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

Inconsistency is an omnipresent phenomenon in logical accounts of knowledge representation and reasoning (KR) [8, 9]. Classical logics usually suffer from the *principle of explosion* which renders reasoning meaningless, as everything can be derived from inconsistent theories. Therefore, reasoning under inconsistency [2, 22, 24] is an important research area in KR. In general, one can distinguish two paradigms in handling inconsistent information. The first paradigm advocates to live with inconsistency but to provide non-classical semantics that allows to derive non-trivial information, such as using paraconsistent reasoning [4], reasoning with possibilistic logic [8, 9], or formal argumentation [1], to name just a few. The second paradigm is about explicitly restoring consistency, thus changing the theory itself, as it is done in e. g. belief revision [18] or belief merging [23]. A quantitative approach for *analyzing* inconsistencies is given by the field *inconsistency measurement* [15, 20, 35, 21, 26, 27, 33, 3, 30, 29, 34] which investigates functions that assign real numbers to (usually classical) theories, with the intuitive meaning that large values indicate severe inconsistency.

Answer set programming (ASP, see [6] for an overview) is an emerging problem solving paradigm. It is based on logic programs under the answer set semantics [14, 13], a popular non-monotonic formalism for knowledge representation and reasoning which consists of rules possibly containing default-negated literals. Inconsistencies occur in ASP for two reasons, cf. [32]. First, the rules allow the derivation of two complementary literals l and $\neg l$—also called *incoherence* in e. g. [26]—thus producing inconsistencies similar to the classic-logical case. Second, due to the use of default negation it may happen that some literal assumed to be false is again derived (called *instability*). Hence, analyzing and handling inconsistency in ASP poses additional challenges (in comparison to the classical setting) that need to be addressed, cf. [12, 10]. Some few

works handle these challenges by adapting the classical techniques mentioned above to ASP, such as paraconsistent reasoning [5] or belief revision [7].

In this paper, we investigate the problem of *measuring* inconsistency in ASP. Due to the non-monotonicity of ASP, the classical framework of inconsistency measurement, and in particular its *rationality postulates* for measures, is not directly applicable. A central paradigm in classical inconsistency measurement is *monotonicity* of inconsistency: adding additional information to a (possibly already inconsistent) theory cannot decrease the value of inconsistency. This paradigm is not suitable for ASP, as we will discuss in more detail in Section 3. More specifically, the main contributions of this paper can be summarized as follows:

1. We revisit the notion of inconsistency measures for ASP, critically examine existing rationality postulates, and develop novel ones taking non-monotonicity into account (Section 4).
2. We develop four new inconsistency measures which are more appropriate for ASP than classical measures (Section 5).
3. We analyze our new measures by checking their compliance with the rationality postulates (Section 6).

Furthermore, we will give necessary preliminaries in Section 2 and conclude in Section 7. Proofs of technical results are omitted due to space restrictions but can be found in an extended version of this paper.[3]

2 Preliminaries

In this paper, we focus on extended logic programs under the answer set semantics [14, 13] which distinguish between classical negation "$\neg$" and default negation "not". Let $\mathcal{A}$ be a set of atoms and $\mathcal{L}$ the corresponding set of literals, i. e., for every $a \in \mathcal{A}$ we have $a \in \mathcal{L}$ and $\neg a \in \mathcal{L}$. For a set M of literals, let $\mathcal{A}$(M) be the set of all atoms occurring in M. Furthermore, let $\mathcal{L}^d = \mathcal{L} \cup \{\text{not } l \mid l \in \mathcal{L}\}$.

An extended logic program $P = \{r_1, ..., r_n\}$ is a set of rules of the form

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

where $l_0, \ldots, l_m \in \mathcal{L}$ and $0 \leq k \leq m$. For a rule r of the form (1) we write $head(r) = l_0$, $body(r) = \{l_1, \ldots, l_k, \text{not } l_{k+1}, \ldots, \text{not } l_m\}$, $pos(r) = \{l_1, \ldots, l_k\}$ and $neg(r) = \{l_{k+1}, \ldots, l_m\}$. We let $\mathcal{A}(r)$ and $\mathcal{L}(r)$ be the set of all atoms and literals occurring in r, respectively. Similarly, let $\mathcal{A}(P)$ and $\mathcal{L}(P)$ be the set of all atoms and literals that occur in a program P, respectively. Further, let $body(P) = \cup_{r \in P} body(r)$, and analogously for $pos(P)$ and $neg(P)$. Let $\mathcal{P}$ be the set of all extended logic programs. We write "l_0." instead of "$l_0 \leftarrow$." and call such rules facts. A rule without default negation is called a classical rule and a program P consisting only of a set of classical rules is called a classical logic program.[4]

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

[4] Similarly, rules (and programs) are called normal if the contain no classical negation, and definite if they contain no negation at all.

