Draft 2014-06-24

Towards Large-scale Inconsistency Measurement

Matthias Thimm

Institute for Web Science and Technologies
Universität Koblenz-Landau, Germany

Abstract. We investigate the problem of inconsistency measurement on large knowledge bases by considering *stream-based inconsistency measurement*, i. e., we investigate inconsistency measures that cannot consider a knowledge base as a whole but process it within a stream. For that, we present, first, a novel inconsistency measure that is apt to be applied to the streaming case and, second, stream-based approximations for the new and some existing inconsistency measures. We conduct an extensive empirical analysis on the behavior of these inconsistency measures on large knowledge bases, in terms of runtime, accuracy, and scalability. We conclude that for two of these measures, the approximation of the new inconsistency measure and an approximation of the *contension* inconsistency measure, large-scale inconsistency measurement is feasible.

1 Introduction

Inconsistency measurement [1] is a subfield of Knowledge Representation and Reasoning (KR) that is concerned with the quantitative assessment of the severity of inconsistencies in knowledge bases. Consider the following two knowledge bases $\mathcal{K}_1$ and $\mathcal{K}_2$ formalized in propositional logic:

$$\mathcal{K}_1 = \{a, b \vee c, \neg a \wedge \neg b, d\} \qquad \mathcal{K}_2 = \{a, \neg a, b, \neg b\}$$

Both knowledge bases are classically inconsistent as for $\mathcal{K}_1$ we have $\{a, \neg a \wedge \neg b\} \models \bot$ and for $\mathcal{K}_2$ we have, e. g., $\{a, \neg a\} \models \bot$. These inconsistencies render the knowledge bases useless for reasoning if one wants to use classical reasoning techniques. In order to make the knowledge bases useful again, one can either use non-monotonic/paraconsistent reasoning techniques [2,3] or one revises the knowledge bases appropriately to make them consistent [4]. Looking again at the knowledge bases $\mathcal{K}_1$ and $\mathcal{K}_2$ one can observe that the *severity* of their inconsistency is different. In $\mathcal{K}_1$, only two out of four formulas (a and $\neg a \wedge \neg b$) are *participating* in making $\mathcal{K}_1$ inconsistent while for $\mathcal{K}_2$ all formulas contribute to its inconsistency. Furthermore, for $\mathcal{K}_1$ only two propositions (a and b) participate in a conflict and using, e. g., paraconsistent reasoning one could still infer meaningful statements about c and d. For $\mathcal{K}_2$ no such statement can be made. This leads to the assessment that $\mathcal{K}_2$ should be regarded *more* inconsistent than $\mathcal{K}_1$. Inconsistency measures can be used to quantitatively assess the inconsistency of knowledge bases and to provide a guide for how to repair them, cf. [5]. Moreover, they can be used as an analytical tool to assess the quality of knowledge representation. For example, one simple inconsistency measure is to take the number of *minimal inconsistent subsets* (MIs) as an indicator for the inconsistency: the more MIs a knowledge base contains,

the more inconsistent it is. For $\mathcal{K}_1$ we have then 1 as its inconsistency value and for $\mathcal{K}_2$ we have 2.

In this paper, we consider the computational problems of inconsistency measurement, particularly with respect to scalable inconsistency measurement on large knowledge bases, as they appear in, e. g., Semantic Web applications. To this end we present a novel inconsistency measure $\mathcal{I}_{hs}$ that approximates the η-inconsistency measure from [6] and is particularly apt to be applied to large knowledge bases. This measure is based on the notion of a *hitting set* which (in our context) is a minimal set of classical interpretations such that every formula of a knowledge base is satisfied by at least one element of the set. In order to investigate the problem of measuring inconsistency in large knowledge bases we also present a stream-based processing framework for inconsistency measurement. More precisely, the contributions of this paper are as follows:

1. We present a novel inconsistency measure $\mathcal{I}_{hs}$ based on hitting sets and show how this measure relates to other measures and, in particular, that it is a simplification of the η-inconsistency measure [6] (Section 3).
2. We formalize a theory of inconsistency measurement in streams and provide approximations of several inconsistency measures for the streaming case (Section 4).
3. We conduct an extensive empirical study on the behavior of those inconsistency measures in terms of runtime, accuracy, and scalability. In particular, we show that the stream variants of $\mathcal{I}_{hs}$ and of the *contension* measure $\mathcal{I}_c$ are effective and accurate for measuring inconsistency in the streaming setting and, therefore, in large knowledge bases (Section 5).

We give necessary preliminaries for propositional logic and inconsistency measurement in Section 2 and conclude the paper with a discussion in Section 6. Proofs of technical results are omitted but can be found in an extended version of this paper[1].

2 Preliminaries

Let At be a propositional signature, i. e., a (finite) set of propositions, and let $\mathcal{L}(\mathsf{At})$ be the corresponding propositional language. We use the symbol $\bot$ to denote contradiction. Then a knowledge base $\mathcal{K}$ is a finite set of formulas $\mathcal{K} \subseteq \mathcal{L}(\mathsf{At})$. Let $\mathbb{K}(\mathsf{At})$ be the set of all knowledge bases. We write $\mathbb{K}$ instead of $\mathbb{K}(\mathsf{At})$ when there is no ambiguity regarding the signature. Semantics to $\mathcal{L}(\mathsf{At})$ is given by *interpretations* $\omega : \mathsf{At} \to \{\mathsf{true}, \mathsf{false}\}$. Let $\mathsf{Int}(\mathsf{At})$ 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 $\omega \models a$ (or $\omega \in \mathsf{Mod}(a)$), if and only if $\omega(a) = \mathsf{true}$. Both $\models$ and $\mathsf{Mod}(\cdot)$ are extended to arbitrary formulas, sets, and knowledge bases as usual.

