Proving Abstractions of Dynamical Systems through Numerical Simulations

Sayan Mitra*

mitras@illinois.edu
Coordinate Science Laboratory
University of Illinois at Urbana Champaign
Urbana, IL 61801

ABSTRACT

A key question that arises in rigorous analysis of cyberphysical systems under attack involves establishing whether or not the attacked system deviates significantly from the ideal allowed behavior. This is the problem of deciding whether or not the ideal system is an abstraction of the attacked system. A quantitative variation of this question can capture how much the attacked system deviates from the ideal. Thus, algorithms for deciding abstraction relations can help measure the effect of attacks on cyberphysical systems and to develop attack detection strategies. In this paper, we present a decision procedure for proving that one nonlinear dynamical system is a quantitative abstraction of another. Directly computing the reach sets of these nonlinear systems are undecidable in general and reach set over-approximations do not give a direct way for proving abstraction. Our procedure uses (possibly inaccurate) numerical simulations and a model annotation to compute tight approximations of the observable behaviors of the system and then uses these approximations to decide on abstraction. We show that the procedure is sound and that it is guaranteed to terminate under reasonable robustness assumptions.

Keywords

cyberphysical systems, adversary, simulation, verification, abstraction.

1. INTRODUCTION

Cyberphysical systems can take the form of anti-lock braking systems in cars, process control systems in factories, and nation-scale networked control systems for traffic, water, and power. Security breaches in cyberphysical systems can be disastrous and expensive. Aside from the obvious social motivation, an inquiry into the security of cyberphysical systems is also propelled by new scientific questions about design and analysis of computing systems that sense and control the physical world. Since these computing systems are embedded in the physical world (a) they require preservation of dynamical properties that cannot be characterized purely in terms of software state, and (b) they can be breached in ways that go beyond vulnerabilities that are exploited in stand alone computing systems. While the dynamical operation remains vulnerable to full-fledged attacks on its computing and the communication components—for instance, a denial-of-service-attack on the computers controlling the power grid can take it down—it is also vulnerable to more elusive *dynamics-aware attacks* that subtly change local behaviors in ways that lead to instability, unsafe behavior, and a loss of availability of the system. In this paper, we present new results that contribute towards developing a framework for analyzing security properties of cyberphysical under different classes of attacks.

Role of Models and Abstractions. As software plays an increasingly important role in the implementation of cyberpysical systems, it is becoming harder to obtain a high level of assurance from these systems using only hardware/software-in-the-loop tests. The model-based design and validation approach is seen as a promising alternative to this approach and indeed it is being adopted by the automotive and aerospace industries. In this model-based approach, design of control software begins with a mathematical model for the underlying physical process [20, 23] described in the language of ordinary differential equations (ODEs). Our framework is designed for analyzing models of cyberphysical systems that combine these ODEs with automaton models that are used for representing computations [19].

A model B is said to be an *abstraction* of another model A if every observable behavior of A is also an observable behavior of B [9, 19]. If A and B are abstractions of each other then sometimes they are said to be observationally equivalent. This notion of abstraction is related to the notions of bisimilarity, equivalence, and implementation used elsewhere in the literature. Abstractions are central to analysis for two different reasons. First, for a given system A, the abstract model B can be used to capture a set of requirements. For example, the safety requirement that "Alarm must go off 6 seconds before car gets within $4m$ of obstacle even if the position sensors are jammed" can be modeled by an abstract

*This work is supported by an NSA Science of Security Lablet (Grant number W911NSF-13-0086).

HotSoS '14, April 08 - 09 2014, Raleigh, NC, USA
ACM 978-1-4503-2907-1/14/04 $15.00.
http://dx.doi.org/10.1145/2600176.2600188

automaton B. Then establishing that B is an abstraction of A implies that each behavior of A is also a behavior of B, that is, A satisfies the above property. Secondly, since an abstraction of A has more behaviors (than A) it often has a simpler description. For example, the abstract model B may not encode the constraints (communication delays, sensor inaccuracies) that come from the specific implementation platform. This simpler description often makes it possible to establish invariant and temporal properties of B effectively, and in some cases automatically (see, for example [2, 26, 28]). And if the property in question is preserved under abstraction, it follows immediately that A also satisfies these properties. This also enables us to substitute A with the simpler model B when we are analyzing a lager system containing A in which only these properties of A are relevant.

For models with continuous dynamics, it is makes sense to relax the notion of abstraction using a metric on the observable behaviors [14, 25]. B is said to be a c-abstraction of A, for some positive constant c, if every observable behavior of A is within c distance of some observable behavior of B, where the distance is measured by some metric on the observables. In this paper we also look at time-bounded versions of abstraction. B is a c-abstraction of A up to time T, if every observable behavior of A of duration T is within c distance of some observable behavior of B (also of duration T).

We can state properties about cyberphysical systems under attack with this relaxed notion. Let B be the nominal model (without any attack) and A_1 be a model of the system under attack 1. If we can prove that B is a c-abstraction of A_1 then it follows that none of the observable behaviors deviate more that c under attack 1. This gives a systematic way of classifying attacks with respect to their impact on deviation from ideal behavior. If B is *not* a c-abstraction of A_2—the model of the system under attack 2—then it follows that attack 2 is worse than attack 1 in the sense that it causes a larger deviation from the ideal. A c-abstraction relation can also be used for reasoning about attack detectability and distinguishability. If B is a c-abstraction of A_1 but our attack sensing mechanisms can only detect deviations in observable behavior from B that are greater than c, then attack 1 will go undetected. If B is also a c-abstraction of A_2, then the same detection mechanism will also fail to distinguish the two attacks.

The above discussion illustrates that many questions related to security and attacks can be formulated in terms of whether or not a model B is an (relaxed) abstraction of another model A. A building-block for our analytical framework is a semi-decision procedure for answering precisely this type of queries. The procedure is sound, that is, whenever it terminates with an answer (c-abstraction or not) the answer is correct. It is a semi-decision procedure because it is guaranteed to terminate, whenever the pair of models satisfy or violate the query robustly. Specifically, our contributions are:

(a) We formalize this quantitative notion of abstraction for models of dynamical systems as the maximum distance from any trace of the concrete model A_1 to some trace of the abstract model A_2.

(b) For nonlinear ODE models, we present a semi-decision procedure for deciding if A_2 is a c-abstraction of A_2 up to a time bound, for any positive constant c. We show that the procedure is sound and it is guaranteed to terminate if either A_2 is at least a $\frac{c}{2}$-abstraction of A_1 or if there exists a trace of A_1 that is more than $2c$ distance away from all traces of A_2.

(c) This semi-decision procedure and some of our earlier works for reachability [13] use representations of reach sets of models. One of the contributions of this paper is the formalization of a natural data structure called *pipes* to represent simulation traces and reachable sets and identifying some of its key properties.

(d) We present a procedure for computing over-approximations of unbounded time reach sets of individual models.

