Temporal Logic Inference for Classification and Prediction from Data

Zhaodan Kong
Boston University
110 Cummington Street
Boston, MA
zhaodan@bu.edu

Austin Jones
Boston University
15 Saint Mary's Street
Brookline, MA
austinmj@bu.edu

Ana Medina Ayala
Boston University
110 Cummington Street
Boston, MA
duvinci@bu.edu

Ebru Aydin Gol
Boston University
15 Saint Mary's Street
Brookline, MA
ebru@bu.edu

Calin Belta
Boston University
110 Cummington Street
Boston, MA
cbelta@bu.edu

ABSTRACT

This paper presents an inference algorithm that can discover temporal logic properties of a system from data. Our algorithm operates on finite time system trajectories that are labeled according to whether or not they demonstrate some desirable system properties (e.g. "the car successfully stops before hitting an obstruction"). A temporal logic formula that can discriminate between the desirable behaviors and the undesirable ones is constructed. The formulae also indicate possible causes for each set of behaviors (e.g. "If the speed of the car is greater than 15 m/s within 0.5s of brake application, the obstruction will be struck") which can be used to tune designs or to perform on-line monitoring to ensure the desired behavior. We introduce reactive parameter signal temporal logic (rPSTL), a fragment of parameter signal temporal logic (PSTL) that is expressive enough to capture causal, spatial, and temporal relationships in data. We define a partial order over the set of rPSTL formulae that is based on language inclusion. This order enables a directed search over this set, i.e. given a candidate rPSTL formula that does not adequately match the observed data, we can automatically construct a formula that will fit the data at least as well. Two case studies, one involving a cattle herding scenario and one involving a stochastic hybrid gene circuit model, are presented to illustrate our approach.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Knowledge Acquisition; D.2.1 [**Software Engineering**]: Requirements/Specifications; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages; D.4.7 [**Organization and Design**]: Real-Time Systems and Embedded Systems

HSCC'14, April 15–17, 2014, Berlin, Germany.

http://dx.doi.org/10.1145/2562059.2562146.

General Terms

Algorithms, Theory

Keywords

Parametric Signal Temporal Logic; Logic inference; Directed Acyclic Graph; Gene network

1. INTRODUCTION

Reverse engineering has always been a cornerstone of physical and biological science. Given a set of input-output pairs, one can interpret and predict the behavior of the underlying system by inferring properties that are compatible with this data. Reverse engineering can largely be divided into three areas: system identification [16], machine learning [21], and inductive logic programming [15]. In general, properties inferred from reverse engineering can either describe the dynamics of a system or capture some high-level specification that the system satisfies. Inferring dynamics can be a challenging task if very little is known about the system. On the other hand, inferred specifications might be too "coarse-grained" to be useful for problems of interest. Temporal logics [11] bridge these two extremes by incorporating quantitative temporal and spatial constraints when describing dynamic behaviors. For instance, we can use temporal logics to express invariance properties such as "If x is greater than x_r, then within T_1 seconds, it will drop below x_r and remain below x_r for at least T_2 seconds".

In this paper, we address the problem of inferring a temporal logic formula that can be used to distinguish between desirable system behaviors, e.g. an airplane lands in some goal configuration on the tarmac, and undesirable behaviors, e.g. the airplane's descent is deemed unsafe. Moreover, in our approach, the inferred formulae can be used as predictive templates for either set of behaviors. This in turn can be used for on-line system monitoring, e.g. aborting a landing if the descent pattern is consistent with unsafe behavior. Since our procedure is automatic and unsupervised beyond the initial labeling of the signals, it is possible that it can discover properties of the system that were previously unknown to designers, e.g. changing the direction of banking too quickly will drive the airplane to an unsafe configuration.

Most of the recent research on temporal logic inference has focused on the estimation of parameters associated with a given temporal logic structure [1, 22, 12, 2]. In the referred papers, the structure of the formula reflects the domain knowledge of the designer as well as the properties of interest of a given system. However, it is possible that the selected formula may not reflect achievable behaviors or may overlook fundamental features. Furthermore, an important feature of reverse engineering that is absent from the current paradigm is the possibility of deriving new knowledge directly from data, since it requires the user to be very specific about the system properties that are to be inferred. Thus, a natural next step is to infer from data the formula structure in addition to its parameters. As a result, in this work, we guide the search via the robustness degree [8, 6], a signed metric on the signal space which quantifies to what degree a signal satisfies or violates a given formula.

In this paper, we solve the structural inference problem and the parameter estimation problem simultaneously. The structural inference problem is generally hard and even ill-posed [9]. We reduce the difficulty of structural learning by imposing a partial order on the set of *reactive parametric signal temporal logic* (rPSTL) formulae. The defined partial order allows us to search for a formula template in an efficient, orderly fashion while the robustness degree allows us to formulate the inference problem as a well-defined optimization problem.

The paper is outlined as follows. Section 2 reviews signal and parametric signal temporal logic. Section 3 uses a herding example to motivate the inference problem. A new logic called reactive parametric signal temporal logic is defined and the formal problem statement is given in this section. Section 4 presents some properties of rPSTL. The details of our inference algorithm are described in Section 5. Two case studies are presented in Section 6. Finally, Section 7 concludes the paper.

2. SIGNAL AND PARAMETRIC SIGNAL TEMPORAL LOGIC

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.

Signal temporal logic (STL) [17] is a temporal logic defined over signals. The *syntax* of STL is inductively defined as

$$\phi := \mu|\neg\phi|\phi_1 \vee \phi_2|\phi_1 \wedge \phi_2|\phi_1 \mathcal{U}_{[a,b)}\phi_2, \tag{1}$$

where $[a, b)$ is a time interval, μ is a *numerical predicate* in the form of an inequality $g_\mu(s(t)) \sim c_\mu$ such that $g_\mu \in \mathcal{F}(\mathbb{R}^n, \mathbb{R})$, $\sim \in \{<, \geq\}$, and c_μ is a constant.

The *semantics* of STL is defined recursively as

$$\begin{array}{rcl} s[t] \models \mu & \text{iff} & g_\mu(s(t)) \sim c_\mu \\ s[t] \models \neg\phi & \text{iff} & s[t] \not\models \phi \\ s[t] \models \phi_1 \wedge \phi_2 & \text{iff} & s[t] \models \phi_1 \text{ and } s[t] \models \phi_2 \\ s[t] \models \phi_1 \vee \phi_2 & \text{iff} & s[t] \models \phi_1 \text{ or } s[t] \models \phi_2 \\ s[t] \models \phi_1 \mathcal{U}_{[a,b)}\phi_2 & \text{iff} & \exists t' \in [t+a, t+b) \\ & & \text{ s. t. } s[t'] \models \phi_2, s[t''] \models \phi_1 \\ & & \forall t'' \in [t+a, t'). \end{array} \tag{2}$$

We also use the constructed temporal operators $\Diamond_{[a,b)}\phi = \top\, \mathcal{U}_{[a,b)}\phi$ (read "eventually ϕ"), where $\top$ is the symbol for Boolean constant True, and $\Box_{[a,b)}\phi = \neg\Diamond_{[a,b)}\neg\phi$ (read "always ϕ"). A signal s satisfies an STL formula ϕ if $s[0] \models \phi$.

