

Measuring Inconsistencies in Ontologies

Xi Deng, Volker Haarslev, and Nematollaah Shiri

Department of Computer Science and Software Engineering
Concordia University, Montreal, Canada
{xi_deng,haarslev,shiri}@encs.concordia.ca

Abstract. In this paper, we propose a novel approach to measure inconsistencies in ontologies based on Shapley values, which are originally proposed for game theory. This measure can be used to identify which axioms in an input ontology or which parts of these axioms need to be removed or modified in order to make the input consistent. We also propose optimization techniques to improve the efficiency of computing Shapley values. The proposed approach is independent of a particular ontology language or a particular reasoning system used. Application of this approach can improve the quality of ontology diagnosis and repair in general.

1 Introduction

Ontologies play an important role in the Semantic Web as they provide a common shared model to represent a domain and to reason about the objects in the domain. As the size of the ontologies grows and applications developed become more complex, inconsistency becomes inevitable in the design and development of ontologies. According to the classical *ex contradictione quodlibet* (ECQ) principle, anything follows from an inconsistent ontology is useless. In order to help users to resolve the inconsistencies in ontologies, several approaches to identify and explain the cause of these inconsistencies have been proposed [1–3]. An assumption often made in these approaches is that all inconsistencies are equally “bad”. However, as shown in real world applications, it is possible for an ontology to contain two or more sources of inconsistencies and they may have different impact towards the inconsistencies. They may not necessarily contain the same contradiction and the same information, and may have overlapping content. The following are just two possible scenarios.

- Non-overlapping: there are more than one set of axioms that are needed to produce an inconsistency in an ontology and they are independent of one another.

Suppose K' and K'' are two inconsistent subsets of ontology \mathcal{O} . When we say \mathcal{O} has non-overlapping sources of inconsistencies, it means that $K' \cap K'' = \emptyset$. Note that inconsistency might occur at different levels: in the level of a single axiom, and in the level of sets of axioms. For example, consider

$K' = \{C \sqcup \neg C \sqsubseteq C \sqcap \neg C\}$ and $K'' = \{\top \sqsubseteq \exists R.B, \top \sqsubseteq \forall R.\neg B\}$.¹ The axiom in K' is inherently inconsistent, while both of the two axioms in K'' are responsible to produce a contradiction.

- Overlapping: there are more than one set of axioms that are needed to produce an inconsistency in an ontology and they are interweaved with one another.

\mathcal{O} having overlapping sources of inconsistencies means that $K' \cap K'' \neq \emptyset$. When two sets of inconsistent axioms are overlapping, it indicates that certain axioms contribute more to the inconsistencies and these axioms are possibly more problematic than others. It is most likely the case that removing one of these axioms from \mathcal{O} will result in resolving other inconsistencies as well.

In this paper we propose a quantitative measure of inconsistencies in ontologies which gives users guidelines on priorities of the axioms to be removed and their consequences. Our approach borrows some ideas from [4], which presents an approach of using the Shapley value to obtain an inconsistency measure for propositional logic. It can be applied to clauses instead of axioms. Since clauses are more fine-grained than axioms, it allows us to take a deeper look inside the axioms and find out which proportion of the axiom contributes to the inconsistency. We also discuss the relationship between the inconsistency measure and the minimal inconsistent subsets of ontologies. The computational complexity of calculating the Shapley value is at least Exp-time, which shows that it does not scale well in general [5]. Therefore we propose to optimize the calculation based on the structural relevance of the axioms and properties of the defined inconsistency measure.

The main contribution of this paper is twofold: we combine previously known game theory strategies into ontology reasoning and present a measure to systematically evaluate the inconsistencies in ontologies. To the best of our knowledge, this is the first work in Description Logics towards providing a quantitative measure of inconsistencies. This approach is independent of a particular species of ontology languages or a particular reasoning system used. Moreover, we illustrate how the application of this method can improve the quality of ontology diagnosis and repair.

The remainder of the paper is organized as follows: Section 2 presents a brief introduction of Description Logics, the underlying logics of the Web Ontology Language OWL. Section 3 introduces the basic concepts used in our approach. Section 4 describes the algorithms. The paper closes with a summary and also a discussion of future work.

2 Preliminaries

In this section, we first review some basic concepts and terms related to the ontology languages in the Semantic Web, and their relationships to Description Logics (DLs). We also define the notion of inconsistency in an ontology.

¹ Please refer to Section 2 for the definition of the notations.

2.1 Ontologies in the Semantic Web

The Web Ontology Language (OWL) has been recommended as the standard web ontology language by the World Wide Web Consortium (W3C) [6]. It is a machine-readable language for sharing and reasoning information on the Internet. OWL is an extension of the Resource Description Framework (RDF) and a revision of the DAML+OIL Web Ontology Language.

