Algorithms for Inconsistency Measurement

Isabelle Kuhlmann

University of Koblenz-Landau, Universitätsstraße 1, 56070 Koblenz, Germany
`iskuhlmann@uni-koblenz.de`

1 Introduction

A significant challenge in the area of artificial intelligence is the management of inconsistent information. Inconsistencies occur whenever a piece of information contradicts another one or is in conflict with it. However, such conflicts cannot realistically be avoided when dealing with formal knowledge representation systems, because they are often based on information acquired from different sources.

The field of *inconsistency measurement* deals with the quantitative assessment of the severity of inconsistencies regarding logical knowledge bases. An inconsistency measure is a function that maps a knowledge base of logical formulas to a non-negative real number. The larger this number, the more severe is the inconsistency.

2 Motivation

Formally, a knowledge base $\mathcal{K}$ is a finite set of formulas $\mathcal{K} \subseteq \mathcal{L}(\mathsf{At})$. $\mathcal{L}(\mathsf{At})$ is the propositional language constructed with respect to a fixed set of atoms At and the usual connectives $\wedge$ (conjunction), $\vee$ (disjunction), and $\neg$ (negation).

The area of inconsistency measurement [4] generally offers an analytical perspective on the subject of inconsistency in knowledge representation formalisms. A very general definition of a inconsistency measure is given in the following.

Definition 1. *An* inconsistency measure $\mathcal{I}$ *is any function* $\mathcal{I} : \mathbb{K} \to \mathbb{R}^{\infty}_{\geq 0}$.

The intuition behind this is that a larger value indicates a more severe inconsistency. For most inconsistency measures, $\mathcal{I}(\mathcal{K}) = 0$ implies the absence of inconsistency, i. e., consistency.

Moreover, the problem of deciding whether a given value $a \in \mathbb{R}^{\infty}_{\geq 0}$ is the inconsistency value $\mathcal{I}(\mathcal{K})$ with regard to a knowledge base $\mathcal{K}$ is denoted as $\textsc{Exact}_{\mathcal{I}}$. In a similar fashion, $\textsc{Upper}_{\mathcal{I}}$ and $\textsc{Lower}_{\mathcal{I}}$ denote the problems whether a given value is an upper, or lower bound of $\mathcal{I}(\mathcal{K})$, respectively. $\textsc{Value}_{\mathcal{I}}$ is the natural function problem which returns the inconsistency value $\mathcal{I}(\mathcal{K})$ for a given knowledge base $\mathcal{K}$.

As inconsistencies cannot be avoided in real-world applications, it is quite crucial to be able to handle them. Inconsistencies can be dealt with in several ways, for example, by deleting formulas that are involved in an inconsistency, or by splitting a formula into its conjuncts in order to isolate the most problematic ones [5]. However, to facilitate this, inconsistencies first have to be identified and analyzed. An important tool for this is inconsistency measurement.

Moreover, inconsistency measurement helps humans who model information as formal representations to identify issues and also to compare alternative formalizations. Consider, as an example, the following two knowledge bases $\mathcal{K}_1$ and $\mathcal{K}_2$ which represent different weather forecasts.

$$\mathcal{K}_1 = \{\text{sunny}, \neg\text{sunny}\,\text{hot}, \neg\text{hot}\} \qquad \mathcal{K}_2 = \{\text{sunny}, \neg\text{sunny}, \text{hot}, \neg\text{rainy}\}$$

Both knowledge bases are classically inconsistent, as there is no interpretation that could satisfy any of them. In $\mathcal{K}_1$, all formulas are directly involved in conflicts, since, as one can easily see, the formulas sunny and ¬sunny, and hot and ¬hot contradict each other, respectively. In $\mathcal{K}_1$, on the other hand, only sunny and ¬sunny contradict each other. Thus, $\mathcal{K}_2$ contains fewer conflicts, and additionally includes some formulas that are not involved in any inconsistency (i. e., the formulas hot and ¬rainy). Consequently, $\mathcal{K}_2$ should be considered more useful, and should also be assigned a lower inconsistency value than $\mathcal{K}_1$.

There are further aspects that can be taken into account when assessing the degree of inconsistency with respect to a knowledge base. For instance, the number of formulas that are necessary to produce a contradiction could be considered as well. For example, the inconsistent knowledge base

$$\mathcal{K}_3 = \{\text{sunny}, \neg\text{hot}, \text{sunny} \rightarrow \text{hot}, \neg\text{rainy}, \neg\text{cloudy}\}$$

includes a conflict that is induced by the three formulas sunny, ¬hot, and sunny $\rightarrow$ hot. Because more formulas are involved in this conflict, i. e., it is less direct than the conflict in $\mathcal{K}_2$, $\mathcal{K}_3$ might be considered less inconsistent.

3 Research Question

Although a lot of research is concerned with the development of new inconsistency measures, or the evaluation of different measures against each other, the focus of my thesis is of algorithmic nature. As the computation of inconsistency measures is in general computationally hard, the development of efficient algorithms is of vital importance. Particularly those measures that belong to the first level of the polynomial hierarchy [11] are most suitable to consider, as they are most likely to be applicable to real-world problems. Some examples of such measures are the contension inconsistency measure [5], the η-inconsistency measure [6], the hitting set inconsistency measure [12], and the MCS-based inconsistency measure [1].

4 Related Work

As mentioned before, most research in the field of inconsistency measurement focuses on the development and comparison of different measures. Nonetheless, there exist some works that perform algorithmic studies of inconsistency measurements.

