4-2018

Parameter Invariant Monitoring for Signal Temporal Logic

Nima Roohi
University of Pennsylvania, roohi2@cis.upenn.edu

Ramneet Kaur
University of Pennsylvania, ramneetk@seas.upenn.edu

James Weimer
University of Pennsylvania, weimerj@seas.upenn.edu

Oleg Sokolsky
University of Pennsylvania, sokolsky@cis.upenn.edu

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

21st ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2018), Porto, Portugal, April 11-13, 2018

Parameter Invariant Monitoring for Signal Temporal Logic

Abstract
Signal Temporal Logic (STL) is a prominent specification formalism for real-time systems, and monitoring these specifications, specially when (for different reasons such as learning) behavior of systems can change over time, is quite important. There are three main challenges in this area: (1) full observation of system state is not possible due to noise or nuisance parameters, (2) the whole execution is not available during the monitoring, and (3) computational complexity of monitoring continuous time signals is very high. Although, each of these challenges has been addressed by different works, to the best of our knowledge, no one has addressed them all together. In this paper, we show how to extend any parameter invariant test procedure for single points in time to a *parameter invariant* test procedure for *efficiently monitoring continuous time* executions of a system against STL *properties*. We also show, how to extend probabilistic error guarantee of the input test procedure to a *probabilistic error guarantee* for the constructed test procedure.

Keywords
Parameter Invariant Test, Signal Temporal Logic, Monitoring, Run-time Verification

Disciplines
Computer Engineering | Computer Sciences

Comments
21st ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2018), Porto, Portugal, April 11-13, 2018

Parameter Invariant Monitoring for Signal Temporal Logic

Nima Roohi
University of Pennsylvania
Philadelphia, Pennsylvania
roohi2@cis.upenn.edu

Ramneet Kaur
University of Pennsylvania
Philadelphia, Pennsylvania
ramneetk@seas.upenn.edu

James Weimer
University of Pennsylvania
Philadelphia, Pennsylvania
weimerj@seas.upenn.edu

Oleg Sokolsky
University of Pennsylvania
Philadelphia, Pennsylvania
sokolsky@cis.upenn.edu

Insup Lee
University of Pennsylvania
Philadelphia, Pennsylvania
lee@cis.upenn.edu

ABSTRACT

*Signal Temporal Logic (*STL*)* is a prominent specification formalism for real-time systems, and monitoring these specifications, specially when (for different reasons such as learning) behavior of systems can change over time, is quite important. There are three main challenges in this area: (1) full observation of system state is not possible due to noise or nuisance parameters, (2) the whole execution is not available during the monitoring, and (3) computational complexity of monitoring continuous time signals is very high. Although, each of these challenges has been addressed by different works, to the best of our knowledge, no one has addressed them all together. In this paper, we show how to extend any parameter invariant test procedure for single points in time to a *parameter invariant* test procedure for *efficiently monitoring continuous time* executions of a system against STL *properties*. We also show, how to extend probabilistic error guarantee of the input test procedure to a *probabilistic error guarantee* for the constructed test procedure.

CCS CONCEPTS

• **Theory of computation** → *Probabilistic computation*; *Timed and hybrid models*; • **Mathematics of computing** → Probabilistic algorithms;

KEYWORDS

Parameter Invariant Test, Signal Temporal Logic, Monitoring, Runtime Verification

ACM Reference Format:
Nima Roohi, Ramneet Kaur, James Weimer, Oleg Sokolsky, and Insup Lee. 2018. Parameter Invariant Monitoring for Signal Temporal Logic. In . ACM, New York, NY, USA, 10 pages. https://doi.org/

1 INTRODUCTION

Many properties of concurrent and reactive systems can be formally specified using Linear Temporal Logic (LTL) [18], in which time is a discrete entity. In this logic, an execution of a system can be considered as a (countably infinite) sequence of events. However, no timing constraints can be specified on these events. This inability to express any timing constraint, severely limits usability of LTL for real time systems. Therefore, researchers looked for different extensions of LTL with the ability to specify timing requirements. *Metric Interval Temporal Logic (*MITL*)* is the most well-known such extension, where temporal operators can have timing constraints as well [15]. In MITL, atomic propositions are abstract entities and might change their truth values arbitrarily over time. For example, suppose velocity of an object at time t is denoted by $v(t)$. According to MITL, for any positive value ϵ, it is possible for $v(t) > 1$ and $v(t + \epsilon) < -1$ to be both true. However, in real world, there is always a bound on possible accelerations which makes this arbitrary change in velocity impossible. This intuitively means MITL abstracts away too much information. *Signal Temporal Logic (*STL*)* [16] is same as MITL, except that atomic propositions in STL are not abstract entities anymore, and one can use the extra information about each atomic proposition to bound how fast each one of them can change in the near future.

STL was first introduced in [16] for *monitoring* continuous signals. As opposed to the model checking problem, in monitoring we do not have access to an underlying model of the system (we might have some assumptions like Lipschitz continuity about it though). We can only look at a single execution of the system *as it happens*, and based on our observations we should decide to either stop or continue the monitoring. In case we decide to stop, we should output one of three possibilities: (1) the execution satisfies the input STL formula, (2) the execution does not satisfy the input STL formula, or (3) no matter how much longer we monitor, it would be impossible to conclude with certainty that one of the previous cases is true. The last case could happen if we fail to observe the system during some critical points in time. It could also happen if the observation cannot be performed precisely, and the amount of information that is lost become too much. Clearly, algorithms that always terminate and correctly return one of the first two outputs are more desirable.

Most research regarding monitoring and verification of STL properties assume exact state of the system can be observed at any point in time [6–8, 12, 16, 22]. However, this is usually not a feasible assumption. Let us consider monitoring of a diabetic patient as an example. Human physiology is inherently partially observable and we can only measure parts of it. These systems are often modeled as dynamical systems with state and output variables [14, 25], and although we can only look at output variables (*i.e.* state variables are not directly observable), many atomic propositions in STL properties are defined over state variables. The situation becomes even worse, when we take two more facts/challenges into account: (1) Output variables are often affected by noise in the environment/sensors. Therefore, even if we know the exact relation

Published at HSCC, April 2018, Porto, Portugal
.

between state and output variables, observing an output which is already affected by noise, does not give us the actual output value. (2) Current human physiology models are usually parametric. This intuitively means, while there is a general agreement on the shape of a model, many of its parameters cannot be estimated by conventional techniques or it would be quite costly and/or invasive to estimate them [4, 25]. Either way, these parameters, which we call *nuisance parameters*, may vary with time. As a result of these challenges, we cannot precisely evaluate atomic propositions used in a STL formula. For example, the metabolic rate affects the insulin-glucose dynamics but is difficult to estimate this effect in real-time and it is therefore considered a nuisance parameter. Note that requiring to only have atomic propositions on output variables is often impossible. As an example, consider an artificial pancreas which is a medical cyber-physical system that integrates a constant glucose monitor, wearable insulin pump, and control algorithms running on embedded computing devices. To monitor an artificial pancreas' response to meal ingestion requires us to know when a meal is taken [25] which is *not* an output variable of this device or its model. Therefore, we avoid relying on an exact model and turn to statistical techniques instead to reason about probabilistic beliefs about the system state.

Authors in [26] have recently introduced a test procedure that can only be used for obtaining probabilistic beliefs about state of the system at single points in time (*a.k.a.* atomic propositions in STL), but is maximally invariant to nuisance parameters (*e.g.* insulin sensitivity and metabolic rate). This intuitively means, not only this test procedure is able to ignore parts of information that are corrupted by the nuisance parameters, it only ignores a minimum amount of them. For example, suppose o_1 and o_2 are two observations that could have been affected by the nuisance parameters, and let o_1' and o_2' be what the algorithm in [26] obtains after filtering some of the information out. Being invariant to nuisance parameters means if they can change o_1 to o_2 or o_2 to o_1 then $o_1' = o_2'$. Being maximally invariant to nuisance parameters means if $o_1' = o_2'$ then we know nuisance parameters can always change o_1 to o_2 and o_2 to o_1.

There are two more challenges in monitoring continuous time system behaviors against STL properties. First, monitoring continuous time signals is computationally very expensive if not undecidable. For example, observing state of a system (*a.k.a.* monitoring state of a system) can only happen at discrete points in time. However, we are interested in verifying behavior of a system over continuous time domain. Second, when behavior of a system is monitored, we want to stop the procedure as soon as whatever has been observed so far is enough for making a decision [1]. This means the monitoring algorithm should be able to proceed without all data being available and should be able to terminate as soon as a decision can be made. Each of these problems has been considered in the past and researchers have already came up with solutions for them. However, to the best of our knowledge, none of these works addresses all these challenges combined. Some monitoring algorithms only work for discrete time executions [2, 3, 5, 11], while others assume piecewise constant or linear continuous time signals [6, 8]. But in our setting, we consider continuous time signals and only assume a bound on their Lipschitz continuity. Most of monitoring approaches, assume one can *fully* observe the system state at any point in time [2, 3, 5–8, 11, 12, 16, 22], however in our setting one can only have some probabilistic belief about system state at any point in time (see Section 5 for related works). Furthermore, we consider the case where nuisance parameters that are involved in our observations, are non-constant non-probabilistic entities affecting what we see as system state at different points in time.

