

# MTL Robust Testing and Verification for LPV Systems

Georgios E. Fainekos and George J. Pappas

**Abstract**—This paper deals with the robust Metric Temporal Logic (MTL) testing and verification of linear systems with parametric uncertainties. This is a very general class of systems that includes not only Linear Time Invariant (LTI) systems with unknown constant parameters, but also Linear Time Varying (LTV) systems and certain classes of nonlinear systems through abstraction. The two main tools for the solution of this problem are the approximate bisimulation relations and a notion of robustness for temporal logic formulas.

## I. INTRODUCTION

One of the fundamental problems in model based development is how you can guarantee that a model is correct. In other words, does your design satisfy a set of user-defined requirements? Even though several industrial platforms offer tools for automatically testing such models [1], the verification problem cannot still be declared as solved. First, because no testing method can handle all the classes of systems and user requirements and, second, because the main issue in the complete verification of models of control systems is the undecidability of the problem [2].

Therefore, several authors have proposed methods for the systematic testing of hybrid systems [3]–[5], i.e., systems that have both continuous and discrete dynamics. Others, inspired by the success of temporal logics in software model checking [6], have introduced temporal logics as a formalism for property based testing of continuous systems [7], [8]. In our earlier work, we introduced robust simulations [9]–[11] as a way for testing models of continuous (and hybrid systems) with coverage guarantees. The concept of robust simulations has since been applied to other testing frameworks [12], [13], too. The principal advantage of robust simulations is that by performing one simulation you can infer the behavior of a neighborhood of trajectories of the system. Especially in [9], we demonstrated how robust simulations can be used for verifying timing requirements expressed in Metric Temporal Logic (MTL) [14] of linear and nonlinear time invariant systems.

In this paper, we extend the results of [9] to Linear Parameter Varying (LPV) [15] systems. LPV systems are a very general class of systems that can model Linear Time Invariant (LTI) systems whose parameters are unknown and possibly time varying, but within certain bounds. In other words, LPVs can model systems which have parameters that cannot be precisely determined. Moreover, LPV systems can

capture (through abstraction) certain classes of nonlinear systems, linear time varying systems and, even, hybrid systems whose discrete dynamics only depend on time.

The approach we follow is similar to the model reduction framework proposed in [16]. Our method comprises the following two steps which are also the contribution of this paper. First, we approximate the LPV system with an LTI system by constructing a bisimulation function. Second, we convert the user requirements into a new “robustified” specification that must hold on the LTI system. The last step can be checked by any method that can perform MTL verification on an LTI system. Here, we use the algorithm that we introduced in [9] to demonstrate the proposed framework on two different numerical problems.

We mention in passing that this not the first work that tries to address the verification and/or testing problem for dynamical systems with uncertain parameters. However, to the best of our knowledge, this is the first attempt to do so for MTL specifications. For example, the authors in [17] present a framework for the verification of linear circuits with uncertain parameters with respect to frequency domain specifications. In [18], a method for constructing Labeled Hybrid Petri Nets (LHPNs) from simulation traces of a circuit is presented. Similar to our verification framework, this approach can also handle varying parameters. Then, the resulting LHPN is model checked using a variety of methods proposed by the authors in their previous works.

## II. PROBLEM FORMULATION

In this paper, we address the problem of temporal logic testing and verification of autonomous linear systems with parametric uncertainties [15], which are commonly referred to as Linear Parametric Varying (LPV) systems. LPV systems usually have uncertain or time varying parameters that appear in the system matrices

$$\begin{aligned} \dot{x}(t) &= A(p(t))x(t) & p(t) &\in P & x(0) &\in X_0 & t &\in R \\ y(t) &= Cx(t) \end{aligned} \quad (1)$$

where  $x(t) \in \mathbb{R}^n$ ,  $y(t) \in \mathbb{R}^m$ ,  $A(p) \in \mathbb{R}^{n \times n}$ ,  $C \in \mathbb{R}^{m \times n}$ ,  $X_0 \subseteq \mathbb{R}^n$  is the set of initial conditions for the system, which is a nonempty compact (convex) polyhedral set,  $R \subseteq \mathbb{R}_{\geq 0}$  is the time domain and  $P \subset \mathbb{R}^q$ . Note that the parameter vector  $p$  may be constant but unknown to us or it may be time varying, that is,  $p(t) \in P$ . The dependence of  $A$  on the parameters  $p$  cannot be arbitrary. In order to derive a tractable computable solution to our problem,  $A$  and  $p$  must satisfy the following assumption.

*Assumption 1:* The parameter  $p : R \rightarrow P$  must be a piecewise continuous function and the matrix

This research has been partially supported by NSF CSR-EHS 0720518. The authors are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA. E-mail: {fainekos+,pappasg+}@grasp.upenn.edu

$A : P \rightarrow \mathbb{R}^{n \times n}$  of system (1) must be a multi-linear function of the components of  $p$ , that is,  $A(p(t)) = \sum_{i_1=0}^1 \sum_{i_2=0}^1 \cdots \sum_{i_q=0}^1 A_{i_1, i_2, \dots, i_q} p_1^{i_1}(t) p_2^{i_2}(t) \cdots p_q^{i_q}(t)$ , where  $p_i(t)$  denotes the  $i$ -th component of the vector function  $p$  and  $A_{i_1, i_2, \dots, i_q} \in \mathbb{R}^{n \times n}$  are constant matrices. Moreover, the set  $P$  must be an axis-aligned hyper-rectangle, that is,  $P := [p_1, \bar{p}_1] \times [p_2, \bar{p}_2] \times \dots \times [p_q, \bar{p}_q]$ .

The fact that  $p$  is piecewise continuous and that  $A$  is continuous guarantees that the system (1) has a unique solution  $x : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n$  for each initial condition  $x(0) = x_0 \in X_0$ . For convenience, we will denote a particular instantiation of system (1), i.e., a known tuple  $(A, C, X_0, P, R)$ , by  $\Sigma^u$ . In our analysis, we will also consider Linear Time Invariant (LTI) systems that arise from an LPV system  $\Sigma^u$  under a constant parameter vector  $\hat{p} \in P$ , i.e.,  $p(t) = \hat{p}$  for all  $t \in R$ . Such systems will be denoted by  $\Sigma^u(\hat{p})$ . Throughout the paper, we will use the hat or superscript notation to denote a point in a set while the letter itself refers usually to a function. For example,  $\hat{p}$  or  $p^0$  refers to a point in  $P$ , while  $p$  to a function that takes values in  $P$ . Some of the results in this paper hold for any dynamical system and not just LPV systems. In these cases, we will refer abstractly to a system  $\Sigma$  with state trajectories  $x(t)$  and observable trajectories  $y(t)$ .