OWL represents the domain of interest by defining hierarchies of classes and properties. An OWL ontology consists of axioms and facts. Axioms define intensional knowledge by building relationships between classes and properties. Facts describe the extensional knowledge about individuals. OWL currently has three flavors: OWL Lite, OWL DL, and OWL Full. OWL Full contains OWL DL, which in turn contains OWL Lite. OWL DL and OWL Lite correspond semantically with certain Description Logic languages. Reasoning tasks are undecidable in OWL Full and currently there is no reasoner that supports reasoning of every feature of OWL Full.

2.2 Description Logics

We shall not give a detailed introduction of Description Logics here, but refer the interested reader to [7]. Description logics are a family of concept-based knowledge representation formalisms. It represents the knowledge of a domain by first defining the relevant concepts of the domain. These concepts are used to specify properties of the objects in the domain. Typically a DL language has two parts: *terminology* (TBox) and *assertion* (ABox). The TBox includes intensional knowledge in the form of axioms whereas the ABox contains the extensional knowledge that is specific to elements in the domain, called individuals. The TBox together with the ABox is called a *knowledge base* in DL. In the TBox, basic descriptions are *atomic concepts*, designated by unary predicates, and *atomic roles*, designated by binary predicates to express relationships between individuals. Concept descriptions can be built on atomic concepts by iteratively applying constructors such as intersection, union, negation, value restriction and existential quantification. Axioms express how concepts and roles are related to each other. Generally, it is a statement of the form $C \sqsubseteq D$, read as “concept C is subsumed by concept D ”, or $C \equiv D$, indicating that $C \sqsubseteq D$ and $D \sqsubseteq C$, where C and D are concept descriptions. An ABox is a set of assertions that of the form $C(a)$ and $R(a, b)$, where R is a role, and a, b are individuals.

An interpretation \mathcal{I} defines the formal semantics of concepts, roles, and individuals. It consists of a non-empty set $\Delta^{\mathcal{I}}$, called the domain. The interpretation function \mathcal{I} maps every atomic concept A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and maps every atomic role R to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. In addition, \mathcal{I} maps each individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. An interpretation \mathcal{I} satisfies $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. It satisfies $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. It satisfies $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and it satisfies $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

The basic inference services in TBoxes include satisfiability, subsumption, equivalence, and disjointness. A concept C in a TBox \mathcal{T} is said to be *satisfiable*

w.r.t \mathcal{T} if there exists a model of \mathcal{T} (that is an interpretation \mathcal{I} that satisfies the axioms of \mathcal{T}), such that $C^{\mathcal{I}}$ is nonempty. The other three inference services can be reduced to (un)satisfiability. Another important reasoning service in TBoxes is to check whether a TBox \mathcal{T} is *consistent*, i.e., whether there exists a model for \mathcal{T} . The basic reasoning tasks in ABoxes include instance checking, realization, and retrieval. The instance check verifies if a given individual is an instance of a specified concept. The realization finds the most specific concept that an individual is an instance of. The retrieval finds the individuals in the knowledge base that are instances of a given concept. An ABox \mathcal{A} is *consistent* w.r.t a TBox \mathcal{T} , if there is an interpretation that is a model of both \mathcal{A} and \mathcal{T} (that is an interpretation \mathcal{I} that both satisfies the axioms of \mathcal{T} and the assertions of \mathcal{A}). Similar to the inference services in TBoxes, the other three inference services in ABoxes can also be reduced to the consistency problem of ABoxes. In this paper, we use the term “consistency” to refer to consistency problems in both TBoxes and ABoxes. We will also discuss the measure regarding unsatisfiable concepts.

Roughly speaking, a concept in DL is referred to as a class in OWL. A role in DL is a property in OWL. The terms axioms and individuals have the same meaning in DL and OWL. OWL DL is based in part on the DL $SHOIN(\mathcal{D})$, which includes special constructors such as oneOf, transitive properties, inverse properties and datatype properties, and its subset OWL Lite which is based on the less expressive DL $SHIF(\mathcal{D})$, is $SHOIN(\mathcal{D})$ without the oneOf constructor and with the number restriction constructors limited to 0 and 1. Due to the close connection between OWL and DLs, in this paper, we will make no distinction between ontologies and knowledge bases in DL, and the examples are given mainly in DL syntax. The DL languages that we work on are those for which consistency checking is decidable.

3 Inconsistency Measures

In this section, we study methods of measuring inconsistencies in DL knowledge bases. The idea is to first define an inconsistency value, and then take it as the characteristic function to compute the Shapley value.

3.1 Motivating Example

The following example is adapted from [3], which we modified to be inconsistent. In the example, $A, B, C, D, A1$ to $A6$ denote concepts, and a and b are individuals.

