General information spaces: measuring inconsistency, rationality postulates, and complexity

John Grant¹ · Francesco Parisi²

Accepted: 16 March 2021 / Published online: 27 April 2021
© The Author(s) 2021

Abstract

AI systems often need to deal with inconsistent information. For this reason, since the early 2000s, some AI researchers have developed ways to measure the amount of inconsistency in a knowledge base. By now there is a substantial amount of research about various aspects of inconsistency measuring. The problem is that most of this work applies only to knowledge bases formulated as sets of formulas in propositional logic. Hence this work is not really applicable to the way that information is actually stored. The purpose of this paper is to extend inconsistency measuring to real world information. We first define the concept of *general information space* which encompasses various types of databases and scenarios in AI systems. Then, we show how to transform any general information space to an *inconsistency equivalent* propositional knowledge base, and finally apply propositional inconsistency measures to find the inconsistency of the general information space. Our method allows for the direct comparison of the inconsistency of different information spaces, even though the data is presented in different ways. We demonstrate the transformation on four general information spaces: a relational database, a graph database, a spatio-temporal database, and a Blocks world scenario, where we apply several inconsistency measures after performing the transformation. Then we review so-called rationality postulates that have been developed for propositional knowledge bases as a way to judge the intuitive properties of these measures. We show that although general information spaces may be nonmonotonic, there is a way to transform the postulates so they can be applied to general information spaces and we show which of the measures satisfy which of the postulates. Finally, we discuss the complexity of inconsistency measures for general information spaces.

Keywords General information space · Inconsistency measures · Rationality postulates

✉ Francesco Parisi
fparisi@dimes.unical.it

John Grant
grant@cs.umd.edu

¹ University of Maryland at College Park, College Park, MD, USA

² DIMES Department, University of Calabria, Calabria, Italy

1 Introduction

As AI systems may need to deal with inconsistency, some AI researchers started in the early 2000s to develop ways of measuring the inconsistency of a propositional knowledge base, that is, a set of formulas in propositional logic. By now a substantial amount of work has been done along these lines, which includes for instance the approaches developed in [35–38, 40, 42, 45, 47, 55, 56, 62]. A survey on the topic can be found in [26].

But information in many cases is not restricted to propositional logic formulas. In this paper we show that much of the work done for measuring inconsistency in propositional logic knowledge bases can be applied to measure inconsistency in more complex frameworks, where real world information is actually stored. Our approach is as follows.

- We first introduce the concept of a *general information space* that covers various types of databases and AI frameworks. A general information space consists of a *framework*, such as a database schema, a set of *information units*, such as tuples in a relation, and a set of *requirements*, such as integrity constraints.
- Then, we formulate a *transformation* from a general information space to a propositional knowledge base in such a way that the inconsistencies are preserved. As expected, the propositional knowledge base resulting from the transformation does not capture the rich content of the general information space, but it does capture all of its inconsistencies in a precise way.
- Finally, after the transformation we apply a propositional inconsistency measure to the just obtained propositional knowledge base to find the inconsistency of the general information space.

Hence, our approach allows for measuring the inconsistency of any information space and the method can be applied in a uniform manner over a variety of information scenarios. To the best of our knowledge, this is the first approach that lifts the idea of inconsistency measure from propositional knowledge bases to a range of different frameworks used for storing real world data. This makes it possible to apply all the results about inconsistency measures for propositional knowledge bases to a wide range of applications. We believe that this is a significant advance for the whole idea of inconsistency measurement that has been developed for propositional knowledge bases, and explored in several other individual settings such as software specifications [46], databases [4, 14, 44, 51], and ontologies [63, 64], among others.

The plan of this paper is as follows. We start by giving some basic information and examples of inconsistency measures for propositional knowledge bases in Section 2. Then, we define the concept of a general information space in Section 3. This is followed by the steps of the transformation and the equivalence of the two for inconsistencies in Section 4. Section 5 contains four examples of general information spaces: a relational database [11] (Section 5.1), a graph database [54] (Section 5.2), a spatio-temporal database [53] (Section 5.3), and a Blocks world scenario [32] (Section 5.4). Our method allows for the important task of evaluating and comparing the inconsistency [23], even for such different ways of storing information. Then we introduce rationality postulates for propositional inconsistency measures in Section 6.1. We show how to modify these postulates in order to check them for general information spaces in Section 6.2. We also construct a table describing the various new postulates satisfied by our inconsistency measures and contrast this with the satisfaction of these measures for propositional knowledge bases in Section 6.3. Finally, in Section 7, we deal with complexity and show that in some cases,

unlike for propositional knowledge bases, the inconsistency measures for general information spaces can be computed in polynomial time. We draw conclusions and outline future work in Section 8.

This paper is a substantially revised and extended version of [31]. We have made the following major modifications. In Section 2, we added a discussion of two classifications for inconsistency measures and added an eighth inconsistency measure. In Section 3 we clarified several important issues concerning the definition of a general information space. We substantially extended Section 5. We clarified several issues concerning the graph database of Section 5.2 and added the constraints in first-order logic. Section 5.3 is completely new; here we discuss a specific spatio-temporal database as a general information space and measure its inconsistency. We also clarified several issues concerning the Blocks world database of Section 5.4 and added the constraints in first-order logic. Sections 6 and 7, where we explore postulates satisfaction and complexity, respectively, are completely new. Finally, Section 8 was rewritten to account for the changed scope of this paper.

2 Brief background on inconsistency measures for propositional knowledge bases

The idea of an inconsistency measure is to assign a nonnegative number to a knowledge base that measures its inconsistency. We start with a propositional language of formulas composed from a countable set of atoms, the fundamental propositions, and the connectives $\wedge$ (conjunction), $\vee$ (disjunction), and $\neg$ (negation). We write $\mathcal{K}$ for the set of all propositional knowledge bases (KBs), i.e. the set of all finite sets of formulas in the language. We write K for an individual KB. 2^X is the set of all subsets (the power set) of any set X . In general, an inconsistency measure assigns each KB a nonnegative real number or infinity.

Definition 1 A function $I : \mathcal{K} \rightarrow \mathbb{R}_{\infty}^{\geq 0}$ is an *inconsistency measure* iff the following Consistency condition holds for all $K, K' \in \mathcal{K}$:

1. *Consistency.* $I(K) = 0$ iff K is consistent.

Consistency is called a (rationality) postulate. It means that all and only consistent knowledge bases get the measure 0, and hence all inconsistent knowledge bases get a positive measure.. In Section 6.1 we will introduce additional postulates. These are properties that conform to some intuitive notion of how an inconsistency measure should behave. However, the only property that is accepted by all researchers is Consistency.

For a knowledge base K , $\text{MI}(K)$ is the set of minimal inconsistent subsets of K . Also, if $\text{MI}(K) = \{M_1, \dots, M_n\}$ then $\text{Problematic}(K) = M_1 \cup \dots \cup M_n$, and $\text{Free}(K) = K \setminus \text{Problematic}(K)$. A *free* formula is not involved in an essential way in any inconsistency, while a *problematic* formula is so involved in at least one inconsistency. As an example, for $K_{ex} = \{a_1, a_2, a_3, a_4, \neg a_1 \vee \neg a_2, \neg a_2 \vee \neg a_3, a_4 \wedge a_5\}$, $\text{MI}(K_{ex}) = \{\{a_1, a_2, \neg a_1 \vee \neg a_2\}, \{a_2, a_3, \neg a_2 \vee \neg a_3\}\}$, and $\text{Problematic}(K_{ex}) = \{a_1, a_2, a_3, \neg a_1 \vee \neg a_2, \neg a_2 \vee \neg a_3\}$, while a_4 and $a_4 \wedge a_5$ are free formulas.

A classical interpretation i for K assigns each atom a that appears in a formula of K the truth value T (true) or F (false), that is, $i : \text{Atoms}(K) \rightarrow \{T, F\}$, where $\text{Atoms}(K)$ denotes the set of atoms of K . However, there is an important propositional inconsistency measure, I_C , that uses 3 truth values: T , F , and B (both), where B indicates inconsistency. This measure uses Priest's 3-valued logic. This interpretation uses an ordering on the truth

values where $F < B < T$ and $\wedge$ computes the minimum value while $\vee$ computes the maximum value; also $\neg(B) = B$. So, for example, $B \wedge F = F$ and $B \vee F = B$. Then, an interpretation i satisfies a formula iff the truth-value of the formula for i is T or B .

Now we are ready to define the propositional inconsistency measures we will consider in this paper. Below the definition we briefly explain the meanings of these measures.

Definition 2 (Propositional Inconsistency Measures) For a knowledge base K , the inconsistency measures I_B , I_M , $I_\#$, I_P , I_H , I_{nc} , I_C , and I_{mv} are such that

1. $I_B(K) = 1$ if K is inconsistent and $I_B(K) = 0$ if K is consistent.
2. $I_M(K) = |\text{MI}(K)|$.
3. $I_\#(K) = \begin{cases} 0 & \text{if } K \text{ is consistent,} \\ \sum_{X \in \text{MI}(K)} \frac{1}{|X|} & \text{otherwise.} \end{cases}$
4. $I_P(K) = |\text{Problematic}(K)|$.
5. $I_H(K) = \min\{|X| \mid X \subseteq K \text{ and } \forall M \in \text{MI}(K) (X \cap M \neq \emptyset)\}$.
6. $I_{nc}(K) = |K| - \max\{n \mid \forall K' \subseteq K : |K'| = n \text{ implies that } K' \text{ is consistent}\}$.
7. $I_C(K) = \min\{|i^{-1}(B)| \text{ such that } i \text{ satisfies every formula in } K\}$.
8. $I_{mv}(M) = \frac{|\text{Atoms}(\text{Problematic}(K))|}{|\text{Atoms}(K)|}$.

We explain the measures as follows. I_B is also called the drastic measure [36]: it simply distinguishes between consistent and inconsistent KBs. I_M counts the number of minimal inconsistent subsets [36]. $I_\#$ also counts the number of minimal inconsistent subsets, but it gives larger sets a smaller weight; the reason is that when a minimal inconsistent set contains more formulas than another minimal inconsistent set, the former is intuitively less inconsistent than the latter [36]. I_P counts the number of formulas that contribute essentially to one or more inconsistencies [24]. I_H counts the minimal number of formulas whose deletion makes the set consistent [25]. I_{nc} uses the largest number such that all sets with that many formulas are consistent [17]. I_C counts the minimal number of atoms that must be assigned the truth-value B in the three-valued logic by an interpretation that satisfies every formula in the KB [24]. Finally, I_{mv} is the ratio of the number of atoms in the problematic formulas to the total number of atoms [62].

For the KB K_{ex} , we have the following results.

1. $I_B(K_{ex}) = 1$ as it is inconsistent.
2. $I_M(K_{ex}) = 2$ as there are 2 minimal inconsistent subsets.
3. $I_\#(K_{ex}) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$ as both minimal inconsistent subsets consist of 3 formulas.
4. $I_P(K_{ex}) = 5$ as there are 5 problematic formulas.
5. $I_H(K_{ex}) = 1$ as deleting a_2 suffices to make K_{ex} consistent.
6. $I_{nc}(K_{ex}) = 7 - 2 = 5$ as 2 is the largest number such that all subsets of size 2 are consistent.
7. $I_C(K_{ex}) = 1$ as the following interpretation satisfies all the formulas: $i(a_1) = i(a_3) = i(a_4) = i(a_5) = T$, $i(a_2) = B$.
8. $I_{mv}(K_{ex}) = \frac{3}{5}$ as 3 of the 5 atoms appear in problematic formulas.

There have been several proposals to classify inconsistency measures. We will deal with two classifications. It should be noted that typically a classification does not encompass all inconsistency measures; rather, it is used to distinguish certain features of the measures.

One classification distinguishes between *syntactic* and *semantic* measures. Syntactic measures are based on the syntax in the sense of the interaction of individual formulas with the minimal inconsistent subsets. On the other hand, a semantic measure takes into account

the meaning of each formula by the truth values of its component atoms. A recent paper, [12], studies this issue in detail. Among the measures given above, the first six are syntactic measures and I_C is a semantic measure. I_{mv} does not fit into either category because it requires breaking the formulas into their atomic components without taking into account the truth values of these atoms.

Another classification distinguishes between *absolute* and *relative* measures. The idea is that an absolute measure measures the totality of the inconsistency, while a relative measure measures the proportion of the inconsistency. The bulk of the research on inconsistency measures considered only absolute measures, but a recent paper [6] studies specifically the properties of relative measures. Among the measures given above, the second through the seventh are absolute measures and I_{mv} is a relative measure. I_B does not fit into either category because it does not measure a totality (it cannot increase after 1) and although its value is between 0 and 1, as is the case for relative measures, it is not a ratio.

3 General information spaces

Our goal is to apply the idea of measuring inconsistency in propositional knowledge bases to more complex scenarios that are useful in AI and databases. Intuitively, a general information space provides specific information concerning some subject matter according to a given format. The underlying language is first-order logic where various concepts such as sets, relationships, graphs, etc. can be expressed.

Definition 3 A general information space $S = \langle F, U, C \rangle$ is a triple where F is the framework for the information, U is a set of information units, and C is a set of requirements that U must satisfy, where the following hold:

- A1 (*Specification of the frame*). The framework F defines all the structures (represented in first-order logic as relations) as well as the domains for all the attributes of the relations.
- A2.1 (*Consistency and unity of the information units*). Each information unit in U is a ground atom.
- A2.2 (*Negative information*). Negative information is handled by the Closed World Assumption.
- A3.1 (*Consistency of the requirements*). C is consistent.
- A3.2 (*Arity of the requirements*). Each requirement has an arity which is a non-negative integer. A positive arity, k , means that there must be k information units involved in an inconsistency with that requirement. The arity 0 indicates an existential constraint where an inconsistency arises because some required information is not in U .
- A3.3 (*Restrictions on the requirements*). We may restrict the requirements to allow only certain kinds of formulas.
- A4.1 (*Inconsistencies*). All inconsistencies of S arise from the interaction of U and one element of C .
- A4.2 (*Finding inconsistencies*). There is an effective procedure to find all inconsistencies of S .

We will also use the symbol S for a set, namely $S = U \cup C$. In such a case F is specified either explicitly or implicitly.

A relational database is an example of a general information space. Here the framework is the database schema as well as the language used to describe the database and the domains for the constants. The information units are the tuples in the relations of the schema. The set of requirements is the set of integrity constraints.

A graph database is another example. Here the framework is the type of information stored in the vertices and edges. The information units are the information given by the vertices and edges. The requirements are the graph constraints.

A spatio-temporal database provides a third example. In this case the framework provides the objects, times, and spatial points in the database. Each information unit indicates that a particular object is at some point in space at some time value. The requirements are the constraints that specify what is not allowed, such as, that an object cannot be at two different points at the same time.

A Blocks world is a general information space as well. Here the framework is the type of information stored about blocks such as block color and which block is on top of another block. The elements of the domain are also part of the framework. The information units describe the Blocks world such as giving the colors of the blocks and which block is on top of another block. The requirements are the rules of the Blocks world such as that a green block cannot be on top of a blue block.

