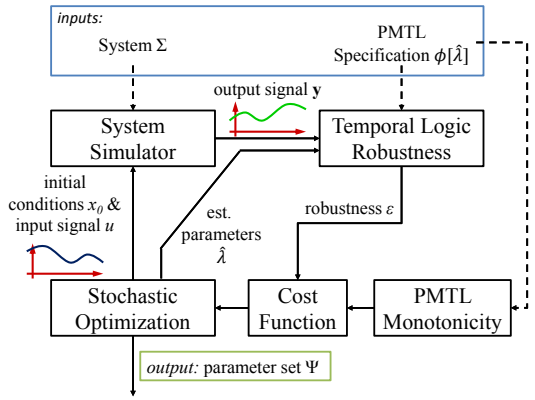


**Fig. 2.** Overview of the solution to Problem 1, the PMTL parameter estimation problem on CPS.

initial conditions and input signal are passed to the system simulator which returns an execution trace (output trajectory and timing function). The trace, in conjunction with the estimated parameters, is then analyzed by the MTL robustness analyzer which returns a robustness value. The robustness score computed is used by the stochastic sampler to decide on next intial conditions, inputs, and estimated parameters to utilize. The process terminates after a maximum number of tests or when no improvement on the parameter estimates has been made after a number of tests. As the number of parameters increases, so does the computational complexity of the problem. Therefore, for formulas with more than one parameter, we present a more efficient approach to explore the parameter falsification domain.

## 3 Robustness of Metric Temporal Logic Formulas

Metric Temporal Logic [5] enables reasoning over quantitative timing properties of boolean signals. In the following, we present MTL in Negation Normal Form (NNF) since this is needed for the presentation of the new results in Section 5. We denote the extended real number line by $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$.

**Definition 1 (Syntax of MTL in NNF)** *Let $\overline{\mathbb{R}}$ be the set of truth degree constants, AP be the set of atomic propositions and $\mathcal{I}$ be a non-empty non-singular interval of $\overline{\mathbb{R}}_{\geq 0}$. The set MTL of all well-formed formulas (wff) is inductively defined using the following rules:*

- *Terms: True ($\top$), false ($\bot$), all constants $r \in \overline{\mathbb{R}}$ and atomic propositions $p$, $\neg p$ for $p \in AP$ are terms.*
- *Formulas: if $\phi_1$ and $\phi_2$ are terms or formulas, then $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$, $\phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$ and $\phi_1 \mathcal{R}_{\mathcal{I}} \phi_2$ are formulas.*

The atomic propositions in our case label subsets of the output space $Y$. In other words, each atomic proposition is a shorthand for an arithmetic expression of the form $p \equiv g(y) \leq c$, where $g : Y \to \mathbb{R}$ and $c \in \mathbb{R}$. We define an observation map $\mathcal{O} : AP \to \mathcal{P}(Y)$ such that for each $p \in AP$ the corresponding set is $\mathcal{O}(p) = \{y \mid g(y) \leq c\} \subseteq Y$.

In the above definition, $\mathcal{U}_{\mathcal{I}}$ is the timed *until* operator and $\mathcal{R}_{\mathcal{I}}$ the timed *release* operator. 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 ($\mathcal{I} \neq \emptyset$) (and, practically speaking, non-singular ($\mathcal{I} \neq \{t\}$)). In the case where $\mathcal{I} = [0, +\infty)$, we remove the subscript $\mathcal{I}$ from the temporal operators, i.e., we just write $\mathcal{U}$ and $\mathcal{R}$. Also, we can define *eventually* ($\Diamond_{\mathcal{I}} \phi \equiv \top \mathcal{U}_{\mathcal{I}} \phi$) and *always* ($\Box_{\mathcal{I}} \phi \equiv \bot \mathcal{R}_{\mathcal{I}} \phi$).

Before proceeding to the actual definition of the robust semantics, we introduce some auxiliary notation. A metric space is a pair $(X, d)$ such that the topology of the set $X$ is induced by a metric $d$. Using a metric $d$, we can define the distance of a point $x \in X$ from a set

$S \subseteq X$. Intuitively, this distance is the shortest distance from $x$ to all the points in $S$. In a similar way, the depth of a point $x$ in a set $S$ is defined to be the shortest distance of $x$ from the boundary of $S$. Both the notions of distance and depth play a fundamental role in the definition of the robustness degree. The metrics and distances utilized in this work are covered in more detail in [6,18].

**Definition 2 (Signed Distance)** *Let $x \in X$ be a point, $S \subseteq X$ be a set and $d$ be a metric on $X$. Then, we define the Signed Distance from $x$ to $S$ to be*

$$\mathbf{Dist}_d(x,S) := \begin{cases} -\mathbf{dist}_d(x,S) := -\inf\{d(x,y) \mid y \in S\} \\ \qquad \textit{if } x \notin S \\ \mathbf{depth}_d(x,S) := \mathbf{dist}_d(x, X \backslash S) \\ \qquad \textit{if } x \in S \end{cases}$$

We utilize the extended definition of the supremum and infimum, i.e., $\sup \emptyset := -\infty$ and $\inf \emptyset := +\infty$.

We define the binary relation $\preceq$ on parameter vectors $\boldsymbol{\theta}, \boldsymbol{\theta'}$ such that $\boldsymbol{\theta} \preceq \boldsymbol{\theta'} \iff \forall i, \theta_i \leq \theta'_i$, where $i$ is the $i^{th}$ entry of the vector. MTL formulas are interpreted over timed state sequences $\mu$. In the past [6], we proposed multi-valued semantics for the MTL where the valuation function on the predicates takes values over the totally ordered set $\overline{\mathbb{R}}$ according to a metric $d$ operating on the output space $Y$. For this purpose, we let the valuation function be the depth (or the distance) of the current point of the signal $\mathbf{y}(i)$ in a set $\mathcal{O}(p)$ labeled by the atomic proposition $p$. Intuitively, this distance represents how robustly is the point $\mathbf{y}(i)$ within a set $\mathcal{O}(p)$. If this metric is zero, then even the smallest perturbation of the point can drive it inside or outside the set $\mathcal{O}(p)$, dramatically affecting membership.

For the purposes of the following discussion, we use the notation $[\![\phi]\!]$ to denote the robustness estimate with which the timed state sequence $\mu$ satisfies the specification $\phi$. Formally, the valuation function for a given formula $\phi$ is $[\![\phi]\!] : Y^N \times \mathfrak{T} \times N \to \overline{\mathbb{R}}$. In the definition below, we also use the following notation : for $Q \subseteq R$, the *preimage* of $Q$ under $\tau$ is defined as : $\tau^{-1}(Q) := \{i \in N \mid \tau(i) \in Q\}$.

**Definition 3 (Robustness Estimate [22])** *Let $\mu = (\mathbf{y}, \tau) \in \mathcal{L}(\Sigma)$, $r \in \overline{\mathbb{R}}$ and $i, j, k \in N$, then the robustness estimate of any formula MTL $\phi$ with respect to $\mu$ is recursively defined as follows*

$$[\![r]\!](\mu,i) := r \qquad [\![\top]\!](\mu,i) := +\infty \qquad [\![\bot]\!](\mu,i) := -\infty$$

$$[\![p]\!](\mu,i) := \mathbf{Dist}_d(\mathbf{y}(i), \mathcal{O}(p))$$

$$[\![\neg p]\!](\mu,i) := -\mathbf{Dist}_d(\mathbf{y}(i), \mathcal{O}(p))$$

$$[\![\phi_1 \vee \phi_2]\!](\mu,i) := \max([\![\phi_1]\!](\mu,i), [\![\phi_2]\!](\mu,i))$$

$$[\![\phi_1 \wedge \phi_2]\!](\mu,i) := \min([\![\phi_1]\!](\mu,i), [\![\phi_2]\!](\mu,i))$$

$$[\![\phi_1 \mathcal{U}_{\mathcal{I}} \phi_2]\!](\mu,i) := \sup_{j \in \tau^{-1}(\tau(i)+\mathcal{I})} \left(\min([\![\phi_2]\!](\mu,j), \inf_{i \leq k < j} [\![\phi_1]\!](\mathbf{y},k))\right)$$

$$[\![\phi_1 \mathcal{R}_{\mathcal{I}} \phi_2]\!](\mu,i) := \inf_{j \in \tau^{-1}(\tau(i)+\mathcal{I})} \left(\max([\![\phi_2]\!](\mu,j), \sup_{i \leq k < j} [\![\phi_1]\!](\mu,k))\right)$$

Recall that we use the extended definition of supremum and infimum. When $i = 0$, then we write $[\![\phi]\!](\mu)$.

The robustness of an MTL formula with respect to a timed state sequence can be computed using several existing algorithms [6,23,24].

If we consider the robustness estimate over systems, the resulting robustness landscape can be both non-linear and non-convex. In Fig. 3 we present the robusntess landscape for the two running examples, namely Examples 1 (AT), 2 (HS) on two specifications.

## 4 Parametric Metric Temporal Logic over Signals

In many cases, it is important to be able to describe an MTL specification with unknown parameters and then, infer the parameters that make the specification false. In [10], Asarin et al. introduced Parametric Signal Temporal Logic (PSTL) and presented two algorithms for computing approximations for parameters over a given signal. Here, we review some of the results in [10] while adapting them in the notation and formalism that we use in this paper.

**Definition 4 (Syntax of Parametric MTL)** *Let $\boldsymbol{\theta}$ be a parameter. The set of all well formed Parametric MTL (PMTL) formulas is the set of all well formed MTL formulas where for all $i$ each parameter $\theta_i$ either appears in an arithmetic expression, i.e., $p[\theta_i] \equiv g(y) \leq \theta_i$, or in the timing constraint of a temporal operator, i.e., $\mathcal{I}[\theta_i]$.*

We will denote a PMTL formula $\phi$ with parameters $\boldsymbol{\theta}$ by $\phi[\boldsymbol{\theta}]$. Given a vector of parameters $\boldsymbol{\theta} \in \Theta$, then the formula $\phi[\boldsymbol{\theta}]$ is an MTL formula. There is an implicit mapping from the vector of parameters $\boldsymbol{\theta}$ to the corresponding arithmetic expressions and temporal operators in the MTL formula.

Since the valuation function of an MTL formula is a composition of minimum and maximum operations quantified over time intervals, a formula $\phi[\theta]$, when $\theta$ is a scalar, is monotonic with respect to $\lambda$. When $\boldsymbol{\lambda}$ is a vector, then the valuation function is monotonic with respect to a priority function $f(\boldsymbol{\theta})$. The priority function will enable the system engineer to prioritize the optimization of some parameters over others by defining specific weights, or setting an optimization strategy such as optimizing the minimum, maximum, or norm of all parameters. The priority function will be defined in detail in the next section.

In the following, we present monotonicity results for single and multiple parameter PMTL formulas.

