Measuring Inconsistency in Answer Set Programs

Markus Ulbricht[1], Matthias Thimm[1,2], and Gerhard Brewka[1]

[1] Department of Computer Science, Leipzig University, Germany
[2] Institute for Web Science and Technologies (WeST), University of Koblenz-Landau, Germany

Abstract. We address the issue of quantitatively assessing the severity of inconsistencies in logic programs under the answer set semantics. While measuring inconsistency in classical logics has been investigated for some time now, taking the non-monotonicity of answer set semantics into account brings new challenges that have to be addressed by reasonable accounts of inconsistency measures. We investigate the behavior of inconsistency in logic programs by revisiting existing rationality postulates for inconsistency measurement and developing novel ones taking non-monotonicity into account. Further, we develop new measures for this setting and investigate their properties.

1 Introduction

Answer set programming (ASP, see [2] for an overview) is a popular non-monotonic formalism for knowledge representation and reasoning. We consider a finite set $\mathcal{L}$ of literals. An extended logic program P (over $\mathcal{L}$) is a set of rules of the form

$$r : \quad l_0 \leftarrow l_1, \ldots, l_k, \text{not } l_{k+1}, \ldots, \text{not } l_m. \tag{1}$$

with $l_0, \ldots, l_m \in \mathcal{L}$, $0 \leq k \leq m$. Let $\mathcal{P}$ be the set of all extended logic programs. We abbreviate $head(r) = l_0$, $pos(r) = \{l_1, \ldots, l_k\}$ and $neg(r) = \{l_{k+1}, \ldots, l_m\}$. For two sets M and L of literals, we say M satisfies L ($M \vDash L$) iff $l \in M$ for each $l \in L$. Now let P be a classical program (without default negation not). For a rule $r \in P$, $M \vDash r$ iff $M \vDash \{head(r)\}$ whenever $M \vDash pos(r)$ and $M \vDash P$ iff $M \vDash r$ for each rule $r \in P$. We let $\mathsf{Cl}(P)$ be the unique $M \subseteq \mathcal{L}$ with $M \vDash P$ and $M' \nvDash P$ for each set $M' \subsetneq M$.

Definition 1. *A set M of literals is called an* answer set *of a classical program P if $M = \mathsf{Cl}(P)$. M is an answer set of an extended logic program P if M is the answer set of P^M, where $P^M = \{head(r) \leftarrow pos(r) \mid r \in P,\ neg(r) \cap M = \emptyset\}$ is the* reduct *of P with respect to M.*

A set M of literals is called *consistent* if it does not contain both a and $\neg a$ for an atom a. A program P is called *consistent* if it has at least one consistent answer set, otherwise it is called *inconsistent*. Let $\text{Ans}(P)$ denote the set of all answer sets of P and $\text{Ans}_{Inc}(P)$ and $\text{Ans}_{Con}(P)$ the inconsistent and consistent ones, respectively. Note that, motivated by the goals of this paper, our definition slightly differs from the original definition in [3] which allows for a single inconsistent answer set only, namely $\mathcal{L}$.

In the classical literature on inconsistency measurement—see e. g. [5, 4, 10]—inconsistency measures are functions that aim at assessing the severity of the inconsistency in knowledge bases formalized in propositional logic. Here, we are interested

in measuring inconsistency for (extended) logic programs and only consider measures defined on those. Let $\mathbb{R}^{\infty}_{\geq 0}$ be the set of non-negative real values including ∞.

Definition 2. *An* inconsistency measure $\mathcal{I}$ *is a function* $\mathcal{I} : \mathcal{P} \to \mathbb{R}^{\infty}_{\geq 0}$.

The basic intuition behind an inconsistency measure $\mathcal{I}$ is that the larger the inconsistency in P the larger the value $\mathcal{I}(P)$. However, even in the setting of propositional logic, inconsistency is a concept that is not easily quantified and there have been a couple of proposals for inconsistency measures in this setting, see [10] for a recent survey.