Checking equivalence of two finite state machines—arguably the simplest class of models—is well-known to be decidable. The problem was shown to be decidable for deterministic push-down automata in the celebrated paper [27]. The same problem becomes undecidable for finite state transducers [16] and nondeterministic pushdown automata. For infinite state models that naturally capture computation and physics, such as timed and hybrid automata [1, 19, 28], not only is equivalence checking undecidable, but so is the conceptually simpler problem of deciding if a single state can be reached by a given automaton. For models described by nonlinear ODEs, exactly computing the state reached from a single initial state at a given time is itself a hard problem. A sequence of recent results [6, 10, 11, 13, 17, 18] circumvent these negative results by taking a more practical view of the reachability problem. Specifically, they aim to compute over-approximations of the reach set over a bounded-time horizon. Although some of these procedures require additional annotations of the models and provide weaker soundness and completeness guarantees, they point towards a practical way forward in automatically analyzing reachability properties of moderately complex cyberphysical systems. The key characteristic of these approaches is that they combine static model information (e.g. the differential equations and the text of the program, and not solutions or program runs) with dynamic information (e.g., possibly inaccurate numerical simulations or data from runtime logs), to compute precise over-approximations of bounded time reach sets. This static-dynamic analysis approach takes advantage of both static analysis techniques like propagating reach sets with dynamically generates information.

This paper takes this static-dynamic analysis approach to checking abstraction relations. If an over-approximation of the reach set of A is close to an over-approximation of the reach set of B, this means that every behavior ν of A is close to some over-approximation of B. But, this *does not* imply that $\mathbf{u}$ is close to some actual behavior of B. Our procedure (Algorithm 1) therefore has to take into account the precision of the over-approximations of A and B in deciding that each behavior of A is indeed close to some (or far from all) behavior(s) of B. We also present a fixpoint procedure (Algorithm 2) which uses this static-dynamic approach to compute unbounded time reach sets. For the sake

of simplifying presentations, in this paper we presented the results for models of nonlinear dynamical systems, but these results can be extended to switched systems [21] in a more or less straightforward fashion. Switched systems can capture commonly used time-triggered control systems which cover a large fraction of practical cyberphysical systems. Analogous extensions for reachability algorithms have been presented in [13]. The extension to hybrid models which can capture event-triggered interaction of software and continuous dynamics will be presented in a future paper.

1.1 The Science

> *"I have observed stars of which the light, it can be proved, must take two million years to reach the earth."* —Sir William Herschel, British astronomer and telescope builder, having identified Uranus (1781), the first planet discovered since antiquity.

This paper presents a piece of mathematical machinery (the semi-decision procedure) that is needed for rigorous analysis of cyberphysical systems under different attacks. This procedure can be seen as a *scientific instrument* that enables new types of attack impact measurements. As we discussed above, abstraction is a central concept in any formal reasoning framework. Abstraction relations in their quantitative form can be used to bound the distance from the set of observable behavior of one system to the set of observable behaviors of another (ideal) system. Thus, abstractions can give approximate measures of the deviation of an implementation from an idealized specification. Such measures can aid in the systematic evaluation of the effects of an attack and in gaining understanding of different classes of attacks. In summary, the static-dynamic analysis techniques and specifically the semi-decision procedure presented in this paper can be seen as humble measuring instruments, but ones that could catalyze the science of security for CPS.

2. DYNAMICAL SYSTEMS

In this section, we present the modeling framework and some technical background used throughout the paper. Some of the standard notations are left out for brevity. We refer the reader to the Appendix A for details.

In this paper, we focus on models of dynamical systems with no inputs. Such models are also called autonomous or closed. An autonomous dynamical system is specified by a collection of ordinary differential equations (ODEs), an output mapping, and a set of initial states.

DEFINITION 1. *An (n, m)-dimensional* autonomous dynamical system *A is a tuple $\langle \Theta, f, g \rangle$ where*

(i) $\Theta \subseteq \mathbb{R}^n$ *is a compact set of* initial states*; $\mathbb{R}^n$ is the state space and it's elements are called* states*.*

(ii) $f : \mathbb{R}^n \to \mathbb{R}^n$ *is a Lipschitz continuous function called the* dynamic mapping*.*

(iii) $g : \mathbb{R}^n \to \mathbb{R}^m$ *is a Lipschitz continuous function called the* output mapping*. The output dimension of the system is m.*

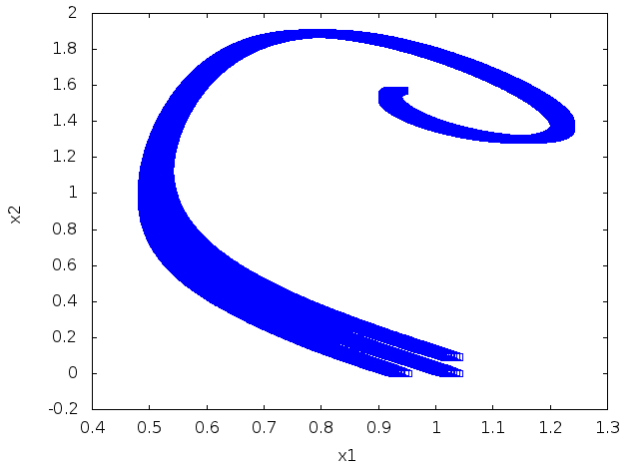

Figure 1: Reachable states and traces of the dynamical system in Example 1.

For a given initial state $\mathbf{x} \in \Theta$, and a time duration $T \in \mathbb{R}_{\geq 0}$, a *solution* (or trajectory) of A is a pair of functions $(\xi_{\mathbf{x}}, \nu_{\mathbf{x}})$: a state trajectory $\xi : [0, T] \to \mathbb{R}^n$ and an output trajectory $\nu : [0, T] \to \mathbb{R}^m$, such that (a) $\xi_{\mathbf{x}}$ satisfies (a) $\xi_{\mathbf{x}}(0) = \mathbf{x}$, (b) for any $t \in [0, T]$, the time derivative of $\xi_{\mathbf{x}}$ at t satisfies the differential equation:

$$\dot{\xi}_{\mathbf{x}}(t) = f(\xi_{\mathbf{x}}(t)), \tag{1}$$

And, (b) at each time instant $t \in [0, T]$, the output trajectory satisfies:

$$\nu_{\mathbf{x}}(t) = g(\xi_{\mathbf{x}}(t)). \tag{2}$$

Under the Lipschitz assumption (iii), the differential equation (1) admits a unique state trajectory defined by the initial state $\mathbf{x}$ which in turn defines the output trajectory. When the initial state is clear from context, we will drop the suffix and write the trajectories as ξ and ν. Given a state trajectory ξ over $[0, T]$, the corresponding output trajectory or *trace* is defined in the obvious way as $\nu(t) = g(\xi(t))$, for each $t \in [0, T]$. The same trace ν, however, may come from a set of state trajectories. The set of all possible state trajectories and output trajectories of A (from different initial states in Θ) are denoted by Execs_A and Traces_A, respectively. A state $\mathbf{x} \in \mathbb{R}^n$ is said to be reachable if there exists $\mathbf{x}' \in \Theta$ and $t \in \mathbb{R}_{\geq 0}$ such that $\xi_{\mathbf{x}'}(t) = \mathbf{x}$. The set of all reachable states of A is denoted by Reach_A. Variants of these notations are defined in the Appendix A.

