Temporal Logics for Learning and Detection of Anomalous Behavior

Zhaodan Kong, *Member, IEEE*, Austin Jones, *Member, IEEE*, and Calin Belta, *Senior Member, IEEE*

***Abstract*—The increased complexity of modern systems necessitates automated anomaly detection methods to detect possible anomalous behavior determined by malfunctions or external attacks. We present formal methods for inferring (via supervised learning) and detecting (via unsupervised learning) anomalous behavior. Our procedures use data to construct a signal temporal logic (STL) formula that describes normal system behavior. This logic can be used to formulate properties such as "If the train brakes within 500 m of the platform at a speed of 50 km/hr, then it will stop in at least 30 s and at most 50 s." Our procedure infers not only the physical parameters involved in the formula (e.g., 500 m in the example above) but also its logical structure. STL gives a more human-readable representation of behavior than classifiers represented as surfaces in high-dimensional feature spaces. The learned formula enables us to perform early detection by using monitoring techniques and anomaly mitigation by using formal synthesis techniques. We demonstrate the power of our methods with examples of naval surveillance and a train braking system.**

***Index Terms*—Anomaly detection, formal methods, learning, networked systems, signal temporal logic (STL).**

I. Introduction

MODERN systems play increasingly important roles in the operation of critical infrastructure such as transportation networks and power grids. The large scale and highly nonlinear nature of some of these systems, however, renders the use of classical analysis tools difficult. Due to their networked nature, these systems are also vulnerable to external attacks or disruptions. Recent high-profile attacks, e.g., the Maroochy water breach [29] and the control system malware Stuxnet [12], highlight the need to understand system security [23], [27], [30]. In [27], a packet delay attack is considered, in which, by accessing and then delaying the sensor data, the adversary can harm the physical components of the system. In [23] and [30], more general issues such as detectability and identifiability for a wide range of attacks are investigated. In all the cited works, the physical system evolves according to a linear model that is known to the designers. This assumption is not consistent with the growing complexity of modern systems and the involvement of agents, such as humans, whose behaviors are generally quite hard to predict.

In this work, we present model-free algorithms for system security. As data classifiers, we use formulae of signal temporal logic (STL), a specification language used in the field of formal methods to specify behaviors of continuous systems. STL can be used to express system properties that include time bounds and bounds on physical system parameters, such as "If the boat remains in region A while maintaining its speed below 10 kph for 10 min., it is guaranteed to reach the port within 15 min.?" STL formulae resemble natural language, which means they can be useful for human operators. Further, the rigorous mathematical definition of the logic means that they can be used in computational routines to automatically monitor systems for undesired behaviors.

We consider two critical security problems. The first is *anomaly learning* via supervised learning. In this case, given system outputs labeled according to whether a system behaves normally or not, we infer a temporal logic formula that can be used to distinguish between normal system behaviors, e.g., the brakes of a train are engaged if the velocity is beyond a certain threshold, and anomalous (or undesired) behaviors, e.g., the brakes are not properly engaged due to attacks. Preliminary results for this problem appeared in [18]. The second problem is *anomaly detection* via unsupervised learning. In this case, the system outputs are not labeled, i.e., there is no expert-in-the-loop that determines whether a given trace represents normal or attacked operation, and we infer a formula to detect out-of-the-ordinary (anomalous) outputs. Preliminary results for this problem appeared in [16].

We extend our previous work further by considering an on-line anomaly learning algorithm. That is, we present an algorithm to modify the classifying formula as more data is collected over time. This procedure has lower computational costs than the supervised learning procedure developed in [18], making it applicable to high dimensional systems producing large amounts of data. Further, our new procedure requires no *a priori* system data, meaning it is applicable to systems from which no prior outputs are available. We have also streamlined the software implementations from [16], [18] to reduce the computation time. The case study results in this paper were generated with this updated software.

Manuscript received July 10, 2015; revised January 11, 2016 and January 18, 2016; accepted May 24, 2016. Date of publication June 27, 2016; date of current version February 24, 2017. This work was partially supported by the ONR under Grants N00014-14-1-0554 and N00014-10-10952 and by the NSF under Grant NSF CNS-1035588. Some material from this paper was presented at the 17th International Conference on Hybrid Systems: Computation and Control (HSCC 2014) and at the 53rd IEEE Conference on Decision and Control (CDC 2014). Recommended by Associate Editor A. Girard.

Z. Kong is with the Department of Mechanical and Aerospace Engineering, University of California, Davis, CA 95616 USA (e-mail: zdkong@ucdavis.edu).

A. Jones is with Mechanical Engineering and Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: austinjones@gatech.edu).

C. Belta is with the Department of Mechanical Engineering and the Division of System Engineering, Boston University, Boston, MA 02215 USA (e-mail: cbelta@bu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2016.2585083

a) Related work: Most of the recent research on logical inference, the problem of inferring from data a logical expression that describes system properties, has focused on the estimation of parameters associated with a given temporal logic structure [2], [3], [15], [33]. That is, a designer gives a structure such as "The speed settles below v m/s within τ seconds" as an input and the inference procedure finds optimal values for v and τ. The given structure reflects domain knowledge of the designer and properties of interest to be queried. However, the selected formula may not reflect achievable behaviors (over-fitting) or may exclude fundamental behaviors of the system (under-fitting). Furthermore, by giving the formula structure *a priori*, the inference procedure cannot derive new knowledge from the data. Thus, in our procedures, we infer the formula structure (the form of the property that the system demonstrates) along with its optimal parameterization. We guide the search via the robustness degree [9], [11], a signed metric on the signal space which quantifies to what degree a signal satisfies or violates a given formula.

Anomaly detection is the problem of detecting patterns from data that do not conform to expected behavior. It has been used in a wide range of applications, such as intrusion detection for cyber-security, fault detection in safety-critical systems, and video surveillance of illicit activities [5]. Tools from statistics, machine learning, data mining and information theory, such as k-means clustering and k-nearest neighbor (k-NN) graphs, have been adapted to solve the anomaly detection problem [19]. In general, existing techniques for anomaly detection infer a surface embedded in a high-dimensional feature space that separates normal and anomalous data. However, it is hard to interpret the meanings of the surfaces even by domain experts [32]. In contrast, STL formulae have concrete meanings that are easy to be interpreted by humans with basic knowledge of temporal logic (not necessarily domain experts). Further, considering STL formulae as models, it is straightforward to construct monitors for run-time verification for potentially complex systems [8].

b) Contributions: In Section III-A, we define a new logic called *inference parametric signal temporal logic* (iPSTL) [16]. iPSTL is a generalization of reactive Parameteric Signal Temporal Logic (rPSTL), which we defined in [18]. iPSTL is expressive enough to capture properties that are crucial to a wide range of applications, e.g., naval surveillance (Section IV-A).

Second, in Section III-D, we show that we are able to build a directed acyclic graph (DAG) for all iPSTL formulae. The DAG representation encapsulates a partial ordering on the inclusivity of formulae, thus enabling us to formulate both the anomaly learning and the anomaly detection problems as optimization problems whose objective functions involve the robustness degree (Section V-A). We solve the optimization problems by combining discrete search over the DAG and a continuous search over the parameters (Section V). To our knowledge, our body of work represents the first time that inference of a temporal logic structure and parameter estimation have been solved simultaneously in the context of formal methods. This is also one of the first instances.

Third, and finally, in Section VI, we use two case studies, a naval surveillance example (adapted from [19]) and a train network monitoring example (adapted from [28]), to demonstrate our algorithms. Our results point to some very promising future application directions, such as on-line monitoring and automated anomaly mitigation.

II. Preliminaries

A. Signal

Given two sets A and B, $\mathcal{F}(A, B)$ denotes the set of all functions from A to B. Given a time domain $\mathbb{R}^+ := [0, \infty)$ (or a finite prefix of it), a *continuous-time, continuous-valued signal* is a function $s \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$. We use $s(t)$ to denote the value of signal s at time t, and $s[t]$ to denote the suffix of signal s from time t, i.e., $s[t] = \{s(\tau) | \tau \geq t\}$. We use x_s to denote the one-dimensional signal corresponding to the variable x of the signal s. Notations such as y_s, v_s can be defined similarly.

B. Signal Temporal Logic

Signal temporal logic (STL) [2], [21] is a temporal logic defined over signals. STL is a predicate logic with interval-based temporal semantics. The syntax of STL is defined as[1]

$$\phi := g | \neg\phi | \phi_1 \wedge \phi_2 | \phi_1 \vee \phi_2 | F_{[a,b)}\phi | G_{[a,b)}\phi \tag{1}$$

where $g : \mathbb{R}^n \to \{\top, \bot\}$ is a predicate, where $\top, \bot$ are shorthand for true and false, respectively. $\neg$, $\wedge$, and $\vee$ are Boolean negation, conjunction, and disjunction, respectively. $F_{[a,b)}$ and $G_{[a,b)}$ are the temporal "Finally" ("eventually") and "globally" ("always") operators, respectively. (Please refer to [2] and [21] for detailed descriptions of the syntax and semantics of STL.)

We use the notation $s \models \phi$ as a shorthand of $(s, 0) \models \phi$, meaning that a signal s satisfies the STL formula ϕ at time 0. We call the set of all signals s such that $s \models \phi$ the *language* of ϕ, denoted $L(\phi)$. We say that ϕ_1 and ϕ_2 are *semantically equivalent*, denoted $\phi_1 \equiv \phi_2$, if $L(\phi_1) = L(\phi_2)$.

C. System

We define a system as any object $\mathcal{S}$ that produces observable output signals that are related to the evolution of some internal state, e.g., a system of ordinary differential equations or a hybrid automaton. The family of all trajectories that a system $\mathcal{S}$ may produce is called the *language* of $\mathcal{S}$, denoted $L(\mathcal{S})$. A trajectory of $\mathcal{S}$ is a mapping $\mathbf{x} : \mathbb{R}^+ \to X$, where $X \subseteq \mathbb{R}^n$ is the (possibly) high-dimensional physical state space of the system. The operation of the system is observed via an output signal $\mathbf{x}(t)$ which is a possibly non-deterministic function of $\mathbf{x}(t)$ (due to measurement noise for instance).