-
- | | |
|--|--|
| 1. $A1 \sqsubseteq A2 \sqcap \neg A \sqcap A3$ | 2. $A2 \sqsubseteq A \sqcap A4$ |
| 3. $A3 \sqsubseteq A5 \sqcap A4$ | 4. $A4 \sqsubseteq C \sqcap \forall S.B$ |
| 5. $A5 \sqsubseteq \exists S.\neg B$ | 6. $D \sqcup \neg D \sqsubseteq D \sqcap \neg D$ |
| 7. $A1(a)$ | 8. $A3(b)$ |
| 9. $A6 \equiv D$ | |
-

A complete DL reasoner, such as FaCT++ [8], RACER [9] or Pellet [10], reports this knowledge base to be inconsistent. However, they can not provide crucial information, e.g., that there are four inconsistent subsets (3, 4, 5 and 8; 1, 2, and 7; 1, 3, 4, 5 and 7; 6) in this knowledge base, and that one axiom (axiom 6) is inherently inconsistent. We will use this as our running example throughout the paper to show how the hidden information can be unraveled using our method.

3.2 Definitions

In this section we review some definitions of the Shapley value in game theory [4], which we tailor for use in DL in this work.

Definition 1. *Given a set of axioms and assertions in a knowledge base K , a characteristic function $v : 2^K \rightarrow \mathbb{R}$ assigns a value to each coalition K' , where $K' \subseteq K$.*

An example of the characteristic function is the drastic inconsistency value, which assigns 1 to a set of axioms if it is inconsistent, and 0 to the set if it is consistent.

Definition 2. *For a set of axioms and assertions $K' \subseteq K$, the drastic inconsistency value of K' is defined as:*

$$I_d(K') = \begin{cases} 0 & \text{if } K' \text{ is consistent or } K' \text{ is empty} \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

Example 1. Some drastic inconsistency values of the running example are as follows, where we only show some of those with the inconsistency value 1, as well as some consistent ones (with this value being 0).²

$$\begin{array}{lll} I_d(\{1\}) = 0 & I_d(\{2\}) = 0 & I_d(\{3\}) = 0 \\ I_d(\{1, 2\}) = 0 & I_d(\{3, 4, 5\}) = 0 & I_d(\{1, 2, 6\}) = 1 \\ I_d(\{3, 4, 5, 8\}) = 1 & I_d(\{3, 4, 5, 6\}) = 1 & \\ I_d(\{1, 3, 4, 5, 7\}) = 1 & & \end{array}$$

As shown above, the coalition $\{1, 2, 6\}$ has the inconsistency value 1. Axiom 6 is often of a great value for a coalition it joins. For example, it can bring 1 to $\{2, 6\}$ for making the coalition $\{1, 2, 6\}$. And it can also bring 1 to the coalition $\{3, 4, 5, 6\}$.

Similarly, we can define another characteristic function which assigns 0 to a set of axioms if a concept A is satisfiable and 1 otherwise.

² For the sake of simplicity, we refer to the axioms and assertions by their numbers.

Definition 3. *The concept-related inconsistency value of a set of axioms K' w.r.t. a concept A (A occurs in K) is defined as:*

$$I_A(K) = \begin{cases} 0 & \text{if } A \text{ is satisfiable w.r.t. } K' \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

Example 2. Some of the concept-related inconsistency values of the working example are:

$$\begin{array}{ll} I_{A1}(\{1, 2\}) = 1 & I_{A1}(\{1, 3, 4, 5\}) = 1 \\ I_{A3}(\{3, 4, 5\}) = 1 & I_{A3}(\{3, 4, 9\}) = 0 \end{array}$$

An inconsistent measure is to quantify the contribution of each axiom or assertion in the knowledge base to the overall inconsistencies. The higher the measure is, the more weight an axiom carries in contributing to the inconsistencies. Shapley [11] proposed such a measure, known as Shapley values, in the context of game theory in 1953, which describes a fair allocation of gains obtained by the cooperation among several agents.

The Shapley value is defined for a game that has n agents. In the game, the agents can form coalitions, which is a subset of the n agents. Each coalition has a gain when all its members work together as a team. A question which may arise here is “which agent contributes the most to different coalitions?” A solution to this problem can help determine which agent values more to the game than the others. The Shapley value is proposed to tackle this problem. The basic idea is as follows. Suppose the agents join a coalition according to a certain order, and the payoff of an agent in this coalition is its marginal contribution to the gain of the coalition. The Shapley value takes all the possible orders of the coalition formation into account and averages the agent’s marginal contribution over them.

As the inconsistency checking can be deemed as a game, each axiom (or assertion) in the knowledge base can be deemed as an agent. Analogously, the contribution of each axiom (or assertion) to the inconsistencies can be measured using the Shapley value.