The issue of measuring inconsistency in logic programs is more challenging compared to the classical setting due to the non-monotonicity of answer set semantics. This becomes apparent when considering the *monotonicity* postulate which is usually satisfied by classical inconsistency measures and demands $\mathcal{I}(P') \geq \mathcal{I}(P)$ whenever $P \subseteq P'$, i. e., the severity of inconsistency cannot be decreased by adding new information. Consider now the two logic programs P_1 and P_2 given as follows:

$$P_1 : \; b \leftarrow \text{not } a. \qquad \neg b \leftarrow \text{not } a. \qquad\qquad P_2 : \; b \leftarrow \text{not } a. \qquad \neg b \leftarrow \text{not } a. \qquad a.$$

We have $P_1 \subseteq P_2$ but P_1 is inconsistent while P_2 is not, so we would expect $\mathcal{I}(P_2) < \mathcal{I}(P_1)$ for any reasonable measure $\mathcal{I}$. Therefore, simply taking classical inconsistency measures and applying them to the setting of logic programs does not yield the desired behavior.

Many rationality postulates such as *monotonicity* from above are already disputed in the classical setting, cf. [1]. Taking non-monotonicity of the knowledge representation formalism into account, a rational account of the severity of inconsistency calls for a specific investigation, which we will undertake in the remainder of this paper. In particular, we will discuss rationality postulates for inconsistency measures in logic programs in Section 2 and propose some novel measures in Section 3. An extended version of this paper can be found online[3].

2 Rationality Postulates

Research in inconsistency measurement is driven by *rationality postulates*, i. e., desirable properties that should hold for concrete approaches. There is a growing number of rationality postulates for inconsistency measurement but not every postulate is generally accepted, see [1] for a recent discussion on this topic. In the following, we revisit a selection of the most popular postulates—see e. g. [6, 9]—and phrase them within our context of logic programs. To do so, we need some further notation.

Definition 3. *The* dependency graph D_P *of a program* P *is a labeled directed graph having all literals of the program as vertices and there is an edge* (l_i, l_j, s) *iff* P *contains a rule* r *such that* $head(r) = l_j$ *and* $l_i \in pos(r) \cup neg(r)$. *The label* $s \in \{+, -\}$ *indicates whether* $l_i \in pos(r)$ *or* $l_i \in neg(r)$. *For any literal* l, *let* $\mathsf{Path}(P, l)$ *be the set of all literals* l' *(including* l *itself) such that there is a path from* l *to* l' *in* D_P.

[3] `http://www.mthimm.de/misc/utb_incasp.pdf`

Definition 4. *A set U of literals is called a* splitting set *[7] for P, if $head(r) \in U$ implies that all literals of atoms appearing in r are contained in U, for every rule $r \in P$. For a splitting set U, let $bot_U(P)$ be the set of all rules $r \in P$ with $head(r) \in U$. This set of rules is called the* bottom part *of P with respect to U.*

Definition 5. *A rule $r^* \in P$ is called* safe *with respect to P if the atom occurring in the head of r^* does not appear elsewhere in the program and $pos(r^*) \cup neg(r^*)$ is a subset of the literals occurring in $P \setminus \{r^*\}$.*

Now let $\mathcal{I}$ be an inconsistency measure. The postulate *Consistency* establishes that 0 is the minimal inconsistency value and that it is reserved for consistent programs.

Consistency P is consistent iff $\mathcal{I}(P) = 0$.

Satisfaction of *Monotonicity* is generally *not* desirable for ASP. However, as we still wish to require some form of monotonicity in special cases, we consider the weaker postulate *CLP-Monotonicity* (CLP stands for "classical logic program"). If a program does not contain any default negation and we only add new information without default negation, we are in the classical setting and monotonicity should hold. A stronger version of *CLP-Monotonicity* is *I-Monotonicity* which is applicable when the head of a new rule is independent of the defaults in the program. Similarly, *Split-Monotonicity* considers monotonicity with respect to the bottom part of splitting sets.

Monotonicity $\mathcal{I}(P) \leq \mathcal{I}(P')$ whenever $P \subseteq P'$.
CLP-Monotonicity If P is a classical logic program and r^* a classical rule, then $\mathcal{I}(P) \leq \mathcal{I}(P \cup \{r^*\})$.
I-Monotonicity If r^* is a rule with $\mathsf{Path}(P \cup r^*, head(r^*)) \cap neg(P \cup r^*) = \emptyset$, then $\mathcal{I}(P) \leq \mathcal{I}(P \cup \{r^*\})$.
Split-Monotonicity If U is a splitting set of P, then $\mathcal{I}(bot_U(P)) \leq \mathcal{I}(P)$.

Finally, *Safe-rule independence* demands that the addition of safe rules does not change the inconsistency value.

Safe-rule independence If P is a logic program and r^* safe with respect to P, then $\mathcal{I}(P) = \mathcal{I}(P \cup \{r^*\})$.