We are interested in the case in which a system behaves abnormally, e.g., in the case that an adversary can affect the sensors or actuators of the system in order to disrupt its normal operation. Given a system $\mathcal{S}$, we define its *normal behavior* as the set of signals $L_N(\mathcal{S})$ and its *abnormal behavior* as the set of signals $L_A(\mathcal{S})$ such that $L_N(\mathcal{S}) \cap L_A(\mathcal{S}) = \emptyset$ (i.e., anomalous

[1]The full syntax of STL include an interval-bounded "Until" operator. Since this operator can be difficult to interpret and is not used in the fragment iPSTL (defined in Section III-A), it is omitted from the syntax here.

behaviors are qualitatively different from normal behaviors) and $L_N(\mathcal{S}) \cup L_A(\mathcal{S}) = L(\mathcal{S})$.

III. Inference Parametric Signal Temporal Logic

In this section, we first give the syntax and semantics of inference parametric signal temporal logic (iPSTL). We then present some properties of iPSTL that are essential for the design of our learning algorithms presented in Section V.

A. Syntax and Semantics of iPSTL

Parametric STL is an extension of STL in which constants involved in predicates and time intervals are replaced with free parameters. A PSTL formula combined with a valuation, i.e., a mapping from parameters to real values, induces an STL formula. In this work, we focus on a fragment of PSTL that we call *inference PSTL* (iPSTL). The *syntax* of iPSTL is given as

$$\varphi ::= F_{[\tau_1,\tau_2)}\varphi_i \tag{2a}$$

$$\varphi_i ::= F_{[\tau_1,\tau_2)}\ell | G_{[\tau_1,\tau_2)}\ell | \varphi_i \wedge \varphi_i | \varphi_i \vee \varphi_i \tag{2b}$$

where ℓ is a linear predicate of the form $(y_s \sim \pi)$, where y_s is a coordinate of the signal s, τ_1 and τ_2 are time parameters, π is a scale parameter, and $\sim \in \{\leq, >\}$. Connectives $\wedge$ and $\vee$ are Boolean conjunction and disjunction, respectively, and $G_{[\tau_1,\tau_2)}$ and $F_{[\tau_1,\tau_2)}$ are the temporal operators "Globally" ("always") and "Finally" ("eventually"), respectively.

The *semantics* of iPSTL is recursively defined as

$$\begin{aligned}
s[t] &\models (y_s \sim \pi) && \text{iff} && y_s(t) \sim \pi \\
s[t] &\models \phi_1 \wedge \phi_2 && \text{iff} && s[t] \models \phi_1 \text{ and } s[t] \models \phi_2 \\
s[t] &\models \phi_1 \vee \phi_2 && \text{iff} && s[t] \models \phi_1 \text{ or } s[t] \models \phi_2 \\
s[t] &\models G_{[\tau_1,\tau_2)}(y_s \sim \pi) && \text{iff} && y_s(t') \sim \pi \\
& && && \forall t' \in [t+\tau_1, t+\tau_2) \\
s[t] &\models F_{[\tau_1,\tau_2)}(y_s \sim \pi) && \text{iff} && \exists t' \in [t+\tau_1, t+\tau_2) \\
& && && \text{s.t. } y_s(t') \sim \pi.
\end{aligned}$$

Formula (2a) can be read as "At some instance t in the time interval $[\tau_1, \tau_2)$, an event described by φ_i occurs." A *valuation* θ is a mapping that assigns real values to the parameters appearing in a iPSTL formula. Any valuation θ of an iPSTL formula induces an STL formula. For example, given $\varphi = F_{[\tau_1,\tau_2)}(y_s \geq \pi)$ and $\theta([\tau_1, \tau_2, \pi]) = [0, 4, 5]$, we have $\phi_\theta = F_{[0,4)}(y_s \geq 5)$. In this paper, we use the valuation function θ and its output (the assigned values) interchangeably. We call the fragment of all such STL formulae inference STL (iSTL). Given an iPSTL formula φ and a valuation θ, we denote the corresponding iSTL formula as ϕ_θ.

B. Signed Distance and Robustness Degree

A *signed distance* from a signal $s \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$ to a set $S \subseteq \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$ is defined as

$$D_\rho(s, S) := \begin{cases} -\inf\{\rho(s, s') | s' \in cl(S)\} & \text{if } s \notin S \\ \inf\{\rho(s, s') | s' \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n) \setminus S\} & \text{if } s \in S \end{cases}$$

with $cl(S)$ denoting the closure of S, ρ is a metric defined as

$$\rho(s, s') = \sup_{t \in T} \{d(s(t), s'(t))\} \tag{3}$$

and d corresponds to the metric defined on the domain $\mathbb{R}^n$ of signal s. We use $D(s, \phi)$ to denote $D_\rho(s, L(\phi))$ if ρ is clear from the context.

The robustness degree of a signal s with respect to an STL formula ϕ at time t is given as $r(s, \phi, t)$ [9], [11]. The robustness degree can be calculated according to the following recursive quantitative semantics:

$$\begin{aligned}
r(s, (y_s \geq c_1), t) &= y_s(t) - c_1 \\
r(s, (y_s < c_1), t) &= c_1 - y_s(t) \\
r(s, \phi_1 \wedge \phi_2, t) &= \min(r(s, \phi_1, t), r(s, \phi_2, t)) \\
r(s, \phi_1 \vee \phi_2, t) &= \max(r(s, \phi_1, t), r(s, \phi_2, t)) \\
r\left(s, G_{[c_1,c_2)}\phi, t\right) &= \min_{t' \in [t+c_1, t+c_2)} r(s, \phi, t') \\
r\left(s, F_{[c_1,c_2)}\phi, t\right) &= \max_{t' \in [t+c_1, t+c_2)} r(s, \phi, t')
\end{aligned}$$

where the c_i are real values. We use $r(s, \phi)$ to denote $r(s, \phi, 0)$. The sign of $r(s, \phi)$ tells whether (positive) or not (negative) s satisfies ϕ. The magnitude of $r(s, \phi)$ gives a measure of how different s would have to be in order for its satisfaction of ϕ to change.

C. Expressivity

iPSTL can be used to express a wide range of important system properties, such as

- Bounded-time invariance, e.g., $F_{[0,\tau_1)}(G_{[\tau_2,\tau_3)}(y_s < \pi))$ ("There exists a time $t \in [0, \tau_1)$ such that y_s will always be less than π in $[t + \tau_2, t + \tau_3)$.")
- Reachability to multiple regions in the state space, e.g., $F_{[0,\tau_1)}\left(F_{[\tau_2,\tau_3)}(y_s \geq \pi_1) \vee F_{[\tau_2,\tau_3)}(y_s < \pi_2)\right)$ ("There exists a time $t \in [0, \tau_1)$ such that eventually y_s is either less than π_1 or greater than π_2 from $t + \tau_2$ seconds to $t + \tau_3$ seconds.")

D. Properties of iPSTL

In this subsection, we first define a partial order over iPSTL, the set of all iPSTL formulae. The formulae in iPSTL can be organized in a directed acyclic graph (DAG) where a path exists from formula φ_1 to formula φ_2 iff φ_1 has a lower order than φ_2. The DAG representation plays a key role in our learning algorithms.

1) Partial Orders Over iSTL and iPSTL: We define two relations $\preceq_S$ and $\preceq_P$ for iSTL formulae and iPSTL formulae, respectively.

Definition 1:

1) For two iSTL formulae ϕ_1 and ϕ_2, $\phi_1 \preceq_S \phi_2$ iff $\forall s \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n), s \models \phi_1 \Rightarrow s \models \phi_2$, i.e., $L(\phi_1) \subseteq L(\phi_2)$.
2) For two iPSTL formulae φ_1 and φ_2, $\varphi_1 \preceq_P \varphi_2$ iff $\forall \theta$, $\phi_{1,\theta} \preceq_S \phi_{2,\theta}$, where the domain of θ is $\Theta(\varphi_1) \cup \Theta(\varphi_2)$, the union of parameters appearing in φ_1 and φ_2.

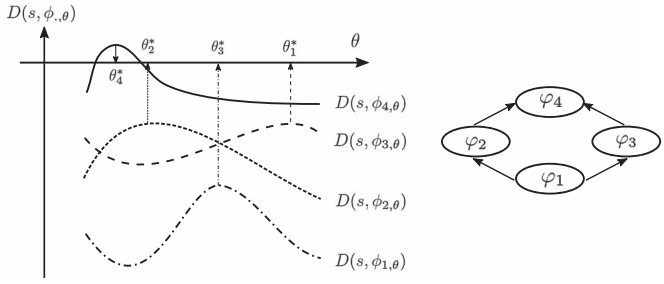

Fig. 1. Illustration of the relationship between iPSTL formulae and signed distances.

Based on these definitions and the semantics of iSTL and iPSTL, we have

Proposition 1: Both $\preceq_S$ and $\preceq_P$ are partial orders.

Proof: See Appendix A. □

Further, we have

Proposition 2: The partial order $\preceq_P$ satisfies the following properties.

1) $\varphi_1 \wedge \varphi_2 \preceq_P \varphi_j \preceq_P \varphi_1 \vee \varphi_2$ for $j = 1, 2$
2) $G_{[\tau_1,\tau_2)}\ell \preceq_P F_{[\tau_1,\tau_2)}\ell$, where ℓ is a linear predicate.

The first property is an extension of the propositional logic rules $A \wedge B \Rightarrow A \Rightarrow A \vee B$. The second property states "If a property is always true over a time interval, then it is trivially true at some point in that interval."

2) DAG and Signed Distance: The structure of iPSTL and the definition of the partial order $\preceq_P$ enable the following theorem.

Theorem 1: The formulae in iPSTL have an equivalent representation as nodes in an infinite DAG. A path exists from formula φ_1 to φ_2 iff $\varphi_1 \preceq_P \varphi_2$. The DAG has a unique top element $(\top)$ and a unique bottom element $(\bot)$.

Proof: See Appendix B. □

An example of such a DAG is shown in Fig. 5.

Next, we establish a relationship between the signed distance of a signal s with respect to iSTL (iPSTL) formulae $\phi(\varphi)$ and the partial order $\preceq_S$ $(\preceq_P)$.

Theorem 2: The following statements are equivalent:

1) $\phi_1 \preceq_S \phi_2$;
2) $\forall s \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$, $D(s,\phi_1) \leq D(s,\phi_2)$.

