Measuring Inconsistency in STL Specifications

Carl Corea[1], Isabelle Kuhlmann[2], Karen Leung[3], John Grant[4], and Matthias Thimm[2]

[1] Research Group Process Science, University of Koblenz, Germany
ccorea@uni-koblenz.de
[2] Artificial Intelligence Group, University of Hagen, Germany
isabelle.kuhlmann@fernuni-hagen.de, matthias.thimm@fernuni-hagen.de
[3] Department of Aeronautics and Astronautics, University of Washington, USA
kymleung@uw.edu
[4] University of Maryland, College Park, USA
grant@cs.umd.edu

Abstract. Declarative specifications provide a formal means to define process constraints, yet integrating specifications created independently by different stakeholders can introduce conflicting requirements. In this work, we address the problem of *measuring* inconsistency in declarative process specifications expressed using Signal Temporal Logic (STL) (intuitively, an extension of LTL that allows also for *real-timed* and *real-valued* constraints). As we will show, existing approaches to STL satisfiability checking cannot provide any meaningful diagnosis, as they assess inconsistency only in a binary manner. We therefore propose a novel framework for measuring the *degree* of inconsistency in STL specifications, which allows for a fine-grained analysis. Our approach captures both the temporal extent and the numerical magnitude of conflicting constraints, enabling a quantitative notion of inconsistency severity.

Keywords: Declarative Specifications · Signal Temporal Logic (STL) · Inconsistency Measurement · IoT-Aware BPM

1 Introduction

In traditional settings of Business Process Management (BPM), process behaviors are often assumed to be of a discrete nature, e.g., a sequence of activities. However, in novel BPM settings—such as IoT-aware processes and Cyber-Physical Systems (CPS)—process behaviors may also be increasingly driven by *continuous* dynamics, such as sensor data or the pose/orientation of physical objects moving through space and time [3, 6, 11, 13]. In turn, modern processes may also involve continuous behaviors, such as real-valued signals [4, 6].

To regulate such (continuous) behaviors, systems are commonly specified using declarative constraints, i.e., logical formulas that describe what must (or must not) hold over time. For instance, constraints may specify a frame of allowed behavior for autonomous agents or robots [2]. Notably, as constraints in such settings may involve *real-timed* and *real-valued* aspects, newer constraint

languages such as Signal Temporal Logic (STL) [4, 8] have evolved to capture these aspects, which we will consider in this work. For example, thinking of a temperature sensor, an STL constraint over a sensor signal could be:

$\mathbf{G}_{[0,100]}(temp \leq 30)$ *("for the first 100 seconds, the temp. will not exceed 30").*

The purpose of declarative specifications—typically given as a set of formulas—is to define the admissible space of process behaviors. However, a problem arises when these specifications are *inconsistent*, i.e., when they contain contradictory constraints, as in these situations, no possible execution can satisfy all constraints simultaneously. Such inconsistency may arise particularly in collaborative or multi-agent settings, where several agents or companies need to collaborate but may have jointly unsatisfiable process requirements, which needs to be resolved.

While recent work [1, 15] has focused on approaches to verify (in a binary manner) whether STL specifications are consistent, those works cannot look "into" those specifications to provide further diagnosis. Such insights would however be needed for a number of use-cases, including, but not limited to:

1. understanding *why* a specification is inconsistent (e.g., for debugging and root-cause analysis)
2. understanding the *distance* to a consistent specification (e.g., for understanding how hard it will be to repair)
3. in agentic BPM settings, given two agents A, B constrained by specifications K_A, K_B, assessing the level of *agreement* or cooperation feasibility (i.e., assessing the inconsistency of $K_A \cup K_B$)

To address these issues, we develop novel techniques for *measuring* a degree of inconsistency in STL specifications. To do so, we extend results from the field of *knowledge representation* (KR), which is—among other topics—concerned with the analysis and resolution of inconsistency in sets of formulas. Importantly, the (quantitative) techniques developed in this work are not to be confused with the 'quantitative semantics' (robustness) for STL formulas as introduced by Donzé et al. [9]. Particularly, robustness quantifies the degree to which a concrete execution (or trace) satisfies an STL formula (trace $\leftrightarrow$ formula), whereas the approaches developed in this work measure the severity of inconsistency *between* conflicting STL formulas themselves (formula $\leftrightarrow$ formula). The approach presented in this work thus represents a novel angle on the management of inconsistent STL specifications. Here, the contributions of this paper are as follows:

- We present a novel **interval-based semantics** for STL, which allows us to define (and reason over) models with uncertainty, especially in cases where (inconsistent) specifications have no classical models that satisfy them.
- We characterize quantitative notions of inconsistency in STL by means of rationality postulates and develop novel **inconsistency measures** for STL.

We continue with a running example and preliminaries (Sections 2, 3). Then, we develop our novel approach in Section 4 and conclude in Section 5.

2 Running Example

To illustrate the need for measuring inconsistency, we now introduce a running example used throughout the paper.

In this work, we interpret STL specifications as declarative knowledge bases encoding admissible process behaviors over continuous signals. For example, in cyber-physical business processes, STL specifications may capture local process requirements specified by different stakeholders. Here, since the requirements are often developed independently and later integrated, inconsistencies may arise.