Let K be a knowledge base, σ_K be the set of all permutations on K , and $n = |K|$ be the cardinality of K . Given an order $\sigma \in \sigma_K$, we use p_σ^α to denote the set of all the axioms and assertions in σ that appear before an axiom (or an assertion) α .

Definition 4. *The Shapley value for an axiom (or an assertion) α in an knowledge base K is defined as:*

$$S_\alpha(K) = \frac{1}{n!} \sum_{\sigma \in \sigma_K} v(p_\sigma^\alpha \cup \{\alpha\}) - v(p_\sigma^\alpha)$$

The Shapley value can be directly computed from the possible coalitions without considering the permutations, with the following expression:

$$S_\alpha(K) = \sum_{C \subseteq K} \frac{(c-1)!(n-c)!}{n!} (v(C) - v(C \setminus \{\alpha\}))$$

where C is any coalition of the axioms and assertions K , $n = |K|$, and $c = |C|$.

3.3 Inconsistency Measure based on the Shapley Value

We can take the drastic inconsistency measure defined in Definition 2 (the same computation can be applied to the concept-related measure defined in Definition 3) as the characteristic function, and then use the Shapley value to compute to what extent an axiom or an assertion is concerned with the inconsistency.

For example, suppose K is a knowledge base and α is an axiom (or an assertion) in K , then the Shapley value of α based on the drastic inconsistency value I_d is defined as:

$$S_\alpha(K) = \sum_{C \subseteq K} \frac{(c-1)!(n-c)!}{n!} (I_d(C) - I_d(C \setminus \{\alpha\})) \quad (3)$$

where $n = |K|$ and $c = |C|$.

The Shapley value of a knowledge base K is a vector with each element denoting the Shapley value of each axiom (or assertion) in K .

Example 3. The Shapley value of the knowledge base $K = \{C \sqcup \neg C \sqsubseteq C \sqcap \neg C, \top \sqsubseteq \exists R.B, \top \sqsubseteq \forall R.\neg B, A \sqsubseteq D\}$ is $(\frac{4}{6}, \frac{1}{6}, \frac{1}{6}, 0)$.

This shows that $\{C \sqcup \neg C \sqsubseteq C \sqcap \neg C\}$ is the most problematic.

Example 4. The Shapley value of the working example is $(\frac{268}{7!}, \frac{250}{7!}, \frac{120}{7!}, \frac{120}{7!}, \frac{120}{7!}, \frac{3774}{7!}, \frac{268}{7!}, \frac{120}{7!}, 0)$.

This shows that axiom 6 is the one that causes the most problems. 3, 4, 5 and 8 are equally responsible for the inconsistencies, so are 1 and 7. Axiom 9 has a value of 0, which means it does not contribute to the inconsistency. Axiom 2 has a higher value than 3, 4 or 5, which shows that the inconsistency is more equally distributed among 3, 4 and 5.

3.4 Properties of the Inconsistency Measures

We make a few observations and remarks regarding the inconsistency measures.

Definition 5. *The characteristic function v is increasing if $X \subseteq Y$, $v(X) \leq v(Y)$.*

An increasing function indicates that adding more agents to the coalition will never decrease the value. In the worst case, they contribute nothing to the coalition. Due to the monotonic nature of DL reasoning, we can prove the following.

Proposition 1. *The drastic (concept-related) inconsistency value (see Definition 2 and 3) is increasing.*

A set of axioms and assertions is called *convergent* if its inconsistency value is convergent, i.e., it is the same as the inconsistency values of its super sets, and adding any other axioms or assertions does not change its inconsistency value.

Definition 6. *The convergent subset K' of a knowledge base K is defined as a set of axioms and assertions that satisfies the following two properties:*

1. $I_d(K') = 1$ (or $I_A(K') = 1$), and
2. $I_d(K'') = 0$ (or $I_A(K'') = 0$), for all $K'' \subset K'$.

Intuitively, K' is a convergent point, in which the inconsistency value flips from 0 to 1. There is a direct relation between the convergent subsets and minimally inconsistent subsets of a knowledge base, defined as follows.

Definition 7 (MIS). *We say that \mathcal{T}' is the minimally inconsistent subset (MIS) of a knowledge base \mathcal{T} if the following two conditions hold:*

1. \mathcal{T}' is inconsistent, and
2. \mathcal{T}'' is consistent, for every \mathcal{T}'' such that $\mathcal{T}'' \subset \mathcal{T}'$.

Proposition 2. *K' defined in Definition 6 is a minimally inconsistent subset.*

Another inconsistency measure of a coalition K' that can be defined is based on the number of minimally inconsistent subsets that would be removed if we remove K' from the knowledge base. In other words, this measures the impact of K' on the knowledge base, formalized as follows.