We now turn to the semantics, i. e., the definition of answer sets. There are actually variants of the definition in the literature which differ in whether inconsistent answer sets are admitted or not. The original definition in [14] allows for a single inconsistent answer set, namely $\mathcal{L}$, in cases where a subset of rules without default negation generates an inconsistency. Later in [13] the inconsistent answer set was abandoned, and all programs either have consistent answer sets or no answer set at all.

In this paper, we consider a third variant where different inconsistent answer sets may arise. This is motivated by the goals of this paper: we want to measure the degree of inconsistency of a program, and in this context disregarding inconsistent answer sets is obviously the wrong thing to do. Moreover, we are interested in more fine-grained distinctions than the single inconsistent answer set $\mathcal{L}$ would allow. For this reason answer sets in this paper can be arbitrary subsets of $\mathcal{L}$. Note that the variants derive from differences in the way answer sets of classical programs are defined.

For a set $M \subseteq \mathcal{L}$ and a literal l we say M satisfies l ($M \models l$) iff $l \in M$, and $M \models \text{not } l$ iff $l \notin M$. For a set $L \subseteq \mathcal{L}^d$, $M \models L$ iff $M \models l$ for all $l \in L$. For a rule r, $M \models r$ iff $M \models head(r)$ whenever $M \models body(r)$ and $M \models P$ for a program P iff $M \models r$ for each rule $r \in P$. Furthermore, for a classical program P we use $\mathsf{Cl}(P)$ to denote the unique $M \subseteq \mathcal{L}$ satisfying $M \models P$ and $M' \not\models P$ for each set M' of literals such that $M' \subsetneq M$.

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 a 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. P is called *strongly consistent* if it is has at least one answer set and all of its answer sets are consistent. 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.

Let P be a program with at least one answer set. We say P *entails* a literal l, denoted by $P \models l$, if $M \models l$ for all consistent answer sets $M \in \text{Ans}_{Con}(P)$. We say a P *strongly entails* l, denoted by $P \models_s l$, if $M \models l$ for all answer sets $M \in \text{Ans}(P)$, consistent or not. Similarly, P (strongly) entails a set L of literals if it (strongly) entails all $l \in L$, denoted as $P \models L$ ($P \models_s L$, respectively). If P is strongly consistent, then entailment and strong entailment obviously coincide.

Example 1. The program $P_1 = \{a \leftarrow \text{not } a.\}$ is inconsistent since it has no answer set. The program P_2 given via

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

has two answer sets, $\{a, \neg a\}$ and $\{b\}$. The latter is consistent and so is the program. b is entailed by the program, but not strongly entailed. If we add the rule "$\neg b \leftarrow b$." the program P_2 becomes inconsistent. Since $\emptyset$ is consistent, the program consisting of a single rule "$a \leftarrow b$." is consistent.

3 Measuring Inconsistency in Logic Programs

In the classical literature on inconsistency measurement—see e. g. [19, 15, 34]—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 classical setting, see [34] for a recent survey. Some further measures have also been proposed for first-order logic [16], description logics [35], and probabilistic and other weighted logics [33, 30, 29]. There are also works for measuring inconsistency for (fuzzy) answer set programming [26, 27] which will be discussed and compared to our work in Section 7.

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_3 and P_4 given as follows:

$$P_3 : \begin{array}{l} b \leftarrow \text{not } a. \\ \neg b \leftarrow \text{not } a. \end{array} \qquad P_4 : \begin{array}{l} b \leftarrow \text{not } a. \\ \neg b \leftarrow \text{not } a. \\ a. \end{array}$$

We have $P_3 \subseteq P_4$ but P_3 is inconsistent while P_4 is not, so we would expect $\mathcal{I}(P_4) < \mathcal{I}(P_3)$ 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. For example, consider one of the most basic classical measures $\mathcal{I}_{\mathsf{MI}}$ [20] defined via $\mathcal{I}_{\mathsf{MI}}(P) = |\mathsf{MI}(P)|$ where $\mathsf{MI}(P)$ is the set of minimal inconsistent subsets of P, i. e.

$$\mathsf{MI}(P) = \{P' \subseteq P \mid \text{Ans}_{Con}(P') = \emptyset, \forall P'' \subsetneq P' : \text{Ans}_{Con}(P'') \neq \emptyset\}.$$

For logic programs, this measure does not behave as expected as consistent programs may contain minimal inconsistent subsets. In particular, for P_3 and P_4 as above both programs have $\{b \leftarrow \text{not } a., \neg b \leftarrow \text{not } a.\}$ as a minimal inconsistent subset, yielding $\mathcal{I}_{\mathsf{MI}}(P_3) = \mathcal{I}_{\mathsf{MI}}(P_4) = 1$.

Many rationality postulates such as *monotonicity* from above are already disputed in the classical setting, cf. [3]. 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 4, propose some novel measures in Section 5, and analyse the latter in detail in Section 6.