The set of state trajectories of  $\Sigma$  which are defined over a time domain  $[0, T]$ , that is, the solutions  $x(t)$  of  $\Sigma$  with  $t \in R = [0, T]$ , are denoted by  $\Lambda_T(\Sigma)$ . If the time domain is unbounded, then we use the notation  $\Lambda_\infty(\Sigma)$  to denote the set of trajectories  $x(t)$  of  $\Sigma$  over the time domain  $R = \mathbb{R}_{\geq 0}$ . When the underlying time domain is not of interest, we just write  $\Lambda(\Sigma)$ . The set of observable trajectories  $y(t)$  of  $\Sigma$  which correspond to a set of internal trajectories  $\Lambda_T(\Sigma)$  are denoted by  $\mathcal{L}_T(\Sigma)$ . For example, in the case of an LPV system  $\Sigma^u$ , we have  $\mathcal{L}_T(\Sigma^u) = C\Lambda_T(\Sigma^u)$ . We similarly define  $\mathcal{L}_\infty(\Sigma)$  and  $\mathcal{L}(\Sigma)$ .

In order to reason about the timing properties of systems, we have to define certain sets of interest in  $\mathbb{R}^m$ , which is the set of observable system values. For example, we would like to know whether all the observable trajectories  $y(t) \in \mathbb{R}$  of a system attain a value in the set  $[10, +\infty)$ . We do so by using a set of atomic propositions  $AP$  which label subsets of  $\mathbb{R}^m$ . In other words, we define an observation map  $\mathcal{O} : AP \rightarrow \mathcal{P}(\mathbb{R}^m)$  such that for each  $\pi \in AP$  the corresponding set is  $\mathcal{O}(\pi) \subseteq \mathbb{R}^m$ . Here,  $\mathcal{P}(S)$  denotes the powerset of a set  $S$ .

Metric Temporal Logic (MTL) formulas are built over a set of propositions, the set  $AP$  in our case, using combinations of the traditional and temporal operators. 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* ( $\square_{\mathcal{I}}$ ), *until* ( $\mathcal{U}_{\mathcal{I}}$ ) and *release* ( $\mathcal{R}_{\mathcal{I}}$ ). The subscript  $\mathcal{I}$  imposes timing constraints on the temporal operators. In this work, we interpret MTL formulas over the observable trajectories of a given system  $\Sigma$ . Similarly to the notation for a system  $\Sigma$ , we denote the set of *signals*<sup>1</sup> which

<sup>1</sup>We prefer the term *signals* over the term *trajectories* since these signals are not generated by a particular system.

are defined over a bounded time domain  $[0, T]$  and which satisfy an MTL formula  $\phi$  under a map  $\mathcal{O}$  by  $\mathcal{L}_T^{\mathcal{O}}(\phi)$ . Again, the subscript  $\infty$  denotes a set of signals defined over an unbounded time domain, while no subscript implies that we do not care whether the signals are defined over a bounded or unbounded time domain.

For such systems and specifications, we formally solve the following problem.

*Problem 1:* Given an MTL specification  $\phi$  built over the set  $AP$ , an observation map  $\mathcal{O}$ , an LPV system  $\Sigma^u$  and a time  $T$ , determine whether  $\mathcal{L}_T(\Sigma^u) \subseteq \mathcal{L}_T^{\mathcal{O}}(\phi)$ .

In other words, we try to determine if every behavior of  $\Sigma^u$  in the time domain  $[0, T]$  is a behavior permitted by the temporal logic specification  $\phi$ . The challenges in solving Problem 1 are twofold. First, we have an infinite number of initial conditions to verify, i.e., the set  $X_0$  and, second, the parameters of the system are not known in advance, i.e., they belong to the uncountably infinite set  $P$ . The latter problem, whose solution is the main contribution of this paper (Section IV-B), is addressed by constructing bisimulation functions that can take into account parametric system uncertainties. The former problem is addressed by reducing the verification problem to MTL verification for LTI systems (Section V) and, then, applying the framework that we have developed in our previous work [9].

*Example 1:* An example instance of Problem 1 is the verification of the transient behavior of an RLC circuit as in Fig. 1. Such circuits are used to represent high voltage transmission lines where the requirement is the protection of the line against traveling waves or the interconnect in ultra-deep submicron integrated circuits where we have to study the interconnect delay. Under the assumption that the values of  $r$ ,  $l$  and  $c$  are constant and known, we can easily derive (see for example [19]) a set of linear differential equations  $\dot{x}(t) = Ax(t) + bu_{in}(t)$ , which form the state space representation of the RLC circuit. Here, the state vector  $x$  consists of the currents in the inductances  $(x_1, x_3, \dots, x_9)$  and the voltages across the capacitances  $(x_2, x_4, \dots, x_{10})$ . The goal of the analysis of such systems usually is to study the transient behavior of the circuit for specific parameter values and initial conditions under a unit step input function, i.e.,  $\forall t \in R. u_{in}(t) = 1$ .

However in practical situations, the exact initial conditions and parameter values might be unknown. For this example, we will assume that  $x(0) \in X_0 = \prod_{i=1}^5 (\{0\} \times [-\alpha, \alpha])$  for some scalar  $\alpha > 0$  and that  $r$  and  $l$  are constant and known. However,  $c$  is unknown and possibly time varying such that  $c(t) \in [c_0 - \beta, c_0 + \beta]$ , where  $c_0 > 0$  is the nominal value of the capacitance, and  $0 < \beta < c_0$  is a scalar. In this case, the model of the circuit becomes an LPV system:  $\dot{x} = A(p)x + b$ , where  $p \in [-\beta, \beta]^5$ . We need to verify that the voltage at all the nodes does not exceed 2 voltage units and that the voltage at the receiving end, i.e.,  $x_{10}$  stabilizes within 3 time units in the range  $[0.75, 1.25]$ . Thus, the observable trajectory of the system is  $y(t) = [x_2(t) \dots x_{10}(t)]^T$ .