Example. We define a $(2, 2)$-dimensional dynamical system. The set of initial states is defined by the rectangle $\Theta = [0.9, 0.95] \times [1.5, 1.6]$. The dynamic mapping is the nonlinear vector valued function:

$$f(x_1, x_2) = [1 + x^2 y - 2.5x, -x^2 y + 1.5x].$$

And the output mapping is the vector valued identity function $g(x_1, x_2) = [x_1, x_2]$. An over-approximation of the set of reachable states upto 10 time units (computed using the algorithm describes in [13]) is shown in Figure 1.

Trace metrics. We define a metric on d the set of traces of the same duration and dimension. Given two traces ν_1, ν_2

of duration T and dimension m, we define

$$d(\nu_1, \nu_2) = \sup_{t \in [0,T]} |\nu_1(t) - \nu_2(t)|.$$

The distance from a set of traces N_1 to another set N_2 (with members of identical duration and dimension) is defined by the one-sided Hausdorff distance d_H from N_1 to N_2.

DEFINITION 2. *Given two autonomous dynamical systems A_1 and A_2 of identical output dimensions, a positive constant $c > 0$ and a time bound $T > 0$, A_2 is said to be* c-abstraction *of A_1 upto time T, if*

$$d_H(\mathsf{Traces}_{A_1}(T), \mathsf{Traces}_{A_2}(T)) \leq c.$$

We write this as $A_1 \preceq_{c,T} A_2$.

Thus, if A_2 is a c-abstraction of A_1, then for every output trace ν_1 of A_1 there exists another output trace of A_2 which is differs from ν_1 at each point in time by at most c. Since, the definition only bounds the one-sided Hausdorff distance, every trace of A_2 may not have a neighboring trace of A_1. With $c = 0$, we recover the standard notion of abstraction, that is, $\mathsf{Traces}_{A_1} \subseteq \mathsf{Traces}_{A_2}$. The next set of results follows immediately from the definitions and triangle inequality.

PROPOSITION 2.1. *Let A_1, A_2 and A_3 be dynamical systems of identical output dimensions and c, c', T be positive constants.*

1. *If $A_1 \preceq_{c,T} A_2$ then for any $c_1 \geq c$ and $T_1 \leq T$ $A_1 \preceq_{c_1,T_1} A_2$.*
2. *If $A_1 \preceq_{c,T} A_2$ and $A_2 \preceq_{c',T} A_3$ then $A_1 \preceq_{c+c',T} A_3$.*

The decision problem. The decision problem we solve in this paper takes as input a pair of autonomous dynamical systems A_1 and A_2 with identical output dimensions, annotations for these systems (namely, discrepancy functions which are to be defined in what follows), a constant c and a time bound T, and decides if $A_1 \preceq_{c,T} A_2$. The computations performed by our algorithm uses *pipes* to represent sets of executions and traces. In the next subsection, we define pipes and their properties.

2.1 Working with Pipes

Pipes are used to represent sets of bounded traces and executions. An n-dimensional *segment* is a pair (P, t) where P_i is a subset of $\mathbb{R}^n$ and t is a nonnegative real number. An n-dimensional *pipe* is a sequence of *segments*

$$\Pi = (P_0, t_0), (P_1, t_1), \ldots, (P_k, t_k),$$

where fore each i in the sequence, $t_i > t_{i-1}$. The *duration* of the pipe is $\Pi.dur = t_k$ and its *length* is the number of segments $\Pi.len = k + 1$.

The semantics of a pipe Π is defined once we fix a dynamical system A. It is the set of executions (or traces) of A of duration t_k defined as:

$$[\![\Pi]\!]_A = \{\xi \in \mathsf{Execs}_A \mid \forall\, t \in [0, t_0], \xi(t) \in P_0, \\ \forall\, 1 \leq i \leq \Pi.len, t \in [t_{i-1}, t_i], \xi(t) \in P_i\}.$$

Our algorithms use pipes with finite representation—the sets P_i's are compact sets represented by polyhedra.

We define $dia(\Pi) = \max_{i \in [\Pi.len]} dia(P_i)$ as the maximum diameter of any of the segments. We say that two pipes Π and Π' are *comparable* if they have the same duration, length, and dimension and furthermore, for each $i \in [\Pi.len]$, $t_i = t'_i$. For two comparable pipes Π, Π', we say that Π is contained in Π', denoted by $\Pi \subseteq \Pi'$, iff for each $i \in [\Pi.len]$, $P_i \subseteq P'_i$. The distance from Π to Π' is defined in the natural way by taking the maximum distance from the corresponding segments of Π to those of Π'.

$$d_H(\Pi, \Pi') = \max_{i \in [\Pi.len]} d_H(P_i, P'_i).$$

Obviously, $\Pi \subseteq \Pi'$ implies that $d_H(\Pi, \Pi') = 0$.

We say that two pipes are *disjoint*, denoted by $\Pi \cap \Pi' = \varnothing$, if and only if for each $i \in [\Pi.len]$, the corresponding segments are disjoint. That is, $P_i \cap P'_i = \varnothing$. The following straightforward propositions relate properties of pipes and the sets of executions (or traces) they represent.

PROPOSITION 2.2. *Consider two comparable pipes Π_1, Π_2. If $\Pi_1 \subseteq \Pi_2$ then for any dynamical system A $[\![Pi_1]\!]_A \subseteq [\![Pi_2]\!]_A$.*

PROPOSITION 2.3. *Consider two comparable pipes Π_1, Π_2. If $d_H(\Pi_1, \Pi_2) \geq c$ then for any two automata A and B, $d_H([\![Pi_1]\!]_A, [\![Pi_2]\!]_B) \geq c$.*

2.2 Discrepancy Functions

Our decision procedure for c-abstractions will use numerical simulations (defined in Section 3) and model annotations called *discrepancy functions*. Here we recall the definition of discrepancy functions which were introduced in [13]. In that earlier paper we showed that with discrepancy functions and numerical simulators we can obtain sound and relatively complete decision procedures for safety verification of nonlinear and switched dynamical system. Moreover, the software implementation of this approach proved to be scalable [12] .

Informally, a discrepancy function gives an upper bound on the distance between two trajectories as a function of the distance between their initial states and the time elapsed.

