Measuring Inconsistency in Declarative Process Specifications[⋆]

Carl Corea[1], John Grant[2], and Matthias Thimm[3]

[1] Institute for IS Research, University of Koblenz-Landau, Koblenz, Germany
`ccorea@uni-koblenz.de`
[2] University of Maryland, College Park, USA
`grant@cs.umd.edu`
[3] Artificial Intelligence Group, University of Hagen, Hagen, Germany
`matthias.thimm@fernuni-hagen.de`

Abstract. We address the problem of measuring inconsistency in declarative process specifications, with an emphasis on linear temporal logic on fixed traces ($\mathrm{LTL}_{\mathrm{ff}}$). As we will show, existing inconsistency measures for classical logic cannot provide a meaningful assessment of inconsistency in LTL in general, as they cannot adequately handle the temporal operators. We therefore propose a novel paraconsistent semantics as a framework for inconsistency measurement. We then present two new inconsistency measures based on these semantics and show that they satisfy important desirable properties. We show how these measures can be applied to declarative process models and investigate the computational complexity of the introduced approach.

Keywords: Inconsistency Measurement · LTL · Declare.

1 Introduction

Linear temporal logic (LTL) is an important logic for specifying the (temporal) behavior of business processes in the form of *declarative process specifications* [1, 18]. The underlying idea is that time is represented as a linear sequence of states $T = (t_0, ..., t_m)$, where t_0 is the designated starting point. At every state, some statements may be true. Temporal operators specify properties that must hold over the sequence of states. For example, the operator **X** (*next*) means that a certain formula holds at the next state. Likewise, the operator **G** (*globally*) means that a certain formula will hold for all following states. Note that we assume the sequence to be finite, i.e., we consider a linear temporal logic over finite traces ($\mathrm{LTL}_{\mathrm{f}}$) [5, 18].

Traditionally, model checking has been used to verify that a particular model—that is, the assignment of truth values for statements over the time sequence—satisfies the requirements. However, a problem in this use case arises if the set

[⋆] This work has been partially supported by the Deutsche Forschungsgemeinschaft (Grant DE 1983/9-1).

of formulas is *inconsistent*, i. e., contains contradictory specifications. In such a case, the set of specifications cannot be applied for its intended purpose of process verification. For example, consider the two sets of formulas $\mathcal{K}_1$ and $\mathcal{K}_2$ (we will formalize syntax and semantics later):

$$\mathcal{K}_1 = \{\mathbf{X}a, \mathbf{X}\neg a\} \qquad\qquad \mathcal{K}_2 = \{\mathbf{G}a, \mathbf{G}\neg a\}$$

Both $\mathcal{K}_1$ and $\mathcal{K}_2$ are inconsistent, as they demand that both a and $\neg a$ hold in (some) following state, which is unsatisfiable. This calls for the analysis of such inconsistencies, to provide insights for inconsistency resolution.

In classical logic, all inconsistent sets are equally bad [12]. However, considering again the two sets, intuitively, $\mathcal{K}_2$ is "more" inconsistent than $\mathcal{K}_1$: The inconsistency in $\mathcal{K}_1$ only affects the next state, while the inconsistency in $\mathcal{K}_2$ affects all following states. This is an important insight that could prove useful for debugging or re-modelling LTL_f specifications or LTL_f-based constraint sets in general such as Declare. While there have been some recent works that can *identify* inconsistent sets in declarative process specifications [2, 3, 20], those works cannot look "into" those sets or compare them. In this work, we therefore show how to distinguish the *severity* of inconsistencies in LTL_f, specifically, LTL_ff.

A scientific field geared towards the quantitative assessment of inconsistency in knowledge representation formalisms is *inconsistency measurement* [8,22], and therefore represents a good candidate for this endeavour. Inconsistency measurement studies measures that aim to assess a *degree* of inconsistency with a numerical value. The intuition here is that a higher value represents a higher degree of inconsistency. Such measures can provide valuable insights for debugging inconsistent specifications, e. g., to determine whether certain sets of formulas are more inconsistent than others. As we will show, existing measures are currently not geared towards LTL_f and temporal operators, and therefore cannot provide a meaningful analysis. Therefore, the main goal of this paper is to develop a new approach for measuring inconsistency in linear temporal logic. To frame this problem, we introduce a variant of LTL_f, which we coin linear temporal logic on *fixed* traces LTL_ff (cf. Section 2.2).

Our contributions are as follows. We formalise the problem of measuring inconsistency in LTL_ff and propose a rationality postulate that should be met by quantitative measures applied to this setting (Section 2). We show that existing inconsistency measures do not satisfy this property, and propose an approach for measuring inconsistency based on a novel paraconsistent semantics for LTL_ff (Section 3). We then show how our approach can be applied for measuring inconsistency in declarative process models (Section 4). For evaluation, we investigate the computational complexity of central aspects regarding inconsistency measurement in LTL_ff (Section 5). A conclusion is provided in Section 6. Proofs for technical results can be found in a supplementary document provided online.

2 Preliminaries

The traditional setting for inconsistency measurement is that of propositional logic. For that, let At be some fixed propositional signature, i. e., a (possibly infinite) set of propositions, and let $\mathcal{L}(\mathsf{At})$ be the corresponding propositional language constructed using the usual connectives $\wedge$ (*conjunction*), $\vee$ (*disjunction*), and $\neg$ (*negation*). A literal is a proposition p or negated proposition $\neg p$.

Definition 1. *A knowledge base $\mathcal{K}$ is a finite set of formulas $\mathcal{K} \subseteq \mathcal{L}(\mathsf{At})$. Let $\mathbb{K}$ be the set of all knowledge bases.*

For a set of formulas X we denote the set of propositions in X by $\mathsf{At}(X)$.

Semantics for a propositional language is given by *interpretations* where an interpretation ω on At is a function $\omega : \mathsf{At} \to \{0,1\}$ (where 0 stands for false and 1 stands for true). Let $\Omega(\mathsf{At})$ denote the set of all interpretations for At. An interpretation ω *satisfies* (or is a *model* of) an atom $a \in \mathsf{At}$, denoted by $\omega \models a$, if and only if $\omega(a) = 1$. The satisfaction relation $\models$ is extended to formulas in the usual way. For $\Phi \subseteq \mathcal{L}(\mathsf{At})$ we also define $\omega \models \Phi$ if and only if $\omega \models \phi$ for every $\phi \in \Phi$. Furthermore, for every set of formulas X, the set of models is $\mathsf{Mod}(X) = \{\omega \in \Omega(\mathsf{At}) \mid \omega \models X\}$. Define $X \models Y$ for (sets of) formulas X and Y if $\omega \models X$ implies $\omega \models Y$ for all ω.

Let $\top$ denote any tautology and $\bot$ any contradiction. If $\mathsf{Mod}(X) = \emptyset$ we write $X \models \bot$ and say that X is *inconsistent*.

2.1 Inconsistency Measurement

Inconsistency as defined above is a binary concept. To provide more fine-grained insights on inconsistency beyond such a binary classification, the field of inconsistency measurement [22] has evolved. The main objects of study in this field are *inconsistency measures*, which are quantitative measures that assess the degree of inconsistency for a knowledge base $\mathcal{K}$ with a non-negative numerical value. Intuitively, a higher value reflects a higher degree, or severity, of inconsistency. This can be useful for determining if one set of formulas is "more" inconsistent than another. Let $\mathbb{R}^{\infty}_{\geq 0}$ be the set of non-negative real values including ∞. Then, an inconsistency measure is defined as follows.