### *4.1 Single parameter PMTL formulas*

**Example 3 (AT)** *Consider the PMTL formula $\phi[\theta] = \Box_{[0,\theta]} p$ where $p \equiv (\omega \leq 3250)$. Given a timed state se-*

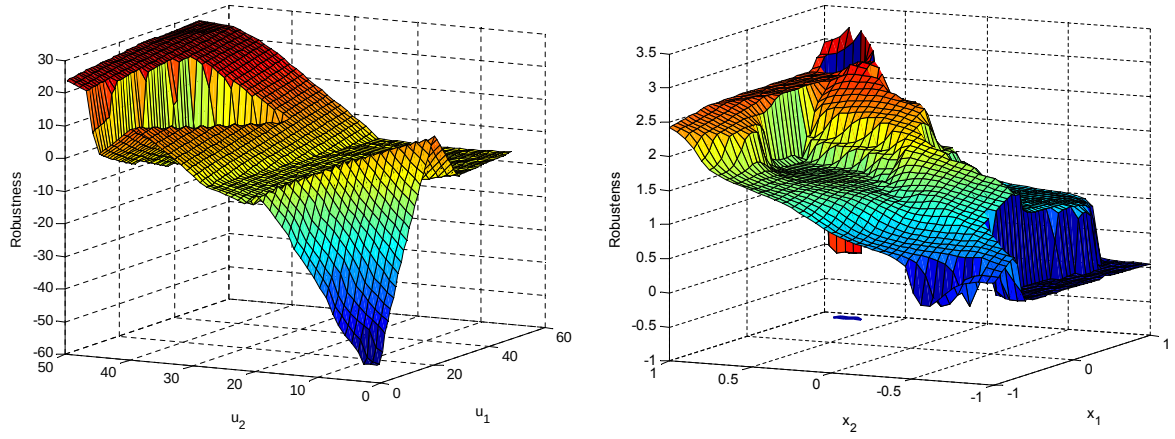


**Fig. 3.** Estimated robustness landscape for system specifications. Left: Example 1 (AT): $\phi_{AT} = \neg(\Diamond_{[0,30]}(v > 100) \wedge \Box(\omega \leq 4500)) \wedge \neg\Diamond_{[10,40]}\Box_{[0,5]}(60 < v \leq 80) \wedge \neg\Diamond_{[50,60]}\Box_{[0,3]}(v \leq 60)$. The input signal to the system is generated by linearly interpolating control points $u_1$, $u_2$; Right: Example 2 (HS): $\phi_{HS} = \Box_{[0,2]}\neg a \wedge \Box_{[0,2]}\neg b$, where $\mathcal{O}(a) = [-1.6, -1.4]^2$ and $\mathcal{O}(b) = [3.4, 3.6] \times [-1.6, -1.4]$. Here $x_1$ and $x_2$ are initial conditions for the hybrid system.

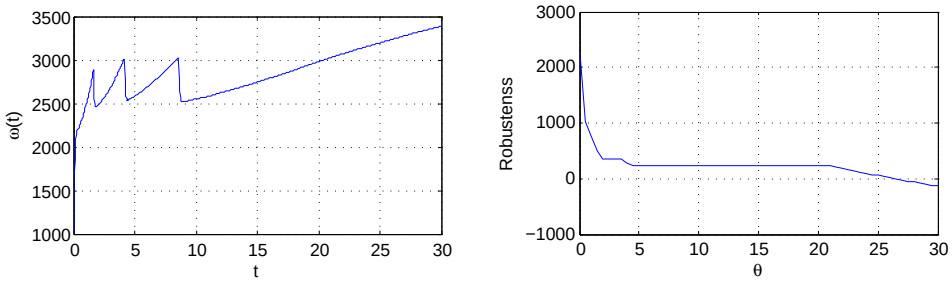


**Fig. 4.** Example 3. Left: Engine speed $\omega(t)$ for constant throttle $u(t) = 50$. Right: The estimated robustness of the specification $\Box_{[0,\theta]}(\omega \leq 3250)$ with respect to $\theta$.

*quence $\mu = (\mathbf{y}, \tau)$ with $\tau(0) = 0$, for $\theta_1 \leq \theta_2$, we have:*

$$[0, \theta_1] \subseteq [0, \theta_2] \implies \tau^{-1}([0, \theta_1]) \subseteq \tau^{-1}([0, \theta_2]).$$

*Therefore, by Definitions (2) (3) we have*

$$[\![\phi[\theta_1]]\!](\mu) = \inf_{i \in \tau^{-1}([0,\theta_1])}(-\mathbf{Dist}_d(\mathbf{y}(i), \mathcal{O}(p)))$$
$$\geq \inf_{i \in \tau^{-1}([0,\theta_2])}(-\mathbf{Dist}_d(\mathbf{y}(i), \mathcal{O}(p))) = [\![\phi[\theta_2]]\!](\mu).$$

*That is, the function $[\![\phi[\theta]]\!](\mu)$ is non-increasing with $\theta$. Intuitively, this relationship holds since by extending the value of $\theta$ in $\phi[\theta]$, it becomes just as or more difficult to satisfy the specification. See Fig. 4 for an example using an output trajectory from the system in Example 1.* ▲

The aforementioned example can be formalized in the following result.

**Lemma 1** *Consider a PMTL formula $\phi[\theta]$ such that it contains a subformula $\phi_1 Op_{\mathcal{I}[\theta]}\phi_2$ where $Op \in \{\mathcal{U}, \mathcal{R}\}$. Then, given a timed state sequence $\mu = (\mathbf{y}, \tau)$, for $\theta_1$, $\theta_2 \in \overline{\mathbb{R}}_{\geq 0}$, such that $\theta_1 \leq \theta_2$, and for $i \in N$, we have:*

- *if (i) $Op = \mathcal{U}$ and $\sup \mathcal{I}(\theta) = \theta$ or (ii) $Op = \mathcal{R}$ and $\inf \mathcal{I}(\theta) = \theta$, then $[\![\phi[\theta_1]]\!](\mu, i) \leq [\![\phi[\theta_2]]\!](\mu, i)$, i.e., the function $[\![\phi[\theta]]\!](\mu, i)$ is non-decreasing with respect to $\theta$, and*
- *if (i) $Op = \mathcal{R}$ and $\sup \mathcal{I}(\theta) = \theta$ or (ii) $Op = \mathcal{U}$ and $\inf \mathcal{I}(\theta) = \theta$, then $[\![\phi[\theta_1]]\!](\mu, i) \geq [\![\phi[\theta_2]]\!](\mu, i)$, i.e., the function $[\![\phi[\theta]]\!](\mu, i)$ is non-increasing with respect to $\theta$.*

*Proof (sketch).* The proof is by induction on the structure of the formula and it is similar to the proofs that appeared in [6].

For completeness of the presentation, we consider the case $[\![\phi_1 \mathcal{U}_{\langle \alpha,\theta \rangle}\phi_2]\!](\mu, i)$, where $\langle \in \{[, (\}$ and $\rangle \in \{], )\}$. The other cases are either similar or they are based on the monotonicity of the max and min operators. Let $\theta_1 \leq \theta_2$, then we want to show that:

$$[\![\phi_1 \mathcal{U}_{\langle \alpha,\theta_1 \rangle}\phi_2]\!](\mu, i) \leq [\![\phi_1 \mathcal{U}_{\langle \alpha,\theta_2 \rangle}\phi_2]\!](\mu, i) \qquad (1)$$

To show that (1) holds, we utilize the robust semantics for MTL given in Definition 3, namely (3) and observe that:

$$[\![\phi_1 \mathcal{U}_{\langle \alpha,\theta_2 \rangle}\phi_2]\!](\mu, i) =$$
$$\sup_{j \in \tau^{-1}(\tau(i)+\langle \alpha,\theta_2 \rangle)} \big(\min([\![\phi_2]\!](\mu, j), \inf_{i \leq k < j}[\![\phi_1]\!](\mathbf{y}, k))\big) =$$
$$\max\left(\sup_{j \in \tau^{-1}(\tau(i)+\langle \alpha,\theta_1 \rangle)} \big(\min([\![\phi_2]\!](\mu, j), \inf_{i \leq k < j}[\![\phi_1]\!](\mathbf{y}, k))\big),\right.$$

$$\sup_{j\in\tau^{-1}(\tau(i)+\overline{\langle}\theta_1,\theta_2\rangle)}\Big(\min([\![\phi_2]\!](\mu,j),\inf_{i\le k<j}[\![\phi_1]\!](\mathbf{y},k))\Big)\Big) =$$
$$\max\Big([\![\phi_1\,\mathcal{U}_{\langle\alpha,\theta_1\rangle}\phi_2]\!](\mu,i),[\![\phi_1\,\mathcal{U}_{\overline{\langle}\theta_1,\theta_2\rangle}\phi_2]\!](\mu,i)\Big)\ge$$
$$[\![\phi_1\,\mathcal{U}_{\langle\alpha,\theta_1\rangle}\phi_2]\!](\mu,i)\qquad\square$$

where $\overline{\langle}\in\{[,(\}$ such that $\langle\alpha,\theta_1\rangle\cap\overline{\langle}\theta_1,\theta_2\rangle=\emptyset$ and $\langle\alpha,\theta_1\rangle\cup\overline{\langle}\theta_1,\theta_2\rangle=\langle\alpha,\theta_2\rangle$.

In the following, we derive similar results when the parameter appears in the numerical expression of the atomic proposition.

**Lemma 2** *Consider a PMTL formula $\phi[\theta]$ such that it contains a parametric atomic proposition $p[\theta]$ in a subformula. Then, given a timed state sequence $\mu=(\mathbf{y},\tau)$, for $\theta_1$, $\theta_2\in\overline{\mathbb{R}}_{\ge0}$, such that $\theta_1\le\theta_2$, and for $i\in N$, we have:*

- *if $p[\theta]\equiv g(x)\le\theta$, then $[\![\phi[\theta_1]]\!](\mu,i)\le[\![\phi[\theta_2]]\!](\mu,i)$, i.e., the function $[\![\phi[\theta]]\!](\mu,i)$ is non-decreasing with respect to $\theta$, and*
- *if $p[\theta]\equiv g(x)\ge\theta$, then $[\![\phi[\theta_1]]\!](\mu,i)\ge[\![\phi[\theta_2]]\!](\mu,i)$, i.e., the function $[\![\phi[\theta]]\!](\mu,i)$ is non-increasing with respect to $\theta$.*

*Proof (sketch).* The proof is by induction on the structure of the formula and it is similar to the proofs that appeared in [6]. For completeness of the presentation, we consider the base case $[\![p[\lambda]]\!](\mu,i)$. Let $\theta_1\le\theta_2$, then $\mathcal{O}(p[\theta_1])\subseteq\mathcal{O}(p[\theta_2])$. We will only present the case for which $\mathbf{y}(i)\notin\mathcal{O}(p[\theta_2])$. We have:

$$\mathcal{O}(p[\theta_1])\subseteq\mathcal{O}(p[\theta_2])\implies$$
$$\mathbf{dist}_d(\mathbf{y}(i),\mathcal{O}(p[\theta_1]))\ge\mathbf{dist}_d(\mathbf{y}(i),\mathcal{O}(p[\theta_2]))\implies$$
$$\mathbf{Dist}_d(\mathbf{y}(i),\mathcal{O}(p[\theta_1]))\le\mathbf{Dist}_d(\mathbf{y}(i),\mathcal{O}(p[\theta_2]))\implies$$
$$[\![p[\theta_1]]\!](\mu,i)\le[\![p[\theta_2]]\!](\mu,i)\qquad\square$$

### *4.2 Multiple parameter PMTL formulas*

Next, we extend the result for multiple parameters.