As an example, consider an autonomous warehouse robot responsible for transporting goods. Let a denote the robot's longitudinal acceleration. The logistics department may require the robot to accelerate quickly to ensure timely deliveries, while the safety department may require the robot to avoid strong acceleration in areas where human workers are present. If these independently specified requirements are combined without reconciliation, conflicting STL specifications can result, where the conflict reflects jointly unsatisfiable constraint about system evolution. For example, consider the following sets of STL formulas $\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3$ (we will formalize syntax and semantics later):

$$\mathcal{K}_1 = \{\mathbf{G}_{[0,5]}(a > 1), \mathbf{G}_{[0,5]}(a < -1)\}$$
$$\mathcal{K}_2 = \{\mathbf{G}_{[0,10]}(a > 1), \mathbf{G}_{[0,10]}(a < -1)\}$$
$$\mathcal{K}_3 = \{\mathbf{G}_{[0,5]}(a > 10), \mathbf{G}_{[0,5]}(a < -10)\}$$

All three sets are inconsistent, as they simultaneously require the robot's acceleration a to be greater than a positive threshold and smaller than a negative threshold (over some time intervals 0-5 or 0-10). However, $\mathcal{K}_2$ seems more inconsistent than $\mathcal{K}_1$ (more time is affected), and $\mathcal{K}_3$ seems more inconsistent than $\mathcal{K}_1$ (the disagreement over what a should be is larger, even though the same time is affected). Also, it is not intuitive what is "more" inconsistent when considering $\mathcal{K}_2$ vs. $\mathcal{K}_3$. These differences point towards some inherent notion of a *severity* of inconsistency and thus call for means to quantitatively assess this notion.

3 Preliminaries

STL specifications. As declarative specifications in novel BPM settings (such as CPS or IOT-aware BPM) may involve *real-timed* and *real-valued* aspects, more expressive languages such as STL have been adapted [4], as the traditional languages such as LTL(f) may not be able to address these aspects. Here, STL defines linear-time properties of real-valued, real-timed process behaviors. For this, let A be a set of state variables over a time domain T (with T of form $[0, m] \subseteq \mathbb{R}$, where m is the last considered instant in time).

An STL formula φ is then defined via the following grammar:

$$\varphi ::= \mathbf{true} \mid a \bowtie k \mid (\neg\varphi) \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1 \, \mathbf{U}_I \, \varphi_2) \tag{1}$$

where $a \in A$, $\bowtie \in \{<, \leq, =, \geq, >\}$, k is a constant in $\mathbb{R}$, and I is a time interval in T with endpoints in $\mathbb{N}$ that may but need not be included. The term $a \bowtie k$

is used to define real-valued conditions over individual variables, e.g., "a must be larger than 100". The interval I is used to express time bounds of the form $[l, r]$ and will be explained below.

Remark 1. For any interval $I = [l, r]$ we assume that $0 \leq l < r \leq m$.

Remark 2. We allow the interval $[0, m]$ ("until the last instant in time") and drop the interval for readability, e.g., $\mathbf{U} \equiv \mathbf{U}_{[0,m]}$.

W.r.t. the above grammar, we can define the usual boolean connectives $(\vee, \rightarrow)$ and further abbreviations, in particular $\mathbf{F}$ (Future) with $\mathbf{F}_I\varphi \equiv \mathbf{true}\ \mathbf{U}_I\varphi$, and $\mathbf{G}$ (Globally) with $\mathbf{G}_I \equiv \neg\mathbf{F}_I\neg\varphi$.

An STL valuation $\omega : A \times T \rightarrow \mathbb{Q}$ is a function that assigns to each state variable and time point a numerical value (visually, a valuation ω can be thought of as a signal over time). The satisfaction of an STL formula ϕ by a valuation ω at time t is denoted as $\omega, t \models \phi$. Satisfaction is inductively defined as follows:

$$
\begin{aligned}
&\omega, t \models \mathbf{true} \\
&\omega, t \models a \bowtie k \text{ iff } \omega(a, t) \bowtie k \\
&\omega, t \models \neg\varphi \text{ iff } \omega, t \not\models \varphi \\
&\omega, t \models \varphi_1 \wedge \phi_2 \text{ iff } \omega, t \models \varphi_1 \text{ and } \omega, t \models \varphi_2 \\
&\omega, t \models \varphi_1 \mathbf{U}_{[l,r]} \varphi_2 \text{ iff } t + r \leq m \text{ and} \\
&\qquad\qquad \exists t' \in\ (t + l, t + r] \text{ s.t. } \omega, t' \models \varphi_2 \text{ and} \\
&\qquad\qquad\quad \forall t'' \in [t, t')\ : \omega, t'' \models \varphi_1
\end{aligned}
$$