4 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 [3] for a recent discussion on this topic. In the following, we revisit a selection of the most popular postulates—see e. g. [21, 33]—and phrase them within our context of logic programs.

Let $\mathcal{I} : \mathcal{P} \to \mathbb{R}^{\infty}_{\geq 0}$ be some inconsistency measure for logic programs and $P, P' \in \mathcal{P}$ some extended logic programs. The most central property of any inconsistency measure is that it is able to distinguish consistency from inconsistency.

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

The above postulate establishes that 0 is the minimal inconsistency value and that it is reserved for consistent programs.

We have already mentioned *monotonicity* as a desirable property for inconsistency measures (in classical logics) in the previous section.

Monotonicity $\mathcal{I}(P') \geq \mathcal{I}(P)$ whenever $P \subseteq P'$.

Satisfaction of this postulate is generally *not* desirable for ASP. However, as we still wish to require some form of monotonicity in special cases, we will consider some alternative formalizations of this principle. First, 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.

CLP-Monotonicity If P is a classical logic program and r^* a classical rule, then $\mathcal{I}(P) \leq \mathcal{I}(P \cup \{r^*\})$.

In the above postulate, CLP stands for "classical logic program". Furthermore, one could argue that it is fine, if both the program P and the added rule r^* contain default negation as long as we make sure that the rule is not "involved in non-monotonicity" of the program. To make this precise, we need the notion of the dependency graph:

Definition 3. *Let P be an extended logic program. The* dependency graph D_P *of a program P is a labeled directed graph having $\mathcal{L}(P)$ 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 $l_i \in pos(r) \cup neg(r)$. $s \in \{+, -\}$ indicates whether $l_i \in pos(r)$ or $l_i \in neg(r)$. For any $l \in \mathcal{L}(P)$, let* 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.*

The dependency graph D_P of a program P allows us to check whether some given literal has an influence on the derivation of other literals. Adding a rule r^* to a program P should not decrease the severity of inconsistency whenever the head of the rule has no influence on default-negated literals. In order to motivate this postulate we can make the following observations.

Lemma 1. *Let P be an extended logic program and r^* a rule with $head(r^*) = a$ such that* Path$(P \cup r^*, a) \cap neg(P \cup r^*) = \emptyset$. *Then, for all answer sets X of P, there is an answer set M of $P \cup \{r^*\}$ with $X \subseteq M$.*

Proof. Let X be an answer set of P. We show that there is a set $X \subseteq M$ such that M is an answer set of $P \cup \{r^*\}$.

By assumption, X is the minimal model of the reduct P^X. Consider $(P \cup \{r^*\})^X$. If $X \not\models \mathit{body}(\{r^*\}^X)$, then X is also a minimal model of $(P \cup \{r^*\})^X$. Thus, X is an answer set of $P \cup \{r^*\}$ which implies that our claim holds for $M = X$. So, we let $X \vDash \mathit{body}(\{r^*\}^X)$ and in particular, $\{r^*\}^X \neq \emptyset$. In this case, X is clearly no model of $(P \cup \{r^*\})^X$. We construct it and show that it contains X and is an answer set of P.

Since X is the minimal model of P^X and due to monotonicity of classical logic programming, $\mathsf{Cl}((P \cup \{r^*\})^X)$ has the form $Y \cup X$ for a set Y of literals. We can w. l. o. g. Y assume to be disjoint from X. To construct $\mathsf{Cl}((P \cup \{r^*\})^X)$, we use the usual fixed point iteration:

$$\Gamma_P^0(I) = \{l \in \mathcal{L}(P) \mid \exists r \in P :\ I \vDash \mathit{body}(r),\ \mathit{head}(r) = l\}\,.$$

We let $\Gamma_P^{i+1}(I) = \Gamma_P^i(\Gamma_P^0(I))$. Let $m \in \mathbb{N}$ be the smallest non-negative integer such that $\Gamma^{m+1}_{(P\cup\{r^*\})^X}(X) = \Gamma^m_{(P\cup\{r^*\})^X}(X)$. One can easily verify that $\Gamma^m_{(P\cup\{r^*\})^X}(X) =: M$ is the minimal model of $(P \cup \{r^*\})^X$.

Now we show:

- If we let $\Gamma^m_{(P\cup\{r^*\})^X}(X) = Y \cup X$ with $Y \cap X = \emptyset$ as mentioned above, then $Y \subseteq \mathsf{Path}(P \cup r^*, a)$.

Supposing, we proved it, then since $Y \subseteq \mathsf{Path}(P \cup r^*, a)$ implies $Y \cap \mathit{neg}(P \cup r^*) = \emptyset$, the reduct remains invariant, i. e., $(P \cup \{r^*\})^X = (P \cup \{r^*\})^{X \cup Y}$. So, since $X \cup Y$ is the minimal model of $(P \cup \{r^*\})^X$, it is also the minimal model of $(P \cup \{r^*\})^{X \cup Y}$. However, this is precisely the definition of $M = X \cup Y$ being an answer set of $P \cup \{r^*\}$, which proves our claim.