The *language* of an STL formula ϕ, $L(\phi)$, is the set of all signals that satisfy ϕ, namely $L(\phi) = \{s \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)|s \models \phi\}$. Given formulae ϕ_1 and ϕ_2, we say that ϕ_1 and ϕ_2 are *semantically equivalent* , i.e., $\phi_1 \equiv \phi_2$, if $L(\phi_1) = L(\phi_2)$.

Parametric signal temporal logic (PSTL) [1] is an extension of STL where c_μ or the endpoints of the time intervals $[a, b)$ are parameters instead of constants. We denote them as *scale* parameters $\pi = [\pi_1, ..., \pi_{n_\pi}]$, and *time* parameters $\tau = [\tau_1, ..., \tau_{n_\tau}]$, respectively. They range over their respective hyper-rectangular domains $\Pi \subset \mathbb{R}^{n_\pi}$ and $\mathrm{T} \subset \mathbb{R}^{n_\tau}$. A full parameterization is denoted by $\theta = [\pi, \tau]$ with $\theta \in \Theta = \Pi \times \mathrm{T}$. The syntax and semantics of PSTL are the same as those for STL. To avoid confusion, we will use ϕ to refer to an STL formula and φ to refer to a PSTL formula. A *valuation* v is a mapping that assigns real values to the parameters appearing in a PSTL formula. Each valuation v of a PSTL formula φ induces an STL formula ϕ_v where each parameter in φ is replaced with its image in v. For example, given $\varphi = (x_s \geq \pi_1)\mathcal{U}_{[0,\tau_1)}(y_s \geq \pi_2)$ and $v([\pi_1, \pi_2, \tau_1]) = [0, 4, 5]$, we have $\phi_v = (x_s \geq 0)\mathcal{U}_{[0,5]}(y_s \geq 4)$.

The robustness degree of a signal s with respect to an STL formula ϕ at time t is given as $r(s, \phi, t)$, where r can be calculated recursively via the *quantitative semantics* [8, 6]

$$\begin{array}{rl} r(s, \mu_\geq, t) & = g_\mu(s(t)) - c_\mu \\ r(s, \mu_<, t) & = c_\mu - g_\mu(s(t)) \\ r(s, \neg\phi, t) & = -r(s, \phi, t) \\ r(s, \phi_1 \wedge \phi_2, t) & = \min(r(s, \phi_1, t), r(s, \phi_2, t)) \\ r(s, \phi_1 \vee \phi_2, t) & = \max(r(s, \phi_1, t), r(s, \phi_2, t)) \\ r(s, \phi_1 \mathcal{U}_{[a,b)}\phi_2, t) & = \sup\limits_{t' \in [t+a, t+b)} (\min(r(s, \phi_2, t'), \\ & \quad \inf\limits_{t'' \in [t, t')} r(s, \phi_1, t''))) \end{array}$$

where $\mu_\geq$ is a predicate of the form $g_\mu(s(t)) \geq c_\mu$ and $\mu_<$ is a predicate of the form $g_\mu(s(t)) < c_\mu$.

We use $r(s, \phi)$ to denote $r(s, \phi, 0)$. A *signed distance* from a point $x \in X := \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$ to a set $S \subseteq X$ is defined as

$$\mathrm{Dist}_\rho(x, S) := \begin{cases} -\inf\{\rho(x, y)|y \in cl(S)\} & \text{if } x \notin S \\ \inf\{\rho(x, y)|y \in X \setminus S\} & \text{if } x \in S \end{cases} \tag{3}$$

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

$$\rho(s, s') = \sup_{t \in T}\{d(s(t), s'(t))\}, \tag{4}$$

and d corresponds to the metric defined on the domain $\mathbb{R}^n$ of signal s. It has been shown in [8] that $r(s, \phi)$ is an under-approximation of $\mathrm{Dist}_\rho(s, L(\phi))$.

3. PROBLEM STATEMENT

3.1 Motivating Example

We first give a motivating scenario that will serve as a running example throughout the rest of this paper. Consider the example shown in Figure 1. A rancher keeps track of the location of his cattle via GPS devices embedded in ear tags. The rancher herds the cattle to new grazing grounds. While most of the cattle are successfully herded to the rancher's grounds, some wander into someone else's land, indicating that there are likely one or more breaches in the fencing separating the two properties. However, searching for a breach

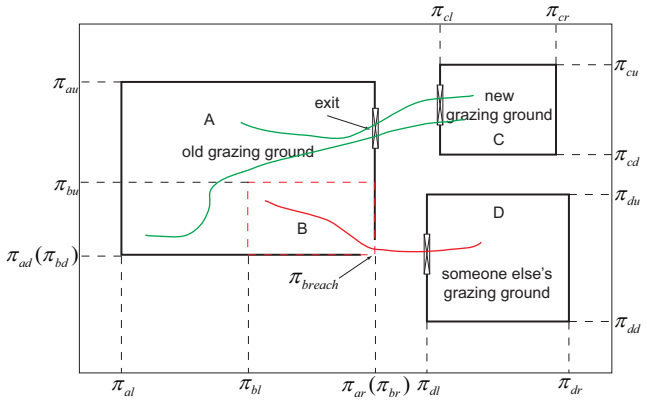

Figure 1: A herding example. The desired behaviors are shown in green, while the undesired ones are shown in red.

in the fencing would be very difficult to do if the property is large.

Consider the following PSTL formula that describes undesirable behaviors

$$\varphi_{und} = \Diamond_{[0,\tau_1)}(\Box_{[0,\tau_2)}(d(s,\pi_{breach}) < d_0) \Rightarrow \Diamond_{[\tau_3,\tau_4)}(\text{Dist}_\rho(s,D) \geq 0)), \tag{5}$$

where π_{breach} is a parameter describing the breach location (the corner of region B), D is someone else's grazing ground, d_0 is a threshold, and d and $\text{Dist}_\rho(.,.)$ are defined in Section 2. The formula reads as "If there is a time t in $[0,\tau_1)$ such that if the cow remains within d_0 of π_{breach} for the next τ_2 seconds, then in at least τ_3 and at most τ_4 seconds, the cow is guaranteed to enter D." The structure of the formula gives quite a bit of insight into the cows' behavior. The sub-formula $\Diamond_{[\tau_3,\tau_4)}(\text{Dist}_\rho(s,D) \geq 0)$ specifies the undesirable behavior (classifies behaviors), and the sub-formula $\Box_{[0,\tau_2)}(d(s,\pi_{breach}) < d_0)$ provides a pre-condition of the undesirable behavior (predicts behaviors). Finding a parameterization of this formula that fits the data has practical value, as searching for the parameter π_{breach} will yield the location of the breach, allowing the farmhands to mend it.

