An Anytime Algorithm for Computing Inconsistency Measurement*

Yue Ma[1], Guilin Qi[2], Guohui Xiao[3], Pascal Hitzler[4], and Zuoquan Lin[3]

[1] Institute LIPN, Université Paris-Nord (LIPN - UMR 7030), France
[2] School of Computer Science and Engineering, Southeast University, Nanjing, China
[3] Department of Information Science, Peking University, China
[4] Kno.e.sis Center, Wright State University, Dayton, OH, USA
yue.ma@lipn.univ-paris13.fr, gqi@aifb.uni-karlsruhe.de,
{xgh,lz}@is.pku.edu.cn, pascal@pascal-hitzler.de

Abstract. Measuring inconsistency degrees of inconsistent knowledge bases is an important problem as it provides context information for facilitating inconsistency handling. Many methods have been proposed to solve this problem and a main class of them is based on some kind of paraconsistent semantics. In this paper, we consider the computational aspects of inconsistency degrees of propositional knowledge bases under 4-valued semantics. We first analyze its computational complexity. As it turns out that computing the exact inconsistency degree is intractable, we then propose an anytime algorithm that provides tractable approximation of the inconsistency degree from above and below. We show that our algorithm satisfies some desirable properties and give experimental results of our implementation of the algorithm.

1 Introduction

Inconsistency handling is one of the central problems in the field of knowledge representation. Recently, there is an increasing interest in quantifying inconsistency in an inconsistent knowledge base. This is because it is not fine-grained enough to simply say that two inconsistent knowledge bases contain the same amount of inconsistency. Indeed, it has been shown that analyzing inconsistency is helpful to decide how to act on inconsistency [1], i.e. whether to ignore it or to resolve it. Furthermore, measuring inconsistency in a knowledge base can provide some context information which can be used to resolve inconsistency [2,3,4], and proves useful in different scenarios such as Software Engineering [5].

Different approaches to measuring inconsistency are based on different views of atomic inconsistency [3]. Syntactic ones put atomicity to formulae, such as taking maximal consistent subsets of formulae [6] or minimal inconsistent sets [7]. Semantic ones put atomicity to propositional letters, such as considering the conflicting propositional letters based on some kind of paraconsistent model [8,2,3,9,10]. In this paper, we focus on the computational aspect of a 4-valued semantics based inconsistency degree which is among the latter view.

* We acknowledge support by OSEO, agence nationale de valorisation de la recherche in the Quaero project.

D. Karagiannis and Z. Jin (Eds.): KSEM 2009, LNAI 5914, pp. 29–40, 2009.

The main contributions of this paper lie in Section 4, 5, and 6 with new proposed interesting theorems (their proofs are omitted due to space limitation). In Section 4, we show that computing exact inconsistency degrees is a computational problem of high complexity (Θ_2^p-complete). In Section 5, we present an anytime algorithm to provide tractable approximations of the inconsistency degree from above and below, by computing *the lower* and *upper bounds* defined in this paper. We show that our algorithm satisfies some desirable properties. Section 6 will give experimental explanations of the algorithm. To the best of our knowledge, this is the first work that (1) analyzes the complexity issues of computing the inconsistency degree and that (2) attempts to alleviate the intractability of computing the exact inconsistency degree for full propositional logic by approximating it from above and from below in an anytime manner. Our results show that the computation of approximating inconsistency degree can be done tractable; and can be performed to full propositional knowledge bases instead of restricting to CNF to design a tractable paraconsistent reasoning [11].

2 Related Work

Most effort has been directed at theoretical accounts of inconsistency measures, i.e. its definitions, properties, and possible applications. But few papers focus on the computational aspect of inconsistency degree. Among the syntactic approaches, [6] shows the possibility to compute inconsistency degrees using the simplex method. Among the semantics methods, [12] and [10] provide algorithms for computing inconsistency degrees that can be implemented. The algorithm in [10] only deals with KBs consisting of first-order formulas in the form $Q_1x_1, ..., Q_nx_n.\bigwedge_i(P_i(t_1, ..., t_{m_i}) \wedge \neg P_i(t_1, ..., t_{m_i}))$, where $Q_1, ..., Q_n$ are universal or existential quantifiers. In [12], an algorithm is proposed for full FOL logic. Although it can be applied to measure inconsistency in propositional logic, its computational complexity is too high to be used in in general cases. The anytime algorithm proposed in this paper for computing approximating inconsistency degrees can avoid these shortcomings.

Although our algorithm is inspired by the algorithm in [12], It is significantly different from the existing one. Firstly, ours is motivated by the theoretical results of the tractability of *S-4 entailment* (Theorems 3,4,5). In contrast, the algorithm in [12] is based on a reduction to hard SAT instances, which makes it inherently intractable. Secondly, ours is designed towards obtaining an approximation with guaranteed lower and upper bounds that gradually converge to the exact solution. Thirdly, we implement a new strategy to achieve polynomial time approximations. We present the preliminary evaluation results of the implementation of the algorithm in Section 6. Our evaluation results show our algorithm outperforms that given in [12] and the approximating values are reasonable to replace the exact inconsistency degree.

3 Preliminaries

Let $\mathcal{P}$ be a countable set of propositional letters. We concentrate on the classical propositional language formed by the usual Boolean connectives $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), and $\neg$ (negation). A propositional knowledge base K consists of

a finite set of formulae over that language. We use $Var(K)$ for the set of propositional letters used in K and $|S|$ for the cardinality of S for any set S.