Another example is a board game configuration. Here the framework is information about the board and the pieces. The information units describe the position of each piece and some action or actions. The requirements are the rules of the game, such as how the pieces may move and what constitutes a winning position.

Thus a general information space encompasses many ways in which information is stored and presented; we will give four concrete examples in Section 5.

It is worth noting that A1, A2, A3, and A4 hold in many real world scenarios, such as those mentioned earlier. For instance, for relational databases, tuples are units of information that are usually assumed to be consistent when considered alone (without interacting with the integrity constraints), integrity constraints are usually satisfiable (there exists a database instance that satisfies them), and procedures for checking inconsistency are well-known for large classes of integrity constraints [1, 60].

A requirement for a general information space is really a constraint but we use this terminology to indicate that there need not be a formal language in which the requirement is presented and we will also use English in some examples.

Next we give some details concerning requirements for relational databases, and illustrate the notion of arity for them.¹ Consider a relational database with two relations, a binary relation R_1 and a ternary relation R_2 . As is usual we call a requirement a constraint. Consider the constraint: $\neg R_1(1, 2)$. This constraint specifically excludes the tuple $(1, 2)$ from R_1 . Hence the existence of that tuple with the constraint is inconsistent. Thus, the arity of this constraint is 1. Consider the more general constraint: $\forall x_1 x_2 x_3 [R_1(x_1, x_2) \wedge R_1(x_1, x_3) \rightarrow x_2 = x_3]$. This states the functional dependency of the second attribute of R_1 on the first. This would be violated, for instance, by the two tuples: $(1, 2)$ and $(1, 3)$ in R_1 causing an inconsistency. So the arity of this constraint is 2. Next consider the constraint: $\forall x_1 x_2 [R_1(x_1, x_2) \rightarrow \exists x_3 x_4 (R_2(x_2, x_3, x_4))]$. This states the inclusion dependency that the elements in the second column of R_1 are included in the first column of R_2 . A violation of

¹We assume the reader is familiar with the classical notions of database scheme, relation scheme, relation instance, and integrity constraint. We will recall these notions in Section 5.1.

this constraint is caused by a single tuple in R_1 whose second element is not in the first column of R_2 . This means that the arity is 1. It is worth pointing out an important difference between this case and the previous two cases. Take the first constraint, $\neg R_1(1, 2)$, the functional dependency case is similar. The existence in R_1 of the tuple $(1, 2)$, that is, $R_1(1, 2)$ together with the constraint $\neg R_1(1, 2)$ form an inconsistent set of formulas. But in the inclusion dependency the existence of some tuple in R_1 , say $R_1(1, 2)$ together with the inclusion dependency is not inconsistent: there may be a tuple in R_2 , say $R_2(2, 3, 4)$ that would satisfy the constraint. Hence for this type of constraint the context is needed, such as what tuples are in R_2 . In any case, all the usual database constraints such as the various types of dependencies, denial and key constraints are included in our concept of a requirement.

There is also a somewhat different type of constraint. Consider that the requirement $R_1(1, 2)$ requires the tuple $(1, 2)$ to be in R_1 . So in this case it is a lack of the tuple that violates the requirement and causes an inconsistency. A more general example is $\exists x_1 x_2 R_2(1, x_1, x_2)$ which states that there must be a tuple in the R_2 relation whose first element is 1. Note how these constraints are similar to the inclusion dependency given above: in all these cases a required tuple is not in the database. The difference is that for an inclusion dependency we can point to a tuple in R_1 that is the source of the problem, i.e., without that tuple there would not be a constraint violation. So for an inclusion dependency it is possible to delete a tuple from R_1 that is the source of the problem. But for the constraints just given, no deletion from the database resolves the violation. We call such a constraint an *existential* constraint and assign it arity 0.

In our approach the violations of requirements are ascribed to information units appearing in the general information space. This is similar to what is done for instance in database repairing approaches that aim at restoring consistency by tuple deletions only, even in the presence of existential constraints' violations [10, 61]. As a matter of fact, for large classes of requirements, e.g., denial constraints, violations can only be ascribed to the *presence* of some information units. However, violations of existential constraints are due to missing information units. We refer the reader to Section 8 for further discussion on the treatment of existential requirements.

4 The transformation

We now show how any general information space can be transformed to a propositional knowledge base in such a way that all the violations of the requirements are inconsistencies in the knowledge base. Note however that the transformation also loses some information: there is no way to go back from the propositional knowledge base to the original general information space. In fact many different information spaces representing different phenomena may be transformed to the same knowledge base. But the transformation is appropriate if we are interested in measuring inconsistency.

Definition 4 (Transformation) The transformation from a general information space $S = \langle F, U, C \rangle$ to a propositional KB K_S is as follows.

- Let $A_U = \{a_1, \dots, a_{|U|}\}$ be a set of $|U|$ propositional atoms.
- Define a bijective function $f : U \rightarrow A_U$ that assigns a distinct propositional atom to each information unit in U .
- Let $B_C = \{b_1, \dots, b_{|C|}\}$ be another set of $|C|$ propositional atoms.

- Define a bijective function $h : C \rightarrow B_C$ that assigns a distinct propositional atom to each requirement in C .
- Let $\mathcal{F}_S$ be the set of propositional formulas using $A_U \cup B_C$.
- Define a function $g : C \rightarrow \mathcal{F}_S$ as follows: For each requirement $c \in C$ do as follows.
 - 1) If there is no violation of the requirement, then set $g(c) = h(c)$. Otherwise, there is at least one violation of c .
 - 2) If the arity of c is greater than 0, then a minimal inconsistency is formed by one or more information units together with c . Find all such sets, say $M_c = \{U_1, \dots, U_k\}$ and suppose that $|U_i| = n$. Let $U_i = \{u_i^1, \dots, u_i^n\}$ (where each u_i^j is an information unit). Define $\rho(U_i) = \neg f(u_i^1) \vee \dots \vee \neg f(u_i^n)$ which is a propositional logic formula. Then, define

$$g(c) = \left(\bigwedge_{U_i \in M_c} \rho(U_i) \right) \wedge h(c).$$
 - 3) When the arity of c is 0, define $g(c) = \neg h(c) \wedge h(c)$.
- Define $K_S = \{f(u) \mid u \in U\} \cup \{g(c) \mid c \in C\}$.

Clearly g is one-to-one because each $g(c)$ is identified with its corresponding unique $h(c)$. Next we show the equivalence between the violation of the requirements C for S and the minimal inconsistent subsets of K_S . A requirement violation causes an inconsistency for S but we need a definition for it.

Definition 5 (Inconsistency of a general information space) An *inconsistency* of S consists of one of two cases:

- (1) The arity of the requirement c is a positive number k . In this case an inconsistency of S is a set of k information units, $\{u_1, \dots, u_k\}$, that violates c . We write such an inconsistency as $\{u_1, \dots, u_k, c\}$.
- (2) The arity of the requirement c is 0. If c is violated by S , there is an inconsistency written as $\{c, \neg c\}$.

Then we define $Inc(S)$ as the set of inconsistencies of S . We say that S is *consistent* iff $Inc(S) = \emptyset$.

Theorem 1 A general information space S and its transformation to a propositional knowledge base K_S are equivalent for inconsistencies in the sense that there is a bijection $m : Inc(S) \rightarrow \text{MI}(K_S)$. Furthermore, for $M \in Inc(S)$, $|M| = |m(M)|$.

Proof ($\Rightarrow$) Let $M \in Inc(S)$. Every inconsistency of S contains exactly one requirement. Suppose the arity of the requirement c is a positive integer k . In this case the inconsistency contains k elements from U , say $u_1, \dots, u_k$, and a requirement $c \in C$. Then, according to the construction, $g(c)$ is a propositional formula in CNF, one of whose conjuncts is $\neg f(u_1) \vee \dots \vee \neg f(u_k)$. Hence $\{f(u_1), \dots, f(u_k), g(c)\} \in \text{MI}(K_S)$. Thus, in this case $m(\{u_1, \dots, u_k, c\}) = \{f(u_1), \dots, f(u_k), g(c)\}$. It is clear from the construction that $|M| = |m(M)|$. In the case where the arity of c is 0, according to the construction, $g(c) = \neg h(c) \wedge h(c)$, which is also a minimal inconsistent set (of size 1) in K_S . Thus, in this case $m(\{c, \neg c\}) = \{\neg h(c), h(c)\}$.

($\Leftarrow$) Let $M \in \text{MI}(K_S)$. Based on the structure of the transformation with the information units transformed to atoms and only the constraints transformed to formulas that may involve negation, there are two cases. In the first case $M = \{a_1, \dots, a_k, g(c)\}$ where $\neg a_1 \vee \dots \vee \neg a_k$ is a conjunct in $g(c)$. As g is one-to-one, g^{-1} exists and $g^{-1}(g(c)) = c$. Therefore $m^{-1}(M) = \{f^{-1}(a_1), \dots, f^{-1}(a_k), c\} \in \text{Inc}(S)$. The other case is where $M = \{\neg b_i \wedge b_i\}$ for some i , $1 \leq i \leq |C|$. This means that $m^{-1}(M) = \{g^{-1}(M)\}$ which is a requirement having arity equal to zero. In both cases the construction gives $|M| = |m^{-1}(M)|$. $\square$

Our approach to measuring the inconsistency of a general information space S according to a propositional inconsistency measure I_x is to apply the inconsistency measure to the transformed space, that is, we define $I_x(S) = I_x(K_S)$.

5 Examples of general information spaces

We will describe four examples to illustrate our approach. In each case we first describe the general information space, then do the transformation, and finally compute the inconsistency measures according to the propositional inconsistency measures of Definition 2.

5.1 A relational database as a general information space

A *relation scheme* $R(A_1, \dots, A_n)$ consists of a relation name R and a sorted list of attribute names $A_1, \dots, A_n$, where each attribute A_i (with $i \in [1..n]$) has associated a domain $\text{DOM}(A_i)$. A *database scheme* $\mathcal{DS}$ is a nonempty finite set of relation schemes. A *tuple* $t = (v_1, \dots, v_n)$ over $R(A_1, \dots, A_n)$ is a mapping assigning to each attribute A_i of R a value $v_i \in \text{DOM}(A_i)$. A *relation instance* is a set of tuples over a given relation scheme, and a *database instance* D is a set of relation instances over a given database scheme. We use e_i for terms, that is, elements of the domains or variables. An atom over a database scheme $\mathcal{DS}$ is an expression of one of the following forms: (a) $R(e_1, \dots, e_n)$ where R is a relation scheme in $\mathcal{DS}$, or (b) $e_i \circ e_j$ where $\circ \in \{=, \neq, >, <, \geq, \leq\}$. An *integrity constraint* over $\mathcal{DS}$ is any (function-free) first-order sentence over the database scheme $\mathcal{DS}$. For a database scheme $\mathcal{DS}$ and a set $\mathcal{C}$ of integrity constraints over $\mathcal{DS}$, an instance D of $\mathcal{DS}$ is said to be *consistent* w.r.t. $\mathcal{C}$ (or, equivalently, $\mathcal{C}$ is *satisfied* by D , $\mathcal{C}$ is not *violated* by D) iff $D \models \mathcal{C}$ in the standard model-theoretic sense.

The components of the general information space $S = \langle F, U, C \rangle$ for a relational database instance D over the database scheme $\mathcal{DS}$ with a set $\mathcal{C}$ of integrity constraints are as follows. The framework F is the database scheme $\mathcal{DS}$ and the (function-free) first-order language using a set of uninterpreted constants and predicate symbols for relation names, as well as domains of the attributes for the evaluation of constants. The set U of information units is the instance D (the set of the tuples in the relation instances), and the set C of requirements is the set $\mathcal{C}$ of integrity constraints.

We now provide an example. Let the framework F be the database scheme consisting of the relation schemes *Asset*(*SN*, *Date Loaned*, *Employee*, *Date Returned*) whose instance contains the serial number, the loan date, the employee's identifier, and the returned date of assets provided by a company to the employees, about whom information is stored in two relations: *Employee*(*SSN*, *Name*, *HiringDate*), and *Family*(*SSN*, *Child*, *Project*). Here U ,

the database instance, is shown in the usual tabular form in Fig. 1. Altogether there are 13 information units (tuples) that for convenience we name t_i , with $1 \leq i \leq 13$.

There are 8 requirements in C as given below both as first-order logic formulas and in English.

- $c_1 = \forall x_1 \dots x_4 [Asset(x_1, x_2, x_3, x_4) \rightarrow x_2 \leq x_4]$, stating that, for every asset, the loan date cannot be later than the return date.
- $c_2 = \forall x_1 \dots x_7 [Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_1, x_5, x_6, x_7) \rightarrow (x_2 = x_5 \wedge x_3 = x_6 \wedge x_4 = x_7)]$, i.e. the constraint that the serial number is a key for *Asset*.
- $c_3 = \forall x_1 \dots x_8 [Asset(x_1, x_2, x_3, x_4) \wedge Asset(x_5, x_2, x_3, x_6) \wedge Asset(x_7, x_2, x_3, x_8) \rightarrow (x_1 = x_5 \vee x_1 = x_7 \vee x_5 = x_7)]$, stating the numerical dependency [29, 30] $DateLoaned, Employee \rightarrow^2 SN$ whose meaning is that for every date and employee there can be at most 2 assets loaned.
- $c_4 = \forall x_1 \dots x_5 [Employee(x_1, x_2, x_3) \wedge Employee(x_1, x_4, x_5) \rightarrow (x_2 = x_4 \wedge x_3 = x_5)]$, stating that *ID* is a key for *Employee*.
- $c_5 = \forall x_1 \dots x_4 [Employee(x_1, x_2, x_3) \wedge Employee(x_4, x_2, x_3) \rightarrow x_1 = x_4]$, that is, the pair of attributes *Name* and *HiringDate* also form a key for *Employee*.
- $c_6 = \forall x_1 \dots x_4 [Asset(x_1, x_2, x_3, x_4) \rightarrow \exists x_5, x_6 Employee(x_3, x_5, x_6)]$ i.e., the inclusion dependency $Asset[Employee] \subseteq Employee[ID]$.
- $c_7 = \forall x_1 \dots x_5 [Family(x_1, x_2, x_3) \wedge Family(x_1, x_4, x_5) \rightarrow Family(x_1, x_2, x_5)]$, i.e. the multivalued dependency [19] $Family: ID \twoheadrightarrow Child$.
- $c_8 = \exists x_1 \dots x_6 [Family(x_1, x_2, x_3) \wedge Family(x_4, x_5, x_6) \wedge x_1 \neq x_4]$ stating that there must be at least two employees referenced in the *Family* relation.