Inconsistency measures are functions $\mathcal{I} : \mathbb{K} \to [0, \infty)$ that aim at assessing the severity of the inconsistency in a knowledge base $\mathcal{K}$, cf. [5]. The basic idea is that the larger the inconsistency in $\mathcal{K}$ the larger the value $\mathcal{I}(\mathcal{K})$. However, inconsistency is a concept that is not easily quantified and there have been a couple of proposals for inconsistency measures so far, see e. g. [6,7,8,1,9,10]. There are two main paradigms for

[1] `http://www.mthimm.de/misc/thimm_inc_ki2014_extended.pdf`

assessing inconsistency [9], the first being based on the (number of) formulas needed to produce inconsistencies and the second being based on the proportion of the language that is affected by the inconsistency. Below we recall some popular measures from both categories but we first introduce some necessary notations. Let $\mathcal{K} \in \mathbb{K}$ be some knowledge base.

Definition 1. *A set $M \subseteq \mathcal{K}$ is called* minimal inconsistent subset *(MI) 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 MIs of $\mathcal{K}$.*

Definition 2. *A formula $\alpha \in \mathcal{K}$ is called* free formula *of $\mathcal{K}$ if there is no $M \in \mathsf{MI}(\mathcal{K})$ with $\alpha \in M$. Let $\mathsf{Free}(\mathcal{K})$ denote the set of all free formulas of $\mathcal{K}$.*

We adopt the following definition of a (basic) inconsistency measure from [5].

Definition 3. *A* basic inconsistency measure *is a function $\mathcal{I} : \mathbb{K} \to [0, \infty)$ that satisfies the following three conditions: 1.) $\mathcal{I}(\mathcal{K}) = 0$ if and only if $\mathcal{K}$ is consistent, 2.) if $\mathcal{K} \subseteq \mathcal{K}'$ then $\mathcal{I}(\mathcal{K}) \leq \mathcal{I}(\mathcal{K}')$, and 3.) for all $\alpha \in \mathsf{Free}(\mathcal{K})$ we have $\mathcal{I}(\mathcal{K}) = \mathcal{I}(\mathcal{K} \setminus \{\alpha\})$.*

For the remainder of this paper we consider the following selection of inconsistency measures: the MI measure $\mathcal{I}_{\mathsf{MI}}$, the MI^c measure $\mathcal{I}_{\mathsf{MI}^c}$, the contension measure $\mathcal{I}_c$, and the η measure $\mathcal{I}_\eta$, which will be defined below, cf. [5,6]. In order to define the contension measure $\mathcal{I}_c$ we need to consider three-valued interpretations for propositional logic [3]. A three-valued interpretation υ on At is a function $\upsilon : \mathsf{At} \to \{T, F, B\}$ where the values T and F correspond to the classical true and false, respectively. The additional truth value B stands for *both* and is meant to represent a conflicting truth value for a proposition. The function υ is extended to arbitrary formulas as shown in Table 1. Then, an interpretation υ satisfies a formula α, denoted by $\upsilon \models^3 \alpha$ if either $\upsilon(\alpha) = T$

α	β	$\alpha \wedge \beta$	$\alpha \vee \beta$	$\neg\alpha$	α	β	$\alpha \wedge \beta$	$\alpha \vee \beta$	$\neg\alpha$	α	β	$\alpha \wedge \beta$	$\alpha \vee \beta$	$\neg\alpha$
T	T	T	T	F	B	T	B	T	B	F	T	F	T	T
T	B	B	T	F	B	B	B	B	B	F	B	F	B	T
T	F	F	T	F	B	F	F	B	B	F	F	F	F	T

Table 1: Truth tables for propositional three-valued logic [3].

or $\upsilon(\alpha) = B$.

For defining the η-inconsistency measure [6] we need to consider probability functions P of the form $P : \mathsf{Int}(\mathsf{At}) \to [0, 1]$ with $\sum_{\omega \in \mathsf{Int}(\mathsf{At})} P(\omega) = 1$. Let $\mathcal{P}(\mathsf{At})$ be the set of all those probability functions and for a given probability function $P \in \mathcal{P}(\mathsf{At})$ define the probability of an arbitrary formula α via $P(\alpha) = \sum_{\omega \models \alpha} P(\omega)$.

Definition 4. *Let $\mathcal{I}_{MI}$, $\mathcal{I}_{MI^c}$, $\mathcal{I}_c$, and $\mathcal{I}_\eta$ be defined via*

$$\mathcal{I}_{MI}(\mathcal{K}) = |\mathsf{MI}(\mathcal{K})|, \qquad \mathcal{I}_\eta(\mathcal{K}) = 1 - \max\{\xi \mid \exists P \in \mathcal{P}(\mathsf{At}) : \forall \alpha \in \mathcal{K} : P(\alpha) \geq \xi\},$$

$$\mathcal{I}_{MI^c}(\mathcal{K}) = \sum_{M \in \mathsf{MI}(\mathcal{K})} \frac{1}{|M|}, \quad \mathcal{I}_c(\mathcal{K}) = \min\{|\upsilon^{-1}(B)| \mid \upsilon \models^3 \mathcal{K}\}$$

All these measures are basic inconsistency measures as defined in Definition 3.

Example 1. For the knowledge bases $\mathcal{K}_1 = \{a, b \vee c, \neg a \wedge \neg b, d\}$ and $\mathcal{K}_2 = \{a, \neg a, b, \neg b\}$ from the introduction we obtain $\mathcal{I}_{\mathsf{MI}}(\mathcal{K}_1) = 1$, $\mathcal{I}_{\mathsf{MI}^c}(\mathcal{K}_1) = 0.5$, $\mathcal{I}_c(\mathcal{K}_1) = 2$, $\mathcal{I}_\eta(\mathcal{K}_1) = 0.5$, $\mathcal{I}_{\mathsf{MI}}(\mathcal{K}_2) = 2$, $\mathcal{I}_{\mathsf{MI}^c}(\mathcal{K}_2) = 1$, $\mathcal{I}_c(\mathcal{K}_2) = 2$, $\mathcal{I}_\eta(\mathcal{K}_2) = 0.5$.

For a more detailed introduction to inconsistency measures see e. g. [1,5,6] and for some recent developments see e. g. [8,11].

As for computational complexity, the problem of computing an inconsistency value wrt. any of the above inconsistency measures is at least FNP-hard[2] as it contains a satisfiability problem as a sub problem.