The evaluation of $\mathbf{U}$ allows us to specify a (strong) "until" condition with a given time-frame, in the sense that $\varphi_1 \mathbf{U}_I \varphi_2$ needs to hold only within the interval I. Note that the shown semantics for intervals is meant for cases where the l, r of the interval are provided as numerical values. If the shorthand notation [0,m] is used (see Remark 2), the semantics is to be understood s.t. t' is anywhere between $[t, m]$. We say that a valuation ω satisfies a formula ϕ, denoted $\omega \models \phi$, iff ϕ is true at time 0 (i.e., $\omega \models \phi \Leftrightarrow \omega, 0 \models \phi$). An example for satisfaction is shown in Figure 1, which shows a valuation ω_1, represented as a signal, for a state variable a, where ω_1 satisfies $\mathbf{G}(a < 5)$. Further, a valuation ω satisfies a *set* of STL formulas Φ iff $\omega \models \phi$ for all $\phi \in \Phi$. A set of STL formulas is consistent iff there exists a valuation that satisfies it. Otherwise, it is inconsistent. We refer to a set of STL formulas as a knowledge base (KB) $\mathcal{K}$, and denote by $\mathsf{vars}(\mathcal{K})$ the set of state variables of $\mathcal{K}$. We denote by $\mathbb{K}$ the universe of all such KBs.

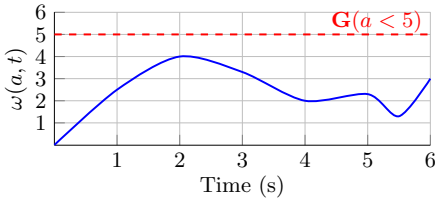


Fig. 1: Exemplary valuation for a state variable a (blue) over time, satisfying the formula $\mathbf{G}(a < 5)$.

Inconsistency Measurement. As stated, an STL knowledge base is inconsistent iff there is no valuation that can satisfy it (examples being $\mathcal{K}_1 - \mathcal{K}_3$ from the running example, where no valuation satisfies all the constraints). However, this notion of inconsistency is a binary concept ("consistent" or "inconsistent"), meaning that the severity of disagreement in $\mathcal{K}_1 - \mathcal{K}_3$ cannot be distinguished on this level. To provide a more fine-grained analysis, the field of *inconsistency measurement* [16] has evolved, which studies quantitative measures for assessing the degree of inconsistency. Intuitively, the measure value 0 means that the KB is consistent, and higher values reflect higher degrees of inconsistency. Let $\mathbb{R}^{\infty}_{\geq 0}$ be the set of non-negative real values including ∞. Then, an inconsistency measure is defined as follows.

Definition 1. *An inconsistency measure $\mathcal{I}$ is a function $\mathcal{I} : \mathbb{K} \to \mathbb{R}^{\infty}_{\geq 0}$ such that $\mathcal{I}(\mathcal{K}) = 0$ if and only if $\mathcal{K}$ is consistent.*

Existing measures from the inconsistency measurement literature are mostly syntactic. For example, common approaches to quantify the severity of inconsistency count the number of minimal inconsistent subsets[5] of the KB (denoted as the measure $\mathcal{I}_{\mathsf{MI}}$), or count the number of distinct formulas involved in the inconsistencies (i.e., that are problematic; denoted as the measure $\mathcal{I}_p$). This limits the application to our use-case, as we motivate in the following.

Related Work and Motivation. The problem of detecting inconsistency in temporal specifications is a fundamental topic in AI and declarative process modeling [7, 10]. In the context of STL, [1, 14, 15] have developed procedures for deciding the satisfiability of STL specifications. However, these approaches assess inconsistency in a purely binary manner (satisfiable or unsatisfiable). Thus they cannot "look into" inconsistent specifications to diagnose the structure or severity of the conflict. However, such diagnostic insight is crucial in practice, where modelers may need to assess the root-causes of inconsistency in the scope of re-modeling. A solely binary answer is therefore insufficient here, motivating *quantitative* notions of inconsistency for STL. In this regard, there are two related streams of research.

First, a prior work on inconsistency measurement in LTL [5] has proposed measures that quantify the number of affected time points in discrete temporal specifications. While this represents an important step toward quantitative inconsistency analysis, the approach is inherently restricted to LTL and discrete time. Consequently, it cannot capture the real-timed and real-valued aspects that are central to STL. In particular, it lacks the ability to reflect differences in the duration of continuous time intervals and the magnitude of numerical conflicts between real-valued constraints, which would be necessary for applications in settings such as IoT-aware business processes, where constraints naturally involve continuous time and sensor-based real values.

Second, the broader field of inconsistency measurement [16] has developed a variety of quantitative measures for logical KBs. However, most of these ap-

[5] W.r.t. a KB $\mathcal{K}$, a set $M \subseteq \mathcal{K}$ is called a *minimal inconsistent subset* of $\mathcal{K}$ if $M \models \bot$ and there is no $M' \subset M$ with $M' \models \bot$.

proaches are syntactic in nature, such as the ones mentioned after Definition 1. Intuitively, existing syntactic measures are not suitable in our setting, as they cannot capture the temporal operators and numerical values of STL. For example, recalling the KBs $\mathcal{K}_1 - \mathcal{K}_3$, we get $\mathcal{I}_p(\mathcal{K}_1) = \mathcal{I}_p(\mathcal{K}_2) = \mathcal{I}_p(\mathcal{K}_3) = 2$ for all sets (as there are two problematic formulas each), so these measures cannot distinguish the conflicts, although there are clear differences in affected time and magnitudes between the expected values.