		Asset				
Atom		<i>SN</i>	<i>DateLoaned</i>	<i>Employee</i>	<i>DateReturned</i>	Tuple
a_1		999	2015-02-01	123456789	2016-03-15	t_1
a_2		999	2015-02-01	123456789	2018-12-31	t_2
a_3		999	2013-06-15	222222222	2017-12-31	t_3
a_4		888	2016-12-01	222222222	2013-12-01	t_4
a_5		555	2014-07-01	333333333	2013-06-20	t_5
a_6		666	2014-07-01	333333333	2015-09-10	t_6
a_7		777	2014-07-01	333333333	2014-05-21	t_7

		Employee			
Atom		<i>ID</i>	<i>Name</i>	<i>HiringDate</i>	Tuple
a_8		333333333	Robert	1980-01-01	t_8
a_9		444444444	William	1975-06-01	t_9
a_{10}		123456789	William	1975-06-01	t_{10}

		Family			
Atom		<i>ID</i>	<i>Child</i>	<i>Project</i>	Tuple
a_{11}		123456789	Steve	Q1	t_{11}
a_{12}		123456789	Mary	Q2	t_{12}
a_{13}		123456789	Steve	Q2	t_{13}

Fig. 1 Instances of *Asset*, *Employee*, and *Family*

5.1.1 Transformation to a propositional knowledge base

We now show how the transformation from a general information space to a propositional knowledge base is applied for this relational database.

1. $A_U = \{a_1, \dots, a_{13}\}$ corresponding to the 13 tuples.
2. $f(t_i) = a_i$ for all i , $1 \leq i \leq 13$.
3. $B_C = \{b_1, \dots, b_8\}$ corresponding to the 8 constraints.
4. $h(c_i) = b_i$ for all i , $1 \leq i \leq 8$.
5. $\mathcal{F}_S$ is the set of propositional formulas using $A_U \cup B_C$.
6. Now we show the mapping g by going over the constraints one at a time.
 - c_1 The arity of c_1 is 1. The 3 tuples t_4 , t_5 , and t_7 each violate c_1 . Hence, $g(c_1) = \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1$.
 - c_2 The arity of c_2 is 2. The 3 tuples t_1 , t_2 , and t_3 all have the same serial number but are not identical. Hence, $g(c_2) = (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2$.
 - c_3 The arity of c_3 is 3. The three tuples t_5 , t_6 , and t_7 together violate this constraint. Hence, $g(c_3) = (\neg a_5 \vee \neg a_6 \vee \neg a_7) \wedge b_3$.
 - c_4 This constraint is satisfied. Hence, $g(c_4) = b_4$.
 - c_5 The arity of c_5 is 2. It is violated by the pair t_9 and t_{10} . Hence, $g(c_5) = (\neg a_9 \vee \neg a_{10}) \wedge b_5$.
 - c_6 The arity of c_6 is 1. It is violated separately by t_3 and t_4 . Hence, $g(c_6) = \neg a_3 \wedge \neg a_4 \wedge b_6$.
 - c_7 The arity of c_7 is 2. It is violated by the pair t_{11} and t_{12} . Hence, $g(c_7) = (\neg a_{11} \vee \neg a_{12}) \wedge b_7$.
 - c_8 The arity of c_8 is 0 and it is violated by U . Hence, $g(c_8) = \neg b_8 \wedge b_8$.
7. Therefore $K_S = \{a_1, \dots, a_{13}, \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2, (\neg a_5 \vee \neg a_6 \vee \neg a_7) \wedge b_3, b_4, (\neg a_9 \vee \neg a_{10}) \wedge b_5, \neg a_3 \wedge \neg a_4 \wedge b_6, (\neg a_{11} \vee \neg a_{12}) \wedge b_7, \neg b_8 \wedge b_8\}$.

5.1.2 The calculation of the inconsistency measures

Below are the results of calculating the inconsistency measures of the relational database example. We use the fact that

$MI(K_S) = \{\{a_4, \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1\}, \{a_5, \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1\}, \{a_7, \neg a_4 \wedge \neg a_5 \wedge \neg a_7 \wedge b_1\}, \{a_1, a_2, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2\}, \{a_1, a_3, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2\}, \{a_2, a_3, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_3) \wedge b_2\}, \{a_5, a_6, a_7, (\neg a_5 \vee \neg a_6 \vee \neg a_7) \wedge b_3\}, \{a_9, a_{10}, (\neg a_9 \vee \neg a_{10}) \wedge b_5\}, \{a_3, \neg a_3 \wedge \neg a_4 \wedge b_6\}, \{a_4, \neg a_3 \wedge \neg a_4 \wedge b_6\}, \{a_{11}, a_{12}, (\neg a_{11} \vee \neg a_{12}) \wedge b_7\}, \{\neg b_8 \wedge b_8\}\}$.

- $I_B(S) = 1$ as K_S is inconsistent.
- $I_M(S) = 12$ as there are 12 minimal inconsistent subsets for K_S .
- $I_{\#}(S) = 1 + 5 \times \frac{1}{2} + 5 \times \frac{1}{3} + \frac{1}{4} = \frac{65}{12}$ as there is one minimal inconsistent subset of size 1, 5 of size 2, 5 of size 3, and 1 of size 4 in K_S .
- $I_P(S) = 11 + 7 = 18$ as 11 atoms (i.e., tuples) plus 7 propositional formulas (i.e., constraints) are problematic in K_S .
- $I_H(S) = 7$ as the deletion of the 7 formulas of $g(c_i)$ for all i , $1 \leq i \leq 3$ and $5 \leq i \leq 8$ makes K_S consistent and there is no set of smaller cardinality that accomplishes the same.
- $I_{nc}(S) = 21$ as the set $\{\neg b_8 \wedge b_8\}$ has size 1 and is inconsistent.

- $I_C(S) = 8$ as there must be at least 8 atoms, for example $a_2, a_3, a_4, a_5, a_7, a_9, a_{11}$, and b_8 , that must be given the value B for a 3-valued interpretation in order to satisfy all the formulas.
- $I_{mv}(S) = \frac{18}{21} = \frac{6}{7}$ as 18 of the 21 atoms are in problematic formulas.

5.2 A graph database as a general information space

We consider a general form of graph databases where vertices may be associated with properties and edges may be labeled [52, 54]. We assume the existence of three arbitrary, but fixed, disjoint sets V , $\mathcal{L}$, $\mathcal{P}$, of *vertex names*, *edge labels*, and *vertex properties*, respectively. Each property $p \in \mathcal{P}$ has an associated domain $dom(p)$ which is a set of values that can be assigned to p .

A *graph database* $\mathcal{G}$ is a triple $\langle V, E, \wp \rangle$, where (i) $E \subseteq V \times \mathcal{L} \times V$ is a set of labeled edges; and (ii) $\wp : V \times \mathcal{P} \rightarrow \bigcup_{p \in \mathcal{P}} dom(p)$ is a (possibly partial) *property assignment* function, such that if $\wp(v, p)$ is defined, then $\wp(v, p) \in dom(p)$. An example of a graph database consisting of 7 vertices is shown in Fig. 2.

The components of the general information space $S = \langle F, U, C \rangle$ for a graph database are as follows. The framework F consists of basic information about the vertices and the edges of the graph, that is, the sets of V , $\mathcal{L}$, and $\mathcal{P}$. Moreover, F describes which property $p \in \mathcal{P}$ is associated with a vertex $v \in V$ through function $\wp$, and its domain $dom(p)$. For instance, a property for the vertices of our graph database example is *type*, whose domain includes person and media. In Fig. 2, the vertices indicated by circles represent people and the vertices indicated by rectangles represent media objects. Moreover, property *age* is associated with people and *resolution* with media objects (their domains are obvious). The edges represent relationships between vertices whose meaning is given by the labels.

The data units consist of the information given by the vertices and the edges. Part of the information is given by the shape of a node. So some nodes are counted twice, once for its shape (one unit of information) and once for its properties, if any (another unit of information). Altogether there are 27 data units that we number as follows:

Fig. 2 Graph database instance

u_1 : (Photo 1, Media)	u_2 : (Photo 2, Media)
u_3 : (Photo 3, Media)	u_4 : (Daniel, Person)
u_5 : (Mark, Person)	u_6 : (Paul, Person)
u_7 : (James, Person)	u_8 : (Photo 1, 12MP)
u_9 : (Photo 2, 16MP)	u_{10} : (Photo 3, 8MP)
u_{11} : (Daniel, 35)	u_{12} : (James, 26)
u_{13} : (Daniel, posted, Photo 1)	u_{14} : (Daniel, knows, Mark)
u_{15} : (Daniel, knows, Paul)	u_{16} : (Mark, likes, Photo 1)
u_{17} : (Mark, posted, Photo 2)	u_{18} : (Mark, knows, Paul)
u_{19} : (Mark, likes, James)	u_{20} : (Paul, knows, Mark)
u_{21} : (Paul, likes, Photo 2)	u_{22} : (Paul, posted, Photo 3)
u_{23} : (Paul, knows, James)	u_{24} : (James, likes, Paul)
u_{25} : (James, posted, Photo 3)	u_{26} : (Photo 1, taken before, Photo 2)
u_{27} : (Photo 2, taken before, Photo 1)	

Finally we write the requirements (the constraints on the graph) in English.

- c_1 : Every circular vertex (i.e, person vertex) must have an associated age value.
- c_2 : Every rectangular vertex (i.e, media vertex) must have an associated resolution.
- c_3 : There may not be a pair of rectangular vertices whose edges form a cycle.
- c_4 : There cannot be 2 edges with the label “posted” going to the same rectangular vertex.
- c_5 : For every edge between circular vertices that has the label “likes” there must be another edge with the label “knows”.

The English version of the requirements, as given above are easy to understand. In order to formalize them in first-order logic, we will use the binary relations *VertexType*, *PersonAge*, *PhotoResolution*, and the ternary relation *Edge*. Below we write these requirements as formulas in first-order logic.

- $c_1 = \forall x (VertexType(x, Person) \rightarrow \exists y PersonAge(x, y)).$
- $c_2 = \forall x (VertexType(x, Media) \rightarrow \exists y PhotoResolution(x, y)).$
- $c_3 = \forall x, y, z_1, z_2 (VertexType(x, Media) \wedge VertexType(y, Media) \rightarrow \neg(Edge(x, z_1, y) \wedge Edge(y, z_2, x))).$
- $c_4 = \forall x, y, z (VertexType(x, Media) \wedge Edge(y, posted, x) \wedge Edge(z, posted, x) \rightarrow y = z).$
- $c_5 = \forall x, y (VertexType(x, Person) \wedge VertexType(y, Person) \wedge Edge(x, likes, y) \rightarrow Edge(x, knows, y)).$

5.2.1 Transformation to a propositional knowledge base

We now show how the transformation from a general information space to a propositional knowledge base is applied for this graph database.

1. $A_U = \{a_1, \dots, a_{27}\}$ corresponding to the 27 data units.
2. $f(u_i) = a_i$ for all $i, 1 \leq i \leq 27$.
3. $B_C = \{b_1, \dots, b_5\}$ corresponding to the 5 requirements.
4. $h(c_i) = b_i$ for all $i, 1 \leq i \leq 5$.

5. $\mathcal{F}_S$ is the set of propositional formulas using $A_U \cup B_C$.
6. Now we show the mapping g by going over the requirements one at a time.

- c_1 The arity of c_1 is 1. The two data units u_5 and u_6 each violate c_1 . Hence, $g(c_1) = \neg a_5 \wedge \neg a_6 \wedge b_1$.
- c_2 This requirement is satisfied. Hence, $g(c_2) = b_2$.
- c_3 The arity of c_3 is 2 and is violated by the pair of edges u_{26} and u_{27} . Hence, $g(c_3) = (\neg a_{26} \vee \neg a_{27}) \wedge b_3$.
- c_4 The arity of c_4 is 2. It is violated by the pair of edges u_{22} and u_{25} . Hence, $g(c_4) = (\neg a_{22} \vee \neg a_{25}) \wedge b_4$.
- c_5 The arity of c_5 is 1. The two edges u_{19} and u_{24} each violate c_5 . Hence, $g(c_5) = \neg a_{19} \wedge \neg a_{24} \wedge b_5$.

7. Therefore $K_S = \{a_1, \dots, a_{27}, \neg a_5 \wedge \neg a_6 \wedge b_1, b_2, (\neg a_{26} \vee \neg a_{27}) \wedge b_3, (\neg a_{22} \vee \neg a_{25}) \wedge b_4, \neg a_{19} \wedge \neg a_{24} \wedge b_5\}$.

5.2.2 The calculation of the inconsistency measures

Below are the results of calculating the inconsistency measures of the graph database example. We use the fact that $MI(K_S) = \{\{a_5, \neg a_5 \wedge \neg a_6 \wedge b_1\}, \{a_6, \neg a_5 \wedge \neg a_6 \wedge b_1\}, \{a_{26}, a_{27}, (\neg a_{26} \vee \neg a_{27}) \wedge b_3\}, \{a_{22}, a_{25}, (\neg a_{22} \vee \neg a_{25}) \wedge b_4\}, \{a_{19}, \neg a_{19} \wedge \neg a_{24} \wedge b_5\}, \{a_{24}, \neg a_{19} \wedge \neg a_{24} \wedge b_5\}\}$.

- $I_B(S) = 1$ as K_S is inconsistent.
- $I_M(S) = 6$ as there are 6 minimal inconsistent subsets for K_S .
- $I_\#(S) = 4 \times \frac{1}{2} + 2 \times \frac{1}{3} = \frac{8}{3}$ as there are 4 minimal inconsistent subsets of size 2 and 2 minimal inconsistent subsets of size 3 for K_S .
- $I_P(S) = 12$ as there are 12 problematic formulas in S .
- $I_H(S) = 4$ as the deletion of the 4 formulas: $g(c_1)$, $g(c_3)$, $g(c_4)$, and $g(c_5)$ makes K_S consistent and there is no smaller cardinality set that accomplishes the same.
- $I_{nc}(S) = 32 - 1 = 31$ as there is a minimal inconsistent subset of size 2.
- $I_C(S) = 6$ as a 3-valued interpretation must give at least a_5, a_6, a_{19}, a_{24} , one of a_{26} and a_{27} , and one of a_{22} and a_{25} the value B to satisfy all the formulas.
- $I_{mv}(S) = \frac{12}{32} = \frac{3}{8}$ as 12 of the 32 atoms appear in problematic formulas.

5.3 A spatio-temporal database as a general information space

In this section, we consider spatio-temporal databases where atomic statements of the form “*object id is/was/will be at point p in space at time t*” are represented. This is a deterministic version of the SPOT (Spatial PRobabilistic Temporal) database [53], a declarative framework for the representation and processing of probabilistic spatio-temporal data, where we assume that each region in space consists of a single point. However, we allow regions to be sets of points when specifying spatio-temporal integrity constraints [49].

We assume the existence of three types of constant symbols: object symbols, time value symbols, and spatial region symbols. The constants are in $ID = \{id_1, \dots, id_m\}$, $T = [0, 1, \dots, tmax]$ (where $tmax$ is an integer), and the set of $r \subseteq Space = \{p_1, \dots, p_n\}$. For simplicity, we use a square grid for $Space$ within which a region r is a rectangle. In the grid, each location can be written as (α, β) , where α and β are integers and $0 \leq \alpha, \beta \leq N$ for some integer N . Thus, $Space$ contains $(N + 1)^2$ points. For a singleton region consisting

of point location p , we simply use p to denote the region (instead of writing $r = \{p\}$). We also use variables for each type: object variables, time variables, and spatial variables.