However, $Y \subseteq \mathsf{Path}(P \cup r^*, a)$ follows straightforward from induction over the number of iterations i: We show $Y \subseteq \mathsf{Path}((P \cup r^*)^X, a)$. Since the dependency graph of $(P \cup r^*)^X$ is a subgraph of $D_{P \cup r^*}$, this is sufficient.

- Base step $i = 0$: Because of $X = \mathsf{Cl}(P^X)$, our assumption $X \vDash \mathit{body}(\{r^*\}^X)$, $\{r^*\}^X \neq \emptyset$ and $\mathit{head}(r^*) = a$, we can calculate

$$\begin{aligned}
\Gamma^0_{(P\cup\{r^*\})^X}(X) &= \{l \in \mathcal{L}(P) \mid \exists r \in P^X \cup \{r^*\}^X :\ X \vDash \mathit{body}(r),\ \mathit{head}(r) = l\} \\
&= \{l \in \mathcal{L}(P) \mid \exists r \in P^X :\ X \vDash \mathit{body}(r),\ \mathit{head}(r) = l\} \\
&\cup \{l \in \mathcal{L}(P) \mid \exists r \in \{r^*\}^X :\ X \vDash \mathit{body}(r),\ \mathit{head}(r) = l\} \\
&= X \cup \{a\}.
\end{aligned}$$

  So, $\Gamma^0_{(P\cup\{r^*\})^X}(X)$ is indeed of the form $X \cup Y$ with $Y \subseteq \mathsf{Path}((P \cup \{r^*\})^X, a)$.
- Induction step $i \to i + 1$: Assume

$$\Gamma^i_{(P\cup\{r^*\})^X}(X) = X \cup Y_i$$

  with $Y_i \subseteq \mathsf{Path}((P \cup \{r^*\})^X, a)$. Consider a literal

$$l \in \Gamma^{i+1}_{(P\cup\{r^*\})^X}(X) \setminus (X \cup Y_i) = \Gamma^{i+1}_{(P\cup\{r^*\})^X}(X) \setminus \Gamma^i_{(P\cup\{r^*\})^X}(X).$$

Then, there is a rule $r \in (P \cup \{r^*\})^X$ such that $head(r) = l$. We have $X \not\models body(r)$, because otherwise,

$$l \in \Gamma^0_{(P\cup\{r^*\})^X}(X) \subseteq \Gamma^i_{(P\cup\{r^*\})^X}(X)$$

would hold. However, $l \in \Gamma^{i+1}_{(P\cup\{r^*\})^X}(X)$ implies $X \cup Y_i \models body(r)$. So, $body(r)$ and Y_i have at least one literal in common and since $Y_i \subseteq \mathsf{Path}((P \cup \{r^*\})^X, a)$, this means that $l = head(r) \in \mathsf{Path}((P \cup \{r^*\})^X, a)$. We obtain

$$\Gamma^{i+1}_{(P\cup\{r^*\})^X}(X) \setminus (X \cup Y_i) \subseteq \mathsf{Path}((P \cup \{r^*\})^X, a).$$

However,

$$\Gamma^{i+1}_{(P\cup\{r^*\})^X}(X) = \left(\Gamma^{i+1}_{(P\cup\{r^*\})^X}(X) \setminus (X \cup Y_i)\right) \cup X \cup Y_i$$

and hence, $\Gamma^{i+1}_{(P\cup\{r^*\})^X}(X)$ is of the form $X \cup Y_{i+1}$ with $Y_{i+1} \subseteq \mathsf{Path}((P \cup \{r^*\})^X, a)$. □

Lemma 1 states that we can augment any answer set of P with some additional literals (or none) to obtain an answer set of $P \cup \{r^*\}$. The other direction works as well.

Lemma 2. *Let P be an extended logic program and $r^* \notin P$ a rule with $head(r^*) = a$ such that $\mathsf{Path}(P \cup \{r^*\}, a) \cap neg(P \cup \{r^*\}) = \emptyset$. Then, for all answer sets M of $P \cup \{r^*\}$, there is an answer set X of P with $X \subseteq M$.*

Proof. Let M be an answer set of $P \cup \{r^*\}$. Then, M is the minimal model of $(P \cup \{r^*\})^M$. Consider P^M. This program has a minimal model, say X. As seen in the proof of Lemma 1, $\mathsf{Cl}((P \cup \{r^*\})^M) = M$ is of the form $X \cup Y$ with $Y \cap neg(P \cup r^*) = \emptyset$. Hence, $P^M = P^X$. Since X was assumed to be the minimal model of P^M, it is also the minimal model of P^X, i. e., an answer set of P. □

In particular, this means that moving from $P\cup\{r^*\}$ to P cannot introduce inconsistency.