3 An Inconsistency Measure based on Hitting Sets

The basic idea of our novel inconsistency measure $\mathcal{I}_{hs}$ is inspired by the measure $\mathcal{I}_\eta$ which seeks a probability function that maximizes the probability of all formulas of a knowledge base. Basically, the measure $\mathcal{I}_\eta$ looks for a minimal number of models of parts of the knowledge base and maximizes their probability in order to maximize the probability of the formulas. By just considering this basic idea we arrive at the notion of a *hitting set* for inconsistent knowledge bases.

Definition 5. *A subset $H \subset \mathsf{Int}(\mathsf{At})$ is called a* hitting set *of $\mathcal{K}$ if for every $\alpha \in \mathcal{K}$ there is $\omega \in H$ with $\omega \models \alpha$. H is called a* card*-minimal hitting set if it is minimal wrt. cardinality. Let $h_\mathcal{K}$ be the cardinality of any* card*-minimal hitting set (define $h_\mathcal{K} = \infty$ if there does not exist a hitting set of $\mathcal{K}$).*

Definition 6. *The function $\mathcal{I}_{hs} : \mathbb{K} \to [0, \infty]$ is defined via $\mathcal{I}_{hs}(\mathcal{K}) = h_\mathcal{K} - 1$ for every $\mathcal{K} \in \mathbb{K}$.*

Note, that if a knowledge base $\mathcal{K}$ contains a contradictory formula (e. g. $a \wedge \neg a$) we have $\mathcal{I}_{hs}(\mathcal{K}) = \infty$. In the following, we assume that $\mathcal{K}$ contains no such contradictory formulas.

Example 2. Consider the knowledge base $\mathcal{K}_3$ defined via

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

Then $\{\omega_1, \omega_2, \omega_3\} \subset \mathsf{Int}(\mathsf{At})$ with $\omega_1(a) = \omega_1(b) = \omega_1(c) = \mathsf{true}$, $\omega_2(a) = \omega_2(c) = \mathsf{true}$, $\omega_1(b) = \mathsf{false}$, and $\omega_3(a) = \omega_3(b) = \mathsf{true}$, $\omega_3(c) = \mathsf{false}$ is a card-minimal hitting set for $\mathcal{K}_3$ and therefore $\mathcal{I}_{hs}(\mathcal{K}_3) = 2$. Note that for the knowledge bases $\mathcal{K}_1$ and $\mathcal{K}_2$ from Example 1 we have $\mathcal{I}_{hs}(\mathcal{K}_1) = \mathcal{I}_{hs}(\mathcal{K}_2) = 1$.

Proposition 1. *The function $\mathcal{I}_{hs}$ is a (basic) inconsistency measure.*

The result below shows that $\mathcal{I}_{hs}$ also behaves well with some more properties mentioned in the literature [9,10]. For that, we denote with $\mathsf{At}(F)$ for a formula or a set of formulas F the set of propositions appearing in F. Furthermore, two knowledge bases $\mathcal{K}_1$, $\mathcal{K}_2$ are *semi-extensionally equivalent* ($\mathcal{K}_1 \equiv^\sigma \mathcal{K}_2$) if there is a bijection $\sigma : \mathcal{K}_1 \to \mathcal{K}_2$ such that for all $\alpha \in \mathcal{K}_1$ we have $\alpha \equiv \sigma(\alpha)$.

[2] FNP is the generalization of the class NP to functional problems.

Proposition 2. *The measure $\mathcal{I}_{hs}$ satisfies the following properties:*

- *If $\alpha \in \mathcal{K}$ is such that $\mathsf{At}(\alpha) \cap \mathsf{At}(\mathcal{K} \setminus \{\alpha\}) = \emptyset$ then $\mathcal{I}_{hs}(\mathcal{K}) = \mathcal{I}_{hs}(\mathcal{K} \setminus \{\alpha\})$ (*safe formula independence*).*
- *If $\mathcal{K} \equiv^{\sigma} \mathcal{K}'$ then $\mathcal{I}_{hs}(\mathcal{K}) = \mathcal{I}_{hs}(\mathcal{K}')$ (*irrelevance of syntax*).*
- *If $\alpha \models \beta$ and $\alpha \not\models \bot$ then $\mathcal{I}_{hs}(\mathcal{K} \cup \{\alpha\}) \geq \mathcal{I}_{hs}(\mathcal{K} \cup \{\beta\})$ (*dominance*).*

The measure $\mathcal{I}_{hs}$ can also be nicely characterized by a consistent *partitioning* of a knowledge base.

Definition 7. *A set $\Phi = \{\Phi_1, \ldots, \Phi_n\}$ with $\Phi_1 \cup \ldots \cup \Phi_n = \mathcal{K}$ and $\Phi_i \cap \Phi_j = \emptyset$ for $i, j = 1, \ldots, n$, $i \neq j$, is called a* partitioning *of $\mathcal{K}$. A partitioning $\Phi = \{\Phi_1, \ldots, \Phi_n\}$ is consistent if $\Phi_i \not\models \bot$ for $i = 1, \ldots, n$. A consistent partitioning Φ is called* card*-minimal if it is minimal wrt. cardinality among all consistent partitionings of $\mathcal{K}$.*

Proposition 3. *A consistent partitioning Φ is a* card*-minimal partitioning of $\mathcal{K}$ if and only if $\mathcal{I}_{hs}(\mathcal{K}) = |\Phi| - 1$.*

As $\mathcal{I}_{hs}$ is inspired by $\mathcal{I}_{\eta}$ we go on by comparing these two measures.

Proposition 4. *Let $\mathcal{K}$ be a knowledge base. If $\infty > \mathcal{I}_{hs}(\mathcal{K}) > 0$ then*

$$1 - \frac{1}{\mathcal{I}_{hs}(\mathcal{K})} < \mathcal{I}_{\eta}(\mathcal{K}) \leq 1 - \frac{1}{\mathcal{I}_{hs}(\mathcal{K}) + 1}$$

Note that for $\mathcal{I}_{hs}(\mathcal{K}) = 0$ we always have $\mathcal{I}_{\eta}(\mathcal{K}) = 0$ as well, as both are basic inconsistency measures.

