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 cyberphysical 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
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-bound 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 $A_2$ then it implies that each behavior 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 $\frac{1}{2}$-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 $\nu$ of $A$ is close to some over-approximation of $B$. But, this does not imply that $\nu$ 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 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 \((\Theta, f, g)\) where

(i) \(\Theta \subseteq \mathbb{R}^n\) is a compact set of initial states; \(\mathbb{R}^n\) is the state space and its 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 \(m\).

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

\[\dot{\xi}_x(t) = f(\xi_x(t)),\]

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

\[\nu_x(t) = g(\xi_x(t)).\]

Under the Lipschitz assumption (iii), the differential equation (1) admits a unique state trajectory defined by the initial state \(x\) 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 \(\xi\) and \(\nu\). Given a state trajectory \(\xi\) 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 \(\nu\), 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 \(\Theta\)) are denoted by \(\text{Execs}_A\) and \(\text{Traces}_A\), respectively. A state \(x \in \mathbb{R}^n\) is said to be reachable if there exists \(x' \in \Theta\) and \(t \in \mathbb{R}_{>0}\) such that \(\xi_x(t) = x\). The set of all reachable states of \(A\) is denoted by \(\text{Reach}_A\). 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:

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

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 up to 10 time units (computed using the algorithm described 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 \(\nu_1, \nu_2\)

![Figure 1: Reachable states and traces of the dynamical system in Example 1.](image-url)
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 \textit{c-abstraction} of $A_1$ up to time $T$, if
\[
d_H(\text{Traces}_{A_1}(T), \text{Traces}_{A_2}(T)) \leq c.
\]
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 $\nu_1$ of $A_1$, there exists another output trace $\nu_2$ of $A_2$ which differs from $\nu_1$ 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, $\text{Traces}_{A_1} \subseteq \text{Traces}_{A_2}$. 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 \textit{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 \textit{segment} is a pair $(P, t)$ where $P$ is a subset of $\mathbb{R}^n$ and $t$ is a nonnegative real number. An $n$-dimensional pipe $\Pi$ is a sequence of segments
\[
\Pi = (P_0, t_0), (P_1, t_1), \ldots, (P_k, t_k),
\]
where for each $i$ in the sequence, $t_i > t_{i-1}$. The duration of the pipe is $\Pi.\text{dur} = t_k$ and its length is the number of segments $\Pi.\text{len} = k + 1$.

The semantics of a pipe $\Pi$ 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:

\[
[\Pi]_A = \{ \xi \in \text{Execs}_A \mid \forall t \in [0, T], \xi(t) \in P_0, \xi \in \Pi.\text{len}, t \in [t_{i-1}, t_i], \xi(t) \in P_i \}.
\]

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

We define $\text{dia}(\Pi) = \max_{\xi \in [\Pi.\text{len}]} \text{dia}(P_i)$ as the maximum diameter of any of the segments. We say that two pipes $\Pi$ and $\Pi'$ are \textit{comparable} if they have the same duration, length, and dimension and furthermore, for each $i \in [\Pi.\text{len}]$, $t_i = t_i'$. For two comparable pipes $\Pi, \Pi'$, we say that $\Pi$ is contained in $\Pi'$, denoted by $\Pi \subseteq \Pi'$, iff for each $i \in [\Pi.\text{len}]$, $P_i \subseteq P_i'$. The distance from $\Pi$ to $\Pi'$ is defined in the natural way by taking the maximum distance from the corresponding segments of $\Pi$ to those of $\Pi'$.

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

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

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

**Proposition 2.2.** Consider two comparable pipes $\Pi_1, \Pi_2$. 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 $\Pi_1, \Pi_2$. 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 \textit{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 $V : \mathbb{R}^n \to \mathbb{R}^2_0$ is called a \textit{discrepancy function} for an $(n, m)$-dimensional dynamical system if and only if there are functions $\alpha_1, \alpha_2 \in C_\infty$ and a uniformly continuous function $\beta : \mathbb{R}^m \times \mathbb{R}^2_0$ with $\beta(x_1, x_2, t) \to 0$ as $|x_1 - x_2| \to 0$ such that for any pair of states $x_1, x_2 \in \mathbb{R}^2$:

\[
x_1 \neq x_2 \iff V(x_1, x_2) > 0,
\]

\[
\alpha_1(|x_1 - x_2|) \leq V(x_1, x_2) \leq \alpha_2(|x_1 - x_2|) \text{ and } V(t_i(t), \xi_2(t)) \leq \beta(x_1, x_2, t),
\]