Corollary 1. *Let P be an extended logic program and r^* a rule with $head(r^*) = a$ such that $\mathsf{Path}(P \cup \{r^*\}, a) \cap neg(P \cup \{r^*\}) = \emptyset$. If $P \cup \{r^*\}$ is consistent, then so is P.*

Proof. Let M be a consistent answer set of $P \cup \{r^*\}$. Then, there is a subset $X \subseteq M$ such that X is an answer set of P. M being consistent implies X is consistent. So, P has a consistent answer set. □

Example 2. Consider the inconsistent program P_5 given as follows:

$$P_5:\ a \leftarrow \text{not } b. \qquad \neg a \leftarrow \text{not } b. \qquad b \leftarrow c. \qquad c \leftarrow d.$$

If we add the fact $r^* = d.$, the program becomes consistent, having $\{b, c, d\}$ as the only answer set. Note that, even though $d \notin neg(P)$, there is a path from d to $b \in neg(P)$ in the dependency graph of P_5. So, we have $\mathsf{Path}(P, head(r^*)) \cap neg(P) \neq \emptyset$ and thus, this example does not contradict Corollary 1. Let us now add a rule that meets the premise of the corollary: Consider $P_5 \cup \{a.\}$. We have $Path(P, a) = \{a\}$ and $\{a\} \cap neg(P) = \emptyset$. Indeed, $P_5 \cup \{a.\}$ is inconsistent since it has $\{a, \neg a, \}$ as the only answer set.

Due to the above considerations, we deem the following postulate (I="Independence") as desirable.

I-Monotonicity If r^* is a rule with $\mathsf{Path}(P, head(r^*)) \cap neg(P \cup r^*) = \emptyset$, then $\mathcal{I}(P) \leq \mathcal{I}(P \cup \{r^*\})$.

One can see that I-Monotonicity is stronger than CLP-Monotonicity.

Proposition 1. *If $\mathcal{I}$ satisfies* I-Monotonicity*, then $\mathcal{I}$ satisfies* CLP-Monotonicity.

Proof. If P is a classical logic program and r^* a classical rule, then $\mathsf{Path}(P, head(r^*)) \cap neg(P) = \emptyset$ since $neg(P) = \emptyset$. $\mathcal{I}$ satisfies I-Monotonicity, yielding $\mathcal{I}(P) \leq \mathcal{I}(P \cup \{r^*\})$. □

To elaborate on the idea of independence in dependency graphs, we can also consider *splitting* of logic programs [25].

Definition 4. *Let P be an extended logic program. A set $U \subseteq \mathcal{L}(P)$ is called a* splitting set *for P, if $head(r) \in U$ implies $\mathcal{L}(r) \subseteq U$ for any 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.*

This means that if l is a literal that is not contained in U, i. e., $l \in \mathcal{L}(P) \setminus U$, then it cannot be the head of any rule outside of $bot_U(P)$. So, intuitively, the derivation of l is independent of $P \setminus bot_U(P)$. For the dependency graph D_P, this means that while there could be a path from a literal $l' \in U$ to l, the converse is not true. Splitting is used, for example, to calculate answer sets because it allows to handle $bot_U(P)$ without taking the rest of the program into account, see [25] for more details. However, since splitting is generally useful to examine the structure of a program, we are also interested in this notion here.

Example 3. Consider the program P_6 given by

$$P_6: \; b \leftarrow \text{not } a. \qquad \neg b \leftarrow \text{not } a. \qquad c \leftarrow b, \text{not } d. \qquad d \leftarrow \neg b, \text{not } c.$$

For the splitting set $U = \{a, \neg b, b\}$, $bot_U(P_6)$ is the program $bot_U(P_6) = \{b \leftarrow \text{not } a., \neg b \leftarrow \text{not } a.\}$.

Theorem 1 ([25]). *Let U be a splitting set for a program P. Every answer set M of P is of the form $M = X \cup Y$ with an answer set X of $bot_U(P)$ and a set Y of literals.*

Corollary 2. *Let U be a splitting set of P. If P is consistent, then so is $bot_U(P)$.*

Proof. Let U be a splitting set of P and M a consistent answer set. Due to Theorem 1, there is a subset X of M such that X is an answer set of $bot_U(P)$. M being consistent implies X is consistent. Thus, $bot_U(P)$ is consistent. □

Example 4. We continue Example 3. The bottom part $bot_U(P_6)$ has one answer set, namely $\{b, \neg b\}$. Due to Theorem 1, all answer sets of P_6 contain both b and $\neg b$. Indeed, one can easily verify that P_6 has the two answer sets $\{b, \neg b, c\}$ and $\{b, \neg b, d\}$. In particular, it is impossible to resolve the inconsistency of the program without changing the bottom part. For example, augmenting P_6 with the fact "a." would induce a consistent program, having $\{a\}$ as the unique answer set, but in this case, we modified $bot_U(P_6)$.