Next we give a brief introduction on Belnap's four-valued (4-valued) semantics. Compared to two truth values used by classical semantics, the set of truth values for four-valued semantics [13,14] contains four elements: *true, false, unknown* and *both*, written by t, f, N, B, respectively. The truth value B stands for contradictory information, hence four-valued logic leads itself to dealing with inconsistencies. The four truth values together with the ordering $\preceq$ defined below form a lattice, denoted by $\mathsf{FOUR} = (\{t, f, B, N\}, \preceq)$: $f \preceq N \preceq t, f \preceq B \preceq t, N \not\preceq B, B \not\preceq N$. The four-valued semantics of connectives $\vee, \wedge$ are defined according to the upper and lower bounds of two elements based on the ordering $\preceq$, respectively, and the operator $\neg$ is defined as $\neg t = f, \neg f = t, \neg B = B$, and $\neg N = N$. The designated set of FOUR is $\{t, B\}$. So a four-valued interpretation $\mathfrak{I}$ is a *4-model* of a knowledge base K if and only if for each formula $\phi \in K, \phi^{\mathfrak{I}} \in \{t, B\}$. A knowledge base which has a 4-model is called *4-valued satisfiable*. A knowledge base K 4-valued entails a formula φ, written $K \models_4 \varphi$, if and only if each 4-model of K is a 4-model of φ. We write K for a knowledge base, and $\mathcal{M}_4(K)$ for the set of 4-models of K throughout this paper. Four-valued semantics provides a novel way to define inconsistentcy measurements [1].

Let $\mathfrak{I}$ be a four-valued model of K. The *inconsistency degree of K with respect to* $\mathfrak{I}$, denoted $\mathrm{Inc}_{\mathfrak{I}}(K)$, is a value in $[0, 1]$ defined as $\mathrm{Inc}_{\mathfrak{I}}(K) = \frac{|Conflict(\mathfrak{I}, K)|}{|Var(K)|}$, where $Conflict(\mathfrak{I}, K) = \{p \mid p \in Var(K), p^{\mathfrak{I}} = B\}$. It measures to what extent a given knowledge base K contains inconsistencies with respect to its 4-model $\mathfrak{I}$. Preferred models defined below are used to define inconsistency degrees and especially useful to explain our approximating algorithm later.

Definition 1 (Preferred Models). *The set of* preferred models, *written $PreferModel(K)$, is defined as $PreferModel(K) = \{\mathfrak{I} \mid \forall \mathfrak{I}' \in \mathcal{M}_4(K), Inc_{\mathfrak{I}}(K) \leq Inc_{\mathfrak{I}'}(K)\}$.*

Definition 2 (Inconsistency Degree). *The* inconsistency degree of K, *denoted by $ID(K)$, is defined as the value $Inc_{\mathfrak{I}}(K)$, where $\mathfrak{I} \in PreferModels(K)$.*

Example 1. *Let $K = \{p, \neg p \vee q, \neg q, r\}$. Consider two 4-valued models $\mathfrak{I}_1$ and $\mathfrak{I}_2$ of K with $p^{\mathfrak{I}_1} = t, q^{\mathfrak{I}_1} = B, r^{\mathfrak{I}_1} = t$; and $p^{\mathfrak{I}_2} = B, q^{\mathfrak{I}_2} = B, r^{\mathfrak{I}_2} = t$. We have $Inc_{\mathfrak{I}_1}(K) = \frac{1}{3}$, while $Inc_{\mathfrak{I}_2}(K) = \frac{2}{3}$. Moreover, $\mathfrak{I}_1$ is a preferred model of K because there is no other 4-model $\mathfrak{I}'$ of K such that $Inc_{\mathfrak{I}'}(K) < Inc_{\mathfrak{I}_1}(K)$. Then $ID(K) = \frac{1}{3}$.*

One way to compute inconsistency degree is to recast the algorithm proposed in [12] to propositional knowledge bases, where *S-4 semantics* defined as follows is used:

Definition 3 (S-4 Model). *For any given set $S \subseteq Var(K)$, an interpretation $\mathfrak{I}$ is called an S-4 model of K if and only if $\mathfrak{I} \in \mathcal{M}_4(K)$ and satisfies the following condition:*

$$\mathfrak{I}(p) \in \begin{cases} \{B\} & \text{if } p \in Var(K) \setminus S, \\ \{N, t, f\} & \text{if } p \in S. \end{cases}$$

For a given $S \subseteq Var(K)$, the knowledge base K is called *S-4 unsatisfiable* iff. it has no S-4 model. Let φ be a formula and $Var(\{\varphi\}) \subseteq Var(K)$. φ is *S-4 entailed* by K, written $K \models_S^4 \varphi$, iff. each S-4 model of K is an S-4 model of φ.

Theorem 1 ([12]). *For any KB K, we have $ID(K) = 1 - A/|Var(K)|$, where $A = \max\{|S| : S \subseteq Var(K), K \text{ is } S\text{-4 satisfiable}\}$.*

Theorem 1 shows that the computation of $ID(K)$ can be reduced to the problem of computing the maximal cardinality of subsets S of $Var(K)$ such that K is S-4 satisfiable.