Consequently, as existing approaches—both from STL satisfiability checking and standard inconsistency measurement—cannot distinguish problems as considered in this work, new approaches are needed to measure inconsistency in STL, which we present in the following.

4 Quantitative Inconsistency in STL

4.1 Towards Quantitative Notions of Inconsistency in STL Specifications

To characterize the notion of inconsistency in STL specifications, we start by defining some desirable properties for the envisaged inconsistency measures. For this, we fix an STL inconsistency measure $\mathcal{I}$. In a previous work on measuring inconsistency in LTL [5], the authors proposed the property of *time sensitivity*, which states that any inconsistency measure $\mathcal{I}$ for LTL should be able to distinguish the number of affected time values[6]:

Time Sensitivity **(TS)** For an LTL formula φ, $\mathcal{I}(\{\mathbf{G}\varphi, \mathbf{G}\neg\varphi\}) > \mathcal{I}(\{\mathbf{X}\varphi, \mathbf{X}\neg\varphi\})$.

As discussed in [5], for temporal specifications, inconsistency measures should ideally be time sensitive in order to reflect temporal operators in the characterization of inconsistency. We therefore lift this postulate (and others) to STL.

We begin with time-sensitivity, where, for our setting, we move from discrete steps to time intervals. Let $|I|$ denote the duration of a time interval $I \subseteq \mathbb{R}_{\geq 0}$.

Time Sensitivity' **(TS-STL)** For an STL formula φ and two intervals I_1, I_2 with $|I_1| < |I_2|$: $\mathcal{I}(\{\mathbf{G}_{I_2}\varphi, \mathbf{G}_{I_2}\neg\varphi\}) > \mathcal{I}(\{\mathbf{G}_{I_1}\varphi, \mathbf{G}_{I_1}\neg\varphi\})$.

In other words, we consider an inconsistency measure to be (real-)time sensitive iff contradictions persisting over longer time intervals are valued as more severe.

Next to the aspect of time, a novel aspect in STL is that formulas are real-valued as well. In this sense, an inconsistency measure for STL should also be able to reflect this aspect. We define this via the following property:

Value (Magnitude) Sensitivity **(VS)** For a state variable a and the real numbers c_1, c_2, d_1, d_2 such that $c_1 > c_2$, $d_1 > d_2$, and $|c_1 - c_2| < |d_1 - d_2|$: $\mathcal{I}(\{\mathbf{G}(a > d_1), \mathbf{G}(a < d_2)\}) > \mathcal{I}(\{\mathbf{G}(a > c_1), \mathbf{G}(a < c_2)\})$.

[6] Recall that LTL is defined over a linear sequence of (discrete) time points, and $\mathbf{X}$ refers to the neXt point in time.

In other words, an inconsistency measure is value sensitive iff larger numerical separation between contradictory bounds are considered as more severe.

As discussed in Section 3, existing *syntactic* measures from the inconsistency measurement literature are neither time nor value sensitive (see the example in Sec. 3). Also, the LTL-inconsistency measure proposed in [5] is not defined over STL, so it is not (real-)time sensitive as well (TS-STL) (and also not value sensitive). In turn, new inconsistency measures for STL are needed that satisfy TS-STL and VS, which we develop in the following.

4.2 Expressing Uncertainty with Interval-Based Valuations

In the examples of inconsistent STL specifications as presented before, it is clear that no classical STL valuation $\omega : A \times T \to \mathbb{Q}$ satisfies all formulas of the specifications. Therefore, we present a novel approach that defines valuations not as single numerical values but as *ranges* of (possible) values. Concretely, we define interval-based valuations ν, that assign to every state variable, at each time point, an *interval*, with the intuition that this interval represents a range, or uncertainty, of the concrete numerical value of the state variable at that time.

Definition 2 (Interval-Based Valuation). *Let A be the set of all state variables, $T \subseteq \mathbb{R}$ be the time domain as before, and let $\mathfrak{Int}$ be the set of all intervals $\subset \mathbb{R}$—of form $[l, r]$—with endpoints in $\mathbb{N}, l \leq r$. Then, an interval-based valuation is defined as a function $\nu : A \times T \to \mathfrak{Int}$ that assigns a (possible) range of values to a state variable a at every time t.*

For an interval-based valuation ν, we define $X_\nu = \{\omega \mid \forall a, t : \ \omega(a,t) \in \nu(a,t)\}$ (in other words, a set of classical valuations over all elements of the assigned range). For an interval-based valuation ν, a formula ϕ and a time point t we write $X_\nu, t \models \phi$ if there is $\omega \in X_\nu$ with $\omega, t \models \phi$ (and $X_\nu \models \phi$ iff $X_\nu, 0 \models \phi$).