Intuitively, if $bot_U(P)$ is inconsistent, changing the rest of the program will not remove the reason why the bottom part is inconsistent; it is imposed on P. So, one could argue that P will always be at least as inconsistent as $bot_U(P)$.

Split-Monotonicity If U is a splitting set of P, then $\mathcal{I}(bot_U(P)) \leq \mathcal{I}(P)$.

Another postulates for classical inconsistency measures is *dominance* [21], which (in its original definition) requires $\mathcal{I}(\mathcal{K} \cup \{\alpha\}) \geq \mathcal{I}(\mathcal{K} \cup \{\beta\})$ for a set of classical formulas $\mathcal{K}$ whenever $\alpha \not\models \bot$ and $\alpha \models \beta$. The idea behind this notion is that, since α carries more information than β, $\mathcal{K}$ augmented with α is more likely to contain contradictions than augmented with β. As expected, this property needs some adjustment for ASP due to non-monotonicity.

Example 5. Consider P_3 and P_4 from above again:

$$P_3: \begin{array}{l} b \leftarrow \text{not } a. \\ \neg b \leftarrow \text{not } a. \end{array} \qquad P_4: \begin{array}{l} b \leftarrow \text{not } a. \\ \neg b \leftarrow \text{not } a. \\ a. \end{array}$$

Clearly, P_3 augmented with $\emptyset$—i. e., additional tautological information—is inconsistent, while $P_4 = P_3 \cup \{a.\}$ is not. So, dominance is not desirable here.

Since non-monotonicity is the reason why we cannot expect this notion of dominance for ASP, one could argue that the postulate should hold for classical logic programs at least.

CLP-Dominance If H, P and P' are classical logic programs and P is consistent such that $P' \models_s l$ implies $P \models_s l$ for all literals l, then $\mathcal{I}(H \cup P') \leq \mathcal{I}(H \cup P)$.

CLP-Dominance is similar to CLP-Monotonicity. In a sense, both postulates state that a classical logic program is considered more inconsistent the more information it carries. The difference is that CLP-Dominance takes the entailed literals into account rather than the program. It is not surprising that there is a relation between both postulates, which is formalized in the following Proposition.

Proposition 2. *If $\mathcal{I}$ satisfies* CLP-Dominance*, then $\mathcal{I}$ satisfies* CLP-Monotonicity.

Proof. CLP-Monotonicity is a special case: Let H be a classical logic program, r^* a classical rule and $\mathcal{I}$ a measure satisfying CLP-Dominance. If we let $P' = \emptyset$ and $P = \{r^*\}$, then P is consistent and P' does not entail anything, so the implication as in the description of CLP-Dominance clearly holds. Since $\mathcal{I}$ satisfies CLP-Dominance, we obtain $\mathcal{I}(H) = \mathcal{I}(H \cup P) \leq \mathcal{I}(H \cup P') = \mathcal{I}(H \cup \{r^*\})$ implying CLP-Monotonicity. □

Interestingly, the converse is not true: In Section 6, we will see that one of our proposed measures satisfies CLP-Monotonicity, but not CLP-Dominance.

We will now discuss a final postulate which considers cases where the inconsistency value should definitely *not* change. For that there is the notion of *safe* formulas [33]. A consistent classical formula $\alpha \in \mathcal{K}$ is *safe* in a set of classical formulas $\mathcal{K}$ if α and $\mathcal{K} \setminus$

$\{\alpha\}$ do not share any atoms. So, removing α from $\mathcal{K}$ will never resolve inconsistency. The corresponding postulate *safe-formula independence* requires $\mathcal{I}(\mathcal{K}) = \mathcal{I}(\mathcal{K} \cup \alpha)$ whenever $\alpha \in \mathcal{K}$ is safe.

One straightforward approach to adapt the notion of safe formulas in the ASP-setting could be defining safe rules in the following way: require $\mathcal{A}(P \setminus \{r^*\}) \cap \mathcal{A}(r^*) = \emptyset$ for a rule r^* and, to make sure r^* is consistent, forbid $head(r^*)$ as a literal in $neg(r^*)$. The latter would render rules like "$a \leftarrow$ not a." unsafe. But this notion of safe formulas would be clumsy: literals in the body of r^* that do not appear elsewhere in the program are meaningless, since they can never be derived. Hence, they always evaluate to "true" if they appear in $neg(r^*)$ and to "false" otherwise. Furthermore, whether r^* can be responsible for contradictions or not depends on the head of the rule only. Taking this into account, the following notion of safe rules seems reasonable.