DEFINITION 3. *A smooth function $V : \mathbb{R}^{2n} \to \mathbb{R}_{\geq 0}$ is called a* discrepancy function *for an (n, m)-dimensional dynamical system if and only if there are functions $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$ and a uniformly continuous function $\beta : \mathbb{R}^{2n} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ with $\beta(\mathbf{x}_1, \mathbf{x}_2, t) \to 0$ as $|\mathbf{x}_1 - \mathbf{x}_2| \to 0$ such that for any pair of states $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$:*

$$\mathbf{x}_1 \neq \mathbf{x}_2 \iff V(\mathbf{x}_1, \mathbf{x}_2) > 0, \quad (3)$$

$$\alpha_1(|\mathbf{x}_1 - \mathbf{x}_2|) \leq V(\mathbf{x}_1, \mathbf{x}_2) \leq \alpha_2(|\mathbf{x}_1 - \mathbf{x}_2|) \text{ and} \quad (4)$$

$$\forall\, t > 0,\ V(\xi_{\mathbf{x}_1}(t), \xi_{\mathbf{x}_2}(t)) \leq \beta(\mathbf{x}_1, \mathbf{x}_2, t), \quad (5)$$

A tuple $(\alpha_1, \alpha_2, \beta)$ satisfying the above conditions is called a *witness* to the discrepancy function V. By discrepancy

function of a dynamical system we will refer to V as well as its witness interchangeably. Note that the output dimension m has no bearing on the discrepancy function of the system. The first condition requires that the function $V(\mathbf{x}_1, \mathbf{x}_2)$ vanishes to zero if and only if the first two arguments are identical. The second condition states that the value of $V(\mathbf{x}_1, \mathbf{x}_2)$ can be upper and lower-bounded by functions of the ℓ^2 distance between $\mathbf{x}_1$ and $\mathbf{x}_2$. The final, and the more interesting, condition requires that the function V applied to trajectories of A at a time t from a pair of initial states is upper bounded and converges to 0 as $\mathbf{x}_1$ converges to $\mathbf{x}_2$.

For linear dynamical systems, discrepancy functions can be computed automatically by solving Lyapunov like equations, and in [13] several strategies for proposed for nonlinear systems. Existing notions such as and Lipschitz constants, contraction metrics [22], and incremental Lyapunov functions [3,4,15] all yield discrepancy functions of varying quality.

3. A SEMI-DECISION PROCEDURE FOR ABSTRACTION

Our semi-decision procedure (Algorithm 1) for c-abstractions will use numerical simulations of the dynamical systems. Given a closed dynamical system A and a particular initial state $\mathbf{x} \in \Theta$, for a step size $\tau > 0$, validated ODE solvers (such as [7,8,24]) compute a sequence of boxes (more generally polyhedra) $R_0, R_2, \ldots, R_k \subseteq \mathbb{R}^n$, such that for each $j \in [k]$, $t \in [(k-1)\tau, k\tau]$, $\xi_{\mathbf{x}}(t) \in R_k$. For a desired error bound $\epsilon > 0$, by reducing the step size τ, the diameter of R_k can be made smaller than ϵ. We define such simulations in terms of pipes below.

Definition 4. *Given a dynamical system A, an initial state $\mathbf{x}$, a time bound T, an error bound $\epsilon > 0$, and time step $\tau > 0$, a $(\mathbf{x}, T, \epsilon, \tau)$-*simulation pipe *is a finite sequence $\phi = (R_0, t_0), (R_1, t_1), \ldots, (R_k, t_k)$ where*

(i) $t_0 = 0$, $t_k = T$, and $\forall\ j \in [k]$, $t_j - t_{j-1} \leq \tau$,

(ii) $\forall j \in [k]$ and $\forall t \in [t_{j-1}, t_j]$, $\xi_{\mathbf{x}}(t) \in R_j$, and

(iii) $\forall j \in [k]$, $dia(R_j) \leq \epsilon$.

Algorithm 1 makes subroutine calls to a *Simulate* function with these parameters which then returns a pipe with the above properties.

The simulation pipe is then bloated using the discrepancy function of the dynamical system as follows.

Definition 5. *Let $sim = (R_0, t_0), (R_1, t_1), \ldots, (R_k, t_k)$ be a $(\mathbf{x}, T, \epsilon, \tau)$-simulation pipe for a dynamical system A. Suppose V be a discrepancy function of A with witness $(\alpha_1, \alpha_2, \beta)$. Then, for $\delta > 0$, $Bloat(sim, \delta, V)$ is defined as the pipe $(P_0, t_0), \ldots, (P_k, t_k)$ such that for each $j \in [k]$,*

$$P_j = \{\mathbf{x}_1 \mid \exists\ \mathbf{x}_2 \in R_j\ \wedge V(\mathbf{x}_1, \mathbf{x}_2) \leq e_j\},$$

where

$$e_j = \sup_{t \in [t_{j-1}, t_j], \mathbf{x}' \in B_\delta(\mathbf{x})} \beta(\mathbf{x}, \mathbf{x}', t).$$

In other words, e_j is an upper-bound on the value of V for two executions $\xi_{\mathbf{x}}$ and $\xi_{\mathbf{x}'}$ starting from within $B_\delta(\mathbf{x})$ over the time interval $[t_{j-1}, t_j]$. And P_j bloats R_j to include all states $\mathbf{x}_1$ for which there exists a state $\mathbf{x}_2$ in R_j with the discrepancy function bounded by e_j. Our algorithm makes subroutine calls to a *Bloat* function which takes a simulation pipe, the function β and the constant δ and returns the pipe $(P_0, t_0), \ldots, (P_k, t_k)$ defined above.

Algorithm 1: Deciding c-abstractions.

input: $\mathcal{A}_1, V_1, \mathcal{A}_2, V_2, T, c$

$Init \leftarrow \Theta_1$;
$\delta \leftarrow \delta_0; \tau \leftarrow \tau_0; \epsilon \leftarrow \epsilon_0$;
while $Init \neq \varnothing$ **do**
  $X_1 \leftarrow Partition(Init, \delta)$;
  $X_2 \leftarrow Partition(\Theta_2, \delta)$;
  for $\mathbf{x}_{10} \in X_1, \mathbf{x}_{20} \in X_2$ **do**
    $sim[\mathbf{x}_{10}] \leftarrow Simulate(A_1, \mathbf{x}_{10}, \epsilon, T, \tau)$;
    $pipe[\mathbf{x}_{10}] \leftarrow Bloat(sim[\mathbf{x}_{10}], \delta, V_1)$;
    $sim[\mathbf{x}_{20}] \leftarrow Simulate(A_2, \mathbf{x}_{20}, \epsilon, T, \tau)$;
    $pipe[\mathbf{x}_{20}] \leftarrow Bloat(sim[\mathbf{x}_{20}], \delta, V_2)$;
  end
  foreach $\mathbf{x}_{10} \in X_1$ **do**
    if $\exists\ \mathbf{x}_{20} \in X_2, d_H(pipe[\mathbf{x}_{10}], pipe[\mathbf{x}_{20}]) \leq \frac{c}{L_g}$ $\wedge dia(pipe[\mathbf{x}_{10}]) \leq \frac{c}{2L_g}$ $\wedge dia(pipe[\mathbf{x}_{20}]) \leq \frac{c}{2L_g}$ **then**
      $Init \leftarrow Init \setminus B_\delta(\mathbf{x}_{01})$;
    else if $\forall\ \mathbf{x}_{20} \in X_2, d_H(pipe[\mathbf{x}_{10}], pipe[\mathbf{x}_{20}]) \geq \frac{c}{S_g}$ $\wedge dia(pipe[\mathbf{x}_{10}]) \leq \frac{c}{2S_g}$ **then**
      return *COUNTEREX* $\mathbf{x}_{10}, \delta$;
    else
      $\delta \leftarrow \frac{\delta}{2}; \tau \leftarrow \frac{\tau}{2}; \epsilon \leftarrow \frac{\epsilon}{2}$;
    end
  end
end
return *c-ABSTRACTION*

3.1 Description of the Algorithm

Inside the while loop of Algorithm 1, first, two δ-covers are computed for *Init*—a subset of the initial states Θ_1 of A_1, and the set of initial states Θ_2 of A_2 Next, in the first for loop, for each of the states $\mathbf{x}_{10} \in X_1$ and $\mathbf{x}_{20} \in X_2$ in the respective covers, a $(\mathbf{x}_{i0}, T, \epsilon, \tau)$-simulation pipe $sim[\mathbf{x}_{i0}]$ is computed. Then this pipe is bloated with the parameter δ and the corresponding discrepancy function V_i. The following proposition summarizes the main property of the bloated pipes.