4 Computational Complexities

Apart from any particular algorithm, let us study the computational complexity of the inconsistency degree to see how hard the problem itself is. First we define following computation problems related inconsistency degrees:

- $\text{ID}_{\leq d}$ (resp. $\text{ID}_{<d}$, $\text{ID}_{\geq d}$,$\text{ID}_{>d}$): Given a propositional knowledge base K and a number $d \in [0, 1]$, is $ID(K) \leq d$ (resp. $ID(K) < d$, $ID(K) \geq d$, $ID(K) > d$)?
- EXACT-ID: Given a propositional knowledge base K and a number $d \in [0, 1]$, is $ID(K) = d$?
- ID: Given a propositional knowledge base K, what is the value of $ID(K)$?

We have the complexities of these problems indicated by following theorem.

Theorem 2. *$ID_{\leq d}$ and $ID_{<d}$ are* **NP***-complete; $ID_{\geq d}$ and $ID_{>d}$ are* **coNP***-complete; EXACT-ID is* **DP***-complete; ID is $\mathbf{FP}^{\mathbf{NP}[\log \mathbf{n}]}$-complete*[1].

5 Anytime Algorithm

According to results shown in the previous section, computing inconsistency degrees is usually intractable. In this section, we propose an anytime algorithm to approximate the exact inconsistency degree. Our results show that in P-time we can get an interval containing the accurate value of $ID(K)$. We first clarify some definitions which will be used to explain our algorithm.

5.1 Formal Definitions

Definition 4. *(Bounding Values) A real number x (resp. y) is a* lower (resp. an upper) bounding value *of the inconsistency degree of K, if and only if $x \leq ID(K)$ (resp. $ID(K) \leq y$).*

Intuitively, a pair of lower and upper bounding values characterizes an interval containing the exact inconsistency degree of a knowledge base. For simplicity, lower (resp. an upper) bounding value is called *lower (resp. upper) bound*.

[1] A language L is in the class **DP**[15] iff there are two languages $L_1 \in \mathbf{NP}$ and $L_2 \in \mathbf{coNP}$ such that $L = L_1 \cap L_2$. Complexity $\mathbf{P}^{\mathbf{NP}[\log \mathbf{n}]}$ is defined to be the class of all languages decided by a polynomial-time oracle machine which on input x asks a total of $\mathcal{O}(log\,|x|)$ SAT (or any other problem in $\mathbf{NP}$) queries. $\mathbf{FP}^{\mathbf{NP}[\log \mathbf{n}]}$ is the corresponding class of functions.

Definition 5. *(Bounding Models) A four-valued interpretation $\mathfrak{I}'$ is a* lower *(resp. an* upper*) bounding model of K if and only if for any preferred model $\mathfrak{I}$ of K, Condition 1 holds (resp. Condition 2 holds and $\mathfrak{I}' \in \mathcal{M}_4(K)$):*

$$\textit{Condition 1: } |\textit{Conflict}(\mathfrak{I}', K)| \leq |\textit{Conflict}(\mathfrak{I}, K)|$$
$$\textit{Condition 2: } |\textit{Conflict}(\mathfrak{I}', K)| \geq |\textit{Conflict}(\mathfrak{I}, K)|$$

Intuitively, the lower and upper bounding models of K are approximations of preferred models from below and above. We call two-valued interpretations $\mathfrak{I}$ trivial lower bounding models since $\textit{Conflict}(\mathfrak{I}, K) = 0$ and $\textit{ID}(K) = 0$ always holds. We are only interested in nontrivial bounding models for inconsistent knowledge bases, which can produce a nonzero lower bound of $\textit{ID}(K)$.

Example 2. *(Example 1 continued) K has a lower bounding model $\mathfrak{I}_3$ and an upper bounding model $\mathfrak{I}_4$ defined as: $p^{\mathfrak{I}_3} = t, q^{\mathfrak{I}_3} = t, r^{\mathfrak{I}_3} = t$; and $p^{\mathfrak{I}_4} = B, q^{\mathfrak{I}_4} = B, r^{\mathfrak{I}_4} = t$.*

Next proposition gives a connection between *lower (resp. upper) bounds* and *lower (resp. upper) bounding models*.

Proposition 1. *If $\mathfrak{I}$ is a lower (an upper) bounding model of K, $\textit{Inc}_{\mathfrak{I}}(K)$ is a lower (an upper) bounding value of $\textit{ID}(K)$.*

By borrowing the idea of guidelines for a theory of approximating reasoning [16], we require that an anytime approximating algorithm for computing inconsistency degrees should be able to produce two sequences $r_1, ..., r_m$ and $r^1, ..., r^k$:

$$r_1 \leq ... \leq r_m \leq \textit{ID}(K) \leq r^k \leq ... \leq r^1, \tag{1}$$

such that these two sequences have the following properties:

- The length of each sequence is polynomial w.r.t $|K|$;
- Computing r_1 and r^1 are both tractable. Generally, computing r_j and r^j becomes exponentially harder as j increases, but it is not harder than computing $\textit{ID}(K)$.
- Since computing r_i and r^j could become intractable as i and j increase, we need to find functions $f(|K|)$ and $g(|K|)$ such that computing r_i and r^j both stay tractable as long as $i \leq f(|K|)$ and $j \leq g(|K|)$.
- each r_i (r^j) corresponds to a lower (an upper) bounding model, which indicates the sense of the two sequences.