Proof: See Appendix C. □

Corollary 1: The following statements are equivalent:

1) $\varphi_1 \preceq_P \varphi_2$;
2) $\forall s \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n), \forall\theta$, $D(s,\phi_{1,\theta}) \leq D(s,\phi_{2,\theta})$.

Corollary 1 is illustrated in Fig. 1. The formulae are organized according to the relation $\varphi_1 \preceq_P \varphi_2, \varphi_3 \preceq_P \varphi_4$, which means that $D(s,\phi_{1,\theta}) \leq D(s,\phi_{2,\theta})$, $D(s,\phi_{3,\theta}) \leq D(s,\phi_{4,\theta})$ for all valuations θ.

Remark 1: It has been shown in [11] that a robustness degree $r(s,\phi)$ is an under-approximation of its corresponding signed distance $D(s,\phi)$. If $D(s,\phi_1)$ and $D(s,\phi_2)$ are replaced by $r(s,\phi)$ and $r(s,\phi)$ in Theorem 2, it is still true that 2) implies 1) but the implication from 1) to 2) doesn't always hold. As a counterexample, consider $\phi_1 = G_{[0,1]}(x_s \geq 1) \wedge G_{[0,1]}(x_s < 1)$ and $\phi_2 = G_{[0,1]}(x_s \geq 2) \wedge G_{[0,1]}(x_s < 2)$. It is clear that $\phi_1 \preceq_S \phi_2$. However, for a constant signal s with $x_s(t) = 0, \forall t \in [0,1]$, we have $r(s,\phi_1) = -1 > r(s,\phi_2) = -2$, which is in contradiction with 2) implies 1).

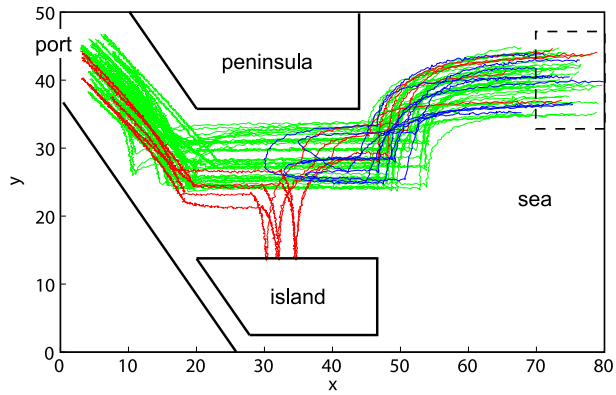

Fig. 2. A naval surveillance example. Trajectories of vessels behaving normally are shown in green. The red trajectories represent possible human trafficking scenarios and the blue trajectories represent possible terrorism scenarios. The layout resembles that of Boston harbor.

IV. Problem Statement

In this section, we give a naval surveillance scenario that will serve as a running example throughout the rest of this paper and present the problems under consideration.

A. A Motivating Example

Example 1 (Naval Surveillance): In maritime surveillance [19], the Automatic Identification System (AIS) enables law enforcement authorities to collect data at regular intervals on a large number of ships. The available data includes the vessels' locations, courses, speeds, and destinations. This information can be used to uncover security threats and suspicious activities such as drug smuggling, human trafficking, arm trading, or terrorism. However, high volumes of traffic make manual inspection of the collected data for anomalous behavior time-consuming and error-prone. Thus there is a need for systems that automate the process of detecting and responding to anomalous events. Consider the academic example shown in Fig. 2.

Normal Behavior: A vessel behaving normally (i.e., its outputs belong to $L_N(\mathcal{S})$) approaches from the sea until it reaches the narrow passage between the peninsula and the island. Then, it heads directly towards the port. Some sample trajectories are shown in green in Fig. 2.

Anomalous Behavior: Consider two scenarios modified from [19] that may indicate illicit actions, i.e., the vessels' trajectories belong to $L_A(\mathcal{S})$. In the first scenario, a vessel (shown in red in Fig. 2) deviates to the island. This may indicate a human trafficking scenario in which the vessel initially follows a normal track, then heads to the shore to pick up people before returning to its original path. In the second scenario, a vessel (shown in blue in Fig. 2) approaches a ferry, loiters, and then quickly returns to the open sea. This behavior may indicate terroristic activity in which the vessel plants an explosive device on or near the ferry. □

The motivating example requires us to learn a classifier that differentiates desired behaviors from undesirable behaviors. A single output of each system can have a large number of data points, which means that finding a classifier using traditional machine learning methods would require the definition of features. For example, we could use time to reach a point in the state space in the maritime example or frequency of oscillation in the brake example. However, these features or set of features must be defined by some expert with knowledge of the problem domain (or by visual data inspection).

Our methods are able to solve each of these problems directly without using such user-defined features, thus minimizing the need for an expert in the anomaly learning and detection processes. The key is to learn an iPSTL formula as a classifier. To give an example, the normal vessel behavior can be described by the iPSTL formula

$$\varphi = F_{[0,590)}\left(G_{[0,200)}(y_s \geq 20) \wedge G_{[0,200)}(y_s < 35) \wedge F_{[0,350)}(x_s < 25)\right)$$

As shown in Fig. 2, the two scale parameters related to y_s, 20 and 35, define the bounds of the normal traces corresponding to the narrow passage between the peninsula and the island. The scale parameter related to x_s, 25, defines the right boundary of the port. In plain English, this formula reads "There is a time t within [0, 590) such that the vessel's y coordinate should always be between 20 and 35 for the next 200 units and the vessel will eventually reach an x coordinate that is less than 25 within 350 units". The people smuggling scenario shown in red in Fig. 2) violates the conjunction of the first and second clauses while the terrorism scenario (shown in blue in Fig. 2) violates the third clause.

B. Off-Line Anomaly Learning

We wish to construct a classifier that can separate outputs from a system behaving normally from outputs from a system behaving abnormally. First we consider the case in which our inference procedure can learn from historical data that has been labeled according to whether or not it represents a normal behavior. More formally, we wish to solve Problem 1.

Problem 1: Let $\{x_i\}_{i=1}^M$ be a set of trajectories generated by $\mathcal{S}$. Let s_i be the observed output signal associated with x_i and p_i be the corresponding label assigned by expert or database knowledge. $p_i = 1$ if s_i represents a normal behavior and $p_i = -1$ if s_i represents an anomalous behavior. From the pairs $\{(s_i, p_i)\}_{i=1}^M$, find an iSTL formula ϕ_N describing normal behaviors (with subscript N indicating "normal" behavior) such that the misclassification rate

$$MR\left(\{(s_i,p_i)\}_{i=1}^M, \phi_N\right) = \frac{FA + MD}{M} \tag{4}$$

is minimized, where $FA = |\{s_i | s_i \not\models \phi, p_i = 1)\}|$ is the number of false alarms (signals improperly classified as anomalous) and $MD = |\{s_i | s_i \models \phi, p_i = -1\}|$ is the number of missed detections (signals improperly classified as normal). In the above $|\cdot|$ denotes the cardinality of a set.

This problem is modified from the supervised learning problem previously addressed in [18]. In [18], we made the assumption that the classifying formula would have the form $\phi = F_{[\tau_1,\tau_2)}(\phi_c \Rightarrow \phi_e)$, where ϕ_c is called the cause formula and ϕ_e is called the effect formula. We assumed that some change in the first part of the system's output (before some time $\tilde{t}$) resulted in some observable phenomenon later. Thus, we used the data after time $\tilde{t}$ to learn ϕ_e and then used all of the data to learn ϕ_c. The problem and solution presented in this paper is more general and doesn't require us to define the time $\tilde{t}$. When an expert has a good estimate for the time $\tilde{t}$, the computation time may be lower using the previous method, but if the assumptions of [18] are not met, the algorithm presented in this paper provides a more general and robust solution.

C. Off-Line Anomaly Detection

Now, we consider the more challenging problem in which the inference procedure learns from historical data without the knowledge of whether a given output was produced by a system behaving normally or abnormally. More formally, we wish to solve Problem 2.

Problem 2: From the set $\{s_i\}_{i=1}^M$ (defined in Problem 1), find an an iSTL formula ϕ_N describing normal behaviors such that the misclassification rate

$$MR_D\left(\{s_i\}_{i=1}^M, \phi_N\right) = \frac{FA_D + MD_D}{M}$$

is minimized, where $FA_D = |\{s_i | s_i \not\models \phi, x_i \in L_N(\mathcal{S})\}|$ and $MD_D = |\{s_i | s_i \models \phi, x_i \notin L_N(\mathcal{S})\}|$.

This problem was previously addressed in [16].

D. On-Line Learning

So far, we have assumed that the inference procedures have access to historical system data. However, this assumption is not valid for systems that have not previously been deployed or from which no data has been recorded. For these systems, we need a way to perform on-line supervised learning. That is, we need to be able to construct and update a classifying formula as more data becomes available over time. More formally, on-line supervised learning is defined by Problem 3.

Problem 3: A system or a group of systems produce outputs s_i with expert-given labels p_i as defined in Problem 1. Maintain a formula ϕ_N^t such that the misclassification rate $MR(\{(s_i,p_i)\}_{i=1}^t, \phi_N^t)$ as defined in Problem 1 is minimized. When a new pair (s_{t+1}, p_{t+1}) becomes available, use ϕ_N^t and the new pair to construct ϕ_N^{t+1}.

Problem 3 has not previously been addressed. An on-line version of the anomaly detection problem can be similarly proposed. However, due to the inherent difficulty of this problem, we will address it in the future.

V. Solutions

In this section, we show how to solve the anomaly learning, anomaly detection and online learning problems presented in Section IV.

A. Learning as Optimization With Robustness Degree

The misclassification rate used in Problem 1–3 is called 0-1 loss in the machine learning literature [4], [31]. One issue with the misclassification rate is it ignores the degree of "wrongness" for misclassified samples, i.e., trajectories. Suppose we have two anomalous signals, s_1 and s_2. Both are misclassified as normal, but with s_1 barely violating the formula ϕ specifying normal behavior while s_2 greatly violates ϕ ($D(s_2, \phi) \ll D(s_1, \phi) < 0$). The consequences of these two mistakes can be dramatically different. Thus, we need to take different samples' *degrees of misclassification* into consideration.