Our main contribution in this paper is to show how one can extend any parameter invariant test procedure for atomic propositions (including the one that is introduced in [26]) to a parameter invariant test procedure for efficiently monitoring STL properties over continuous time signals. Furthermore, if the test procedure for atomic propositions is maximally invariant then the test procedure that we construct will be maximally invariant as well. Therefore, this would be the first time one addresses all these challenges in a single procedure. Note that our procedure relies only on probabilistic beliefs about truth values of atomic propositions at different points in time, and not on the actual truth values. If the test procedure for atomic propositions provides guaranteed error probability, our constructed test procedure provides guaranteed error probability as well. In this case, our algorithm takes an error parameter $\alpha : (0, 1)$ as one of its inputs, and guarantees the probability of returning an incorrect answer is always less than α. Although, in theory one can make α very small, making it too small may cause the test procedure for single points in time to fail.

Outline of the paper. In Section 2, we review definitions and results that we use in the rest of the paper. In Section 3, we show how to develop a parameter invariant monitor for STL properties. In Section 4, we show experimental results about using our algorithm. Finally, in Section 5, we review related works in this area, and conclude the paper in Section 6.

2 PRELIMINARIES

We denote the set of *natural*, *positive natural*, *real*, *positive real*, and *non-negative real* numbers by $\mathbb{N}$, $\mathbb{N}_+$, $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{R}_{\geq 0}$, respectively. For any two sets A and B, *size* of A is denoted by $|A|$, *Cartesian product* of A and B is denoted by $A \times B$, and *the set of functions from A to B* is denoted by $A \to B$ or B^A. For any set $C \subseteq A$ and function $f : A \to B$, we use $f_{\restriction C}$ to denote the restriction of f to C. We denote domain of f by $\mathtt{dom}(f)$. For any two functions $f : A \to B$ and $g : B \to C$, composition of f and g is denoted by $f \circ g$.

For any value $r : \mathbb{R}$, absolute value of r is denoted by $|r|$. Furthermore, we define r^+ and r^- to respectively be $\max\{0, r\}$ and $\max\{0, -r\}$. Intuitively, they are positive and negative parts of r. Similarly, for any function $f : A \to \mathbb{R}$, functions f^+ and f^- map a to respectively $(f(a))^+$ and $(f(a))^-$. For any two real numbers $a, b : \mathbb{R}$, we use $a \sqcup b$ and $a \sqcap b$ to denote $\sup\{a, b\}$ and $\inf\{a, b\}$, respectively. Finally, for any function $f : \mathbb{R}_{\geq 0} \to A$ and value $r : \mathbb{R}_{\geq 0}$, function $f^r : \mathbb{R}_{\geq 0} \to A : t \mapsto f(r + t)$ shifts f by r.

An *interval* is a convex subset of real numbers. For any $a : \mathbb{R} \cup \{-\infty\}$ and $b : \mathbb{R} \cup \{\infty\}$, we use the usual notations $[a, b]$, (a, b), $(a, b]$, and $[a, b)$ to denote different types of intervals. We define *width* of

[1] In this paper we do *not* consider *offline* monitoring, which means looking at a system execution after it is over.

an interval to be zero if it is empty, and $b-a$, otherwise. Finally, we use $\mathcal{I}_{\geq 0}$ to denote the set of non-empty intervals that have positive width and only contain non-negative values.

For any two sets A and B, *matrix with rows in A and columns in B* is a function of type $\mathbb{R}^{A\times B}$. For any matrix $M:\mathbb{R}^{A\times B}$, *transpose* of M, denoted by $M^\top$ is a function of type $\mathbb{R}^{B\times A}$ that maps (b,a) to $M(a,b)$. Also, *norm* of M, denoted by $\|M\|$, is defined to be $\sqrt{\sum_{a:A,b:B}(M(a,b))^2}$. A *(column) vector* is a matrix in which $|B|=1$. With a slight abuse of notation, we do not show B for vectors. Multiplication of matrices and vectors are defined in the usual way. If $\{M_1,\ldots,M_n\}$ is a set of matrices with the same set of row indices, columns of $M_1,\ldots,M_n$ are called *orthonormal* iff for any two columns c_1 and c_2 of these matrices, $\|c_1\|=\|c_2\|=1$ and inner product of c_1 and c_2 be zero.

2.1 Signal Temporal Logic

Let Z be a finite set of variables of a system that does not include variable t. In this paper a *system signal* is a function of type $\mathbb{R}_{\geq 0}\to\mathbb{R}^{\mathsf{Z}}$ that maps a point in time to a valuation of variables in Z. Let f be an arbitrary system signal, and let $\theta:\mathbb{R}^{\mathsf{Z}}\to\mathbb{R}$ be an arbitrary function. Function $\theta\circ f$ maps a point in time t to a real value r. Whenever $r>0$, we say θ is *true* on f at t, whenever $r<0$, we say θ is *false* on f at t, and whenever $r=0$, we say truth value of θ is *unknown* on f at t. Intuitively, absolute value of r represents a robustness degree for its truth value (larger absolute values of r represent stronger truth values). Every θ defines two predicates over $\mathbb{R}^{\mathsf{Z}}$ (set of points for which θ is positive and set of points for which θ is negative) and because of that we call θ a *test function*. Therefore, one can use θ as an atomic proposition in different kinds of temporal logic. We let Θ be a finite set of functions of type $\mathbb{R}^{\mathsf{Z}}\to\mathbb{R}$ that are used in this paper as atomic propositions. First, in Definition 2.1 we define syntax of STL formulas. We then define semantics of STL formulas for both continuous and discrete times signals. These semantics that are first introduced in [9] are usually called robust semantics, and are specifically introduced to tackle computational complexity of verifying continuous time signals against STL properties.

*Definition 2.1 (*STL *Syntax).* Syntax of a STL formula is defined using the following BNF formula, where by Θ and $\mathcal{I}_{\geq 0}$ we mean an arbitrary element of these sets.

$$\varphi ::= \top \mid \bot \mid \Theta \mid \neg\Theta \mid \varphi\vee\varphi \mid \varphi\wedge\varphi \mid \varphi\mathcal{U}_{\mathcal{I}_{\geq 0}}\varphi \mid \varphi\mathcal{R}_{\mathcal{I}_{\geq 0}}\varphi$$

In this definition, atomic propositions are by definition test functions. Other operators can be defined as syntactic sugars. For example, $\Diamond_{\mathtt{I}}\varphi$ (*i.e.* eventually ϕ within $\mathtt{I}$) and $\Box_{\mathtt{I}}\varphi$ (*i.e.* always ϕ within $\mathtt{I}$) operators are respectively defined to be $\top\mathcal{U}_{\mathtt{I}}\varphi$ and $\bot\mathcal{R}_{\mathtt{I}}\varphi$. Note that allowing negation only in front of atomic propositions, which is called *negated normal form* is not a restriction, and in general every formula that is not in negated normal form can be converted to an equivalent one that is in negated normal form [1, 9, 10, 19]. Therefore, for any STL formula φ, we use $\neg\varphi$ to denote a STL formula in negated normal form that is equivalent with negation of φ.

Semantics of a STL formula can be defined on both continuous and discrete time domains. While continuous semantics are usually what one uses for specifying desired behavior of a cyber-physical system, directly verifying them is often computationally very expensive if not undecidable [1, 9, 17, 19]. One the other hand, discrete semantics are usually easier to verify. An approach introduced in [9, 10] is to first prove a property holds using the discrete time semantics and then conclude that the same property or a slight modification of it holds using the continuous time semantics. Let $f:\mathbb{R}_{\geq 0}\to\mathbb{R}^{\mathsf{Z}}$ be a signal defined over continuous time domain. Let $\tau:\mathbb{N}\to\mathbb{R}_{\geq 0}$ be a function that maps natural numbers to points in time. Function $g\coloneqq f\circ\tau$ maps a natural number n to the state of the system at time $\tau(n)$ defined by f. In other words, g is a discrete signal that always agrees with f. We call τ a *sampling time* function. We first define continuous semantics of STL formulas in Definition 2.2, then we define discrete semantics of STL formulas in Definition 2.3. We use the symbol $\models$ in definition of semantics. To make distinguishing these two semantics easier, we write words CNT and DSC, in gray, below $\models$ to respectively refer to continuous and discrete semantics.

Note that in both Definition 2.2 and Definition 2.3, if robustness degree is positive, we say input signal satisfies the STL formula, and if robustness degree is negative, we say input signal does not satisfy the STL formula. Robustness degree 0 implies the signal neither satisfies nor not satisfies the formula. Special treatment of 0 implies semantics of STL as specified in these definitions, do not follow the *law of excluded middle* (*i.e.* it is possible for both a formula and its negation not to be true over a signal). However, the *law of non-contradiction* is still implied by these definitions (*i.e.* it is impossible for both a formula and its negation to be true over a signal). This is quite common whenever one considers some kind of robust semantics [9, 10, 20].

Definition 2.2 (Continuous Time STL *Semantics).* Let $f:\mathbb{R}_{\geq 0}\to\mathbb{R}^{\mathsf{Z}}$ be a system signal, and φ be an arbitrary STL formula. *Robustness degree* for f and φ at time $r:\mathbb{R}_{\geq 0}$, denoted by $f,r\models_{\text{CNT}}\varphi$, is a function that maps f, φ, and r, to a value in $\mathbb{R}\cup\{-\infty,\infty\}$. It is defined according to the following inductive rules:

$$\begin{aligned}
f,r\models_{\text{CNT}}\top &\coloneqq \infty & f,r\models_{\text{CNT}}\bot &\coloneqq -\infty\\
f,r\models_{\text{CNT}}\theta &\coloneqq \theta(f(r)) & f,r\models_{\text{CNT}}\neg\theta &\coloneqq -\theta(f(r))\\
f,r\models_{\text{CNT}}\varphi\vee\psi &\coloneqq (f,r\models_{\text{CNT}}\varphi)\sqcup(f,r\models_{\text{CNT}}\psi)\\
f,r\models_{\text{CNT}}\varphi\wedge\psi &\coloneqq (f,r\models_{\text{CNT}}\varphi)\sqcap(f,r\models_{\text{CNT}}\psi)\\
f,r\models_{\text{CNT}}\varphi\mathcal{U}_{\mathtt{I}}\psi &\coloneqq \bigsqcup_{t:r+\mathtt{I}}(f,t\models_{\text{CNT}}\psi\sqcap\bigsqcap_{r\leq t'<t} f,t'\models_{\text{CNT}}\varphi)\\
f,r\models_{\text{CNT}}\varphi\mathcal{R}_{\mathtt{I}}\psi &\coloneqq \bigsqcap_{t:r+\mathtt{I}}(f,t\models_{\text{CNT}}\psi\sqcup\bigsqcup_{r\leq t'<t} f,t'\models_{\text{CNT}}\varphi)
\end{aligned}$$

We use $f\models_{\text{CNT}}\varphi$ to denote $f,0\models_{\text{CNT}}\varphi$.