In the rest, we will describe an anytime algorithm which can produce such two sequences.

5.2 Tractable Approximations from Above and Below

We know that S-4 entailment is generally intractable, which makes algorithms based on S-4 semantics to compute inconsistency degrees time-consuming. In this section, we will distinguish a tractable case of S-4 entailment (proportional to the size of input knowledge base), by which we can compute approximating inconsistency degrees.

Lemma 1. *Let $S = \{p_1, ..., p_k\}$ be a subset of $Var(K)$ and φ be a formula such that Var(φ) $\subseteq$ Var(K). $K \models_S^4 \varphi$ if and only if*

$$K \wedge \bigwedge_{q \in Var(K) \setminus S} (q \wedge \neg q) \models_4 \varphi \vee (c_1 \vee ... \vee c_k)$$

holds for any combination $\{c_1, ..., c_k\}$, where each c_i is either p_i or $\neg p_i (1 \leq i \leq k)$.

This lemma shows a way to reduce the S-4 entailment to the 4-entailment. Specially note that if φ is in CNF (conjunctive normal formal), the righthand of the reduced 4-entailment maintains CNF form by a little bit of rewriting, as follows: Suppose $\varphi = C_1 \wedge ... \wedge C_n$. Then $\varphi \vee (c_1 \vee ... \vee c_k) = (C_1 \vee c_1 \vee ... \vee c_k) \wedge ... \wedge (C_n \vee c_1 \vee ... \vee c_k)$ which is still in CNF and its size is linear to that of $\varphi \vee (c_1 \vee ... \vee c_k)$.

Lemma 2 ([17]). *For K in any form and φ in CNF, there exists an algorithm for deciding if $K \models_4 \varphi$ in $\mathcal{O}(|K| \cdot |\varphi|)$ time.*

By Lemma 1 and 2, we have the following theorem:

Theorem 3 (Complexity). *There exists an algorithm for deciding if $K \models_S^4 \varphi$ and deciding if K is S-4 satisfiable in $\mathcal{O}(|K||\varphi||S| \cdot 2^{|S|})$ and $\mathcal{O}(|K||S| \cdot 2^{|S|})$ time, respectively.*

Theorem 3 shows that S-4 entailment and S-4 satisfiability can both be decided in polynomial time w.r.t the size of K, exponential w.r.t that of S, though. So they can be justified in P-time if $|S|$ is limited by a logarithmic function of $|K|$.

Next we study how to use Theorem 3 to tractably compute upper and lower bounding values of inconsistency degrees.

Lemma 3. *Given two sets S and S' satisfying $S \subseteq S' \subseteq \mathcal{P}$, if a theory K is S-4 unsatisfiable, then it is S'-4 unsatisfiable.*

By Lemma 3, we get a way to compute upper and lower bounds of $ID(K)$ shown by Theorems 4 and 5, respectively.

Theorem 4. *Given $S \subseteq Var(K)$, if K is S-4 satisfiable, then $ID(K) \leq 1 - |S|/|Var(K)|$.*

Theorems 3 and 4 together show that for a monotonic sequence of sets S_1,...,S_k, where $|S_i| < |S_{i+1}|$ for any $1 \leq i \leq k-1$, if we can show that K is S_i-4 ($i = 1, ..., k$) satisfiable one by one, then we can get a sequence of decreasing upper bounding values of the inconsistency degree of K in time $\mathcal{O}(|K||S_i| \cdot 2^{|S_i|})$. If $|S_i| = \mathcal{O}(\log |K|)$, it is easy to see that the computation of an upper bound is done in polynomial time with respect to the size of K. In the worst case (i.e., when $S = Var(K)$), the complexity of the method coincides with the result that $\mathrm{ID}_{\leq}$ is NP-complete (Theorem 2).

Theorem 5. *Given a w ($1 \leq w \leq |Var(K)|$), if for each w-size subset S of $Var(K)$, K is S-4 unsatisfiable, then $ID(K) \geq 1 - (w-1)/|Var(K)|$.*

Theorems 3 and 5 together show that for a monotonic sequence of sets $S_1, ..., S_m$ satisfying $|S_i| < |S_{i+1}|$, if we can prove that K is $|S_i|$-4 unsatisfiable[2] for each $i \in [1, m]$,

[2] For the sake of simplicity, we say that K is l-4 satisfiable for $l \in \mathbb{N}$, if there is a subset $S \subseteq Var(K)$ such that K is S-4 satisfiable. We say that K is l-4 unsatisfiable if K is not l-4 satisfiable.

then we can get a series of increasing lower bounds of the inconsistency degree of K. For each w, it needs at most $\binom{|Var(K)|}{w}$ times tests of S-4 unsatisfiability. So it takes $\mathcal{O}(\binom{|Var(K)|}{w}|K|w\cdot 2^w)$ time to compute a lower bound $1-(w-1)/|Var(K)|$. If and only if w is limited by a constant, we have that each lower bound is obtained in polynomial time by Proposition 2.