Example 1. Consider the formula $a < 1$ and the valuations $\nu_1 - \nu_4$, with

$$\nu_1(a,0) = [-2,0], \quad \nu_2(a,0) = [0,0], \quad \nu_3(a,0) = [1,2], \quad \nu_4(a,0) = [-2,2].$$

First, observe that the corresponding sets $X_{\nu_1} - X_{\nu_4}$ are *sets* of classical valuations $\omega_i : A \times T \to \mathbb{Q}$ over all elements of the respective intervals from $\nu_1 - \nu_4$. For example, X_{ν_1} is a set of valuations $\{\omega \mid \omega(a,0) \in [-2,0]\}$, and X_{ν_2} is a set containing only one valuation $\omega(a,0) = 0$ (i.e., over [0,0]). Recalling the formula $a < 1$, we then have that

1. $X_{\nu_1} \models a < 1$ (as actually all elements in X_{ν_1} satisfy this formula)
2. $X_{\nu_2} \models a < 1$ (analogously)
3. $X_{\nu_3} \not\models a < 1$ (as there is no element in X_{ν_3} that satisfies this formula)
4. $X_{\nu_4} \models a < 1$ (as there exist some elements in X_{ν_4} that satisfy this formula)

We say that an interval-based interpretation ν satisfies an STL formula ϕ (denoted $\nu \models \phi$) iff $X_\nu \models \phi$. Furthermore, an interval-based valuation satisfies a *set* of formulas $\mathcal{K}$ iff $\nu \models \phi$ for all $\phi \in \mathcal{K}$ (denoted $\nu \models \mathcal{K}$). In this case, we say that ν is an interval-based model of $\mathcal{K}$.

The advantage of interval-based models is that they compactly represent the possible space of signal values over time. Rather than committing to a single value, an interval-based model describes a set X_ν of classical valuations that remain possible and thus captures the uncertainty about the concrete execution.

This is particularly useful in the presence of inconsistency. If a knowledge base K is inconsistent, no classical valuation ω exists such that $\omega \models K$, and classical semantics yields no further structural insight. In contrast, an interval-based model may still exist and captures the space of valuations that approximate K as closely as possible. In this case, the assigned intervals necessarily reflect conflicting requirements, i.e., for every $\omega \in X_\nu$ at least one formula in K is violated at some time point. Hence, interval-based models retain information about the conflict rather than collapsing to the absence of a (classical) model.

Example 2. Consider $\mathcal{K}_4 = \{a < -1, a > 1\}$ and an interval-based valuation ν with $\nu(a, 0) = [-2, 2]$. Then, the set X_ν contains valuations ω_i, ω_j s.t. $\omega_i \models a < -1$ and $\omega_j \models a > 1$. Hence, $\nu \models \mathcal{K}_4$ as $X_\nu \models \phi$ for all $\phi \in \mathcal{K}_4$.

Since interval-based models retain information about the conflict (instead of collapsing to the absence of a classical model), this information can be leveraged to quantify a degree of inconsistency, which we introduce in the following.

4.3 Measuring Inconsistency in STL

As motivated in Section 4.1, there can be several dimensions of inconsistency in STL (namely time and values), and both of these dimensions capture different aspects of conflicts among constraints. We begin by considering the *time* dimension. For this, we need some further notation.

Definition 3. *For an STL knowledge base $\mathcal{K}$ and an interval-based valuation ν,* $\mathsf{AffectedTime}_{\mathcal{K}}(\nu) = \{t \mid \exists \phi \in \mathcal{K}\ s.t.\ \exists \omega \in X_\nu : \omega, t \not\models \phi\}$.

Intuitively, $\mathsf{AffectedTime}_{\mathcal{K}}(\nu)$ collects all time points at which at least one formula of $\mathcal{K}$ is violated by some valuation contained in X_ν. These are precisely the time instants at which the specification exhibits conflicting requirements relative to the uncertainty described by ν. For example, for $\mathcal{K}_4$ from Example 2 and the interval $\nu(a, 0) = [-2, 2]$, at time point 0, one of the (classical) valuations $\omega \in X_\nu$ will always violate one of the constraints in $\mathcal{K}_4$ (as there can be no classical valuation that satisfies both formulas at once, hence, this time point is "affected"). If no time points are affected by conflicts (i.e., $\mathcal{K}$ is consistent), $\mathsf{AffectedTime}$ will simply be the empty set.

Note that $\mathsf{AffectedTime}_{\mathcal{K}}(\nu)$ is a subset of the (possibly continuous) time domain $T \subseteq \mathbb{R}_{\geq 0}$. In general, this set decomposes into a finite union of intervals but may also contain isolated time points. Ignoring time points for now, to quantify the temporal extent of the conflict, we employ the *Lebesgue measure* [12]. For an interval $[l, r]$, the Lebesgue measure μ is such that $\mu([l, r]) = r - l$, with $\mu(A1 \cup A2) = \mu(A1) + \mu(A2)$ whenever $A1, A2$ are disjoint.

Example 3. $\mu([0, 2] \cup [4, 5.5]) = (2 - 0) + (5.5 - 4) = 3.5$

The Lebesgue measure provides the total length of the affected periods of time. At any isolated time point, the Lebesgue measure is 0. In order not to miss any inconsistencies, on account of Definition 1 we must assign a positive value even if the inconsistency occurs at an isolated time point. In such a case we assign $\mu([t,t]) = \epsilon$ where we think of epsilon either as a very small unspecified positive number or as an infinitesimal. This is important to warrant that μ can distinguish between consistent and inconsistent KBs, even if inconsistency arises only at an isolated time point. Likewise, we define $\mu(\emptyset) = 0$ (for cases where no time is affected). We then leverage μ to define a time-based inconsistency measure over continuous time domains.