Definition 2. *An inconsistency measure $\mathcal{I}$ is any function $\mathcal{I} : \mathbb{K} \to \mathbb{R}^{\infty}_{\geq 0}$.*

To constrain the desired behavior of concrete inconsistency measures, several properties, called *rationality postulates*, have been proposed. A well-agreed upon property is that of *consistency*, which states that an inconsistency measure should return a value of 0 iff there is no inconsistency.

***Consistency* (CO)** $\mathcal{I}(\mathcal{K}) = 0$ if and only if $\mathcal{K}$ is consistent.

Further important postulates introduced in [10] are *monotony, dominance* and *free-formula independence*, which we will define below. For that, we need some further notation.

First, a set $M \subseteq \mathcal{K}$ is called a *minimal inconsistent subset* (MIS) of $\mathcal{K}$ if $M \models \bot$ and there is no $M' \subset M$ with $M' \models \bot$. Let $\mathsf{MI}(\mathcal{K})$ be the set of all MISs of $\mathcal{K}$. Second, a formula $\alpha \in \mathcal{K}$ is called a *free formula* if $\alpha \notin \bigcup \mathsf{MI}(\mathcal{K})$. Let $\mathsf{Free}(\mathcal{K})$ be the set of all free formulas of $\mathcal{K}$.

For the remainder of this section, let $\mathcal{I}$ be an inconsistency measure, $\mathcal{K}, \mathcal{K}' \in \mathbb{K}$, and $\alpha, \beta \in \mathcal{L}(\mathsf{At})$. Then, the basic postulates from [10] are defined as follows.

Monotony **(MO)** If $\mathcal{K} \subseteq \mathcal{K}'$ then $\mathcal{I}(\mathcal{K}) \leq \mathcal{I}(\mathcal{K}')$.
Free-formula independence **(IN)** If $\alpha \in \mathsf{Free}(\mathcal{K})$ then $\mathcal{I}(\mathcal{K}) = \mathcal{I}(\mathcal{K} \setminus \{\alpha\})$.
Dominance **(DO)** If $\alpha \not\models \perp$ and $\alpha \models \beta$ then $\mathcal{I}(\mathcal{K} \cup \{\alpha\}) \geq \mathcal{I}(\mathcal{K} \cup \{\beta\})$.

MO states that adding formulas to the knowledge base cannot decrease the inconsistency value. IN means that removing free formulas from the knowledge base does not change the inconsistency value. DO consists of several cases, depending on the presence or absence of α or β in $\mathcal{K}$: the idea is that substituting a consistent formula α by a weaker formula β cannot increase the inconsistency.

Numerous inconsistency measures have been proposed (see [23] for a survey), many of which differ in regard to their compliance w.r.t. the introduced postulates. In this work, we will consider six measures as defined below. In order to define the *contension measure* $\mathcal{I}_c$ [7] we need some additional background on Priest's three-valued semantics [19]. A three-valued interpretation is a function $\nu : \mathsf{At} \rightarrow \{0, 1, \mathrm{B}\}$, which assigns to every atom either 0, 1 or B, where 0 and 1 correspond to $false$ and $true$, respectively, and B (standing for *both*) denotes a conflict. Assuming the *truth order* $\prec_T$ with $0 \prec_T B \prec_T 1$, the function ν can be extended to arbitrary formulas as follows: $\nu(\alpha \wedge \beta) = \min_{\prec_T}(\nu(\alpha), \nu(\beta))$, $\nu(\alpha \vee \beta) = \max_{\prec_T}(\nu(\alpha), \nu(\beta))$, $\nu(\neg\alpha) = 1$ if $\nu(\alpha) = 0$, $\nu(\neg\alpha) = 0$ if $\nu(\alpha) = 1$, and $\nu(\neg\alpha) = B$ if $\nu(\alpha) = B$. We say that an interpretation ν satisfies a formula α, denoted by $\nu \models^3 \alpha$, iff $\nu(\alpha) = 1$ or $\nu(\alpha) = \mathrm{B}$.

We will now define the measures used in this work.

Definition 3. *Let the measures $\mathcal{I}_d$, $\mathcal{I}_{\mathsf{MI}}$, $\mathcal{I}_p$, $\mathcal{I}_r$, $\mathcal{I}_c$, and $\mathcal{I}_{at}$ be defined as follows:*

$$\mathcal{I}_d(\mathcal{K}) = \begin{cases} 1 & \textit{if } \mathcal{K} \models \perp \\ 0 & \textit{otherwise} \end{cases}$$

$$\mathcal{I}_{\mathsf{MI}}(\mathcal{K}) = |\mathsf{MI}(\mathcal{K})|$$

$$\mathcal{I}_p(\mathcal{K}) = |\bigcup_{M \in \mathsf{MI}(\mathcal{K})} M|$$

$$\mathcal{I}_r(\mathcal{K}) = \min\{|X| \mid X \subseteq \mathcal{K} \textit{ and } \mathcal{K} \setminus X \not\models \perp\}$$

$$\mathcal{I}_c(\mathcal{K}) = \min\{|\nu^{-1}(B) \cap \mathsf{At}| \mid \nu \models^3 \mathcal{K}\}$$

$$\mathcal{I}_{at}(\mathcal{K}) = |\bigcup_{M \in \mathsf{MI}(\mathcal{K})} \mathsf{At}(M)|$$

A baseline approach is the drastic inconsistency measure $\mathcal{I}_d$ [11], which only differentiates between inconsistent and consistent knowledge bases. The MI-inconsistency measure $\mathcal{I}_{\mathsf{MI}}$ [11] counts the number of minimal inconsistent subsets. A similar version is the problematic inconsistency measure $\mathcal{I}_p$ [7], which counts the number of distinct formulas appearing in any inconsistent subset. The repair measure $\mathcal{I}_r$ counts the smallest number of formulas that must be removed in order to restore consistency. The contension measure $\mathcal{I}_c$ [7] quantifies inconsistency by seeking a three-valued interpretation that assigns B to a minimal number of propositions. Finally, the $\mathcal{I}_{at}$ measure counts the number of atoms in the non-free formulas.

We conclude this section with a small example illustrating the behavior of the considered inconsistency measures.

Example 1. Consider $\mathcal{K}_3$, defined via

$$\mathcal{K}_3 = \{a, \neg a, b, \neg b \wedge c \wedge d, \neg a \vee \neg b\}$$

Then we have that

$$\mathsf{MI}(\mathcal{K}_3) = \{\{a, \neg a\}, \{b, \neg b \wedge c \wedge d\}, \{a, \neg a \vee \neg b, b\}\}$$

Thus

$$\begin{array}{ccc} \mathcal{I}_d(\mathcal{K}_3) = 1 & \mathcal{I}_{\mathsf{MI}}(\mathcal{K}_3) = 3 & \mathcal{I}_p(\mathcal{K}_3) = 5 \\ \mathcal{I}_r(\mathcal{K}_3) = 2 & \mathcal{I}_c(\mathcal{K}_3) = 2 & \mathcal{I}_{at}(\mathcal{K}_3) = 4 \end{array}$$

The main focus of study in inconsistency measurement, and the introduced measures, has been on propositional logic. In this work, our aim is to apply inconsistency measures for linear time logic, which we introduce now.