**Example 4 (AT)** *Consider the PMTL formula $\phi[\boldsymbol{\lambda}]=\neg(\diamondsuit_{[0,\lambda_1]}\ q\wedge\square p[\lambda_2])$ where $\boldsymbol{\lambda}=[\lambda_1,\lambda_2]$, $p[\lambda_2]\equiv(\omega\le\lambda_2)$ and $q\equiv(v\ge100)$. Given a timed state sequence $\mu=(\mathbf{y},\tau)$ with $\tau(0)=0$, for two vectors of parameters $\boldsymbol{\theta},\boldsymbol{\theta'}\in\mathbb{R}^2$ where $\boldsymbol{\theta}\preceq\boldsymbol{\theta'}$, we have:*

$$\theta_2\le\theta_2'\implies\mathcal{O}(p[\theta_2])\subseteq\mathcal{O}(p[\theta_2'])\implies$$
$$\mathbf{Dist}_d(\mathbf{y},\mathcal{O}(p[\theta_2]))\le\mathbf{Dist}_d(\mathbf{y},\mathcal{O}(p[\theta_2']))\implies$$
$$-\mathbf{Dist}_d(\mathbf{y}(i),\mathcal{O}(p[\theta_2]))\ge-\mathbf{Dist}_d(\mathbf{y}(i),\mathcal{O}(p[\theta_2'])) \quad (2)$$

$$\theta_1\le\theta_1'\implies[0,\theta_1]\subseteq[0,\theta_1']\implies$$
$$\tau^{-1}([0,\theta_1])\subseteq\tau^{-1}([0,\theta_1']) \quad (3)$$

*Therefore, by (2) and (3) we obtain:*

$$[\![\phi[\theta]]\!](\mu)=\inf_{i\in\tau^{-1}([0,\theta_1])}(-\mathbf{Dist}_d(\mathbf{y}(i),\mathcal{O}(p[\theta_2])))$$
$$\ge\inf_{i\in\tau^{-1}([0,\theta_1'])}(-\mathbf{Dist}_d(\mathbf{y}(i),\mathcal{O}(p[\theta_2'])))=[\![\phi[\theta']]\!](\mu)$$

*That is, the function $[\![\phi[\theta]]\!](\mu)$ is non-increasing for all $\boldsymbol{\theta}$ for which the relation $\preceq$ holds. The robustness landscape of two parameters over constant input is presented in Fig. 5.* ▲

**Example 5 (AT)** *Consider the PMTL formula $\phi[\boldsymbol{\lambda}]=\square(p[\lambda_1]\wedge q[\lambda_2])$ where $p[\lambda_1]\equiv(v\le\lambda_1)$ and $q[\lambda_2]\equiv(\omega\le\lambda_2)$. Given a timed state sequence $\mu=(\mathbf{y},\tau)$ with $\tau(0)=0$, for two vectors of parameters $\boldsymbol{\theta},\boldsymbol{\theta'}$ where $\boldsymbol{\theta}\preceq\boldsymbol{\theta'}$, we have:*

$$\mathcal{O}(p[\theta_1])\subseteq\mathcal{O}(p[\theta_1'])\implies$$
$$\mathbf{Dist}_d(\mathcal{O}(p[\theta_1]))\le\mathbf{Dist}_d(\mathcal{O}(p[\theta_1']))\implies$$
$$[\![p[\theta_1]]\!](\mu,i)\le[\![p[\theta_1']]\!](\mu,i)\ \text{and}\ \mathcal{O}(q[\theta_2])\subseteq\mathcal{O}(q[\theta_2'])\implies$$
$$\mathbf{Dist}_d(\mathcal{O}(p[\theta_2]))\le\mathbf{Dist}_d(\mathcal{O}(p[\theta_2']))\implies$$
$$[\![q[\theta_2]]\!](\mu,i)\le[\![q[\theta_2']]\!](\mu,i)$$

*Therefore, $[\![\phi[\boldsymbol{\theta}]]\!](\mu)\le[\![\phi[\boldsymbol{\theta'}]]\!](\mu)$. That is, the function $[\![\phi[\theta]]\!](\mu)$ is non-decreasing for all $\theta$ for which the relation $\preceq$ holds. Figure 5 presents the robustness landscape of two parameters over constant input.* ▲

Now we are in position to state the main monotonicity theorem for multiple parameters. We remark that for convenience we define the parametric subformulas over all the possible parameters even though only some of them are used in each subformula.

**Theorem 1.** *Consider a PMTL formula $\phi[\boldsymbol{\theta}]$, where $\boldsymbol{\theta}$ is a vector of parameters, such that $\phi[\boldsymbol{\theta}]$ contains temporal subformulas $\phi[\boldsymbol{\theta}]=\phi_1[\boldsymbol{\theta}]Op_{\mathcal{I}[\theta_s]}\phi_2[\boldsymbol{\theta}]$, $Op\in\{\mathcal{U},\mathcal{R}\}$, or propositional subformulas $\phi[\boldsymbol{\theta}]=p[\boldsymbol{\theta}]$. Then, given a timed state sequence $\mu=(\mathbf{y},\tau)$, for $\boldsymbol{\theta}$, $\boldsymbol{\theta'}\in\overline{\mathbb{R}}^n_{\ge0}$, such that $\boldsymbol{\theta}\preceq\boldsymbol{\theta'}$, where $1\le j\le n$, and for $i\in N$, we have:*

- *if for all such subformulas (i) $Op=\mathcal{U}$ and $\sup\mathcal{I}(\theta_s)=\lambda_s$ or (ii) $Op=\mathcal{R}$ and $\inf\mathcal{I}(\theta_s)=\theta_s$ or (iii) $p[\boldsymbol{\theta}]\equiv g(x)\le\boldsymbol{\theta}$, then $[\![\phi[\boldsymbol{\theta}]]\!](\mu,i)\le[\![\phi[\boldsymbol{\theta'}]]\!](\mu,i)$, i.e., the function $[\![\phi[\boldsymbol{\lambda}]]\!](\mu,i)$ is nondecreasing with respect to $\boldsymbol{\lambda}$, and*
- *if for all such subformulas (i) $Op=\mathcal{R}$ and $\sup\mathcal{I}(\theta_s)=\theta_s$ or (ii) $Op=\mathcal{U}$ and $\inf\mathcal{I}(\theta_s)=\lambda_s$ or (iii) $p[\boldsymbol{\lambda}]\equiv g(x)\ge\boldsymbol{\lambda}$, then $[\![\phi[\boldsymbol{\theta}]]\!](\mu,i)\ge[\![\phi[\boldsymbol{\theta'}]]\!](\mu,i)$, i.e., the function $[\![\phi[\boldsymbol{\lambda}]]\!](\mu,i)$ is non-increasing with respect to $\boldsymbol{\lambda}$.*

*Proof (sketch).* The proof is by induction on the structure of the formula. The base case is given by Lemmas 1 and 2.

Consider the first case where $\phi[\boldsymbol{\lambda}]=\phi_1[\boldsymbol{\lambda}]\,\mathcal{U}_{\mathcal{I}[\lambda_s]}\phi_2[\boldsymbol{\lambda}]$. Let $\boldsymbol{\theta},\boldsymbol{\theta'}\in\overline{\mathbb{R}}^n_{\ge0}$, where $\boldsymbol{\theta}\preceq\boldsymbol{\theta'}$. Let $i,j,k\in N$. Then $\mathcal{I}[\theta_s]\subseteq\mathcal{I}[\theta_s']$ and, for all $j$, by the induction hypothesis we have

$$\min([\![\phi_2[\boldsymbol{\theta}]]\!](\mu,j))\le\min([\![\phi_2[\boldsymbol{\theta'}]]\!](\mu,j)) \quad (4)$$

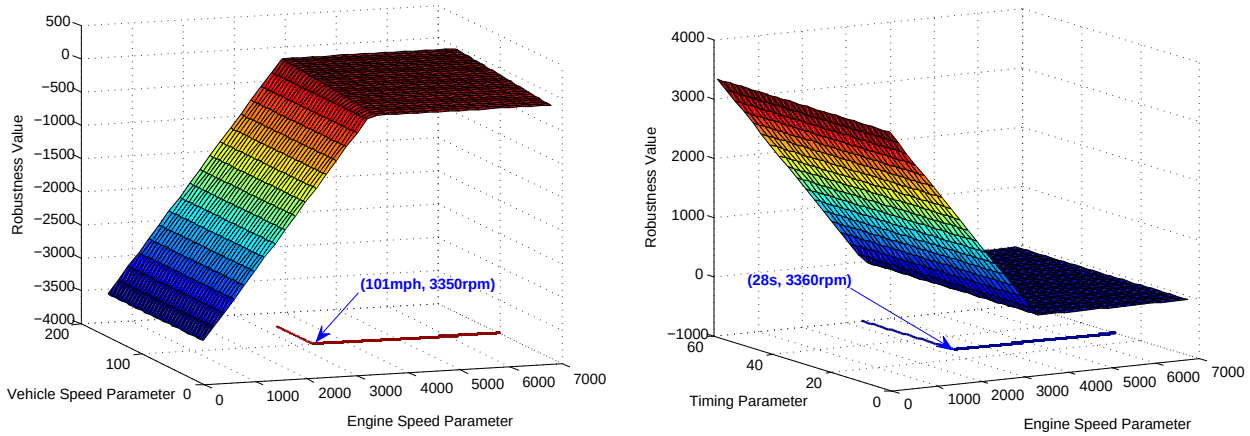


**Fig. 5.** Left: Example 4: Estimated robustness landscape for varying parameters for engine and vehicle speed for constant throttle $u(t) = 50$. Right: Example 5: Robustness landscape for varying parameters for timing parameter and engine speed for constant throttle $u(t) = 50$. The contour line shows the intersection of the robustness landscape with the zero level set.

For all k, by the induction hypothesis we have

$$\inf_{i\le k<j}(\llbracket\phi_1[\boldsymbol{\theta}]\rrbracket(\mathbf{y},k))) \le \inf_{i\le k<j}(\llbracket\phi_1[\boldsymbol{\theta'}]\rrbracket(\mathbf{y},k))) \quad (5)$$

Then by (4) and (5) we have

$$\llbracket\phi[\boldsymbol{\theta}]\rrbracket(\mu,i) = \llbracket\phi_1[\boldsymbol{\theta}]\,\mathcal{U}_{\mathcal{I}[\theta_s]}\phi_2[\boldsymbol{\theta}]\rrbracket(\mu,i) =$$
$$\sup_{j\in\tau^{-1}(\tau(i)+\mathcal{I}[\theta_s])}\Big(\min(\llbracket\phi_2[\boldsymbol{\theta}]\rrbracket(\mu,j), \inf_{i\le k<j}\llbracket\phi_1[\boldsymbol{\theta}]\rrbracket(\mathbf{y},k))\Big)$$
$$\le \sup_{j\in\tau^{-1}(\tau(i)+\mathcal{I}[\theta'_s])}\Big(\min(\llbracket\phi_2[\boldsymbol{\theta'}]\rrbracket(\mu,j),$$
$$\inf_{i\le k<j}\llbracket\phi_1[\boldsymbol{\theta'}]\rrbracket(\mathbf{y},k))\Big) = \llbracket\phi_1[\boldsymbol{\theta'}]\,\mathcal{U}_{\mathcal{I}[\theta'_s]}\phi_2[\boldsymbol{\theta'}]\rrbracket(\mu,i) =$$
$$\llbracket\phi[\boldsymbol{\theta'}]\rrbracket(\mu,i)$$

Therefore,

$$\llbracket\phi[\boldsymbol{\theta}]\rrbracket(\mu,i) \le \llbracket\phi[\boldsymbol{\theta'}]\rrbracket(\mu,i) \qquad \square$$

## 5 Temporal Logic Parameter Bound Computation

The notion of robustness of temporal logics will enable us to pose the parameter estimation problem as an optimization problem. In order to solve the resulting optimization problem, falsification methods and S-TaLiRo can be utilized in order to estimate $\Theta^*$ for Problem 1.