Definition 5. *Let P be a logic program. A rule $r^* \in P$ is called* safe *with respect to P if $\mathcal{A}(head(r^*)) \cap \mathcal{A}(P \setminus \{r^*\}) = \emptyset$ and $pos(r^*) \cup neg(r^*) \subseteq \mathcal{L}(P)$.*

Now let P be a program, r^* safe with respect to P and M an answer set of P (consistent or not). Consider the reduct P^M. Then: $pos(r^*) \cup neg(r^*) \subseteq \mathcal{L}(P)$ ensures that whether $\{r^*\}^M = \emptyset$ or not and whether $M \vDash body(\{r^*\}^M)$ or not depends on $P \setminus \{r^*\}$ (and M) only. Additionally, $\mathcal{A}(head(r^*)) \cap \mathcal{A}(P \setminus \{r^*\}) = \emptyset$ ensures that in any case, $head(r^*)$ is not involved in contradictions in M. Furthermore, as long as $M \vDash head(\{r^*\}^M)$ if and only if $M \vDash body(\{r^*\}^M)$, this rule does not influence whether $M = \mathsf{Cl}(P^M)$ or not. Since this guarantees that r^* can never be responsible for (the absence of) inconsistency, the following postulate is desirable.

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

Example 6. Let P_7 be a program consisting of the following rules:

$$P_7: \; b \leftarrow \text{not } a. \qquad \neg b \leftarrow \text{not } a. \qquad c \leftarrow \text{not } a.$$

The rule "$c \leftarrow$ not a." is safe because c does not appear elsewhere in P_7. P_7 has one answer set, namely $\{b, \neg b, c\}$, while $P_7 \setminus \{c \leftarrow \text{not } a.\}$ has $\{b, \neg b\}$ as answer set. An inconsistency measure satisfying *safe-rule independence* would assess P_7 and $P_7 \setminus \{c \leftarrow \text{not } a.\}$ equally inconsistent.

Consider an extended logic program P and a rule r^* that is safe with respect to P. We can view $U := \mathcal{L}(P \setminus \{r^*\})$ as splitting set of $P \cup \{r^*\}$ and obtain P as bottom part. Then, for any measure $\mathcal{I}$ that satisfies Split-Monotonicity, we have $\mathcal{I}(P) \leq \mathcal{I}(P \cup \{r^*\})$. The same holds for any measure $\mathcal{I}$ satisfying I-Monotonicity, since $\mathsf{Path}(P, head(r^*)) \cap neg(P) = \emptyset$ is clear for a safe rule r^*. Both Split-Monotonicity and I-Monotonicity do not imply Safe-rule independence, though, since we do not obtain $\mathcal{I}(P) \geq \mathcal{I}(P \cup \{r^*\})$.

5 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. [32]. 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*

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

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

Example 7. Let P_8 be the program given via

$$P_8 : \qquad a_1 \leftarrow \text{not } b. \qquad \neg a_1 \leftarrow \text{not } b.$$

P_8 is inconsistent, but since $P_8 \cup \{b.\}$ is already consistent, we obtain $\mathcal{I}_\pm(P_8) = 1$. Now let P_9 be defined via

$$\begin{array}{lll} P_9 : & a_1 \leftarrow \text{not } b. & \neg a_1 \leftarrow \text{not } b. \\ & a_1 \leftarrow \text{not } c. & \neg a_1 \leftarrow \text{not } c. \\ & a_1 \leftarrow \text{not } d. & \neg a_1 \leftarrow \text{not } d. \end{array}$$

P_9 contains three contradicting pairs of rules. Since one can delete one rule in each of them (or make them inapplicable by adding the corresponding fact), $\mathcal{I}_\pm(P_9) = 3$. Now consider P_{10} given by

$$\begin{array}{lll} P_{10} : & a_1 \leftarrow \text{not } b. & \neg a_1 \leftarrow \text{not } b. \\ & a_2 \leftarrow \text{not } b. & \neg a_2 \leftarrow \text{not } b. \\ & a_3 \leftarrow \text{not } b. & \neg a_3 \leftarrow \text{not } b. \end{array}$$

Even though P_{10} also contains three contradicting pairs of rules, $\mathcal{I}_\pm(P_{10}) = 1$ since $P_{10} \cup \{b.\}$ is consistent.

The definition of $\mathcal{I}_\pm$ allows the addition of any rule in order to restore consistency. But in fact, it is sufficient to only consider addition of facts instead of general rules, as the next proposition shows.

Proposition 3. *Let* P *be an inconsistent program. If* r^* *is a rule such that* $P \cup \{r^*\}$ *is consistent, then* $P \cup \{head(r^*).\}$ *is also consistent.*

Proof. Let M be an answer set of $P \cup \{r^*\}$. P being inconsistent implies that M is no answer set of P. Thus, $M \vDash body(\{r^*\}^M)$ and $\{r^*\}^M \neq \emptyset$ because otherwise one could delete the rule while maintaining M as answer set. Since M is a model of P^M, we obtain $head(r^*) \in M$. However, this means M is an answer set of $P \cup \{head(r^*).\}$. □

$\mathcal{I}_\pm$ 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 $A, D \in \mathcal{P}$ such that $|A| \leq k$ and $|D| \leq l$. 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*

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

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

In other words, $\mathcal{I}^\pm$ counts the minimal amount of rules we have to add to or delete from the reduct of M to obtain a classical logic program that has M as the minimal model.