2.2 Linear Temporal Logic on Fixed Traces

In this work, we consider a specific variant of $\mathrm{LTL_f}$ that we coin linear temporal logic on *fixed* traces ($\mathrm{LTL_{ff}}$). We consider a linear sequence of states $t_0, \ldots, t_m$, where every t_i is the state at instant i. We assume that $m > 1$ to avoid the trivial case. Note that the difference with $\mathrm{LTL_f}$—where interpretations can vary in their length as long as they are finite—is that we keep the length of this sequence finite and fixed across all interpretations. This variant of $\mathrm{LTL_f}$ is introduced mainly to discuss matters of inconsistency measurement, as here, the inconsistency value is computed in regard to a comparable length for all formulas. However, the ideas presented in the next sections can be extended to $\mathrm{LTL_f}$ [4] in a straightforward manner: In the unbounded case we can use a parameter N and then proceed as in the bounded case. This also means that m must not necessarily be known or provided a priori, as a parameter N can be selected.

The syntax of $\mathrm{LTL_{ff}}$ is the same as the syntax of LTL and $\mathrm{LTL_f}$ [5]. Formulas are built from a set of propositional symbols At and are closed under the Boolean connectives, the unary operator $\mathbf{X}$ (*next*), and the binary operator $\mathbf{U}$ (*until*). Formally, any formula φ of $\mathrm{LTL_{ff}}$ is built using the grammar rule

$$\varphi ::= a|(\neg\varphi)|(\varphi_1 \wedge \varphi_2)|(\varphi_1 \vee \varphi_2)|(\mathbf{X}\varphi)|(\varphi_1 \mathbf{U} \varphi_2).$$

with $a \in \mathsf{At}$. Intuitively, $\mathbf{X}\varphi$ denotes that φ will hold at the next state and $(\varphi_1 \mathbf{U} \varphi_2)$ denotes that φ_1 will hold until the state when φ_2 holds. Let $d(\varphi) \in \mathbb{N}$ denote the maximal number of nested temporal operators in φ.[1]

[1] $d(\varphi)$ is inductively defined via $d(a) = 0$ for $a \in \mathsf{At}$, $d(\neg\phi) = d(\phi)$, $d(\phi_1 \wedge \phi_2) = d(\phi_1 \vee \phi_2) = \max\{d(\phi_1), d(\phi_2)\}$, $d(\mathbf{X}\phi) = 1 + d(\phi)$, and $d(\phi_1 \mathbf{U} \phi_2) = 1 + \max\{d(\phi_1), d(\phi_2)\}$.

From the basic operators, some useful abbreviations can be derived, including $\mathbf{F}\varphi$ (defined as $\top\mathbf{U}\varphi$), which denotes that φ will hold (eventually) in the future and $\mathbf{G}\varphi$ (defined as $\neg\mathbf{F}\neg\varphi$), which denotes that φ will hold for all following states. Again, let $\top$ be any tautology and $\perp$ any contradiction.

An LTL$_{\mathrm{ff}}$-interpretation $\hat{\omega}$ w.r.t. At is a function mapping each state and proposition to 0 or 1, meaning that $\hat{\omega}(t,a)=1$ if proposition a is assigned 1 (true) in state t.[2] Then the satisfaction of a formula ϕ by an interpretation $\hat{\omega}$, denoted by $\hat{\omega}\models\phi$, is defined via

$$\hat{\omega}\models\phi \quad\Leftrightarrow\quad \hat{\omega},t_0\models\phi$$

where $\hat{\omega},t_i\models\phi$ for any interpretation $\hat{\omega}$ as above and for every $t_i\in\{t_0,\ldots,t_m\}$ is inductively defined as follows:

$$\begin{aligned}
&\hat{\omega},t_i\models a \text{ iff } \hat{\omega}(t_i,a)=1 \text{ for } a\in\mathsf{At}\\
&\hat{\omega},t_i\models\neg\varphi \text{ iff } \hat{\omega},t_i\not\models\varphi\\
&\hat{\omega},t_i\models\varphi_1\wedge\varphi_2 \text{ iff } \hat{\omega},t_i\models\varphi_1 \text{ and } \hat{\omega},t_i\models\varphi_2\\
&\hat{\omega},t_i\models\varphi_1\vee\varphi_2 \text{ iff } \hat{\omega},t_i\models\varphi_1 \text{ or } \hat{\omega},t_i\models\varphi_2\\
&\hat{\omega},t_i\models\mathbf{X}\varphi \text{ iff } i<m \text{ and } \hat{\omega},t_{i+1}\models\varphi\\
&\hat{\omega},t_i\models\varphi_1\mathbf{U}\varphi_2 \text{ iff } \hat{\omega},t_j\models\varphi_2 \text{ for some } j\in\{i+1,\ldots,m\}\\
&\qquad\qquad\qquad\text{and } \hat{\omega},t_k\models\varphi_1 \text{ for all } k\in\{i,\ldots,j-1\}
\end{aligned}$$

An interpretation $\hat{\omega}$ satisfies a set of formulas K iff $\hat{\omega}\models\phi$ for all $\phi\in K$. A set K is consistent iff there exists $\hat{\omega}$ such that $\hat{\omega}\models K$. Define $X\models Y$ for (sets of) formulas X and Y if $\hat{\omega}\models X$ implies $\hat{\omega}\models Y$ for all $\hat{\omega}$.

2.3 Related Work and Contributions

This work is related to consistency- and model checking in declarative process specifications, see e.g. [9, 17, 21]. In particular, our approach extends recent works [2, 3, 17, 20] on the identification of inconsistent sets in declarative process specifications by allowing to look "into" those sets and leverage inconsistency resolution with quantitative insights. For example, existing resolution approaches mainly try to minimize the *number* of deleted formulas [2,3,14]. This however completely leaves aside the semantics of those formulas or their impact on any corresponding process. Given this motivation, it is useful to consider also the degree to which certain formulas affect the following behavior, which is why we propose time sensitive inconsistency measures.

This paper is related to [6] which presents several, what we call time sensitive, inconsistency measures for branching time logics (BTL). However, in this work we are able to avoid the complicated overload of branching time as the process specifications are provided in linear time logic. Using branching time logic adds a

[2] Recall that we assume time of a fixed length $t_0,\ldots,t_m$ and interpretations only vary in what is true at each state.

layer of complexity that is unnecessary when dealing with a linear time situation. Just to take one example, consider the set $\{\mathbf{X}a, \mathbf{X}\neg a\}$. In linear time logic this gives one inconsistency at the next state. But in the case of branching time logic what does $\mathbf{X}$ mean? There may be many "next" states. If $\mathbf{X}$ means "some next state" then the set is consistent because a and $\neg a$ may hold in different next states. If $\mathbf{X}$ means "all next states" then it is inconsistent but how inconsistent depends on the number of next states. We avoid such issues by dealing only with linear temporal logic. Note also that BTL takes a different view on time than $\mathrm{LTL_f}$ as studied in this paper and is therefore expressively incomparable (cf. [24]).

3 Inconsistency Measurement in $\mathbf{LTL_{ff}}$

In this section, we address the issue of measuring inconsistency in $\mathrm{LTL_{ff}}$. As we will show, existing inconsistency measures cannot provide meaningful insights when dealing with temporal logic. Therefore, we develop a novel paraconsistent semantics as a framework for handling inconsistency and propose two concrete inconsistency measures for $\mathrm{LTL_{ff}}$.

3.1 Motivation for Inconsistency Measures for $\mathbf{LTL_{ff}}$

We recall the sets of $\mathrm{LTL_{ff}}$ formulas $\mathcal{K}_1$ and $\mathcal{K}_2$:

$$\mathcal{K}_1 = \{\mathbf{X}a, \mathbf{X}\neg a\} \qquad\qquad \mathcal{K}_2 = \{\mathbf{G}a, \mathbf{G}\neg a\}$$

The knowledge base $\mathcal{K}_1$ states that a is both true and false in the next state while $\mathcal{K}_2$ states that a is both true and false in all future states. Obviously, both knowledge bases are inconsistent. Yet, the inconsistencies are different in regard to the number of states they affect. For $\mathcal{K}_1$ the number is 1 and for $\mathcal{K}_2$ the number is $m > 1$. It would therefore be desirable for an inconsistency measure to take this information into account and assign $\mathcal{K}_2$ a larger inconsistency value.