3 Inconsistency Measures

We now propose concrete inconsistency measures for logic programs. Inconsistency of programs can occur due to two different reasons, namely because the program has no answer set at all or because all answer sets are inconsistent, cf. [8]. Different measures should assess those reasons differently. Furthermore, to measure inconsistency of a program, one could either take the program itself or the answer sets into account. We will cover both approaches.

Our first measure $\mathcal{I}_\pm$ aims at measuring the distance of the program to a consistent one. More specifically, it quantifies the number of modifications in terms of deleting and adding rules, necessary in order to restore consistency. Deleting certain rules can surely be sufficient to prevent P from entailing contradictions, but as already pointed out before, adding rules can also resolve inconsistency.

Definition 6. *Define* $\mathcal{I}_\pm : \mathcal{P} \to \mathbb{R}^\infty_{\geq 0}$ *via*

$$\mathcal{I}_\pm(P) = \min\{|A| + |D| \mid A, D \in \mathcal{P} \text{ such that } (P \cup A) \setminus D \text{ is consistent}\}$$

for all $P \in \mathcal{P}$.

Example 1. Consider the program P_3 defined via

$$\begin{array}{lccc} P_3: & a_1 \leftarrow \text{not } b. & a_1 \leftarrow \text{not } c. & a_1 \leftarrow \text{not } d. \\ & \neg a_1 \leftarrow \text{not } b. & \neg a_1 \leftarrow \text{not } c. & \neg a_1 \leftarrow \text{not } d. \end{array}$$

and P_4 given as follows.

$$\begin{array}{lccc} P_4: & a_1 \leftarrow \text{not } b. & a_2 \leftarrow \text{not } b. & a_3 \leftarrow \text{not } b. \\ & \neg a_1 \leftarrow \text{not } b. & \neg a_2 \leftarrow \text{not } b. & \neg a_3 \leftarrow \text{not } b. \end{array}$$

Note that P_3 contains three contradicting pairs of rules. Since one can delete one rule in each of them (or make the rule inapplicable by adding the corresponding fact), $\mathcal{I}_\pm(P_3) = 3$. Even though P_4 is similar, $\mathcal{I}_\pm(P_4) = 1$ since $P_4 \cup \{b.\}$ is consistent.

The measure $\mathcal{I}_\pm$ performs a *hypothetical* modification of the original program P itself to obtain consistency. Another approach is to relax the definition of answer sets and consider modifications of the reduct P^M instead.

Definition 7. *A consistent set* M *of literals is called a* k*-*l*-model of a classical logic program* P *if* M *is a model of* $(P \cup A) \setminus D$ *with* $A, D \in \mathcal{P}$ *and* $|A| \leq k$, $|D| \leq l$. *M is called a* k*-*l*-answer set of an extended logic program* P *if M is a* k*-*l*-model of* P^M.

Definition 8. *Define* $\mathcal{I}^\pm : \mathcal{P} \to \mathbb{R}^\infty_{\geq 0}$ *via*

$$\mathcal{I}^\pm(P) = \min_{M \subseteq \mathcal{L}} \{k + l \mid M \text{ is a } k\text{-}l\text{-answer set of } P\}$$

for all $P \in \mathcal{P}$.

Interestingly, however, these two different points of view—considering the reduct or the program itself—are equivalent.

Proposition 1. *For any extended logic program* P, $\mathcal{I}_\pm(P) = \mathcal{I}^\pm(P)$.

While for any program P, one can find a set M of literals such that M is a model of P^M, one cannot always guarantee M being the minimal model of the reduct. Our next measure minimizes the distance between M and $\mathsf{Cl}(P^M)$. We only consider the number of literals in the *symmetric difference* of two sets. Investigating other distances is left for future work. Recall that the symmetric difference d_{sd} of two sets M and M' is defined via $d_{sd}(M, M') = |(M \cup M') \setminus (M \cap M')|$.

Definition 9. *Define* $\mathcal{I}_{sd} : \mathcal{P} \to \mathbb{R}^\infty_{\geq 0}$ *via*

$$\mathcal{I}_{sd}(P) = \min_{M \in \mathsf{ConCl}_P} d_{sd}(M, \mathsf{Cl}(P^M))$$

with $\mathsf{ConCl}_P = \{M \subseteq \mathcal{L} \mid M, \mathsf{Cl}(P^M) \text{ is consistent}\}$ *and* $\min \emptyset = \infty$.