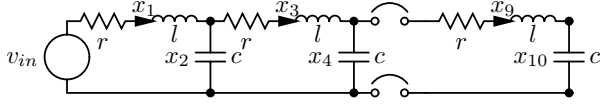


Fig. 1. A ladder network representing a transmission line with 5 sections.

### III. METRIC TEMPORAL LOGIC OVER SIGNALS

Metric Temporal Logic (MTL) was introduced in [14] in order to reason about the quantitative timing properties of boolean signals. In this section, we review the basics of propositional MTL.

*Definition 1 (MTL Syntax):* Let  $AP$  be the set of atomic propositions, then the set  $\Phi_{AP}$  of all well-formed formulas is inductively defined using the following grammar:

$$\phi ::= \top \mid \perp \mid \pi \mid \neg\pi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \mathcal{U}_{\mathcal{I}} \psi \mid \phi \mathcal{R}_{\mathcal{I}} \psi$$

where  $\top$  and  $\perp$  are the constants *true* and *false* respectively,  $\pi \in AP$  and  $\mathcal{I}$  ranges over non-empty intervals of  $\mathbb{R}_{\geq 0}$ .

Sometimes for clarity in the presentation, we replace  $\mathcal{I}$  with pseudometric expressions, e.g.,  $\mathcal{U}_{[0,1]}$  is written as  $\mathcal{U}_{\leq 1}$ . 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  $\mathcal{R}$ . When all the subscripts of the temporal operators are of the form  $[0, +\infty)$ , then the MTL formula  $\phi$  reduces to a Linear Temporal Logic (LTL) formula [6].

In this work, MTL formulas are interpreted over signals. Formally, a *signal*  $s$  is simply a function  $s : R \rightarrow \mathbb{R}^m$  from a time domain  $R$  to some value in the set  $\mathbb{R}^m$ . We define the semantics of MTL using the relation  $\models$ . When we write  $(s, \mathcal{O}) \models \phi$ , we mean that the signal  $s$  satisfies formula  $\phi$  at time 0 under the observation map  $\mathcal{O}$ . If  $s$  does not satisfy formula  $\phi$  at time 0, then we write  $(s, \mathcal{O}) \not\models \phi$ . In the following definition, we also use a time shift operator  $|_t$  on the signals. The signal  $s|_t$  is defined as  $s|_t(t) = s(t'+t)$ .

*Definition 2 (MTL Semantics):* Let  $\phi \in \Phi_{AP}$  be an MTL formula,  $\mathcal{O} : AP \rightarrow \mathcal{P}(\mathbb{R}^m)$  be an observation map and  $s : R \rightarrow \mathbb{R}^m$  be a signal, then the semantics of formula  $\phi$  is defined recursively as

$$\begin{aligned} (s, \mathcal{O}) \models \top, \quad (s, \mathcal{O}) \models \pi \text{ iff } s(0) \in \mathcal{O}(\pi) \\ (s, \mathcal{O}) \not\models \perp, \quad (s, \mathcal{O}) \models \neg\pi \text{ iff } s(0) \notin \mathcal{O}(\pi) \\ (s, \mathcal{O}) \models \phi_1 \vee \phi_2 \text{ iff } (s, \mathcal{O}) \models \phi_1 \vee (s, \mathcal{O}) \models \phi_2 \\ (s, \mathcal{O}) \models \phi_1 \wedge \phi_2 \text{ iff } (s, \mathcal{O}) \models \phi_1 \wedge (s, \mathcal{O}) \models \phi_2 \\ (s, \mathcal{O}) \models \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2 \text{ iff } \exists t_2 \in (\mathcal{I} \cap R). (s|_{t_2}, \mathcal{O}) \models \phi_2 \wedge \\ \quad \wedge \forall t_1 \in (0, t_2). (s|_{t_1}, \mathcal{O}) \models \phi_1 \\ (s, \mathcal{O}) \models \phi_1 \mathcal{R}_{\mathcal{I}} \phi_2 \text{ iff } \forall t_2 \in (\mathcal{I} \cap R). (s|_{t_2}, \mathcal{O}) \models \phi_2 \vee \\ \quad \vee \exists t_1 \in (0, t_2). (s|_{t_1}, \mathcal{O}) \models \phi_1 \end{aligned}$$

Informally, the formula  $\phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$  expresses the property that over the signal  $\sigma$  and in the time interval  $(t + \mathcal{I}) \cap R$ , there exists some time that  $\sigma$  makes  $\phi_2$  true and, furthermore, for all previous times,  $\sigma$  satisfies  $\phi_1$ . The release operator  $\phi_1 \mathcal{R}_{\mathcal{I}} \phi_2$  states that  $\phi_2$  should always hold during the interval  $(t + \mathcal{I}) \cap R$ , a requirement which is released if  $\phi_1$  becomes true. In the above definition, the intersection with the set

$R$  has been added because we also consider time domains which are bounded. We can also define the derived temporal operators *eventually*  $\diamond_{\mathcal{I}} \phi = \top \mathcal{U}_{\mathcal{I}} \phi$  and *always*  $\square_{\mathcal{I}} \phi = \perp \mathcal{R}_{\mathcal{I}} \phi$  which are self-explanatory.

Formally, the language of  $\phi$  under a map  $\mathcal{O}$  is defined as  $\mathcal{L}^{\mathcal{O}}(\phi) = \{s : R \rightarrow \mathbb{R}^m \mid (s, \mathcal{O}) \models \phi\}$  (and similarly for  $\mathcal{L}_T^{\mathcal{O}}(\phi)$  and  $\mathcal{L}_{\infty}^{\mathcal{O}}(\phi)$ ). It is easy to see that the sets  $\mathcal{L}^{\mathcal{O}}(\phi)$  and  $\mathcal{L}^{\mathcal{O}}(\neg\phi)$  form a partition of the underlying signal space.