Definition 4 (Time-Based Inconsistency Measure). *Let $\mathcal{K}$ be a set of formulas. Then, define $\mathcal{I}_T$ via $\mathcal{I}_T(\mathcal{K}) = \min_{\nu \models K} \mu(\mathsf{AffectedTime}_K(\nu))$, if there is ν with $\nu \models \mathcal{K}$, and $\mathcal{I}_T(\mathcal{K}) = \infty$ otherwise.*

Example 4. Recall $\mathcal{K}_1 - \mathcal{K}_3$ with

$$\mathcal{K}_1 = \{\mathbf{G}_{[0,5]} a > 1, \mathbf{G}_{[0,5]} a < -1\}$$
$$\mathcal{K}_2 = \{\mathbf{G}_{[0,10]} a > 1, \mathbf{G}_{[0,10]} a < -1\}$$
$$\mathcal{K}_3 = \{\mathbf{G}_{[0,5]} a > 10, \mathbf{G}_{[0,5]} a < -10\}$$

Then we have $\mathcal{I}_T(\mathcal{K}_1) = 5$, $\mathcal{I}_T(\mathcal{K}_2) = 10$, and $\mathcal{I}_T(\mathcal{K}_3) = 5$.

For $\mathcal{K}_1$ we have that in the time interval [0,5] any interval-based valuation ν has to assign (to the state variable a) an interval such as [-2,2] in order for both formulas in $\mathcal{K}_1$ to be satisfied. Conversely, X_ν will contain a valuation ω that violates one of the constraints in $\mathcal{K}_1$ in this time interval, thus, the lower bound for an affected time period is the interval [0,5], with $\mu([0,5]) = 5 - 0 = 5$. The minimal affected time for $\mathcal{K}_2, \mathcal{K}_3$ behaves analogously.

To clarify, the $\mathcal{I}_T$ measure quantifies inconsistency as the minimal total duration of time points at which conflicts necessarily occur ("minimal" over all valuations ν that satisfy the KB). As such, $\mathcal{I}_T$ is time sensitive because contradictions persisting over longer intervals yield larger measure values.

Proposition 1. *$\mathcal{I}_T$ satisfies TS-STL.*

Proof. Let φ be an STL formula, $\mathcal{K} = \{\boldsymbol{G}_{I_2}\varphi, \boldsymbol{G}_{I_2}\neg\varphi\}$ and $\mathcal{K}' = \{\boldsymbol{G}_{I_1}\varphi, \boldsymbol{G}_{I_1}\neg\varphi\}$, with $|I_1| < |I_2|$. As I_2 has a longer duration, for any interval-based valuation ν (satisfying $\mathcal{K}, \mathcal{K}'$) it holds that $\mu(\mathsf{AffectedTime}_{\mathcal{K}}(\nu)) > \mu(\mathsf{AffectedTime}_{\mathcal{K}'}(\nu))$ per definition of μ (even in case I_1 is only an individual time point, $|I_2| > |I_1|$ per assumption so again μ will return a greater value here). So $\mathcal{I}_T(\mathcal{K}) > \mathcal{I}_T(\mathcal{K}')$, hence, $\mathcal{I}_T$ satisfies TS-STL.

By the definition of μ, it is important to note that $\mathcal{I}_T$ also correctly distinguishes consistent from inconsistent KBs even if the inconsistency occurs at an isolated point in time (i.e., for such inconsistencies, a value of $\epsilon > 0$ is assigned).

Next to the time dimension, the *real-valued* nature of STL constraints also means that the inconsistency may be different with respect to the magnitude

of the numerical disagreement. We therefore develop measures that can assess inconsistency in this dimension also. For this purpose, we consider some further notation on properties of the assigned intervals of interval-based valuations.

Definition 5 (Delta of an assigned interval). *Let ν be an interval-based valuation, a a state variable, and t a time point. For any $\nu(a,t) = [l,r]$, define the delta of this interval as $\Delta_\nu(a,t) = |r-l|$.*

Definition 6 (Delta-Minimality). *Let $\mathcal{K}$ be a knowledge base and ν any interval-based valuation such that $\nu \models \mathcal{K}$. We say that ν is delta-minimal iff for every state variable a and every time point t, there exists no interval $I = [l', r']$ with $|r' - l'| < \Delta_\nu(a,t)$ such that for the valuation ν', obtained from ν by replacing only $\nu(a,t)$ by I, we would have that $\nu' \models \mathcal{K}$. Let $\mathsf{DeltaMin}(\mathcal{K})$ denote the set of all such delta-minimal valuations.*

Example 5. Consider the KB $\mathcal{K}_4 = \{a > 1, a < -1\}$, defined in Example 2, and the valuations $\nu_1 - \nu_3$, with $\nu_1(a,0) = [-1,1]$, $\nu_2(a,0) = [-2,2]$, and $\nu_3(a,0) = [-3,3]$. ν_1 does not satisfy $\mathcal{K}_4$ in any case. $\nu_2 \models \mathcal{K}_4$ and is delta-minimal, as there is no smaller interval for $(a,0)$ (with endpoints in $\mathbb{N}$) that could be assigned to still satisfy both formulas. Conversely, ν_3 satisfies $\mathcal{K}_4$ but is not delta-minimal as $\nu_2 \models \mathcal{K}_4$ but the interval $\nu_2(a,0)$ has a smaller delta.