Definition 8. *The impact inconsistency measure of a subset K' in a knowledge base K can be defined as follows:*

$$I_i(K') = |MIS(K)| - |MIS(K - K')|$$

where $|MIS(K')|$ denotes the number of minimally inconsistent subsets of K' .

Example 5. There are four convergent subsets, i.e., four MISs in the working example.

$$\begin{aligned} MIS_1 &= \{1, 2, 7\}, MIS_2 = \{3, 4, 5, 8\}, MIS_3 = \{1, 3, 4, 5, 7\}, MIS_4 = \{6\} \\ I_i(\{1\}) &= I_i(\{7\}) = I_i(\{3\}) = I_i(\{4\}) = I_i(\{5\}) = 2 \\ I_i(\{2\}) &= I_i(\{6\}) = I_i(\{8\}) = 1 \\ I_i(\{9\}) &= 0 \end{aligned}$$

Obviously, the removal of 1, 7, 3, 4, or 5 will remove the most number of inconsistencies. The removal of axiom 9 will not affect the inconsistencies at all.

3.5 Apply the Inconsistency Measures to Clauses

The inconsistency measures discussed in this paper so far are applied to axioms in ontologies. It excludes the possibility of a more fine-grained inspection of the content of the axioms. In particular, if the inconsistency is in the level of a single axiom, then only two values can be obtained: consistent or inconsistent. In our previous work [1,12], we have developed a resolution based technique to explain inconsistency in ontologies. We found that clauses work on a more fine-grained level than DL axioms, so an approach based on clauses can identify specific parts of axioms that are responsible for an inconsistency. Consequently, the inconsistency measures can also be applied to clauses and this allows us to look inside the axiom and identify which proportion of the axiom is contributing to the inconsistency.

4 Computational Complexity Concerns

The most important source of computational complexity in calculating the Shapley value is the difficulty to obtain the inconsistency value of an axiom. It is directly dependent on the complexity of consistency checking in DL reasonings. One of the possible optimizations is to reduce the number of consistency checks. Besides, the complexity is also related to the computation process itself. The computation of the Shapley value considers all the subsets of the axioms/assertions in the knowledge base, and hence it results in Exp-time. However, we do not really need to compute the inconsistency values of all the subsets. In the following sections we will discuss such optimizations.

4.1 Partition based on Structural Relevance

In DLs, axioms³ can be related to each other through structure relevance. For example, the axiom $A \sqsubseteq B$ is structurally related to $\neg B \sqsubseteq \top$ but not to $\neg C \sqsubseteq D$. It is clear that adding a structurally unrelated axiom to a coalition will not change the relative inconsistency value. Structural relevance is an equivalence relation, hence it can be exploited to induce a partitioning of the axioms, which as shown below, can be used as an optimization to speed up the computation of the Shapley inconsistency value.

Definition 9. *We say an axiom is directly structurally related to another axiom if the intersection of their signature (the set of all (negated) concept names and role names occurring in the axiom) is not empty. The structural relevance is the transitive closure of direct structural relevance.*

Example 6. In the motivating example, $A1 \sqsubseteq A2 \sqcap \neg A \sqcap A3$ is directly structurally related to $A2 \sqsubseteq A \sqcap A4$. It is structurally related to $A4 \sqsubseteq C \sqcap \forall S.B$ (because $A3 \sqsubseteq A5 \sqcap A4$), but it is not related to $D \sqcup \neg D \sqsubseteq D \sqcap \neg D$.

³ For the sake of simplicity, we only refer to the axioms, assertions can be considered in the same way.

Example 7. There are two partitions in the motivating example: $\{A1 \sqsubseteq A2 \sqcap \neg A \sqcap A3, A2 \sqsubseteq A \sqcap A4, A3 \sqsubseteq A5 \sqcap A4, A4 \sqsubseteq C \sqcap \forall S.B, A5 \sqsubseteq \exists S.\neg B, A1(a), A3(b)\}$, $\{D \sqcup \neg D \sqsubseteq D \sqcap \neg D, A6 \equiv D\}$.

The following result suggests that partitioning according to structural relevance can be used to reduce the computational complexity of the Shapley value in our context.

Lemma 1. *If $K = \sum_{i=1}^T K_i$ is a partitioning of a knowledge base, and all K_i have the same inconsistency value function I , then for any axiom (or assertion) $\alpha \in K_i$,*

$$S_\alpha(K_i) = \sum_{C \subseteq K_i} \frac{(c-1)!(n-c)!}{n!} (I(C) - I(C \setminus \{\alpha\}))$$

where $n = |K_i|$ and $c = |C|$.

After the partitioning, the Shapley value of an axiom (or an assertion) can be computed in $O(\sum_{i=1}^T 2^{|K_i|})$.