In order to capture $\mathrm{LTL_{ff}}$ by the inconsistency measurement framework of Section 2.1, from now on a knowledge base $\mathcal{K}$ (Definition 1) will be a finite set of $\mathrm{LTL_{ff}}$ formulas and $\mathbb{K}$ is the set of all $\mathrm{LTL_{ff}}$ knowledge bases. So we can apply the inconsistency measures for $\mathcal{K}_1$ and $\mathcal{K}_2$ in a straightforward manner.

Example 2. Consider $\mathcal{K}_1$ and $\mathcal{K}_2$. Then we have that

$$\begin{aligned}
\mathcal{I}_d(\mathcal{K}_1) &= 1 & \qquad \mathcal{I}_d(\mathcal{K}_2) &= 1\\
\mathcal{I}_{\mathsf{MI}}(\mathcal{K}_1) &= 1 & \qquad \mathcal{I}_{\mathsf{MI}}(\mathcal{K}_2) &= 1\\
\mathcal{I}_p(\mathcal{K}_1) &= 2 & \qquad \mathcal{I}_p(\mathcal{K}_2) &= 2\\
\mathcal{I}_r(\mathcal{K}_1) &= 1 & \qquad \mathcal{I}_r(\mathcal{K}_2) &= 1\\
\mathcal{I}_c(\mathcal{K}_1) &= 1 & \qquad \mathcal{I}_c(\mathcal{K}_2) &= 1\\
\mathcal{I}_{at}(\mathcal{K}_1) &= 1 & \qquad \mathcal{I}_{at}(\mathcal{K}_2) &= 1
\end{aligned}$$

Note that all six inconsistency measures give identical values for $\mathcal{K}_1$ and $\mathcal{K}_2$, because they, or for that matter, any other propositional logic inconsistency measure, cannot distinguish between $\mathbf{X}$ and $\mathbf{G}$. But intuitively $\mathcal{K}_2$ is more inconsistent than $\mathcal{K}_1$ because the inconsistency persists through all future states in $\mathcal{K}_2$ as opposed to the single state in $\mathcal{K}_1$. Thus, we believe that a proper inconsistency measure for LTL$_{\text{ff}}$ should distinguish between these operators. Therefore, we propose a new rationality postulate.

Time Sensitivity **(TS)** For all formulas φ of propositional logic,
$\mathcal{I}(\{\mathbf{G}\varphi, \mathbf{G}\neg\varphi\}) > \mathcal{I}(\{\mathbf{X}\varphi, \mathbf{X}\neg\varphi\})$.

In other words, the number of affected states should be reflected in the inconsistency value, i.e., inconsistency measures for LTL$_{\text{ff}}$ should be time sensitive.

Proposition 1. *$\mathcal{I}_d, \mathcal{I}_{\mathsf{MI}}, \mathcal{I}_p, \mathcal{I}_r, \mathcal{I}_c, \mathcal{I}_{at}$ violate TS.*

Following Proposition 1, the existing measures that we have from propositional logic cannot capture the desired behavior. Therefore, we introduce a novel approach to measure inconsistency in LTL$_{\text{ff}}$.

3.2 A Paraconsistent Semantics for LTL$_{\text{ff}}$

Our first contribution towards measuring inconsistency in LTL$_{\text{ff}}$ is to define an LTL$_{\text{ff}}$-variant of the three-valued semantics of [19]. By doing so, we not only develop a means to neatly express inconsistency measures for LTL$_{\text{ff}}$, but also define a general applicable paraconsistent semantics for LTL$_{\text{ff}}$.

A three-valued interpretation $\hat{\nu}$ for LTL$_{\text{ff}}$ is a function mapping each state and proposition to 0, 1 or B, that is, $\hat{\nu} : \{t_0, t_1, \dots t_m\} \times \mathsf{At} \rightarrow \{0, 1, \mathrm{B}\}$ where as before 0 and 1 correspond to the classic logical false and true, respectively, and B (standing for *both*) denotes a conflict. We then assign

$$\hat{\nu}(\phi) = \hat{\nu}(t_0, \phi)$$

where $\hat{\nu}(t_i, \phi)$, for any interpretation $\hat{\nu}$ as above and state $t_i \in \{t_0, \dots, t_m\}$, is inductively defined as follows:

$$\hat{\nu}(t_i, a) = \hat{\nu}(t_i, a) \text{ for } a \in \mathsf{At}$$

$$\hat{\nu}(t_i, \neg\phi) = \begin{cases} 1 \text{ if } \hat{\nu}(t_i, \phi) = 0 \\ 0 \text{ if } \hat{\nu}(t_i, \phi) = 1 \\ \mathrm{B} \text{ if } \hat{\nu}(t_i, \phi) = \mathrm{B} \end{cases}$$

$$\hat{\nu}(t_i, \varphi_1 \wedge \varphi_2) = \begin{cases} 1 \text{ if } \hat{\nu}(t_i, \varphi_1) = \hat{\nu}(t_i, \varphi_2) = 1 \\ 0 \text{ if } \hat{\nu}(t_i, \varphi_1) = 0 \text{ or } \hat{\nu}(t_i, \varphi_2) = 0 \\ \mathrm{B} \text{ otherwise} \end{cases}$$

$$\hat{\nu}(t_i, \varphi_1 \vee \varphi_2) = \begin{cases} 1 \text{ if } \hat{\nu}(t_i, \varphi_1) = 1 \text{ or } \hat{\nu}(t_i, \varphi_2) = 1 \\ 0 \text{ if } \hat{\nu}(t_i, \varphi_1) = \hat{\nu}(t_i, \varphi_2) = 0 \\ \mathrm{B} \text{ otherwise} \end{cases}$$

$$\hat{\nu}(t_i, \mathbf{X}\varphi) = \begin{cases} \hat{\nu}(t_{i+1}, \varphi) & \text{if } i < m \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{\nu}(t_i, \varphi_1 \mathbf{U} \varphi_2) = \begin{cases} 1 & \text{if there is } j \in \{i+1, \ldots, m\} \text{ with} \\ & \hat{\nu}(t_j, \varphi_2) = \hat{\nu}(t_i, \varphi_1) = \ldots \\ & = \hat{\nu}(t_{j-1}, \varphi_1) = 1 \\ \mathrm{B} & \text{if there is } j \in \{i+1, \ldots, m\} \text{ with} \\ & \{\hat{\nu}(t_j, \varphi_2), \hat{\nu}(t_i, \varphi_1), \ldots, \\ & \hat{\nu}(t_{j-1}, \varphi_1)\} = \{1, \mathrm{B}\} \\ 0 & \text{otherwise} \end{cases}$$

Some comments on the above definition are in order. First, note that the evaluation of the classical Boolean connectives is the same as for propositional three-valued semantics (see Section 2.1). Furthermore, the evaluation of $\mathbf{X}\phi$ is simply the truth value of ϕ at the next state, or, if there is no next state, 0 (as for the classical semantics of LTL$_{\mathrm{ff}}$). The main new feature, however, is the three-valued evaluation of a formula of the form $\varphi_1 \mathbf{U} \varphi_2$. This formula evaluates to 1 as in the classical case, i. e., if ϕ_2 evaluates to 1 in some future state and ϕ_1 evaluates to 1 in between. We evaluate $\varphi_1 \mathbf{U} \varphi_2$ to B if ϕ_2 evaluates to 1 or B in some future state and ϕ_1 evaluates to 1 or B in between (and at least one of these evaluations must be to B). Finally, $\varphi_1 \mathbf{U} \varphi_2$ evaluates to 0 otherwise, i.e., if either ϕ_2 always evaluates to 0 in the future or in-between φ_1 evaluates at least once to 0.