Ideally, one natural choice for such a degree is signed distance $D(s, \phi)$, which can be used as a *fitness function* (or *utility function*). Then, the learning problems under consideration can be converted into optimization problems. According to Theorem 1 and Corollary 1, the optimization problem can be solved by combining a discrete search over a DAG to find an iPSTL formula φ with a continuous search to find its appropriate parameterization θ.

Fig. 1 conceptually illustrates the solution to this problem. A formula is sought to describe the single normal output s. The discrete search starts from the most exclusive formula and follows directed edges until a satisfying formula is found. A continuous search is performed on each node to find a valuation θ that maximizes $D(s, \phi_{i,\theta})$, where i is the index of the current node. It can be observed that the formulae induced from optimal valuations (denoted with * superscripts) of formulae $\varphi_1, \varphi_2, \varphi_3$ are all still violated by s (have negative signed distances). Thus, we have to go up the DAG to formula φ_4 to find a formula that s "barely" satisfies, i.e., a formula with a small yet positive signed distance.

Unfortunately, $D(s, \phi)$, or more precisely $L(\phi)$, cannot be computed or represented analytically for most real systems [11]. Thus, its approximation $r(s, \phi)$ has been used as a surrogate. Even though $r(s, \phi)$ is an approximation of $D(s, \phi)$ [11], $r(s, \phi)$ has been utilized successfully in model checking, analysis and formal synthesis of a wide range of systems [3], [9], [10], [13], [24].

In this paper, we also use robustness degree $r(s, \phi)$ to approximate $D(s, \phi)$. Fig. 3 illustrates the interaction between the graph search and parameter estimation using robustness degree. Suppose we have a single boat's trajectory s, whose x and y coordinates are shown in the top left and right plots, respectively. The center left figure shows the robustness degree with respect to $\varphi_1 := F_{[0,\tau)}(x_s > 100)$ for various values of τ, while the center right figure shows the robustness degree with respect to $\varphi_2 := F_{[0,40)}(y_s < \pi)$ for various values of π. Note that by selecting the parameter τ or π for each φ_i, we can maximize or minimize the robustness degree of the signal with respect to the induced formula $\phi_{i,\theta}$. The bottom left plot shows the robustness degree for $\varphi_3 := \varphi_1 \wedge \varphi_2$ for various pairs (τ, π) and the bottom right plot shows the robustness degree with respect to $\varphi_4 := \varphi_1 \vee \varphi_2$. Note that $\varphi_3 \preceq_P \varphi_1(\varphi_2) \preceq_P \varphi_4$. By considering φ_3 rather than φ_1 or φ_2 alone, we can find a larger class of iSTL formulae that strongly violate the specification, which is useful for mining formulae with respect to undesirable behavior. Similarly, by considering φ_4, we can find a larger class of formulae that robustly satisfy the behavior.

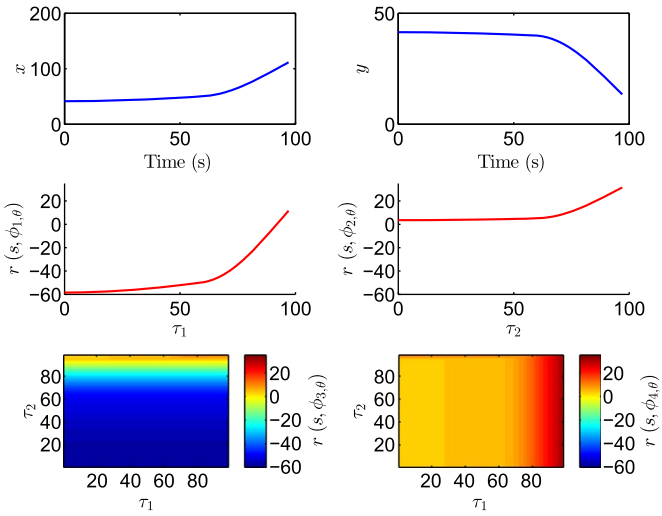

Fig. 3. Simple example of formula search using robustness degree.

Remark 2: For a given signal s, an iPSTL formula φ is either monotonically increasing or monotonically decreasing with respect to its parameter θ [15]. Take $\varphi := F_{[0,40)}(y < \pi)$ for instance. The robustness degree of φ for a particular signal s is shown in the center right plot of Fig. 3. It can be observed that if $\pi_1 > \pi_2$, $r(s, \phi_{\pi_1}) \geq r(s, \phi_{\pi_2})$, a case of monotonic increase. The monotonic property alludes to bisection search heuristics [7]. We choose simulated annealing to solve offline learning problems (Section V-B and C) due to its proven success in others' work [1]. The monotonic property also justifies our utilization of stochastic gradient descent to solve online learning problems (Section V-D).

B. Anomaly Learning

1) Optimization: Problem 1 can be cast as the following optimization problem.

Problem 4: Find an iSTL formula ϕ_{N,θ_N} such that the iPSTL formula φ_N and valuation θ_N minimize

$$J_a(\varphi, \theta) = \frac{1}{M} \sum_{i=1}^{M} l\left(p_i, r(s_i, \phi_\theta)\right) + \lambda \|\phi_\theta\| \tag{5}$$

where r is the robustness degree defined in Section III-A, ϕ_θ is derived from φ with valuation θ, M is the number of labeled signals, λ is a weighting parameter, $\|\phi_\theta\|$ is the length of ϕ_θ (number of linear predicates that appear in ϕ_θ) and l is a loss function, which is chosen to be hinge loss [31] in our case

$$l\left(p_i, r(s_i, \phi_\theta)\right) = \max\left(0, \epsilon_r - p_i r(s_i, \phi_\theta)\right) \tag{6}$$

where $\epsilon_r \ll 1$.

We continue l by using the robustness degree as an intermediary *fitness function*, a measure of how well a given formula fits observed data. Formula length is penalized in our approach because if ϕ_{N,θ_N} grows arbitrarily long, it becomes as complex to represent as the data itself, which would render the inference process redundant.

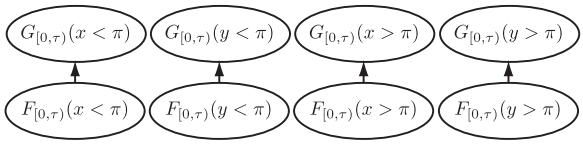

Fig. 4. The initial graph $\mathcal{G}_1$ constructed from x, y coordinates.

2) Algorithm: The framework for solving Problem 1 is detailed in Alg. 1.

Algorithm 1: Anomaly Learning

Input:
A set of labeled signals $\{(s_i, p_i)\}_{i=1}^{M}$;
A variable set V;
A misclassification rate threshold δ;
A formula length bound W
Output:
An iPSTL formula φ and valuation θ.

```
1: for i = 1 to W do
2:   if i = 1 then
3:     G_1 ← DAGInitialization(V);
4:     List ← ListInitialization(G_1);
5:   else
6:     G_i ← PruningAndGrowing(G_{i−1});
7:     List ← Ranking(G_i \ G_{i−1}, );
8:   while List ≠ ∅ do
9:     φ ← List.pop();
10:    (θ, MR) ← ParameterEstimation({(s_i, p_i)}_{i=1}^M, φ)
11:    if MR ≤ δ then
12:      return (φ, θ).
13: return MinimumCostNode(G_W);
```

Remark 3: In this section, for compactness, we denote a formula $\phi = F_{[0,T)}\phi_i$ in the iPSTL fragment by ϕ_i for short. For instance, when we show a formula $G_{[\tau_1,\tau_2)}(x_s < \pi)$ as a node in a DAG, the iPSTL formula it represents is $F_{[0,T)}(G_{[\tau_1,\tau_2)}(x_s < \pi))$.

Initialization: Our algorithm operates on V, the set of all variables represented in the output signals from the system. The inference process begins in line 3 of Alg. 1, where DAGInitialization(V) generates the basis of the candidate formulae. The basis is a set of linear predicates with temporal operators, called *basis nodes*, of the form $O_{[\tau_1,\tau_2)}(x_s \sim \pi_1)$ where $O \in \{G, F\}$, $\sim \in \{\geq, <\}$ and $x_s \in V$. That is, the basis represents all internal formulae φ_i of length 1. Edges are constructed from φ_j to φ_k in the initial graph $\mathcal{G}_1$ iff $\varphi_j \preceq_P \varphi_k$. For example, in the naval surveillance example if we only consider the (x, y) position of the boat, then the initial graph is shown in Fig. 4.

ListInitialization$(\mathcal{G}_1)$ (line 4) generates a ranked list of formulae from the basis nodes. Since we do not yet know anything about how well each of the basis nodes classifies behaviors, the rank (used in parameter estimation) is generated randomly.

Parameter Estimation: After the graph is constructed, we find the optimal parameters for each of the nodes. The candidate formulae in List are iterated through from lowest rank to highest (line 9). ParameterEstimation$(\{(s_i, p_i)\}_{i=1}^{M}, \varphi)$(line 10) uses simulated annealing to find an optimal valuation for φ by minimizing the cost J_a.

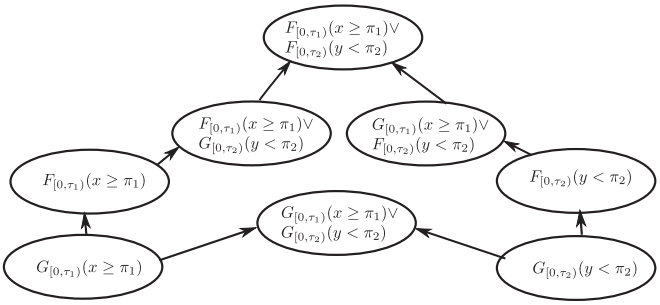

Fig. 5. A subset of the DAG $\mathcal{G}_2$ after pruning and expansion. For compact representation, only the internal formulae φ_i are shown and the unique top element $(\top)$ and the unique bottom element $(\bot)$ are not illustrated. This DAG corresponds to expanding the graph from formulae of length 1 to formulae of length 2 in order to search for a formula with a larger, more inclusive language.