3.2 Reactive Parametric Signal Temporal Logic

We define *reactive parametric signal temporal logic* (rPSTL), a fragment of PSTL that is expressive enough to capture causal relationships that are crucial to a wide range of applications.

The set of predicates used in rPSTL is restricted to the set of linear predicates of the form $(y_s \sim \pi)$ where $\sim \in \{<, \geq\}$, and π is a scale parameter.

The *syntax* of rPSTL is given as

$$\varphi ::= \Diamond_{[\tau_1,\tau_2)}(\varphi_c \Rightarrow \varphi_e) \tag{6a}$$

$$\varphi_c ::= \Diamond_{[\tau_1,\tau_2)}\ell | \Box_{[\tau_1,\tau_2)}\ell | \varphi_c \wedge \varphi_c | \varphi_c \vee \varphi_c \tag{6b}$$

$$\varphi_e ::= \Diamond_{[\tau_1,\tau_2)}m | \Box_{[\tau_1,\tau_2)}m | \varphi_e \wedge \varphi_e, | \varphi_e \vee \varphi_e \tag{6c}$$

where ℓ and m are linear predicates, and τ_1 and τ_2 are time parameters. We refer to φ_c as the *cause formula* and φ_e as the *effect formula*.

The semantics of rPSTL is the same as PSTL. Since rPSTL is a fragment of PSTL, any valuation v of an rPSTL formula induces an STL formula. We call the fragment of all such STL formulae reactive STL (rSTL). Let c_i be real values. The *quantitative semantics* of a signal with respect to an rSTL formula are given by

$$\begin{aligned} r(s,(y_s \geq c_1),t) &= y_s(t) - c_1 \\ r(s,(y_s < c_1),t) &= c_1 - y_s(t) \\ r(s,\Box_{[c_1,c_2)}\phi,t) &= \min_{t' \in [t+c_1,t+c_2)} r(s,\phi,t') \\ r(s,\Diamond_{[c_1,c_2)}\phi,t) &= \max_{t' \in [t+c_1,t+c_2)} r(s,\phi,t') \end{aligned} \tag{7}$$

Formula (6a) can be read as "If there is an instance t in the time interval $[\tau_1,\tau_2)$ such that an event described by formula φ_c occurs, then an event described by formula φ_e will be triggered." For this reason, in an rPSTL (rSTL) formula, we call sub-formula φ_c (ϕ_c) the cause formula and sub-formula φ_e (ϕ_e) the effect formula. We call this fragment reactive PSTL because we say that the system reacts to the cause φ_c by producing an effect φ_e. The causal structure fits the practical needs of automatically identifying causes of certain events, such as unauthorized network intrusion [3]. In this case, the learned cause formula can serve as an on-line monitor for intrusion detection.

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

- Bounded-time invariance, e.g. $\Diamond_{[0,\tau_1)}(\Box_{[0,\tau_2)}(y_s < \pi_1) \Rightarrow \Box_{[\tau_3,\tau_4)}(y_s < \pi_2))$ ("If there exists a time $t \in [0,\tau_1)$ such that if y_s is less than π_1 for the next τ_2 seconds, then it will be less than π_2 everywhere in $[t+\tau_3, t+\tau_4)$.")
- Reachability to multiple regions in the state space, e.g. $\Diamond_{[0,\tau_1)}(\Diamond_{[0,\tau_2)}(y_s \geq \pi_1) \Rightarrow \Diamond_{[\tau_3,\tau_4)}(y_s \geq \pi_2) \vee \Diamond_{[\tau_3,\tau_4)}(y_s < \pi_3))$ ("If there exists a time $t \in [0,\tau_1)$ such that if y_s is greater than π_1 within the next τ_2 seconds, then eventually y_s is either less than π_3 or greater than π_2 from $t+\tau_3$ seconds to $t+\tau_4$ seconds.")

We can approximate (5) with the rPSTL formula

$$\begin{aligned} \varphi^*_{und} = \quad & \Diamond_{[0,\tau_1)}(\varphi^*_{und,c} \Rightarrow \varphi^*_{und,e}) \\ \varphi^*_{und,c} = \quad & \Box_{[0,\tau_2)}(y_s \geq \pi_{bd}) \wedge \Box_{[0,\tau_2)}(y_s < \pi_{bu}) \wedge \\ & \Box_{[0,\tau_2)}(x_s \geq \pi_{bl}) \wedge \Box_{[0,\tau_2)}(x_s < \pi_{br}) \\ \varphi^*_{und,e} = \quad & \Box_{[\tau_3,\tau_4)}(y_s \geq \pi_{dd}) \wedge \Box_{[\tau_3,\tau_4)}(y_s < \pi_{du}) \wedge \\ & \Box_{[\tau_3,\tau_4)}(x_s \geq \pi_{dl}) \wedge \Box_{[\tau_3,\tau_4)}(x_s < \pi_{dr}) \end{aligned} \tag{8}$$

The scale parameters are shown in Figure 1. The formula φ^*_{und} can be interpreted as "If a cow is in B for the next τ_2 seconds ($\varphi^*_{und,c}$), then it is guaranteed to be in D within τ_3 seconds and remain in D for $\tau_4 - \tau_3$ seconds($\varphi^*_{und,e}$)."

Remark 1. (Limitations of rPSTL) There are some temporal properties that cannot be described directly in rPSTL, namely,

- Concurrent eventuality, e.g. $\varphi_{c,e} = \Diamond_{[0,\tau_1)}((y_s < \pi_1) \wedge (x_s \geq \pi_2))$. ("Within τ_1 seconds, y_s is less than π_1 and x_s is greater than π_2 at the same time.")
- Nested "always eventually", e.g. $\varphi_{c,e} = \Box_{[0,\tau_1)}\Diamond_{[\tau_2,\tau_3)}(y_s < \pi_1)$. ("At any time t in the next τ_1 seconds, y_s will be less than π_1 at some point in the interval $[t+\tau_2, t+\tau_3)$.")

The lack of concurrent eventuality means that we cannot directly specify that a trajectory will eventually reach

some intersection of half-spaces in the state-space, though we can approximate such properties by specifying $\varphi_{c,e} = \Diamond_{[0,\tau_1)}(y_s < \pi_1) \wedge \Diamond_{[0,\tau_1)}(x_s > \pi_2)$.

The lack of nested "always eventually" limits the periodic properties that may be expressed, but we can approximate such properties by specifying $\varphi_{c,e} = \Diamond_{[\tau_2,\tau_3)}(y_s < \pi_1) \wedge \ldots \wedge \Diamond_{[\tau_2+n\epsilon,\tau_3+n\epsilon)}(y_s < \pi_1)$, that is by selecting n points in the interval $[0, \tau_1)$ ϵ apart and specifying that the property $\Diamond_{[\tau_2,\tau_3)}(y_s < \pi_1)$ is true at all points.