*Example 2:* The informal requirement of Example 1 can now be captured by the MTL formula  $\psi_1 = \square \pi_1 \wedge \diamond_{\leq 3} \square \pi_2$ , where  $\mathcal{O}(\pi_1) = \{\hat{y} \in \mathbb{R}^5 \mid \wedge_{i=1}^5 |\hat{y}_i| \leq 2\}$  and  $\mathcal{O}(\pi_2) = \{\hat{y} \in \mathbb{R}^5 \mid 0.8 \leq \hat{y}_5 \leq 1.2\}$ .

### IV. ENABLING ROBUST TESTING FOR LPV SYSTEMS

In this section, we present a framework for approximating or reducing an LPV system into an LTI system. The main tool behind our approach is the notion of bisimulation functions [20]. Then, the timing properties of the LTI are tested – or even verified – using our prior work on the MTL verification of time invariant systems [9].

#### A. Approximate Bisimulation Relations

Approximate bisimulation relations [20], as opposed to bisimulation relations [21], do not require that two trajectories are identical, but that they remain always close enough. In the following, we review some of the definitions and results from Girard and Pappas [20] and, then, we construct bisimulation functions for LPV systems.

*Definition 3 (Bisimulation Relation):* A relation  $\mathcal{B}_{\delta} \subseteq \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$  is an approximate bisimulation relation of precision  $\delta$  between two systems  $\Sigma_1$  and  $\Sigma_2$  if for all  $(x_1^0, x_2^0) \in \mathcal{B}_{\delta}$ ,  $\|y_1^0 - y_2^0\| \leq \delta$  and

- 1) For all  $x_1 \in \Lambda(\Sigma_1)$  such that  $x_1(0) = x_1^0$  there exists an  $x_2 \in \Lambda(\Sigma_2)$  such that  $x_2(0) = x_2^0$  and  $\forall t \in R$ ,  $(x_1(t), x_2(t)) \in \mathcal{B}_{\delta}$ .
- 2) For all  $x_2 \in \Lambda(\Sigma_2)$  such that  $x_2(0) = x_2^0$  there exists an  $x_1 \in \Lambda(\Sigma_1)$  such that  $x_1(0) = x_1^0$  and  $\forall t \in R$ ,  $(x_1(t), x_2(t)) \in \mathcal{B}_{\delta}$ .

Two systems  $\Sigma_1$  and  $\Sigma_2$  are bisimilar with precision  $\delta$  (noted  $\Sigma_1 \sim_{\delta} \Sigma_2$ ) if for all  $x_1^0 \in X_1^0$ , there exists  $x_2^0 \in X_2^0$  such that  $(x_1^0, x_2^0) \in \mathcal{B}_{\delta}$  and vice versa.

Informally, Def. 3 states that for each observable trajectory  $y_1$  of  $\Sigma_1$  there exists an observable trajectory  $y_2$  of  $\Sigma_2$  such that their distance at each point in time remains bounded by  $\delta$  and conversely. Another way to think about approximate bisimulation relations is to observe that they provide us with a quantifiable bound on how far away are two systems from being (exactly) bisimilar. Even though approximate bisimulation relations cannot be computed, they can be approximated using the notion of bisimulation functions. The relationship between approximate bisimulation relations and bisimulation functions is that the former can be characterized by the level sets of the latter.

*Definition 4 (Bisimulation Function):* A function  $\mathcal{F} : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}_{\geq 0} \cup \{+\infty\}$  is a bisimulation function between  $\Sigma_1$  and  $\Sigma_2$  if for all  $\delta \geq 0$ ,  $\mathcal{B}_{\delta} = \{(\hat{x}_1, \hat{x}_2) \in$

$\mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mid \mathcal{F}(\hat{x}_1, \hat{x}_2) \leq \delta$  is a  $\delta$ -approximate bisimulation relation between  $\Sigma_1$  and  $\Sigma_2$ .

A result proved in [20] is that we can compute a tight upper bound  $\delta$  such that  $\Sigma_1 \sim_\delta \Sigma_2$  by solving two games.

*Theorem 1:* Let  $\mathcal{F}$  be a bisimulation function between  $\Sigma_1$  and  $\Sigma_2$  and  $\delta \geq \max\{\sup_{x_1^0 \in X_1^0} \inf_{x_2^0 \in X_2^0} \mathcal{F}(x_1^0, x_2^0), \sup_{x_2^0 \in X_2^0} \inf_{x_1^0 \in X_1^0} \mathcal{F}(x_1^0, x_2^0)\}$ . If  $\delta$  has finite a value, then  $\Sigma_1 \sim_\delta \Sigma_2$ .

In the case of autonomous systems with uncertain parameters, as the ones we consider in this paper, a bisimulation function is any function that satisfies the conditions of the following theorem.

*Theorem 2:* Let  $\Sigma_1^u$  and  $\Sigma_2^u$  be two LPV systems with the same observation space. Let  $\mathcal{V} : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}_{\geq 0} \cup \{+\infty\}$  be a continuously differentiable function such that for all  $(\hat{x}_1, \hat{x}_2) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$  we have

$$\mathcal{V}(\hat{x}_1, \hat{x}_2) \geq \|C_1 \hat{x}_1 - C_2 \hat{x}_2\|^2 \quad (2)$$

$$\forall \hat{p}_1 \in P_1 \cdot \forall \hat{p}_2 \in P_2 \cdot \nabla \mathcal{V}(\hat{x}_1, \hat{x}_2) \cdot \begin{bmatrix} A_1(\hat{p}_1) \hat{x}_1 \\ A_2(\hat{p}_2) \hat{x}_2 \end{bmatrix} \leq 0. \quad (3)$$

Then,  $\mathcal{F}(\hat{x}_1, \hat{x}_2) = \sqrt{\mathcal{V}(\hat{x}_1, \hat{x}_2)}$  is a bisimulation function between  $\Sigma_1$  and  $\Sigma_2$ .

Informally, Theorem 2 says that for any system parameters  $\hat{p}_1$  and  $\hat{p}_2$  the distance between the observations of the two systems will remain bounded and, moreover, it will not increase.

### B. Approximating LPV with LTI Systems

The fundamental difficulty in testing an uncertain linear system  $\Sigma^u$  is that we have an uncountable number of parameter values to test both in the set of initial conditions  $X_0$  and in the set of unknown parameters  $P$ . This problem can be alleviated by employing the notion of approximate bisimilarity which was introduced in the previous section.