A tuple $(\alpha_1, \alpha_2, \beta)$ satisfying the above conditions is called a \textit{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 $V(x_1, x_2)$ vanishes to zero if and only if the first two arguments are identical. The second condition states that the value of $V(x_1, x_2)$ can be upper and lower-bounded by functions of the $L^2$ distance between $x_1$ and $x_2$. 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 $x_1$ converges to $x_2$.

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, and in [13] several strategies for proposed for nonlinear systems. Our semi-decision procedure (Algorithm 1) for $\phi = (R_0, t_0), (R_1, t_1), \ldots, (R_k, t_k)$ where

**Definition 4.** Given a dynamical system $A$, an initial state $x$, a time bound $T$, an error bound $\epsilon > 0$, and time step $\tau > 0$, a $(x, T, \epsilon, \tau)$-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 $\forall j \in [k], t_{j+1} - t_j \leq \tau$,

(ii) $\forall j \in [k]$ and $\forall t \in [t_j, t_{j+1}], \xi(t) \in R_{j+1}$, and

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

Algorithm 1 makes subroutine calls to a $\text{Simulate}$ 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 $\text{sim} = (R_0, t_0), (R_1, t_1), \ldots, (R_k, t_k)$ be a $(x, T, \epsilon, \tau)$-simulation pipe for a dynamical system $A$. Suppose $V$ be a discrepancy function of $A$ with witness $(\alpha_1, \alpha_2, \beta)$. Then, for $\delta > 0$, $\text{Bloat} (\text{sim}, \delta, V)$ is defined as the pipe $(P_0, t_0), \ldots, (P_k, t_k)$ such that for each $j \in [k]$,

$$P_j = \{ x_1 | \exists x_2 \in R_j \land V(x_1, x_2) \leq \epsilon_j \},$$

where

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

In other words, $\epsilon_j$ is an upper-bound on the value of $V$ for two executions $\xi$ and $\xi'$ starting from within $B_\delta(x)$ over the time interval $[t_{j-1}, t_j]$. And $P_j$ bloats $R_j$ to include all states $x_1$ for which there exists a state $x_2$ in $R_j$ with the discrepancy function bounded by $\epsilon_j$. Our algorithm makes subroutine calls to a $\text{Bloat}$ function which takes a simulation pipe, the function $\beta$ and the constant $\delta$ and returns the pipe $(P_0, t_0), \ldots, (P_k, t_k)$ defined above.

**Algorithm 1:** Deciding $c$-abstractions.

**Input:** $A_1, V_1, A_2, V_2, T, c$

1. $\text{Init} \leftarrow \Theta_1$
2. $\delta \leftarrow \delta_0; \tau \leftarrow \tau_0; \epsilon \leftarrow \epsilon_0$
3. while $\text{Init} \neq \emptyset$
   4. $X_1 \leftarrow \text{Partition} (\text{Init}, \delta)$
   5. $X_2 \leftarrow \text{Partition} (\Theta_2, \delta)$
   6. $\text{foreach } x_{10} \in X_1, x_{20} \in X_2$
      7. $\text{sim}(x_{10}) \leftarrow \text{Simulate}(x_{10}, x_{1}, \epsilon, T, \tau)$
      8. $\text{pipe}(x_{10}) \leftarrow \text{Bloat}(\text{sim}(x_{10}), \delta, V_1)$
      9. $\text{sim}(x_{20}) \leftarrow \text{Simulate}(x_{20}, x_{2}, \epsilon, T, \tau)$
      10. $\text{pipe}(x_{20}) \leftarrow \text{Bloat}(\text{sim}(x_{20}), \delta, V_2)$
   end
   11. $\text{end for each}$