Corollary 1. *If $\mathcal{I}_{\eta}(\mathcal{K}_1) \leq \mathcal{I}_{\eta}(\mathcal{K}_2)$ then $\mathcal{I}_{hs}(\mathcal{K}_1) \leq \mathcal{I}_{hs}(\mathcal{K}_2)$.*

However, the measures $\mathcal{I}_{\eta}$ and $\mathcal{I}_{hs}$ are not equivalent as the following example shows.

Example 3. Consider the knowledge bases $\mathcal{K}_1 = \{a, \neg a\}$ and $\mathcal{K}_2 = \{a, b, \neg a \vee \neg b\}$. Then we have $\mathcal{I}_{hs}(\mathcal{K}_1) = \mathcal{I}_{hs}(\mathcal{K}_2) = 1$ but $\mathcal{I}_{\eta}(\mathcal{K}_1) = 0.5 > 1/3 = \mathcal{I}_{\eta}(\mathcal{K}_2)$.

It follows that the order among knowledge bases induced by $\mathcal{I}_{\eta}$ is a refinement of the order induced by $\mathcal{I}_{hs}$. However, $\mathcal{I}_{hs}$ is better suited for approximation in large knowledge bases than $\mathcal{I}_{\eta}$, cf. the following section.

The idea underlying $\mathcal{I}_{hs}$ is also similar to the contension inconsistency measure $\mathcal{I}_c$. However, these measures are not equivalent as the following example shows.

Example 4. Consider the knowledge bases $\mathcal{K}_1 = \{a \wedge b \wedge c, \neg a \wedge \neg b \wedge \neg c\}$ and $\mathcal{K}_2 = \{a \wedge b, \neg a \wedge b, a \wedge \neg b\}$. Then we have $\mathcal{I}_{hs}(\mathcal{K}_1) = 2 < 3 = \mathcal{I}_{hs}(\mathcal{K}_2)$ but $\mathcal{I}_c(\mathcal{K}_1) = 3 > 2 = \mathcal{I}_c(\mathcal{K}_2)$.

4 Inconsistency Measurement in Streams

In the following, we discuss the problem of inconsistency measurement in large knowledge bases. We address this issue by using a stream-based approach of accessing the formulas of a large knowledge base. Formulas of a knowledge base then need to be processed one by one by a stream-based inconsistency measure. The goal of this formalization is to obtain stream-based inconsistency measures that approximate given inconsistency measures when the latter would have been applied to the knowledge base as a whole. We first formalize this setting and, afterwards, provide concrete approaches for some inconsistency measures.

4.1 Problem Formalization

We use a very simple formalization of a stream that is sufficient for our needs.

Definition 8. *A* propositional stream $\mathcal{S}$ *is a function* $\mathcal{S} : \mathbb{N} \to \mathcal{L}(\mathsf{At})$. *Let* $\mathbb{S}$ *be the set of all propositional streams.*

A propositional stream models a sequence of propositional formulas. On a wider scope, a propositional stream can also be interpreted as a very general abstraction of the output of a linked open data crawler (such as LDSpider [12]) that crawls knowledge formalized as RDF (*Resource Description Framework*) from the web, enriched, e. g. with OWL semantics. We model large knowledge bases by propositional streams that indefinitely repeat the formulas of the knowledge base. For that, we assume for a knowledge base $\mathcal{K} = \{\phi_1, \ldots, \phi_n\}$ the existence of a *canonical enumeration* $\mathcal{K}^c = \langle \phi_1, \ldots, \phi_n \rangle$ of the elements of $\mathcal{K}$. This enumeration can be arbitrary and has no specific meaning other than to enumerate the elements in an unambiguous way.

Definition 9. *Let* $\mathcal{K}$ *be a knowledge base and* $\mathcal{K}^c = \langle \phi_1, \ldots, \phi_n \rangle$ *its canonical enumeration. The* $\mathcal{K}$-stream $\mathcal{S}_\mathcal{K}$ *is defined as* $\mathcal{S}_\mathcal{K}(i) = \phi_{(i \bmod n)+1}$ *for all* $i \in \mathbb{N}$.

Given a $\mathcal{K}$-stream $\mathcal{S}_\mathcal{K}$ and an inconsistency measure $\mathcal{I}$ we aim at defining a method that processes the elements of $\mathcal{S}_\mathcal{K}$ one by one and approximates $\mathcal{I}(\mathcal{K})$.

Definition 10. *A* stream-based inconsistency measure $\mathcal{J}$ *is a function* $\mathcal{J} : \mathbb{S} \times \mathbb{N} \to [0, \infty)$.

Definition 11. *Let* $\mathcal{I}$ *be an inconsistency measure and* $\mathcal{J}$ *a stream-based inconsistency measure. Then* $\mathcal{J}$ approximates *(or* is an approximation of*)* $\mathcal{I}$ *if for all* $\mathcal{K} \in \mathbb{K}$ *we have* $\lim_{i\to\infty} \mathcal{J}(\mathcal{S}_\mathcal{K}, i) = \mathcal{I}(\mathcal{K})$.

4.2 A Naive Window-based Approach

The simplest form of implementing a stream-based variant of any algorithm or function is to use a window-based approach, i. e., to consider at any time point a specific excerpt from the stream and apply the original algorithm or function on this excerpt. For any propositional stream $\mathcal{S}$ let $\mathcal{S}^{i,j}$ (for $i \leq j$) be the knowledge base obtained by taking the formulas from $\mathcal{S}$ between positions i and j, i. e., $\mathcal{S}^{i,j} = \{\mathcal{S}(i), \ldots, \mathcal{S}(j)\}$.

Definition 12. *Let* $\mathcal{I}$ *be an inconsistency measure,* $w \in \mathbb{N} \cup \{\infty\}$*, and* g *some function* $g : [0, \infty) \times [0, \infty) \to [0, \infty)$ *with* $g(x, y) \in [\min\{x, y\}, \max\{x, y\}]$. *We define the* naive window-based measure $\mathcal{J}^{w,g}_\mathcal{I} : \mathbb{S} \times \mathbb{N} \to [0, \infty)$ *via*

$$\mathcal{J}^{w,g}_\mathcal{I}(\mathcal{S}, i) = \begin{cases} 0 & \text{if } i = 0 \\ g(\mathcal{I}(\mathcal{S}^{\max\{0,i-w\},i}), \mathcal{J}^{w,g}_\mathcal{I}(\mathcal{S}, i-1)) & \text{otherwise} \end{cases}$$

for every $\mathcal{S}$ *and* $i \in \mathbb{N}$.