A three-valued LTL$_{\mathrm{ff}}$ interpretation $\hat{\nu}$ satisfies a formula ϕ, denoted by $\hat{\nu} \models^3 \phi$, iff $\hat{\nu}(\phi, t_0) \in \{1, \mathrm{B}\}$. A three-valued interpretation $\hat{\nu}$ satisfies a set of formulas $\mathcal{K}$ iff $\hat{\nu} \models^3 \phi$ for all $\phi \in \mathcal{K}$.

Example 3. Let $\mathsf{At} = \{a, b\}$ and assume $m = 2$. Consider the knowledge base $\mathcal{K}$ defined via

$$\mathcal{K} = \{\mathbf{X}\neg a, a\mathbf{U}b\}$$

and the three-valued interpretation $\hat{\nu}$ defined via

$$\begin{aligned} \hat{\nu}(t_0, a) &= 1 & \hat{\nu}(t_0, b) &= 0 \\ \hat{\nu}(t_1, a) &= \mathrm{B} & \hat{\nu}(t_1, b) &= 0 \\ \hat{\nu}(t_2, a) &= 0 & \hat{\nu}(t_2, b) &= 1 \end{aligned}$$

Then we have $\hat{\nu}(t_0, a\mathbf{U}b) = \mathrm{B}$ as b evaluates to 1 in t_2 and a evaluates to B in t_1. Moreover, we have $\hat{\nu}(t_0, \mathbf{X}\neg a) = \mathrm{B}$ and therefore $\hat{\nu} \models^3 \mathcal{K}$.

Define $X \models^3 Y$ for formulas X and Y if $\hat{\nu} \models X$ implies $\hat{\nu} \models Y$ for all $\hat{\nu}$.

In the propositional logic case, $\models^3$ is a faithful extension of $\models$, meaning that $\omega \models \phi$ if and only if $\omega \models^3 \phi$ for every two-valued interpretation ω and every ϕ. Our LTL$_{\mathrm{ff}}$ extension of the three-valued semantics enjoys the same property (note that every two-valued interpretation is a also a three-valued interpretation that does not use the value B).

Proposition 2. *For every (two-valued) LTL$_{ff}$ interpretation $\hat{\omega}$ and LTL$_{ff}$ formula ϕ, $\hat{\omega} \models \phi$ if and only if $\hat{\omega} \models^3 \phi$.*

The three-valued semantics of [19] has another nice property in propositional logic, namely the non-existence of inconsistency: every propositional formula is trivially satisfiable by the interpretation that assigns B to all propositions. In general, an LTL$_{\mathrm{ff}}$ formula may become unsatisfiable w.r.t. to the three-valued semantics if it affects a state "beyond" t_m. However, for other formulas we obtain the following result regarding universal satisfiability.

Proposition 3. *For any LTL$_{\mathrm{ff}}$ formula ϕ with $d(\phi) \leq m$ there is $\hat{\nu}$ with $\hat{\nu} \models^3 \phi$.*

The semantics presented in this section allows for inconsistency-tolerant reasoning in LTL$_{\mathrm{ff}}$ (and it can straightforwardly be adapted for LTL$_{\mathrm{f}}$ and LTL). This provides a useful tool for the usual application scenarios of temporal logics, such as model checking and verification. While it may be worthwhile to investigate this aspect in more depth, in the remainder of this work we will focus on the application of this semantics for inconsistency measurement and postpone that endeavour to future work.

3.3 Time Sensitive Inconsistency Measures for LTL$_{\mathrm{ff}}$

We will now exploit our three-valued semantics for LTL$_{\mathrm{ff}}$ to define some new inconsistency measures. We do this similarly as for propositional logic by assessing the amount of usage of the paraconsistent truth value B in models of an LTL$_{\mathrm{ff}}$ knowledge base $\mathcal{K}$ but refine it by two different levels of granularity. This yields two new inconsistency measures.

Our first approach measures the number of states affected by inconsistency. For any three-valued interpretation $\hat{\nu}$, define

$$\mathsf{AffectedStates}(\hat{\nu}) = \{t \mid \exists a : \hat{\nu}(t,a) = \mathrm{B}\}$$

In other words, $\mathsf{AffectedStates}(\hat{\nu})$ is the set of states where $\hat{\nu}$ assigns B to at least one proposition. We can define an inconsistency measure by considering those 3-valued models of the knowledge base that affect the minimal number of states.

Definition 4 (LTL time measure). *Let $\mathcal{K}$ be a set of formulas. Then, the LTL time measure is defined via*

$$\mathcal{I}_d^{LTL}(\mathcal{K}) = \min_{\hat{\nu} \models^3 \mathcal{K}} |\mathit{AffectedStates}(\hat{\nu})|$$

if there is $\hat{\nu}$ with $\hat{\nu} \models^3 \mathcal{K}$ and $\mathcal{I}_d^{LTL}(\mathcal{K}) = \infty$ otherwise.

This measure counts the number of states for which the knowledge base is inconsistent. It is, in fact, the extension of the drastic measure, $\mathcal{I}_d$, in that for each state it adds 1 if there is an inconsistency and 0 otherwise. This measure can be used to distinguish the knowledge bases $\mathcal{K}_1$ and $\mathcal{K}_2$, i.e., it is time sensitive.

Example 4. We recall the knowledge bases $\mathcal{K}_1 = \{\mathbf{X}a, \mathbf{X}\neg a\}$ and $\mathcal{K}_2 = \{\mathbf{G}a, \mathbf{G}\neg a\}$. Then we have

$$\mathcal{I}_d^{LTL}(\mathcal{K}_1) = 1 \qquad\qquad \mathcal{I}_d^{LTL}(\mathcal{K}_2) = m$$

As an example where there is no $\hat{\nu}$ s.t. $\hat{\nu} \models^3 \mathcal{K}$, consider the formula $\mathbf{XXX}a$. This formula cannot be satisfied for $m = 2$, so $\mathcal{I}_d^{LTL}$ would return ∞ here.

Example 4 shows that the proposed measure $\mathcal{I}_d^{LTL}$ can already provide meaningful insights for measuring inconsistency in LTL. But a potential limitation is that it can only distinguish inconsistency in individual states in a binary manner. For example, $\mathcal{I}_d^{LTL}$ cannot distinguish the knowledge base $\mathcal{K}_4 = \{\mathbf{X}a, \mathbf{X}\neg a, \mathbf{X}b, \mathbf{X}\neg b\}$ from $\mathcal{K}_1$ because all inconsistencies occur at one state, namely t_1. For this reason we believe it is useful to be able to look inside states for inconsistency. In order to do so, given a three-valued interpretation $\hat{\nu}$, define

$$\mathsf{Conflictbase}(\hat{\nu}) = \{(t, a) \mid \hat{\nu}(t, a) = \mathrm{B}\}$$

Then, define the LTL contension measure as follows.

Definition 5 (LTL contension measure). *Let $\mathcal{K}$ be a set of formulas and*

$$\mathcal{I}_c^{LTL}(\mathcal{K}) = \min_{\hat{\nu} \models^3 \mathcal{K}} |\mathsf{Conflictbase}(\hat{\nu})|$$

if there is $\hat{\nu}$ with $\hat{\nu} \models^3 \mathcal{K}$ and $\mathcal{I}_c^{LTL}(\mathcal{K}) = \infty$ otherwise.