Structural Inference: After the first set of parameters and costs have been found, the iterative process begins. The definition of the partial order allows for dynamic extension of the formula search space. We cannot explicitly represent the infinite DAG, so we construct a finite subgraph of possible candidate formulae and expand it when the candidate formulae perform insufficiently. In machine learning, subset selection is the problem of selecting an explanatory subset of features that best classifies sets of data. Feature subset selection (FSS) is an iterative heuristic solution in which at each iteration, the best-performing features are retained in the candidate explanatory subset while the worst-performing features are discarded [31]. PruningAndGrowing$(\mathcal{G}_{i-1})$ (line 7) first applies the principle of FSS by eliminating a fixed number of high-cost nodes from $\mathcal{G}_{i-1}$, i.e., those formulae that do not fit the observed data. It then grows the graph $\mathcal{G}_{i-1}$ to include nodes with length i according to graph expansion rules derived from Proposition 2. Consider a formula φ_j of length $\ell - 1$ such that the missed detection rate of ϕ_{j,θ_j^*} is higher than the false alarm rate. This means that the inferred formula needs to be made less restrictive, i.e., its language needs to be enlarged to include more of the desirable traces. In order to do this, PruningAndGrowing adds to the graph a formula $\varphi' = \varphi_j \vee \varphi_b$ where $\varphi_b \in$ Basis is a basis formula with good performance. If the false alarm rate is higher, then a more restrictive formula would be required and PruningAndGrowing would add $\varphi'' = \varphi_j \wedge \varphi_b$ to the graph. An example of a subset of a graph $\mathcal{G}_2$ grown from the (pruned) basis graph is given in Fig. 5.

Ranking$(\mathcal{G}_i \setminus \mathcal{G}_{i-1})$ (line 8) ranks the newly grown nodes based on a heuristic function

$$\frac{1}{|pa(k_i)|} \sum_{k_{i-1} \in pa(k_i)} J_a(k_{i-1}) \tag{7}$$

where k_i is a node in $\mathcal{G}_{i-1}$, $pa(k_i)$ is the set of k_i's parents, and $|pa(k_i)|$ is the size of $pa(k_i)$. For example, in Fig. 5, for $k_i = (F_{[0,\tau_1)}(x_s \geq \pi_1) \wedge (F_{[0,\tau_2)}(y_s < \pi_2))$, $pa(k_i) = \{F_{[0,\tau)}(x_s \geq \pi), (F_{[0,\tau)}(y_s < \pi)\}$ and $|pa(k_i)| = 2$.

The iterative graph growing and parameter estimation procedure is performed until a formula with low enough misclassification rate is found or W iterations are completed. At this point, MinimumCostNode($\mathcal{G}_i$) returns the node with the minimum cost within $\mathcal{G}_i$.

Complexity: Without pruning, the discrete layer of the described algorithm runs in time $\mathcal{O}(W \cdot 2^{|V|})$. Since PruningAndGrowing prunes a constant number of nodes at each iteration, the complexity of the discrete layer is reduced to $\mathcal{O}(W \cdot |V|^2)$ when pruning is applied. The continuous layer of the algorithm, runs in time $\mathcal{O}(W(n^2 m) \cdot \log(M))$, where n is the number of times the "temperature" (average step size) of the algorithm is lowered during an iteration of the algorithm and m is the number of sampled valuations evaluated at each temperature.

Remark 4: Simulated annealing (SA) is used in this paper to attempt to find the optimum parameter θ^* of $J_a(\varphi, \theta)$ with a fixed iPSTL formula φ. Other global optimization techniques, such as particle swarm optimization and Monte-Carlo sampling, have also been applied to solve similar parameter estimation problems [13], [22]. Even though SA converges in probability to the global optimum, there is no theoretical proof on the convergence rate and how far the solution is from the global optimum after a number of iterations. Some promising avenues include utilizing the monotonicity of the robustness function $r(s, \phi_\theta)$ with respect to θ [15], [33] and using cross-entropy method guided by robustness degree to sample inputs [25].

C. Anomaly Detection

1) Optimization: Since Problem 2 is an unsupervised learning problem, we use some notions from classical unsupervised learning to aid in our approach. In particular, we consider one-class support vector machines (SVMs). A one-class SVM is an optimization technique that, given a set of data, lifts the data to a higher-dimensional feature space and constructs a surface in this space that separates normal data from anomalous data [26]. We map Problem 2 to the following optimization.

Problem 5: Find an iPSTL formula ϕ_{N,θ_N} such that the formula φ_N and valuation θ_N minimize

$$\min_{\phi_\theta, \epsilon} d(\phi_\theta) + \frac{1}{\nu N} \sum_{i=1}^{N} \mu_i - \epsilon \tag{8}$$

such that

$$\mu_i = \begin{cases} 0 & r(s_i, \phi_\theta) > \frac{\epsilon}{2} \\ \frac{\epsilon}{2} - r(s_i, \phi_\theta) & \text{else} \end{cases} \quad \forall i \tag{9}$$

where ϕ_θ is an iSTL formula, ϵ is the "gap" in signal space between outputs identified as normal and outputs defined as anomalous, ν is the upper bound of the *a priori* probability that a signal $x_i \in L_A(\mathcal{S})$ [26], and μ is a slack variable. μ_i is positive if s_i does not satisfy ϕ_θ with minimum robustness $\epsilon/2$. The function d is a "tightness" function that penalizes the size of $L(\phi_\theta)$.

By minimizing the sum of the μ_i, optimization (8) minimizes the number of traces the learned formula ϕ_θ classifies as anomalous. By maximizing the gap ϵ, optimization (8) attempts to maximize the separation between normal and anomalous outputs. By minimizing the function $d(\phi_\theta)$, optimization (8) prevents the learned formula from trivially describing all observed signals (i.e., finding a formula such that $L(\phi) = L(\mathcal{S})$), which would render the optimization redundant.

2) Algorithm: Similar to the anomaly learning case, solving (8) requires searching over the set of continuous variables (θ and ϵ) as well as over the discrete set of iPSTL formula structures (the structure φ of ϕ_θ). Alg. 1 can be adapted to solve (8) with the following two changes:

1) The input signals are not labeled, i.e., the inputs are $\{s_i\}_{i=1}^{M}$.
2) ParameterEstimation solves (8) instead of (5).

The ParameterEstimation procedure uses the heuristic tightness function d when calculating the objective function in (8). In this paper, we use a heuristic that penalizes the size of τ_1, as for monitoring purposes we would prefer to infer formulae that can describe behaviors of the early parts of the system's outputs. For each predicate appearing in ϕ, if the comparison operator is $<$, the size of π is penalized, as the size of the language of $(x_i < \pi)$ increases with π. If the comparison operator is $>$, small values of π are penalized for the same reason. Please see [16] for more details. The time complexity of the anomaly detection algorithm is the same as that of the anomaly learning algorithm.

D. Online Learning

Here, we consider how to extend Alg. 1 to solve Problem 3. In principle, optimization problems such as (5) can be solved for on-line settings via stochastic gradient descent [17], [31]. With mild assumptions, for a fixed iPSTL formula structure φ, such a method can find its optimal parameterization θ^* if there exists a ϕ_{θ^*} with structure φ that can classify the data. Let θ_i be the parameterization of φ after i pairs of signals and labels have been observed. The stochastic gradient descent that minimizes the loss function l is given by

$$\theta_{i+1} = \begin{cases} \theta_i & \text{if } p_i r\left(s_i, \phi_{\theta_i}\right) \geq \epsilon_r \\ \theta_i + \eta \frac{\partial r}{\partial \theta} p_i & \text{otherwise.} \end{cases} \tag{10}$$

where $\eta > 0$ is a learning rate and the partial derivative is calculated according to the centered first difference. If φ is the correct iPSTL formula for classification, it should be expected that there exist a step $\bar{i}$ such that $p_i r(s_i, \phi_{\theta_i}) \geq 0$ for all $i \geq \bar{i}$. That is to say the total misclassification rate approaches 0.[2] If, on the other hand, φ is not the correct formula, then the improvement on classification performance saturates at a certain step $\tilde{i}$ and a new formula should be sought.

We propose an on-line learning procedure described by Algorithm 2. This new procedure can learn a formula ϕ_θ as the labeled signals (s_i, p_i) arriving sequentially. This procedure operates on a collection of N_f candidate formulae $\{\varphi_j, \theta^j\}_{j=1}^{N_f}$. The algorithm operates on this collection instead of considering

[2]The difference between subsequent valuations θ_i and θ_{i+1} should not be too extreme, as this would cause the valuation to oscillate about the optimal value θ^*. Therefore, the learning rate η should be chosen to be a small, positive number or to decrease with respect to the iteration number.

a single formula at a time because there is initially very little information about what kinds of behaviors the system may satisfy. In practice, we choose N_f to be at least as large as the size of the basis so that we do not exclude any rectangular predicate from consideration. It is still more computationally efficient than the offline learning method, however, because for each trajectory and label pair (s_i, p_i) are introduced to the algorithm, N_f robustness calculations are performed, in contrast to the n^2m calculations that are performed by the simulated annealing algorithm.

Algorithm 2: Online Learning

Input:
A sequence of labeled signals $\{(s_i, p_i)\}_{i=1}^{M}$;
Database of candidate STL classifiers formulae;
Maximum and minimum learning rates $\eta_{\max}, \eta_{\min}$;
Geometric rate α;
Number of iterations before updating formula database checkInt;
Maximum number of iterations numIter
Output:
An iPSTL formula φ along with its corresponding valuation θ.

for $\varphi_k \in$ formulae **do**
  $\theta^k \leftarrow$ ParameterEstimation$((s_i, p_i), \varphi_k)$;
for $i = 1, \ldots$ numIters **do**
  traces $\leftarrow$ UpdateTraces(traces, s_i, p_i);
  for (φ_k, θ^k) in formulae **do**
    $\theta^k \leftarrow$ ParameterUpdate$((s_i, p_i), \varphi_k, \theta^k)$;
  $\eta \leftarrow \max(\alpha\eta, \eta_{\min})$;
  if $i \bmod$ checkInt $== 0$ **then**
    formulae $\leftarrow$ UpdateFormulae(formulae);
    $\eta \Rightarrow \eta_{\min}$;
return bestFormula(formulae,traces)

The algorithm initializes the set of formulae to N_f formulae from the basis and corresponding initial valuation guesses that are found via simulated annealing with small values of n, m. Then, when a new trajectory and label pair (s_i, p_i) becomes available, the function ParameterUpdate is called to update each value θ^j in the formula database according to the rule (10). Every checkInt trajectories, the misclassification rates of each φ_{j,θ^j} with respect to the trace database are evaluated and the formula database is then populated with new formulae.