Definition 2.3 (Discrete Time STL *Semantics).* Let $f:\mathbb{R}_{\geq 0}\to\mathbb{R}^{\mathsf{Z}}$ be a system signal, φ be an arbitrary STL formula, and $\tau:\mathbb{N}\to\mathbb{R}_{\geq 0}$ be sampling time function. *Robustness degree* for $g\coloneqq f\circ\tau$, φ, and step $n:\mathbb{N}$, denoted by $g,n\models_{\text{DSC}}\varphi$, is a function that maps g, φ, and n to a value in $\mathbb{R}\cup\{-\infty,\infty\}$. It is defined according to the following

inductive rules:

$$
\begin{aligned}
g,n \models_{\text{DSC}} \top &:= \infty & g,n \models_{\text{DSC}} \bot &:= -\infty\\
g,n \models_{\text{DSC}} \theta &:= \theta(g(n)) & g,n \models_{\text{DSC}} \neg\theta &:= -\theta(g(n))\\
g,n \models_{\text{DSC}} \varphi \vee \psi &:= (g,n \models_{\text{DSC}} \varphi) \sqcup (g,n \models_{\text{DSC}} \psi)\\
g,n \models_{\text{DSC}} \varphi \wedge \psi &:= (g,n \models_{\text{DSC}} \varphi) \sqcap (g,n \models_{\text{DSC}} \psi)\\
g,n \models_{\text{DSC}} \varphi \mathcal{U}_{\mathrm{I}} \psi &:= \bigsqcup_{i:\tau^{-1}(\tau(n)+\mathrm{I})} \Big(g,i \models_{\text{DSC}} \psi \sqcap \prod_{n\le j<i} g,j \models_{\text{DSC}} \varphi\Big)\\
g,n \models_{\text{DSC}} \varphi \mathcal{R}_{\mathrm{I}} \psi &:= \prod_{i:\tau^{-1}(\tau(n)+\mathrm{I})} \Big(g,i \models_{\text{DSC}} \psi \sqcup \bigsqcup_{n\le j<i} g,j \models_{\text{DSC}} \varphi\Big)
\end{aligned}
$$

We use $g \models_{\text{DSC}} \varphi$ to denote $g, 0 \models_{\text{DSC}} \varphi$.

Before we state the relation between the two semantics specified in Definition 2.2 and Definition 2.3, we need to define an auxiliary function. It slightly shorten intervals of $\mathcal{U}$ formulas and slightly widen intervals of $\mathcal{R}$ formulas. The intuition is that robustness with respect to atomic propositions is not enough, and robustness with respect to time constraints is also required.

Definition 2.4 (Strengthening STL *Formulas).* For any STL formula φ and parameter $\delta : \mathbb{R}_{\ge 0}$, formula φ^{δ} is defined according to the following inductive rules:

$$
\begin{aligned}
&\top^{\delta} := \top \quad \bot^{\delta} := \bot \quad p^{\delta} := p \quad (\neg p)^{\delta} := \neg p\\
&(\varphi \vee \psi)^{\delta} := \varphi^{\delta} \vee \psi^{\delta} \qquad (\varphi \wedge \psi)^{\delta} := \varphi^{\delta} \wedge \psi^{\delta}\\
&(\varphi \mathcal{U}_{\mathrm{I}} \psi)^{\delta} := \varphi^{\delta} \mathcal{U}_{\left(\underline{\mathrm{I}}+\delta,\, \overline{\mathrm{I}}-\delta\right)} \psi^{\delta} \qquad (\varphi \mathcal{R}_{\mathrm{I}} \psi)^{\delta} := \varphi^{\delta} \mathcal{R}_{\left((\underline{\mathrm{I}}-\delta)^{+},\, \overline{\mathrm{I}}+\delta\right)} \psi^{\delta}
\end{aligned}
$$

If τ is strictly increasing and diverges to infinity then it would be easy to see that for any STL formula φ, system signal f, value $\delta : \mathbb{R}_{+}$, and sample time function τ with $\Delta\tau := \bigsqcup_{n:\mathbb{N}_{+}} (\tau(n) - \tau(n-1))$, if $(f \circ \tau \models_{\text{DSC}} \varphi^{\Delta\tau}) > \delta$ holds then $(f \circ \tau \models_{\text{DSC}} \varphi) > \delta$ holds as well [13]. Note that this immediately gives us $(f \circ \tau \models_{\text{DSC}} \varphi^{2\Delta\tau}) > \delta$ implies $(f \circ \tau \models_{\text{DSC}} \varphi^{\Delta\tau}) > \delta$. However, to conclude $(f \models_{\text{CNT}} \varphi) > 0$ from $(f \circ \tau \models_{\text{DSC}} \varphi^{\Delta\tau}) > \delta$, function τ and value δ should satisfy additional constraints. Theorem 2.5 formalizes a set of sufficient conditions and the relation between discrete and continuous semantics they entail.

Theorem 2.5 (Main Result of [9]). *Let φ be a* STL *formula, $f : \mathbb{R}_{\ge 0} \to \mathbb{R}^{\mathsf{Z}}$ be a system signal, and $\tau : \mathbb{N} \to \mathbb{R}_{\ge 0}$ be a sampling time function, with $\Delta\tau := \bigsqcup_{n:\mathbb{N}_{+}} (\tau(n) - \tau(n-1))$. For any $\delta : \mathbb{R}_{+}$, the following conditions guarantee*

$$
\begin{aligned}
(f \circ \tau \models_{\text{DSC}} \varphi^{\Delta\tau}) &> \delta \quad \Rightarrow \quad (f \models_{\text{CNT}} \varphi) > 0\\
(f \circ \tau \models_{\text{DSC}} (\neg\varphi)^{\Delta\tau}) &> \delta \quad \Rightarrow \quad (f \models_{\text{CNT}} \varphi) < 0
\end{aligned}
$$

(1) τ must be started early, i.e. $\tau(0) < \Delta\tau$.
(2) τ must be strictly increasing, i.e. $\forall i, j : \mathbb{N} \bullet i < j \Rightarrow \tau(i) < \tau(j)$.
(3) τ must diverges to infinity, i.e. $\forall r : \mathbb{R} \bullet \exists n : \mathbb{N} \bullet \tau(n) > r$.
(4) There must be $\lambda : \mathbb{R}_{+}$ such that for any $\theta : \Theta$, function $\theta \circ f$ is λ-Lipschitz continuous.
(5) δ must be large enough, i.e. $\lambda\Delta\tau < \delta$.
(6) $\Delta\tau$ must be small enough, i.e. $\Delta\tau < \frac{1}{3} \min_{\mathrm{I}:\mathcal{I}_{\varphi}} (\overline{\mathrm{I}} - \underline{\mathrm{I}})$, *where $\mathcal{I}_{\varphi}$ is the set of intervals that are appeared in temporal operators of formula φ.*

We use Theorem 2.5 to address computational complexity of monitoring continuous time system signals. We assume $\lambda : \mathbb{R}_{+}$ is given such that for any $\theta : \Theta$, function $\theta \circ f$ is λ-Lipschitz continuous. We then use Theorem 2.5 and find a small enough $\Delta\tau$ and large enough δ that satisfy all conditions in this theorem. Knowing λ, $\Delta\tau$, and δ, instead of trying to prove $(f \models_{\text{CNT}} \varphi) > 0$ or $(f \models_{\text{CNT}} \varphi) < 0$ directly, we try to prove $(f \circ \tau \models_{\text{DSC}} \varphi^{\Delta\tau}) > \delta$ or $(f \circ \tau \models_{\text{DSC}} (\neg\varphi)^{\Delta\tau}) > \delta$ [2]. Of course, Theorem 2.5 does not address the other two challenges. We cannot directly look at values of $\theta \circ f$ at different points in time, because of noise and nuisance parameters in the environment. Also, we cannot wait until the execution is completely over and then compute robustness degree defined by discrete semantics, which is what is required according to Definition 2.3.

When we verify a system signal f using a sampling time function τ, ideally we would like to see $f(0)$, which can only happen when $\tau(0) = 0$. Unfortunately, this cannot be guaranteed in practice. However, if we let τ' maps n to $\tau(n) - \tau(0)$, and $g' := f^{\tau(0)} \circ \tau'$ then it is easy to see $(g \models_{\text{DSC}} \varphi) = (g' \models_{\text{DSC}} \varphi)$. Using Theorem 2.5, intuitively this means if g' robustly satisfies φ then both f and $f^{\tau(0)}$ satisfy φ. To make the notations simpler, for the rest of the paper, *wlog.*, we assume $\tau(0) = 0$.