A *spatio-temporal atom* (*st-atom*, for short) is an expression of the form $loc(x, y, z)$, where: (i) x is an object variable or a constant $id \in ID$, (ii) y is a space variable or a constant $r \subseteq Space$, and (iii) z is a time variable or a constant $t \in T$. We say that st-atom $loc(x, y, z)$ is *ground* if all of its arguments x, y, z are constants. For instance, $loc(id, p, t)$, where $id \in ID$, $\{p\} \subseteq Space$, and $t \in T$ is a ground st-atom. The intuitive meaning of $loc(id, p, t)$ is that object id is/was/will be at point location p at time t .

A spatio-temporal database ST is a finite set of ground st-atoms, each of them consisting of a singleton region. Integrity constraints have the form of *spatio-temporal denial* formulas [49], each of them being a universally quantified negation of conjunctions of st-atoms and built-in predicates. The meaning of a spatio-temporal database ST with a set of constraints is given an interpretation that satisfy it. An *interpretation* I is a function $I : ID \times T \rightarrow Space$. Thus, an interpretation specifies a trajectory for each $id \in ID$. That is, for each $id \in ID$, I says where in $Space$ object id was/is/will be at each time t in T .

The components of the general information space $S = \langle F, U, C \rangle$ for a spatio-temporal database ST over the set ID of ids, T of time points, and $Space$, with a set of spatio-temporal denial formulas, are as follows. The framework F consists of the first-order language for defining st-atoms where id, time, and spatial attributes are interpreted using the domains ID , T , and $Space$, respectively. The set U of information units is the set of atoms in ST , and the set C of requirements is the set of spatio-temporal denial formulas (integrity constraints).

We now provide an example. Consider an airport security system which collects data from sensors. A simplified plan of an airport area is shown in Figure 3(a), where regions a , b and c are highlighted. Once entered in this area, passengers typically move through the path delimited by queue dividers (represented by dotted lines in the figure, and overlapping with region a), and reach the room on the upper-half right side where security checks are performed (region b is included in this room). Next, passengers can spend some time in the hall room, and finally go towards the exit (near region c).

Suppose that the security system uses a spatio-temporal database to represent the information collected. For instance, st-atom $loc(id_1, (7, 4), 9)$ says that a passenger having id

Fig. 3 a A map of an airport area (names of regions are on their bottom-right corner); b atoms

id_1 was at point (7, 4) at time 9. Atom $loc(id_1, (2, 0), 5)$ says that id_1 was recognized at point (2, 0) at the earlier time 5. Here U , the spatio-temporal database, consists of the set of 12 ground st-atoms in Fig. 3b, which includes the two atoms above, namely data units $u_1, \dots, u_{12}$. Herein, the framework F is defined by using $ID = \{id_1, id_2\}$, $T = [0, 20]$, and $Space$ consisting of the set of points (α, β) such that $0 \leq \alpha \leq 7$ and $0 \leq \beta \leq 7$.

There are 4 requirements in C as given below both in English and as spatio-temporal denial formulas. The underlying language of general information spaces, and thus of requirements, is first-order logic. However, in specific cases, like spatio-temporal databases, we can have restrictions on the requirements (cf. A3.3 of Definition 3). Spatio-temporal denial formulas are an example of such restrictions.

The first requirement for our spatio-temporal database is that “an object cannot be in two non-overlapping regions at any time between 1 and 20”, and it can be expressed by the following spatio-temporal denial formula:

$c_1 = \forall x, y_1, y_2, z \neg[loc(x, y_1, z) \wedge loc(x, y_2, z) \wedge y_1 \text{ nov } y_2 \wedge z \geq 1 \wedge 20 \geq z]$, where *nov* stands for “does not overlap”.

Moreover, in region b security checks on one individual at a time are performed. The constraint “there cannot be two distinct objects in region b at any time between 1 and 20” can be expressed by the following spatio-temporal denial formula:

$c_2 = \forall x_1, x_2, z \neg[loc(x_1, b, z) \wedge loc(x_2, b, z) \wedge x_1 \neq x_2 \wedge z \geq 1 \wedge 20 \geq z]$.

Due to the distance and the several obstacles between the entrance and the exit, we also have the constraint “no object can reach region c starting from region a in less than 10 time units”, that can be expressed as:

$c_3 = \forall x, z_1, z_2 \neg[loc(x, a, z_1) \wedge loc(x, c, z_2) \wedge z_1 < z_2 \wedge z_2 < z_1 + 10]$.

Finally, as the security check on each individual takes at least 2 time units, we know that “object id can go away from region b only if it stayed there for at least 2 time units”, that can be expressed as:

$c_4 = \forall y_1, y_2, z_1, z_2, z_3 \neg[loc(id, y_1, z_1) \wedge loc(id, b, z_2) \wedge loc(id, y_2, z_3) \wedge y_1 \text{ nov } b \wedge y_2 \text{ nov } b \wedge z_2 = z_1 + 1 \wedge z_2 < z_3 \wedge z_2 + 2 \geq z_3]$.

5.3.1 Transformation to a propositional knowledge base

We now show how the transformation from a general information space to a propositional knowledge base is applied for this spatio-temporal database.

1. $A_U = \{a_1, \dots, a_{12}\}$ corresponding to the 12 st-atoms in Fig. 3b,
2. $f(u_i) = a_i$ for all i , $1 \leq i \leq 12$.
3. $B_C = \{b_1, \dots, b_4\}$ corresponding to the 4 spatio-temporal denial formulas.
4. $h(c_i) = b_i$ for all i , $1 \leq i \leq 4$.
5. $\mathcal{F}_S$ is the set of propositional logic formulas using $A_U \cup B_C$.
6. Now we show the mapping g by going over the constraints one at a time.

c_1 The arity of c_1 is 2. It is violated by the pair of st-atoms u_8 and u_9 . Hence, $g(c_1) = (\neg a_8 \vee \neg a_9) \wedge b_1$.

c_2 This constraint is satisfied. Hence, $g(c_2) = b_2$.

c_3 The arity of c_3 is 2. It is violated by the pairs of st-atoms u_1, u_6 and u_2, u_6 . Hence, $g(c_3) = (\neg a_1 \vee \neg a_6) \wedge (\neg a_2 \vee \neg a_6) \wedge b_3$.

c_4 The arity of c_4 is 3. It is violated by the st-atoms u_3, u_4 and u_5 . Hence, $g(c_4) = (\neg a_3 \vee \neg a_4 \vee \neg a_5) \wedge b_4$.

7. Therefore $K_S = \{a_1, \dots, a_{12}, (\neg a_8 \vee \neg a_9) \wedge b_1, b_2, (\neg a_1 \vee \neg a_6) \wedge (\neg a_2 \vee \neg a_6) \wedge b_3, (\neg a_3 \vee \neg a_4 \vee \neg a_5) \wedge b_4\}$.

5.3.2 The calculation of the inconsistency measures

Below are the results of calculating the inconsistency measures of the spatio-temporal database example. We use the fact that

$MI(K_S) = \{\{a_8, a_9, (\neg a_8 \vee \neg a_9) \wedge b_1\}, \{a_1, a_6, (\neg a_1 \vee \neg a_6) \wedge (\neg a_2 \vee \neg a_6) \wedge b_3\}, \{a_2, a_6, (\neg a_1 \vee \neg a_6) \wedge (\neg a_2 \vee \neg a_6) \wedge b_3\}, \{a_3, a_4, a_5, (\neg a_3 \vee \neg a_4 \vee \neg a_5) \wedge b_4\}\}$.

- $I_B(S) = 1$ as K_S is inconsistent.
- $I_M(S) = 4$ as there are 4 minimal inconsistent subsets for K_S .
- $I_\#(S) = 3 \times \frac{1}{3} + \frac{1}{4} = \frac{5}{4}$ as there are three minimal inconsistent subsets of size 3, and 1 of size 4 in K_S .
- $I_P(S) = 8 + 3 = 11$ as there are 11 problematic formulas in K_S .
- $I_H(S) = 3$ as the deletion of the 3 formulas of $g(c_1)$, $g(c_2)$, and $g(c_3)$ makes K_S consistent and there is no set of smaller cardinality that accomplishes the same.
- $I_{nc}(S) = 16 - 2 = 14$ as 2 is the largest number such that all sets of 2 are consistent.
- $I_C(S) = 3$ as there must be at least 3 atoms, for example a_8 , a_6 , and a_3 that must be given the value B for a 3-valued interpretation in order to satisfy all the formulas.
- $I_{mv}(S) = \frac{8}{16} = \frac{1}{2}$ as 8 of the 16 atoms are in problematic formulas.

5.4 A blocks world configuration as a general information space

Blocks-world planning has been widely investigated in AI, as it captures several aspects of planning systems [32]. In this case, the components of $S = \langle F, U, C \rangle$ are as follows. The framework indicates that there is a finite number of colored blocks of the same size in stacks on a table, which is large enough to hold all (i.e., the number of stacks can be equal to number of blocks). A blocks world configuration is shown in Fig. 4.

The data units specify the color of each block and the location of each block. Each ground atom is a 4-tuple (*blockid*, *color*, *stack number*, *position*) for which we use the shorthand st_{ij} to indicate that this block is in stack i at location j (from the bottom). The data units in this example are as follows:

Fig. 4 Blocks world configuration

$st_{11} : green$	$st_{12} : blue$	$st_{13} : blue$	
$st_{21} : red$	$st_{22} : yellow$	$st_{23} : blue$	$st_{24} : red$
$st_{31} : yellow$	$st_{32} : red$	$st_{33} : blue$	$st_{41} : red$

So there are 4 stacks with 3, 4, 3, and 1 blocks respectively.

Finally we get to the requirements that are the constraints in this Blocks world; we write them in English first.

- c_1 : No blue block can be on top of another blue block.
- c_2 : There cannot be a yellow block that is on top of a red block and that has a red block above it.
- c_3 : There cannot be a red block at the bottom of a stack.
- c_4 : No stack has both a green block and a blue block.
- c_5 : At least one of the blocks is purple.
- c_6 : There must be a blue block in at least 3 stacks.

In order to write the constraints in first-order logic we use a 4-ary predicate symbol *Block* and obtain the following formulas.

- $c_1 = \forall x, y, i, j \neg (Block(x, blue, i, j) \wedge Block(y, blue, i, j + 1)).$
- $c_2 = \forall x, y, z, i, j, k \neg (Block(x, yellow, i, j) \wedge Block(y, red, i, j - 1) \wedge Block(z, red, i, k) \wedge j < k).$
- $c_3 = \forall x, i \neg Block(x, red, i, 1).$
- $c_4 = \forall x, y, i, j, k \neg (Block(x, green, i, j) \wedge Block(y, blue, i, k)).$
- $c_5 = \exists x, i, j Block(x, purple, i, j).$
- $c_6 = \exists x, y, z, i_1, i_2, i_3, j_1, j_2, j_3 (Block(x, blue, i_1, j_1) \wedge Block(y, blue, i_2, j_2) \wedge Block(z, blue, i_3, j_3) \wedge i_1 \neq i_2 \wedge i_1 \neq i_3 \wedge i_2 \neq i_3).$

5.4.1 Transformation to a propositional knowledge base

We now show how the transformation from a general information space to a propositional knowledge base is applied for this blocks world example.

1. $A_U = \{a_1, \dots, a_{11}\}$ corresponding to the 11 blocks.
2. $f(st_{11}) = a_1, f(st_{12}) = a_2, f(st_{13}) = a_3, f(st_{21}) = a_4, f(st_{22}) = a_5, f(st_{23}) = a_6, f(st_{24}) = a_7, f(st_{31}) = a_8, f(st_{32}) = a_9, f(st_{33}) = a_{10}, f(st_{41}) = a_{11},$
3. $B_C = \{b_1, \dots, b_6\}$ corresponding to the 6 constraints.
4. $h(c_i) = b_i$ for all $i, 1 \leq i \leq 6.$
5. $\mathcal{F}_S$ is the set of propositional formulas using $A_U \cup B_C.$
6. Now we show the mapping g by going over the constraints one at a time.

- c_1 The arity of c_1 is 2. The two blocks st_{12} and st_{13} together violate c_1 . Hence, $g(c_1) = (\neg a_2 \vee \neg a_3) \wedge b_1.$
- c_2 The arity of c_2 is 3. The 3 blocks that together violate this constraint are $st_{21}, st_{22},$ and st_{24} . Hence, $g(c_2) = (\neg a_4 \vee \neg a_5 \vee \neg a_7) \wedge b_2.$
- c_3 The arity of c_3 is 1. The blocks st_{21} and st_{41} both violate this constraint. Hence, $g(c_3) = \neg a_4 \wedge \neg a_{11} \wedge b_3.$
- c_4 The arity of c_4 is 2. The blocks st_{11} and st_{12} as well as the blocks st_{11} and st_{13} violate this constraint. Hence, $g(c_4) = (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge b_4.$
- c_5 The arity of c_5 is 0. There is no purple block in any stack. Hence, $g(c_5) = \neg b_5 \wedge b_5.$
- c_6 This constraint is satisfied. Hence, $g(c_6) = b_6.$

7. Therefore $K_S = \{a_1, \dots, a_{11}, (\neg a_2 \vee \neg a_3) \wedge b_1, (\neg a_4 \vee \neg a_5 \vee \neg a_7) \wedge b_2, \neg a_4 \wedge \neg a_{11} \wedge b_3, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge b_4, \neg b_5 \wedge b_5, b_6\}$.

5.4.2 The calculation of the inconsistency measures

Below are the results of calculating the inconsistency measures of the Blocks world example. We use the fact that $MI(K_S) = \{\{a_2, a_3, (\neg a_2 \vee \neg a_3) \wedge b_1\}, \{a_4, a_5, a_7, (\neg a_4 \vee \neg a_5 \vee \neg a_7) \wedge b_2\}, \{a_4, \neg a_4 \wedge \neg a_{11} \wedge b_3\}, \{a_{11}, \neg a_4 \wedge \neg a_{11} \wedge b_3\}, \{a_1, a_2, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge b_4\}, \{a_1, a_3, (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge b_4\}, \{\neg b_5 \wedge b_5\}\}$.

- $I_B(S) = 1$ as K_S is inconsistent.
- $I_M(S) = 7$ as there are 7 minimal inconsistent subsets for K_S .
- $I_{\#}(S) = 1 + 2 \times \frac{1}{2} + 3 \times \frac{1}{3} + 1 \times \frac{1}{4} = \frac{13}{4}$ as there is 1 minimal inconsistent subset of size 1, 2 minimal inconsistent subsets of size 2, 3 minimal inconsistent subsets of size 3, and 1 minimal inconsistent subset of size 4 for K_S .
- $I_P(S) = 7 + 5 = 12$ as 7 atoms (i.e., colored block locations) plus 5 propositional formulas (i.e., the transformations of the requirements) are problematic in K_S .
- $I_H(S) = 5$ as the deletion of the 5 formulas: $g(c_1)$, $g(c_2)$, $g(c_3)$, $g(c_4)$, and $g(c_5)$ makes K_S consistent and there is no smaller cardinality set that accomplishes the same.
- $I_{nc}(S) = 17$ as there is a minimal inconsistent subset of size 1.
- $I_C(S) = 5$ as a 3-valued interpretation that satisfies all the formulas must give a_4 , a_{11} , b_5 , and at least 2 other atoms, for example, a_1 and a_2 the value B .
- $I_{mv}(S) = \frac{12}{17}$ as 12 of the 17 atoms are in problematic formulas.