3.3 Problem Description

In this paper, we consider the following problem:

Problem 1. Given a set of labeled signals $\{(s_i, p_i)\}_{i=1}^N$, where signal s_i has a finite duration and $p_i = 1$ if s_i demonstrates a desired behavior and $p_i = 0$ if s_i does not, find an rSTL formula ϕ_{des} (or ϕ_{und}) such that

- $s_i \models \phi_{des}$ iff $p_i = 1$ (or $s_i \models \phi_{und}$ iff $p_i = 0$) $\forall i = 1, \ldots, N$ (classification).
- ϕ_{des} (or ϕ_{und}) can be used to determine p_i from a prefix of s_i $\forall i = 1, \ldots, N$ (prediction).

The nature of the problem of interest determines whether ϕ_{des} or ϕ_{und} is needed. For instance, when trying to find possible causes of aircraft crashes, ϕ_{und} is more relevant. In the following, for brevity, we only define the problems involved with ϕ_{des}. Problems involved with ϕ_{und} can be defined similarly.

We approximate the solution to Problem 1 by finding the cause and effect formulae (see (6b)) and (6c)) separately. First, we solve the classification problem by searching for an effect formula $\phi_{des,e}$ that can adequately classify the s_i based on the last $\tilde{t}$ seconds of the observed signals. That is, we assume that the observed desirable or undesirable behaviors occur in the last $\tilde{t}$ seconds of the observed signal s_i. (Please refer to Remark 3 for guidelines for selecting $\tilde{t}$.) Making this assumption yields a significant computational speedup: inferring time bounds for a single formulae over the timescale T requires more computation than inferring time bounds for two formulae over timescales $\tilde{t}$ and $T - \tilde{t}$, respectively.

The classification procedure can be cast as the following optimization problem.

Problem 2. (Classification) Let σ_i be the signal that results from truncating s_i to its final $\tilde{t}$ seconds. Find an effect formula $\phi_{des,e}$ with syntax given by (6c) such that the rPSTL formula $\varphi_{des,e}$ and valuation $v_{des,e}$ minimize

$$J_e(\varphi, v) = \frac{1}{N}\sum_{i=1}^{N} l(p_i, r(s_i, \phi_v)) + \lambda||\phi_v||, \qquad (9)$$

where r is the robustness degree defined in Section 2, ϕ_v is derived from φ with valuation v, l is a loss function, λ is a weighting parameter, and $||\phi_v||$ is the length of ϕ_v (number of linear predicates that appear in ϕ_v).

A natural loss function l is the total number of signals that ϕ_v mis-classifies. Unfortunately, such a discrete measure of success is not helpful for iterative optimization procedures. Instead, we propose to continuize l by using the robustness degree as an intermediary *fitness function,* a measure of how well a given formula fits observed data. We penalize formula length in our approach because if $\phi_{des,e}$ grows arbitrarily long, it becomes as complex to represent as the data itself, which would render the inference process redundant.

After solving Problem 2, we need to find a cause formula $\phi_{des,c}$ that is consistent with the mined rPSTL template $\varphi_{des,e}$ and the full signals s_i. We do this by performing the following optimization.

Problem 3. (Prediction) Find a formula ϕ_{des} that minimizes the cost

$$J_c(\varphi, v) = \frac{1}{N}\sum_{i=1}^{N} l(p_i, r(s_i, \phi_v)) + \lambda||\phi_v||, \qquad (10)$$

where $\varphi = \Diamond_{[0,\tau_1)}(\varphi_c \Rightarrow \varphi_{des,e})$ and $\varphi_{des,e}$ is the solution to Problem 2.

The minimization of l in Problem 2 maximizes the classification quality. Solving Problem 3 after Problem 2 yields a cause formula $\phi_{des,c}$ such that if a system produces a signal prefix that satisfies $\phi_{des,c}$, then p_i is guaranteed to be 1.

4. PROPERTIES OF RPSTL

In this section, we present some properties of rSTL and rPSTL that are essential for the design of our inference algorithm. We define a partial order over $rPSTL$, the set of all rPSTL formulae. The formulae in $rPSTL$ 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. Finally, for any parameterization, the robustness degree of a signal with respect to a formula $\phi_{1,v}$ is greater than with respect to $\phi_{2,v}$ if φ_1 has a higher order than φ_2. This enables us to find an rSTL formula against which a signal is more robust by searching for a parameterization of an rPSTL formula that is further down the DAG.

4.1 Partial Orders Over rSTL and rPSTL

We define two relations $\preceq_S$ and $\preceq_P$ for rSTL formulae and rPSTL formulae, respectively.

Definition 1.

1. For two rSTL 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 rPSTL formulae φ_1 and φ_2, $\varphi_1 \preceq_P \varphi_2$ iff $\forall v$, $\phi_{1,v} \preceq_S \phi_{2,v}$, where the domain of v is $\Theta(\varphi_1) \cup \Theta(\varphi_2)$, the union of parameters appearing in φ_1 and φ_2.

Based on these definitions and the semantics of $rSTL$ and $rPSTL$, we have

PROPOSITION 1. *Both $\preceq_S$ and $\preceq_P$ are partial orders.*

PROOF. (Sketch) A partial order $\preceq$ is a binary relation that is reflexive, transitive and antisymmetric. The equivalence with language inclusion of $\preceq_S$ is used to prove that it is a partial order. The relationship of $\preceq_P$ with $\preceq_S$ is used to show that $\preceq_P$ is a partial order. For example, the antisymmetry of $\preceq_P$ can be proved as follows. If $\varphi_1 \preceq_P \varphi_2$ and $\varphi_2 \preceq_P \varphi_1$, we have

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

□

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. $\square_{[\tau_1,\tau_2)}\ell \preceq_P \Diamond_{[\tau_1,\tau_2)}\ell$, *where* ℓ *is a linear predicate;*
3. *For two rPSTL formulae,* $\varphi_1 := \Diamond_{[\tau_1,\tau_2)}(\varphi_{c1} \Rightarrow \varphi_{e1})$ *and* $\varphi_2 := \Diamond_{[\tau_1,\tau_2)}(\varphi_{c2} \Rightarrow \varphi_{e2})$, $\varphi_1 \preceq_P \varphi_2$ *iff* $\varphi_{c2} \preceq_P \varphi_{c1}$ *and* $\varphi_{e1} \preceq_P \varphi_{e2}$.

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". The third property is easy to verify once we consider the semantic equivalence of $\varphi_c \Rightarrow \varphi_e$ and $\neg\varphi_c \vee \varphi_e$. An rPSTL formula can be made more inclusive by either making the effect formula more inclusive or the cause formula more exclusive.

4.2 DAG and Robustness Degree

The structure of rPSTL and the definition of the partial order $\preceq_P$ enable the following theorem.