Thimm [12] introduces approximation algorithms for the contension inconsistency measure and the hitting set inconsistency measure. The approximation is based on the principle of evolutionary algorithms. An evaluation regarding runtime and accuracy is performed, however, there is no comparison to other approaches.

McAreavey et al. [10] focus their work on those measures that are based on the enumeration of minimal inconsistent subsets. A connection between minimal inconsistent subsets and minimal unsatisfiable sets of clauses, a notion which originates from the SAT community, is established. The authors further present a tool where two different approaches to finding minimal inconsistent subsets are implemented, one of which utilizes minimal unsatisfiable subsets. An extensive evaluation of the tool on random arbitrary knowledge bases is conducted as well.

Another approach is presented by Ma et al. [9]. Here, an anytime algorithm for the contension inconsistency measure is developed. The algorithm approximates inconsistency values from above and below. The authors also perform an experimental evaluation which shows that approximating an inconsistency value is faster than computing it exactly.

5 Approach

The overall objective of the thesis is to develop practically feasible algorithms for a number of existing inconsistency measures. A strategy to accomplish this is to utilize another problem-solving paradigm. More specifically, one could encode an inconsistency measure as a satisfiability (SAT) [2] problem and make use of existing SAT solvers. Another approach is to encode inconsistency measures in answer set programming (ASP) [3, 8]. More specifically, the problem $\textsc{Upper}_{\mathcal{I}}$ with respect to a certain inconsistency measure can be encoded as a SAT problem or an answer set program. It is then possible to determine $\textsc{Value}_{\mathcal{I}}$ by iteratively sending calls to a SAT, or ASP solver. By using binary search on the search space of possible inconsistency values, only logarithmic many calls are required. In [7], we already successfully applied the ASP approach to the contension inconsistency measure [5].

Yet another idea to develop more efficient algorithms for inconsistency measures is to utilize approximation techniques. Further, heuristic methods, such as local search or evolutionary algorithms, could be examined.

6 Evaluation

The core component of the evaluation is a comparison between the developed algorithms and, if possible, existing algorithms. For instance, in [7] we compared an existing brute-force algorithm for computing the contension inconsistency measure with a newly developed algorithm which uses reductions to answer set programming. We created a set of 800 knowledge bases, and defined a timeout of 60 seconds. Then we used both algorithms to measure the contension inconsistency of each knowledge base and took the respective runtimes. The results may be visualized as displayed in Fig. 6.

Another task that is to be considered regarding the evaluation is the compilation of an suitable dataset. So far, to the best of my knowledge, a standard dataset dedicated to the evaluation of inconsistency measures does not exist.

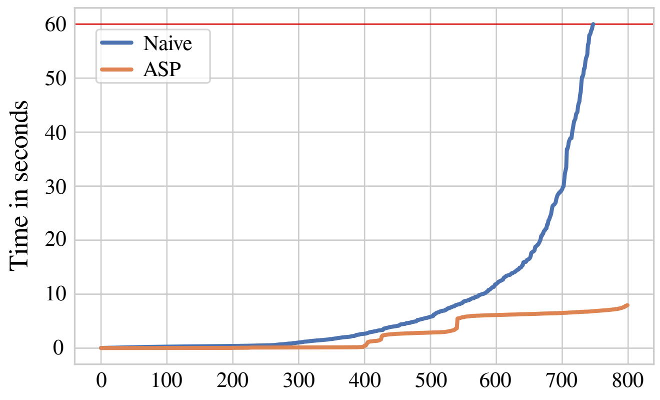

Fig. 1. Comparison of a naive (brute-force) implementation of the contension inconsistency measure and an implementation of the ASP-based algorithm proposed in [7]. The horizontal red line indicates a timeout of 60 seconds.

References

1. Ammoura, M., Salhi, Y., Oukacha, B., Raddaoui, B.: On an mcs-based inconsistency measure. International Journal of Approximate Reasoning **80**, 443–459 (2017)
2. Biere, A., Heule, M., van Maaren, H.: Handbook of satisfiability, vol. 185. IOS press (2009)
3. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New generation computing **9**(3-4), 365–385 (1991)
4. Grant, J.: Classifications for inconsistent theories. Notre Dame Journal of Formal Logic **19**(3), 435–444 (1978)
5. Grant, J., Hunter, A.: Measuring consistency gain and information loss in stepwise inconsistency resolution. In: Proceedings ECSQARU'11. pp. 362–373. Springer (2011)
6. Knight, K.: Measuring inconsistency. Journal of Philosophical Logic **31**(1), 77–98 (2002)
7. Kuhlmann, I., Thimm, M.: An algorithm for the contension inconsistency measure using reductions to answer set programming. In: 14th International Conference on Scalable Uncertainty Management (2020)
8. Lifschitz, V.: What is answer set programming? In: Proceedings AAAI'08. pp. 1594–1597 (2008)

9. Ma, Y., Qi, G., Xiao, G., Hitzler, P., Lin, Z.: An anytime algorithm for computing inconsistency measurement. In: International Conference on Knowledge Science, Engineering and Management. pp. 29–40. Springer (2009)
10. McAreavey, K., Liu, W., Miller, P.: Computational approaches to finding and measuring inconsistency in arbitrary knowledge bases. International Journal of Approximate Reasoning **55**(8), 1659–1693 (2014)
11. Thimm, M., Wallner, J.: On the complexity of inconsistency measurement. Artificial Intelligence **275** (2019)
12. Thimm, M.: Stream-based inconsistency measurement. International Journal of Approximate Reasoning **68**, 68–87 (2016)