2.2 Parameter Invariant Test Statistic

For every test function θ and system signal f, Theorem 2.5 assumes at every sampling time that is defined by τ, we can fully observe value of $\theta \circ f$. However, in this paper we assume that direct observation of this value is impossible. This could happen for different reasons. For example, it could be because of noise in the environment or sensors, or because of some (unknown) nuisance parameters in the system. In Definition 2.6 from [26], we formally define what can be observed for each test function at different points in time, and later we define what can be observed for each system signal and sampling function.

Definition 2.6 (Observable States for a Single Test Function). To every test function θ and point in time $t : \mathbb{R}_{\ge 0}$, we associate a tuple $(\mathsf{Y}, \mu, \rho_0, \rho_1, \sigma, \mathsf{F}, \mathsf{G}_0, \mathsf{G}_1, n)$, where

- Y is an arbitrary finite non-empty set of *observable variables*. We require the set Y be independent of time.
- $\mu : \mathsf{dom}(\mu) \to \mathbb{R}$ and $\rho_i : \mathsf{dom}(\rho_i) \to \mathbb{R}$, for $i : \{0,1\}$, are unknown *nuisance vectors*. We require $\|\rho_0\| = \|\rho_1\| = 1$.
- $\sigma : \mathbb{R}_{+}$ is an unknown *nuisance noise multiplier*.
- $\mathsf{F} : \mathbb{R}^{\mathsf{Y}\times\mathsf{dom}(\mu)}$ and $\mathsf{G}_i : \mathbb{R}^{\mathsf{Y}\times\mathsf{dom}(\rho_i)}$, for $i : \{0,1\}$, are known *signal matrices*. We require that columns of $\{\mathsf{F}, \mathsf{G}_0, \mathsf{G}_1\}$ to be orthonormal.
- $n : (\mathbb{R}^{\mathsf{Y}}, \mathcal{F}, \mathbb{P})$ is a probability space of a *noise*, where $\mathbb{R}^{\mathsf{Y}}$ is a sample space, $\mathcal{F}$ is a sigma algebra on $\mathbb{R}^{\mathsf{Y}}$, and $\mathbb{P}$ is a probability measure on $\mathcal{F}$.

Every time we look at signal f, instead of observing $\theta(f(t))$, we only see $O(\theta, f, t) := F\mu + \theta^{+}(f(t))G_0\rho_0 + \theta^{-}(f(t))G_1\rho_1 + \sigma n$, which is a random variable with unknown parameters. Note that probability space of $O(\theta, f, t)$ is uniquely determined by its parameters (including those associated to θ and t). We refer to the elements associated to θ and t using subscripts. For example, we refer to nuisance noise multiplier using $\sigma_{\theta,t}$. Also, since Y is independent in time, we refer to observable variables of θ using Y_{θ}. We may

[2] Monitoring correctness of a guessed λ and defining a proper action whenever it is falsified would be among our future works.

drop the subscripts whenever it causes no confusion. We require $\forall\theta,\theta' : \Theta \bullet \theta \neq \theta' \Rightarrow \mathsf{Y}_\theta \cap \mathsf{Y}_{\theta'} = \emptyset$. Finally, we define $\mathsf{X} \coloneqq \bigcup_{\theta:\mathsf{AP}} \mathsf{Y}_\theta$ to be the set of all observable variables.

Definition 2.6 specifies what can be observed for every test function θ, time t, and system signal f. Having X defined and knowing different test functions use disjoint set of observable variables, what can be observed at time t using all elements of Θ has a probability space $O(f,t) \coloneqq (\mathbb{R}^{\mathsf{X}}, \mathcal{F}, \mathbb{P})$, which is uniquely determined by probability spaces of $O(\theta, f, t)$ for different $\theta : \Theta$.

Let $\tau : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be an injective sampling time function. According to Definition 2.3, for every test function $\theta : \Theta$ and step $n : \mathbb{N}$, we find value of $\theta(f(\tau(n)))$. Therefore, we would like to observe a function of type $\mathbb{N} \to \mathbb{R}^{\Theta}$ that maps n to values of atomic propositions (*a.k.a.* test functions) at time $\tau(n)$. However, we already know that we cannot directly observe these values, and our observation at each point in time is a random value with a probability space $O(f,t) \coloneqq (\mathbb{R}^{\mathsf{X}}, \mathcal{F}_{f,t}, \mathbb{P}_{f,t})$. Let $\Omega_{f,\tau} \coloneqq \mathbb{N} \to \mathbb{R}^{\mathsf{X}}$ be the product space of all possible observable values at different sample times. Let F be the set of all subsets of $\Omega_{f,\tau}$ that can be written in the form of $\{\omega \mid \forall n : \mathbb{N} \bullet \omega(n) \in A_n\}$, where $\forall n : \mathbb{N} \bullet A_n \in \mathcal{F}_{f,\tau(n)}$ and for all but finitely many $n : \mathbb{N}$, $A_n = \mathbb{R}^{\mathsf{X}}$ (every element of F is called a *cylinder set*). Define $\mathcal{F}_{f,\tau}$ to be the sigma algebra generated by F, and let $P : F \to [0,1]$ be a function that maps every element of F that is obtained from $A_0, A_1, \ldots$ to $\Pi_{n:\mathbb{N}} \mathbb{P}_{f,\tau(n)}(A_n)$. Author in [21] proved that there is a unique probability measure on $\mathcal{F}_{f,\tau}$ that extends P. We let $\mathbb{P}_{f,\tau}$ denote to that probability measure.

According to Definition 2.6, nuisance parameters can change the probability space of what can be observed at each point in time. Clearly, these nuisance parameters can also change the probability space of observable signals. Let Ω be a sampling space of an arbitrary probability space. A *group of transformation* on Ω is a set $\mathcal{K}$ of functions of type $\Omega \to \Omega$ that includes identity. Intuitively, functions in $\mathcal{K}$ are the result of nuisance parameters and any one of them can be applied to our observation; Meaning just by changing nuisance parameters it is possible to observe $k(\omega)$ instead of ω, where $k : \mathcal{K}$ is some transformation and $\omega : \Omega$ is an observation.

Knowing that $\mathcal{K}$ can affect our observations, one can benefit from designing a test statistic that is invariant to nuisance parameters [26]. Intuitively, a test statistic η is invariant to nuisance parameters iff different transformations in $\mathcal{K}$ cannot change its value. Definition 2.7 precisely states when a test statistic is invariant. Unfortunately, only requiring a test statistic to be invariant is not enough. For example, if η maps every input to 0 then it is clearly invariant to all nuisance parameters. However, it is obvious that such a test statistic is useless as it provides no information about the observed event. Definition 2.8 defines what is called maximally invariant test statistic. Intuitively, a test statistic is maximally invariant iff it only looses those parts of information that can be changed by nuisance parameters.

Definition 2.7 (Invariant Test Statistic). Let Ω be a sampling space of an arbitrary probability space, and $\mathcal{K} \subseteq \Omega \to \Omega$ be a group of transformation. A test statistic $\eta : \Omega \to \Omega'$, for some sampling space Ω', is called *invariant* to the group of transformations $\mathcal{K}$, iff

$$\forall \omega : \Omega, k : \mathcal{K} \bullet \eta(\omega) = \eta(k(\omega))$$

Definition 2.8 (Maximally Invariant Test Statistic). Let Ω, $\mathcal{K}$, and η be as defined in Definition 2.7. Test statistic η is called *maximally invariant* to the group of transformations $\mathcal{K}$ iff in addition to be an invariant test statistic it satisfies the following condition:

$$\forall \omega, \omega' : \Omega \bullet \eta(\omega) = \eta(\omega') \Rightarrow \exists k : \mathcal{K} \bullet \omega = k(\omega')$$

3 PARAMETER INVARIANT TEST FOR STL

In this section, let f be a system signal, τ be a sampling time function, $g \coloneqq f \circ \tau$ be the corresponding discrete signal, h be the corresponding observable random signal, and φ be a STL formula. Let $\mathcal{H}_0 \coloneqq (f \models_{\text{CNT}} \varphi) > 0$ and $\mathcal{H}_1 \coloneqq (f \models_{\text{CNT}} \varphi) < 0$ be the null and alternative hypotheses. Note that these hypotheses are by definition disjoint. Furthermore, knowing τ, every system signal can be thought as a parameter that induces a probability space defined in Section 2.2. Therefore, $\mathcal{H}_0$ and $\mathcal{H}_1$ are composite hypotheses. In this section, we develop a statistical procedure for testing $\mathcal{H}_0$ against $\mathcal{H}_1$. We assume there is a procedure for estimating values of test functions at different points in time, and is precisely specified in Assumption 1.

Assumption 1. *For any sequence $\mathsf{x} : \Omega_{f,\tau}$, test function $\theta : \Theta$, error parameter $\alpha' : \mathbb{R}_+$, and indifference region $\delta' : \mathbb{R}_+$, one can find an algorithm $\mathcal{A}(\mathsf{x}, \tau, \theta, \alpha', \delta')$ that terminates with probability 1 and outputs value $r : \mathbb{R}$. The algorithm guarantees $R_{\theta,\alpha',\delta'} \coloneqq \{\mathsf{x} : \Omega_{f,\tau} \mid |\theta(f(\tau(0))) - r| > \delta'\}$ is measurable (*i.e.* $R_{\theta,\alpha',\delta'} \in \mathcal{F}_{f,\tau}$), and $\mathbb{P}_{f,\tau}(R_{\theta,\alpha',\delta'}) < \alpha'$ (*i.e.* the probability of returning an answer that is more than δ' away from the true answer is less than α').*

Suppose $\mathcal{A}$ in Assumption 1 uses a (maximally) invariant test statistic. It would be valuable if we can use that test statistic and construct a test statistic for observable signals that is also (maximally) invariant. In Theorem 3.1, we show this is always possible. In other words, for any test statistic η that is defined for single points in time, one can construct a test statistic η' that is defined for random observable signals, such that if η is (maximally) invariant then η' is also (maximally) invariant.