The partitioning based on structural relevance preserves the total ordering on the Shapley values of the axioms (and assertions) inside the same partition. In other words, if an axiom has a higher Shapley value than another axiom in the partition, then it will also have a higher Shapley value in the knowledge base.

Theorem 1. *If $K = \sum_{i=1}^T K_i$ is a partitioning of a knowledge base, and all K_i have the same inconsistency value function I , then for any $\alpha, \beta \in K_i$, if $S_\alpha(K_i) > S_\beta(K_i)$, then $S_\alpha(K) > S_\beta(K)$.*

Proof. For each partition $K_i \subseteq K$ and $\alpha, \beta \in K_i$, if $S_\alpha(K_i) > S_\beta(K_i)$, then $\sum_{C \subseteq K_i} \frac{(c-1)!(n-c)!}{n!} (I(C) - I(C \setminus \{\alpha\})) > \sum_{C \subseteq K_i} \frac{(c-1)!(n-c)!}{n!} (I(C) - I(C \setminus \{\beta\}))$. Let us prove $S_\alpha(K) > S_\beta(K)$ by cases: for any $C_i \subseteq K$, if $C_i \subseteq C$, as previously indicated, $\sum_{C_i \subseteq K} \frac{(c-1)!(n-c)!}{n!} (I(C_i) - I(C_i \setminus \{\alpha\})) > \sum_{C_i \subseteq K} \frac{(c-1)!(n-c)!}{n!} (I(C_i) - I(C_i \setminus \{\beta\}))$. Otherwise, if $I(C_i - C) = 1$, then $I(C_i) - I(C_i \setminus \{\alpha\}) = I(C_i) - I(C_i \setminus \{\beta\})$, if $I(C_i - C) = 0$, then $I(C_i) - I(C_i \setminus \{\alpha\}) > I(C_i) - I(C_i \setminus \{\beta\})$. So $\sum_{C_i \subseteq K} \frac{(c-1)!(n-c)!}{n!} (I(C_i) - I(C_i \setminus \{\alpha\})) \geq \sum_{C_i \subseteq K} \frac{(c-1)!(n-c)!}{n!} (I(C_i) - I(C_i \setminus \{\beta\}))$. So $\sum_{C \subseteq K} \frac{(c-1)!(n-c)!}{n!} (I(C) - I(C \setminus \{\alpha\})) > \sum_{C \subseteq K} \frac{(c-1)!(n-c)!}{n!} (I(C) - I(C \setminus \{\beta\}))$, considering all the possible cases. Hence $S_\alpha(K) > S_\beta(K)$.

4.2 Optimization Based on Properties of the Inconsistency Measure

Partitioning the knowledge base according to structural relevance can work very well when the sizes $|K_i|$ are small, especially when the $|K_i|$ are bounded by a constant. This, however, is largely dependent on the particular knowledge base considered. In what follows, we propose an algorithm to calculate the Shapley value based on the properties of the inconsistency measure, especially the convergent property. This method aims to reduce the number of consistency checks.

The basic idea is quite simple: according to Definition 6 in Section 3.4, a convergent subset of a knowledge base is the maximal subset whose inconsistency value has to be calculated. Any superset of a convergent knowledge base can have its inconsistency value derived to be 1 due to the monotonicity of DLs. Hence once a subset is convergent, there is no necessity to compute its supersets. The detailed algorithm is shown in Figure 1.

Input: an axiom (or assertion) α in K
Output: the Shapley value of α

for all the subsets $K' \subseteq K$, sorted by the cardinality of K'
 while $I(K' \cup \alpha) = 0$
 move to the next unvisited K'
 for all supersets K'' of K'
 $I(K'') = 1$
 tag K'' as visited
 move to the next unvisited K'
Compute the Shapley value of α

Fig. 1. Computing Shapley values.

Example 8. To see how this algorithm works, we can apply the optimization to one of the partitions in the motivating example: $\{A1 \sqsubseteq A2 \sqcap \neg A \sqcap A3, A2 \sqsubseteq A \sqcap A4, A3 \sqsubseteq A5 \sqcap A4, A4 \sqsubseteq C \sqcap \forall S.B, A5 \sqsubseteq \exists S.\neg B, A1(a), A3(b)\}$. The algorithm will first compute the inconsistency value of $A1(a)$, and then the coalition of $A1(a)$ and any one of the other axioms, and then the coalition of $A1(a)$ and any two of the other axioms, since the coalition of $A1 \sqsubseteq A2 \sqcap \neg A \sqcap A3, A2 \sqsubseteq A \sqcap A4$ and $A1(a)$ has an inconsistency value of 1, we will skip computing the inconsistency value of its supersets. It is the same case with the coalition of $A1 \sqsubseteq A2 \sqcap \neg A \sqcap A3, A3 \sqsubseteq A5 \sqcap A4, A4 \sqsubseteq C \sqcap \forall S.B, A5 \sqsubseteq \exists S.\neg B$ and $A1(a)$.