6 Rationality postulates for inconsistency measures

In Section 2 we defined several inconsistency measures for propositional logic. There are many others. In order to distinguish among them some researchers proposed various properties that, intuitively, a “good” inconsistency measure should possess. These properties are called rationality postulates; we will just use the term “postulate”. In this section we present ten well-known postulates for propositional inconsistency measures.

Our goal is to discover what happens when we consider these postulates in the context of general information spaces. It turns out, that in order to be applicable, the postulates need to be modified.

6.1 Postulates for propositional inconsistency measures

We give the definitions of ten well-known postulates and explain them.

Definition 6 (Postulates for Propositional Inconsistency Measures) Let K, K' be KBs, ϕ, ψ formulas, and I and inconsistency measure. The postulates for inconsistency measures are as follows:

Monotony If $K \subseteq K'$, then $I(K) \leq I(K')$.

Free-Formula Independence If $\phi \in \text{Free}(K)$, then $I(K) = I(K \setminus \{\phi\})$.

Penalty If $\phi \in \text{Problematic}(K)$, then $I(K) > I(K \setminus \{\phi\})$.

Dominance If ϕ is consistent and ϕ logically implies ψ , then $I(K \cup \{\phi\}) \geq I(K \cup \{\psi\})$.

Super-Additivity If $K \cap K' = \emptyset$, then $I(K \cup K') \geq I(K) + I(K')$.

MI-Separability If $MI(K \cup K') = MI(K) \cup MI(K')$ and $MI(K) \cap MI(K') = \emptyset$, then $I(K \cup K') = I(K) + I(K')$.

MI-Normalization If $M \in MI(K)$, then $I(M) = 1$.

Attenuation If $M, M' \in MI(K)$ and $|M| < |M'|$, then $I(M) > I(M')$.

Equal Conflict If $M, M' \in MI(K)$ and $|M| = |M'|$, then $I(M) = I(M')$.

Almost Consistency If $M_1, M_2, \dots$ is a sequence of minimal inconsistent sets with $\lim_{i \rightarrow \infty} |M_i| = \infty$, then $\lim_{i \rightarrow \infty} I(M_i) = 0$.

Monotony means that the enlargement of a KB cannot decrease its measure. The independence postulates mean that free formulas do not change the inconsistency measure. Penalty states that deleting a problematic formula decreases the measure. Dominance deals with the case where a KB and two formulas ϕ and ψ are given and ϕ is consistent and logically implies ψ . Then the addition of ψ to the KB cannot have a larger measure than the addition of ϕ . Super-Additivity and MI-Separability give information about the union of two knowledge bases under certain conditions. Super-Additivity deals with the case where the knowledge bases are disjoint in which case the measure of the union is at least as great as the sum of the measures. MI-Separability requires that the minimal inconsistent sets of the two knowledge bases partition the minimal inconsistent sets of the union, in which case the measure of the union is the sum of the measures. MI-Normalization, Attenuation, Equal Conflict, and Almost Consistency deal specifically with minimal inconsistent sets. MI-Normalization requires every minimal inconsistent set to have measure 1. Attenuation requires larger size minimal inconsistent sets to have smaller measures; Equal Conflict requires minimal inconsistent set of the same size to have the same measure. Finally, Almost Consistency requires that as minimal inconsistent sets get larger the measures get closer and closer to 0. Clearly, the satisfaction of MI-Normalization implies the satisfaction of Equal Conflict and the violation of Attenuation and Almost Consistency. Table 1 shows the satisfaction and violation of these postulates for propositional logic from [57].

Table 1 Postulates satisfaction of propositional inconsistency measures

	Propositional inconsistency measures							
	I_B	I_M	$I_{\#}$	I_P	I_H	I_{nc}	I_C	I_{mv}
Monotony	✓	✓	✓	✓	✓	✓	✓	✗
Free-Formula Independence	✓	✓	✓	✓	✓	✗	✗	✗
Penalty	✗	✓	✓	✓	✗	✓	✗	✗
Dominance	✓	✗	✗	✗	✗	✗	✓	✗
Super-Additivity	✗	✓	✓	✓	✓	✓	✗	✗
MI-Separability	✗	✓	✓	✗	✗	✗	✗	✗
MI-Normalization	✓	✓	✗	✗	✓	✓	✗	✓
Attenuation	✗	✗	✓	✗	✗	✗	✗	✗
Equal Conflict	✓	✓	✓	✓	✓	✓	✗	✓
Almost Consistency	✗	✗	✓	✗	✗	✗	✗	✗

✓ means satisfied, ✗ means violated

6.2 Postulates for general information spaces

Recall how we measure inconsistency in a general information space. We transform it to a propositional knowledge base and then apply a propositional inconsistency measure to it. We cannot just assume that an inconsistency measure applied to a general information space has the same properties as the measure when applied to a propositional knowledge base. In fact, we will consider this issue in detail in the next subsection. In this subsection we focus on the definition of the postulates in our context.

The propositional inconsistency measures use concepts such as minimal inconsistent subsets and free and problematic formulas. Some of the postulates involve the set operations union, intersection, and difference. In order to deal with the postulates we need to have definitions for them for general information spaces. The most critical issue is that the logic of general information spaces is not monotonic. For example, consider the relational database given in Section 5.1. Recall that there is a violation of c_8 because there is only one *ID* referenced in the *Family* relation. This leads to the formula $\neg b_8 \wedge b_8$ in K_S , where S is the general information space under consideration. But if a tuple is inserted into *Family* with a different *ID* to form S' , then $S \subseteq S'$ but $K_S \not\subseteq K_{S'}$ because the formula $\neg b_8 \wedge b_8$ is not in $K_{S'}$ (it is replaced by b_8) and so $Inc(S) \not\subseteq Inc(S')$. But propositional logic is monotonic: adding a formula to a propositional knowledge base cannot make a minimal inconsistent set disappear.

Dealing with inconsistency measures in nonmonotonic logic has been thoroughly investigated [59]. In that work the inconsistency measures and rationality postulates are defined directly in the nonmonotonic logic. Specifically, [59] adjusts the Monotonicity postulate in order to obtain a meaningful one for nonmonotonic frameworks, where monotonicity cannot be required to hold for each additional piece of information. This is achieved by restricting the postulate to the case where additional information has no influence on the conflicts contained in the original knowledge base. From a technical standpoint, [59] relies on the concept of *strong inconsistency* to define inconsistency measures for nonmonotonic logics. A set of formulas is said to be strongly inconsistent if no superset is consistent. Thus, [59] defines *strong monotonicity* which, despite the name that emphasizes the role of strongly inconsistent subsets, actually *weakens* monotonicity due to its additional precondition.

In our approach, we start with a nonmonotonic logic and transform it to a monotonic logic in order to compute the inconsistency measures. This way we can use directly all the inconsistency measures previously defined for propositional logic. But we still need to give appropriate definitions for the concepts used for the postulates. One thing we can do is to restrict the requirements so that the general information space is monotonic. When that is not the case, we define what we call the monotonic elements and restrict the postulates to apply only to them. In the case mentioned above, where the requirements are restricted for monotonicity, all the elements are monotonic. We use the terminology “monotonic” for information units whose removal cannot cause a new inconsistency. This essentially corresponds to the way that [59] uses strong inconsistency and the preservation of conflicts for nonmonotonic logics in general.

Two important postulates consider what happens when a formula is removed from the knowledge base. In our case, either an information unit or a requirement is removed. The problem is that the nonmonotonicity of general information spaces means that the removal of an information unit may cause an inconsistency while the addition of an information

unit may resolve an inconsistency. To handle this, we exclude such information units from consideration.

In the following, we assume that a general information space $S = \langle F, U, C \rangle$ is given. Recall that we also use the symbol S for the set $S = U \cup C$.

Definition 7 We call $u \in S$ *monotonic* iff $\text{Inc}(S \setminus \{u\}) \subseteq \text{Inc}(S)$.

Note that requirements are automatically monotonic.

Next we define the free and problematic formulas in our context.

Definition 8 Let $u \in U$. Then u is *m-free* (in S) iff u is monotonic and $f(u) \in \text{Free}(K_S)$. Let $c \in C$. Then c is *m-free* iff $g(c) \in \text{Free}(K_S)$.

$\text{m-Free}(S)$ is the set of m-free information units and requirements in S .

Definition 9 Let $u \in U$. Then u is *m-problematic* (in S) iff u is monotonic and $f(u) \in \text{Problematic}(K_S)$. Let $c \in C$. Then c is *m-problematic* iff $g(c) \in \text{Problematic}(K_S)$.

$\text{m-Problematic}(S)$ is the set of m-problematic information units and requirements in S .

In defining the postulates for general information space inconsistency measures, we will be using the m-version of various concepts and substitute Inc for MI . Dominance does not change but all other postulates are changed to their m-versions as given below. Note that Dominance applies only to requirements as an information unit cannot logically imply another information unit.

Now we define the postulates we will consider for inconsistency measures in general information spaces.

Definition 10 (Postulates for General Information Space Inconsistency Measures) Let S, S' be general information spaces and I an inconsistency measure. The postulates for inconsistency measures are as follows:

m-Monotony If $S \subseteq S'$ and $\text{Inc}(S) \subseteq \text{Inc}(S')$ then $I(S) \leq I(S')$.

m-Free-Formula Independence There are two parts.

If $u \in \text{m-Free}(S)$, then $I(S) = I(S \setminus \{u\})$.

If $c \in \text{m-Free}(S)$, then $I(S) = I(S \setminus \{c\})$.

m-Penalty There are two parts.

If $u \in \text{m-Problematic}(S)$, then $I(S) > I(S \setminus \{u\})$.

If $c \in \text{m-Problematic}(S)$, then $I(S) > I(S \setminus \{c\})$.

Dominance If c logically implies c' , then $I(S \cup \{c\}) \geq I(S \cup \{c'\})$.

m-Super-Additivity If $S \cap S' = \emptyset$ and $\text{Inc}(S) \cup \text{Inc}(S') \subseteq \text{Inc}(S \cup S')$, then $I(S \cup S') \geq I(S) + I(S')$.

m-Separability If $\text{Inc}(S \cup S') = \text{Inc}(S) \cup \text{Inc}(S')$ and $\text{Inc}(S) \cap \text{Inc}(S') = \emptyset$, then $I(S \cup S') = I(S) + I(S')$.

m-Normalization If $S' \in \text{Inc}(S)$, then $I(S') = 1$.

m-Attenuation If $S_1, S_2 \in \text{Inc}(S)$ and $|S_1| < |S_2|$, then $I(S_1) > I(S_2)$.

m-Equal Conflict If $S_1, S_2 \in \text{Inc}(S)$ and $|S_1| = |S_2|$, then $I(S_1) = I(S_2)$.

m-Almost Consistency If $S_1, S_2, \dots$ is a sequence of inconsistencies of S with

$$\lim_{i \rightarrow \infty} |S_i| = \infty, \text{ then } \lim_{i \rightarrow \infty} I(S_i) = 0.$$

6.3 Postulate satisfaction for general information spaces

In this subsection we show which of the ten postulates for general information spaces is satisfied by the eight inconsistency measures given in Section 2. We cannot just use the results from the propositional case for two reasons: first, our definitions for the postulates are different and second, the transformation of a general information space allows only formulas in propositional logic of a certain form. For example, iceberg inconsistencies [13] cannot occur.

Theorem 2 *The satisfaction of postulates for general information space inconsistency measures is as given in Table 2. The differences with the propositional case are as follows: for I_C m-Free-Formula Independence, m-Super-Additivity, m-MI-Normalization, and m-Equal Conflict are satisfied and Dominance is violated; for I_{mv} m-Penalty is satisfied.*

Proof We go through the inconsistency measures one by one. When a postulate is satisfied, we explain why. When a postulate is violated, we give a counterexample. For all the counterexamples, the framework consists of two relations: R_1 is unary and R_2 is binary. The domain for all attributes is the set of integers. When we write “by the antecedent”, we mean the condition of the postulate under consideration.

I_B m-Monotony If S is inconsistent then $I_B(S) = 1$ and since $Inc(S) \subseteq Inc(S')$, $I_B(S') = 1$. If S is consistent then $I_B(S) = 0 \leq I_B(S')$.

m-Free-Formula Independence If $I_B(S) = 0$ and $Inc(S \setminus \{u\}) \subseteq Inc(S)$ then $I_B(S \setminus \{u\}) = 0$. If $I_B(S) = 1$ and $u \in m\text{-Free}(S)$ then $f(u) \in Free(K_S)$. So u is not involved in any inconsistency in K_S which is inconsistent. Hence, $S \setminus \{u\}$ is also inconsistent. So, $I_B(S \setminus \{u\}) = 1$. The proof is similar for requirements.

m-Penalty Let $S = \{R_1(0), R_1(1), \forall x(R_1(x) \rightarrow x > 1)\}$. Then $I_B(S) = 1$. Also, $R_1(0)$ is m-problematic and we get $I_B(S \setminus \{R_1(0)\}) = 1$. So $I_B(S) \not\geq I_B(S \setminus \{R_1(0)\})$.