Proposition 2. *Let $f(n)=\mathcal{O}(\binom{n}{k}\cdot 2^k)$ where $0\leq k\leq n$. There exists a $p\in\mathbb{N}$ such that $f(n)=\mathcal{O}(n^p)$ if and only if k is limited by a constant which is independent of n.*

Suppose r_i, r^j in Inequation 1 are defined as follows:

$$r^j = 1-|S|/|Var(K)|, \text{ where } K \text{ is } |S|\text{-4 satisfiable, } j=|S|;$$

$$r_i = 1-\frac{|S|-1}{|Var(K)|}, \text{ where } K \text{ is } |S|\text{-4 unsatisfiable, } i=|S|.$$

By Theorems 3, 4 and 5 and Proposition 2, we get a way to compute the upper and lower bounds of $ID(K)$ which satisfy: if $j\leq \log(|K|)$ and $i\leq M$ (M is a constant independent of $|K|$), r^j and r_i are computed in polynomial time; Both i and j cannot be greater than $|Var(K)|$. This is a typical approximation process of a **NP**-complete problem $\text{ID}_{\geq d}$ (resp. co**NP**-complete problem $\text{ID}_{\leq d}$) via polynomial intermediate steps, because each intermediate step provides a partial solution which is an upper (resp. lower) bound of $ID(K)$.

Example 3. *Suppose $K=\{p_i\vee q_j, \neg p_i, \neg q_j \mid 1\leq i,j\leq N\}$. So $|Var(K)|=2N$. To know whether $ID(K)<\frac{3}{4}$, by Theorem 4 we only need to find an S of size $\lceil\frac{2N}{4}\rceil$ such that K is S-4 satisfiable. This is true by choosing $S=\{p_i\mid 1\leq i\leq\lceil\frac{2N}{4}\rceil\}$. To know whether $ID(K)>\frac{1}{3}$, Theorem 5 tells us to check whether K is S-4 unsatisfiable for all S of size $\lfloor\frac{4N}{3}\rfloor+1$. This is true also. So $ID(K)\in[\frac{1}{3},\frac{3}{4}]$.*

An interesting consequence of the above theoretical results is that we can compute the exact inconsistency of some knowledge bases in P-time. Let us first look at an example.

Example 4. *Let $K=\{(p_i\vee p_{i+1})\wedge(\neg p_i\vee\neg p_{i+1}), p_{i_1}\wedge\ldots\wedge p_{i_{N-5}}, \neg p_{j_1}\wedge\ldots\wedge\neg p_{j_{N-10}}, p_{2t}, \neg p_{3j+1}\vee\neg p_{5u+2},\}(1\leq i\leq N-1, 1\leq 2t, 3j+1, 5u+2\leq N)$. $Var(K)=N$. To approximate $ID(K)$, we can check whether K is l-4 satisfiable for l going larger from 1 by one increase on the value each time. Obviously, K's inconsistency degree is close to 1 if $N\gg 10$. By Theorem 3, we can see that all of these operations can be done in P-time before the exact value obtained.*

More formally, we have the following proposition.

Proposition 3. *If $ID(K)\geq 1-M/|Var(K)|$, where M is an arbitrary constant which is independent of $|K|$, then $ID(K)$ can be computed in polynomial time.*

Given a knowledge base K with $|Var(K)|=n$. By the analysis given after Theorem 4 and Theorem 5, we know that in the worst case, it takes $\mathcal{O}(\binom{n}{w}|K|w2^w)$ time to get

an upper (resp. a lower) bounding value. By Fermat's Lemma[3], its maximal value is near $w = \lceil \frac{2n+1}{3} \rceil$ when n is big enough. It means that *to do dichotomy directly on size $\frac{n}{2}$ will be of high complexity.* To get upper and lower bounding values in P-time instead of going to intractable computation straight, we propose Algorithm 1, which consists of two stages: The first one is to localize an interval $[l_1, l_2]$ that contains the inconsistency degree (line 1-8), while returning upper and lower bounding values in P-time; The second one is to pursue more accurate approximations within the interval $[l_1, l_2]$ by binary search (line 9-17).

Algorithm 1 is an anytime algorithm that can be interrupted at any time and returns a pair of upper and lower bounding values of the exact inconsistency degree. It has five parameters: the knowledge base K we are interested in; the precision threshold ε which is used to control the precision of the returned results; the constant $M \ll |Var(K)|$ to guarantee that the computation begins with tractable approximations; a pair of positive reals a, b which determines a linear function $h(l_2) = al_2 + b$ that updates the interval's right extreme point l_2 by $h(l_2)$ during the first stage (line 5). $h(\cdot)$ decides how to choose the sizes l to test l-4 satisfiability of K. For example, if $h(l_2) = l_2 + 2$, line 5 updates l from i to $i + 1$ (suitable for $ID(K)$ near 1); If $h(l_2) = 2l_2$, line 5 updates l from i to $2i$ (suitable for $ID(K)$ near 0.5); While if $h(l_2) = 2(|Var(K)| - M)$, line 5 updates l by $|Var(K)| - M$ (suitable for $ID(K)$ near 0). We remark that $h(l_2)$ can be replaced by other functions.