The function g in the above definition is supposed to be an aggregation function that combines the new obtained inconsistency value $\mathcal{I}(\mathcal{S}^{\max\{0,i-w\},i}_\mathcal{K})$ with the previous value $\mathcal{J}^{w,g}_\mathcal{I}(\mathcal{S}, i-1)$. This function can be, e. g., the maximum function $\max$ or a smoothing function $g_\alpha(x, y) = \alpha x + (1-\alpha)y$ for some $\alpha \in [0, 1]$ (for every $x, y \in [0, \infty)$).

Proposition 5. *Let $\mathcal{I}$ be an inconsistency measure, $w \in \mathbb{N} \cup \{\infty\}$, and g some function $g : [0, \infty) \times [0, \infty) \to [0, \infty)$ with $g(x, y) \in [\min\{x, y\}, \max\{x, y\}]$.*

1. *If w is finite then $\mathcal{J}_{\mathcal{I}}^{w,g}$ is* not *an approximation of $\mathcal{I}$.*
2. *If $w = \infty$ and $g(x, y) > \min\{x, y\}$ if $x \neq y$ then $\mathcal{J}_{\mathcal{I}}^{w,g}$ is an approximation of $\mathcal{I}$.*
3. *$\mathcal{J}_{\mathcal{I}}^{w,g}(\mathcal{S}_{\mathcal{K}}, i) \leq \mathcal{I}(\mathcal{K})$ for every $\mathcal{K} \in \mathbb{K}$ and $i \in \mathbb{N}$.*

4.3 Approximation Algorithms for $\mathcal{I}_{hs}$ and $\mathcal{I}_c$

The approximation algorithms for $\mathcal{I}_{hs}$ and $\mathcal{I}_c$ that are presented in this subsection are using concepts of the programming paradigms of *simulated annealing* and *genetic programming* [13]. Both algorithms follow the same idea and we will only formalize the one for $\mathcal{I}_{hs}$ and give some hints on how to adapt it for $\mathcal{I}_c$.

The basic idea for the stream-based approximation of $\mathcal{I}_{hs}$ is as follows. At any processing step we maintain a candidate set $C \in 2^{\mathsf{Int(At)}}$ (initialized with the empty set) that approximates a hitting set of the underlying knowledge base. At the beginning of a processing step we make a random choice (with decreasing probability the more formulas we already encountered) whether to remove some element of C. This action ensures that C does not contain superfluous elements. Afterwards we check whether there is still an interpretation in C that satisfies the currently encountered formula. If this is not the case we add some random model of the formula to C. Finally, we update the previously computed inconsistency value with $|C|-1$, taking also some aggregation function g (as for the naive window-based approach) into account. In order to increase the probability of successfully finding a minimal hitting set we do not maintain a single candidate set C but a (multi-)set $Cand = \{C_1, \ldots, C_m\}$ for some previously specified parameter $m \in \mathbb{N}$ and use the average size of these candidate hitting sets.

Definition 13. *Let $m \in \mathbb{N}$, g some function $g : [0, \infty) \times [0, \infty) \to [0, \infty)$ with $g(x, y) \in [\min\{x, y\}, \max\{x, y\}]$, and $f : \mathbb{N} \to [0, 1]$ some monotonically decreasing function with $\lim_{n\to\infty} f(n) = 0$. We define $\mathcal{J}_{hs}^{m,g,f}$ via*

$$\mathcal{J}_{hs}^{m,g,f}(\mathcal{S}, i) = \begin{cases} 0 & \text{if } i = 0 \\ \texttt{update}_{hs}^{m,g,f}(\mathcal{S}(i)) & \text{otherwise} \end{cases}$$

for every $\mathcal{S}$ and $i \in \mathbb{N}$. The function $\texttt{update}_{hs}^{m,g,f}$ is depicted in Algorithm 1.

At the first call of the algorithm $\texttt{update}_{hs}^{m,g,f}$ the value of $currentValue$ (which contains the currently estimated inconsistency value) is initialized to 0 and the (mulit-)set $Cand \subseteq 2^{\mathsf{Int(At)}}$ (which contains a population of candidate hitting sets) is initialized with m empty sets. The function f can be any monotonically decreasing function with $\lim_{n\to\infty} f(n) = 0$ (this ensures that at any candidate C reaches some stable result). The parameter m increases the probability that at least one of the candidate hitting sets attains the global optimum of a card-minimal hitting set.

As $\mathcal{J}_{hs}^{m,g,f}$ is a random process we cannot show that $\mathcal{J}_{hs}^{m,g,f}$ is an approximation of $\mathcal{I}_{hs}$ in the general case. However, we can give the following result.

Proposition 6. *For every probability $p \in [0, 1)$, g some function $g : [0, \infty) \times [0, \infty) \to [0, \infty)$ with $g(x, y) \in [\min\{x, y\}, \max\{x, y\}]$ and $g(x, y) > \min\{x, y\}$ if $x \neq y$, a monotonically decreasing function $f : \mathbb{N} \to [0, 1]$ with $\lim_{n\to\infty} f(n) = 0$, and*

Algorithm 1 $\texttt{update}_{hs}^{m,g,f}(form)$

1: Initialize $currentValue$ and $Cand$
2: $N = N + 1$
3: $newValue = 0$
4: **for all** $C \in Cand$ **do**
5: $\quad rand \in [0, 1]$
6: $\quad$ **if** $rand < f(N)$ **then**
7: $\quad\quad$ Remove some random ω from C
8: $\quad$ **if** $\neg\exists\omega \in C : \omega \models form$ **then**
9: $\quad\quad$ Add random $\omega \in \mathsf{Mod}(form)$ to C
10: $\quad newValue = newValue + (|C| - 1)/|Cand|$
11: $currentValue = g(newValue, currentValue)$
12: **return** $currentValue$

$\mathcal{K} \in \mathbb{K}$ there is $m \in \mathbb{N}$ such that with probability greater or equal p it is the case that $\lim_{i\to\infty} \mathcal{J}_{hs}^{m,g,f}(\mathcal{S}_\mathcal{K}, i) = \mathcal{I}_{hs}(\mathcal{K})$.