Proposition 3.1. *For the dynamical system $\mathcal{A}_i, i \in \{1, 2\}$ and constants δ, ϵ, τ and T, $\mathsf{Execs}_{A_i}(B_\delta(\mathbf{x}_{i0}), T) \subseteq [\![pipe[\mathbf{x}_{i0}]]\!]$.*

Proof. Let $sim[\mathbf{x}_{i0}] = (R_0, t_0), \ldots, (R_k, t_k)$ and $pipe[\mathbf{x}_{i0}] = (P_0, t_0), \ldots, (P_k, t_k)$. We fix an initial state $\mathbf{x}' \in B_\delta(\mathbf{x}_{i0})$, and show that for any $t \leq t_k$, the state $\xi'_{\mathbf{x}}(t)$ is contained in the set P_j, where $t_{j-1} \leq t \leq t_j$. Let us fix t, which also fixes t_{j-1} and t_j. From the definition of the *Simulation* (Definition 4) function we know that $\xi_{\mathbf{x}_0}(t) \subseteq R_j$. And from Definition 5, we know that since $\mathbf{x}' \in B_\delta(\mathbf{x}_{i0})$, the $V(\xi_{\mathbf{x}'}(t), \xi_{\mathbf{x}_{i0}}(t)) \leq \beta(\mathbf{x}', \mathbf{x}_{i0}, t)$ and therefore $\xi_{\mathbf{x}'}(t) \in P_j$. □

COROLLARY 3.2. *For the dynamical system $\mathcal{A}_i, i \in \{1,2\}$ and constants δ, ϵ, τ and T,*

$$\mathsf{Reach}_{A_i}(\Theta_i, T) \subseteq \bigcup_{\mathbf{x}_{i0} \in X_i} \bigcup_{j \in [T/\tau]} pipe[\mathbf{x}_{i0}].P_j,$$

here we use $pipe[\mathbf{x}_{i0}].P_j$ to denote the subset of $\mathbb{R}^n$ in the j^{th} segment of the pipe $pipe[\mathbf{x}_{i0}]$.

Every time a new set of bloated simulation pipes are computed, $pipe[\mathbf{x}_{10}]$ for each $\mathbf{x}_{10} \in Init$ and $pipe[\mathbf{x}_{20}]$ for each $\mathbf{x}_{10} \in X_2$, the algorithm performs the following checks. If there exists a $pipe[\mathbf{x}_{10}]$ and a $pipe[\mathbf{x}_{20}]$, both less than $c/2L_g$ in diameter and within c/L_g distance then $B_\delta(\mathbf{x}_{10})$ is eliminated from *init*. Here L_g is the Lipschitz constant and S_g is the sensitivity constant of the common output function g. If there exists a $pipe[\mathbf{x}_{10}]$ such that for all the $pipe[\mathbf{x}_{20}]$'s, $\mathbf{x}_{20} \in X_2$, the diameter of the first is less than $c/2S_g$ and they are at least c/S_g distance away from each other, then $\mathbf{x}_{10}$ (and δ) is produced as a counter-example to the c-abstraction. The while loop ends when *Init* becomes empty.

3.2 Soundness and Termination of Algorithm

In this section, we prove the correctness of the algorithm. We assume that the output mappings (the function g) is the same for the two models.

THEOREM 3.3. *For automata with identical observation mappings, the algorithm is sound.*
That is, if the output is c-ABSTRACTION, then, A_2 is a c-abstraction of A_1 upto time T, and if the output is $(COUNTEREX, \mathbf{x}_{10}, \delta)$ then A_2 is not a c-abstraction of A_1. In the latter case, all the traces of A_1 corresponding to executions starting from $B_\delta(\mathbf{x}_0)$ are at least c distance away from any trace of A_2.

PROOF. For the first part, assume that the algorithm returns *c-ABSTRACTION* and we will show that for any initial state $\mathbf{x} \in \Theta_1$, there exists an initial state $\mathbf{x}' \in \Theta_2$ such that $d(\nu_{\mathbf{x}}, \nu'_{\mathbf{x}'}) \leq c$. Here $\nu_{\mathbf{x}}$ is the output trace of A_1 from $\mathbf{x}$ and ν' is the output trace of A_2 from $\mathbf{x}'$.

The algorithm returns *c-ABSTRACTION* only when *Init* becomes empty. This occurs when each initial state $\mathbf{x} \in \Theta_1$ of A_1 is in the δ-ball of some state $\mathbf{x}_{10} \in \Theta_1$ such that $\mathbf{x}_{10}$ is in a cover X_1 and satisfies the condition in Line 13. It suffices to show that this condition $d_H(pipe[\mathbf{x}_{10}], pipe[\mathbf{x}_{20}]) \leq c/L_g$ implies that there exists $\mathbf{x}' \in \Theta_2$, $d(\nu_{\mathbf{x}}, \nu'_{\mathbf{x}'}) \leq c$.

From Proposition 3.1 it follows that for any $\mathbf{x}' \in B_\delta(\mathbf{x}_{20})$, and for any $t \in [0,T]$, $|\xi_{\mathbf{x}}(t) - \xi'_{\mathbf{x}'}(t)| \leq c/L_g$. Let us fix a $\mathbf{x}' \in B_\delta(\mathbf{x}_{20})$. Then, the distance between the corresponding traces is:

$$\begin{aligned} d(\nu_{\mathbf{x}}, \nu'_{\mathbf{x}'}) &= \sup_{t \in [0,T]} |\nu_{\mathbf{x}}(t) - \nu'_{\mathbf{x}'}(t)| \\ &= \sup_{t \in [0,T]} |g(\xi_{\mathbf{x}}(t)) - g(\xi'_{\mathbf{x}'}(t))| \\ &\leq L_g \sup_{t \in [0,T]} |\xi_{\mathbf{x}}(t) - \xi'_{\mathbf{x}'}(t)| \\ &\leq L_g \frac{c}{L_g} = c. \end{aligned}$$

Here we have used the assumption that the two systems have the same output mapping g and recall that L_g is the Lipschitz constant of this mapping.