Dominance It suffices to consider the case where $I_B(S \cup \{c'\}) = 1$. There are two subcases. If $c' \in m\text{-Free}(S \cup \{s'\})$ then c' does not cause an inconsistency, hence there

Table 2 Postulates satisfaction of inconsistency measures for general information spaces

	Inconsistency Measures							
	I_B	I_M	$I_{\#}$	I_P	I_H	I_{nc}	I_C	I_{mv}
m-Monotony	✓	✓	✓	✓	✓	✓	✓	✗
m-Free-Formula Independence	✓	✓	✓	✓	✓	✗	✓*	✗
m-Penalty	✗	✓	✓	✓	✗	✓	✗	✓*
Dominance	✓	✗	✗	✗	✗	✗	✗*	✗
m-Super-Additivity	✗	✓	✓	✓	✓	✓	✓*	✗
m-MI-Separability	✗	✓	✓	✗	✗	✗	✗	✗
m-MI-Normalization	✓	✓	✗	✗	✓	✓	✓*	✓
m-Attenuation	✗	✗	✓	✗	✗	✗	✗	✗
m-Equal Conflict	✓	✓	✓	✓	✓	✓	✓*	✓
m-Almost Consistency	✗	✗	✓	✗	✗	✗	✗	✗

✓ means satisfied, ✗ means violated, and * means that it is different from the propositional case

- must be some $c'' \in \text{m-Problematic}(S \cup \{c'\})$ and $c'' \in S \cup \{c\}$, hence $I_B(S \cup \{c\}) = 1 \geq I_B(S \cup \{c'\})$. For the second subcase, $c' \in \text{m-Problematic}(S \cup \{c'\})$. Then $c \in \text{m-Problematic}(S \cup \{c\})$ and again $I_B(S \cup \{c\}) = 1 \geq I_B(S \cup \{c'\})$.
- m-Super-Additivity** Let $S = \{R_1(0), \forall x(R_1(x) \rightarrow x \neq 0)\}$ and $S' = \{R_1(1), \forall x(R_1(x) \rightarrow x \neq 1)\}$. Then $\text{Inc}(S) = \{S\}$ and $\text{Inc}(S') = \{S'\}$ and the two conditions are satisfied. But $I_B(S \cup S') = 1 \not\geq I_B(S) + I_B(S') = 2$.
- m-Separability** The counterexample for m-Super-Additivity is also a counterexample for m-Separability.
- m-Normalization** If $S' \in \text{Inc}(S)$ then S' is inconsistent; hence $I_B(S') = 1$.
- m-Attenuation** The satisfaction of m-Normalization means that m-Attenuation is violated.
- m-Equal Conflict** This follows from the satisfaction of m-Normalization.
- m-Almost Consistency** The satisfaction of m-Normalization means that m-Almost Consistency is violated.
- I_M **m-Monotony** This follows from the bijection in Theorem 1 as $\text{MI}(K_S) \subseteq \text{MI}(K_{S'})$.
- m-Free-Formula Independence** If $u \in \text{m-Free}(S)$ then $f(u) \in \text{Free}(K_S)$ and as u is monotonic $\text{MI}(K_S) = \text{MI}(K_{S \setminus \{u\}})$. Hence, $I_M(S) = I_M(S \setminus \{u\})$. The proof is similar for requirements.
- m-Penalty** If $u \in \text{m-Problematic}(S)$ then u must be in some inconsistency of S . As u is monotonic, $S \setminus \{u\}$ must have fewer inconsistencies than S . So, by Theorem 1, $I_M(S \setminus \{u\}) < I_M(S)$. For $c \in \text{m-Problematic}(S)$ the result follows by the same reasoning.
- Dominance** Let $S = \{R_1(0), R_1(1), \forall x(R_1(x) \rightarrow x = 2)\}$, $c = \forall x(R_1(x) \rightarrow x = 2)$ and $c' = \forall x(R_1(x) \rightarrow (x = 1 \vee x = 2))$. Note that $S \cup \{c\} = S$ and c logically implies c' . Then, $I_M(S \cup \{c\}) = 2 \not\geq I_M(S \cup \{c'\}) = 3$.
- m-Super-Additivity** This follows from the antecedent and Theorem 1.
- m-Separability** This follows from the antecedent and Theorem 1.
- m-Normalization** This follows from Theorem 1.
- m-Attenuation** The satisfaction of m-Normalization means that m-Attenuation is violated.
- m-Equal Conflict** This follows from the satisfaction of m-Normalization.
- m-Almost Consistency** The satisfaction of m-Normalization means that m-Almost Consistency is violated.
- $I_\#$ **m-Monotony** The proof is the same as for I_M .
- m-Free-Formula Independence** The proof is the same as for I_M .
- m-Penalty** The proof is the same as for I_M .
- Dominance** The same counterexample can be used as for I_M : $I_\#(S \cup \{c\}) = \frac{1}{2} + \frac{1}{2} = 1 \not\geq I_\#(S \cup \{c'\}) = \frac{1}{2} + \frac{1}{2} + \frac{1}{2} = \frac{3}{2}$.
- m-Super-Additivity** The proof is the same as for I_M .
- m-Separability** The proof is the same as for I_M .
- Normalization** Let $S = \{R_1(0), \forall x(R_1(x) \rightarrow x = 1)\}$. Then $S \in \text{Inc}(S)$ and $I_\#(S) = \frac{1}{2} \neq 1$.
- m-Attenuation** This follows from the definition of $I_\#$ and Theorem 1.
- m-Equal Conflict** This follows from the definition of $I_\#$ and Theorem 1.
- m-Almost Consistency** This follows from the definition of $I_\#$ and Theorem 1.
- I_P **m-Monotony** This follows from $\text{Inc}(S) \subseteq \text{Inc}(S')$ and the fact that $\text{Problematic}(K_S) \subseteq \text{Problematic}(K_{S'})$.
- m-Free-Formula Independence** As in the proof for I_M , $\text{MI}(K_S) = \text{MI}(K_{S \setminus \{u\}})$, from which the result follows.

- m-Penalty** As in the proof for I_M , $MI(K_{S \setminus \{u\}}) \subset MI(K)$, from which the result follows.
- Dominance** The same counterexample works as for I_M : $I_P(S \cup \{c\}) = 3 \not\geq I_P(S \cup \{c'\}) = 4$.
- m-Super-Additivity** The antecedent and Theorem 1 imply that $\text{Problematic}(K_S) \cup \text{Problematic}(K'_S) \subseteq \text{Problematic}(K_{S \cup S'})$. From this the result follows.
- m-Separability** Let $S = \{R_2(1, 1), R_2(1, 2), \forall x, y, z(R_2(x, y) \wedge R_2(x, z) \rightarrow y = z)\}$ and $S' = \{R_2(1, 1), R_2(0, 1), \forall x, y, z(R_2(x, y) \wedge R_2(z, y) \rightarrow x = z)\}$. Then, $Inc(S) = \{S\}$, $Inc(S') = \{S'\}$, and $Inc(S \cup S') = \{S, S'\}$. So, $I_P(S \cup S') = 5 \neq I_P(S) + I_P(S') = 3 + 3 = 6$.
- m-Normalization** For the m-Separability example, $S \in Inc(S)$ and $I_P(S) = 3 \neq 1$.
- m-Attenuation** This is always violated because if $S_1 \in Inc(S)$ and $S_2 \in Inc(S)$ and $|S_1| < |S_2|$, then $I_P(S_1) = |S_1| \neq I_P(S_2) = |S_2|$.
- m-Equal Conflict** $I_P(S_1) = |S_1| = I_P(S_2) = |S_2|$.
- m-Almost Consistency** The reasoning is similar to the proof of the violation of m-Attenuation.
- I_H **m-Monotony** This follows from $Inc(S) \subseteq Inc(S')$ and $MI(K_S) \subseteq MI(K_{S'})$.
- m-Free-Formula Independence** As shown in the proof of m-Free-Formula Independence for I_M , $MI(K_S) = MI(K_{S \setminus \{u\}})$, from which the result follows.
- m-Penalty** Let $S = \{R_2(1, 1), R_2(1, 2), R_2(0, 1), \forall x, y, z(R_2(x, y) \wedge R_2(x, z) \rightarrow y = z), \forall x, y, z(R_2(x, y) \wedge R_2(z, y) \rightarrow x = z)\}$. Then, $R_2(1, 2) \in \text{m-Problematic}(S)$, but $I_H(S) = 1 \neq I_H(S \setminus \{R_2(1, 2)\}) = 1$.
- Dominance** The same counterexample as the one used for I_M can be used here: $I_H(S \cup \{c\}) = 1 \not\geq I_H(S \cup \{c'\}) = 2$.
- m-Super-Additivity** This follows directly from the antecedent and Theorem 1.
- m-Separability** The same counterexample as the one used for I_P can be used here: $I_H(S \cup S') = 1 \neq I_H(S) + I_H(S') = 1 + 1 = 2$.
- m-Normalization** This follows also from Theorem 1.
- m-Attenuation** The satisfaction of m-Normalization means that m-Attenuation is violated.
- m-Equal Conflict** This follows from the satisfaction of m-Normalization.
- m-Almost Consistency** The satisfaction of m-Normalization means that m-Almost Consistency is violated.
- I_{nc} **m-Monotony** If $|S| < |S'|$ then $|K_S| < |K_{S'}|$ and so in the definition of I_{nc} the $\max\{n\}$ for $K_S \geq \max\{n\}$ for $K_{S'}$. Therefore, $I_{nc}(S) < I_{nc}(S')$.
- m-Free-Formula Independence** Let $S = \{R_1(0), R_1(1), \forall x(R_1(x) \rightarrow x = 0)\}$. Then $R_1(0) \in \text{m-Free}(S)$. But $I_{nc}(S) = 3 - 1 = 2 \neq I_{nc}(S \setminus \{R_1(0)\}) = 2 - 1 = 1$.
- m-Penalty** When an m-problematic information unit or requirement is deleted, the size of S , and hence the size of K_S decreases by 1. It is also possible that in the definition of I_{nc} $\max\{n\}$ increases; in any case, it cannot decrease. Hence, the deletion of an m-problematic element decreases the value of I_{nc} .
- Dominance** The same counterexample as the one used for I_M can be used here: $I_{nc}(S \cup \{c\}) = 3 - 1 = 2 \not\geq I_{nc}(S \cup \{c'\}) = 4 - 1 = 3$.
- m-Super-Additivity** By the antecedent, $|S \cup S'| = |S| + |S'|$ and so $|K_{S \cup S'}| = |K_S| + |K_{S'}|$ and the $\max\{n\}$ for $S \cup S'$ is the smaller of the $\max\{n\}$ for S and S' . This gives the result: $I_{nc}(S \cup S') \geq I_{nc}(S) + I_{nc}(S')$.

- m-Separability** Let $S = \{R_1(0), \forall x(R_1(x) \rightarrow x \neq 0)\}$ and $S' = \{R_1(1), \forall x(R_1(x) \rightarrow x \neq 1)\}$. Then, $Inc(S \cup S') = \{S, S'\} = Inc(S) \cup Inc(S')$ and $Inc(S) \cap Inc(S') = \emptyset$. But $I_{nc}(S \cup S') = 4 - 1 = 3 \neq I_{nc}(S) + I_{nc}(S') = 2 - 1 + 2 - 1 = 2$.
- m-Normalization** This follows from Theorem 1 and the fact that for a minimal inconsistent subset M of K_S , $I_{nc}(M) = |M| - (|M| - 1) = 1$.
- m-Attenuation** The satisfaction of m-Normalization means that m-Attenuation is violated.
- m-Equal Conflict** This follows from the satisfaction of m-Normalization.
- m-Almost Consistency** The satisfaction of m-Normalization means that m-Almost Consistency is violated.
- I_C m-Monotony** By the antecedent, no inconsistency is removed as elements are added to S ; hence, $I_C(S) \leq I_C(S')$.
- m-Free-Formula Independence** By the antecedent, the element deleted from K_S after the transformation must be an atom not in any minimal inconsistent subset such that the deletion of the corresponding element from S does not create a new inconsistency there. Hence, the value of I_C does not change after the deletion.
- m-Penalty** Let $S = \{R_2(1, 1), R_2(1, 2), \forall x, y, z(R_2(x, y) \wedge R_2(x, z) \rightarrow y = z), \forall x, y (R_2(x, y) \rightarrow y \neq 1)\}$. Every element of S is m-problematic. $K_S = \{a_1, a_2, (\neg a_1 \vee \neg a_2) \wedge b_1, \neg a_1 \wedge b_2\}$. So, $I_C(S) = 1$ as it suffices to give a_1 the value B . If the second requirement is deleted, we get $S \setminus \{\forall x, y (R_2(x, y) \rightarrow y \neq 1)\} = \{R_2(1, 1), R_2(1, 2), \forall x, y, z(R_2(x, y) \wedge R_2(x, z) \rightarrow y = z)\}$ and $K_{S \setminus \{\forall x, y (R_2(x, y) \rightarrow y \neq 1)\}} = \{a_1, a_2, (\neg a_1 \vee \neg a_2) \wedge b_1\}$ for which I_C gives the same value. Hence, $I_C(S) = 1 \neq I_C(S \setminus \{\forall x, y (R_2(x, y) \rightarrow y \neq 1)\}) = 1$.
- Dominance** Let $S = \{\exists x \exists y (x \neq y \wedge R_1(x) \wedge R_1(y)), c = \exists x \exists y (x \neq y \wedge R_1(x) \wedge R_1(y))\}$, and $c' = \exists x R_1(x)$. Clearly, c logically implies c' . But $S \cup \{c\} = S$ and $S \cup \{c'\} = \{\exists x \exists y (x \neq y \wedge R_1(x) \wedge R_1(y)), \exists x R_1(x)\}$. So, $K_{S \cup \{c\}} = \{\neg b_1 \wedge b_1\}$, while $K_{S \cup \{c'\}} = \{\neg b_1 \wedge b_1, \neg b_2 \wedge b_2\}$. So, $I_C(S \cup \{c\}) = 1 \not\geq I_C(S \cup \{c'\}) = 2$.
- m-Super-Additivity** By the antecedent, the information units in S and S' are all distinct and the union contains all the inconsistencies of S and S' . This means that in $K_{S \cup S'}$ all the atoms in K_S and $K_{S'}$ that have to be given the value B must still be given the value B . Hence, $I_C(S \cup S') \geq I_C(S) + I_C(S')$.
- m-Separability** The same counterexample as the one used for I_P can be used here. Then, $I_C(S \cup S') = 1 \neq I_C(S) + I_C(S') = 1 + 1 = 2$.
- m-Normalization** If S' is an inconsistency then by Theorem 1 $K_{S'}$ is a minimal inconsistent set and it suffices to give one atom the value B .
- m-Attenuation** The satisfaction of m-Normalization means that m-Attenuation is violated.
- m-Equal Conflict** This follows from the satisfaction of m-Normalization.
- m-Almost Consistency** The satisfaction of m-Normalization means that m-Almost Consistency is violated.
- I_{mv} m-Monotony** Let $S = \{R_1(0), \forall x(R_1(x) \rightarrow x > 0)\}$ and $S' = S \cup \{R_1(1)\}$. Then, $S \subset S'$ and $Inc(S) = Inc(S')$. We get $K_S = \{a_1, \neg a_1 \wedge b_1\}$ and $K_{S'} = \{a_1, a_2, \neg a_1 \wedge b_1\}$. So $I_{mv}(S) = \frac{2}{2} \not\geq I_{mv}(S') = \frac{2}{3}$.
- m-Free-Formula Independence** Consider the S' given above for the m-Monotonic property. Here $R_1(1)$ is m-free, but as shown above, $I_{mv}(S') = \frac{2}{3} \neq I_{mv}(S' \setminus \{R_1(1)\}) = 1$.
- m-Penalty** Suppose that $u \in \text{m-Problematic}(S)$ and $I_{mv}(S) = \frac{n}{d}$. Then $f(u) \in \text{Problematic}(K_S) \setminus \text{Problematic}(K_{S \setminus \{u\}})$. There may also be other information units and

requirements whose transformed atoms are in $\text{Problematic}(K_S) \setminus \text{Problematic}(K_{S \setminus \{u\}})$. However, all of them, except for $f(u)$ are still atoms in $K_{S \setminus \{u\}}$. Hence, $I_{mv}(K_{S \setminus \{u\}}) = \frac{n-\ell}{d-1}$ where $\ell \geq 1$, so $I_{mv}(S) > I_{mv}(S \setminus \{u\})$. The proof is similar for $c \in \text{m-Problematic}(S)$.