This result states that $\mathcal{J}_{hs}^{m,g,f}$ indeed approximates $\mathcal{I}_{hs}$ if we choose the number of populations large enough. In the next section we will provide some empirical evidence that even for small values of m results are satisfactory.

Both Definition 13 and Algorithm 1 can be modified slightly in order to approximate $\mathcal{I}_c$ instead of $\mathcal{I}_{hs}$, yielding a new measure $\mathcal{J}_c^{m,g,f}$. For that, the set of candidates $Cand$ contains three-valued interpretations instead of sets of classical interpretations. In line 7, we do not remove an interpretation from C but flip some arbitrary proposition from B to T or F. Similarly, in line 9 we do not add an interpretation but flip some propositions to B in order to satisfy the new formula. Finally, the inconsistency value is determined by taking the number of B-valued propositions. For more details see the implementations of both $\mathcal{J}_{hs}^{m,g,f}$ and $\mathcal{J}_c^{m,g,f}$, which will also be discussed in the next section.

5 Empirical Evaluation

In this section we describe our empirical experiments on runtime, accuracy, and scalability of some stream-based inconsistency measures. Our Java implementations[3] have been added to the *Tweety Libraries for Knowledge Representation* [14].

5.1 Evaluated Approaches and Experiment Setup

For our evaluation, we considered the inconsistency measures $\mathcal{I}_\mathsf{MI}$, $\mathcal{I}_{\mathsf{MI}^c}$, $\mathcal{I}_\eta$, $\mathcal{I}_c$, and $\mathcal{I}_{hs}$. We used the SAT solver *lingeling*[4] for the sub-problems of determining consis-

[3] $\mathcal{I}_\mathsf{MI}$, $\mathcal{I}_{\mathsf{MI}^c}$, $\mathcal{I}_\eta$, $\mathcal{J}_\mathcal{I}^{w,g}$: `http://mthimm.de/r?r=tweety-inc-commons`
$\mathcal{I}_c$, $\mathcal{I}_{hs}$: `http://mthimm.de/r?r=tweety-inc-pl`
$\mathcal{J}_{hs}^{m,g,f}$: `http://mthimm.de/r?r=tweety-stream-hs`
$\mathcal{J}_c^{m,g,f}$: `http://mthimm.de/r?r=tweety-stream-c`
Evaluation framework: `http://mthimm.de/r?r=tweety-stream-eval`

[4] `http://fmv.jku.at/lingeling/`

tency and to compute a model of a formula. For enumerating the set of MIs of a knowledge base (as required by $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{MI}^c}$) we used MARCO[5]. The measure $\mathcal{I}_\eta$ was implemented using the linear optimization solver *lp_solve*[6]. The measures $\mathcal{I}_{\mathsf{MI}}$, $\mathcal{I}_{\mathsf{MI}^c}$, and $\mathcal{I}_\eta$ were used to define three different versions of the naive window-based measure $\mathcal{J}_{\mathcal{I}}^{w,g}$ (with $w = 500, 1000, 2000$ and $g = \max$). For the measures $\mathcal{I}_c$ and $\mathcal{I}_{hs}$ we tested each three versions of their streaming variants $\mathcal{J}_c^{m,g_{0.75},f_1}$ and $\mathcal{J}_{hs}^{m,g_{0.75},f_1}$ (with $m = 10, 100, 500$) with $f_1 : \mathbb{N} \to [0, 1]$ defined via $f_1(i) = 1/(i + 1)$ for all $i \in \mathbb{N}$ and $g_{0.75}$ is the smoothing function for $\alpha = 0.75$ as defined in the previous section.

For measuring the runtime of the different approaches we generated 100 random knowledge bases in CNF (*Conjunctive Normal Form*) with each 5000 formulas (=disjunctions) and 30 propositions. For each generated knowledge base $\mathcal{K}$ we considered its $\mathcal{K}$-stream and processing of the stream was aborted after 40000 iterations. We fed the $\mathcal{K}$-stream to each of the evaluated stream-based inconsistency measures and measured the average runtime per iteration and the total runtime. For each iteration, we set a time-out of 2 minutes and aborted processing of the stream completely if a time-out occurred.

In order to measure accuracy, for each of the considered approaches we generated another 100 random knowledge bases with specifically set inconsistency values[7], used otherwise the same settings as above, and measured the returned inconsistency values.

To evaluate the scalability of our stream-based approach of $\mathcal{I}_{hs}$ we conducted a third experiment[8] where we fixed the number of propositions (60) and the specifically set inconsistency value (200) and varied the size of the knowledge bases from 5000 to 50000 (with steps of 5000 formulas). We measured the total runtime up to the point when the inconsistency value was within a tolerance of ± 1 of the expected inconsistency value.

The experiments were conducted on a server with two Intel Xeon X5550 QuadCore (2.67 GHz) processors with 8 GB RAM running SUSE Linux 2.6.

5.2 Results

Our first observation concerns the inconsistency measure $\mathcal{I}_\eta$ which proved to be not suitable to work on large knowledge bases[9]. Computing the value $\mathcal{I}_\eta(\mathcal{K})$ for some knowledge base $\mathcal{K}$ includes solving a linear optimization problem over a number of variables which is (in the worst-case) exponential in the number of propositions of the signature. In our setting with $|\mathsf{At}| = 30$ the generated optimization problem contained therefore $2^{30} = 1073741824$ variables. Hence, even the optimization problem itself could not be constructed within the timeout of 2 minutes for every step. As we are not aware of any more efficient implementation of $\mathcal{I}_\eta$, we will not report on further results for $\mathcal{I}_\eta$ in the following.