To clarify, for a KB $\mathcal{K}$, an interval-based valuation ν is delta-minimal iff none of its intervals can be made strictly smaller without causing $\nu \not\models \mathcal{K}$.

We will now define a new inconsistency measure based on delta-minimal valuations. Essentially, this measure aims to calculate the "area" affected by the inconsistency, which is the total magnitude of the numerical disagreements of constraints, over all affected times. To correctly consider inconsistencies at isolated time points, we first decompose the set of affected times as $\mathsf{AffectedTime}_K(\nu) = \mathsf{A}_c \cup \mathsf{A}_d$, where A_c is the union of intervals and A_d is a finite set of isolated time points. We are now ready to define the area-based inconsistency measure.

Definition 7 (Area-Based Inconsistency Measure). *Let $\mathcal{K}$ be a knowledge base. Then, define $\mathcal{I}_{Area}$ via*

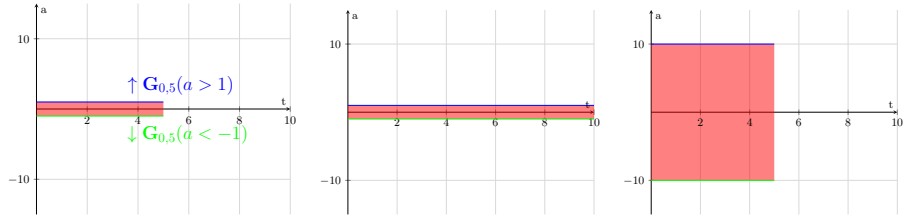
$$\mathcal{I}_{Area}(\mathcal{K}) = \min_{\nu \in \mathsf{DeltaMin}(\mathcal{K})} \left(\int_{A_c} \Big(\sum_{a \in \mathsf{vars}(\mathcal{K})} \Delta_\nu(a,t) \Big) dt \; + \; \epsilon \sum_{t \in A_d} \sum_{a \in \mathsf{vars}(\mathcal{K})} \Delta_\nu(a,t) \right)$$

if there is ν with $\nu \models \mathcal{K}$, and $\mathcal{I}_{Area}(\mathcal{K}) = \infty$ otherwise.

Example 6. Recall $\mathcal{K}_1 - \mathcal{K}_3$. For $\mathcal{K}_1, \mathcal{K}_3$ there are 5 affected times and for $\mathcal{K}_2$ there are 10 affected times (see also $\mathcal{I}_T$). Considering delta-minimal valuations (i.e., for $\mathcal{K}_1, \mathcal{K}_2$ with intervals $[-2,2]$ over all affected times (with delta of 4), and for $\mathcal{K}_3$ an interval $[-11,11]$ over affected times (delta of 22)) we have that

$$\mathcal{I}_{Area}(\mathcal{K}_1) = \int_{[0,5]} 4 \, dt = 4 \cdot (5-0) = 20$$
$$\mathcal{I}_{Area}(\mathcal{K}_2) = ... = 4 \cdot (10-0) = 40$$
$$\mathcal{I}_{Area}(\mathcal{K}_3) = ... = 22 \cdot (5-0) = 110$$

To clarify, $\mathcal{I}_{Area}$ integrates over the affected time the (minimal) deltas of the required value intervals. The combined value of $\mathcal{I}_{Area}$ can thus be interpreted as the minimal accumulated "area" of conflict (over time · magnitude). An example of this is shown in Figure 2. The green and blue lines are the (contradictory) bounds by the respective constraints in $\mathcal{K}_1 - \mathcal{K}_3$, and the red area is the area affected by inconsistency. As can be seen, although $\mathcal{K}_1$ and $\mathcal{K}_3$ affect the same time interval, the required value range w.r.t. the constraints in $\mathcal{K}_3$ is substantially larger, resulting in a greater overall area. Likewise, while $\mathcal{K}_2$ and $\mathcal{K}_1$ require identical value range in the assigned intervals, the affected area for the inconsistency in $\mathcal{K}_2$ is larger due to the longer duration of the conflict.



(a) $\mathcal{K}_1$ (b) $\mathcal{K}_2$ (c) $\mathcal{K}_3$

Fig. 2: Vizualization of inconsistencies for the example, with the corresponding contradictory bounds (blue/green), and the affected area of conflict (red).