Let us assume that we are given an uncertain linear system  $\Sigma^u$  and a vector of parameters  $\hat{p}_0$  in  $P$ . This vector could be either the nominal operating point of the system, which is provided along with the model of the system, or we can uniformly sample a point from the set  $P$ . Then, all we need to do is to find an approximate bisimulation function  $\mathcal{F}$  between the systems  $\Sigma^u$  and  $\Sigma^u(\hat{p}_0)$ . If we find one and the game in Theorem 1 returns a  $\delta$  which has a finite value, then we know that  $\Sigma^u(\hat{p}_0)$  can approximate  $\Sigma^u$  with precision  $\delta$  and vice versa. Therefore, we will have essentially reduced the testing problem for  $\Sigma^u$  to testing  $\Sigma^u(\hat{p}_0)$ .

The next Theorem indicates that such a bisimulation function  $\mathcal{F}$  between  $\Sigma^u$  and  $\Sigma^u(\hat{p}_0)$  can be efficiently computed despite the fact we have an uncountable number of parameter values. In the following, the notation  $\mathcal{EP}(Z)$  denotes the extreme points of a (convex and bounded) polytope  $Z$ . In particular, for  $P$  as in Assumption 1, we have  $\mathcal{EP}(P) = \{(w_1, w_2, \dots, w_q) \mid w_i = \underline{p}^i \text{ or } w_i = \bar{p}^i \text{ for all } i = 1, 2, \dots, q\}$ .

*Theorem 3:* Consider a system  $\Sigma^u$  and a vector  $\hat{p}_0 \in P$ . If there exists a positive semidefinite matrix  $M$  such that  $M \geq C_b$  and  $\forall \hat{p} \in \mathcal{EP}(P) \cdot A_b^T(\hat{p})M + MA_b(\hat{p}) \leq 0$ , with

$$C_b = \begin{bmatrix} C^T C & -C^T C \\ -C^T C & C^T C \end{bmatrix} \text{ and } A_b(\hat{p}) = \begin{bmatrix} A(\hat{p}_0) & 0 \\ 0 & A(\hat{p}) \end{bmatrix},$$

then the function  $\mathcal{F}(\hat{x}) = \sqrt{\hat{x}^T M \hat{x}}$  with  $\hat{x} = [\hat{x}_i^T \ \hat{x}_u^T]^T$  is a bisimulation function between  $\Sigma_i = \Sigma^u(\hat{p}_0)$  and  $\Sigma_u = \Sigma^u$ .

The system of equations in Theorem 3 can be efficiently solved with a semidefinite programming solver. Notice, however, that the number of equations increases exponentially with the number of unknown parameters.

If an approximate bisimulation function exists between  $\Sigma^u$  and  $\Sigma^u(\hat{p}_0)$ , then the next step is to compute how well  $\Sigma^u(\hat{p}_0)$  approximates  $\Sigma^u$  by solving the static games in Theorem 1. As the next proposition establishes, the sup-inf optimizations in Theorem 1 can be reduced into a number of quadratic programs.

*Proposition 1:* Consider a system  $\Sigma^u$ ,  $\hat{p}_0 \in P$  and let

$$\mathcal{F}(\hat{x}_i, \hat{x}_u) = \sqrt{\mathcal{V}(\hat{x}_i, \hat{x}_u)} = \sqrt{[\hat{x}_i^T \ \hat{x}_u^T] M [\hat{x}_i^T \ \hat{x}_u^T]^T}$$

be a bisimulation function between  $\Sigma_i = \Sigma^u(\hat{p}_0)$  and  $\Sigma_u = \Sigma^u$ . Then, the solution of the static games

$$\max\{\max_{x_i^0 \in \mathcal{EP}(X_0)} \inf_{x_u^0 \in X_0} \mathcal{V}(x_i^0, x_u^0), \max_{x_u^0 \in \mathcal{EP}(X_0)} \inf_{x_i^0 \in X_0} \mathcal{V}(x_i^0, x_u^0)\} \quad (4)$$

computes the optimal points  $\hat{x}_i^*$  and  $\hat{x}_u^*$  which provide an upper bound  $\delta \geq \delta^* = \mathcal{F}(\hat{x}_i^*, \hat{x}_u^*)$  for Theorem 1.

## V. MTL ROBUST TESTING AND VERIFICATION

In system verification, we are usually interested in answering the question whether each trajectory in  $\mathcal{L}(\Sigma)$  satisfies a temporal logic formula  $\phi$ . In other words, whether  $\mathcal{L}(\Sigma) \subseteq \mathcal{L}^\mathcal{O}(\phi)$ . Note that if the space can be equipped with a metric and the distance between the two sets  $\mathcal{L}(\Sigma)$  and  $\mathcal{L}^\mathcal{O}(\neg\phi)$  is greater than zero (under the assumption that  $\mathcal{L}^\mathcal{O}(\phi)$  and  $\mathcal{L}^\mathcal{O}(\neg\phi)$  form a partition), then the set inclusion  $\mathcal{L}(\Sigma) \subseteq \mathcal{L}^\mathcal{O}(\phi)$  holds.

*Definition 5 (Distance between Sets):* Let  $S_1, S_2$  be subsets of a set  $X$  which is equipped with a metric  $d$ . Then, the distance between  $S_1$  and  $S_2$  is defined as  $\text{dist}_d(S_1, S_2) = \inf\{d(x_1, x_2) \mid x_1 \in S_1, x_2 \in S_2\}$ .

Note that if the two sets intersect, then their distance is zero. With the previous definition at hand, it is straightforward to define the robustness degree of a system with respect to an MTL specification in a similar fashion to the robustness degree of a signal [22].

*Definition 6 (Robustness Degree for Systems):* Given a dynamical system  $\Sigma$ , an MTL formula  $\phi \in \Phi_{AP}$  and a map  $\mathcal{O} : AP \rightarrow \mathcal{P}(\mathbb{R}^m)$ , the robustness degree  $\varepsilon \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$  of  $\Sigma$  with respect to the formula  $\phi$  is  $\varepsilon = \text{dist}_\rho(\mathcal{L}(\Sigma), \mathcal{L}^\mathcal{O}(\neg\phi))$ , where  $\rho$  is the supremum metric :  $\rho(s, s') = \sup_{t \in \mathbb{R}} \{\|s(t) - s'(t)\|\}$ .