As described in the previous section, the parametric robustness functions that we are considering are monotonic with respect to the search parameters. Therefore, if we are searching for a parameter vector over an interval $\boldsymbol{\Theta} = [\underline{\boldsymbol{\theta}}, \overline{\boldsymbol{\theta}}]$, where $\underline{\boldsymbol{\theta}} = [\underline{\theta}_1, \underline{\theta}_2, ..., \underline{\theta}_n]^T$ and $\overline{\boldsymbol{\theta}} = [\overline{\theta}_1, \overline{\theta}_2, ..., \overline{\theta}_n]^T$, we know that $\boldsymbol{\Theta}^*$ is going to be either of the form $[\underline{\boldsymbol{\theta}}, \boldsymbol{\theta}^*]$ or $[\boldsymbol{\theta}^*, \overline{\boldsymbol{\theta}}]$. In other words, depending on the structure of the parametric formula $\phi$, we are either trying to minimize or maximize a function $f$ of $\boldsymbol{\theta}$ such that for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}^*$, we have $\llbracket\phi[\boldsymbol{\theta}]\rrbracket(\Sigma) \le 0$.

**Example 6 (AT)** *Let us consider again the automotive transmission example and the specification* $\phi[\theta] = \square_{[0,\theta]}p$ *where* $p \equiv (\omega \le 4500)$. *The specification robustness* $\llbracket\phi[\theta]\rrbracket(\Delta_\Sigma(u))$ *as a function of* $\theta$ *and the input* $u$ *appears in Fig. 6 for constant input signals. The creation of the graph required* $100 \times 30 = 3,000$ *tests. The contour under the surface indicates the zero level set of the robustness surface, i.e., the* $\theta$ *and* $u$ *values for which we get* $\llbracket\phi[\theta]\rrbracket(\Delta_\Sigma(u)) = 0$. *From the graph, we can infer that* $\theta^* \approx 2.8$ *and that for any* $\theta \in [2.8, 30]$, *we have* $\llbracket\phi[\theta]\rrbracket(\Sigma) \le 0$. *The approximate value of* $\theta^*$ *is an estimate based on the granularity of the grid that we used to plot the surface.* ▲

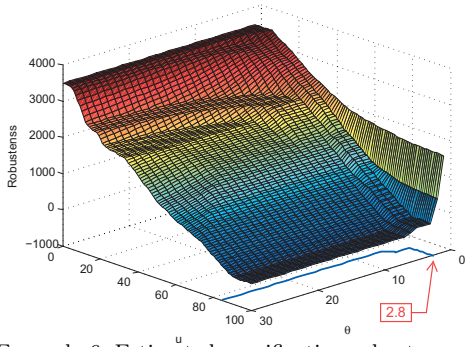


**Fig. 6.** Example 6: Estimated specification robustness as a function of parameter $\theta$ and input $u$ for specification $\phi[\theta] = \square_{[0,\theta]}(\omega \le 4500)$.

In summary, in order to solve Problem 1, we would have to solve the following optimization problem:

$$\begin{aligned} &\text{optimize} && f(\theta) \\ &\text{subject to} && \theta \in \Theta \text{ and} \\ & && \llbracket\phi[\theta]\rrbracket(\Sigma) = \min_{\mu\in\mathcal{L}_\tau(\Sigma)}\llbracket\phi[\theta]\rrbracket(\mu) \le 0 \end{aligned} \quad (6)$$

Where $f : \mathbb{R}^n \to \mathbb{R}$ is a monotonic function, i.e. for two vector parameter values $\theta$, $\theta'$, if $\theta \preceq \theta'$ and $\forall j\ \theta_j \ge 0$ then $f(\theta) \le f(\theta')$.

However, $\llbracket\phi[\theta]\rrbracket(\Sigma)$ neither can be computed using reachability analysis algorithms nor is known in closed form for the systems that we are considering. Therefore, we will have to compute an under-approximation of $\Theta^*$.

Our focus will be to formulate an optimization problem that can be solved using stochastic search methods. In particular, we will reformulate optimization problem (6) into a new one where the constraints due to the specification are incorporated into the cost function:

$$\text{optimize}_{\theta\in\Theta}\left(f(\theta)+\begin{cases}\gamma \pm [\![\phi[\theta]]\!](\Sigma) \\ \quad \text{if } [\![\phi[\theta]]\!](\Sigma)\geq 0 \\ 0 \quad \text{otherwise}\end{cases}\right) \tag{7}$$

where the sign ($\pm$) and the parameter $\gamma$ depend on whether the problem is a maximization or a minimization problem. The parameter $\gamma$ must be properly chosen so that the optimizer of problem (7) is in $\Theta$ if and only if $[\![\phi[\theta]]\!](\Sigma)\leq 0$. Therefore, if the problem in Eq. (6) is feasible, then the optimum of equations (6) and (7) is the same.

### 5.1 Non-increasing Robustness Functions

First, we consider the case of non-increasing robustness functions $[\![\phi[\theta]]\!](\Sigma)$ with respect to the search variable $\theta$. In this case, the optimization problem is a minimization problem.

To see why this is the case, assume that $[\![\phi[\theta_M]]\!](\Sigma)\leq 0$. Since for $\theta\leq\theta_M$, we have $[\![\phi[\theta]]\!](\Sigma)\geq[\![\phi[\theta_M]]\!](\Sigma)$, we need to find the minimum $\theta$ such that we still have $[\![\phi[\theta]]\!](\Sigma)\leq 0$. That $\theta$ value will be the desired $\theta^*$ since for all $\theta'\in[\theta^*,\theta_M]$, we will have $[\![\phi[\theta']]\!](\Sigma)\leq 0$.

We will reformulate the problem of Eq. (7) so that we do not have to solve two separate optimization problems. From (7), we have:

$$\begin{aligned}&\min_{\theta\in\Theta}\left(f(\theta)+\begin{cases}\gamma+\min_{\mu\in\mathcal{L}_\tau(\Sigma)}[\![\phi[\theta]]\!](\mu) \\ \quad \text{if } \min_{\mu\in\mathcal{L}_\tau(\Sigma)}[\![\phi[\theta]]\!](\mu)\geq 0 \\ 0 \quad \text{otherwise}\end{cases}\right)= \\ &=\min_{\theta\in\Theta}\left(f(\theta)+\min_{\mu\in\mathcal{L}_\tau(\Sigma)}\begin{cases}\gamma+[\![\phi[\theta]]\!](\mu) \\ \quad \text{if } [\![\phi[\theta]]\!](\mu)\geq 0 \\ 0 \quad \text{otherwise}\end{cases}\right)= \\ &=\min_{\theta\in\Theta}\min_{\mu\in\mathcal{L}_\tau(\Sigma)}\left(f(\theta)+\begin{cases}\gamma+[\![\phi[\theta]]\!](\mu) \\ \quad \text{if } [\![\phi[\theta]]\!](\mu)\geq 0 \\ 0 \quad \text{otherwise}\end{cases}\right)\end{aligned} \tag{8}$$

The previous discussion is formalized in the following result.

**Proposition 1** *Let $\theta^*$ be the parameters and $\mu^*$ be the system trajectory returned by an optimization algorithm that is applied to the problem in Eq. (8). If $[\![\phi[\theta^*]]\!](\mu^*)\leq 0$, then for all $\theta\in\Theta^*=[\theta^*,\theta_M]$, we have $[\![\phi[\theta]]\!](\Sigma)\leq 0$.*

*Proof.* If $[\![\phi[\theta^*]]\!](\mu^*)\leq 0$, then $[\![\phi[\theta^*]]\!](\Sigma)\leq 0$. Since $[\![\phi[\theta]]\!](\Sigma)$ is non-increasing with respect to $\theta$, then for all $\theta\in[\theta^*,\theta_M]$, we also have $[\![\phi[\theta]]\!](\Sigma)\leq 0$.

**Proposition 2** *If $f(\theta)=\|\theta\|$ and the robustness function is non-increasing, then $\gamma=\|\overline{\theta}\|$ is a valid choice for parameter $\gamma$.*

*Proof.* The interesting case to prove here is when $\theta\in[\underline{\theta},\theta^*]$ such that $[\![\phi[\theta]]\!](\Sigma)\geq 0$ and $\theta'\in[\theta^*,\overline{\theta}]$ such that $[\![\phi[\theta']]\!](\Sigma)<0$. In this case

$$\begin{gathered}\gamma=\|\overline{\theta}\|\geq\|\theta'\|\geq\|\theta\| \text{ and } [\![\phi[\theta]]\!](\Sigma)\geq 0 \implies \\ \|\theta\|+\gamma+[\![\phi[\theta]]\!](\Sigma)\geq\|\theta'\|\end{gathered}$$

Therefore, if the problem in Eq. (6) is feasible, then the optimum of equations (6) and (7) is the same.

Since we are utilizing stochastic optimization methods [7,11,8,4] to solve problem (8), if $[\![\phi[\theta^*]]\!](\mu^*)>0$, then we cannot infer that the system is correct for all parameter values in $\Theta$.

**Example 7 (AT)** *Using Eq. (8) as a cost function, we can now compute a parameter for Example 6 using* S-TaLiRo *[11, 12]. In particular, using Simulated Annealing as a stochastic optimization function,* S-TaLiRo *returns $\theta^*\approx 2.45$ as optimal parameter for constant input $u(t)=99.81$. The corresponding temporal logic robustness for the specification $\Box_{[0,2.45]}(\omega\leq 4500)$ is $-0.0445$. The number of tests performed for this example was 500 and, potentially, the accuracy of estimating $\theta^*$ can be improved if we increase the maximum number of tests. However, based on several tests the algorithm converges to a good solution within 200 tests.* ▲

**Example 8 (HS)** *Let us consider the specification $\phi[\boldsymbol{\theta}]=\Box_{[0,\theta_1]}\neg a$ where $\mathcal{O}(a)=[1.5,1.6+\theta_2]\times[1,1.1+\theta_3]$ on our hybrid system running example. Here, the timing parameter $\theta_1\in[0,3]$ and state parameters $\theta_2,\theta_3\in[0,0.5]$. The ranges for the parameters are chosen based on prior knowledge and experience about the system. The parameter estimation algorithm from* S-TaLiRo *returns $\theta_1^*=4.449$, $\theta_2^*=0.187$, and $\theta_3^*=0.371$ after running 1000 tests on the system. The returned parameters guarantee that the system does not satisy the specification for all parameters $\boldsymbol{\lambda}$ where $\boldsymbol{\theta}^*\preceq\boldsymbol{\lambda}$.* ▲