$\mathcal{I}_c^{LTL}$ seeks an interpretation that assigns B to a minimal number of propositions individually over all the states and uses this number for the inconsistency measure. This is an extension of $\mathcal{I}_d^{LTL}$, and for that matter, of $\mathcal{I}_c$ as it calculates $\mathcal{I}_c^{LTL}$ for each state and sums the numbers obtained this way.

Example 5. We recall the knowledge bases $\mathcal{K}_1 = \{\mathbf{X}a, \mathbf{X}\neg a\}$, $\mathcal{K}_4 = \{\mathbf{X}a, \mathbf{X}\neg a, \mathbf{X}b, \mathbf{X}\neg b\}$, and consider $\mathcal{K}_5 = \{\mathbf{G}a, \mathbf{G}\neg a, \mathbf{G}b, \mathbf{G}\neg b\}$. If $m = 3$, then we have

$$\begin{array}{lll} \mathcal{I}_d^{LTL}(\mathcal{K}_1) = 1 & \mathcal{I}_d^{LTL}(\mathcal{K}_4) = 1 & \mathcal{I}_d^{LTL}(\mathcal{K}_5) = 3 \\ \mathcal{I}_c^{LTL}(\mathcal{K}_1) = 1 & \mathcal{I}_c^{LTL}(\mathcal{K}_4) = 2 & \mathcal{I}_c^{LTL}(\mathcal{K}_5) = 6 \end{array}$$

As can be seen in Example 5, the two inconsistency measures proposed in this work can, contrary to previously existing measures, be used to provide meaningful insights into inconsistency in linear temporal logic, i.e., they are in fact time sensitive. As the two measures have a different granularity in regard to time, selecting which of the two to use depends on the intended use case.

Intuitively, it would be possible to devise further time-sensitive inconsistency measures for LTL$_{\mathrm{ff}}$. We will however leave this discussion for future work. Importantly, the aim of this paper is to show that traditional inconsistency measures cannot be plausibly applied to temporal logics, and to present means for time sensitive inconsistency measurement. In this regard, the measures proposed in this work can be used as a baseline for measuring inconsistency in LTL. Also, they (broadly) satisfy other desirable properties and can therefore be seen as strictly better (w.r.t. the considered postulates) than their propositional logic "counterpart", i.e., $\mathcal{I}_d$ for $\mathcal{I}_d^{LTL}$, respectively $\mathcal{I}_c$ for $\mathcal{I}_c^{LTL}$. The results of this section are summarized in Table 1. Proofs can be found online.

Note that only the measures we introduced satisfy TS. Note also that $\mathcal{I}_c^{LTL}$ does not satisfy IN due to the problem of iceberg inconsistencies, cf. the provided proofs.

$\mathcal{I}$	CO	MO	IN	DO	TS
$\mathcal{I}_d$	✓	✓	✓	✓	✗
$\mathcal{I}_{\mathsf{MI}}$	✓	✓	✓	✗	✗
$\mathcal{I}_p$	✓	✓	✓	✗	✗
$\mathcal{I}_r$	✓	✓	✓	✗	✗
$\mathcal{I}_c$	✓	✓	✗	✓	✗
$\mathcal{I}_{at}$	✓	✗	✗	✗	✗
$\mathcal{I}_d^{LTL}$	✓	✓	✓	✓	✓
$\mathcal{I}_c^{LTL}$	✓	✓	✗	✓	✓

Table 1: Compliance of inconsistency measures with rationality postulates.

4 Application to Declarative Process Models

A common application scenario for $\mathrm{LTL_f}$ is that of declarative process models [16], which are sets of ($\mathrm{LTL_f}$-based) constraints. For such declarative process models, the issue of inconsistency is equally as problematic, as any inconsistencies between the constraints make the declarative process model unsatisfiable.

There have been a number of works addressing the issue of inconsistency in declarative process models [2,3,14]. However, those works mainly look at whether a process model is inconsistent at all (in a binary manner), or try to identify sets of inconsistent constraints. Those works can however not look "into" those sets or assess their severity. For this use case, our proposed approach can be extended to declarative process models as follows.

4.1 Inconsistency Measurement in Declarative Process Models

A declarative process model consists of a set of constraints. Typically, these constraints are constructed using predefined templates, i. e., predicates, that are specified relative to a set of propositions (e. g., company activities).

Definition 6 (Declarative Process Model). *A declarative process model is a tuple $\boldsymbol{M} = (\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{C})$, where $\boldsymbol{A}$ is a set of propositions, $\boldsymbol{T}$ is a set of constraint types, and $\boldsymbol{C}$ is the set of constraints, which instantiate the template elements in $\boldsymbol{T}$ with activities in $\boldsymbol{A}$.*[3]

In this work, we consider the declarative modelling language Declare [16], which offers a set of "standard" templates. We will use a selection of templates shown in Table 2. We refer the reader to [3] for an overview of other Declare template types and corresponding semantics.

[3] For readability, we will denote declarative process models as a set of constraints ($\mathbf{C}$)

Template	**LTL$_{\text{ff}}$ Semantics**
Init(a)	a
End(a)	$\mathbf{G}(a \vee \mathbf{F}a)$
Response(a,b)	$\mathbf{G}(a \rightarrow \mathbf{F}b)$
NotResponse(a,b)	$\mathbf{G}(a \rightarrow \neg\mathbf{F}b)$
ChainResponse(a,b)	$\mathbf{G}(a \rightarrow \mathbf{X}b)$
NotChainResponse(a,b)	$\mathbf{G}(a \rightarrow \neg\mathbf{X}b)$
AtLeast(a,n)	$\mathbf{F}(a \wedge \mathbf{X}(\text{atLeast}(a, n\text{-}1))), \text{atLeast}(a, 1) = a \vee \mathbf{F}(a)$
AtMost(a,n)	$\mathbf{G}(\neg a \vee \mathbf{X}(\text{atMost}(a, n-1))), \text{atMost}(a, 0) = \mathbf{G}(\neg a)$

Table 2: LTL$_{\text{ff}}$ Semantics for a selection of Declare templates.

By rewriting the constraints of a declarative process model into LTL$_{\text{ff}}$ formulas, our approach for measuring inconsistency in LTL$_{\text{ff}}$ can be applied to Declare in a straightforward manner.

Example 6. Consider the sets of constraints C_a and C_b, defined via

$$\begin{aligned}
C_a &= \{\textsc{Init}(a), \textsc{Response}(a,b), \textsc{NotResponse}(a,b)\}\\
&\quad (\Leftrightarrow \{a, \mathbf{G}(a \rightarrow \mathbf{F}b), \mathbf{G}(a \rightarrow \neg\mathbf{F}b)\})\\
C_b &= \{\textsc{Init}(a), \textsc{Response}(a,b), \textsc{NotResponse}(a,b),\\
&\qquad \textsc{Response}(a,c), \textsc{NotResponse}(a,c)\}\\
&\quad (\Leftrightarrow \{a, \mathbf{G}(a \rightarrow \mathbf{F}b), \mathbf{G}(a \rightarrow \neg\mathbf{F}b), \mathbf{G}(a \rightarrow \mathbf{F}c), \mathbf{G}(a \rightarrow \neg\mathbf{F}c)\})
\end{aligned}$$

then we have that $\mathcal{I}_c^{LTL}(C_a) = 1$ and $\mathcal{I}_c^{LTL}(C_b) = 2$.