Results in [26] are defined when we only consider one test function and look at a single point in time. However, in this paper we are dealing with multiple test functions at multiple points in time. Therefore, in order to establish our theoretical results, we also need to extend group of transformations twice. Once group of transformations that deals with multiple test functions, and once group of transformations that deals with multiple points over time. Let $\mathcal{K}_{\theta,f,\tau(n)} \subseteq \mathbb{R}^{\mathsf{Y}_\theta} \to \mathbb{R}^{\mathsf{Y}_\theta}$ be the group of transformations for the test function θ and signal f at time $\tau(n)$. Group of transformations for what can be observed at every step using all test functions, denoted by $\mathcal{K}_{f,\tau(n)}$, is a subset of $\mathbb{R}^{\mathsf{X}} \to \mathbb{R}^{\mathsf{X}}$ and is completely determined by $\mathcal{K}_{\theta,f,\tau(n)}$, since if $\theta \neq \theta'$ then $\mathsf{Y}_\theta \cap \mathsf{Y}_{\theta'} = \emptyset$. More precisely, for every $\nu : \mathbb{R}^{\mathsf{X}}$, $\theta : \Theta$, and $k : \mathcal{K}_{f,\tau(n)}$, function k maps $\nu_{\restriction \mathsf{Y}_\theta}$ to $k_\theta(\nu_{\restriction \mathsf{Y}_\theta})$, for some arbitrary $k_\theta : \mathcal{K}_{\theta,f,\tau(n)}$, and there is no restriction on choosing k_θ. Finally, group of transformations for the whole observable signal, denoted by $\mathcal{K}_{f,\tau}$, is a subset of $\Omega_{f,\tau} \to \Omega_{f,\tau}$ and is completely determined by $\mathcal{K}_{f,\tau(n)}$, since according to Definition 2.6, nuisance parameters at different times are not required to have any relation with each other. More precisely, for every $\omega : \Omega_{f,\tau}$, $n : \mathbb{N}$, and $k : \mathcal{K}_{f,\tau}$, function k maps

$\omega(n)$ to $k_n(\omega(n))$, for some arbitrary $k_n : \mathcal{K}_{f,\tau(n)}$, and there is no restriction on choosing k_n.

Theorem 3.1. *For any system signal f, sampling time function τ, test function θ, and step n, let test statistic $\eta_{\theta,f,\tau(n)} : \mathbb{R}^{\mathsf{Y}} \to \Omega'_{\theta,f,\tau(n)}$ be invariant with respect to the group of transformations $\mathcal{K}_{\theta,f,\tau(n)}$. There is a test statistic $\eta_{f,\tau} : \Omega_{f,\tau} \to \Omega'_{f,\tau}$ that is invariant to the group of transformations $\mathcal{K}_{f,\tau}$. Furthermore, if every $\eta_{\theta,f,\tau(n)}$ is maximally invariant then $\eta_{f,\tau}$ is also maximally invariant.*

A crucial assumption in proving Theorem 3.1 is that not only nuisance parameters of different test functions are not assumed to have any particular relation with each other, but they can also change arbitrarily over discrete time steps specified by τ. If this assumption does not hold in practice, the test statistic that is created in Theorem 3.1 may not be maximal anymore. However, it will remain invariant to the group of transformations.

Theorem 3.1 immediately gives us a (maximally) invariant test procedure to verify discrete time signals against STL properties. Note that using Theorem 2.5, if a discrete time signal $f \circ \tau$ robustly satisfies STL formula $\varphi^{\Delta\tau}$ then the continuous time signal f satisfies φ as well (not necessarily robustly). In this procedure, an estimation of robustness degrees for atomic propositions at all points in times that are specified by τ is obtained using Assumption 1 [3]. Then one uses Definition 2.3 to compute the robustness degree $f \circ \tau \models_{\text{DSC}} \varphi^{\Delta\tau}$. Theorem 3.1 guarantees this procedure is (maximally) invariant with respect to the group of transformations $\mathcal{K}_{f,\tau}$. However, neither Theorem 3.1 nor Theorem 2.5 consider the next challenge in monitoring, which is the requirement to be able to proceed when not all the data is provided and to stop the procedure as soon as enough information has been obtained.

3.1 Algorithm

We use the results from [6] for monitoring. Authors in that paper, monitor a STL signal with *continuous time* semantics by assuming it is *piecewise constant*. However, this is very similar (but not the same) to what one can assume about our *discrete time* signal, meaning value of observed signal is constant between consecutive observations [4]. First, in Assumption 2, we precisely state the main result of [6]. Later we show how to use this result in our case.

Assumption 2 (Monitoritng Piecewise Constant Signals). *Let $\tau : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be a strictly increasing sampling time function, $g' : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\mathsf{X}}$ be continuous time signal, and φ be a* STL *formula. If g' is piecewise constant with respect to τ, meaning $\forall n : \mathbb{N}, t : [\tau(n), \tau(n+1)) \bullet g'(t) = g'(\tau(n))$ then there is an algorithm $\mathcal{M}$ for monitoring g' against φ. Let $n : \mathbb{N}$ be an arbitrary value and $N := \{0, \ldots, n\}$ be the set of steps up to n. Also, let $l : N \to \mathbb{R}^{\Theta}$ be robustness degrees for every test function in Θ at every time step in N. More precisely, l maps $n : \mathbb{N}$ and $\theta : \Theta$ to $\theta(g'(\tau(n))_{\restriction \mathsf{Y}_\theta})$. The algorithm takes three inputs: (1) l, as what has been observed/monitored so far, (2) $\tau_{\restriction N}$, as times at which each observation has been made, and (3) φ, as a* STL *formula to monitor the input signal against. The algorithm outputs an interval $[a, b]$. It guarantees that a and b are respectively the infimum and supremum of all possible robustness degrees (*i.e. *$g' \models_{\text{CNT}} \varphi$) that one can possibly gets if what has been already observed (*i.e. *input functions l and $\tau_{\restriction N}$) be extended to functions with $\mathbb{N}$ as their domain in an arbitrary way.*

Three remarks are in order regarding Assumption 2 and the algorithm introduced in [6]. First, the implementation in [6] creates an internal state for the algorithm. There will be some initialization, but at any step $n : \mathbb{N}$, one only gives $l(n)$ and $\tau(n)$ to the algorithm. Clearly, this could save a lot of time, but the functionality remains the same. Second, if all intervals used in the input STL formula are bounded then it can be handled by [6]. However, STL formulas with unbounded intervals are not fully supported in that paper. Nonetheless, [6, Theorem 1] proves the following STL formulas can be monitored by their algorithm: (1) $\Box\varphi$, (2) $\Diamond\varphi$, (3) $\varphi\mathcal{U}\psi$, (4) $\Box(\varphi \vee \Diamond\psi)$, (5) $\Diamond(\varphi \wedge \Box\psi)$, (6) $\Box\Diamond\varphi$, (7) $\Diamond\Box\varphi$, (8) $\Diamond(\varphi \wedge \Diamond\psi)$, and (9) $\Box(\varphi \wedge \Box\psi)$, where φ and ψ are arbitrary non-temporal STL formulas, and interval in all temporal operators is $(0, \infty)$. The third remark is about whether or not the input STL formula should be given in negated normal form. Theorem 2.5 requires the input formula to be in negated normal form, however, the algorithm $\mathcal{M}$, as described in [6] does not require that. As a result, [6] does not directly support $\vee$ nor $\mathcal{R}$ operators, because one can always encode them using $\wedge$, $\mathcal{U}$, and $\neg$ operators. We need to make sure that this inconsistency is not going to be a problem, and in fact it is not. The assumption of input formula being in negated normal form is only used in Definition 2.4, where intervals of $\mathcal{U}$ operators are always shortened and intervals of $\mathcal{R}$ are always widened. However, if it was possible to write $\neg(\varphi\mathcal{U}_{\mathtt{I}}\psi)$ then interval $\mathtt{I}$ must have been extended. After this step, one can remove all $\vee$ and $\mathcal{R}$ operators by encoding them using $\wedge$ and $\mathcal{U}$ operators [10, P. 4268] [5].

Let g' be the piecewise constant continuous time signal that is constructed from g and τ as specified in Assumption 2. According to Theorem 2.5, in order to conclude $(f \models_{\text{CNT}} \varphi) > 0$, it is enough to verify $(g \models_{\text{DSC}} \varphi^{\Delta\tau}) > \delta$. However, using Assumption 2, we can only check $(g' \models_{\text{CNT}} \varphi^{\Delta\tau}) > \delta$. Lemma 3.2 solves this problem, by proving $(g' \models_{\text{CNT}} \varphi^{2\Delta\tau}) \leq (g \models_{\text{DSC}} \varphi^{\Delta\tau})$. Therefore, in order to conclude $(f \models_{\text{CNT}} \varphi) > 0$, it is enough to verify $(g' \models_{\text{CNT}} \varphi^{2\Delta\tau}) > \delta$. Similarly, in order to conclude $(f \models_{\text{CNT}} \varphi) < 0$, it is enough to verify $(g' \models_{\text{CNT}} (\neg\varphi)^{2\Delta\tau}) > \delta$ (note that, as stated in Section 2.1, by $\neg\varphi$ we mean a formula in negated normal form that is equivalent with $\neg\varphi$). Also, since we are monitoring $g' \models_{\text{CNT}} \varphi^{2\Delta\tau}$ and $g' \models_{\text{CNT}} (\neg\varphi)^{2\Delta\tau}$, we can stop the algorithm as soon as $\mathcal{M}$ outputs $[a, b]$ with $a > \delta$ for any of $\varphi^{2\Delta\tau}$ or $(\neg\varphi)^{2\Delta\tau}$ formulas.