Next we give detailed explanations about Algorithm 1. To guarantee that it runs in P-time run at the beginning to return approximations, we begin with a far smaller search interval $[l_1, l_2] = [0, M]$ compared to $|Var(K)|$. The *while* block (line 3) iteratively tests whether the difference between upper and lower bounding values is still lager than the precision threshold and whether K is l-satisfiable, where $l = \lceil \frac{l_2}{2} \rceil$. If both yes, the upper bound r_+ is updated, the testing interval becomes $[l, h(l_2)]$, and the iteration continues; Otherwise (line 7), the lower bound r_- is updated and the search interval becomes $[l_1, l]$. This completes the first part of the algorithm to localize an interval. If $r_+ - r_-$ is already below the precision threshold, the algorithm terminates (line 8). Otherwise, we get an interval $[l_1, l_2]$ such that K is l_1-4 satisfiable and l_2-4 unsatisfiable. Then the algorithm turns to the second "while" iteration (line 9) which executes binary search within the search internal $[l_1, l_2]$ found in the first stage. If there is a subset $|S| = l_1 + \lceil \frac{l_2 - l_1}{2} \rceil$ such that K is S-4 satisfiable, then the search internal shortens to the right half part of $[l_1, l_2]$ (line 12), otherwise to the left half part (line 14). *During this stage, K keeps l_2-4 unsatisfiable and l_1-4 satisfiable for $[l_1, l_2]$.* Until $r_+ - r_-$ below the precision threshold, the algorithm finishes and returns upper and lower bounds.

Theorem 6 (Correctness of Algorithm 1). *Let r_+ and r_- be values computed by Algorithm 1. We have $r_- \leq ID(K)$ and $r_+ \geq ID(K)$. Moreover, $r_+ = r_- = ID(K)$ if $\varepsilon = 0$.*

The following example gives a detailed illustration.

Example 5. *(Example 3 contd.) Let $\varepsilon = 0.1, h(l_2) = 2l_2$, and $M = 4 \ll N$. Algorithm 1 processes on K as follows:*

[3] V.A. Zorich, *Mathematical Analysis*, Springer, 2004.

Algorithm 1. Approx_Incons_Degree$(K, \varepsilon, M, a, b)$

Input: KB K; precision threshold $\varepsilon \in [0, 1[$; constant $M \ll |Var(K)|$; $a, b \in \mathbb{R}^+$
Output: Lower bound r_- and upper bound r_+ of $ID(K)$

1: $r_- \leftarrow 0; r_+ \leftarrow 1$ {Initial lower and upper bounds}
2: $\overline{\varepsilon} \leftarrow r_+ - r_-; n \leftarrow |Var(K)|; l_1 \leftarrow 0; l_2 \leftarrow M; l \leftarrow \lceil \frac{l_2}{2} \rceil$
3: **while** $\overline{\varepsilon} > \varepsilon$ and K is l-4 satisfiable **do**
4: $r_+ \leftarrow (1 - l/n); \overline{\varepsilon} \leftarrow r_+ - r_-$ {Update upper bound}
5: $l_1 \leftarrow l; l_2 \leftarrow h(l_2); l \leftarrow \lceil \frac{l_2}{2} \rceil$ {Update search interval}
6: **end while**
7: $r_- \leftarrow 1 - (l-1)/n; \overline{\varepsilon} \leftarrow r_+ - r_-; l_2 \leftarrow l$
8: **if** $\overline{\varepsilon} \leq \varepsilon$ **then** **return** r_+ and r_- **end if**
9: **while** $\overline{\varepsilon} > \varepsilon$ **do**
10: $l \leftarrow l_1 + \lceil \frac{l_2 - l_1}{2} \rceil$
11: **if** K is l-4 satisfiable **then**
12: $r_+ \leftarrow (1 - l/n); \overline{\varepsilon} \leftarrow r_+ - r_-; l_1 \leftarrow l$
13: **else**
14: $r_- \leftarrow 1 - (l-1)/n; \overline{\varepsilon} \leftarrow r_+ - r_-; l_2 \leftarrow l$
15: **end if**
16: **end while**
17: **return** r_+ and r_-

Denote the initial search interval $[l_1^0, l_2^0] = [0, 4]$. After initializations, $l = 2$ and line 3 is executed. Obviously, K is S-4 satisfiable for some $|S| = l$ (e.g. $S = \{p_1, p_2\}$). So we get a newer upper bound $r_+ = \frac{2N-l}{2N}$. Meanwhile, the difference between upper and lower bounds $\overline{\varepsilon}$ becomes $\frac{2N-l}{2N} > \varepsilon$, and the search interval is updated as $[l_1^1, l_2^1] = [l, 2l_2]$ and $l = 4$.

Stage 1. *The* while *iteration from line 3 is repeatedly executed with double size increase of l each time. After c times such that $2^{c-1} \leq N < 2^c$, $l = 2^c$ and K becomes l-4 unsatisfiable. The localized interval is $[2^{c-1}, 2^c]$. It turns to line 7 to update the lower bound by $1 - \frac{l-1}{2N}$. The newest upper bound is $1 - 2^{c-2}/N$, so $\overline{\varepsilon} = 2^{c-2}/N$. If $\overline{\varepsilon} \leq \varepsilon$, algorithm ends by line 8. Otherwise, it turns to stage 2.*

Stage 2. *By dichotomy in the interval $[2^{c-1}, 2^c]$, algorithm terminates until $\overline{\varepsilon} \leq \varepsilon$.*

Unlike Example 5, for the knowledge base in Example 4, since its inconsistency degree is quite close to 1, it becomes S-4 satisfiable for an S such that $|S|$ is less than a constant M. Therefore, after the first stage of Algorithm 1 applying on this knowledge base, the localized interval $[l_1, l_2]$ is bounded by M. For such an interval, the second stage of the algorithm runs in P-time according to Theorem 3 and Proposition 2. So Algorithm 1 is a P-time algorithm for the knowledge base given in Example 4. However, it fails for other knowledge bases whose inconsistency degrees are far less than 1. Fortunately, the following proposition shows that by setting the precision threshold ε properly, Algorithm 1 can be executed in P-time to return approximating values.