THEOREM 1. *The formulae in rPSTL 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. (Sketch) The proof of this theorem requires the intermediate results that the family of formulae Φ with syntax (6b) and (6c) (e.g. cause and effect subformulae) form a lattice when ordered according to $\preceq_P$. A partially ordered set $< X, \preceq >$ forms a *lattice* if any two elements $x_1, x_2 \in X$ have a join and a meet [4]. In our case, we need to prove that for all $\varphi_1, \varphi_2 \in \Phi$, their join $\varphi_1 \sqcap \varphi_2$ and meet $\varphi_1 \sqcup \varphi_2$ exist and are unique. This can be done by first treating the subformulae $\square_I p$ and $\Diamond_I p$ where p is a linear predicate and I is a time interval $I := [\tau_1, \tau_2)$ as different Boolean predicates. Then the existence and uniqueness of $\varphi_1 \sqcap \varphi_2$ ($\varphi_1 \sqcup \varphi_2$) can be proved by the existence and uniqueness of $\varphi_1 \wedge \varphi_2$ ($\varphi_1 \vee \varphi_2$)[11] by putting formulae in Disjunctive (Conjunctive) Normal Form . Finally, since $< rPSTL, \preceq_P >$ is a lattice, it has an equivalent representation as an infinite DAG [4]. □

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

Next, we establish a relationship between the robustness degrees of a signal s with respect to rSTL (rPSTL) formulae ϕ (φ) and the partial order $\preceq_S$ ($\preceq_P$).

THEOREM 2. *The following statements are equivalent:*

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

PROOF. $2 \Rightarrow 1$ can be easily proved by using contradiction. As for $1 \Rightarrow 2$, since $L(\phi_1) \subset L(\phi_2)$, for any $s \in \mathcal{F}(\mathbb{R}^+, \mathbb{R}^n)$, we need to first enumerate the following three cases and prove that $1 \Rightarrow 2$ is true for each: (a) $s \in L(\phi_1)$; (b) $s \in L(\phi_1) \cap L(\neg\phi_2)$; and (c) $s \in L(\neg\phi_1) \cap L(\neg\phi_2)$. This can be done by using the relationship between $r(s, \phi)$ and $\text{Dist}_\rho(s, L(\phi))$ as described by (3). □

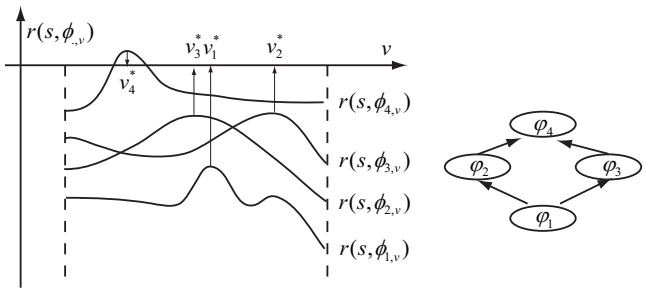

Figure 2: Illustration of the relationship between rPSTL formulae and robustness degree.

COROLLARY 1. *The following statements are equivalent:*

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

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

Theorem 1 and Corollary 1 have important implications for solving Problem 1. The formula inferred by our procedure should be a close representation of the properties that differentiate between desired and undesired behavior. Restricting the inferred formula (shrinking its language) by a small amount should result in a formula that cannot discriminate between the two cases. Thus, the mined formula should in principle be the lowest ordered satisfying formula. The DAG representation of $rPSTL$ can naturally be used to find such a "barely" satisfying formula. The search starts from the most exclusive formula and follows directed edges until a satisfying formula is found. This is shown in Figure 2. 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 robustness degrees). 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 robustness degree.

The interaction between the graph search and parameter estimation is further illustrated in Figure 3. The top left and right plots show the x and y coordinates, respectively, of a single cow's trajectory. The center left (right) figure shows the robustness degree with respect to $\varphi_1 := \Diamond_{[0,\tau)}(x > 100)$ ($\varphi_2 := \Diamond_{[0,40)}(y < \pi)$) for various values of τ (π). Note that by selecting the parameter $\tau(\pi)$ for each φ_i, we can maximize or minimize the robustness degree of the signal with respect to the induced formula $\phi_{i,v}$. 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 rSTL 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. This is useful when we consider large groups of traces, as it is more likely that for two signals s_1, s_2 where $p_1 = 1, p_2 = 0$, we can find a formula $\phi_{j,v}, j \in \{3, 4\}$ such that $r(s_1, \varphi_{j,v}) > 0$ and $r(s_2, \varphi_{j,v}) < 0$ for $i = 1, 2$ than to

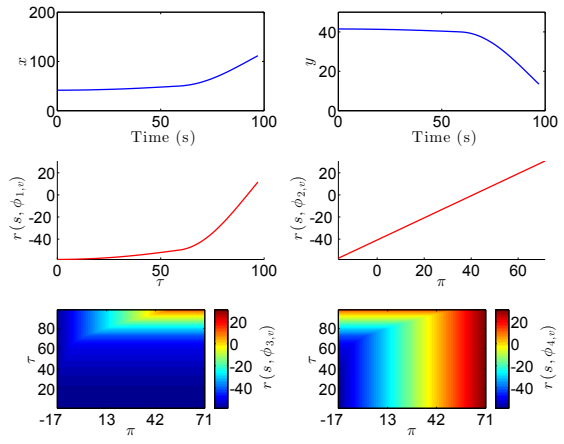

Figure 3: Simple example of formula search using cattle herding data.

be able to find a formula $\phi_{1,v}$ or $\phi_{2,v}$ that achieves the same classification.

5. SOLUTION

In this work, we infer an rPSTL formula by inferring the effect and cause formulae separately. Inferring the formulae separately still constitutes a search on finite subgraphs of the infinite DAG. When searching for the effect formula ϕ_e, we search over only the nodes of form $\Diamond_{[0,\tau_1)}(\top \Rightarrow \varphi_e)$. When searching for the entire formula in the prediction phase, we only search among the nodes of the form $\Diamond_{[0,\tau_1)}(\varphi_c \Rightarrow \varphi_{des,e})$, where $\varphi_{des,e}$ is the effect formula found in the classification phase. The framework for solving Problem 2 is detailed in Alg. 1. The prediction algorithm to solve Problem 3 is similar to Alg. 1.

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 4 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 $O \in \{\Box, \Diamond\}$, $\sim \in \{\geq, <\}$ and $x \in V$. Edges are constructed from φ_i to φ_j in the initial graph $\mathcal{G}_1$ iff $\varphi_j \preceq_P \varphi_i$. For example, in the cow herding example if we only consider the (x, y) position of the cow, then the initial graph is shown in Figure 4.

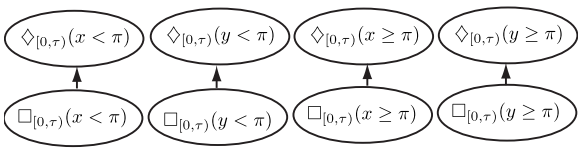

Figure 4: The initial graph $\mathcal{G}_1$ constructed from x, y coordinates.

ListInitialization($\mathcal{G}_1$) (line 5) 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

Algorithm 1: Classification Algorithm

Input:
A set of labeled signals $\mathcal{S}_e := (s_i, p_i), i = 1, ..., N$;
A variable set V;
A misclassification rate threshold δ;
A formula length bound L_{max}
Output:
A rPSTL formula φ_e along with the corresponding valuation v_e and the misclassification rate q_e.