The function BestFormula returns the candidate formula from the database that minimizes the misclassification rate of the $(\phi_{b,\theta^b}, \phi_{sb,\theta^{sb}})$ are therefore the first and second best performing candidate formulae. If the missed detection rates are greater than the false alarm rates, e.g., the size of the languages of the formulae are too large, then the conjunction of the two formulae is added to the formula database. This corresponds to moving several "hops" down the DAG of iPSTL formulae. Otherwise, the disjunction of the two is added. This corresponds to moving several hops up the DAG. The subroutine simplify removes tautologies from the constructed formula. The subroutine getValuation maps the two valuations θ^b, θ^{sb} to the corresponding valuation θ_{new} of the simplified combined formula.

Our algorithm uses a variable learning rate η throughout the on-line inference procedure. We initially start at a high rate $\eta_{\max}$ and decrease it in a geometric fashion with rate $0 \ll \alpha < 1$ until it reaches a level $\eta_{\min}$. Whenever the formula database is replaced, this rate is reset to its original level. The variable learning rate allows the ParameterUpdate formula to make bigger steps whenever we initially know very little about the optimal parameterizations for each structure and make smaller, finer steps after more information has been collected.

Algorithm 3: UpdateFormulae

Input:
trace database tr;
formula database f;
Output:
updated database f

for $k = 1$ to N_f **do**
  $(\varphi_b, \theta^b) \leftarrow$ bestFormula(f,tr)
  uf1 $=$ uf1 $\cup \{(\varphi_b, \theta^b)\}$
  (mdb, fab) $\leftarrow$ calculateRates$(\varphi_b, \theta^b,$ tr)
  for $m = 1$ to $N_f - k$ **do**
    $(\varphi_{sb}, \theta^{sb}) \leftarrow$ bestFormula(f, tr $\setminus$ (uf1 $\cup$ uf2), tr)
    (mdsb, fasb) $\leftarrow$ calculateRates$(\varphi_{sb}, \theta^{sb},$ tr)
    if $(mdsb + mdb > fab + fasb)$ **then**
      $\varphi_{\text{new}} =$ simplify$(\varphi_b \wedge \varphi_{sb})$
    else
      $\varphi_{\text{new}} =$ simplify$(\varphi_b \vee \varphi_{sb})$
    $\theta_{\text{new}} =$ getValuation$(\varphi_{\text{new}}, \theta^b, \theta^{sb})$
    f $=$ f $\cup \{(\varphi_{\text{new}}, \theta_{\text{new}})\}$
    uf2 $=$ uf2 $\cup (\varphi_{sb}, \theta_{sb})$
return f;

Complexity: As mentioned above, each time a new trajectory and label pair is introduced to the online learning procedure, $\mathcal{O}(N_f)$ robustness calculations with complexity $\mathcal{O}(|\varphi_j|)$ are performed. In contrast, a robustness calculation is performed n^2m times for each candidate formula and trace in the off-line algorithm. It is difficult to directly compare the two complexities due to variations in parameter sizes, but in practice many more robustness computations are performed in off-line learning. If N_r formula database updates are performed, the worst-case formula length is 2^{N_r-1}, though this represents an unlikely extreme situation in which no tautologies are introduced and the longest formulae in the database are always among the best-performing. The user has some control over the maximum length of the formula via the parameter checkInt which determines how often the formula replacement occurs. In practice, the maximum formula length is determined by the length of the shortest formula that can separate the two classes of trajectories.

VI. Implementation and Case Studies

The algorithms described in Section V were implemented as a software tool called TempLogIn (TEMPoral LOGic

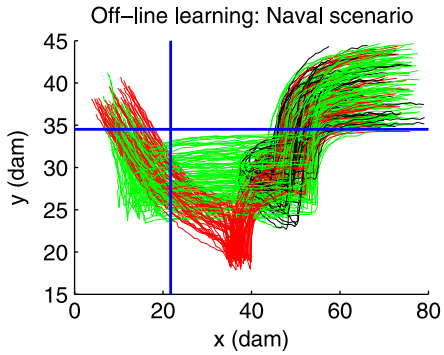

Fig. 6. Results of offline inference. The green trajectories represent normal behaviors, the red trajectories represent human trafficking, and the black trajectories represent terroristic activity. The blue lines are boundaries given by the formula (12). The values of x and y are expressed in decameters (dam) with 1 dam equal to 10 m.

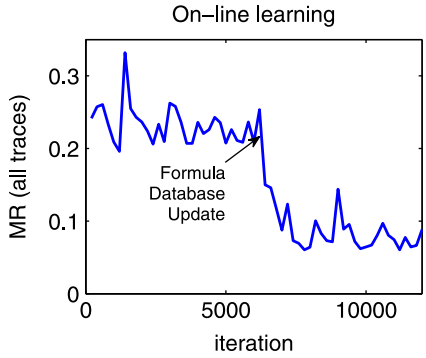

Fig. 7. The misclassification rate over time for on-line learning with respect to all traces.

INference) in MATLAB. We developed all of the components of our solution in-house, including the graph construction, search algorithms, and the simulated annealing algorithm. The software is available at http://hyness.bu.edu/Software.html.

A. Naval Surveillance

1) Scenario Setup: In this sub-section, we use our anomaly learning and on-line learning algorithms to the naval surveillance scenario used in Example 1. We model each vessel as a Dubins' vehicle

$$\begin{cases} \dot{x} &= v\cos\alpha \\ \dot{y} &= v\sin\alpha \\ \dot{\alpha} &= \omega \end{cases} \tag{11}$$

where x and y are the vessel's coordinates, v is its constant speed, α is its heading, and ω is its angular velocity.[3] Further, assume that the x and y coordinates collected by the AIS are subjected to an additive white Gaussian noise $\mathcal{N}(0, 0.1)$.

2) Anomaly Learning: We generated 50 trajectories demonstrating normal behaviors, 25 trajectories demonstrating suspicious behaviors consistent with human trafficking behaviors, and 25 trajectories demonstrating suspicious behaviors consistent with terroristic behaviors. A subset of these trajectories are shown in Fig. 6. Our goal was to find a formula that described only the normal behaviors from this training set. Using our implementation of Alg. 1 with $n = 15$, $m = 15$ yielded the formula

$$\phi_N = F_{[0,320)}\big(G_{[28,227)}y_s > 21.73\big) \wedge \big(G_{[308,313)}x_s < 34.51\big) \tag{12}$$

with total misclassification rate 0.0950. The total computation time was 1313 s (approx. 22 minutes) on a computer with 2.41 GHz processor and 7.4 GB RAM.

In plain English, this formula reads "Within 320 minutes, there exists a time t such that the boat's y coordinate remains above 21.73 dam between $t + 28$ minutes and $t + 227$ minutes and such that the x coordinate remains below 34.51 dam between $t + 308$ minutes and $t + 313$ minutes." The blue lines in Fig. 6 correspond to the thresholds in (12).

3) On-Line Learning: Next, we used a larger set of signals of the naval scenario and inferred a formula by using our on-line learning algorithm. At each iteration of our algorithm, we drew a signal uniformly at random from a set of 2000 trajectories, which consisted of 1000 normal trajectories, 500 human trafficking trajectories, and 500 terrorist trajectories. The learning rate parameters we used were $\alpha = 0.995$, $\eta_{\max} = 0.2$, and $\eta_{\min} = 0.01$. Fig. 7 shows the misclassification rate of the inferred formula at time i with respect to all 2000 traces. As we can see, the rate does not monotonically decrease, but it does decrease to a point. The formula that was inferred after all 2000 trajectories were used is

$$\phi_N = F_{[0,320)}\big(G_{[174,228)}y_s > 19.88\big) \wedge \big(G_{[92,297)}x_s < 34.08\big) \tag{13}$$

The total misclassification rate of the final formula was 0.0885. The total computation time was 996 s (approximately 16 minutes) on a computer with 2.41 GHz processor and 7.4 GB RAM. This computation time included evaluating misclassification rates for the entire set at certain time intervals, i.e., generating Fig. 7. Using the same parameters and not doing this calculation yields a computation time of 648 s (approx. 11 minutes). This represents a significant computational speedup over the off-line method, especially when we consider that the method operated on a larger data set.

B. Train Network Monitoring

1) Model: Consider a train using an electronically-controlled pneumatic (ECP) braking system. The train has 3 cars, each of which has its own braking system. We model the train as a classical hybrid automaton, whose definition is given in [20]. In this model, the braking system is automated to regulate the velocity v below unsafe speeds and above low speeds to ensure that the train reaches its destination, as shown in the top-left sub-figure of Fig. 8. However, an adversary can disable the brakes of the system and cause its velocity to become unregulated, as shown in the other three sub-figures of

[3]The simple dynamics (11) is chosen for reader familiarity. The choice of simulated dynamics does not affect the validity of our results, as our algorithm depends on labeled traces and not explicit system models.

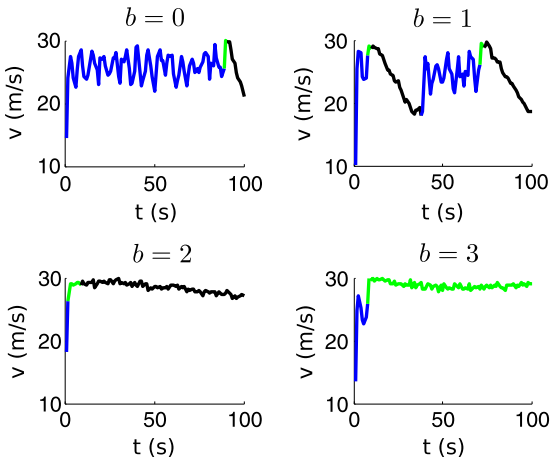

Fig. 8. Outputs of the train velocity system under normal (upper-left) and attack scenarios (other three). An adversary has the ability to disable one, two, or three of the trains brakes in order to deregulate its velocity. The variable b is the number of brakes affected by attack.

Fig. 8. More details on the particular model we used can be found in [16], [28].