For the second part, we assume that the algorithm returns *COUNTEREX* and show that there exists a trace $\nu_{\mathbf{x}_{10}}$ of A_1 which is at least c distance away from all traces of A_2. Examining the algorithm, output of *COUNTEREX* occurs if there exists a constant $\delta > 0$ and $\mathbf{x}_{10} \in \Theta_1$, such that for any initial state $\mathbf{x}_{20}$ of A_2,

$$d_H(pipe[\mathbf{x}_{10}], pipe[\mathbf{x}_{20}]) \geq \frac{c}{S_g},$$

where S_g is the sensitivity of the output mapping. From Proposition 2.3 it follows that for any $\mathbf{x}' \in \Theta_2$, for all $t \in [0,T]$, $\xi_{\mathbf{x}_{10}}(t) - \xi'_{\mathbf{x}'}(t) \geq \frac{c}{S_g}$. Now, consider the distance between the corresponding traces:

$$\begin{aligned} d(\nu_{\mathbf{x}_{10}}, \nu'_{\mathbf{x}'}) &= \sup_{t \in [0,T]} |g(\xi_{\mathbf{x}_{10}}(t)) - g(\xi'_{\mathbf{x}'}(t))| \\ &\geq \sup_{t \in [0,T]} S_g |\xi_{\mathbf{x}_0}(t) - \xi'_{\mathbf{x}'}(t)| \\ &\geq S_g \frac{c}{S_g} = c. \end{aligned}$$

Thus, the traces are at least c apart, and A_2 is not a c-abstraction of A_1. Here again we have used the assumption that the two systems have the same output mapping g S_g is the Lipschitz constant of this mapping.

□

Next, we prove that the algorithm terminates provided either (a) $A_1 \preceq_{\frac{c}{2},T} A_2$ or (b) $d_H(\mathsf{Traces}_{A_1}, \mathsf{Traces}_{A_2}) > 2c$.

THEOREM 3.4. *The c-abstraction algorithm terminates either if A_2 is a c_1-abstraction of A_1 for any $c_1 < \frac{c}{2}$ or if there exists a trace of A_1 which is c_2 distance away from all traces of A_2, for some $c_2 > 2c$.*

PROOF. Assume without loss of generality that $L_g < 2S_g$. For the first part, assume that the A_2 is a $\frac{c}{2}$-abstraction of A_1. First, we will show that Line 15 returning a counter-example is never executed. For the sake of contradiction, let us assume that this line is executed. Then $dia(pipe[\mathbf{x}]) \leq \frac{c}{2S_g}$. Also, for any execution $\xi_{\mathbf{x}}$ of A_1 there exists an execution $\xi'_{\mathbf{x}'}$ of A_2 such that for any $t \in [0,T]$, $|\xi_{\mathbf{x}}(t) - \xi'_{\mathbf{x}'}(t)| < \frac{c}{2S_g}$ (this is because $A_1 \preceq_{\frac{c}{2},T} A_2$). Thus, from any pipe containing $\xi_{\mathbf{x}}$, the distance to any pipe containing any $\xi'_{\mathbf{x}'}$ is less than $\frac{c}{S_g}$. This violates the precondition for returning a counter-example.

It suffices to show that every initial state $\mathbf{x} \in \Theta_1$ is eventually removed from *Init* in Line 13. Let us fix an execution $\xi'_{\mathbf{x}'}$ of A_2 such that for any $t \in [0,T]$, $|\xi_{\mathbf{x}}(t) - \xi'_{\mathbf{x}'}(t)| < \frac{c}{2S_g}$. Under the above conditions, in each iteration of the while loop, δ, τ, and ϵ are halved in Line 18. From the Definition 4 the diameter $dia(sim[\mathbf{x}_{10}]) \leq \epsilon$ for $\mathbf{x}_{10}$ with $|\mathbf{x}_{10} - \mathbf{x}| \leq \delta$. Similarly, the diameter $dia(sim[\mathbf{x}_{20}]) \leq \epsilon$ for $\mathbf{x}_{20}$ with $|\mathbf{x}_{20} - \mathbf{x}'| \leq \delta$. Notice that as these parameters

decrease, from the definition of the discrepancy function, $\beta_i(\mathbf{x}, \mathbf{x}', .) \to 0$. Thus, the distance between the bloated pipes containing $\xi_{\mathbf{x}}$ and $\xi'_{\mathbf{x}'}$ also converge to $\frac{c}{2S_g}$. With the assumption that $L_g < 2S_g$, it follows that eventually two points $\mathbf{x}_{10} \in X_1$ and $\mathbf{x}_{20} \in X_2$ will satisfy the condition $(d_H(pipe[\mathbf{x}_{10}], pipe[\mathbf{x}_{20}]) \leq \frac{c}{L_g})$ in Line 13 with $\mathbf{x}' \in B_\delta(\mathbf{x}_{20})$, and more importantly, $\mathbf{x} \in B_\delta(\mathbf{x}_{10})$.

For the second part, suppose there exists a trace $\nu_{\mathbf{x}}$ of A_1 such that for any trace $\nu'_{\mathbf{x}'}$ of A_2, $d(\nu_{\mathbf{x}}, \nu_{\mathbf{x}}) > 2c$. Then we know that for each $t \in [0, T]$, $|\xi_{\mathbf{x}}(t) - \xi'_{\mathbf{x}'}(t)| > 2c/L_g$. For the sake of contradiction, let us assume that $\mathbf{x}$ is eliminated from *Init* in Line 13. Then there must exist $pipe[\mathbf{x}_{10}]$ containing $\xi_{\mathbf{x}}$ $pipe[\mathbf{x}_{20}]$ containing $\xi'_{\mathbf{x}'}$ with $dia(pipe[\mathbf{x}_{10}]) \leq c/2L_g$ and $dia(pipe[\mathbf{x}_{20}]) \leq c/2L_g$. Then, $d_H(pipe[\mathbf{x}_{10}], pipe[\mathbf{x}_{10}]) > 2c/L_g - c/2L_g - c/2L_g = c/L_g$ which contradicts the first condition in Line 13.

So, $\mathbf{x}$ is never eliminated from *Init*. Analogous to the argument presented for the first part, the pipe computed containing $\xi_{\mathbf{x}}$ become smaller and smaller in diameter as τ, ϵ and δ are reduced and the pipes computed containing all the executions of A_2 starting from Θ_2, including $\xi'_{\mathbf{x}'}$ also become smaller. Eventually, $d_H(pipe[\mathbf{x}_{10}], pipe[\mathbf{x}_{20}]) > 2c/L_g$ as for each $d(\xi_{\mathbf{x}}, \xi'_{\mathbf{x}'}| > 2c/L_g$. At this point the condition in Line 15 becomes true and $\mathbf{x}$ is produced with the *COUNTEEX* output. □

Thus, there is a range of values of the distance between the sets of traces $d_H(\mathsf{Traces}_{A_1}, \mathsf{Traces}_{A_2})$ in $[\frac{c}{2}, 2c]$ where the algorithm is not guaranteed to terminate. This range can be made arbitrarily small by choosing small values of c.