According to our definition of robustness, the greater the distance between the sets  $\mathcal{L}(\Sigma)$  and  $\mathcal{L}^\mathcal{O}(\neg\phi)$ , the greater the robustness of the system. Unlike our definition of the robustness degree for signals [22], the robustness degree for systems does not provide us with a measure of how robustly  $\Sigma$  does not satisfy  $\phi$  when  $\mathcal{L}(\Sigma) \subseteq \mathcal{L}^\mathcal{O}(\neg\phi)$ .

Now, assume that we are given two systems  $\Sigma_1$  and  $\Sigma_2$  which are approximately bisimilar with precision  $\delta$ . Then, for

any trajectory in  $\mathcal{L}(\Sigma_1)$ , there exists a trajectory in  $\mathcal{L}(\Sigma_2)$ , such that their distance remains bounded by  $\delta$ . Thus, it is easy to see that for any  $\delta' > \delta$ , we have  $\mathcal{L}(\Sigma_1) \subseteq B_\rho(\mathcal{L}(\Sigma_2), \delta')$ . Here, we use the notation  $B_d(x, \varepsilon) = \{y \in X \mid d(x, y) < \varepsilon\}$  and  $B_d(S, \varepsilon) = \cup_{x \in S} B_d(x, \varepsilon)$  for some  $\varepsilon > 0$ , a point  $x \in X$  and a subset  $S \subseteq X$  of a metric space  $(X, d)$ . Therefore, if  $\varepsilon = \text{dist}_\rho(\mathcal{L}(\Sigma_2), \mathcal{L}^\mathcal{O}(\neg\phi)) > \delta'$ , then we can conclude that  $\Sigma_1$  satisfies  $\phi$  since  $B_\rho(\mathcal{L}(\Sigma_2), \delta') \cap \mathcal{L}^\mathcal{O}(\neg\phi) = \emptyset$ . Formally, we can state the following result.

*Proposition 2:* Given a formula  $\phi \in \Phi_{AP}$  and a map  $\mathcal{O} : AP \rightarrow \mathcal{P}(\mathbb{R}^m)$ , if  $\Sigma_1 \sim_\delta \Sigma_2$  and  $B_\rho(\mathcal{L}(\Sigma_2), \delta) \subseteq \mathcal{L}^\mathcal{O}(\phi)$ , then  $\mathcal{L}(\Sigma_1) \subseteq \mathcal{L}^\mathcal{O}(\phi)$ .

The next question that we have to answer is how do we check the inclusion  $B_\rho(\mathcal{L}(\Sigma_2), \delta) \subseteq \mathcal{L}^\mathcal{O}(\phi)$ ? Instead of trying to compute the set  $\mathcal{L}(\Sigma_2)$  and then expanding it by  $\delta$ , we follow a different approach. Essentially, we give a  $\delta$  robust interpretation to  $\phi$  – let us denote the resulting map by  $\mathcal{O}^\delta$  – and we check whether  $\mathcal{L}(\Sigma_2) \subseteq \mathcal{L}^{\mathcal{O}^\delta}(\phi)$ . The rest of this section briefly describes the related process. A more detailed presentation of the “robustification” in the case of LTL formulas appears in [23].

Intuitively, in order to achieve a robust interpretation of the specification  $\phi$ , we have to define a new observation map  $\mathcal{O}_\delta$  which contracts by  $\delta$  the areas that must be visited and expands by  $\delta$  the regions that must be avoided. In order to classify the atomic propositions in the input MTL formula  $\phi$  according to whether they represent regions that must be reached or avoided, we introduce an *extended set of atomic propositions*  $AP^\varepsilon$ . In detail, we first define two new sets of symbols  $AP^+ = \{\pi_+ \mid \pi \in AP\}$  and  $AP^- = \{\pi_- \mid \pi \in AP\}$  and, then, we set  $AP^\varepsilon = AP^+ \cup AP^-$ . We also define a translation algorithm  $\text{pos} : \Phi_{AP} \rightarrow \Phi_{AP^\varepsilon}$  which takes as input an MTL formula  $\phi$  and it returns a formula  $\text{pos}(\phi)$  where the occurrences of the terms  $\pi$  and  $\neg\pi$  have been replaced by the members  $\pi_+$  and  $\pi_-$  of  $AP^\varepsilon$  respectively.

Since we have a new set of atomic propositions, namely  $AP^\varepsilon$ , we need to define a new map  $\mathcal{O}^\varepsilon : AP^\varepsilon \rightarrow \mathcal{P}(\mathbb{R}^m)$  for the interpretation of the propositions. This is straightforward : for all  $\xi \in AP^\varepsilon$ , we have  $\mathcal{O}^\varepsilon(\xi) := \mathcal{O}(\pi)$  if  $\xi = \pi_+$  and  $\mathcal{O}^\varepsilon(\xi) := X \setminus \mathcal{O}(\pi)$  if  $\xi = \pi_-$ . Then, the following result is immediate from the definition of  $\mathcal{O}^\varepsilon$ .

*Lemma 1:* Given a signal  $s : R \rightarrow \mathbb{R}^m$ , a formula  $\phi \in \Phi_{AP}$  and a map  $\mathcal{O} : AP \rightarrow \mathcal{P}(\mathbb{R}^m)$ , we have  $(s, \mathcal{O}) \models \phi$  iff  $(s, \mathcal{O}^\varepsilon) \models \text{pos}(\phi)$ .

The importance of the previous lemma is the following. Since a formula  $\phi \in \Phi_{AP}$  is equivalent to the formula  $\phi' = \text{pos}(\phi)$  under the maps  $\mathcal{O} : AP \rightarrow \mathcal{P}(\mathbb{R}^m)$  and  $\mathcal{O}^\varepsilon : AP^\varepsilon \rightarrow \mathcal{P}(\mathbb{R}^m)$  respectively, for the rest of the paper we can assume that the input specification is given without any negation operators. That is, the next results are given with respect to a formula  $\phi' \in \Phi_{AP^\varepsilon}$  and a map  $\mathcal{O}^\varepsilon : AP^\varepsilon \rightarrow \mathcal{P}(\mathbb{R}^m)$ . For clarity in the presentation, we denote all MTL formulas without any negation operator using primed Greek letters, e.g.,  $\phi'$ ,  $\phi'_1$ .