1 $\mathcal{G}_0 \leftarrow \emptyset$;
2 **for** $i = 1$ *to* L_{max} **do**
3 **if** $i = 1$ **then**
4 $\mathcal{G}_1 \leftarrow$ DAGInitialization(V);
5 $List \leftarrow$ ListInitialization($\mathcal{G}_1$);
6 **else**
7 $\mathcal{G}_i \leftarrow$ PruningAndGrowing($\mathcal{G}_{i-1}$);
8 $List \leftarrow$ Ranking($\mathcal{G}_i \setminus \mathcal{G}_{i-1}$);
9 **end**
10 **while** $List \neq \emptyset$ **do**
11 $\varphi \leftarrow$ PopOutFirstFormula($List$);
12 $v_{ini} \leftarrow$ ParameterInitialization($\varphi, \mathcal{G}_{i-1}$);
13 $(v, c, q) \leftarrow$ ParameterEstimation($\mathcal{S}_e, \varphi, v_{ini}$);
14 **if** $q \leq \delta$ **then**
15 **return** (φ, v, q).
16 **end**
17 $\mathcal{G}_i \leftarrow$ Maintenance(φ, v, c, q);
18 **end**
19 **end**
20 $k^* \leftarrow$ MinimumCostNode($\mathcal{G}_{L_{max}}$);
21 **return** $(\varphi_{k^*}, v_{k^*}, q_{k^*})$.

behaviors, the rank is generated randomly. After the graph is constructed, we find the optimal parameters for each of the nodes.

Parameter Estimation.

PopOutFirstFormula($List$) (line 11) pops out the lowest ranked formula from $List$. ParameterInitialization($\varphi, \mathcal{G}_{i-1}$) (line 12) randomly generates an initial valuation for φ if $\mathcal{G}_{i-1} = \emptyset$. Otherwise, it initializes the valuations based on those of its parents. ParameterEstimation($\mathcal{S}_e, \varphi, v_{ini}$) (line 13) uses simulated annealing [19] to find an optimal valuation for φ. The robustness degree of a formula generally increases or decreases monotonically in each parameter. However, we use simulated annealing rather than binary searches over the parameters because we are interested in optimizing a loss function of the robustness degree and are not necessarily trying to directly minimize it.

Function Maintenance(φ, v, c, q) (line 17) maintains the DAG by updating the nodes of $\mathcal{G}_i$ with the computed tuple (φ, v, c, q) where φ is the formula, v is the optimal valuation, c is the corresponding cost, and q is the corresponding *misclassification rate*, which is defined as the number of misclassified signals divided by the total number of signals N.

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. PruningAndGrowing($\mathcal{G}_{i-1}$) (line 7) does this by first eliminating a fixed number of nodes with high costs, i.e. those formulae that do not fit the observed data. Pruning the graph to eliminate high cost formulae follows naturally from forward subset selection ideas developed in machine learning [21]. Then, the function grows the pruned $\mathcal{G}_{i-1}$ to include nodes with length i according to graph manipulation rules detailed in Section 4.2. An example of a subset of a graph $\mathcal{G}_2$ grown from the (pruned) basis graph is given in Figure 5.

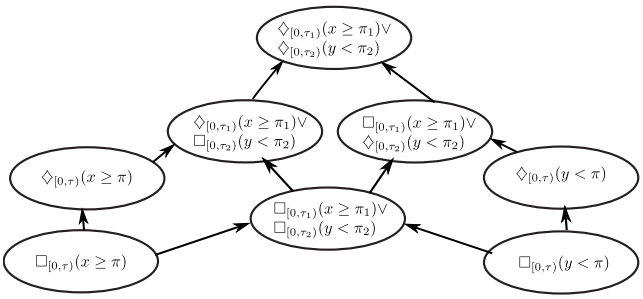

Figure 5: A subset of the DAG $\mathcal{G}_2$ after pruning and expansion

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

$$\frac{1}{|pa(k_i)|} \sum_{k_{i-1} \in pa(k_i)} J_e(k_{i-1}), \tag{11}$$

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 Figure 5, for $k_i = (\Diamond_{[0,\tau_1)}(x \geq \pi_1) \wedge (\Diamond_{[0,\tau_2)}(y < \pi_2))$, $pa(k_i) = \{\Diamond_{[0,\tau)}(x \geq \pi), (\Diamond_{[0,\tau)}(y < \pi)\}$ 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 L_{max} iterations are completed. At this point, MinimumCostNode($\mathcal{G}_i$) returns the node with the minimum cost within $\mathcal{G}_i$.

Compared to Alg. 1, the prediction algorithm searches inside the subsection of the DAG of $rPSTL$ corresponding to formulae $\Diamond_{[0,\tau_1)}(\varphi_c \Rightarrow \varphi_{des,e})$, where $\varphi_{des,e}$ is the output of Alg. 1. The prediction algorithm employs the same procedures and continues the search until a formula with a low enough misclassification rate is found.

Complexity.

Without pruning, the discrete layer of the described algorithms runs in time $\mathcal{O}(L_{max} \cdot 2^{|V|})$. Since PruningAndGrowing prunes a constant number of nodes at each iteration, the complexity of the discrete layer is reduced to $\mathcal{O}(L_{max} \cdot |V|^2)$ when pruning is applied. The continuous layer of the algorithm, based on the simulated annealing algorithm, runs in time $\mathcal{O}(L_{max}(n^2 + m) \cdot \log(N))$, with n being the number of samples used in simulated annealing, and m being the number of data points per signal.

Remark 2. It has been shown in [7] that the set of all linear temporal logic (LTL) formulae can also be organized in a DAG using a partial order similar to $\preceq_S$. However, unlike LTL, PSTL can express temporal specifications involving continuous-time intervals and constraints on continuously valued variables. To our own knowledge, our algorithm is the first of its kind which can be used to infer an STL formula by inferring both its PSTL structure and its optimal valuation.

Remark 3. The truncation time $\tilde{t}$ is specified by the user. The truncation time represents a temporal threshold between possible causes of behaviors (described by ϕ_c) and the observed effects of these behaviors (described by ϕ_e). Thus, it should be chosen such that the desirable and undesirable effects can be clearly seen in the truncated signals σ_i. In the absence of any intuition about the value of $\tilde{t}$, its value can be set to half the duration of signals. This is the value we use for both case studies in Section 6.

6. IMPLEMENTATION AND CASE STUDIES

The classification and prediction algorithms were implemented as a software tool called TempLogIn (TEMPoral LOGic INference) in MATLAB. We developed all of the components of our solution in-house, including the graph construction and search algorithms and the simulated annealing algorithm. Our procedure takes as inputs sets of labeled trajectories, desired confidence, a truncation time ($\tilde{t}$) and a maximum formula length, and infers an rSTL formula. The software is available at `http://hyness.bu.edu/Software.html`.