### 5.2 Non-decreasing Robustness Functions

The case of non-decreasing robustness functions is symmetric to the case of non-increasing robustness functions. In particular, the optimization problem is a maximization problem. We will reformulate the problem of Eq. (7) so that we do not have to solve two separate optimization problems. From (7), we have:

$$\begin{aligned}&\max_{\theta\in\Theta}\left(f(\theta)+\begin{cases}\gamma-\max_{\mu\in\mathcal{L}_\tau(\Sigma)}[\![\phi[\theta]]\!](\mu) \\ \quad \text{if } \max_{\mu\in\mathcal{L}_\tau(\Sigma)}[\![\phi[\theta]]\!](\mu)\geq 0 \\ 0 \quad \text{otherwise}\end{cases}\right)= \\ &=\max_{\theta\in\Theta}\left(f(\theta)+\max_{\mu\in\mathcal{L}_\tau(\Sigma)}\begin{cases}\gamma-[\![\phi[\theta]]\!](\mu) \\ \quad \text{if } -[\![\phi[\theta]]\!](\mu)\leq 0 \\ 0 \quad \text{otherwise}\end{cases}\right)= \\ &=\max_{\theta\in\Theta}\max_{\mu\in\mathcal{L}_\tau(\Sigma)}\left(f(\theta)+\begin{cases}\gamma-[\![\phi[\theta]]\!](\mu) \\ \quad \text{if } [\![\phi[\theta]]\!](\mu)\geq 0 \\ 0 \quad \text{otherwise}\end{cases}\right)\end{aligned} \tag{9}$$

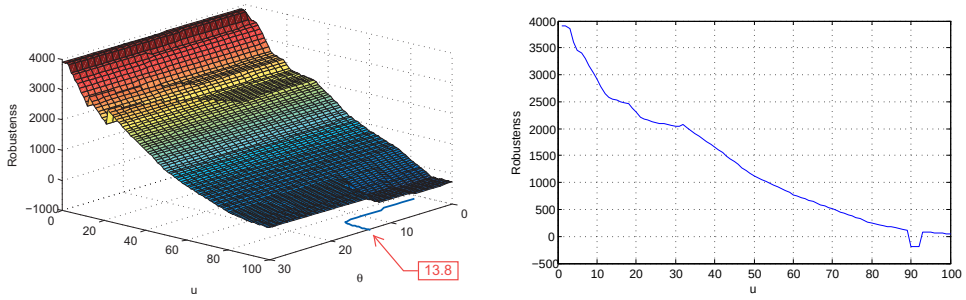


**Fig. 7.** Example 9. Left: Specification robustness as a function of the parameter $\theta$ and the input $u$. Right: The robustness function $[\![\Box_{[12.59,30]}(\omega \leq 4500)]\!](\Delta_\Sigma(u))$.

The previous discussion is formalized in the following result.

**Proposition 3** *Let $\theta^*$ be the parameters and $\mu^*$ be the system trajectory returned by an optimization algorithm that is applied to the problem in Eq. (9). If $[\![\phi[\theta^*]]\!](\mu^*) \leq 0$, then for all $\theta \in \Theta^* = [\theta_m, \theta^*]$, we have $[\![\phi[\theta]]\!](\Sigma) \leq 0$.*

*Proof.* If $[\![\phi[\theta^*]]\!](\mu^*) \leq 0$, then $[\![\phi[\theta^*]]\!](\Sigma) \leq 0$. Since $[\![\phi[\theta]]\!](\Sigma)$ is non-decreasing with respect to $\theta$, then for all $\theta \in [\theta_m, \theta^*]$, we also have $[\![\phi[\theta]]\!](\Sigma) \leq 0$.

**Proposition 4** *If $f(\theta) = \|\theta\|$ and the robustness function is non-decreasing, then $\gamma = -\|\bar{\theta}\|$ is a valid choice for parameter $\gamma$.*

*Proof.* Clearly the interesting case to prove here is when $\theta \in [\underline{\theta}, \theta^*]$ such that $[\![\phi[\theta]]\!](\Sigma) < 0$ and $\theta' \in [\theta^*, \bar{\theta}]$ such that $[\![\phi[\theta']]\!](\Sigma) \geq 0$. In this case

$$\gamma = -\|\bar{\theta}\| \text{ and } [\![\phi[\theta']]\!](\Sigma) \geq 0 \text{ and}$$
$$\|\bar{\theta}\| \geq \|\theta'\| \geq \|\theta\| \implies \|\theta\| \geq \|\theta'\| + \gamma - [\![\phi[\theta']]\!](\Sigma)$$

Therefore, if the problem in Eq. (6) is feasible, then the optimum of equations (6) and (7) is the same.

Again, if $[\![\phi[\theta^*]]\!](\mu^*) > 0$, then we cannot infer that the system is correct for all parameter values in $\Theta$.

**Example 9 (AT)** *Let us consider the specification $\phi[\theta] = \Box_{[\theta,30]}\ (\omega \leq 4500)$ on our running example. The specification robustness $[\![\phi[\theta]]\!](\Delta_\Sigma(u))$ as a function of $\theta$ and the input $u$ appears in Fig. 9 for constant input signals. The creation of the graph required $100 \times 30 = 3,000$ tests. The contour under the surface indicates the zero level set of the robustness surface, i.e., the $\theta$ and $u$ values for which we get $[\![\phi[\theta]]\!](\Delta_\Sigma(u)) = 0$. We remark that the contour is actually an approximation of the zero level set computed by a linear interpolation using the neighboring points on the grid. From the graph, we could infer that $\theta^* \approx 13.8$ and that for any $\theta \in [0, 13.8]$, we would have $[\![\phi[\theta]]\!](\Sigma) \leq 0$. Again, the approximate value of $\theta^*$ is a rough estimate based on the granularity of the grid.*

*Using Eq. (9) as a cost function, we can now compute the a parameter for Example 9 using our toolbox* S-TaLiRo *[11, 12].* S-TaLiRo *returns $\theta^* \approx 12.59$ as optimal parameter for constant input $u(t) = 90.88$ within 250 tests. The temporal logic robustness for the specification $\Box_{[12.59,30]}(\omega \leq 4500)$ with respect to the input $u$ appears in Fig. 7.* ▲

## 6 Parameter Falsification Domain

We utilize the solution to Problem 1 and exploit the robustness landscape of a specific class of temporal logic formulas to present two algorithms to estimate $\Psi = \{\theta^* \in \Theta \mid \Sigma \not\models \phi[\theta^*]\}$ for Problem 1. In fact, we can reduce this problem to finding the set $\Theta^{bd} = \Psi \cap \{\theta^* \in \Theta \mid [\![\phi[\theta^*]]\!](\Sigma) = 0\}$ since the robustness landscape is monotonic. Here, $\Theta^{bd}$ represents the intersection of the robustness function with the zero level set. As a preprocessing step, the PMTL parameters are normalized in the range $[0, 1]$ to avoid biases during the optimization process.

The first method explores $\Theta^{bd}$ by modifying the priority function $f$ and thereby slightly shifting the minimum or maximum of the objective function in eq. 8 or eq. 9, respectively. Note that the magnitude of the shift is dependent on the shape of the robustness landscape of the model and specification.

As shown in Algorithm 1, set $\Psi$ is explored iteratively. For every iteration, we draw a random vector $\omega$ with size equals to the dimension of $\Theta$. The random vector will be used as parameter weights for the priority function $f(\theta)$. We run parameter estimation, which returns an approximation for Eq. (7). In case $\phi[\boldsymbol{\lambda}]$ is non-decreasing, the optimization algorithm $\eta$ is a maximization algorithm.
Similarly, if $\phi[\boldsymbol{\lambda}]$ is non-increasing, the optimization algorithm $\eta$ is a minimization algorithm. We utilize the parameters estimated and the corresponding robustness value to expand $\Psi$ and reduce the unknown parameter range for the next iteration. We present the iterative process in Fig. 8.

We define a PMTL specification monotonicity function $\mathcal{M}$ : PMTL $\rightarrow \{-1, 0, 1\}$ where

$$\mathcal{M}(\phi[\boldsymbol{\lambda}]) = \begin{cases} 1 & \text{if } \phi[\boldsymbol{\lambda}] \text{ is non-decreasing;} \\ -1 & \text{if } \phi[\boldsymbol{\lambda}] \text{ is non-increasing;} \\ 0 & \text{otherwise.} \end{cases}$$

The monotonicity computation is presented in [10] and generalized in [25].

**Algorithm 1** Robustness Guided Parameter Falsification Domain Algorithm RGDA($\eta$, $\Gamma$, $\Theta$, $\phi$, $\Sigma$, $n$, $t$)

**Require:** Stochastic optimization algorithm $\eta$, search space $\Gamma$, parameter range $\Theta$, specification $\phi$, system $\Sigma$, number of iterations $n$ and tests $t$

$\langle \Psi, \omega, \theta^*, \gamma \rangle \leftarrow \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$
**for** $i = 0$ to $n$ **do**
  $\omega \leftarrow$ RandomVector($[0,1]$, dim($\Theta$))
  $[\theta^*, \gamma] \leftarrow \eta(\Gamma, \Theta, \phi, \Sigma, t, \omega, \mathcal{M}(\phi[\theta^*]))$ ▷ run parameter estimation and robustness computation
  **if** $(\gamma \leq 0)$ **then**
    **if** $(\mathcal{M}(\phi[\theta^*]) = 1)$ **then**
      $\Psi \leftarrow \Psi \cup \{\theta \in \Theta \mid \forall i \ (0 \leq \theta_i \leq \theta_i^*\}$ ▷ expand the falsification domain $\Psi$
    **else if** $(\mathcal{M}(\phi[\theta^*]) = -1)$ **then**
      $\Psi \leftarrow \Psi \cup \{\theta \in \Theta \mid \forall i \ (\theta_i \geq \theta_i^* \geq 0)\}$
    **end if**
  **end if**
**end for**
**return** $\Psi$

The second algorithm explores the set $\Theta^{bd}$ by iteratively expending the set of falsifying parameters, namely, the set $\Psi$. However in this case, the search is finely structured and does not depend on randomized weights. For presentation purposes, let us consider the case for specifications with non-decreasing monotonicity. Given a normalized parameter range with dimension $n$, in each iteration of the algorithm, we conduct the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & c \\ \text{subject to} \quad & c * \mathbf{b} + \mathbf{p} \in \Theta \text{ and} \\ & \Sigma \not\models \phi[\ c * \mathbf{b} + \mathbf{p}\ ] \end{aligned} \tag{10}$$

where $\mathbf{p}$ is the starting point of the optimization problem in each iteration and $\mathbf{b}$ is the bias vector which enables to prioritize specific parameters in the search. We refer to the solution of eq. 10 in the $i^{th}$ iteration of the algorithm as $\mathbf{marker}(i)$. Initially, for the first iteration, the value of $\mathbf{p}$ is set to $\mathbf{0}$. The returned $\mathbf{marker}(1)$ from eq. 10 is then utilized to update $\Psi$, the set of parameters for which the system does not satisfy the specifications. Next, we generate at most $2^n - 2$ initial position vectors at the corners of the boundary of $\Psi$. The algorithm terminates when one of the following conditions is met: 1) The distance between **markers** is less than some value $\epsilon$, or 2) no new markers are generated from the current set of initial position vectors. Figure 9 represents the iterative process for Algorithm 2.