2) Anomaly Detection: We used the model given in the previous section to generate 50 outputs of H. 43 of the trajectories were from normal operation and 7 were from an attacked operation. We only considered attacks in which all of the brakes were disabled ($b = 3$). Our algorithm inferred the formula

$$\phi = F_{[0,100)}\left(F_{[10,69)}(v_s < 24.9) \wedge F_{[13.9,44.2)}(v_s > 17.66)\right). \tag{14}$$

In plain English, (14) means "Within 100 seconds, there exists a time t, such that between $t + 10$ s and $t + 69$ s in the future the velocity dips below 24.9 m/s and such that the velocity exceeds 17.66 m/s between $t + 13.9$ s and $t + 44.2$ s." This formula is consistent with the observed un-attacked signals (shown in top left of Fig. 8), as the properly-functioning brake system forces the velocity to be below 24.9 m/s regularly while ensuring that the speed never deviates too far below some desired minimum speed. In contrast, when under attack (Fig. 8, bottom right), this regulation never occurs. and the velocity is allowed to remain about 24.9 m/s indefinitely.

The formula (14) perfectly separates the data, i.e., the misclassification rate is 0. The formula was inferred using 15 simulated annealing cycles with 15 sample points per cycle. The computation time was 154 s on an 8 core PC with 2.1 GHz processors and 8 GB RAM.

VII. Conclusion

In this paper, we brought together techniques from formal methods and machine learning to develop a framework for anomaly learning and detection. For three different scenarios (off line supervised and unsupervised and online supervised learning), we designed and implemented algorithms that infer classifiers in the form of formulae in a specially tailored signal temporal logic. While capturing many features of traditional classifiers, these new types of classifiers include time-based and logical semantics, which resemble natural language. We demonstrated our approach using two case studies, a naval surveillance example and a train braking system. In future work, we plan to exploit the quantitative semantics of the logic for on-line monitoring and anomaly mitigation.

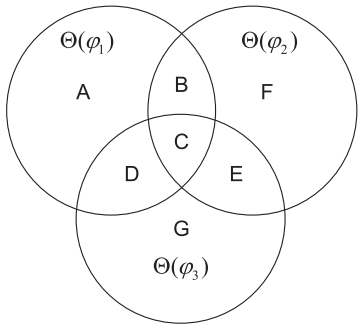

Fig. 9. Relationship among $\Theta(\varphi_1)$, $\Theta(\varphi_2)$ and $\Theta(\varphi_3)$.

Appendix A
Proof of Proposition 1

Proof: A *partial order* $\preceq$ is a binary relation that is reflexive, transitive and antisymmetric.

($\preceq_S$) *Reflexivity* $\phi_1 \preceq_S \phi_1$ is equivalent to $L(\phi_1) \subseteq L(\phi_1)$, which is trivially true. *Transitivity* $\phi_1 \preceq_S \phi_2$ and $\phi_2 \preceq_S \phi_3$ is equivalent to $L(\phi_1) \subseteq L(\phi_2)$ and $L(\phi_2) \subseteq L(\phi_3)$. It implies $L(\phi_1) \subseteq L(\phi_3)$, which means $\phi_1 \preceq_S \phi_3$. *Antisymmetry* $\phi_1 \preceq_S \phi_2$ and $\phi_2 \preceq_S \phi_1$ is equivalent to $L(\phi_1) \subseteq L(\phi_2)$ and $L(\phi_2) \subseteq L(\phi_1)$. It implies $L(\phi_1) = L(\phi_2)$, which means $\phi_1 \equiv \phi_2$.

($\preceq_P$) Regardless of the relationship among formulae φ_1, φ_2 and φ_3, the relationship among their parameter sets $\Theta(\varphi_1)$, $\Theta(\varphi_2)$ and $\Theta(\varphi_3)$ can be generally represented as in Fig. 9. Due to the independence of assignment to each parameter, the valuation of a formula's parameters can be decomposed into the valuations of its parameter subset. For instance, the valuation of formula φ_1 can be written as $\theta = [\theta_A, \theta_B, \theta_C, \theta_D]$, where row vectors θ_A, θ_B, θ_C and θ_D denote the valuations of parameter subsets A, B, C and D, respectively.

Reflexivity: $\varphi_1 \preceq_P \varphi_1$ is equivalent to $\forall\theta$, $\phi_{1,\theta} \equiv \phi_{2,\theta}$ or $L(\phi_{1,\theta}) = L(\phi_{1,\theta})$, which is trivially true.

Transitivity: If $\varphi_1 \preceq_P \varphi_2$ and $\varphi_2 \preceq_P \varphi_3$, we have

$$\begin{aligned}
& \forall\theta = [\theta_A, \theta_B, \theta_C, \theta_D, \theta_E, \theta_F], \phi_{1,\theta} \preceq_S \phi_{2,\theta} \\
& \text{and } \forall\theta' = [\theta'_B, \theta'_C, \theta'_D, \theta'_E, \theta'_F, \theta'_G], \phi_{2,\theta'} \preceq_S \phi_{3,\theta'} \\
\Rightarrow\ & \forall\theta'' = [\theta''_A, \theta''_B, \theta''_C, \theta''_D, \theta''_E, \theta''_F, \theta''_G], \phi_{1,\theta''} \preceq_S \phi_{2,\theta''} \\
& \text{and } \phi_{2,\theta''} \preceq_S \phi_{3,\theta''} \\
\Rightarrow\ & \forall\theta'' = [\theta''_A, \theta''_B, \theta''_C, \theta''_D, \theta''_E, \theta''_F, \theta''_G], \phi_{1,\theta''} \preceq_S \phi_{3,\theta''} \\
& \text{due to transitivity of } \preceq_S \\
\Rightarrow\ & \forall\theta''' = [\theta''_A, \theta''_B, \theta''_C, \theta''_D, \theta''_E, \theta''_G], \phi_{1,\theta'''} \preceq_S \phi_{3,\theta'''} \\
\Rightarrow\ & \varphi_1 \preceq_P \varphi_3.
\end{aligned}$$

Antisymmetry: If $\varphi_1 \preceq_P \varphi_2$ and $\varphi_2 \preceq_P \varphi_1$, we have

$$\begin{aligned}
& \forall\theta, \phi_{1,\theta} \preceq_S \phi_{2,\theta} \text{ and } \phi_{2,\theta} \preceq_S \phi_{1,\theta} \\
\Rightarrow\ & \forall\theta, \phi_{1,\theta} \equiv \phi_{2,\theta} \text{ due to antisymmetry of } \preceq_S \\
\Rightarrow\ & \varphi_1 \equiv \varphi_2
\end{aligned}$$

□

APPENDIX B
PROOF OF THEOREM 1

Proof: A partially ordered set $\langle X, \preceq \rangle$ forms a *lattice* if any two elements $x_1, x_2 \in X$ have a join and a meet [6]. The join and meet can be computed by means of two binary operators, $\sqcup : X \times X \to X$ and $\sqcap : X \times X \to X$, using the supremum and infimum functions, i.e.,

$$\begin{aligned} x_1 \sqcup x_2 &:= \sup\{x_1, x_2\} \\ x_1 \sqcap x_2 &:= \inf\{x_1, x_2\} \end{aligned} \tag{15}$$

Any partially ordered set $\langle X, \preceq \rangle$ with a lattice structure can be represented by a directed acyclic graph (DAG). First, a Hasse diagram [6] can be constructed with each node of the diagram corresponding to an element of X. Then, the DAG can be obtained by adding a direction to each line segment of the Hasse diagram, which point from a "lower" element (in Cartesian coordinates, has a strictly smaller second coordinate) to a "higher" element (has a strictly larger second coordinate). The join (meet) of two elements x_1 and x_2 is the "lowest" ("highest") node where two paths starting from node x_1 and node x_2 and along forward (backward) edges meets.

Proving Theorem 1 is equivalent to proving that the set of iPSTL formulae with partial order $\preceq_P$ form a lattice. More formally,

Lemma 1: For all $\varphi_1, \varphi_2 \in i\text{PSTL}$, where iPSTL is the set of all iPSTL formulae, their join $\varphi_1 \sqcap \varphi_2$ and meet $\varphi_1 \sqcup \varphi_2$ exist and are unique.

Proof:

Join: Treat the subformulae $G_I p$ and $F_I p$ where p is a linear predicate and I is a time interval $I := [\tau_1, \tau_2)$ as different Boolean predicates. Calculate the Disjunctive Normal Form (DNF) of $\varphi_1 \wedge \varphi_2$ [14]. Then, if $G_I p$ and $F_I p$ coexist in a term replace them with $G_I p$; if $G_I p$ and $F_I \neg p$ coexist in a term, we replace them with $\perp$ (False) or equivalently delete the corresponding term; similarly, if $G_I \neg p$ and $F_I p$ coexist in a term, we delete the corresponding term. The resulting formula is the join $\varphi_1 \sqcap \varphi_2$, which is unique because DNFs are unique.

Meet: The existence and uniqueness of $\varphi_1 \sqcup \varphi_2$ can be proved similarly by utilizing the Conjunctive Normal Form (CNF) of $\varphi_1 \vee \varphi_2$. □

Remark 5: Notice that the nodes corresponding to $\varphi_1 := F_{I_1}(F_{I_2} p)$ and $\varphi_2 := F_{I_3}((F_{I_4} p) \vee (F_{I_5} p))$, where $I_1 - I_5$ are time intervals, are different in the DAG. According to the definition of partial order $\preceq_P$ over iPSTL, we have $\varphi_2 \preceq_P \varphi_1$,[4] meaning that the unique meet of the two nodes is the node corresponding to φ_1 and the unique join of the two nodes is the node corresponding to φ_2.

Thus, $\langle i\text{PSTL}, \preceq_P \rangle$ is a lattice and therefore has an equivalent representation as an infinite DAG. □

[4]It is worth pointing out that, during ordering of the two formulas, constraints on the parameters are added. For this particular case, $I_1 = I_3$ and $I_2 = I_4$, meaning the bounds of the paired time intervals should be the same.