At this point, we have distinguished the regions that must be avoided ( $AP^-$ ) and the regions that must be reached ( $AP^+$ ). We proceed to formally define what we mean by

region contraction in order to define our notion of robustness.

*Definition 7 ( $\varepsilon$ -Contraction):* Given a radius  $\varepsilon \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$  and a subset  $S$  of a metric space  $(X, d)$ , the  $\varepsilon$ -contraction of the set  $S$  is defined as  $C_d(S, \varepsilon) = \{x \in X \mid cl(B_d(x, \varepsilon)) \subseteq S\}$ .

Here, the operator  $cl(S)$  denotes the closure of a set  $S$ , that is, the intersection of all closed sets containing  $S$ .

Now, the  $\delta$ -robust interpretation of a given MTL formula  $\phi$  can be achieved by simply introducing a new map  $\mathcal{O}_\delta : AP^\varepsilon \rightarrow \mathcal{P}(\mathbb{R}^m)$ . For a given  $\delta \geq 0$ , the definition of the map  $\mathcal{O}_\delta$  is founded on the map  $\mathcal{O}^\varepsilon$  as follows :  $\forall \xi \in AP^\varepsilon$ ,  $\mathcal{O}_\delta(\xi) := cl(C_d(\mathcal{O}^\varepsilon(\xi), \delta))$ .

The following proposition is the connecting link between the two problems  $B_\rho(\mathcal{L}(\Sigma_2), \delta) \subseteq \mathcal{L}^{\mathcal{O}^\varepsilon}(\phi')$  and  $\mathcal{L}(\Sigma_2) \subseteq \mathcal{L}^{\mathcal{O}_\delta}(\phi')$ . Informally, it states that given  $\delta > 0$ , if a trajectory  $y_2$  of  $\Sigma_2$  satisfies the  $\delta$ -robust interpretation of the input specification  $\phi'$ , then any other trajectory  $y'_2$  that remains  $\delta$ -close  $y_2$  satisfies the same non-robust specification  $\phi'$ .

*Proposition 3:* Consider a formula  $\phi' \in \Phi_{AP^\varepsilon}$ , a map  $\mathcal{O}^\varepsilon : AP^\varepsilon \rightarrow \mathcal{P}(\mathbb{R}^m)$  and a number  $\delta > 0$ , then  $\mathcal{L}(\Sigma) \subseteq \mathcal{L}^{\mathcal{O}_\delta}(\phi')$  implies  $B_\rho(\mathcal{L}(\Sigma), \delta) \subseteq \mathcal{L}^{\mathcal{O}^\varepsilon}(\phi')$  for any dynamical system  $\Sigma$ .

The advantage of Proposition 3 is that the problem  $\mathcal{L}_T(\Sigma) \subseteq \mathcal{L}_T^{\mathcal{O}_\delta}(\phi')$  can be solved using an existing method such as the one presented in [9]. The next result is immediate from Lemma 1 and Propositions 2 and 3.

*Corollary 1:* Given a formula  $\phi \in \Phi_{AP}$  and a map  $\mathcal{O} : AP \rightarrow \mathcal{P}(\mathbb{R}^m)$ , if  $\Sigma_1 \sim_\delta \Sigma_2$  and  $\mathcal{L}(\Sigma_2) \subseteq \mathcal{L}^{\mathcal{O}_\delta}(\text{pos}(\phi))$ , then  $\mathcal{L}(\Sigma_1) \subseteq \mathcal{L}^\mathcal{O}(\phi)$ .

## VI. MTL TESTING OF LPV SYSTEMS AND NUMERICAL RESULTS

The previous two sections outlined the theoretical results that comprise the basic building blocks for a framework for the MTL testing and verification of LPV systems. This section briefly discusses how these results can be put together into a practical and efficient testing algorithm and it concludes with some numerical results.

Assume that we are given an LPV system  $\Sigma^u$  and an MTL formula  $\phi$  (along with the corresponding observation map  $\mathcal{O}$ ). First, we choose a random vector  $\hat{p}_0$  from the set parameter values  $P$ . If a nominal parameter vector  $\hat{p}_0$  has been defined, then we use that instead of a random vector. Then using Theorem 3, we compute a bisimulation function  $\mathcal{F}$  between  $\Sigma^u(\hat{p}_0)$  and  $\Sigma^u$ . If such a bisimulation function exists, then the next step is to determine the accuracy of the approximate bisimulation relation. This is done using Proposition 1. Note that all the above steps can be efficiently computed within MATLAB<sup>®</sup> using the Optimization Toolbox<sup>TM</sup> and SeDuMi [24]. Finally, the resulting problem  $\mathcal{L}_T(\Sigma) \subseteq \mathcal{L}_T^{\mathcal{O}_\delta}(\text{pos}(\phi))$  can be solved using the algorithm from [9]. Next, we present some numerical examples using the prototype MATLAB toolbox that we have developed. All the numerical experiments were performed on a PIII mobile 1.2GHz with 1GB of RAM.

*Example 3:* Consider the Example 1 with parameter values  $r = 2.5$ ,  $l = 1.25$ ,  $c_0 = 3.75 \cdot 10^{-3}$  and  $\beta =$

Instance	1	2	3	4
$\alpha$	0.01	0.03	0.06	0.1
$\delta$	0.1124	0.1150	0.1198	0.1281
safe	✓	✓	✓	✓
# of sim.	1	1	33	1057

TABLE I

APPROXIMATION BOUNDS AND VERIFICATION RESULTS FOR THE PROBLEMS OF EXAMPLE 3.

$2 \cdot 10^{-5}$ . As the nominal parameter value, we pick the vector  $\hat{p}_0 = [0\ 0\ 0\ 0]^T$ . Table I indicates how the approximation  $\delta$  between the LPV system and the corresponding LTI system changes with respect to the parameter  $\alpha$ , i.e., the size of the set of initial conditions  $X_0$ . The computation time for the approximation bound  $\delta$  for this problem size takes about 12 sec in MATLAB. The initial MTL formula  $\psi_1$  is given in Example 2. Table I summarizes the verification results.