Dominance Let $S = \{R_1(0), R_1(2), \forall x(R_1(x) \rightarrow x > 1)\}$, $c = \forall x(R_1(x) \rightarrow x > 1)$, and $c' = \forall x(R_1(x) \rightarrow x > 0)$. Clearly, c logically implies c' . Also, $K_{S \cup \{c\}} = \{a_1, a_2, \neg a_1 \wedge b_1\}$, while $K_{S \cup \{c'\}} = \{a_1, a_2, \neg a_1 \wedge b_1, \neg a_1 \wedge b_2\}$. We obtain $I_{mv}(S \cup \{c\}) = \frac{2}{3} \not\geq I_{mv}(S \cup \{c'\}) = \frac{3}{4}$.

m-Super-Additivity The same counterexample used for I_B can be used here as $I_{mv}(S \cup S') = \frac{4}{4} = 1 \not\geq I_{mv}(S) + I_{mv}(S') = \frac{2}{2} + \frac{2}{2} = 2$.

m-Separability The same counterexample given above for m-Super-Additivity also satisfies the antecedent of m-Separability and the two I_{mv} values are not equal.

m-Normalization If S' is an inconsistency, then $K_{S'}$ is a minimal inconsistent set and $I_{mv}(S') = I_{mv}(K_{S'}) = \frac{n}{n} = 1$ where n is the number of atoms in $K_{S'}$.

m-Attenuation The satisfaction of m-Normalization means that m-Attenuation is violated.

m-Equal Conflict This follows from the satisfaction of m-Normalization.

m-Almost Consistency The satisfaction of m-Normalization means that m-Almost Consistency is violated. $\square$

This result shows that the satisfaction results for the m versions of the postulates are the same as in the propositional case for the first 6 measures, all of which are syntactic measures. However, for I_C there were several differences and there was also a difference for I_{mv} .

7 Complexity

As shown in [58] the complexity of computing inconsistency measures in propositional logic is generally high. The fact that even I_B , the simplest inconsistency measure, cannot be computed in polynomial time intuitively suggests that it is impossible to compute any propositional inconsistency measure efficiently for general knowledge bases. However, some problems related to inconsistency measurement become computationally easier when considering knowledge bases where every formula is of one of the following two types: *i*) a Horn clause, that is, a disjunction of literals where at most one literal in the formula is unnegated, or *ii*) a Krom clause, i.e., a disjunction of at most two literals [58]. In particular, considering the inconsistency measures considered in this paper, it turns out that only the computation of I_B becomes polynomial under these restrictions.

Here we consider the complexity of computing inconsistency measures for general information spaces. There are two steps in this process. The first step is transforming the general information space to a propositional knowledge base. Clearly, this step is the same for all inconsistency measures. The second step is computing the specific inconsistency measure of the transformed propositional knowledge base. We consider these issues separately, starting with the second one.

In the second step the results from [58] for general propositional knowledge bases provide an upper bound on the complexity of computing the value of the considered inconsistency measures. We cannot use the results for restricted propositional knowledge bases, where every formula is a Horn/Krom clause, because the transformation does not yield a knowledge base of one of those forms. Nevertheless, as the transformation yields a knowl-

edge base in a special format, we can show that the all syntactic measures considered but I_H can be computed in polynomial time, and the same holds even for the measure I_{mv} .

Proposition 1 *Given a propositional knowledge base K_S of the form in Definition 4, for each measure I_x with $x \in \{B, M, \#, P, mv\}$, $I_x(K_S)$ can be computed in polynomial time with respect to the size of K_S .*

Proof We first observe that all formulas in K_S belong to one of the following 4 types (cf. Definition 4):

1. a_i , a propositional atom (one for each information unit),
2. b_j , a propositional atom for each satisfied requirement,
3. $\neg b_j \wedge b_j$, a self-contradiction for each requirement of arity 0 that is violated,
4. a conjunction of some b_j with a conjunction of disjunctions of r negated a_i s, where r is the arity of c_j .

We start with the measure I_M . The result follows from the fact that the minimal inconsistent subsets of K_S can be obtained by scanning the formulas of type 3. and 4. A type 3. formula is a minimal inconsistent subset. A type 4. formula yields n minimal inconsistent subsets, where n is the number of conjuncts of the disjunctions. For example, for the formula $\phi = g(c_1) = (\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_3 \vee \neg a_4) \wedge b_1$ (where the requirement c_1 has arity 2) there are 3 minimal inconsistent subsets: $\{a_1, a_2, \phi\}$, $\{a_1, a_3, \phi\}$, and $\{a_3, a_4, \phi\}$. Doing this process takes linear time in the size of K_S .

Reasoning analogously, it can be shown that I_B , $I_\#$, and I_P can be computed in polynomial time as well. Finally, I_{mv} can be computed this way because it is the number of distinct atoms in formulas of type 3. and 4. $\square$

The situation is different for the measure I_H whose computation in the transformed knowledge base remains as hard as for the case of general propositional knowledge bases.

Theorem 3 *The problem of computing $I_H(K_S)$ for a propositional knowledge base K_S of the form in Definition 4 is $FP^{NP[\log n]}$ -complete.*

Proof The membership in $FP^{NP[\log n]}$ follows from the fact that the problem belongs to this class for general propositional knowledge bases [58].

We prove the hardness result by showing a reduction from the $FP^{NP[\log n]}$ -complete problem MIN-VERTEX COVER [48], which is defined as follows. Given a graph $G = \langle V, E \rangle$, compute the size of a smallest vertex cover for G . A vertex cover for $G = \langle V, E \rangle$ is a subset $V' \subseteq V$ such that for each edge $(u, w) \in E$ at least one of u or w belongs to V' .

Given an instance $G = \langle V, E \rangle$ of MIN-VERTEX COVER, we define a propositional knowledge base K_S of the form in Definition 4 as follows. K_S consists of an atom a_i for each $v_i \in V$ and of a formula $\phi_j = (\neg a_x \vee \neg a_y) \wedge b_j$ for each $e_j = (v_x, v_y) \in E$.

We now show that the size of a smallest vertex cover for G is k iff $I_H(D) = k$. As explained below, the statement follows from the fact that minimal inconsistent subsets of K_S are in a one-to-one correspondence with the edges of G . In fact, $MI(K_S) = \{\{a_x, a_y, \phi_j\} \mid e_j = (v_x, v_y) \in E\}$.

($\Rightarrow$) Let $V' \subseteq V$, with $|V'| = k$, be a smallest vertex cover for G . Since the set of minimal inconsistent subsets of K_S is $MI(K_S) = \{\{a_x, a_y, \phi_j\} \mid e_j = (v_x, v_y) \in E\}$, and since $V' \subseteq V$ contains at least one vertex from each edge in E , it follows that there is $X = \{a_i \mid v_i \in V'\}$ such that $X \subseteq K_S$, $|X| = k$, and for each $M \in MI(K_S)$, $X \cap M \neq \emptyset$;

moreover, the size of X is minimum, otherwise V' would not be a smallest vertex cover. Therefore, $I_H(K_S) = k$.

($\Leftarrow$) If $I_H(D) = k$, then there is $X \subseteq K_S$ such that $|X| = k$ and $\forall M \in \text{MI}(K_S)$, $X \cap M \neq \emptyset$, and X is a smallest set that has a nonempty intersection with every minimal inconsistent subset. Starting from X we define a minimum vertex cover V' of size k for G as follows.

Case A. If X contains only atoms a_i s, then $V' = \{v_i \mid a_i \in X\}$. Indeed, since X has a nonempty intersection with every minimal inconsistent subset $M \in \text{MI}(K_S)$, and every $M \in \text{MI}(K_S)$ one-to-one corresponds to an edge in E , it follows that V' is a minimum vertex cover of size k for G .

Case B. X contains some formula ϕ_j . We show that there is an equivalent set of formulas $X^* \subseteq K_S$ such that (i) $I_H(K_S) = |X^*| = |X|$ and (ii) $\forall M \in \text{MI}(K_S)$, $X^* \cap M \neq \emptyset$. Let X^* be the set of formulas obtained from X by replacing every formula $\phi_j = (\neg a_x \vee \neg a_y) \wedge b_j \in X$ with a_x (or, equivalently, a_y). Since each formula of form $\phi_j = (\neg a_x \vee \neg a_y) \wedge b_j$ belongs to exactly one minimal inconsistent subset $\{a_x, a_y, \phi_j\} \in \text{MI}(K_S)$, by replacing ϕ_j with a_x (or a_y) we obtain that $\{M \mid M \in \text{MI}(K_S), X^* \cap M \neq \emptyset\}$ coincides with $\{M \mid M \in \text{MI}(K_S), X \cap M \neq \emptyset\}$. Moreover, since X is a smallest set that has a nonempty intersection with every minimal inconsistent subset, $|X^*| = |X|$; otherwise $I_H(K_S) < |X|$. That is, no formula $\phi_j = (\neg a_x \vee \neg a_y) \wedge b_j \in X^*$ can be replaced with a formula a_x (or a_y) already in X^* . Finally, let $V' = \{v_i \mid a_i \in X^*\}$; reasoning as above, it follows that V' is a minimum vertex cover of size k for G . $\square$

The following theorem states that computing the value of I_C is $FP^{NP[\log n]}$ -complete. Also in this case, the computation of this measure for K_S is not easier than that for general propositional knowledge bases, though K_S has a specific form.

Theorem 4 *The problem of computing $I_C(K_S)$ for a propositional knowledge base K_S of the form in Definition 4 is $FP^{NP[\log n]}$ -complete.*

Proof The membership in $FP^{NP[\log n]}$ follows from the fact that the problem belongs to this class for general propositional knowledge bases [58]. The hardness for $FP^{NP[\log n]}$ can be proved by showing a reduction from MIN-VERTEX COVER [48]. It suffices to use the construction provided in the proof of Theorem 3. In fact, the propositional knowledge base K_S defined starting from an instance of MIN-VERTEX COVER is such that $I_C(K_S) = I_H(K_S)$, and in particular it can be shown that an atom a_i is assigned the truth value B iff the corresponding vertex v_i belongs to a minimal vertex cover, from which the statement follows. $\square$

As for the complexity of computing $I_{nc}(K_S)$, we do not have a better result than the $FP^{\Sigma_2^P[\log n]}$ -membership provided in [58] for the case of general propositional knowledge bases. Finding a tighter result remains open. In fact, this problem is also open for propositional knowledge bases where, to the best of our knowledge, no hardness for the above-mentioned class has been proved.

Now we go back to the first step of the process of computing inconsistency measures for a general information space S , that is the transformation. In fact, this is the bigger issue because it must be done for all inconsistency measures and requires finding $Inc(S)$.

Proposition 2 *The complexity of computing the transformation from a general information space $S = \langle F, U, C \rangle$ to the propositional KB K_S is*

$$\mathcal{O} \left(\sum_{c \in C} \text{Comb}(|U|, \text{arity}(c)) \cdot z(c) \right)$$

where:

- $\text{Comb}(n, k)$ denotes the number of k -combinations from a given set of n elements;
- $\text{arity}(c)$ denotes the arity of a requirement $c \in C$; and
- $z(c)$ is the cost of checking whether an $\text{arity}(c)$ -tuple of information units violates requirement $c \in C$.

Proof The costly part of the transformation is finding all minimal inconsistencies of S for each requirement $c \in C$. For each $c \in C$, in the worst case, all $\text{arity}(c)$ -combinations of U should be considered, that is, a number of elements equal to $\text{Comb}(|U|, \text{arity}(c))$. Thus, the cost of the transformation is bounded by $\sum_{c \in C} \text{Comb}(|U|, \text{arity}(c)) \cdot z(c)$, where $z(c)$ is the cost of checking whether an $\text{arity}(c)$ -tuple of information units violates requirement c . $\square$

Thus, in general, the cost of the transformation is exponential w.r.t. the number of information units and requirements of a general information space. However, it becomes polynomial under some reasonable conditions. For instance, consider first requirements of arity 0. A simple such requirement may be $\exists y R(1, y)$, which can be checked in linear time. However, the requirement $\forall x, y (R_1(x, y) \rightarrow \exists z R_2(y, z))$ needs a loop and takes quadratic time. Consider next requirements of arity 1. A simple example is $\forall x, y (R(x, y) \rightarrow x < 0 \wedge y > 1)$ which is a matter of checking all the tuples, leading again to a polynomial case. We get the following result.

Proposition 3 *Given a general information space $S = \langle F, U, C \rangle$, if i) the arity of requirements in C is bounded by a constant k , and ii) the cost of checking violations of $c \in C$ is bounded by a polynomial $p(|U|)$, then the complexity of computing the transformation from S to the propositional KB K_S is $\mathcal{O}(|C| \cdot p(|U|) \cdot |U|^k)$.*

Proof This follows from the fact that $\text{Comb}(n, k) \leq n^k$, where k is assumed to be a constant. $\square$

Therefore, since under the assumptions of Proposition 3, the size of the transformed knowledge base is polynomial w.r.t. the number of information units and requirements of the input general information space, using the result of Proposition 1, we obtain the following corollary.

Corollary 1 *Given a general information space $S = \langle F, U, C \rangle$, if i) the arity of requirements in C is bounded by a constant k , and ii) the cost of checking violations of $c \in C$ is bounded by a polynomial $p(|U|)$, then $I_x(K_S)$ can be computed in polynomial time for each $x \in \{B, M, \#, P, mv\}$.*

8 Conclusions and future work

As inconsistency in real-world information systems can not be easily avoided, many inconsistency-tolerant approaches have been developed to live with inconsistency [21], and provide appropriate mechanisms to handle inconsistent data [5, 8, 18]. For instance, approaches for dealing with inconsistency in databases include consistent query answering frameworks [2, 9], data repairing [1, 39, 41, 43], interactive data repairing and cleaning systems (e.g. [20, 33, 34]), as well as interactive data exploration tools [7, 22]. An important issue in such kinds of situations is measuring the amount of inconsistency to assess its nature and understand the degree of the dirtiness of data.

In this paper, we developed a general approach for measuring inconsistency in general information spaces which encompasses various ways in which information is stored in real-world systems. Before outlining directions for future work, we discuss some advantages and limitations of our approach.

Independence of the data representation and choice of the inconsistency measure.