Proposition 4. *Let s be an arbitrary constant independent of $|K|$. If $\varepsilon \geq 1 - \frac{h^s(M)}{2|Var(K)|}$, where $h^s(\cdot)$ is s iterations of $h(\cdot)$, Algorithm 1 terminates in polynomial time with the difference between upper and lower bounds less than ε ($r_+ - r_- \leq \varepsilon$).*

The following proposition shows that r_- and r_+ computed by Algorithm 1 have a sound semantics in terms of *upper* and *lower bounding models* defined in Definition 5.

Proposition 5. *There is a lower (an upper) bounding model $\mathfrak{I}'$ ($\mathfrak{I}''$) of K such that $Inc_{\mathfrak{I}'}(K) = r_-$ ($Inc_{\mathfrak{I}''}(K) = r_+$).*

Summing up, we have achieved an anytime algorithm for approximately computing inconsistency degrees which is:

- *computationally tractable:* Each approximating step can be done in polynomial time if $|S|$ is limited by a logarithmic function for upper bounds (Theorems 3 and 4) and by a constant function for lower bounds (Theorems 3 and 5).
- *dual and semantical well-founded:* The accurate inconsistency degree is approximated both from above and from below (Theorem 6), corresponding to inconsistency degrees of some upper and lower bounding models of *K* (Proposition 5).
- *convergent:* More computation resource available, more precise values returned (Theorems 4 and 5). It always converges to the accurate value if there is no limitation of computation resource (Theorem 6) and terminates in polynomial time for special knowledge bases (Proposition 3).

6 Evaluation

Our algorithm has been implemented in Java using a computer with Intel E7300 2.66G, 4G, Windows Server 2008. Algorithm 1 gives a general framework to approximate inconsistency degrees from above and below. In our implementation, we set $M = 2, h(l_2) = l_2 + 2$. That is, the first *while* loop (see line 3) keeps testing l-4 satisfiability of K from $l = i$ to $i + 1$. So the interval $[l_1, l_2]$ localized in the first stage of the algorithm satisfies $l_2 = l_1 + 1$ and the second binary search is not necessary. According to our analysis in Section 5, this avoids direct binary search which needs to test all $\frac{n!}{(n/2)!(n/2)!}$ subsets of $Var(K)$, where $n = |Var(K)|$.

There are tow main sources of complexity to compute approximating inconsistency degrees: *the complexities of S-4 satisfiability* and *of search space.* The S-4 satisfiability that we implemented is based on the reduction given in Lemma 1 and the tractable algorithm for 4-satisfiability in [17]. Our experiments told us that search space could heavily affect efficiency. So we carefully designed a truncation strategy to limit the search space based on the monotonicity of S-4 unsatisfiability. That is, if we have found an S such that K is S-4 unsatisfiable, then we can prune all supersets S' of S which makes K S'-4 unsatisfiable. We implemented this strategy in breadth-first search on the binomial tree [18,19] of subsets of $Var(K)$.

Figure 1 shows the evaluation results over knowledge bases[4] in Example 3 with $|K| = N^2 + 2N$ and $|Var(k)| = 2N$ for $N = 5, 7, 8, 9, 10$. The left part of the figure shows how the preset precision threshold ε affects the run time performance of our algorithm: the smaller ε is, the longer it executes. If $\varepsilon \geq 0.7$, the algorithm terminated

[4] We use instances of Example 3 because they are the running examples through the paper and meet the worst cases of the algorithm (e.g. the *truncation strategy* discussed later cannot be applied). We want to show the performance of our algorithm in its worst case.

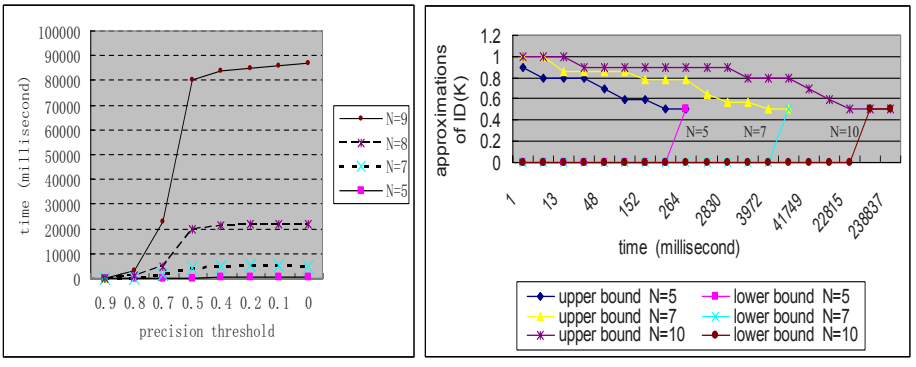

Fig. 1. Evaluation results over KBs in Example 3 with $|K| = N^2 + 2N$ and $|Var(K)| = 2N$ for $N = 5, 7, 8, 9, 10$

easily (at most $18.028s$ for $N = 9$ and much less for $N < 9$). The quality of the approximations at different time points is shown on the right part of the figure. The decreasing (resp. increasing) curves represent upper (resp. lower) bounds for $N = 5, 7, 10$, respectively. Note that the inconsistency degrees of all the three knowledge bases are 0.5.