As for the runtime of the naive window-based approaches of $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{MI}^c}$ and our stream-based approaches for $\mathcal{I}_c$ and $\mathcal{I}_{hs}$ see Table 2. There one can see that $\mathcal{J}_{\mathcal{I}_{\mathsf{MI}}}^{w,g}$ and

[5] `http://sun.iwu.edu/~mliffito/marco/`

[6] `http://lpsolve.sourceforge.net`

[7] The sampling algorithms can be found at `http://mthimm.de/r?r=tweety-sampler`

[8] We did the same experiment with our stream-based approach of $\mathcal{I}_c$ but do not report the results due to the similarity to $\mathcal{I}_{hs}$ and space restrictions.

[9] More precisely, our implementation of the measure proved to be not suitable for this setting

Measure	RT (iteration)	RT (total)	Measure	RT (iteration)	RT (total)
$\mathcal{J}^{500,\max}_{\mathcal{I}_{\mathsf{MI}}}$	198ms	133m	$\mathcal{J}^{10,g_{0.75},f_1}_{c}$	0.16ms	6.406s
$\mathcal{J}^{1000,\max}_{\mathcal{I}_{\mathsf{MI}}}$	359ms	240m	$\mathcal{J}^{100,g_{0.75},f_1}_{c}$	1.1ms	43.632s
$\mathcal{J}^{2000,\max}_{\mathcal{I}_{\mathsf{MI}}}$	14703ms	9812m	$\mathcal{J}^{500,g_{0.75},f_1}_{c}$	5.21ms	208.422s
$\mathcal{J}^{500,\max}_{\mathcal{I}_{\mathsf{MI}^c}}$	198ms	134m	$\mathcal{J}^{10,g_{0.75},f_1}_{hs}$	0.07ms	2.788s
$\mathcal{J}^{1000,\max}_{\mathcal{I}_{\mathsf{MI}^c}}$	361ms	241m	$\mathcal{J}^{100,g_{0.75},f_1}_{hs}$	0.24ms	9.679s
$\mathcal{J}^{2000,\max}_{\mathcal{I}_{\mathsf{MI}^c}}$	14812ms	9874m	$\mathcal{J}^{500,g_{0.75},f_1}_{hs}$	1.02ms	40.614s

Table 2: Runtimes for the evaluated measures; each value is averaged over 100 random knowledge bases of 5000 formulas; the total runtime is after 40000 iterations

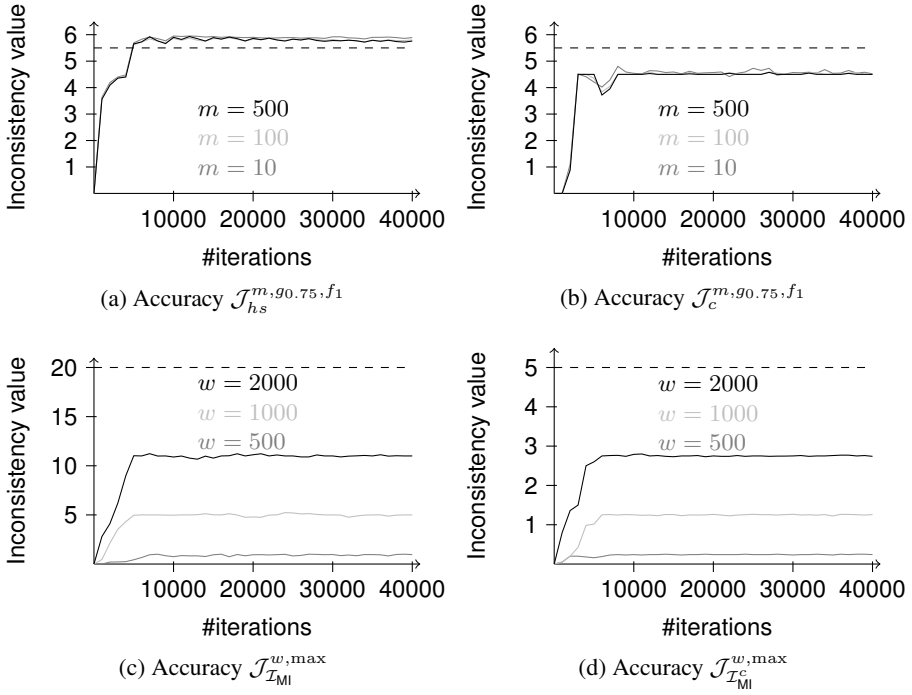
$\mathcal{J}^{w,g}_{\mathcal{I}_{\mathsf{MI}^c}}$ on the one hand, and $\mathcal{J}^{m,g,f}_{c}$ and $\mathcal{J}^{m,g,f}_{hs}$ on the other hand, have comparable runtimes, respectively. The former two have almost identical runtimes, which is obvious as the determination of the MIs is the main problem in both their computations. Clearly, $\mathcal{J}^{m,g,f}_{c}$ and $\mathcal{J}^{m,g,f}_{hs}$ are significantly faster per iteration (and in total) than $\mathcal{J}^{w,g}_{\mathcal{I}_{\mathsf{MI}}}$ and $\mathcal{J}^{w,g}_{\mathcal{I}_{\mathsf{MI}^c}}$, only very few milliseconds for the latter and several hundreds and thousands of milliseconds for the former (for all variants of m and w). The impact of increasing w for $\mathcal{J}^{m,g,f}_{c}$ and $\mathcal{J}^{m,g,f}_{hs}$ is expectedly linear while the impact of increasing the window size w for $\mathcal{J}^{w,g}_{\mathcal{I}_{\mathsf{MI}}}$ and $\mathcal{J}^{w,g}_{\mathcal{I}_{\mathsf{MI}^c}}$ is exponential (this is also clear as both solve an FNP-hard problem).

As for the accuracy of the different approaches see Figure 1. There one can see that both $\mathcal{J}^{m,g,f}_{hs}$ and $\mathcal{J}^{m,g,f}_{c}$ (Figures 1a and 1b) converge quite quickly (almost right after the knowledge base has been processed once) into a $[-1, 1]$ interval around the actual inconsistency value, where $\mathcal{J}^{m,g,f}_{c}$ is even closer to it. The naive window-based approaches (Figures 1c and 1d) have a comparable bad performance (this is clear as those approaches cannot *see* all MIs at any iteration due to the limited window size). Surprisingly, the impact of larger values of m for $\mathcal{J}^{m,g,f}_{hs}$ and $\mathcal{J}^{m,g,f}_{c}$ is rather small in terms of accuracy which suggests that the random process of our algorithm is quite robust. Even for $m = 10$ the results are quite satisfactory.