An important advantage of defining inconsistency measures for general information spaces is its wide range of use. Consider an inconsistency measure defined specifically for relational databases, as proposed for instance in [3, 14–16, 44, 51]. Such a measure allows for comparing two relational databases and determining if one is less inconsistent than the other one or if they have the same inconsistency. But using the concept of a general information space and the uniformity of the definition of an inconsistency measure allows the comparison of the inconsistency of different types of information systems, such as a relational database and a graph database. For instance, we can compare the inconsistency measures obtained for the 4 examples using the 8 inconsistency measures we have considered. These inconsistency measures measure different aspects of the inconsistencies. Hence we find that for some measures one of the examples is more inconsistent than another but for a different measure it is the reverse. So for comparing these general information spaces we need to decide which aspect we really want to measure. In particular, measuring the number of inconsistent subsets, possibly by weighing them, that is, using I_M or $I_{\#}$, are good ways to get a general sense of the amount of inconsistency. Using I_M we find that the answer is largest for the first example, the relational database, where $I_M(S) = 12$. The values for the other 3 examples are closer together: 6 for the graph database, 4 for the spatio-temporal database, and 7 for the Blocks world example. However, since I_M counts the number of minimal inconsistent subsets, it may not be appropriate for cases where, for instance, a single information unit causes many inconsistencies. Consider a case where the address attribute functionally depends on the id attribute in a relation containing data about customers, and there is a single customer address in a tuple t that is misspelled, while it is correct in all the other n tuples for the same customer id. This generates n two-element minimal inconsistent sets, each of them having the wrong information unit t in common. Hence, measure I_M would count n minimal inconsistent subsets in the transformed propositional knowledge base, even though they are all due to a single incorrect value in t . Instead, we could use the “repair” measure I_H , which counts the minimal number of formulas in the translation whose deletion resolves all inconsistencies; I_H counts such an information unit only once. This example shows that the choice of the inconsistency measure is critical for deciding how inconsistencies are counted. For the sake of the presentation we have focused on showing how our approach works with the inconsistency measures listed in Definition 2. But it is important to observe that the transformation creates a propositional knowledge base; hence all propositional inconsistency measures ever proposed are applicable. Therefore, issues that

can be ascribed to the choice of the inconsistency measures are prone to be easily solved in our framework, as it is apt to deal with any propositional inconsistency measure having some desired properties, e.g., those of the repair-based measure I_H in the example above.

Existential requirements and trade-off between expressivity and tractability.

Our transformation-based approach maps violations of general information spaces' requirements to propositional formulas where the propositions are associated with the information units. An existential requirement asserts the existence of an information unit; the lack of such a unit causes an inconsistency that we transform into an inconsistency in propositional logic. Specifically, we obtain a self-contradiction for existential requirements, for which no information-unit deletion would negate the requirement's violation (recall that we assign arity equal to zero to such requirements). For an existential requirement, an alternative approach would be making explicit what are all the possible options for missing information units. But this would mean dealing with a potentially huge set of information units from attributes' domains. However, we might be able to place a limit to some information units only. For instance, if the existence of an object (e.g., a person) is required in a given relation (e.g. *Family* relation), as in the case of requirement c_8 in Section 5.1, then we could assume that the only objects explicitly appearing in a relation (e.g., *Person*) are allowed. Even so, this seems to be an arbitrary decision. Therefore, to close the loop, what remains to be done is to attempt to design a formal language to express such limitations on existence for requirements of general information spaces and then provide an expanded transformation dealing with them, a task outside of this work's aim and scope but that we will pursue as part of our research project. It is worth noting that some advantages of our framework, such as the nice trade-off between expressivity and polynomial time computation, could be lost this way. As a matter of fact, most of the results of the paper would be substantially affected by dealing with existential requirements as outlined above.

Additional directions for future work Besides the above-mentioned attractive research possibility, we plan to continue our work on measuring inconsistency in general information spaces in several other directions. We wish to define *disjunctive* general information spaces where the information units may be disjunctions. This has additional complications: for example, it may be that several requirements are needed for an inconsistency. In particular, we are interested in handling null values. In another direction, consider that although we showed the applicability of our approach to spatio-temporal databases [28], how to encode probabilistic spatio-temporal knowledge bases [49, 50], and in general probabilistic information, into a general information space needs further investigation that may lead to a definition of a concept of *probabilistic* general information space. Finally, we also plan to study what aspects of inconsistency the various inconsistency measures actually measure to determine which ones are the most appropriate to use for general information spaces.

Acknowledgements The authors wish to thank the reviewers for their helpful comments as well as for suggestions on possible extensions of our approach, particularly the treatment of existential requirements outlined in Section 8.

Funding Open access funding provided by Università della Calabria within the CRUI-CARE Agreement.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence,

and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Afrati, F.N., Kolaitis, P.G.: Repair checking in inconsistent databases: algorithms and complexity. In: Proceedings of International Conference on Database Theory (ICDT), pp. 31–41 (2009)
2. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Proceedings of ACM Symposium on Principles of Database Systems (PODS), pp. 68–79 (1999)
3. Bertossi, L.E.: Measuring and computing database inconsistency via repairs. In: Proceedings of International Conference on Scalable Uncertainty Management (SUM), pp. 368–372 (2018)
4. Bertossi, L.E.: Repair-based degrees of database inconsistency. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR), pp. 195–209 (2019)
5. Bertossi, L.E., Hunter, A., Schaub, T.: Introduction to Inconsistency Tolerance. In: Inconsistency Tolerance, pp. 1–14 (2005)
6. Besnard, P., Grant, J.: Relative inconsistency measures. *Artif. Intell.* **280**, 103231 (2020)
7. Bleifuß, T., Bornemann, L., Kalashnikov, D.V., Naumann, F., Srivastava, D.: Dbhex: Interactive exploration of data and schema change. In: Proceedings of Biennial Conference on Innovative Data Systems Research (CIDR) (2019)
8. Calautti, M., Caroprese, L., Fazzinga, B., Flesca, S., Furfaro, F., Greco, S., Molinaro, C., Parisi, F., Pugliese, A., Saccà, D., Trubitsyna, I., Zumpano, E.: Dealing with Inconsistency in Databases: an Overview. In: A Comprehensive Guide through the Italian Database Research over the Last 25 Years., pp. 143–167. Springer International Publishing (2018)
9. Calautti, M., Libkin, L., Pieris, A.: An operational approach to consistent query answering. In: Proceedings of ACM Symposium on Principles of Database Systems (PODS), pp. 239–251 (2018)
10. Chomicki, J., Marcinkowski, J.: Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.* **197**(1–2), 90–121 (2005)
11. Codd, E.F.: The relational model for database management, version 2. Addison-Wesley (1990)
12. De Bona, G., Grant, J., Hunter, A., Konieczny, S.: Classifying inconsistency measures using graphs. *J. Artif. Intell. Res.* **66**, 937–987 (2019)
13. De Bona, G., Hunter, A.: Localizing iceberg inconsistencies. *Artif. Intell.* **246**, 118–151 (2017)
14. Decker, H.: Inconsistency-tolerant database repairs and simplified repair checking by measure-based integrity checking. *Trans. Large Scale Data Knowl. Centered Syst.* **34**, 153–183 (2017)
15. Decker, H.: Measuring Inconsistency in Information, *Studies in Logic*. In: Grant, J., Martinez, M.V. (eds.), vol. 73, pp. 271–311. College Publications, Norway (2018)
16. Decker, H., Misra, S.: Database inconsistency measures and their applications. In: Proceedings of International Conference on Information and Software Technologies (ICIST), pp. 254–265 (2017)
17. Doder, D., Raskovic, M., Markovic, Z., Ognjanovic, Z.: Measures of inconsistency and defaults. *Int. J. Approx. Reason.* **51**(7), 832–845 (2010)
18. Eiter, T., Fink, M., Schüller, P., Weinzierl, A.: Finding explanations of inconsistency in multi-context systems. *Artif. Intell.* **216**, 233–274 (2014)
19. Fagin, R.: Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.* **2**(3), 262–278 (1977)
20. Fazzinga, B., Flesca, S., Furfaro, F., Parisi, F.: DART: A Data Acquisition and Repairing Tool. In: EDBT Workshops on Inconsistency and Incompleteness in Databases (IIDB), pp. 297–317 (2006)
21. Gabbay, D.M., Hunter, A.: Making inconsistency respectable: a logical framework for inconsistency in reasoning. In: Proceedings of International Workshop on Fundamentals of Artificial Intelligence Research (FAIR), pp. 19–32 (1991)
22. Giuzio, A., Mecca, G., Quintarelli, E., Roveri, M., Santoro, D., Tanca, L.: INDIANA: an interactive system for assisting database exploration. *Inf. Syst.* **83**, 40–56 (2019)
23. Grant, J., Hunter, A.: Measuring inconsistency in knowledgebases. *J. Intell. Inf. Syst.* **27**(2), 159–184 (2006)

24. Grant, J., Hunter, A.: Measuring consistency gain and information loss in stepwise inconsistency resolution. In: *Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pp. 362–373 (2011)
25. Grant, J., Hunter, A.: Distance-based measures of inconsistency. In: *Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pp. 230–241 (2013)
26. Grant, J., Martinez, M.V.: *Measuring inconsistency in information*. College Publications (2018)
27. Grant, J., Martinez, M.V. (eds.): *Measuring Inconsistency in Information, Studies in Logic*, vol. 73. College Publications, Norway (2018)
28. Grant, J., Martinez, M.V., Molinaro, C., Parisi, F.: Measuring Inconsistency in Information, *Studies in Logic*. In: Grant, J., Martinez, M.V. (eds.), vol. 73, pp. 313–342. College Publications, Norway
29. Grant, J., Minker, J.: Inferences for numerical dependencies. *Theor. Comput. Sci.* **41**, 271–287 (1985)
30. Grant, J., Minker, J.: Normalization and axiomatization for numerical dependencies. *Inf. Control.* **65**(1), 1–17 (1985)
31. Grant, J., Parisi, F.: Measuring inconsistency in a general information space. In: Herzig, A., Kontinen, J. (eds.) *Proceedings of International Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, vol. 12012, pp. 140–156 (2020)
32. Gupta, N., Nau, D.S.: Complexity results for blocks-world planning. In: *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pp. 629–633 (1991)
33. Hao, S., Tang, N., Li, G., He, J., Ta, N., Feng, J.: A novel cost-based model for data repairing. *IEEE Trans. Knowl. Data Eng.* **29**(4), 727–742 (2017)
34. He, J., Veltri, E., Santoro, D., Li, G., Mecca, G., Papotti, P., Tang, N.: Interactive and deterministic data cleaning. In: *Proceedings of International Conference on Management of Data (SIGMOD)*, pp. 893–907 (2016)
35. Hunter, A.: Measuring inconsistency in knowledge via quasi-classical models. In: *Proceedings of National Conference on Artificial Intelligence and Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pp. 68–73 (2002)
36. Hunter, A., Konieczny, S.: Measuring inconsistency through minimal inconsistent sets. In: *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 358–366 (2008)
37. Hunter, A., Konieczny, S.: On the measure of conflicts: Shapley inconsistency values. *Artif. Intell.* **174**(14), 1007–1026 (2010)
38. Jabbour, S., Ma, Y., Raddaoui, B., Sais, L.: Quantifying conflicts in propositional logic through prime implicates. *Int. J. Approx. Reason.* **89**, 27–40 (2017)
39. Kimelfeld, B., Livshits, E., Peterfreund, L.: Detecting ambiguity in prioritized database repairing. In: *Proceedings of International Conference on Database Theory (ICDT)*, pp. 17:1–17:20 (2017)
40. Knight, K.: Measuring inconsistency. *J. Philos. Log.* **31**(1), 77–98 (2002)
41. Livshits, E., Kimelfeld, B.: Counting and enumerating (preferred) database repairs. In: *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pp. 289–301 (2017)
42. Ma, Y., Qi, G., Xiao, G., Hitzler, P., Lin, Z.: An anytime algorithm for computing inconsistency measurement. In: *Proceedings of International Conference on Knowledge Science, Engineering and Management (KSEM)*, pp. 29–40 (2009)
43. Martinez, M.V., Parisi, F., Pugliese, A., Simari, G.I., Subrahmanian, V.S.: Policy-based inconsistency management in relational databases. *Int. J. Approx. Reason.* **55**(2), 501–528 (2014)
44. Martinez, M.V., Pugliese, A., Simari, G.I., Subrahmanian, V.S., Prade, H.: How dirty is your relational database? an axiomatic approach. In: *Proceedings of European Conference Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pp. 103–114 (2007)
45. McAreavey, K., Liu, W., Miller, P.C.: Computational approaches to finding and measuring inconsistency in arbitrary knowledge bases. *Int. J. Approx. Reason.* **55**(8), 1659–1693 (2014)
46. Mu, K., Jin, Z., Lu, R., Liu, W.: Measuring inconsistency in requirements specifications. In: *Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pp. 440–451 (2005)
47. Mu, K., Liu, W., Jin, Z., Bell, D.A.: A syntax-based approach to measuring the degree of inconsistency for belief bases. *Int. J. Approx. Reason.* **52**(7), 978–999 (2011)
48. Papadimitriou, C.M.: *Computational Complexity*. Addison-Wesley, Reading (1994)
49. Parisi, F., Grant, J.: Knowledge representation in probabilistic spatio-temporal knowledge bases. *J. Artif. Intell. Res.* **55**, 743–798 (2016)
50. Parisi, F., Grant, J.: On repairing and querying inconsistent probabilistic spatio-temporal databases. *Int. J. Approx. Reason.* **84**, 41–74 (2017)

51. Parisi, F., Grant, J.: On measuring inconsistency in relational databases with denial constraints. In: Proceedings of the 24th European Conference on Artificial Intelligence (ECAI), pp. 857–864 (2020)
52. Parisi, F., Park, N., Pugliese, A., Subrahmanian, V.S.: Top-k user-defined vertex scoring queries in edge-labeled graph databases. *ACM Trans. Web (TWEB)* **12**(4), 21:1–21:35 (2018)
53. Parker, A., Subrahmanian, V.S., Grant, J.: A logical formulation of probabilistic spatial databases. *IEEE Trans. Knowl. Data Eng. (TKDE)* **19**(11), 1541–1556 (2007)
54. Robinson, I., Webber, J., Eifrem, E.: *Graph Databases*. O'Reilly Media, Inc (2013)
55. Thimm, M.: On the expressivity of inconsistency measures. *Artif. Intell.* **234**, 120–151 (2016)
56. Thimm, M.: Stream-based inconsistency measurement. *Int. J. Approx. Reason.* **68**, 68–87 (2016)
57. Thimm, M.: Measuring Inconsistency in Information, *Studies in Logic*. In: Grant, J., Martinez, M.V. (eds.), vol. 73, pp. 19–60. College Publications, Norway (2018)
58. Thimm, M., Wallner, J.P.: On the complexity of inconsistency measurement. *Artif. Intell.* **275**, 411–456 (2019)
59. Ulbricht, M., Thimm, M., Brewka, G.: Handling and measuring inconsistency in non-monotonic logics. *Artif. Intell.* **103344**, 286 (2020)
60. Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: Proceedings of Symposium on Theory of Computing (STOC), pp. 137–146 (1982)
61. Wijsen, J.: On condensing database repairs obtained by tuple deletions. In: Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA), pp. 849–853. IEEE Computer Society (2005)
62. Xiao, G., Ma, Y.: Inconsistency measurement based on variables in minimal unsatisfiable subsets. In: Proceedings of European Conference on Artificial Intelligence (ECAI), pp. 864–869. IOS Press (2012)
63. Zhang, X., Wang, K., Wang, Z., Ma, Y., Qi, G., Feng, Z.: A distance-based framework for inconsistency-tolerant reasoning and inconsistency measurement in DL-lite. *Int. J. Approx. Reason.* **89**, 58–79 (2017)
64. Zhou, L., Huang, H., Qi, G., Ma, Y., Huang, Z., Qu, Y.: Measuring inconsistency in DL-lite ontologies. In: Proceedings of International Conference on Web Intelligence (WI), pp. 349–356 (2009)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.