4. UNBOUNDED TIME EXTENSION

In the previous section, we presented a semi-decision procedure for reasoning about bounded-time abstraction relations between models of cyberphysical systems. Since cyberphysical systems typically run for long time horizons, ideally we would like to perform unbounded time analysis. The following procedure uses bound-time simulations and attempts to compute an over-approximation of the unbounded-time reach set of a dynamical system.

The algorithm adapts a standard fixpoint procedure to now work with our simulation-based technique for computing reach set approximations. The set *newreach* stores the newly discovered reachable states and *reach* accumulates all the reachable states. Both are initialized to Θ. The while loop iterates until no newly reachable states are discovered; at that point *reach* is produced as the output. Inside the while loop, *newreach* is δ-partitioned. Then, as in Algorithm 1, an array of $(\mathbf{x}_1, k\tau, \epsilon, \tau)$-simulations $sim[\mathbf{x}_1]$ are computed for each $\mathbf{x}_1 \in X_1$ and they are bloated to compute the array of pipes $pipe[\mathbf{x}_1]$. The union of the segments in all these pipes give the set *post* and the *newreach* set is obtained by subtracting *reach* from *post*.

The following theorem states that if the above algorithm returns a set of states R then this set is an over-approximation of the *unbounded* reach set of the dynamical system A.

Algorithm 2: Unbounded time reachability.

input: $\mathcal{A}, V, k, \tau, \delta, \epsilon$
1 $newreach \leftarrow \Theta$;
2 $reach \leftarrow \Theta$;
3 **while** $newreach \neq \varnothing$ **do**
4 $X_1 \leftarrow Partition(newreach, \delta)$;
5 **for** $\mathbf{x}_1 \in X_1$ **do**
6 $sim[\mathbf{x}_1] \leftarrow Simulate(A, \mathbf{x}_1, \epsilon, k\tau, \tau)$;
7 $pipe[\mathbf{x}_1] \leftarrow Bloat(sim[\mathbf{x}_1], \delta, V)$;
8 **end**
9 $post \leftarrow \cup_{i \in [k]} \cup_{\mathbf{x}_1 \in X_1} pipe[\mathbf{x}_1].P_i$;
10 $newreach \leftarrow post \setminus reach$;
11 $reach \leftarrow reach \cup newreach$;
12 **end**
13 **return** *reach*

THEOREM 4.1. *If Algorithm 2 returns a set of states R then* $\mathsf{Reach}_A(\Theta) \subseteq R$.

Proof sketch. From Corollary 3.2 it follows that in each iteration of the while loop $\mathsf{Reach}_A(newreach, k\tau) \subseteq post$, that is the set computed using simulations and bloating in Line 9. The set *newreach* is updated to be an over-approximation of the states that are reached for the first time in the current iteration. A simple induction on the number of iterations show that at the i^{th} iteration, *reach* contains all states that are reachable from Θ in $i(k\tau)$ time. The computation halts in an iteration when no new reachable states are discovered and the corresponding output *reach* is the least fixpoint of the algorithm containing Θ and therefore it over-approximates the unbounded-time reach set from Θ.

5. DISCUSSIONS

The simulation-based reachability algorithms [5,10,13,17,18] provide a general and scalable building-block for analysis of nonlinear, switched, and hybrid models. Since simulation-based analysis can be made embarrassingly parallel, these approaches can scale to real-world models with dozens and possibly hundreds of continuous dimensions. This paper takes this static-dynamic analysis approach to checking abstraction relations. As we discussed in the introduction, computing reach set over-approximations are not sufficient for reasoning about abstraction relations. Our procedure takes into account the precision of the over-approximations in deciding that each behavior of A is indeed close to some behavior of B or that there exits a behavior of A that is far from all behaviors of B. For the sake of simplifying presentations, in this paper we presented the results for models of nonlinear dynamical systems, but these results can be extended to switched systems [21] in a more or less straightforward fashion (see [13] for analogous extensions for reachability algorithms). This work suggests several directions for future research in developing new notions of abstraction, corresponding decision procedures, and in extending them to be applicable to broader classes of models that arise in analysis of cyberphysical systems under attacks.

5.1 Future Research Directions

Switched system models and models with inputs. The switched system [21] formalism is useful where it suffices

to view the software or the adversary as something that only changes the continuous dynamics. They are useful for modeling time-triggered control systems and timing-based attacks. A switched system is described by a collection of dynamical systems (Definition 1) and a piece-wise constant switching signal that determines which particular ODE from the collection that governs the evolution of the system at a given time. A timing attack can be modeled as altered switching signal (as well as the changed dynamics). One nice property of switched system models is that the executions ξ are continuous functions of time just like ODEs. If all the ODEs are equipped with discrepancy functions, then we show in [13] that it is possible to compute reach set over-approximations for a set of switching signals by partitioning both the initial set and the set of switching signals. This technique essentially works also for analyzing abstraction relation between switched models.

Switched and ODE models with inputs will enable us to model open cyberphysical systems and adversaries that feed bad inputs to such systems. The main challenge here is reasoning about the distance between trajectories that start from different initial states, as well as, experience different input signals. In our recent paper [17] we have used an input-to-state discrepancy function to reason about reachability of such models and a similar approach can work for abstractions.

Nondeterministic models and games. In all of the above models, uncertainty is encoded in the choice of the initial state and in the choice of the switching signal. We plan on addressing this limitation of the current approach in our future work. In order to apply out analytical framework to a broader class of system models and attacks, we have to develop decision procedures for hybrid model with nondeterministic transitions as well as nondeterministic dynamics. A general framework will enable us to model systems where the controller and the adversary take turns in controlling the plant. The algorithms for analyzing such systems will essentially have to solve satisfiability of quantified nonlinear formulas with single quantifier alternation.

6. REFERENCES

[1] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, P.-H. Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[2] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[3] D. Angeli. Further results on incremental input-to-state stability. *Automatic Control, IEEE Transactions on*, 54(6):1386–1391, 2009.

[4] David Angeli. A lyapunov approach to incremental stability properties. *Automatic Control, IEEE Transactions on*, 47(3):410–421, 2002.

[5] Yashwant Annapureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *TACAS*, 2011.

[6] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. *S-taliro: A tool for temporal logic falsification for hybrid systems.* Springer, 2011.

[7] Olivier Bouissou and Matthieu Martel. Grklib: a guaranteed runge kutta library. In *Scientific Computing, Computer Arithmetic and Validated Numerics, 2006. SCAN 2006. 12th GAMM-IMACS International Symposium on*, pages 8–8. IEEE, 2006.

[8] CAPD. Computer assisted proofs in dynamics, 2002.

[9] Edmund M Clarke, Orna Grumberg, and David E Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(5):1512–1542, 1994.

[10] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification*, pages 167–170. Springer, 2010.

[11] Parasara Sridhar Duggirala, Taylor T. Johnson, Adam Zimmerman, and Sayan Mitra. Static and dynamic analysis of timed distributed traces. In *RTSS*, pages 173–182, 2012.

[12] Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. The compute execute check engine (c2e2), 2013.