3.1 Description of the Algorithm

Inside the while loop of Algorithm 1, first, two $\delta$-covers are computed for $\text{Init}$—a subset of the initial states $\Theta_1$ of $A_1$, and the set of initial states $\Theta_2$ of $A_2$. Next, in the first for loop, for each of the states $x_{10} \in X_1$ and $x_{20} \in X_2$ in the respective covers, a $(x_{10}, T, \epsilon, \tau)$-simulation pipe $\text{sim}(x_{10})$ is computed. Then this pipe is bloated with the parameter $\delta$ 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 $A_{i}, i \in \{1, 2\}$ and constants $\delta, \epsilon, \tau$ and $T$, $\text{Execs}_{A_i}(B_\delta(x_{10}), T) \subseteq \{\text{pipe}(x_{10})\}$.

**Proof.** Let $\text{sim}(x_{10}) = (R_0, t_0), \ldots, (R_k, t_k)$ and $\text{pipe}(x_{10}) = (P_0, t_0), \ldots, (P_k, t_k)$. We fix an initial state $x' \in B_\delta(x_{10})$, and show that for any $t \leq t_k$, the state $\xi(t)$ 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 $\xi(t) \subseteq R_j$. And from Definition 5, we know that since $x' \in B_\delta(x_{10})$, the $V(\xi(t), \xi'(t)) \leq \beta(x', x_{10}, t)$ and therefore $\xi'(t) \in P_j$. 

**End of proof.**
COROLLARY 3.2. For the dynamical system $A_i, i \in \{1, 2\}$ and constants $\delta, \epsilon, \tau$ and $T$,

$\text{Reach}_{A_i}(\Theta_i, T) \subseteq \bigcup_{x_0 \in \Theta_i} \bigcup_{j \in [T/\tau]} \text{pipe}[x_0], P_j$.

Here we use $\text{pipe}[x_0], P_j$ to denote the subset of $\mathbb{R}^n$ in the $j$th segment of the pipe $\text{pipe}[x_0]$.

Every time a new set of bloated simulation pipes are computed, $\text{pipe}[x_10]$ for each $x_10 \in \text{Init}$ and $\text{pipe}[x_20]$ for each $x_20 \in \Theta_1$, the algorithm performs the following checks. If there exists a $\text{pipe}[x_10]$ and a $\text{pipe}[x_20]$, both less than $c/2Lg$ in diameter and within $c/Lg$ distance then $B_3(x_{10})$ is eliminated from init. Here $Lg$ is the Lipschitz constant and $Sg$ is the sensitivity constant of the common output function $g$. If there exists a $\text{pipe}[x_10]$ such that for all the $\text{pipe}[x_{20}]$‘s, $x_{20} \in \Theta_2$, the diameter of the first is less than $c/2Sg$ and they are at least $c/Sg$ distance away from each other, then $x_{10}$ and $\delta$ is produced as a counter-example to the c-abstraction. The while loop ends when $\text{Init}$ 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 $A_1$ upon time $T$, and if the output is (COUNTEREX, $x_{10}, \delta$) 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 $B_3(x_{10})$ 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 $x \in \Theta_1$, there exists an initial state $x' \in \Theta_2$ such that $d(\nu_x, \nu'_x) \leq c$. Here $\nu_x$ is the output trace of $A_1$ from $x$ and $\nu'_x$ is the output trace of $A_2$ from $x'$.

The algorithm returns c-ABSTRACTION only when $\text{Init}$ becomes empty. This occurs when each initial state $x \in \Theta_1$ of $A_1$ is in the $\delta$-ball of some state $x_{10} \in \Theta_1$ such that $x_{10}$ is in a cover $\mathcal{X}$ and recall that the condition in Line 13. It suffices to show that this condition $d_H(\text{pipe}[x_{10}], \text{pipe}[x_{20}]) \leq c/Lg$ implies that there exists $x' \in \Theta_2$, $d(\nu_x, \nu'_x) \leq c$.