**Algorithm 2** Structured Parameter Falsification Domain Algorithm SDA($\eta$, $\Gamma$, $\Theta$, $\phi$, $\Sigma$, $t$, $\epsilon$, $\mathbf{b}$)

**Require:** Stochastic optimization algorithm $\eta$, search space $\Gamma$, parameter range $\Theta$, specification $\phi$, system $\Sigma$, number of tests $t$, minimum distance between markers $\epsilon$, bias vector $\mathbf{b}$

$\langle \Psi, \mathbf{p}, \mathcal{TC}, \mathcal{ML}, \mathcal{TL} \rangle \leftarrow \langle \emptyset, \emptyset, \bot, \{\}, \{\} \rangle$
**if** $(\mathcal{M}(\phi[\theta]) = 1)$ **then**
  $\mathbf{p} \leftarrow \mathbf{0}$(dim($\Theta$))
**else if** $(\mathcal{M}(\phi[\theta]) = -1)$ **then**
  $\mathbf{p} \leftarrow \mathbf{1}$(dim($\Theta$))
**end if**
$\mathcal{ML}$.Add($\mathbf{p}$)
**while** $\mathcal{TC} = \bot$ **do**
  $\mathcal{TL} \leftarrow \{\}$
  **for** $\mathbf{v}$ in $\mathcal{ML}$ **do**
    $[\theta^*, \gamma] \leftarrow \eta(\Gamma, \Theta, \phi, \Sigma, t, \omega, \mathcal{M}(\phi[\theta]), \mathbf{b}, \mathbf{v})$ ▷ run parameter estimation starting at $\mathbf{v}$ and search along the directional vector $\mathbf{b}$
    **if** $(\gamma \leq 0)$ **then**
      $\mathcal{TL}$.Add(GenerateMarkers($\theta^*$, $\mathcal{M}(\phi[\theta])$))
      **if** $(\mathcal{M}(\phi[\theta^*]) = 1)$ **then**
        $\Psi \leftarrow \Psi \cup \{\theta \in \Theta \mid \forall i \ (0 \leq \theta_i \leq \theta_i^*\}$
        $\Theta \leftarrow \Theta \setminus \Psi$
      **else if** $(\mathcal{M}(\phi[\theta^*]) = -1)$ **then**
        $\Psi \leftarrow \Psi \cup \{\theta \in \Theta \mid \forall i \ (\theta_i \geq \theta_i^* \geq 0)\}$
        $\Theta \leftarrow \Theta \setminus \Psi$
      **end if**
    **end if**
  **end for**
  $\mathcal{ML} \leftarrow \mathcal{TL}$
  **if** $\mathcal{ML}$.IsEmpty() or DistanceBetweenMarkers($\mathcal{ML}$) $< \epsilon$ **then** $\mathcal{TC} \leftarrow \top$
  **end if**
**end while**
**return** $\Psi$

## 7 Experiments and a Case Study

The algorithms presented in this work are implemented and publicly available through the Matlab toolbox S-TaLiRo [11,12].

The parametric MTL exploration of CPS is motivated by a challenge problem published by Ford in 2002 [26]. In particular, the report provided a simple – but still realistic – model of a powertrain system (both the physical system and the embedded control logic) and posed the question whether there are constant operating conditions that can cause a transition from gear two to gear one and then back to gear two. That behavior would imply that the transition was not necessary in the first place.

The system is modeled in Checkmate [27]. 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. 10. The system dynamics and switching conditions are linear. However, some switching conditions depend on the initial conditions of

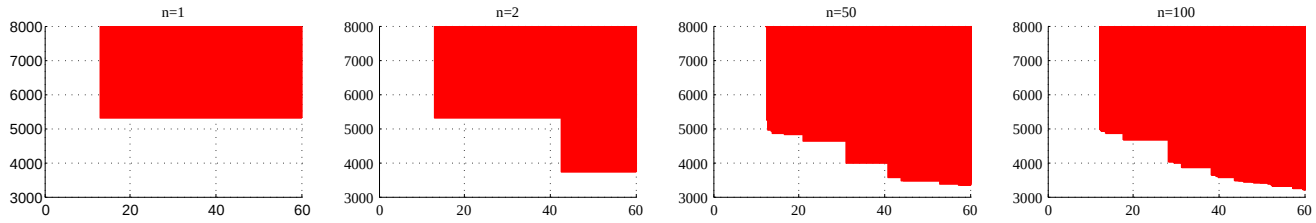


**Fig. 8.** Illustration of the iterative process for Algorithm 1. Specification: $\phi[\boldsymbol{\lambda}] = \neg(\Diamond_{[0,\lambda_1]} q \wedge \Box p[\lambda_2])$ where $p[\lambda_2] \equiv (\omega \leq \lambda_2)$ and $q \equiv (v \geq 100)$. Model: Automatic Transmission as described in Example 1. The red colored set represents set $\Psi = \{\theta^* \in \Theta \mid \Sigma \not\models \phi[\theta^*]\}$ i.e. the set of parameter values such that the system does not satisfy the specification.

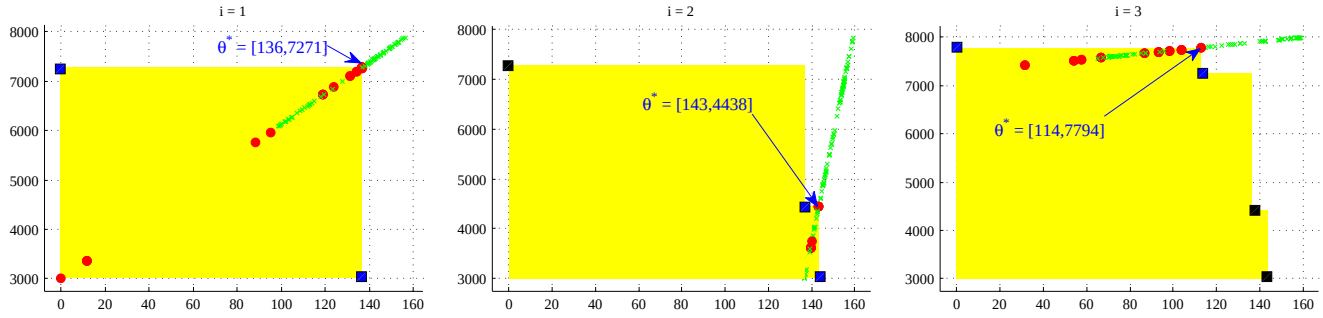


**Fig. 9.** Illustration of the iterative process for Algorithm 2. Specification: $\phi[\boldsymbol{\lambda}] = \Box(p[\lambda_1] \wedge q[\lambda_2])$ where $p[\lambda_1] \equiv (v \leq \lambda_1)$ and $q \equiv (\omega \leq \lambda_2)$. Model: Automatic Transmission as described in Example 1. The red colored set represents set $\Psi = \{\theta^* \in \Theta \mid \Sigma \not\models \phi[\theta^*]\}$ i.e. the set of parameter values such that the system does not satisfy the specification. The red circles represent parameter values for which system inputs and initial conditions are found that falsify the specification. The green marks represent parameter values for which falsification is not found. The blue squares indicate the initial positions for the next iteration.

the system. The latter makes the application of standard system verification tools not a straightforward task.

In [23], we demonstrated that S-TaLiRo [11,12] can successfully solve the challenge problem (see Fig. 10) by formalizing the requirement as an MTL specification $\phi^P_{e1} = \neg\Diamond(g_2 \wedge \Diamond(g_1 \wedge \Diamond g_2))$, where $g_i$ is a proposition that is true when the system is in gear $i$. Stochastic search methods can be applied to solve the resulting optimization problem where the cost function is the robustness of the specification. Moreover, inspired by the success of S-TaLiRo on the challenge problem, we tried to ask a more complex question. Namely, does a transition exists from gear two to gear one and back to gear two in less than 2.5 sec? An MTL specification that can capture this requirement is $\phi^P_{e2} = \Box((\neg g_1 \wedge X g_1) \rightarrow \Box_{(0,2.5]} \neg g_2)$. The natural question that arises is what would be the smallest time for which such a transition can occur? We can formulate a parametric MTL formula to query the model of the powertrain system: $\phi^P_{e3}[\lambda] = \Box((\neg g_1 \wedge X g_1) \rightarrow \Box_{(0,\lambda]} \neg g_2)$. We have extended S-TaLiRo to be able to handle parametric MTL specifications. The total simulation time of the model is set to 60 sec and the search interval is $\Theta = [0, 60]$. S-TaLiRo returned $\theta^* \approx 0.4273$ as the minimum parameter found (See Fig. 10) using about 300 tests of the system.

The challenge problem is extended to an industrial size high-fidelity engine model. The model is part of the SimuQuest Enginuity [28] Matlab/Simulink tool package. The Enginuity tool package includes a library of modules for engine component blocks. It also includes pre-assembled models for standard engine configurations, see Fig. 12. In this work, we will use the Port Fuel Injected (PFI) spark ignition, 4 cylinder inline engine configuration. It models the effects of combustion from first physics principles on a cylinder-by-cylinder basis, while also including regression models for particularly complex physical phenomena. Simulink reports that this is a 56 state model. The model includes a tire-model, brake system model, and a drive train model (including final drive, torque converter and transmission). The model is based on a zero-dimensional modeling approach so that the model components can all be expressed in terms of ordinary differential equations. The inputs to the system are the throttle and brake schedules, and the road grade, which represents the incline of the road. The outputs are the vehicle and engine speed, the current gear and a timer that indicates the time spent on a gear. We search for a particular input for the throttle schedule, brake schedule, and grade level. The inputs are parametrized using 12 search variables, where 7 are used to model the throttle schedule, 3 for the brake schedule, and 2 for the grade level. The search variables for each input are interpolated with the Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) function provided as a Matlab function by Mathworks. The simulation time is 60s. We demonstrate the parameter estimation method for two specifications:

1. The specification $\phi^S_1[\lambda] = \Box_{[0,60]}((g2 \wedge X g1) \rightarrow \Box_{[0,\lambda_1]}((\tau \leq \lambda_1) \rightarrow g1)$, where $\tau$ is the time spent in a gear. The specification states that after shifting into gear one from gear two, there should be no shift from gear one to any other gear within $\lambda$ sec. Clearly, the

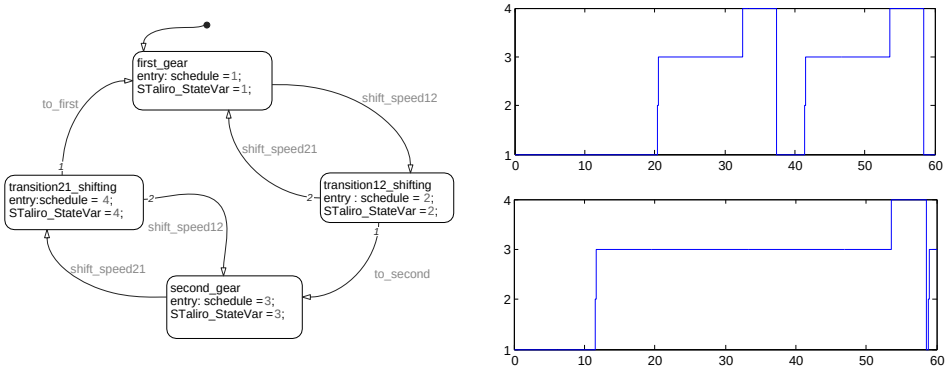


**Fig. 10.** Left: The shift scheduler of the powertrain challenge problem. Right: Shift schedules. The numbers correspond to the variables in the states of the shift scheduler. Right Top: The shift schedule falsifying requirement $\phi^P_{e1}$. Right Bottom: The shift schedule falsifying requirement $\phi^P_{e3}[0.4273]$.