6.1 Herding Example

Table 1: Parameters Defining Relevant Areas (Unit: m)

	$\pi_{.l}$	$\pi_{.r}$	$\pi_{.d}$	$\pi_{.u}$
A	0	50	40	100
B	25	50	40	60
C	100	150	60	100
D	100	150	0	45

The first example we consider is the herding example given in Section 3.1. We generated 600 signals, 120 of which are shown in Figure 6. Table 1 shows the values of the parameters describing the boundaries of the different regions A, B, C, D shown in Figure 1 and Figure 6. The columns correspond to the boundaries: the left boundary $\pi_{.l}$, the right boundary $\pi_{.r}$, the lower boundary $\pi_{.d}$ and the upper boundary $\pi_{.u}$. The starting locations of the signals are chosen randomly within A. The starting velocities are all set to 0. The dynamics of the cow are described by

$$\begin{cases} \dot{x} &= v \cos \alpha \\ \dot{y} &= v \sin \alpha \\ \dot{v} &= a_t \\ \dot{\alpha} &= \omega, \end{cases} \tag{12}$$

where x and y are the cow's coordinates, v is its speed, and α is its heading.[1] The two controls are the tangential

[1]The simple unicycle dynamics (12) 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.

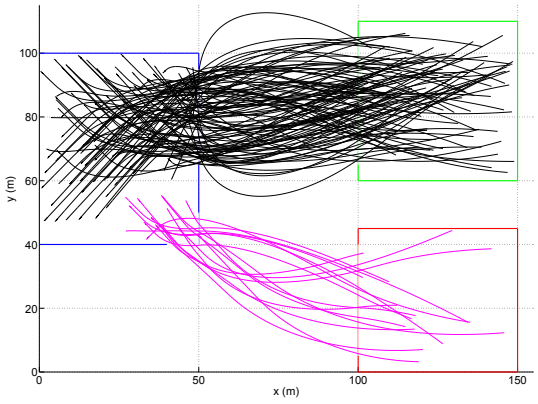

Figure 6: Synthesized trajectories with positive examples shown in black and negative ones shown in purple.

acceleration a_t and the angular velocity ω. These controls are generated based on a hybrid strategy. For a signal with $p_i = 1$ (a positive example), a random point is selected along the exit of A and acts as an attractor. The controls are generated from a potential function, a function of the distance between the selected point and the current location of the cow [14], until the cow reaches the exit. Then, the potential function-based control strategy is used to drive the cow to C. Signals in which $p_i = 0$ (a negative example) are generated similarly. All times are in minutes.

Table 2: Misclassification Rates for $i = 1$ (Classification)

	$\Diamond, <$	$\Box, <$	$\Diamond, \geq$	$\Box, \geq$
x	0.140	0.140	0.213	0.173
y	0	0	0.270	0.233
v	0.325	0.140	0.207	0.140
α	0.140	0.140	0.140	0.140

Classification For this case study, we are interested in inferring $\phi_{und,e}$, the formula describing the signals in which $p_i = 0$ (shown as purple in Figure 6). The variable set V is $\{x, y, v, \alpha\}$. Table 2 shows the misclassification rates of the nodes generated by our algorithm after parameter estimation occurs. The rows correspond to different variable choices. The columns correspond to different temporal operator and inequality combinations. For instance, the first row x and first column $\Diamond, <$ represent $\Diamond_{[\tau_1,\tau_2)}(x < \pi)$. In the following, we use the triple $(x, \Diamond, <)$ to represent it. It can be seen from the table that there are two formulae that have zero misclassification rate (underlined), which means the classification algorithm terminates with $i = 1$. To break the tie between these two formulae, the algorithm chooses the one with a lower cost $J_e(\varphi, v)$. The inferred effect formula is

$$\Box_{[0,3.02)}(y < 47.56), \tag{13}$$

which says that the undesirable cow behavior can be classified as "the y coordinate is always smaller than 47.56 meters for a period of 3.02 minutes". Notice that $y = 47.65$ is located between C and D which means that, for this specific case, we can classify signals by simply looking at their y coordinates.

Table 3: Misclassification Rates for $i = 1$ (Prediction)

	$\Diamond, \geq$	$\Box, \geq$	$\Diamond, <$	$\Box, <$
x	0.168	0.138	0.310	0.245
y	0.243	0.213	0.032	0.008
v	0.140	0.140	0.860	0.140
α	0.140	0.140	0.140	0.140

Prediction We again focus on signals in which $p_i = 0$. Table 3 shows the misclassification rates of the 16 basis nodes. It can be seen that there is no formula ϕ_c of length one that has zero misclassification rate. The three formulae with the lowest misclassification rates are underlined. The algorithm next grows the DAG to include formulae with length 2.

The algorithm then searches the new formulae in order according to the heuristic given in (11). The basis node with lowest cost is $(y, \Box, <)$ and the basis node with second lowest cost is $(y, \Diamond, <)$. The lowest ranked formula is the child of $(y, \Diamond, <)$, i.e. $(y, \Box, <)$, which has already been checked. Thus, the algorithm proceeds to the next lowest ranked candidate,which is the child of $(y, \Box, <)$ and $(x, \Box, \geq)$. This formula is $\Box_{[\tau_1,\tau_2)}(x \geq \pi_1) \wedge \Box_{[\tau_3,\tau_4)}(y < \pi_2)$. The algorithm was able to infer a cause formula with this template that has 0 misclassification rate. The total inferred rSTL formula is

$$\begin{aligned}\phi_{und} = \quad & \Diamond_{[0,12.00)}((\Box_{[0,0.65)}(x \geq 25.00) \wedge \\ & \Box_{[0,8.00)}(y < 58.99)) \Rightarrow \Box_{[8.98,12.00)}(y < 47.56)).\end{aligned} \tag{14}$$

The similarity between (14) and (8) shows that our inference algorithm can accurately capture the essence of the undesirable behaviors with the cause formula (LHS of (14)). The two scale parameters inferred by the algorithm are 25.00 and 58.99, which are close to the fencing breach location $(\pi_{bl}, \pi_{bu}) = (25, 60)$ (see Section 3.1 and Table 1). These results are achieved without any expert knowledge.

On a Mac with a 3.06 GHz Intel Core 2 Duo CPU and 6 GB RAM, the classification took 305.5 seconds, and the prediction took 643.1 seconds with n, the number of samples generated in simulated annealing, equal to 100, and m, the maximum number of data point per signal, equal to 200.

6.2 Biological Network

The second example we consider is from synthetic biology. In this field, gene networks are engineered to achieve specific functions [13, 20]. The robustness degree has previously been exploited in gene network design and analysis [18, 2, 5], but to our knowledge, template discovery has never been used to analyze a gene network. We consider the gene network presented in [10]. The network, shown in Figure 7, controls the production of two proteins, namely $tetR$ and RFP. This network is expected to work as an inverter in which the concentrations of $tetR$ and RFP can be treated as the input and the output, respectively. In particular, $tetR$ represses the production of RFP. A high $tetR$ concentration decreases the production rate of RFP, hence the concentration of RFP eventually decreases and stays low. Similarly, if the concentration of $tetR$ is low, then the

production of RFP is not repressed, and its concentration eventually increases and stays high.