We can either compute the convergent sub-terminologies on the fly during the computation of the Shapley value, or use algorithms for MIS in [13] to compute them in advance.

5 Experimental Results

To evaluate whether our proposed method is useful for ontologies, we have implemented the algorithm with the suggested optimizations in the previous section, and have performed some preliminary experiments. It was run on two ontologies: the University ontology with 32 concepts and 24 axioms, the Koala ontology with 14 concepts and 19 axioms.⁴ Tests were performed on a Windows XP system

⁴ The University and Koala ontologies were obtained from the website of SWOOP: <http://www.mindswap.org/2005/debugging/ontologies/>.

with a 3.0GHz Intel Pentium 4 processor, and 2GB memory. The implementation was developed in Java (JDK 1.5.0). The original University and Koala ontologies are consistent and they have 15 and 3 unsatisfiable concepts respectively. They are provided as sample ontologies to test the repair service of the OWL ontology editor SWOOP 2.3 beta 3, which uses Pellet as the default DL reasoner. In our experiments, we asserted fresh individuals for these unsatisfiable concepts in order to make the ontologies inconsistent. After the modification, repair plans can no longer be generated for these inconsistent ontologies in SWOOP.

There were four sources of inconsistencies in the University ontology. Calculating the measure took about 35 seconds. In the Koala ontology, there were three sources of inconsistencies, and it took 12 seconds to calculate the inconsistency measure. If optimization techniques discussed in Section 4 are adopted, then the running time are 1.6 seconds and 467 ms, respectively.

Let's look at part of the Koala ontology as follows:

1. $Koala \sqsubseteq Marsupials$
2. $Quokka \sqsubseteq Marsupials$
3. $Person \sqsubseteq \neg Marsupials$
4. $Koala \sqsubseteq \exists isHardworking\{false\}$
5. $KoalaWithPhD \sqsubseteq \exists hasDegree\{PhD\} \sqcap Koala$
6. $Quokka \sqsubseteq \exists isHardworking\{true\}$
7. $\exists isHardworking.\top \sqsubseteq Person$
8. $\exists hasDegree.\top \sqsubseteq Person$
9. $Koala(Suzy)$
10. $KoalaWithPhD(Lizzy)$
11. $Quokka(Pan)$

There are three sources of inconsistencies in this ontology. A koala named Suzy is forced to be a member of the disjoint classes marsupial and person. She is a marsupial because koala is a subclass of marsupial and she is a person because person is the domain of isHardWorking and every koala must have at least one isHardWorking property. A koala with a PhD named Lizzy is also forced to be a member of the disjoint classes marsupial and person. She is a person because person is the domain of hasDegree and every koala with a PhD must have at least one hasDegree property. Similarly, a Quokka named Pan is also causing inconsistency problems.

After the computation, axiom 3 gets the highest Shapley value, followed by axiom 7 and 1. In addition, axiom 3 has an impact inconsistency measure of 3. For the naive user of these ontologies, this information can be very helpful. Removing Axiom 3 will render the ontology consistent, and therefore enables most of the reasoning services that users have expected from their ontology development tools.

For the purpose of this paper, the implementation uses Racer as a black box to check the satisfiability. This black box approach comes with the advantage of independence of any particular reasoner or DL language. However, even with the optimization techniques, the worst case computational complexity is still exponential time. If the convergent ontologies are computed in a preprocessing

step using algorithms presented in [14], the computational time can be further reduced.

6 Related Work

Measures of inconsistency have been studied in [15, 16, 4]. These proposals for measuring inconsistency can be classified in two approaches. The first involves counting the minimal number of formulae needed to produce the inconsistency. The more formulae needed to produce the inconsistency, the less inconsistent the set [16]. The second approach involves inspecting the proportion of the language that is affected by the inconsistency. [4] is the closest to our work. Our work is inspired by their proposal to use the Shapley value to obtain an inconsistency measure for propositional logic. However, our work differs from theirs in two aspects: our approach works for a much more expressive logic and we also consider the optimizations of this method from a practical point of view.

In the DL community, several approaches have been proposed to deal with diagnosis of a knowledge base. [17] provides a general theory for diagnosis of Description Logics knowledge bases, based on Reiter’s diagnosis theory from first principles. [13] uses a modified tableau algorithm for \mathcal{ALC} to first find a minimally inconsistent subset of a knowledge base, and then use Reiter’s hitting set algorithm to find the maximally consistent subsets. They also propose to choose and eliminate axioms that most frequently participate in the underlying logical contradictions. [14] modifies the tableau rules and extends the DL language to \mathcal{SHOIN} . These approaches focus on the incoherence problem, i.e., if there exists an unsatisfiable concept in the ontology and they cannot render a repair plan if the ontology contains more than one inconsistency. Our approach also differs from these proposals in that they assume that all inconsistencies are equally bad, while we present a quantitative way to differentiate these inconsistencies.