Due to the recursive definition of some "existence" constraints (cf. Table 2), note that also inconsistencies concerned with cardinalities can be assessed correctly.

Example 7. Consider $C_c = \{\textsc{AtMost}(a,1), \textsc{AtLeast}(a,2)\}$ and $C_d = \{\textsc{AtMost}(a,1), \textsc{AtLeast}(a,100)\}$, then $\mathcal{I}_d^{LTL}(C_c) < \mathcal{I}_d^{LTL}(C_d)$.

As a border case, note that any inconsistency referring to a point in time beyond the assumed sequence of states will return a value of ∞ per definition, as we cannot assess any error that leaves the boundaries of our logical framework.

Example 8. Let $C_e = \{\textsc{End}(a), \textsc{ChainResponse}(a,b)\}$, then $\mathcal{I}_d^{LTL}(C_e) = \infty$.

These examples show that our approach can provide detailed insights on the severity of inconsistency in declarative process models. Such insights can prove useful for prioritizing or re-modelling different issues of the process specification. In this context, it seems intuitive that conflicts affecting only the next state ($\mathbf{X}$) should be considered as less severe than conflicts affecting multiple following states ($\mathbf{G}$), i. e., for any LTL$_{\text{ff}}$ formula φ, $\mathcal{I}(\{\mathbf{G}\varphi, \mathbf{G}\neg\varphi\}) > \mathcal{I}(\{\mathbf{X}\varphi, \mathbf{X}\neg\varphi\})$. In this regard, there are still open questions on how to distinguish the operators $\mathbf{X}$ and $\mathbf{F}$, in particular: for an LTL$_{\text{ff}}$ formula φ, what is the relation between $\mathcal{I}(\{\mathbf{X}\varphi, \neg\mathbf{X}\varphi\})$ and $\mathcal{I}(\{\mathbf{F}\varphi, \neg\mathbf{F}\varphi\})$? We address this question in the following.

4.2 On Potentially Inconsistent States

Consider the following sets of constraints C_m and C_n, defined via

$C_m =$	$C_n =$
$\{\textsc{Init}(a) \Leftrightarrow a,$	$\{\textsc{Init}(a),$
$\textsc{Response}(a,b) \Leftrightarrow \mathbf{G}(a \rightarrow \mathbf{F}b),$	$\textsc{ChainResponse}(a,b) \Leftrightarrow \mathbf{G}(a \rightarrow \mathbf{X}b),$
$\textsc{NotResponse}(a,b) \Leftrightarrow \mathbf{G}(a \rightarrow \neg\mathbf{F}b)\}$	$\textsc{NotChainResponse}(a,b) \Leftrightarrow \mathbf{G}(a \rightarrow \neg\mathbf{X}b)\}$

Both sets are inconsistent, as they demand that b should and should not follow. However, the point in time at which the actual inconsistency can occur is different. Naturally, one question arises: which inconsistency is more severe? Or are they equally severe? We encourage the reader to come up with an own answer to this question at this point before we continue with our view on this matter.

Using the measures introduced in this work, the absolute number of affected states is 1 in both cases. So regarding the minimal number of affected states, the inconsistencies are equally severe. However, the certainty of where the inconsistency can occur at is clearly different, as visualized in Figure 1.

(a) Potentially inconsistent states for C_m (b) Certainty of inconsistency for C_n

Fig. 1: Visualization of the (un)certainty of where the inconsistency may occur for C_m and C_n.

In C_m, there are m different possible states to which a minimal interpretation could assign the truth value B to the proposition b, whereas the inconsistency can only occur in exactly 1 state for C_n. This could entail different severities for the inconsistencies, depending on the viewpoint:

Consider a running process which is in state t_0. For C_m, it is unclear when the inconsistency will occur. For C_n, it is directly known that the next state is inconsistent. Recovery mechanisms for such cases are well known [13], e. g., it would be possible to just skip the next state and continue with a consistent process. This is not possible for C_m without skipping all following states until the end of the process. So one might argue that the inconsistency in C_m is more severe. However, for C_n, this also means there is in fact no possible continuation as the process is in a dead-end state, thus, C_n needs to be attended to more urgently (So one might as well argue that the inconsistency in C_n is more severe).

In the field of inconsistency measurement, the dominance property states that substituting a consistent formula by a weaker formula cannot increase the inconsistency value [10]. However, when moving from C_m to C_n or vice-versa, we both replace one constraint with a stronger one and the other with a weaker one (every ChainResponse is also a Response but every NotResponse is a NotChainResponse). So the dominance property is not applicable here and the question remains which inconsistency is more severe. In this work, we will not give a definitive answer to this question and leave this discussion for future work. However, based on the two possible views given above, we will argue that they are, in fact, different. It would therefore be desirable to be able to distinguish the inconsistency in C_m and C_n. Here, the introduced contension concept can be adapted to quantify the certainty of *when* the inconsistency will occur.

The introduced measures quantify inconsistency by seeking an interpretation that assigns B to a minimal number of states. We denote the set of all such interpretations that assign B to a minimal number of states (at least to one) as

$$\hat{V}^{B>0}_{\mathsf{min}}(\mathcal{K}) = \{\hat{\nu} \models^3 \mathcal{K} : |\mathsf{AffectedStates}(\hat{\nu})| > 0 \wedge |\mathsf{AffectedStates}(\hat{\nu})| = \mathcal{I}^{LTL}_d(\mathcal{K})\}$$

Every such (minimal) interpretation also encodes which exact states are affected by the inconsistency. For C_n, only one state is necessarily affected (cf. Fig 1 (b)), thus, there exists only one minimal interpretation. For C_m, there are m different interpretations that are all equally minimal in terms of how many states are affected. So the number of minimal interpretations relates to the number of distinct (sets of) states that can potentially be affected.

Definition 7 (Number of Minimal Interpretations). *Let $\mathcal{K}$ be a set of formulas. Then, define the number of minimal interpretations via*

$$\#\mathit{minInterpretations}(\mathcal{K}) = |\hat{V}^{B>0}_{\mathit{min}}(\mathcal{K})|$$

Example 9. We recall C_m and C_n. Then we have that $\#\mathsf{minInterpretations}(C_m) = m$ and $\#\mathsf{minInterpretations}(C_n) = 1$ as expected (cf. the above discussion)

Importantly, the function $\#\mathsf{minInterpretations}$ is not an inconsistency measure, i. e., a higher value does not indicate a higher degree of inconsistency. It therefore also does not matter where the inconsistency in C_m eventually triggers. The value merely expresses the "certainty" of knowing where the conflict can occur at. The semantics of which is worse depends on the use case.

5 Computational Complexity

We conclude with an investigation of computational complexity in measuring inconsistency in LTL_{ff}. We assume familiarity with computational complexity, see [15] for an introduction. Proofs can be found online.

Note that deciding satisfiability is PSPACE-complete for LTL_{f} [5] and also intractable for many variants of LTL_{f} [4]. For our variant LTL_{ff}, as m is fixed, we get NP-completeness (think for example of a non-deterministic algorithm that guesses $\hat{\omega}$ and verifies (in polynomial time) that $\hat{\omega} \models \phi$).

Theorem 1. *Deciding whether a formula ϕ is satisfiable in LTL_{ff} is NP-complete.*

If the parameter m is given in unary, the complexity result holds as it is. However, if m is given in binary then the complexity will likely increase (in the membership proof, we need to guess an interpretation and if m is given in binary, that interpretation may be exponential in the size of the input).