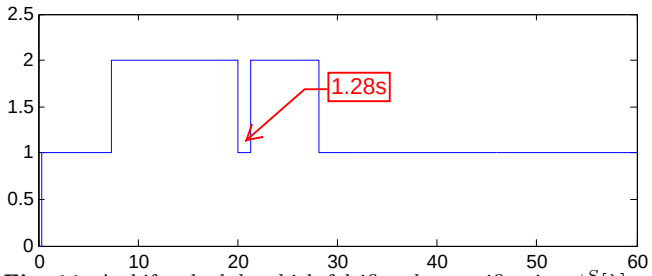


**Fig. 11.** A shift schedule which falsifies the specification $\phi^S_1[\lambda] = \square_{[0,60]}((g_2 \wedge Xg_1) \rightarrow \square_{[0,1.29]}((\tau \leq 1.29) \rightarrow g_1)$.

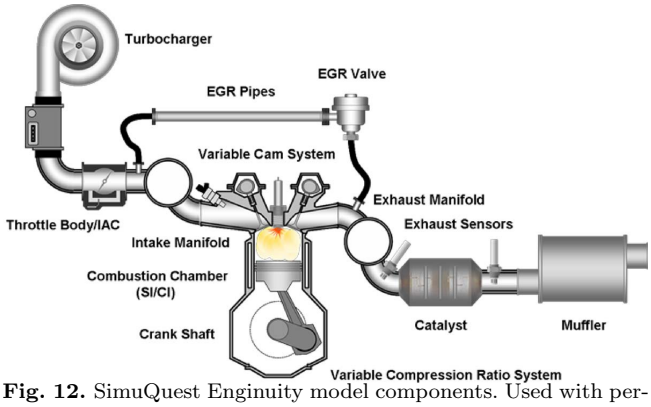


**Fig. 12.** SimuQuest Enginuity model components. Used with permission, ©SimuQuest [28].

property defined is equivalent to the property defined in the challenge problem in the sense that the set of trajectories that satisfy/falsify the property is the same. The reason for the change made is the improved performance of the hybrid distance metric [29] with the modified specification. The estimated parameter for the specification returned is $1.29s$. Figure 11 presents a shift schedule for which a transition out of gear one occurs in 1.28sec.

2. The specification $\phi^S_2[\boldsymbol{\lambda}] = \square((v < \lambda_1) \wedge (\omega < \lambda_2))$, where $\lambda_1$, $\lambda_2$ represent the vehicle and engine speed parameters, respectively. The specification states that the vehicle and engine speed is always less than $\lambda_1$ and $\lambda_2$, respectively. The estimated parameters for the specification returned are 137.1mph and 4870rpm.

In Table 1, we present experimental results for specifications on the Powertrain, Automotive Transmission, and Simuquest Enginuity high-fidelity Engine models.

A detailed description of the benchmark problems can be found in [18,7] and the benchmarks can be downloaded with the S-TaLiRo distribution [12].

## 8 Related Work

The topic of testing embedded software and, in particular, embedded control software is a well studied problem that involves many subtopics well beyond the scope of this paper. We refer the reader to specialized book chapters and textbooks for further information [31,32]. Similarly, a lot of research has been invested on testing methods for Model Based Development (MBD) of embedded systems [3]. However, the temporal logic testing of embedded and hybrid systems has not received much attention [33,34,4,35].

Parametric temporal logics were first defined over traces of finite state machines [36]. In parametric temporal logics, some of the timing constraints of the temporal operators are replaced by parameters. Then, the goal is to develop algorithms that will compute the values of the parameters that make the specification true under some optimality criteria. That line of work has been extended to real-time systems and in particular to timed automata [37] and continuous-time signals [10]. The authors in [22,38] define a parametric temporal logic called quantifier free Linear Temporal Logic over real valued signals. However, they focus on the problem of determining system parameters such that the system satisfies a given property rather than on the problem of exploring the properties of a given system.

Another related problem is specification mining or model exploration for finite state machines. The problem was initially introduced by William Chan in [39] under the term Temporal Logic Queries. The goal of model

**Table 1.** Experimental results of Parameter Estimation with S-TaLiRo and comparison with Breach. In S-TaLiRo the parameters were estimated by running 1000 tests. Legend: $\mathbf{f}(\boldsymbol{\lambda})$ : the priority function used, $\phi_{\mathbf{i}}^{\mathbf{AT}}$ : Specifications tested on the Automotive Transmission Model, $\phi^{\mathbf{P}}$ : Specification tested on the Powertrain Model, $\phi^{\mathbf{S}}$ : Specification tested on the Simuquest Enginuity high-fidelity Engine Model. The gray colored rows are first presented in [9] and are included for completeness.

| | S-TaLiRo | | |
|---|---|---|---|
| Specification | $f(\lambda)$ | Time | Parameter Estimate |
| $\phi_1^{AT}[\lambda] = \neg\Diamond((v \geq 120) \wedge \Diamond_{[0,\lambda]}(\omega \geq 4500))$ | $\lambda$ | 135s | $7.7s$ |
| $\phi_2^{AT}[\lambda] = \neg\Diamond((v \geq 120) \wedge \Diamond_{[0,\lambda]}(v \geq 125))$ | $\lambda$ | 138s | $10.00s$ |
| $\phi_3^{AT}[\lambda] = \neg\Diamond((v \geq 120) \wedge \Diamond_{[0,\lambda]}(\omega \geq 4500))$ | $\lambda$ | 137s | $7.57s$ |
| $\phi_4^{AT}[\lambda] = \neg\Diamond((v \geq 120) \wedge \Diamond_{[0,\lambda]}(\omega \geq 4500))$ | $\lambda$ | 132s | $7.56s$ |
| $\phi_5^{AT}[\boldsymbol{\lambda}] = \Box((v \leq \lambda_1) \wedge (\omega \leq \lambda_2))$ | $\|\boldsymbol{\lambda}\|$ | 139s | $\langle 138mph, 5981rpm\rangle$ |
| | $\lambda_1$ | 137s | $\langle 57mph, 6000rpm\rangle$ |
| | $\lambda_2$ | 138s | $\langle 180mph, 2910rpm\rangle$ |
| | $max(\boldsymbol{\lambda})$ | 138s | $\langle 109mph, 6000rpm\rangle$ |
| | $min(\boldsymbol{\lambda})$ | 138s | $\langle 154mph, 5300rpm\rangle$ |
| $\phi_6^{AT}[\boldsymbol{\lambda}] = \neg(\Diamond_{[0,\lambda_1]}(v \geq 100) \wedge \Box(\omega \leq \lambda_2))$ | $\|\boldsymbol{\lambda}\|$ | 144s | $\langle 15.7s, 4820rpm\rangle$ |
| | $\lambda_1$ | 142s | $\langle 44.6s, 3598rpm\rangle$ |
| | $\lambda_2$ | 138s | $\langle 12.2s, 6000rpm\rangle$ |
| | $max(\boldsymbol{\lambda})$ | 140s | $\langle 37.3s, 3742rpm\rangle$ |
| | $min(\boldsymbol{\lambda})$ | 142s | $\langle 12.3s, 5677rpm\rangle$ |
| $\phi_7^{AT}[\boldsymbol{\lambda}] = \Box((v \leq \lambda_1) \wedge (\omega \leq \lambda_2)) \wedge \Diamond_{[0,\lambda_3]}(v \geq 150) \wedge \Diamond_{[0,\lambda_4]}(\omega \geq 4500)$ | $\|\boldsymbol{\lambda}\|$ | 145s | $\langle 198mph, 4932rpm, 59.5s, 55s\rangle$ |
| | $max(\boldsymbol{\lambda})$ | 143s | $\langle 129mph, 6000rpm, 48.9s, 28.3s\rangle$ |
| | $min(\boldsymbol{\lambda})$ | 142s | $\langle 190mph, 5575rpm, 55.1s, 54.8s\rangle$ |
| $\phi_8^{AT}[\boldsymbol{\lambda}] = \Box((v \leq \lambda_1) \wedge (\omega \leq \lambda_2)) \wedge \Diamond_{[0,\lambda_3]}(v \geq 150) \wedge \Diamond_{[0,\lambda_4]}(\omega \geq 4500) \wedge \Box_{[\lambda_5,60]}(v \geq 170) \wedge \Box_{[\lambda_6,60]}(\omega \geq 4750)$ | $\|\boldsymbol{\lambda}\|$ | 146s | $\langle 159mph, 5700rpm, 48.3s, 36.2s, 54.2s, 53.9s\rangle$ |
| | $max(\boldsymbol{\lambda})$ | 145s | $\langle 85.9mph, 6000rpm, 3.8s, 38.8s, 44.5s, 51.5s\rangle$ |
| | $min(\boldsymbol{\lambda})$ | 143s | $\langle 191mph, 4958rpm, 43s, 55.3s, 42s, 47.1s\rangle$ |
| $\phi_{e3}^{P}[\lambda] = \Box((\neg g_1 \wedge Xg_1) \rightarrow \Box_{(0,\lambda]}\neg g_2)$ | $\lambda$ | 2600s | $0.1s$ |
| $\phi_1^{S}[\lambda] = \Box_{[0,60]}((g_2 \ \wedge \ Xg_1) \rightarrow \Box_{[0,\lambda_1]}((t \leq \lambda_1) \rightarrow g_1)$ | $\lambda$ | 21803s | $1.29s$ |

**Table 2.** Experimental Comparison of the method presented in this paper (PE) and the parameter synthesis method presented in [30], (PS). Legend: **#Sim.**: the number of system simulations, **#Rob**: the number of robustness computations.

| Specification | Method | Parameter Estimate | | Time | #Sim | #Rob |
|---|---|---|---|---|---|---|
| $\phi_2^{S}[\lambda] = \Box((v \leq \lambda_1) \wedge (\omega \leq \lambda_2))$ | PE | 137.1 mph | 4870 rpm | 20170s | 1000 | 1000 |
| | PS | 149.8 mph | 4883 rpm | 50017s | 2386 | 5130 |
| $\phi_5^{AT}[\boldsymbol{\lambda}] = \Box((v \leq \lambda_1) \wedge (\omega \leq \lambda_2))$ | PE | 100.2 mph | 5987.6 rpm | 106s | 1000 | 1000 |
| | PS | 137.5 mph | 6000 rpm | 253s | 2176 | 11485 |
| $\phi_6^{AT}[\boldsymbol{\lambda}] = \neg(\Diamond_{[0,\lambda_1]}(v \geq 100) \wedge \Box(\omega \leq \lambda_2)$ | PE | 21s | 3580 rpm | 110s | 1000 | 1000 |
| | PS | 59.06s | 3296 rpm | 397s | 3443 | 9718 |

exploration is to help the designer achieve a better understanding and explore the properties of a model of the system. Namely, the user can pose a number of questions in temporal logic where the atomic propositions are replaced by a placeholder and the algorithm will try to find the set of atomic propositions for which the temporal logic formula evaluates to true. Since the first paper [39], several authors have studied the problem and proposed different versions and approaches [40–43]. A related approach is based on specification mining over temporal logic templates [44] rather than special placeholders in a specific formula. In [45], the authors present an inference algorithm that finds temporal logic properties of a system from data. The authors introduce a reactive parameter signal temporal logic and define a partial order over it to aid the property definition process.