Lemma 3.2 (Piecewise Constant and Discrete Semantics). *For any system signal f, strictly increasing sampling time function τ with finite $\Delta\tau := \sup_{n:\mathbb{N}} \tau(n+1) - \tau(n)$ and $\tau(0) = 0$, let $g := f \circ \tau$ be the corresponding discrete signal, and $g' : \mathbb{R}_{\geq 0} \to \mathbb{R}^{\mathsf{X}}$ that maps $t : \mathbb{R}_{\geq 0}$ to $g(n)$ with n being the unique number determined by $\tau(n) \leq t < \tau(n+1)$, be the piecewise constant signal corresponding to g. The following inequality holds:*

$$(g' \models_{\text{CNT}} \varphi^{\Delta\tau}) \leq (g \models_{\text{DSC}} \varphi)$$

To provide the values of signal at discrete times, we use Assumption 1. Let $\theta : \Theta$ be an arbitrary test function, and $\alpha : (0, 1)$ be an

[3] An explanation on how to handle error probability and indifference region is be given in Section 3.1.

[4] Authors in [7] introduced an algorithm for monitoring discrete time signals. In Section 5, we explain why we did not use their results.

[5] Furthermore, neither $\top$ nor $\bot$ are supported in [6]. However, one can trivially encode them using a constant test function which is always ∞ or always $-\infty$.

arbitrary bound on the error probability. Using Assumption 1 at step $n : \mathbb{N}$, suppose we know with error probability $< \alpha$, value of $p \coloneqq \theta(f(\tau(n)))$ is within the interval $[p' - \delta', p' + \delta']$, where p' is the output of algorithm and $\delta' : \mathbb{R}_+$ is an arbitrary value. If $p' - \delta' > 0$ then we know (with probability at least $1 - \alpha$) $p > 0$ and if $p' + \delta' < 0$ then we know (with probability at least $1 - \alpha$) $p < 0$. However, if neither of these cases hold we cannot conclude $p > 0$ nor $p < 0$. In the first two cases, we respectively use $p' - \delta'$ and $p' + \delta'$ as the robustness degree of θ at time $\tau(n)$. In the third case we use 0 as the robustness degree of θ, meaning we don't know enough about value of θ at time $\tau(n)$.

In the previous paragraph, we have explained how to use Assumption 1 to provide input to the monitoring algorithm in [6]. However, using Assumption 1 involves probabilistic errors and we have to bound this error. In order to estimate value of an arbitrary $\theta : \Theta$ at an arbitrary step $n : \mathbb{N}$, if we use error bound $\frac{\alpha}{2^{n+1}|\Theta|}$ then the total error after any number of steps will be $< \alpha$. Note that we do not assume any independence between outputs of the algorithm in Assumption 1 at different steps or for different test functions.

Algorithm 1 puts all the steps of our approach together. Inputs to this algorithm are a system signal f, a STL formula φ, a bound on error probability $\alpha : (0, 1)$, a Lipschitz constant $\lambda : \mathbb{R}_+$, and an indifference region $\delta' : \mathbb{R}_+$. Note that the algorithm cannot look at f directly. Parameter δ' will be used when $\mathcal{A}$ from Assumption 1 is invoked. Smaller values make our algorithm more precise, but too much small values may cause $\mathcal{A}$ to fail. Parameter α is a bound on error probability that our algorithm guarantees, and will be discussed later. However, similar to δ', smaller values of α make the algorithm more precise with the possibility of causing $\mathcal{A}$ to fail for values that are too small.

First an empty sequence of observations is created and the current step is set to 0. Then, using Theorem 2.5 and the input λ, two values are fixed for δ and $\Delta\tau$. Value of $\Delta\tau$ is assumed to be an upper bound on the actual value of $\inf_{n:\mathbb{N}} \tau(n+1) - \tau(n)$, meaning time of consecutive samples must not be more than $\Delta\tau$ apart. In practice, this value is determined based on the system/sensor specifications (meaning how fast one can sample from the input signal). Knowing $\Delta\tau$ and λ and using Theorem 2.5, we set value of δ to be slightly larger than $\lambda\Delta\tau$. After initializing these variables at the beginning, we initialize two instances of algorithm $\mathcal{M}$ for $\varphi^{2\Delta\tau}$ and $(\neg\varphi)^{2\Delta\tau}$ at Line 24 and Line 25. Inside the while loop at Line 28, we monitor the signal for as long as we could not prove robustness degree for $\varphi^{2\Delta\tau}$ or $(\neg\varphi)^{2\Delta\tau}$ is larger than δ. At any point in time, if $a_1 > \delta$ then using Lemma 3.2 and Theorem 2.5, we know $(f \models_{\text{CNT}} \varphi) > 0$. Similarly, if $a_2 > \delta$ then using the same lemma and theorem, we know $(f \models_{\text{CNT}} \varphi) < 0$. Note that since the law of non-contradiction is satisfied by both Definition 2.2 and Definition 2.3, using Lemma 3.2 and Theorem 2.5, we know it would be impossible for $a_1 > \delta$ and $a_2 > \delta$ to hold at the same time. However, it is possible that $b_1 \leq \delta$ and $b_2 \leq \delta$ both become true. If this happens, using what is guaranteed about $\mathcal{M}$ in Assumption 2, we know no matter how much longer we monitor the system, none of $a_1 > \delta$ and $a_2 > \delta$ would ever become true. In this case, we immediately stop the algorithm and return unknown. As far as termination is concerned, the algorithm will not continue if $a_1 = b_1$ and $a_2 = b_2$ are both true. If intervals in the input STL formula are all bounded, this will always happen. However, if input STL formula has unbounded intervals it is possible for the algorithm not to terminate. For example, consider $\varphi \coloneqq \Diamond_{(0,\infty)}\psi$ formula and assume the input system signal f does not satisfy it (*i.e.* robustness degree is < 0). Value of a_1 will always be negative and value of b_1 will always be ∞. Therefore, our algorithm will not terminate (unless it makes a mistake). However, this is true for all monitoring algorithms, since in this case, any finite prefix of f can be extended in way to satisfy φ or not to satisfy φ. Also, in this paper it is assumed that the input signal is defined over all of $\mathbb{R}_{\geq 0}$. However, if for any reason, the signal is defined only over a bounded time domain and if the algorithm did not terminate after all the observations are made, one can look at the last value of $[a_1, b_1]$ and $[a_2, b_2]$ and take a proper action based on those values.

Function *estimate* is where we use Assumption 1. First, note that according to Assumption 1, we need to give $\mathcal{A}$ two infinite sequences. However, it is impossible to actually wait for infinite amount of time and obtain all those data. Furthermore, if $\mathcal{A}$ ever terminates, it can only read finite amount of data from these sequences. That is why, instead of reading an infinite amount of information, we pass the *observe* function to $\mathcal{A}$. This way $\mathcal{A}$ can access all the information that are needed and function *observe* will provide them on the fly. However, even if $\mathcal{A}$ looks at each point in time at most once, we need to temporarily save all the observations as later invokations of *estimate* may need them. As soon as estimation of test functions at time t is over, we remove all data collected at time t from memory. Note that function *observe* returns both time and observation at that time. That is why we do not give two separate sequences to $\mathcal{A}$ (one for observations and one for times of those observations).

It is only remained to show that the probability of returning an *incorrect answer* is $< \alpha$. Output accept is considered incorrect iff $(f \models_{\text{CNT}} \varphi) < 0$, and output reject is considered incorrect iff $(f \models_{\text{CNT}} \varphi) > 0$. However, the algorithm can also return unknown as its output. We consider unknown as an incorrect answer iff the piecewise constant signal that is induced by f and τ, either robustly satisfies $\varphi^{2\Delta}$ or robustly satisfies $(\neg\varphi)^{2\Delta}$. Theorem 3.3 precisely states all of these cases about our error guarantee.

Theorem 3.3. *For every system signal f, Lipschitz constant λ, sampling time function τ, and* STL *formula φ, if f, λ, τ, and φ, satisfy conditions of Theorem 2.5 then Algorithm 1 can be used to test $\mathcal{H}_0$ against $\mathcal{H}_1$. The algorithm guarantees*

$$\begin{aligned}
&\mathbb{P}_{f,\tau}(\texttt{accept} \mid \mathcal{H}_1 \textit{ is true}) < \alpha\\
&\mathbb{P}_{f,\tau}(\texttt{reject} \mid \mathcal{H}_0 \textit{ is true}) < \alpha\\
&\mathbb{P}_{f,\tau}(\texttt{unknown} \mid (g' \models_{\text{CNT}} \varphi^{2\Delta\tau}) > \delta + \delta' \vee\\
&\qquad\qquad (g' \models_{\text{CNT}} (\neg\varphi)^{2\Delta\tau}) > \delta + \delta') < \alpha
\end{aligned}$$

where, g' is the peicewise constant continuous time signal corresponding to $f \circ \tau$ and is formally specified in Lemma 3.2. Also, δ is defined in Theorem 2.5, and δ' is the indifference parameter given as input to Algorithm 1.

Proof. First note that what Algorithm 1 constructs is g'', which is not exactly equal to g'. However, with error probability $< \alpha$, we know $\forall t : \mathbb{R}_{\geq 0}, \theta : \Theta \bullet |\theta(g''(t))| \leq |\theta(g'(t))| \wedge |\theta(g''(t)) - \theta(g'(t))| \leq \delta'$, which means (1) abosolute value of robustness degree of every test function in Θ is always smaller in g'' than g',

and (2) robustness dgrees of any test function Θ is always δ'-close in g'' and g'.