We continue with an investigation of the computational complexity of measuring inconsistency in LTL$_{\text{ff}}$. For this, let $\mathbb{L}$ denote the set of all LTL$_{\text{ff}}$ knowledge bases. Following [23], we consider the following computational problems:

$\textsc{Exact}_{\mathcal{I}}$	**Input**:	$\mathcal{K} \in \mathbb{L}$, $x \in \mathbb{R}^{\infty}_{\geq 0}$
	Output:	TRUE iff $\mathcal{I}(\mathcal{K}) = x$
$\textsc{Upper}_{\mathcal{I}}$	**Input**:	$\mathcal{K} \in \mathbb{L}$, $x \in \mathbb{R}^{\infty}_{\geq 0}$
	Output:	TRUE iff $\mathcal{I}(\mathcal{K}) \leq x$
$\textsc{Lower}_{\mathcal{I}}$	**Input**:	$\mathcal{K} \in \mathbb{L}$, $x \in \mathbb{R}^{\infty}_{\geq 0} \setminus \{0\}$
	Output:	TRUE iff $\mathcal{I}(\mathcal{K}) \geq x$
$\textsc{Value}_{\mathcal{I}}$	**Input**:	$\mathcal{K} \in \mathbb{L}$
	Output:	The value of $\mathcal{I}(\mathcal{K})$

For $\textsc{Upper}_{\mathcal{I}}$, the same general non-deterministic algorithm can be applied.

Theorem 2. *$\textsc{Upper}_{\mathcal{I}^{LTL}_d}$ and $\textsc{Upper}_{\mathcal{I}^{LTL}_c}$ are NP-complete.*

Using the results in [23] we also get the following results for the other problems.

Corollary 1. *$\textsc{Lower}_{\mathcal{I}^{LTL}_d}$ and $\textsc{Lower}_{\mathcal{I}^{LTL}_c}$ are coNP-complete. $\textsc{Exact}_{\mathcal{I}^{LTL}_d}$ and $\textsc{Exact}_{\mathcal{I}^{LTL}_c}$ are in DP. $\textsc{Value}_{\mathcal{I}^{LTL}_d}$ and $\textsc{Value}_{\mathcal{I}^{LTL}_c}$ are in $FP^{NP[\log n]}$.*

In regard to the algorithmic implementation of our approach, a general approach of SAT encodings can be used. Corollary 1 gives a straightforward implementation for an algorithm to compute the measures by combining binary search with iterative calls to a SAT solver using an encoding of the problem Upper (see proof of Corollary 1). This encoding would be based on a SAT encoding for LTL$_{\text{ff}}$ satisfiability, which is straightforward.

6 Conclusion

In this work, we have presented an approach for measuring the severity of inconsistencies in declarative process specifications, in particular those based on linear temporal logic. In this regard, we introduced a paraconsistent semantics for LTL$_{\text{ff}}$ and developed two inconsistency measures. This provides useful insights for debugging or re-modelling declarative specifications, e. g., by allowing to compare or prioritize different inconsistencies. Here, our approach extends recent works [2,3,20] on the identification of inconsistent sets in declarative process specifications by allowing a look "into" those sets.

In future work, we aim to investigate the application of our approach to other languages such as GSM or DCR. Note that this is however not trivial, as the process models there might not be represented as orthogonal formulas. As a further limitation of our work, the current approach treats time as discrete time steps where any number of activities (within the bounds of the constraints) are allowed to occur at the same time. Real processes may however contain activities that take real time and may not be parallelizable because of resource constraints. As a result, a logically equivalent inconsistency may weigh more than another. In future work, we aim to address this issue with data-aware versions of LTL$_{\text{ff}}$.

References

1. Cecconi, A., De Giacomo, G., Di Ciccio, C., Maggi, F.M., Mendling, J.: A temporal logic-based measurement framework for process mining. In: Proceedings of the 2nd ICPM. pp. 113–120. IEEE (2020)
2. Corea, C., Nagel, S., Mendling, J., Delfmann, P.: Interactive and minimal repair of declarative process models. In: BPM Forum, Rome. pp. 3–19. Springer (2021)
3. Di Ciccio, C., Maggi, F.M., Montali, M., Mendling, J.: Resolving inconsistencies and redundancies in declarative process models. Inf. Systems **64**, 425–446 (2017)
4. Fionda, V., Greco, G.: The compl. of LTL on finite traces: Hard and easy fragments. In: Proc. of the 30th AAAI Conference on AI, Phoenix. pp. 971–977. AAAI (2016)
5. Giacomo, G.D., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: Proceedings of the 23rd IJCAI, Beijing. pp. 854–860. AAAI (2013)
6. Grant, J.: Measuring inconsistency in some branching time logics. Journal of Applied Non-Classical Logics **31**, 85–107 (2021)
7. Grant, J., Hunter, A.: Measuring consistency gain and inf. loss in stepwise inc. resolution. In: Proc. of the 11th ECSQARU, Belfast. pp. 362–373. Springer (2011)
8. Grant, J., Martinez, M.V.: Measuring Inc. in Information. College Pub. (2018)
9. Hildebrandt, T., Mukkamala, R.R., Slaats, T., Zanitti, F.: Contracts for cross-organizational workflows as timed dynamic condition response graphs. The Journal of Logic and Algebraic Programming **82**(5-7), 164–185 (2013)
10. Hunter, A., Konieczny, S., et al.: Shapley inc. values. KR **6**, 249–259 (2006)
11. Hunter, A., Konieczny, S., et al.: Measuring inconsistency through minimal inconsistent sets. KR **8**, 358–366 (2008)
12. Knight, K.: Measuring inconsistency. Journal of Phil. Logic **31**(1), 77–98 (2002)
13. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.: A framework for the systematic comparison and evaluation of compliance monitoring approaches. In: 17th IEEE EDOC, Vancouver. pp. 7–16. IEEE (2013)
14. Maggi, F.M., Westergaard, M., Montali, M., van der Aalst, W.M.: Runtime verification of ltl-based declarative process models. In: Proceedings of the 2nd RV, San Francisco. pp. 131–146. Springer (2011)
15. Papadimitriou, C.: Computational Complexity. Addison-Wesley (1994)
16. Pesic, M., Schonenberg, H., Van der Aalst, W.M.: Declare: Full support for loosely-structured processes. In: 11th IEEE EDOC, Annapolis. pp. 287–287. IEEE (2007)
17. Pill, I., Quaritsch, T.: Behavioral diagnosis of ltl specifications at operator level. In: 23rd International Joint Conference on Artificial Intelligence. Citeseer (2013)
18. Pnueli, A.: The temporal logic of programs. In: 18th Symposium on Foundations of Computer Science, Rhode Island. pp. 46–57. IEEE Computer Society (1977)
19. Priest, G.: Logic of Paradox. Journal of Phil. Logic **8**, 219–241 (1979)
20. Roveri, M., Di Ciccio, C., Di Francescomarino, C., Ghidini, C.: Computing unsatisfiable cores for LTLf specifications (Preprint). arXiv (2022)
21. Solomakhin, D., Montali, M., Tessaris, S., Masellis, R.D.: Verification of artifact-centric systems. In: Proceedings of the 11th ICSOC. pp. 252–266. Springer (2013)
22. Thimm, M.: Inconsistency measurement. In: Proceedings of the 13th International Conference on Scalable Uncertainty Management, Compiègne. Springer (2019)
23. Thimm, M., Wallner, J.P.: On the complexity of inc. meas. AI **275**, 411–456 (2019)
24. Vardi, M.Y.: Branching vs. linear time: Final showdown. In: Proceedings of the 7th TACAS, Italy. pp. 1–22. Springer (2001)