As for the scalability of $\mathcal{J}^{m,g_{0.75},f_1}_{hs}$ see Figure 2. There one can observe a linear increase in the runtime of all variants wrt. the size of the knowledge base. Furthermore, the difference between the variants is also linearly in the parameter m (which is also clear as each population is an independent random process). It is noteworthy, that the average runtime for $\mathcal{J}^{10,g_{0.75},f_1}_{hs}$ is about 66.1 seconds for knowledge bases with 50000 formulas. As the significance of the parameter m for the accuracy is also only marginal, the measure $\mathcal{J}^{10,g_{0.75},f_1}_{hs}$ is clearly an effective and accurate stream-based inconsistency measure.

6 Discussion and Conclusion

In this paper we discussed the issue of large-scale inconsistency measurement and proposed novel approximation algorithms that are effective for the streaming case. To the best of our knowledge, the computational issues for measuring inconsistency, in partic-



(a) Accuracy $\mathcal{J}_{hs}^{m,g_{0.75},f_1}$

(b) Accuracy $\mathcal{J}_{c}^{m,g_{0.75},f_1}$

(c) Accuracy $\mathcal{J}_{\mathcal{I}_{\mathsf{MI}}}^{w,\max}$

(d) Accuracy $\mathcal{J}_{\mathcal{I}_{\mathsf{MI}}^c}^{w,\max}$

Fig. 1: Accuracy performance for the evaluated measures (dashed line is actual inconsistency value); each value is averaged over 100 random knowledge bases of 5000 formulas (30 propositions) with varying inconsistency values

ular with respect to scalability problems, have not yet been addressed in the literature before. One exception is the work by Ma and colleagues [7] who present an anytime algorithm that approximates an inconsistency measure based on a 4-valued paraconsistent logic (similar to the contension inconsistency measure). The algorithm provides lower and upper bounds for this measure and can be stopped at any point in time with some guaranteed quality. The main difference between our framework and the algorithm of [7] is that the latter needs to process the whole knowledge base in each atomic step and is therefore not directly applicable for the streaming scenario. The empirical evaluation [7] also suggests that our streaming variant of $\mathcal{I}_{hs}$ is much more performant as Ma et al. report an average runtime of their algorithm of about 240 seconds on a knowledge base with 120 formulas and 20 propositions (no evaluation on larger knowledge bases is given) while our measure has a runtime of only a few seconds for knowledge bases with 5000 formulas with comparable accuracy[10]. A deeper comparison of these different approaches is planned for future work.

Our work showed that inconsistency measurement is not only a theoretical field but can actually be applied to problems of reasonable size. In particular, our stream-based

[10] Although hardware specifications for these experiments are different this huge difference is significant.

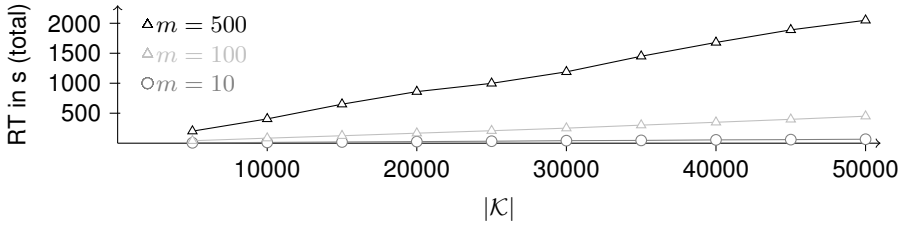


Fig. 2: Evaluation of the scalability of $\mathcal{J}_{hs}^{m,g_{0.75},f_1}$; each value is averaged over 10 random knowledge bases of the given size

approaches of $\mathcal{I}_{hs}$ and $\mathcal{I}_c$ are accurate and effective for measuring inconsistencies in large knowledge bases. Current and future work is about the application of our work on linked open data sets [12].

References

1. Grant, J., Hunter, A.: Measuring inconsistency in knowledgebases. Journal of Intelligent Information Systems **27** (2006) 159–184
2. Makinson, D.: Bridges from Classical to Nonmonotonic Logic. College Publications (2005)
3. Priest, G.: Logic of Paradox. Journal of Philosophical Logic **8** (1979) 219–241
4. Hansson, S.O.: A Textbook of Belief Dynamics. Kluwer Academic Publishers (2001)
5. Grant, J., Hunter, A.: Measuring consistency gain and information loss in stepwise inconsistency resolution. In: Proc. of the 11th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2011). (2011) 362–373
6. Knight, K.M.: A Theory of Inconsistency. PhD thesis, University Of Manchester (2002)
7. Ma, Y., Qi, G., Xiao, G., Hitzler, P., Lin, Z.: An anytime algorithm for computing inconsistency measurement. In: Knowledge Science, Engineering and Management. Springer (2009) 29–40
8. Grant, J., Hunter, A.: Distance-based Measures of Inconsistency. In: Proceedings of the 12th Europen Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'13). (2013) 230–241
9. Hunter, A., Konieczny, S.: On the measure of conflicts: Shapley inconsistency values. Artificial Intelligence **174**(14) (July 2010) 1007–1026
10. Thimm, M.: Inconsistency measures for probabilistic logics. Artificial Intelligence **197** (April 2013) 1–24
11. Jabbour, S., Ma, Y., Raddaoui, B.: Inconsistency measurement thanks to mus decomposition. In: Proc. of the 13th Int. Conference on Autonomous Agents and Multiagent Systems. (2014)
12. Isele, R., Umbrich, J., Bizer, C., Harth, A.: LDSpider: An open-source crawling framework for the web of linked data. In: Proceedings of 9th International Semantic Web Conference (ISWC 2010) Posters and Demos. (2010)
13. Lawrence, D.: Genetic Algorithms and Simulated Annealing. Pitman Publishing (1987)
14. Thimm, M.: Tweety - A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation. In: Proceedings of the 14th Int. Conference on Principles of Knowledge Representation and Reasoning (KR'14). (2014)