7 Conclusion and Future Work

With the development of more expressive ontologies in the Semantic Web community, inconsistency has become an increasing problem that can seriously hamper the construction and application of web ontologies. In this paper, we have presented a technique for measuring inconsistencies which uses the Shapley value in game theory. Since the Shapley value aims to distribute the gains from cooperation in a fair manner, it can be used to impute the inconsistency to each axioms in the problematic ontology. The idea is to first define an inconsistency value, and then take it as the characteristic function, using the Shapley value to compute the contribution of each axiom or assertion to the inconsistencies in the ontology. This technique is conceptually flexible in the sense that although we focus on inconsistency problems in this paper, by choosing different inconsistency value functions, we can also address incoherence problems. It should be an easy extension to adopt other consistency values in the future. The measure

associated with an axiom shows the degree of its responsibility for the inconsistency, therefore it can give guidelines for repairing the ontologies. We have also proposed and implemented some optimization techniques based on the structural relevance of the axioms and properties of the defined inconsistency measure, in order to reduce the computational complexity. An implementation of the proposed algorithm to be applied to clauses is underway to be incorporated as part of our explanation module developed earlier for DL reasoning.

8 Acknowledgements

This work was supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada, by Genome Quebec, and by Faculty of ENCS, Concordia University. We also thank anonymous reviewers for their valuable comments.

References

1. Deng, X., Haarslev, V., Shiri, N.: A framework for explaining reasoning in description logics. In: Proceedings of the AAAI Fall Symposium on Explanation-aware Computing, Washington, DC, USA, AAAI Press (2005) 189–204
2. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging owl ontologies. In: Proceedings of the 14th International World Wide Web Conference (WWW 2005), Chiba, Japan, ACM Press (2005) 633–640
3. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proceedings of the eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03), Acapulco, Mexico, Morgan Kaufmann (2003) 355–362
4. Hunter, A., Konieczny, S.: Shapley inconsistency values. In: Proceedings of the International Conference on Knowledge Representation (KR'06), Windermere, UK, AAAI Press (2006) 249–259
5. Conitzer, V., Sandholm, T.: Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In McGuinness, D.L., Ferguson, G., eds.: Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence (AAAI 2004), San Jose, California, USA, AAAI Press/The MIT Press (2004) 219–225
6. In: OWL Web Ontology Language Overview. (2004) <http://www.w3.org/TR/owl-features/>.
7. Baader, F., Nutt, W.: Basic description logic. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003) 5–44
8. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006). Volume 4130 of Lecture Notes in Artificial Intelligence., Seattle, Washington, USA, Springer (2006) 292–297

9. Haarslev, V., Möller, R.: Racer system description. In R. Gori, A. Leitsch, T.N., ed.: Proceedings of International Joint Conference on Automated Reasoning (IJ-CAR 2001), Siena, Italy, Springer-Verlag (2001) 701–705
10. Sirin, E., Parsia, B.: Pellet: An owl dl reasoner. In: Proceedings of the 2004 International Workshop on Description Logics (DL2004), Whistler, British Columbia, Canada (2004)
11. Shapley, L.: A value for n-person games. In Kuhn, H., Tucker, A., eds.: Contributions to the Theory of Games. Volume 2. Princeton University Press (1953) 307–317
12. Deng, X., Haarslev, V., Shiri, N.: Resolution based explanations for reasoning in the description logic \mathcal{ALC} . In: Proceedings of the Canadian Semantic Web Working Symposium, Quebec City, Canada, Springer (2006) 55–61
13. Schlobach, S.: Diagnosing terminologies. In: Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI 2005), Pittsburgh, Pennsylvania, USA, AAAI Press (2005) 670–675
14. Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.C.: Repairing unsatisfiable concepts in owl ontologies. In Sure, Y., Domingue, J., eds.: Proceedings of the 3rd European Semantic Web Conference (ESWC 2006). Volume 4011 of Lecture Notes in Computer Science., Budva, Montenegro, Springer (2006) 170–184
15. Grant, J.: Classifications for inconsistent theories. Notre Dame Journal of Formal Logic **19**(3) (1978) 435–444
16. Knight, K.: Measuring inconsistency. Journal of Philosophical Logic **31**(2) (2002) 77–98
17. Friedrich, G., Shehekotikhin, K.M.: A general diagnosis method for ontologies. In Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A., eds.: Proceedings of 4th International Semantic Web Conference (ISWC 2005). Volume 3729 of Lecture Notes in Computer Science., Galway, Ireland, Springer (2005) 232–246