Proposition 2. *$\mathcal{I}_{Area}$ satisfies VS.*

Proof. Let $\mathcal{K} = \{\boldsymbol{G}(a > d_1), \boldsymbol{G}(a < d_2)\}$ and $\mathcal{K}' = \{\boldsymbol{G}(a > c_1), \boldsymbol{G}(a < c_2)\}$ with $c_1 > c_2$, $d_1 > d_2$, and $|c_1 - c_2| < |d_1 - d_2|$. For any interval-based valuation ν (satisfying $\mathcal{K}$, $\mathcal{K}'$), the assigned interval for state variable a at any time point must cover both bounds imposed by the formulas. Hence, for $\mathcal{K}$ the minimal interval must contain $[d_2, d_1]$, yielding $\Delta_\nu(a,t) \geq |d_1 - d_2|$, while for $\mathcal{K}'$ it must contain $[c_2, c_1]$, yielding $\Delta_\nu(a,t) \geq |c_1 - c_2|$. Since $|d_1 - d_2| > |c_1 - c_2|$ by assumption, every delta-minimal valuation for $\mathcal{K}$ assigns strictly larger interval width than for $\mathcal{K}'$ at all affected time points. As $\mathcal{I}_{Area}$ integrates these widths over the affected time, it follows that $\mathcal{I}_{Area}(\mathcal{K}) > \mathcal{I}_{Area}(\mathcal{K}')$. Hence, $\mathcal{I}_{Area}$ satisfies VS.

5 Conclusions

In this paper, we introduced a novel framework for measuring inconsistency in STL specifications. Here, the introduced measures can be seen as complementary means, characterizing different aspects of inconsistency. Both introduced measures solve the shortcomings of existing measures from the literature (cf. Section 3), namely, that those are insensitive to temporal operators and real-valued bounds.

At this stage, the two newly introduced measures have been implemented by means of Answer Set Programming and applied to benchmark datasets to

demonstrate general feasibility.[7] As future work, we plan to extend on a comprehensive experimental evaluation to assess the practical feasibility and scalability of the approach. Also, we will develop element-based inconsistency measures, which will allow to provide explanations on the inconsistency on an constraint-level. Furthermore, we plan to exploit inconsistency measures for diagnosis, repair, and synthesis under conflicting specifications, enabling systems to reason about inconsistency during plan generation and adaptation.

References

1. Bae, K., Lee, J.: Bounded model checking of signal temporal logic properties using syntactic separation. Proc. of the ACM on Programming Languages **3**, 1–30 (2019)
2. Calvanese, D., Casciani, A., De Giacomo, G., Dumas, M., Fournier, F., Kampik, T., La Malfa, E., Limonad, L., Marrella, A., Metzger, A., et al.: Agentic business process management: A research manifesto. Information Systems **140**, 102738 (2026)
3. Cimatti, A., Roveri, M., Susi, A., Tonetta, S.: Validation of requirements for hybrid systems: A formal approach. TOSEM **21**(4), 1–34 (2013)
4. Corea, C., Alman, A., Maggi, F.M., Wittlinger, P.H.: Declarative process specifications over discrete/continuous event data. In: International Conference on Advanced Information Systems Engineering. pp. 277–294. Springer (2025)
5. Corea, C., Kuhlmann, I., Thimm, M., Grant, J.: Paraconsistent reasoning for inconsistency measurement in decl. process specifications. Inf. Systems **122** (2024)
6. De Luzi, F., Leotta, F., Marrella, A., Mecella, M., Monti, F.: On the interplay between business process management and internet-of-things. In: Internet of Things Meets BPM: A Synergistic Integration, pp. 1–24. Springer (2026)
7. Di Ciccio, C., Montali, M.: Declarative process specifications: reasoning, discovery, monitoring. In: Process mining handbook, pp. 108–152. Springer (2022)
8. Donzé, A.: On signal temporal logic. In: International conference on runtime verification. pp. 382–383. Springer (2013)
9. Donzé, A., Ferrere, T., Maler, O.: Efficient robust monitoring for stl. In: International conference on computer aided verification. pp. 264–279. Springer (2013)
10. Emerson, E.A.: Temporal and modal logic. In: Formal models and semantics, pp. 995–1072. Elsevier (1990)
11. Graja, I., Kallel, S., Guermouche, N., Cheikhrouhou, S., Hadj Kacem, A.: Modelling and verifying time-aware processes for cp environments. IET **13** (2019)
12. Hartman, S., Mikusinski, J.: The theory of lebesgue measure and integration. Canadian Mathematical Bulletin **5**(2), 205–206 (1962)
13. Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., Fortino, G., Gal, A., Kannengiesser, U., Leotta, F., et al.: The internet of things meets business process management: a manifesto. IEEE Systems, Man, and Cybernetics Magazine **6**(4), 34–44 (2020)
14. Lee, J., Yu, G., Bae, K.: Efficient smt-based model checking for signal temporal logic. In: ASE. pp. 343–354. IEEE (2021)
15. Melani, B., Bartocci, E., Chiari, M.: A tree-shaped tableau for checking the satisfiability of signal temporal logic with bounded temporal operators. ACM Transactions on Embedded Computing Systems **24**, 1–26 (2025)
16. Thimm, M.: Inconsistency measurement. In: International Conference on Scalable Uncertainty Management. pp. 9–23. Springer (2019)

[7] `https://github.com/iKuhlmann/ASP-Based-Approach-for-Measuring-Inconsistency-in-STL`