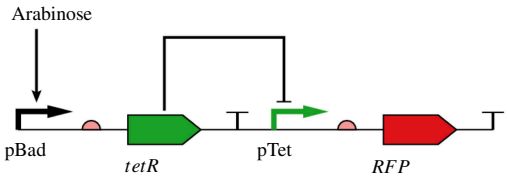

Figure 7: A synthetic gene network. The genes coding for proteins $tetR$ and RFP are shown as colored polygons. The promoters (pBad and pTet) regulating protein production rates are indicated by bent arrows. The regulators ($arabinose$ and $tetR$) are connected to the corresponding promoters.

In [10], a stochastic hybrid system modeling the gene network was constructed from characterization data of the biological network components. Statistical model checking was used to check a temporal logic formula (expressing a property chosen by biology experts) that describes the inverter behavior of the network. In this paper, sample trajectories of this system [2] are used to find a formula that describes the inverter behavior without any prior expert knowledge.

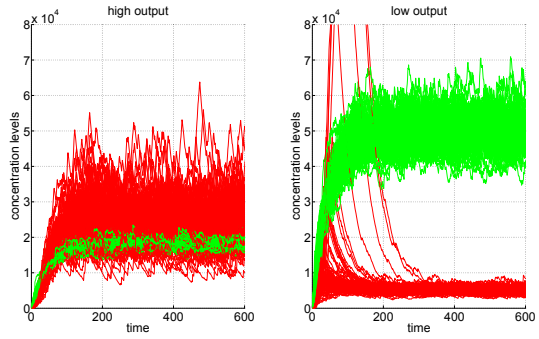

Figure 8: Concentration levels x_{tetR} (green) and x_{RFP} (red) for the high and low output cases. 100 signals are plotted for each protein and each case.

We generated 600 signals, half of which correspond to the low output case (repression) and half of which correspond to the high output case (no repression). Figure 8 shows 200 of these signals. Assume that we are interested in characterizing the low output case ($p_i = 1$ for low output signals). The inferred rSTL formula ϕ_{des} which classifies both cases and describes the pre-conditions for low output is

$$\phi_{des} = \begin{array}{l} \Diamond_{[0,118)}(\Box_{[0,340)}(x_{tetR} \geq 23209) \Rightarrow \\ \Box_{[188,323)}(x_{RFP} < 13479)) \end{array} \quad (15)$$

This formula captures the repressing effect of $tetR$, and shows that the designed gene network works as expected. In particular, the formula implies that $tetR$ represses the production of RFP when its concentration is higher than 23209 for 340 time units. Moreover, when the production of RFP is repressed, its concentration drops below 13479 within 188 time units. Such quantitative information learned from the formula helps the user to design more complex gene networks.

On the same computer used in the first case study, the classification procedure took 494.0 seconds while the prediction algorithm took 693.7 seconds. For the biological network case study, the number of samples generated by the simulating annealing, n, and the maximum number of data points per signal, m, were 100 and 600, respectively.

[2] $Arabinose$ regulates the production rate of $tetR$. We use trajectories generated at different concentration levels of $arabinose$. As we are interested in cause-effect relationship between $tetR$ and RFP, we omit the concentration of $arabinose$.

7. FINAL REMARKS AND FUTURE WORK

In this paper, we present a temporal logic inference framework, which, given a collection of labeled continuously valued signals, produces temporal logic formulae that can be used to describe and predict desirable and undesirable behaviors. We define reactive parametric signal temporal logic (rPSTL), a fragment of parametric signal temporal logic (PSTL) that can be used to describe causal relationships in systems. We exploit the properties of rPSTL and develop a hybrid temporal logic inference algorithm that searches for a formula template, as well as as its parameterization, that best fits the observed data. Two case studies, one on herding and the other one on biological networks, are used to illustrate our algorithm. The formulae mined from each case study appear to be consistent with the observed behaviors of each system.

While the example considered in Section 6 is relatively simple, the algorithm presented in the paper can be used to infer characteristics of more complex systems. Since our algorithm infers properties without any expert inputs, it is well-suited to tasks such as system reconstruction, e.g. inferring the purpose and capabilities of legacy code, and knowledge discovery, e.g. finding relevant properties of a complex biological network directly from the data. It can also serve as a first step for developing high-fidelity models of a complex system from a massive data set. For instance, from the inferred formula (15), it can be seen that the underlying system acts as an inverter. This conclusion can guide further revisions in experimental design as well as modeling.

Future research in this area includes applying our inference algorithm to more complex data, expanding the class of formulae which may be inferred automatically, and revising our algorithm to fit large data sets and unsupervised learning cases.

8. ACKNOWLEDGMENT

This work was supported in part by ONR MURI N00014-10-1-0952, ONR MURI N00014-09-1051, and NSF CNS-1035588. The authors would like to acknowledge members of CIDAR group at Boston University for providing experimental data for the gene network presented in [10].

9. REFERENCES

[1] E. Asarin, A. Donzé, O. Maler, and D. Nickovic. Parametric identification of temporal properties. In *Runtime Verification*, pages 147–160. Springer, 2012.

[2] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. On the robustness of temporal properties for stochastic models. In *Proceedings*

Second International Workshop on Hybrid Systems and Biology, volume 125, pages 3–19, 2013.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[4] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[5] A. Donzé, E. Fanchon, L. M. Gattepaille, O. Maler, and P. Tracqui. Robustness analysis and behavior discrimination in enzymatic reaction networks. *PLoS One*, 6(9):e24246, 2011.

[6] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems*, pages 92–106. Springer, 2010.

[7] G. E. Fainekos. Revising temporal logic specifications for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 40–45. IEEE, 2011.

[8] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[9] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[10] E. A. Gol, D. Densmore, and C. Belta. Data-driven verification of synthetic gene networks. In *52nd IEEE Conference on Decision and Control (CDC)*, 2013.

[11] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2004.

[12] X. Jin, A. Donze, J. Deshmukh, and S. Seshia. Mining requirements from closed-loop control models. In *Hybrid Systems: Computation and Control (HSCC)*, 2013.

[13] J. R. Kirby. Synthetic biology: Designer bacteria degrades toxin. *Nature Chemical Biology*, 6(6):398–399, 2010.

[14] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[15] N. Lavrac and S. Dzeroski. *Inductive Logic Programming*. E. Horwood, 1994.

[16] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1987.

[17] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 71–76, 2004.

[18] A. Rizk, G. Batt, F. Fages, and S. Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *Computational Methods in Systems Biology*, pages 251–268. Springer, 2008.

[19] S. J. Russell and P. Norvig. *Artificial Intelligence: a Modern Approach*. Prentice Hall, 1995.

[20] H. Salis, A. Tamsir, and C. Voigt. Engineering bacterial signals and sensors. *Contrib Microbiol*, 16:194–225, 2009.

[21] H. Trevor, T. Robert, and J. J. H. Friedman. *The elements of Statistical Learning*. Springer, 2001.

[22] H. Yang, B. Hoxha, and G. Fainekos. Querying parametric temporal logic properties on embedded systems. In *Testing Software and Systems*, pages 136–151. Springer, 2012.