From the first property, we know if $(g'' \models_{\text{CNT}} \varphi^{2\Delta\tau}) > \delta$ then $(g' \models_{\text{CNT}} \varphi^{2\Delta\tau}) > \delta$. Similarly, if $(g'' \models_{\text{CNT}} (\neg\varphi)^{2\Delta\tau}) > \delta$ then $(g' \models_{\text{CNT}} (\neg\varphi)^{2\Delta\tau}) > \delta$. Therefore, the first two probabilities are clear from the explanations right before Algorithm 1 is explained and Line 14 in this algorithm.

The last inequality intuitively says that the probability of returning unknown, if $\varphi^{2\Delta\tau}$ or $(\neg\varphi)^{2\Delta\tau}$ are robustly satisfied by g', is less than α. The second property in the first paragraph of this proof implies if $(g' \models_{\text{CNT}} \varphi^{2\Delta\tau}) > \delta + \delta'$ then $(g'' \models_{\text{CNT}} \varphi^{2\Delta\tau}) > \delta$. Similarly, if $(g' \models_{\text{CNT}} (\neg\varphi)^{2\Delta\tau}) > \delta + \delta'$ then $(g'' \models_{\text{CNT}} (\neg\varphi)^{2\Delta\tau}) > \delta$. Either way, with probability $> 1 - \alpha$, unknown will not be returned by the algorithm. □

Using Theorem 3.1, we know if the test procedure in Assumption 1 is invariant to the nuisance parameters then our algorithm defines a parameter invariant test procedure for monitoring STL properties. The same theorem also proves that if the algorithm in Assumption 1 is maximally invariant then our algorithm will be maximally invariant as well. Using Theorem 3.3, we know if the algorithm in Assumption 1 fulfills its guarantee then the probability that our algorithm returns a wrong answer would be $< \alpha$.

4 EXPERIMENTAL RESULTS

There are about 1.25 million people with Type 1 diabetes (T1D) in the United States; a number that is expected to rise to 5 million by 2050 [25]. T1D patients control their glucose levels through daily insulin therapy to avoid numerous long-term complications associated with hyperglycemia. A significant disturbance to a T1D patient's glucose level can be caused by meal carbohydrates. Therefore, it is crucial to monitor insulin injections around meal times to avoid postprandial hyperglycemia and subsequent postcorrection hypoglycemia. Whenever a patient receives a meal, (s)he is supposed to either already received a bolus not longer than t_1 units of time ago, or be going to receive a bolus not later than t_2 units of time in the future. Similarly, whenever a patient receives a bolus, (s)he is supposed to either already received a meal not longer than t_1 units of time ago, or be going to receive a meal not later than t_2 units of time in the future. These two requirements are specified in Formula 1a and Formula 1b.

The amount of bolus that is received by a patient, denoted by B, is known exactly (*i.e.* there is no noise or nuisance parameter involved in measuring B). However, the amount of meal that is received by a patient, denoted by M, cannot be observed directly and our observations can be affected by noise as well as nuisance parameters. We used maximally invariant test procedure that is developed in [25, 26] to monitor values of M at different points in time. We used $c_1 = 0.4697$, which is exactly what is used in the experimental results of [25] (*i.e.* we consider a meal is taken whenever maximally invariant test statistic developed in that paper is larger than 0.4697). We used $c_2 = 0.1$, since we want to consider $B = 0$ as no bolus with non-zero robustness degree. The time bound on $\Box$ operator starts from t_1 as opposed to 0. Because, for example, if a patient takes a meal at some time $t < t_1$ then $t - t_1 < 0$ and we don't know if any bolus has been taken before the monitor started. We used $t_1 = t_2 = 30$ minutes in our experiments.

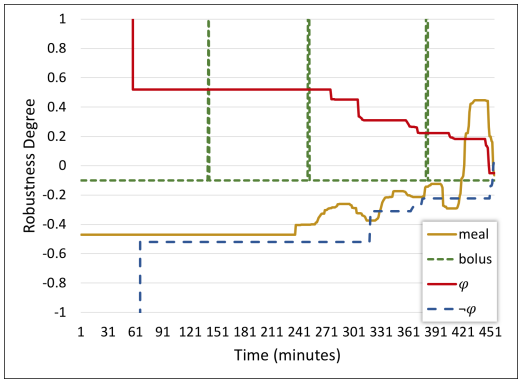

Figure 1: Sample signal of one observation of a patient. No error is involved in measuring bolus. Meal cannot be observed directly and its graph is the output of the algorithm in [26].

$$\Box_{[t_1,\infty)}\Big((M > c_1) \rightarrow \Diamond_{(-t_1,t_2)}(B > c_2)\Big) \tag{1a}$$

$$\Box_{[t_1,\infty)}\Big((B > c_2) \rightarrow \Diamond_{(-t_1,t_2)}(M > c_1)\Big) \tag{1b}$$

The problem with these formulae is that intervals used in them contains negative values. Strictly speaking, this makes them *not* STL formulae. The two formulae are very similar to each other. Therefore, we only present results of monitoring Formula 1a. Formula 2a specifies the same requirement as in Formula 1a using a STL formula (intervals are all non-negative values). It intuitively says the following must be true at all points $r : \mathbb{R}_{\geq 0}$ in time: Either there must be a bolus somewhere during $(r, r + t_1 + t_2)$, or if this is not the case then at any point $r' : [r + t_1, r + t_1 + t_2)$ if a meal is taken then a bolus should be taken not later than t_2 units of time in the future. Note that we already know that no bolus is taken during interval of times $(r - t_1, r]$, therefore we only need to look into the future.

$$\Box_{[0,\infty)}\left(\begin{array}{l} \Diamond_{(0,t_1+t_2)}(B > c_2) \quad \vee \\ \Box_{[t_1,t_1+t_2)}\Big((M > c_1) \rightarrow \Diamond_{(0,t_2)}(B > c_2)\Big) \end{array}\right) \tag{2a}$$

We ran our algorithm on real medical data gathered from 61 patients. For each patient we have different number of observed signals that span over time from few hours to few days. Since the STL formula in Formula 2a is not currently supported by the implementation in [6], we have implemented a monitor for this formula ourselves. All the experiments were performed on a laptop with Intel i7 2.4GHz CPU, 8 GB of memory, and Kubuntu as its operating system.

Figure 1 shows robustness degrees of Formula 2a over time for $M > c_1$ and $B > c_2$ for one observed signal of a patient. The lower bound(a_1) on the robustness degree of the Formula 2a will always be $-\infty$ because we want it to be true always, even at all times in future. Similarly, eventually true property of the negation of Formula 2a will fix the upper bound(b_2) on its robustness degree to $+\infty$. So, in order to check for the termination condition of our algorithm, we

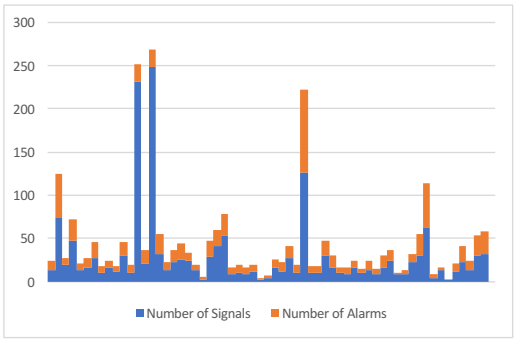

Figure 2: Number of sample executions and number of alarms for different patients

need to keep track of the upper bound(b_1) on the robustness degree of Formula 2a and lower bound(a_2) on the robustness degree of its negation. In this figure, red and blue lines represent b_1 and a_2 respectively. We can see that at $t = 421$, the patient had his/her first meal but did not inject bolus in the time window $391 \leq t \leq 451$. So, at t slightly greater than 451, a_2 becomes greater than δ(which is equal to 0.007) and the algorithm terminates by raising an alarm on rejection of the Formula 2a.

Figure 2 compares the number of alarms (*i.e.* number of times that the algorithm proves property specified in Formula 1a is false) that is raised by our algorithm for each patient with total number of signals(or observations) we have in our data for that patient. This figure is created when δ', an input to our algorithm, is set to 0.05. Smaller values of δ' increase number of raised alarms with the possibility that the algorithm in [26] fails. Similarly, larger values of δ' decreases number of raised alarms with the possibility of returning to many unknown answers.

Finally, the current implementation for the algorithm in [26] does not provide any probabilistic error guarantee (in particular it does not satisfy Assumption 1). Therefore, our current implementation does not provide any probabilistic error guarantee either. However, the current implementation in [26] guarantees being maximally invariant to nuisance parameters, and hence our current implementation provides the same guarantee as well. Note that, being invariant to nuisance parameters implies that the probability of returning an incorrect answer does not depend on value of nuisance parameters. Therefore, even when we don't provide probabilistic error guarantee, the actual probability of returning an incorrect answer does not change with nuisance parameters.

5 RELATED WORK

In the context of LTL and ω-regular languages, monitoring executions has been considered before in [2, 3, 5, 11]. Beside not being developed for real-time systems, none of these works consider the inability to fully observe system states at different points in time. Note that one of the algorithms developed in [11], is stochastic. But even in this algorithm, one knows exactly which atomic proposition is true and which one is not.