[13] Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Verificationcation of annotated models from executions. In *Proceedings of International Conference on Embedded Software (EMSOFT 2013)*, pages 1–10, Montreal, QC, Canada, September 2013. ACM SIGBED, IEEE.

[14] Antoine Girard, A. Agung Julius, and George J. Pappas. Approximate simulation relations for hybrid systems. In *IFAC Analysis and Design of Hybrid Systems*, Alghero, Italy, June 2006.

[15] Antoine Girard, Giordano Pola, and Paulo Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. In Magnus Egerstedt and Bud Mishra, editors, *HSCC*, volume 4981 of *Lecture Notes in Computer Science*, pages 201–214. Springer, 2008.

[16] Timothy V Griffiths. The unsolvability of the equivalence problem for λ-free nondeterministic generalized machines. *Journal of the ACM (JACM)*, 15(3):409–413, 1968.

[17] Zhenqi Huang and Sayan Mitra. Proofs from simulations and modular annotations. In *In 17th International Conference on Hybrid Systems: Computation and Control*, Berlin, Germany. ACM press.

[18] Zhenqi Huang and Sayan Mitra. Computing bounded reach sets from sampled simulation traces. In *In The 15th International Conference on Hybrid Systems: Computation and Control (HSCC 2012), Beijing, China.*, 2012.

[19] Dilsun K. Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager. *The Theory of Timed I/O Automata.* Synthesis Lectures on Computer Science. Morgan Claypool, November 2005. Also available as Technical Report MIT-LCS-TR-917.

[20] H. K. Khalil. *Nonlinear Systems.* Prentice Hall, New Jersey, 3rd edition, 2002.

[21] Daniel Liberzon. *Switching in Systems and Control.*

Systems and Control: Foundations and Applications. Birkhauser, Boston, June 2003.

[22] W. Lohmiller and J. J. E. Slotine. On contraction analysis for non-linear systems. *Automatica*, 1998.

[23] David G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications*. John Wiley and Sons, Inc., New York, 1979.

[24] Nedialko S Nedialkov, Kenneth R Jackson, and George F Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.

[25] Giordano Pola, Antoine Girard, and Paulo Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[26] Pavithra Prabhakar, Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Hybrid automata-based cegar for rectangular hybrid systems. In *VMCAI*, pages 48–67, 2013.

[27] Géraud Sénizergues. The equivalence problem for deterministic pushdown automata is decidable. In *Automata, languages and programming*, pages 671–681. Springer, 1997.

[28] T. A. Henzinger and P. -H. Ho. Algorithmic analysis of nonlinear hybrid systems. In P. Wolper, editor, *Proceedings of the 7th International Conference On Computer Aided Verification*, volume 939, pages 225–238, Liege, Belgium, 1995. Springer Verlag.

APPENDIX

A. BASIC DEFINITIONS AND NOTATIONS

For a natural number $n \in \mathbb{N}$, $[n]$ is the set $\{1, 2, \ldots, n\}$. For a sequence A of objects of any type with n elements, we refer to the i^{th} element, $i \leq n$ by A_i. For a real-valued vector $\mathbf{x}$, $|\mathbf{x}|$ denotes the ℓ^2-norm unless otherwise specified. The diameter of a compact set $R \subseteq \mathbb{R}^n$, $dia(R)$ is defined as the maximum distance between any two points in it: $dia(R) = sup_{\mathbf{x},\mathbf{x}' \in R}|\mathbf{x} - \mathbf{x}'|$.

Variable valuations. Let V be a finite set of real-valued variables. Variables are names for state and input components. A *valuation* $\mathbf{v}$ *for* V is a function mapping each variable name to its value in $\mathbb{R}$. The set of valuations for V is denoted by $\mathit{Val}(V)$. Valuations can be viewed as vectors in $\mathbb{R}^{|V|}$ dimensional space with by fixing some arbitrary ordering on variables. $B_\delta(\mathbf{v}) \subseteq \mathit{Val}(V)$ is the closed ball of valuations with radius δ centered at $\mathbf{v}$. The notions of continuity, differentiability, and integration are lifted to functions defined over sets of valuations in the usual way.

For any function $f : A \to B$ and a set $S \subseteq A$, $f \lceil S$ is the restriction of f to S. That is, $(f \lceil S)(s = f(s)$ for each $s \in S$. So, for a variable $v \in V$ and a valuation $\mathbf{v} \in \mathit{Val}(V)$, $\mathbf{v} \lceil v$ is the function mapping $\{v\}$ to the value $\mathbf{v}(v)$. A function $f : A \to \mathbb{R}$ is *Lipschitz* if there exists a constant $L \geq 0$—called the *Lipschitz constant*—such that for all $a_1, a_2 \in A$ $|f(a_1) - f(a_2)| \leq L|a_1 - a_2|$. We define a function f to have *sensitivity* of S_f if for all $a_1, a_2 \in A$ $|f(a_1) - f(a_2)| \geq S_f|a_1 - a_2|$. A continuous function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is in the *class of* $\mathcal{K}$ *functions* if $\alpha(0) = 0$ and it is strictly increasing. Class $\mathcal{K}$ functions are closed under composition and inversion. A class $\mathcal{K}$ function α is a *class* $\mathcal{K}_\infty$ function if $\alpha(x) \to \infty$ as $x \to \infty$. A continuous function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is called a *class* $\mathcal{KL}$ *function* if for any t, $\beta(x, t)$ is a class $\mathcal{K}$ function in x and for any x, $\beta(x, t) \to 0$ as $t \to \infty$.

Trajectories. Trajectories model the continuous evolution of variable valuations over time. A *trajectory* for V is a differentiable function $\tau : \mathbb{R}_{\geq 0} \to \mathit{Val}(V)$. The set of all possible trajectories for V is denoted by $Traj(V)$. For any function $f : C \to [A \to B]$ and a set $S \subseteq A$, $f \downarrow S$ is the restriction of $f(c)$ to S. That is, $(f \downarrow S)(c) = f(c) \lceil S$ for each $c \in C$. In particular, for a variable $v \in V$ and a trajectory $\tau \in Traj(V)$, $\tau \downarrow v$ is the trajectory of v defined by τ.

Dynamical systems. The set of all trajectories of A with respect to a set of initial states $\Theta' \subseteq \mathit{Val}(X)$ and a set of is denoted by $Traj(A, \Theta')$. The components of dynamical system A and A_i are denoted by $X_A, \Theta_A, f_\mathcal{A}$ and X_i, Θ_i, f_i, respectively. We will drop the subscripts when they are clear from context. The set of all possible state trajectories and output trajectories of A (from different initial states in Θ) are denoted by Execs_A and Traces_A, respectively. The set of executions (and traces) from the set of initial states Θ and upto time bound T is denoted by $\mathsf{Execs}_A(\Theta, T)$ (and $\mathsf{Traces}_A(\Theta, T)$, repectively). A state $\mathbf{x} \in \mathbb{R}^n$ is *reachable* if there exists and execution ξ and a time t such that $\xi(t) = \mathbf{x}$. The set of reachable states from initial set Θ within time T is denoted by $\mathsf{Reach}_A(\Theta, T)$.