APPENDIX C
PROOF OF THEOREM 2

Proof: ($\Rightarrow$) Since $L(\phi_1) \subset L(\phi_2)$, for any $s \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$, there are three possibilities: 1) $s \in L(\phi_1)$; 2) $s \in L(\neg\phi_1) \cap L(\phi_2)$; 3) $s \in L(\neg\phi_1) \cap L(\neg\phi_2)$. Here, $L(\neg\phi_1) := \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n) \setminus L(\phi_1)$ and $L(\neg\phi_2) := \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n) \setminus L(\phi_2)$. For Condition 1

$$\begin{aligned} L(\phi_1) \subset L(\phi_2) &\Rightarrow L(\neg\phi_2) \subset L(\neg\phi_1) \\ &\Rightarrow L(\neg\phi_1) = L(\neg\phi_2) \cup (L(\neg\phi_1) \cap L(\phi_2)) \end{aligned}$$

Thus

$$\begin{aligned} D(s, \phi_1) &= \inf\{\rho(s, y) | y \in cl\,(L(\neg\phi_1))\} \\ &= \inf\{\rho(s, y) | y \in cl(L(\neg\phi_2) \\ &\quad \cup (L(\neg\phi_1) \cap L(\phi_2)))\} \\ &= \inf\{\rho(s, y) | y \in cl\,(L(\neg\phi_2)) \\ &\quad \text{or } y \in cl\,(L(\neg\phi_1) \cap L(\phi_2))\} \\ &= \inf\{\inf\{\rho(s, y) | y \in cl\,(L(\neg\phi_2))\}, \\ &\quad \inf\{\rho(s, y) | y \in cl\,(L(\neg\phi_1) \cap L(\phi_2))\}\} \\ &= \min\{\inf\{\rho(s, y) | y \in cl\,(L(\neg\phi_2))\}, \\ &\quad \inf\{\rho(s, y) | y \in cl\,(L(\neg\phi_1) \cap L(\phi_2))\}\} \\ &\quad \leq \inf\{\rho(s, y) | y \in cl\,(L(\neg\phi_2))\} \\ &= D(s, \phi_2) \end{aligned}$$

Condition 3 can be proved similarly. For Condition 2, since $s \notin L(\phi_1)$ and $s \in L(\phi_2)$, we have $D(s, \phi_1) \leq 0$ and $D(s, \phi_2) \geq 0$. Then, it is true that $D(s, \phi_1) \leq D(s, \phi_2)$.

($\Leftarrow$) Assume otherwise, then there exists an $s \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$ such that $D(s, \phi_1) \leq D(s, \phi_2)$ and $s \in L(\phi_1)$ but $s \notin L(\phi_2)$. Thus, we have $D(s, \phi_1) \geq 0$ and $D(s, \phi_2) \leq 0$, which results a contradiction, since $D(s, \phi_1)$ and $D(s, \phi_2)$ cannot be zero, simultaneously. □

REFERENCES

[1] H. Abbas and G. Fainekos, "Convergence proofs for simulated annealing falsification of safety properties," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput. Allerton*, 2012, pp. 1594–1601.

[2] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, "Parametric identification of temporal properties," in *Runtime Verification*. Berlin, Germany: Springer-Verlag, 2012, pp. 147–160.

[3] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti, "On the robustness of temporal properties for stochastic models," in *Proc. Second Int. Workshop Hybrid Syst. Biology*, vol. 125, 2013, pp. 3–19.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[6] B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*. Cambridge, MA, USA: Cambridge Univ. Press, 2002.

[7] J. Deshmukh, X. Jin, J. Kapinski, and O. Maler, "Stochastic local search for falsification of hybrid systems," in *Automated Technology for Verification and Analysis*. Berlin, Germany: Springer-Verlag, 2015, pp. 500–517.

[8] A. Dokhanchi, B. Hoxha, and G. Fainekos, "On-line monitoring for temporal logic robustness," in *Runtime Verification*. Berlin, Germany: Springer, 2014, pp. 231–246.

[9] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed System*. Berlin, Germany: Springer, 2010, pp. 92–106.
[10] G. E. Fainekos, "Revising temporal logic specifications for motion planning," in *Proc. IEEE Int. Conf. Robot. and Autom. (ICRA)*, IEEE, Shanghai, China, 2011, pp. 40–45.
[11] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Comput. Sci.*, vol. 410, no. 42, pp. 4262–4291, 2009.
[12] J. P. Farwell and R. Rohozinski, "Stuxnet and the future of cyber war," *Survival*, vol. 53, no. 1, pp. 23–40, 2011.
[13] I. Haghighi, A. Jones, Z. Kong, E. Bartocci, R. Gros, and C. Belta, "Spatel: A novel spatial-temporal logic and its applications to networked systems," in *Proc. 18th Int. Conf. Hybrid Syst.: Comput. Control*, ACM, Seattle, WA, 2015, pp. 189–198.
[14] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge Univ. Press, 2004.
[15] X. Jin, A. Donze, J. Deshmukh, and S. Seshia, "Mining requirements from closed-loop control models," in *Proc. 16th Int. Conf. Hybrid Syst.: Comput. Control*, ACM, Philadelphia, USA, 2013, pp. 1704–1717.
[16] A. Jones, Z. Kong, and C. Belta, "Anomaly detection in cyber-physical systems: A formal methods approach," in *Proc. IEEE 53rd Annu. Conf. Decision and Control (CDC)* IEEE, Los Angeles, CA, 2014, pp. 848–853.
[17] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, 2004.
[18] Z. Kong, A. Jones, A. M. Ayala, E. A. Gol, and C. Belta, "Temporal logic inference for classification and prediction from data," in *Proc. 17th Int. Conf. Hybrid Syst.: Comput. and Control (HSCC)*, Berlin, Germany, 2014, pp. 273–282.
[19] K. Kowalska and L. Peel, "Maritime anomaly detection using gaussian process active learning," in *Proc. 15th Int. Conf. Information Fusion (FUSION)*, IEEE, Singapore, 2012, pp. 1164–1171.
[20] J. Lygeros, K. H. Johansson, S. Sastry, and M. Egerstedt, "On the existence of executions of hybrid automata," in *Proc. 38th IEEE Conf. Decision Control*, Phoenix, AZ, 1999, vol. 3, pp. 2249–2254.
[21] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," *Formal Techniques, Modelling Anal. Timed Fault-Tolerant Syst.*, vol. 3253, pp. 71–76, 2004.
[22] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancić, A. Gupta, and G. J. Pappas, "Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems," in *Proc. 13th ACM Int. Conf. Hybrid Syst.: Comput. Control*, ACM, Stockholm, Sweden, 2010, pp. 211–220.
[23] F. Pasqualetti, F. Dorfler, and F. Bullo, "Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design," in *Proc. 50th IEEE CDC-ECC*, IEEE, Orlando, Florida, 2011, pp. 2195–2201.
[24] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proc. 18th Int. Conf. Hybrid Syst.: Comput. and Control (HSCC)*, ACM, Seattle, WA, 2015, pp. 239–248.
[25] S. Sankaranarayanan and G. Fainekos, "Falsification of temporal properties of hybrid systems using the cross-entropy method," in *Proc. 15th ACM Int. Conf. Hybrid Syst.: Comput. Control*, ACM, Beijing, China, 2012, pp. 125–134.
[26] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines—An application in machine fault detection and classification," *Comput. Ind. Eng.*, vol. 48, no. 2, pp. 395–408, 2005.
[27] Y. Shoukry, J. Araujo, P. Tabuada, M. Srivastava, and K. H. Johansson, "Minimax control for cyber-physical systems under network packet scheduling attacks," in *Proc. Second ACM Int. Conf. High Confidence Netw. Syst.*, ACM, Berlin, Germany, 2013, pp. 93–100.
[28] A. P. Sistla, M. Žefran, and Y. Feng, "Monitorability of stochastic dynamical systems," in *Computer Aided Verification*. Berlin, Germany: Springer, 2011, pp. 720–736.
[29] J. Slay and M. Miller, *Lessons Learned From the Maroochy Water Breach*. Berlin, Germany: Springer, 2007.
[30] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, "Attack models and scenarios for networked control systems," in *Proc. First Int. Conf. High Confidence Netw. Syst.*, ACM, Berlin, Germany, 2012, pp. 55–64.
[31] H. Trevor, T. Robert, and J. J. H. Friedman, *The Elements of Statistical Learning*. Berlin, Germany: Springer, 2001.
[32] B. Ustun, S. Tracà, and C. Rudin, Supersparse Linear Integer Models for Interpretable Classification. *arXiv preprint arXiv:1306.6677*, 2013.
[33] H. Yang, B. Hoxha, and G. Fainekos, "Querying parametric temporal logic properties on embedded systems," in *Testing Software and System*. Berlin, Germany: Springer, 2012, pp. 136–151.

Zhaodan Kong (M'11) is an Assistant Professor in the Department of Mechanical and Aerospace Engineering at the University of California, Davis. He received the B.S. and M.S. degrees in astronautics and mechanics from Harbin Institute of Technology, Harbin, China, in 2004 and 2006, respectively, and the Ph.D. degree in aerospace engineering with a minor in cognitive science from the University of Minnesota, Twin Cities in 2012. During 2012 and 2014 he was a postdoctoral research fellow at Boston University. His current research interests include cyber-physical systems, bio-inspired robotics, formal methods, and human-machine systems.

Austin Jones (M'13) was born in Kentucky, USA in 1988. He received a B.S. and M.S. in Systems Science at Washington University in 2010 and a Ph.D. from Boston University in Systems Engineering in 2015. Austin joined Numerica Corporation in 2010–2011 and was a post-doctoral fellow in the schools of Mechanical Engineering and Electrical and Computer Engineering at Georgia Institute of Technology in 2015. Austin's research interests include robotics, formal methods, and stochastic systems.

Calin Belta (SM'11) is a Professor in the Department of Mechanical Engineering, Department of Electrical and Computer Engineering, and the Division of Systems Engineering at Boston University, where he is also affiliated with the Center for Information and Systems Engineering (CISE) and the Bioinformatics Program. His research focuses on dynamics and control theory, with particular emphasis on hybrid and cyber-physical systems, formal synthesis and verification, and applications in robotics and systems biology. Calin Belta is a Senior Member of the IEEE and an Associate Editor for the SIAM Journal on Control and Optimization (SICON) and the IEEE TRANSACTIONS ON AUTOMATIC CONTROL. He received the Air Force Office of Scientific Research Young Investigator Award and the National Science Foundation CAREER Award.