For large knowledge bases, it is time-consuming to compute the exact inconsistency degrees. For example, for $N = 10$, our algorithm took 239.935s to get the accurate inconsistency degree. In contrast, by costing much less time, approximating values (upper bounds for these examples) can provide a good estimation of the exact value and are much easier to compute. For example, when $N = 10$, the algorithm told us that the inconsistency degree is less than 0.8 at 3.9s; and when $N = 5$, we got the upper bound 0.6 at 0.152s. Note that in these experiments, the lower bounds were updated slowly. In fact, the exact inconsistency degrees were obtained as soon as the first nonzero lower bounding values were returned. This is because we set $M = 2, h(l_2) = l_2 + 2$ in our implementation. If we set M and $h(\cdot)$ differently, the results will be changed, as shown in Example 3 in Section 5.

We need to point out that our truncation strategy cannot be applied to the test data used in the experiments because no subsets can be pruned. Therefore, although our experiments show the benefits of the approximations, our algorithm can increase significantly when the truncation strategy is applicable and if we carefully set M and $h(\cdot)$. Take $\{p_i, \neg p_j \mid 0 \leq i, j < 20, j \text{ is odd}\}$ for example, our optimized algorithm run less than 1s whilst it run over 5min without the truncation strategy.

7 Conclusion

In this paper, we investigated computational aspects of the inconsistency degree. We showed that the complexities of several decision problems about inconsistency degree are high in general. To compute inconsistency degrees more practically, we proposed an general framework of an anytime algorithm which is *computationally tractable, dual*

and semantical well-founded, and improvable and convergent. The experimental results of our implementation show that computing approximating inconsistency degrees is much faster than computing the exact inconsistency degrees in general. The approximating inconsistency degrees can be useful in many applications, such as knowledge base evaluation and merging inconsistent knowledge bases. We will further study on the real applications of approximating inconsistency degree in the future work.

References

1. Hunter, A.: How to act on inconsistent news: Ignore, resolve, or reject. Data Knowl. Eng. 57, 221–239 (2006)
2. Hunter, A.: Measuring inconsistency in knowledge via quasi-classical models. In: Proc. of AAAI 2002, pp. 68–73. AAAI Press, Menlo Park (2002)
3. Hunter, A., Konieczny, S.: Approaches to measuring inconsistent information. In: Bertossi, L., Hunter, A., Schaub, T. (eds.) Inconsistency Tolerance. LNCS, vol. 3300, pp. 191–236. Springer, Heidelberg (2005)
4. Hunter, A., Konieczny, S.: Shapley inconsistency values. In: Proc. of KR 2006, pp. 249–259. AAAI Press, Menlo Park (2006)
5. Mu, K., Jin, Z., Lu, R., Liu, W.: Measuring inconsistency in requirements specifications. In: Godo, L. (ed.) ECSQARU 2005. LNCS (LNAI), vol. 3571, pp. 440–451. Springer, Heidelberg (2005)
6. Knight, K.: Measuring inconsistency. Journal of Philosophical Logic 31(1), 77–98 (2002)
7. Hunter, A., Konieczny, S.: Measuring inconsistency through minimal inconsistent sets. In: Proc. of KR 2008, pp. 358–366 (2008)
8. Grant, J.: Classifications for inconsistent theories. Notre Dame J. of Formal Logic 19, 435–444 (1978)
9. Grant, J., Hunter, A.: Measuring inconsistency in knowledgebases. Journal of Intelligent Information Systems 27, 159–184 (2006)
10. Grant, J., Hunter, A.: Analysing inconsistent first-order knowledge bases. Artif. Intell. 172, 1064–1093 (2008)
11. Coste-Marquis, S., Marquis, P.: A unit resolution-based approach to tractable and paraconsistent reasoning. In: Proc. of ECAI, pp. 803–807 (2004)
12. Ma, Y., Qi, G., Hitzler, P., Lin, Z.: An algorithm for computing inconsistency measurement by paraconsistent semantics. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 91–102. Springer, Heidelberg (2007)
13. Belnap, N.D.: A useful four-valued logic. In: Modern uses of multiple-valued logics, pp. 7–73. Reidel Publishing Company, Boston (1977)
14. Arieli, O., Avron, A.: The value of the four values. Artif. Intell. 102, 97–141 (1998)
15. Papadimitriou, C.H. (ed.): Computational Complexity. Addison Wesley, Reading (1994)
16. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artificial Intelligence 74, 249–310 (1995)
17. Cadoli, M., Schaerf, M.: On the complexity of entailment in propositional multivalued logics. Ann. Math. Artif. Intell. 18, 29–50 (1996)
18. Malouf, R.: Maximal consistent subsets. Computational Linguistics 33, 153–160 (2007)
19. Mu, K., Jin, Z., Liu, W., Zowghi, D.: An approach to measuring the significance of inconsistency in viewpoints framework. Technical report, Peking University (2008)