The next example demonstrates how the framework can be used for the MTL verification of nonlinear systems.

*Example 4:* Consider the following nonlinear system

$$\begin{aligned}\dot{x}_1(t) &= 0.05 \sin^2(x_2(t))x_1(t) - 2.5x_2(t) \\ \dot{x}_2(t) &= 0.5x_1(t) - x_2(t)\end{aligned}\quad (5)$$

with  $y(t) = x(t)$  and initial conditions  $X_0 = [0.4, 0.8] \times [-0.3, -0.1]$ . The MTL specification we would like to verify is  $\psi_2 = \square\pi_3 \wedge \square_{\geq 8}\pi_4$ , where  $\mathcal{O}(\pi_3) = \mathbb{R} \times [-0.6, 0.6]$  and  $\mathcal{O}(\pi_4) = [-0.4, 0.4] \times [-0.4, 0.4]$ . Notice that for any  $x \in \mathbb{R}$ , we have  $\sin^2(x) \in [0, 1]$ . Thus instead of the nonlinear system, we can verify the following LPV system

$$\dot{x}(t) = \begin{bmatrix} 0.05p(t) & -2.5 \\ 0.5 & -1 \end{bmatrix} x(t)\quad (6)$$

where  $p(t) \in P = [0, 1]$ . Formally speaking, system (6) (exactly) simulates system (5). For the verification, we pick as nominal parameter value  $\hat{p}_0 = 0.5$ . For these values, the upper bound on the approximate bisimulation relation is  $\delta = 0.2125$ , which took about 0.93 sec to compute. The verification process required 13 simulations which took about 6.5 sec of computation time.

## VII. CONCLUSIONS

In this paper, we have presented a framework for the Metric Temporal Logic (MTL) testing and verification of Linear Parameter Varying (LPV) systems. This class of systems includes linear systems with uncertain parameters and can abstract some instances of linear time varying, nonlinear and hybrid systems whose discrete dynamics depend on time.

The main contribution of this paper is the construction of a bisimulation function that enables the approximation of an LPV system by a Linear Time Invariant (LTI) system in an computationally efficient way. Moreover, the results of this paper have been developed in such a way that allow the temporal logic verification step to be performed by any algorithm that can check MTL properties of an LTI system.

In order to deal with more general classes of systems, we will need to integrate the proposed approach with our robust testing framework for hybrid automata [11]. Our future research efforts will be focused along this direction.

## REFERENCES

- [1] J. Tung, "Using model-based design to test auto embedded software," EETimes.com, Sep. 2007.
- [2] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" *J. Comput. Syst. Sci.*, vol. 57, no. 1, pp. 94–124, 1998.
- [3] J. M. Esposito, J. Kim, and V. Kumar, "Adaptive RRTs for validating hybrid robotic control systems," in *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, 2004.
- [4] T. Nahhal and T. Dang, "Test coverage for continuous and hybrid systems," in *CAV*, ser. LNCS, vol. 4590. Springer, 2007, pp. 449–462.
- [5] E. Plaku, L. E. Kavrakı, and M. Y. Vardi, "Hybrid systems: From verification to falsification," in *CAV*, ser. LNCS, W. Damm and H. Hermanns, Eds., vol. 4590. Springer, 2007, pp. 463–476.
- [6] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. Cambridge, Massachusetts: MIT Press, 1999.
- [7] L. Tan, J. Kim, O. Sokolsky, and I. Lee, "Model-based testing and monitoring for hybrid embedded systems," in *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration*, 2004, pp. 487–492.
- [8] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proceedings of FORMATS-FTRTFT*, ser. LNCS, vol. 3253, 2004, pp. 152–166.
- [9] G. E. Fainekos, A. Girard, and G. J. Pappas, "Temporal logic verification using simulation," in *FORMATS*, ser. LNCS, vol. 4202. Springer, 2006, pp. 171–186.
- [10] A. Girard and G. J. Pappas, "Verification using simulation," in *Hybrid Systems: Computation and Control (HSCC)*, ser. LNCS, vol. 3927. Springer, 2006, pp. 272–286.
- [11] A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas, "Robust test generation and coverage for hybrid systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, no. 4416. Springer, 2007, pp. 329–342.
- [12] A. Donzé and O. Maler, "Systematic simulation using sensitivity analysis," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 4416. Springer, 2007, pp. 174–189.
- [13] F. Lerda, J. Kapinski, E. M. Clarke, and B. H. Krogh, "Verification of supervisory control software using state proximity and merging," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 4981. Springer, 2008, pp. 344–357.
- [14] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990.
- [15] F. Amato, *Robust Control of Linear Systems Subject to Uncertain Time-Varying Parameters*. Springer, 2006.
- [16] A. Girard and G. J. Pappas, "Approximate bisimulation relations for constrained linear systems," *Automatica*, vol. 43, no. 8, pp. 1307–1317, 2007.
- [17] L. Hedrich and E. Barke, "A formal approach to verification of linear analog circuits with parameter tolerances," in *DATE*. Washington, DC, USA: IEEE Computer Society, 1998, pp. 649–655.
- [18] S. Little, D. Walter, K. Jones, and C. J. Myers, "Analog/mixed-signal circuit verification using models generated from simulation traces," in *Proceedings of the 5th ATVA*, ser. LNCS, vol. 4762. Springer, 2007, pp. 114–128.
- [19] C. S. Indulkar, "State space analysis of a ladder network representing a transmission line," *International Journal of Electrical Engineering Education*, vol. 42, no. 4, pp. 383–392, 2005.
- [20] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Trans. Auto. Cont.*, vol. 52, no. 5, pp. 782–798, 2007.
- [21] G. J. Pappas, "Bisimilar linear systems," *Automatica*, vol. 39, no. 12, pp. 2035–2047, December 2003.
- [22] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications," in *FATES and RV*, ser. LNCS, vol. 4262. Springer, 2006, pp. 178–192.
- [23] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, Feb. 2009.
- [24] J. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11–12, pp. 625–653, 1999.