Table 1. Compliance of inconsistency measures with respect to our rationality postulates

	$\mathcal{I}_\pm = \mathcal{I}^\pm$	$\mathcal{I}_{sd}$	$\mathcal{I}_\#$
Consistency	✓	✓	✓
Monotonicity	✗	✗	✗
CLP-Monotonicity	✓	✓	✓
I-Monotonicity	✓	✓	✓
Split-Monotonicity	✓	✓	✓
Safe-rule independence	✓	✓	✓

Example 2. If a program P contains two contradicting facts, $\mathcal{I}_{sd}(P) = \infty$ since in this case, $\mathsf{Cl}(P^M)$ is inconsistent for any set M of literals. For the programs P_3 and P_4 from Example 1, we have $\mathcal{I}_{sd}(P_3) = 3$ and $\mathcal{I}_{sd}(P_4) = 1$.

Our last measure $\mathcal{I}_\#$ takes the answer sets of a program into account rather than the rules. For this purpose, we need the following notion.

Definition 10. *A set M of literals is called* k-inconsistent*, $k \in \mathbb{N} \cup \{0\}$, if there are exactly k atoms a such that $a \in M$ and $\neg a \in M$.*

Furthermore, programs might have no answer set at all, which is a special case for $\mathcal{I}_\#$.

Definition 11. *Define $\mathcal{I}_\# : \mathcal{P} \to \mathbb{R}^\infty_{\geq 0}$ via*

$$\mathcal{I}_\#(P) = \min_{M \in \mathit{Ans}(P)} \{k \mid M \text{ is } k\text{-inconsistent}\}$$

with $\min \emptyset = \infty$.

Example 3. For $\mathcal{I}_\#$, we obtain $\mathcal{I}_\#(P_3) = 1$ and $\mathcal{I}_\#(P_4) = 3$.

Table 1 gives an overview on the compliance of our measures with respect to the rationality postulates from Section 2. Note that, naturally, none of our measures satisfies the classical *monotonicity* postulate which is also not desired for ASP.

4 Summary

In this paper, we addressed the challenge of measuring inconsistency in ASP by critically reviewing the classical framework of inconsistency measurement and taking nonmonotonicity into account. We developed novel rationality postulates and measures that are more apt for analyzing inconsistency in ASP than classical approaches. Intuitively, some of our measures take the effort needed to restore the consistency of programs into account ($\mathcal{I}_\pm$, $\mathcal{I}^\pm$), and our results show that it does not matter whether this is done on the level of the original program or on the level of the reduct. Others measure inconsistency in terms of the quality of the produced output, e. g., $\mathcal{I}_\#$ which considers the minimal number of inconsistencies in an answer set.

Acknowledgements
This work has been partially funded by the DFG Research Training Group 1763.

References

1. Besnard, P.: Revisiting postulates for inconsistency measures. In: Proceedings of the 14th European Conference on Logics in Artificial Intelligence (JELIA'14). pp. 383–396 (2014)
2. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM 54(12), 92–103 (2011), http://doi.acm.org/10.1145/2043174.2043195
3. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. 9(3/4), 365–386 (1991), http://dx.doi.org/10.1007/BF03037169
4. Grant, J., Hunter, A.: Measuring Inconsistency in Knowledgebases. Journal of Intelligent Information Systems 27, 159–184 (2006)
5. Hunter, A., Konieczny, S.: Approaches to Measuring Inconsistent Information. In: Inconsistency Tolerance, Lecture Notes in Computer Science, vol. 3300, pp. 189–234. Springer International Publishing (2004)
6. Hunter, A., Konieczny, S.: On the measure of conflicts: Shapley inconsistency values. Artificial Intelligence 174(14), 1007–1026 (July 2010)
7. Lifschitz, V., Turner, H.: Splitting a logic program. In: Logic Programming, Proceedings of the Eleventh International Conference on Logic Programming, Santa Marherita Ligure, Italy, June 13-18, 1994. pp. 23–37 (1994)
8. Schulz, C., Satoh, K., Toni, F.: Characterising and explaining inconsistency in logic programs. In: Calimeri, F., Ianni, G., Truszczynski, M. (eds.) Logic Programming and Nonmonotonic Reasoning: 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings, pp. 467–479. Springer International Publishing, Cham (2015)
9. Thimm, M.: Inconsistency Measures for Probabilistic Logics. Artificial Intelligence 197, 1–24 (2013)
10. Thimm, M.: On the expressivity of inconsistency measures. Artificial Intelligence 234, 120–151 (2016)