From Proposition 3.1 it follows that for any $x' \in B_3(x_{20})$, and for any $t \in [0, T]$, $|\xi_x(t) - \xi'_{x'}(t)| \leq c/Lg$. Let us fix a $x' \in B_3(x_{20})$. Then, the distance between the corresponding traces is:

$$d(\nu_x, \nu'_x) = \sup_{t \in [0, T]} |\nu_x(t) - \nu'_x(t)|$$

$$= \sup_{t \in [0, T]} |g(\xi_x(t)) - g(\xi'_{x'}(t))|$$

$$\leq Lg \sup_{t \in [0, T]} |\xi_x(t) - \xi'_{x'}(t)|$$

$$\leq Lg \frac{\delta}{Lg} = c.$$
Theorem 4.1. If Algorithm 2 returns a set of states $R$ then $\text{Reach}_A(\Theta) \subseteq R$.

Proof sketch. From Corollary 3.2 it follows that in each iteration of the while loop $\text{Reach}_A(\text{newreach}, k) \subseteq \text{post}$, that is the set computed using simulations and bloating in Line 9. The set $\text{neureach}$ 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, $\text{reach}$ contains all states that are reachable from $\Theta$ in $(i\cdot k)$ time. The computation halts in an iteration when no new reachable states are discovered and the corresponding output $\text{reach}$ is the least fixpoint of the algorithm containing $\Theta$ and therefore it over-approximates the unbounded-time reach set from $\Theta$.

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 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

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 $x$, $|x|$ denotes the $L^2$-norm unless otherwise specified. The diameter of a compact set $R \subseteq \mathbb{R}^n$, $\text{dia}(R)$ is defined as the maximum distance between any two points in it: $\text{dia}(R) = \sup_{x,x' \in R} |x - x'|$.

Variable valuations. Let $V$ be a finite set of real-valued variables. Variables are names for state and input components. A valuation $v$ for $V$ is a function mapping each variable name to its value in $\mathbb{R}$. The set of valuations for $V$ is denoted by $\text{Val}(V)$. Valuations can be viewed as vectors in $\mathbb{R}^{|V|}$ dimensional space with with fixing some arbitrary ordering on variables. $B_2(v) \subseteq \text{Val}(V)$ is the closed ball of valuations with radius $\delta$ centered at $v$. The notions of continuity, differentiability, and integration are lifted to functions defined over sets of valuations in the usual way.

For any function $f : A \rightarrow B$ and a set $S \subseteq A$, $f \upharpoonright S$ is the restriction of $f$ to $S$. That is, $(f \upharpoonright S)(s) = f(s)$ for each $s \in S$. So, for a variable $v \in V$ and a valuation $v \in \text{Val}(V)$, $v \upharpoonright v$ is the function mapping $\{v\}$ to the value $v(v)$. A function $f : A \rightarrow \mathbb{R}$ 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 $\alpha : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is in the class of $K$ functions if $\alpha(0) = 0$ and it is strictly increasing. Class $K$ functions are closed under composition and inversion. A class $K$ function $\alpha$ is a class $K_{\infty}$ function if $\alpha(x) \rightarrow \infty$ as $x \rightarrow \infty$. A continuous function $\beta : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is called a class $K_{\infty}$ function if for any $t$, $\beta(x,t)$ is a class $K$ function in $x$ and for any $x$, $\beta(x,t) \rightarrow 0$ as $t \rightarrow \infty$.

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

Dynamical systems. The set of all trajectories of $A$ with respect to a set of initial states $\Theta \subseteq \text{Val}(X)$ and a set of is denoted by $\text{Traj}(A, \Theta')$. The components of dynamical system $A$ and $A_i$ are denoted by $X_A, \Theta_A, f_A$ and $X_i, \Theta_i, f_i$, 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 $\Theta$) are denoted by $\text{Execs}_A$ and $\text{Traces}_A$, respectively. The set of executions (and traces) from the set of initial states $\Theta$ and upto time bound $T$ is denoted by $\text{Execs}_A(\Theta, T)$ (and $\text{Traces}_A(\Theta, T)$, respectively). A state $x \in \mathbb{R}^n$ is reachable if there exists an execution $\xi$ and a time $t$ such that $\xi(t) = x$. The set of reachable states from initial set $\Theta$ within time $T$ is denoted by $\text{Reach}_A(\Theta, T)$.