(Robust) monitoring MITL and STL properties have been studied in [6–8, 12, 16, 22] before. None of these works consider the effect of noise or nuisance parameters. [16] is one of the early papers that introduced STL in the context of monitoring temporal properties. The algorithm in this paper only supports bounded temporal operators and is suitable only for offline monitoring, meaning the whole execution should be at hand. In [6] authors assume signals remain constant between consecutive observations. In [8] authors assume signals are piecewise linear between consecutive observations. They also consider only offline monitoring. In [12, 22] authors consider only discrete signals (each event is time stamped). We could not directly use any of these papers instead of [6] in Assumption 2, because in these papers, at every step one has to specify which atomic proposition is true and which one is false. But since we cannot look at the system state directly, it is possible that at some step we won't know if an atomic proposition is true or false. Also, output of the algorithms in [12, 22] is binary true or false. But output of [6] is a robustness degree which is what we need in Theorem 2.5. The main benefit of [12, 22] comparing to [6, 8] is that the memory that is used by the algorithms in [12, 22] does not depend on the length of execution. This is exactly, why the algorithm in [6] does not fully support unbounded temporal operators. The algorithm developed in [7] can replace the algorithm in [6] that we use in our paper. However, the focus of [7] is on supporting the past time temporal operators when one only has bounded future time operators. Since results in [9, 10] do not consider past time temporal operators, Theorem 2.5 does not allow any such operator. Therefore, we decided to use the results from [6], since it partially supports unbounded future time temporal operators.

The inability to directly look at system state, has been considered in [20, 23, 24]. However, none of these works considered nuisance parameters. Furthermore, [20] is for synthesizing controllers and not monitoring them. Also, algorithms in [23, 24] are statistical verifiers, meaning they can sample from each point in time as many time as they want, which is never possible in monitoring.

6 CONCLUSION

In this paper we developed a test procedure for monitoring continuous time signals against STL properties. We proved if the test procedure used for atomic propositions is (maximally) invariant to nuisance parameters then our test procedure will also be (maximally) invariant to nuisance parameters. We also proved that as soon as the test procedure for atomic propositions fulfills its probabilistic error guarantee, our algorithm guarantees the probability of returning an incorrect answer can be made arbitrarily close to zero. We assumed there is a procedure for monitoring piecewise constant signals against STL properties. However, the input signal to our algorithm could be any continuous signal with a known Lipschitz constant.

There are at least two future directions for this work. The first one is to develop a method that can monitor Lipschitz continuity and adapt time steps accordingly. Furthermore, if this new monitor says the Lipschitz continuity is violated, we believe, there is no need to cancel the whole procedure and it would be enough to set the robustness degree of the corresponding atomic proposition at the corresponding time to zero. The second direction is to combine continuous time monitoring algorithms for MITL formulas with the

Algorithm 1: Parameter Invariant Monitoring Algorithm for Signal Temporal Logic

Input: system signal f, STL formula φ, error parameter $\alpha : (0,1)$, Lipschitz constant $\lambda : \mathbb{R}_+$, and indifference parameter $\delta' : \mathbb{R}_+$.
Output: accept, reject, or unknown

```
x ← nil   /* sequence of observations tagged with their times */
n ← 0                                   /* current step in monitoring */
δ, Δτ ← some values according to Theorem 2.5 using input λ
  /* an upper bound for δ and a lower bound for Δτ is enough */
Function observe(n : ℕ/* step number */, θ : Θ/* test func. */)
    /* returns τ(n) + an observation for θ(f(τ(n))) (from ℝ^{Υθ}) */
    while |x| ≤ n do                    /* |x| is the length of x */
        ν : ℝ^X
        t ← current time
        forall θ ← Θ do
            ν_{↾Υθ} ← current sensor value for θ /* sampling from
              Ω_{θ,f,t} */
        Add (t, ν) to the end of x
    (t, ν) ← the n^th element of x
    return (t, ν_{↾Υθ})
Function estimate()
    /* returned object maps θ to an estimation of θ(f(τ(n)))
       using Assumption 1                                    */
    α' ← α / (2^{n+1}|Θ|)
    ν : ℝ^Θ
    forall θ ← Θ do
        a ← 𝒜(observe, θ, α', δ')          /* 𝒜 is defined in
          Assumption 1 */
        if a − δ' > 0 then ν(θ) ← a − δ'
        else if a + δ' < 0 then ν(θ) ← a + δ'
        else ν(θ) ← 0
    (t, _) ← first element of x /* we won't use the 2nd element */
    /* next line is required for saving memory and correctness
       of observe function                                   */
    x ← remove the first element of x
    return (t, ν)
M1 ← initialize ℳ for φ^{2Δτ} /* ℳ is defined in Assumption 2 */
M2 ← initialize ℳ for (¬φ)^{2Δτ}
[a1, b1] ← [−∞, ∞]
[a2, b2] ← [−∞, ∞]
while a1 ≤ δ ∧ a2 ≤ δ ∧ (δ < b1 ∨ δ < b2) do
    (t, ν) ← estimate()
    [a1, b1] ← M1(ν, t)
    [a2, b2] ← M2(ν, t)
    n ← n + 1
if a1 > δ then return accept
if a2 > δ then return reject
return unknown
```

full support for unbounded time operators that only require constant amount of memory with respect to the execution length and monitoring algorithms for STL formulas that return the robustness degree instead of a binary/ternary value.

Acknowledgement. This work was supported in part by NSF CNS-1505799 and the Intel-NSF Partnership for Cyber-Physical Systems Security and Privacy and by ONR N000141712012.

REFERENCES

[1] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. 1996. The Benefits of Relaxing Punctuality. *J. ACM* 43, 1 (1996), 116–146.
[2] Andreas Bauer. 2010. Monitorability of omega-regular languages. *CoRR* abs/1006.3638 (2010). http://arxiv.org/abs/1006.3638
[3] Andreas Bauer and Yliès Falcone. 2016. Decentralised LTL monitoring. *Formal Methods in System Design* 48, 1 (2016), 46–93.
[4] Sanjian Chen, Matthew O'Kelly, James Weimer, Oleg Sokolsky, and Insup Lee. 2015. An Intraoperative Glucose Control Benchmark for Formal Verification. *IFAC-PapersOnLine* 48, 27 (2015), 211 – 217.
[5] Marcelo d'Amorim and Grigore Roşu. 2005. *Efficient Monitoring of ω-Languages.* 364–378.
[6] Jyotirmoy V. Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A. Seshia. 2015. *Robust Online Monitoring of Signal Temporal Logic.* 55–70.
[7] Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. 2014. *On-Line Monitoring for Temporal Logic Robustness.* 231–246.
[8] Alexandre Donzé, Thomas Ferrère, and Oded Maler. 2013. *Efficient Robust Monitoring for STL.* 264–279.
[9] Georgios E. Fainekos and George J. Pappas. 2007. *Robust Sampling for MITL Specifications.* Berlin, Heidelberg, 147–162.
[10] Georgios E. Fainekos and George J. Pappas. 2009. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science* 410, 42 (2009), 4262 – 4291.
[11] Kalpana Gondi, Yogeshkumar Patel, and A. Prasad Sistla. 2009. *Monitoring the Full Range of ω-Regular Properties of Stochastic Systems.* 105–119.
[12] Hsi-Ming Ho, Joël Ouaknine, and James Worrell. 2014. Online Monitoring of Metric Temporal Logic. In *Runtime Verification.* 178–192.
[13] Jinfeng Huang, Jeroen Voeten, and Marc Geilen. 2003. Real-time Property Preservation in Approximations of Timed Systems. In *MEMOCODE.* Washington, DC, USA, 163–171.
[14] R. Ivanov, J. Weimer, A. F. Simpao, M. A. Rehman, and I. Lee. 2016. Prediction of Critical Pulmonary Shunts in Infants. *IEEE Transactions on Control Systems Technology* 24, 6 (2016), 1936–1952.
[15] Ron Koymans. 1990. Specifying real-time properties with metric temporal logic. *Real-Time Systems* 2, 4 (1990), 255–299.
[16] Oded Maler and Dejan Nickovic. [n. d.]. *Monitoring Temporal Properties of Continuous Signals.*
[17] Joël Ouaknine and James Worrell. 2008. Some Recent Results in Metric Temporal Logic. In *FORMATS.* 1–13.
[18] Amir Pnueli. 1977. The Temporal Logic of Programs. In *SFCS.* 46–57.
[19] Nima Roohi and Mahesh Viswanathan. 2018. Revisiting MITL to Fix Decision Procedures. In *VMCAI.* Springer International Publishing, 474–494.
[20] Dorsa Sadigh and Ashish Kapoor. 2015. Safe Control under Uncertainty. *CoRR* abs/1510.07313 (2015).
[21] Sadahiro Saeki. 1996. A Proof of the Existence of Infinite Product Probability Measures. (1996).
[22] Prasanna Thati and Grigore RoÅ§u. 2005. Monitoring Algorithms for Metric Temporal Logic Specifications. *Electronic Notes in Theoretical Computer Science* 113 (2005), 145 – 162. Proceedings of the Fourth Workshop on Runtime Verification (RV 2004).
[23] Yu Wang, Nima Roohi, Matthew West, Mahesh Viswanathan, and Geir E. Dullerud. 2015. Statistical Verification of Dynamical Systems Using Set Oriented Methods. In *HSCC.* 169–178.
[24] Y. Wang, N. Roohi, M. West, M. Viswanathan, and G. E. Dullerud. 2016. Verifying Continuous-time Stochastic Hybrid Systems via Mori-Zwanzig model reduction. In *IEEE CDC.* 3012–3017.
[25] James Weimer, Sanjian Chen, Amy Peleckis, Michael R. Rickels, and Insup Lee. 2016. Physiology-Invariant Meal Detection for Type 1 Diabetes. *Diabetes Technology & Therapeutics* 18, 10 (2016), 616–624.
[26] J. Weimer, R. Ivanov, S. Chen, A. Roederer, O. Sokolsky, and I. Lee. 2017. Parameter-Invariant Monitor Design for Cyber Physical Systems. *Proc. IEEE* PP, 99 (2017), 1–22.