In [30], the authors provide a parameter synthesis algorithm for Parametric Signal Temporal Logic (PSTL), a similar formalism to MTL. To conduct parameter synthesis for multiple parameters, a binary search is utilized to set the parameter value for each parameter in sequence. After a set of parameters is proposed, a stochastic optimization algorithm is utilized to search for trajectories that falsify the specification. If it fails to do so, the algorithm stops, otherwise this two step process continues until the termination condition is met.

In the following, we present three main differences between the method proposed here (PE) and the method proposed in [30], (PS). First, PE is a best effort algorithm for which the termination condition is the number of tests the system engineer is interested to conduct. Clearly, the more tests, the better the search space is explored. Since the parameter estimation problem is presented as a single optimization problem, runtime is not affected by the number of parameters in the specification. In contrast, in PS, the runtime of the algo-

rithm through binary search is affected by the number of parameters in the PSTL formula. For each iteration of the binary search, multiple robustness computations have to be conducted, which for systems that output a large trace and contain complex specifications, could become costly. The second step in the framework is the falsification of the parameters proposed. This algorithm needs to be performed on every iteration, until a falsification is found. If a falsifying trajectory is not found, the stopping condition is met and the parameters are returned. Second, in PE, the parameters returned are the "best" parameters for which a falsifying trajectory is found. In PS, the proposed parameters are parameters for which no falsifying trajectory is found. Proving that a specification holds for hybrid systems, in general, is undecidable and therefore the failure to find a falsifying trajectory does not imply that one does not exist. Third, in PE, through the priority function, we enable the system engineer to have flexibility when assigning weights and priorities to parameters. In PS, parameter synthesis through binary search implicitly prioritizes one parameter over others.

We compare the two methods using the Simuquest Enginuity high-fidelity Engine model and the Automotive Transmission model. To enable the comparison of the two methods, we have implemented the PS method in S-TaLiRo. Note that the simulation time is 60s. The experimental results are presented in Table 2. For the PE method, the number of simulations and robustness computations is predefined. On the other hand, for the PS method, these numbers vary following the reasons presented in the previous paragraph. As a result, the difference in computation time between the two methods is significant.

The results for the Automotive Transmission model can be reproduced by running the experiments in the S-TaLiRo distribution [12].

## 9 Conclusion

An important stage in Model Based Development (MBD) of software for CPS is the formalization of system requirements. We advocate that Metric Temporal Logic (MTL) is an excellent candidate for formalizing interesting design requirements. In this paper, we have presented a solution on how we can explore system properties using Parametric MTL (PMTL) [10]. Based on the notion of robustness of MTL [6], we have converted the parameter estimation problem into an optimization problem which we approximate using S-TaLiRo [11,12]. We have presented a method for estimating the range for multiple parameters as long as the robustness function has the same monotonicity with respect to all the parameters. Finally, we have demonstrated that our method can provide interesting insights to the powertrain challenge problem [26].We demonstrated the method on an industrial size engine model and examples from related works.

*Acknowledgements.* This work has been partially supported by award NSF CNS 1116136.

## References

1. Lions, J.L., Lbeck, L., Fauquembergue, J.L., Kahn, G., Kubbat, W., Levedag, S., Mazzini, L., Merle, D., O'Halloran, C.: Ariane 5, flight 501 failure, report by the inquiry board. Technical report, CNES (1996)
2. Hoffman, E.J., Ebert, W.L., Femiano, M.D., Freeman, H.R., Gay, C.J., Jones, C.P., Luers, P.J., Palmer, J.G.: The near rendezvous burn anomaly of december 1998. Technical report, Applied Physics Laboratory, Johns Hopkins University (1999)
3. Tripakis, S., Dang, T.: Modeling, Verification and Testing using Timed and Hybrid Automata. In: Model-Based Design for Embedded Systems. CRC Press (2009) 383–436
4. Nghiem, T., Sankaranarayanan, S., Fainekos, G.E., Ivancic, F., Gupta, A., Pappas, G.J.: 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) 211–220
5. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems **2** (1990) 255–299
6. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theoretical Computer Science **410** (2009) 4262–4291
7. Sankaranarayanan, S., Fainekos, G.: Falsification of temporal properties of hybrid systems using the cross-entropy method. In: ACM International Conference on Hybrid Systems: Computation and Control. (2012)
8. Annapureddy, Y.S.R., Fainekos, G.E.: Ant colonies for temporal logic falsification of hybrid systems. In: Proceedings of the 36th Annual Conference of IEEE Industrial Electronics. (2010) 91–96
9. Yang, H., Hoxha, B., Fainekos, G.: Querying parametric temporal logic properties on embedded systems. In: Int. Conference on Testing Software and Systems. (2012)
10. Asarin, E., Donzé, A., Maler, O., Nickovic, D.: Parametric identification of temporal properties. In: Runtime Verification. Volume 7186 of LNCS., Springer (2012) 147–160
11. Annapureddy, Y.S.R., Liu, C., Fainekos, G.E., Sankaranarayanan, S.: 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., Springer (2011) 254–257
12. S-TaLiRo: Temporal Logic Falsification Of Cyber-Physical Systems. `https://sites.google.com/a/asu.edu/s-taliro/s-taliro` (2013) [Online; accessed April-2014].
13. Hoxha, B., Bach, H., Abbas, H., Dokhanchi, A., Kobayashi, Y., Fainekos, G.: (Towards formal specification visualization for testing and monitoring of cyber-physical systems)

14. Sankaranarayanan, S., Homaei, H., Lewis, C.: Model-based dependability analysis of programmable drug infusion pumps. In: Formal modeling and analysis of timed systems. Springer (2011) 317–334
15. Sankaranarayanan, S., Fainekos, G.: Simulating insulin infusion pump risks by in-silico modeling of the insulin-glucose regulatory system. In: International Conference on Computational Methods in Systems Biology. (2012)
16. Jiang, Z., Pajic, M., Mangharam, R.: Cyber-physical modeling of implantable cardiac medical devices. Proceedings of the IEEE **100** (2012) 122–137
17. Chen, T., Diciolla, M., Kwiatkowska, M.Z., Mereacre, A.: A simulink hybrid heart model for quantitative verification of cardiac pacemakers. In: Proceedings of the 16th international conference on Hybrid systems: computation and control, ACM (2013) 131–136
18. Abbas, H., Fainekos, G.E., Sankaranarayanan, S., Ivancic, F., Gupta, A.: Probabilistic temporal logic falsification of cyber-physical systems. ACM Transactions on Embedded Computing Systems **(In Press)** (2011)
19. Alur, R., Henzinger, T.A.: Real-Time Logics: Complexity and Expressiveness. In: Fifth Annual IEEE Symposium on Logic in Computer Science, Washington, D.C., IEEE Computer Society Press (1990) 390–401
20. Zhao, Q., Krogh, B.H., Hubbard, P.: Generating test inputs for embedded control systems. IEEE Control Systems Magazine **August** (2003) 49–57
21. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical computer science **138** (1995) 3–34
22. Fages, F., Rizk, A.: On temporal logic constraint solving for analyzing numerical data time series. Theor. Comput. Sci. **408** (2008) 55–65
23. Fainekos, G., Sankaranarayanan, S., Ueda, K., Yazarel, H.: Verification of automotive control applications using s-taliro. In: Proceedings of the American Control Conference. (2012)
24. Donze, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Formal Modelling and Analysis of Timed Systems. Volume 6246 of LNCS., Springer (2010)
25. Donze, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Computer Aided Verification. Volume 6174 of LNCS. Springer (2010) 167–170
26. Chutinan, A., Butts, K.R.: Dynamic analysis of hybrid system models for design validation. Technical report, Ford Motor Company (2002)
27. Silva, B.I., Krogh, B.H.: Formal verification of hybrid systems using CheckMate: a case study. In: Proceedings of the American Control Conference. Volume 3. (2000) 1679 – 1683
28. Simuquest: Enginuity. (`http://www.simuquest.com/products/enginuity`) Accessed: 2013-10-14.
29. Abbas, H., Fainekos, G.: Linear hybrid system falsification through local search. In: Automated Technology for Verification and Analysis. Volume 6996 of LNCS., Springer (2011) 503–510
30. Jin, X., Donzé, A., Deshmukh, J.V., Seshia, S.A.: Mining requirements from closed-loop control models. In: Proceedings of the 16th international conference on Hybrid systems: computation and control, ACM (2013) 43–52
31. Conrad, M., Fey, I.: Testing automotive control software. In: Automotive Embedded Systems Handbook. CRC Press (2008)
32. Koopman, P.: Better Embedded System Software. Drumnadrochit Education LLC (2010)
33. Tan, L., Kim, J., Sokolsky, O., Lee, I.: Model-based testing and monitoring for hybrid embedded systems. In: Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration. (2004) 487–492
34. Plaku, E., Kavraki, L.E., Vardi, M.Y.: 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). Volume 5505 of LNCS., Springer (2009) 368 – 382
35. Zuliani, P., Platzer, A., Clarke, E.M.: Bayesian statistical model checking with application to simulink/stateflow verification. In: Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control. (2010) 243–252
36. Alur, R., Etessami, K., La Torre, S., Peled, D.: Parametric temporal logic for model measuring. ACM Trans. Comput. Logic **2** (2001) 388–407
37. Di Giampaolo, B., La Torre, S., Napoli, M.: Parametric metric interval temporal logic. In Dediu, A.H., Fernau, H., Martin-Vide, C., eds.: Language and Automata Theory and Applications. Volume 6031 of LNCS. Springer (2010) 249–260
38. Rizk, A., Batt, G., Fages, F., Soliman, S.: On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In: International Conference on Computational Methods in Systems Biology. Volume 5307 of LNCS., Springer (2008) 251–268
39. Chan, W.: Temporal-logic queries. In: Proceedings of the 12th International Conference on Computer Aided Verification. Volume 1855 of LNCS., London, UK, Springer (2000) 450–463
40. Bruns, G., Godefroid, P.: Temporal logic query checking. In: Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society (2001) 409 – 417
41. Chechik, M., Gurfinkel, A.: Tlqsolver: A temporal logic query checker. In: Proceedings of the 15th International Conference on Computer Aided Verification. Volume 2725., Springer (2003) 210–214
42. Gurfinkel, A., Devereux, B., Chechik, M.: Model exploration with temporal logic query checking. SIGSOFT Softw. Eng. Notes **27** (2002) 139–148
43. Singh, A., Ramakrishnan, C., Smolka, S.: Query-based model checking of adhocnetworkprotocols. In Bravetti, M., Zavattaro, G., eds.: Concurrency Theory. Volume 5710 of LNCS. Springer (2009) 603–619
44. Wasylkowski, A., Zeller, A.: Mining temporal specifications from object usage. In: 24th IEEE/ACM International Conference on Automated Software Engineering. (2009)
45. Kong, Z., Jones, A., Medina Ayala, A., Aydin Gol, E., Belta, C.: Temporal logic inference for classification and prediction from data. In: Proceedings of the 17th international conference on Hybrid systems: computation and control, ACM (2014) 273–282