Example 8. Let us consider the programs P_8, P_9 and P_{10} from Example 7 again. The reduct $P_8^{\{a_1\}}$ is given by two facts, namely $\{a_1., \neg a_1.\}$. Since $\{a_1\}$ is a model of it after deleting the fact $\{\neg a_1.\}$, it is a 0-1-model of the reduct. Hence, $\mathcal{I}^\pm(P_8) = 1$. In Example 7, we obtained a consistent program by adding the fact $\{b.\}$ to P_8. Indeed, $\{b\}$ is a 1-0-answer set of P_8: The reduct $P_8^{\{b\}}$ is the empty program and $\{b\}$ is clearly a 1-0-model of it. One can also easily verify that $\{a_1\}$ is a 0-3-answer set of P_9 and $\{b\}$ a 1-0-answer set of P_{10} and hence, $\mathcal{I}^\pm(P_9) = 3$ and $\mathcal{I}^\pm(P_{10}) = 1$.

Note that $\mathcal{I}^\pm$ is similar in spirit to $\mathcal{I}_\pm$ but considers hypothetical modifications of the reduct rather than the original program. Interestingly, however, these two different points of view are actually equivalent.

Proposition 4. *For any extended logic program P, $\mathcal{I}_\pm(P) = \mathcal{I}^\pm(P)$.*

Proof. "$\geq$": Let $\mathcal{I}_\pm(P) = k$. Suppose $\tilde{P} = (P \cup A) \backslash D$ is consistent and let $|A| = k_A$ and $|D| = k_D$ with $k_A + k_D = k$. Let M be an answer set of $(P \cup A) \backslash D$. We show that M is a k_A-k_D-answer set of P. Hence, $\mathcal{I}^\pm(P) \leq k_A + k_D = k$. We denote P, A and D by $\{r_1, ..., r_n\}, \{r_{A1}, ..., r_{Ak_A}\}$ and $\{r_{D1}, ..., r_{Dk_D}\}$, respectively. The reduct $\tilde{P}^M$ is given by

$$\tilde{P}^M = \left(\{r_1, ..., r_n\}^M \cup \{r_{A1}, ..., r_{Ak_A}\}^M\right) \setminus \{r_{D1}, ..., r_{Dk_D}\}^M$$

and by assumption, M is the minimal model of $\tilde{P}^M$. Now consider $P^M = \{r_1, ..., r_n\}^M$. By deleting $\{r_{D1}, ..., r_{Dk_D}\}^M$ and adding $\{r_{A1}, ..., r_{Ak_A}\}^M$, we obtain $\tilde{P}^M$ having M as a minimal model. Hence, M is a k_A-k_D-answer set of P.

"$\leq$": Now let M be a k_A-k_D-answer set of P s. t. $k_A + k_D = k$ and $\mathcal{I}^\pm(P) = k$. Suppose M is the minimal model of $\tilde{P}^M$ with

$$\tilde{P}^M = \left(P^M \cup \{r_{A1}, ..., r_{Ak_A}\}\right) \setminus \{r_{D1}, ..., r_{Dk_D}\}^M.$$

Since $r_{A1}, ..., r_{Ak_A}$ do not contain default negation, $\{r_{A1}, ..., r_{Ak_A}\}^M = \{r_{A1}, ..., r_{Ak_A}\}$. So, the reduct of

$$\hat{P} = (P \cup \{r_{A1}, ..., r_{Ak_A}\}) \setminus \{r_{D1}, ..., r_{Dk_D}\}$$

coincides with $\tilde{P}^M$. Thus, M is an answer set of $\hat{P}$ and hence, $\mathcal{I}_{\pm}(P) \leq k_A + k_D = k$. □

The next measure we propose mainly focuses on programs without any answer set. 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. However, for any M the reduct P^M is a classical logic program and thus has a unique answer set $\mathsf{Cl}(P^M)$. So, it seems reasonable to measure the distance between M and $\mathsf{Cl}(P^M)$. In the following, we only consider the number of literals in the symmetric difference of two sets as an example of a distance measure between sets. Investigating other distances is left for future work.

Definition 9. *Let M and M' be two sets of literals. The* sd-distance *(sd="symmetric difference") $d_{sd}(M, M')$ between M and M' is defined via $d_{sd}(M, M') = |(M \cup M') \setminus (M \cap M')|$.*

Definition 10. *Define $\mathcal{I}_{sd} : \mathcal{P} \to \mathbb{R}^{\infty}_{\geq 0}$ via*

$$\mathcal{I}_{sd}(P) = \min_{\substack{M \subseteq \mathcal{L} \\ M \text{ consistent}}} \{d_{sd}(M, \mathsf{Cl}(P^M)) \mid \mathsf{Cl}(P^M) \text{ is consistent}\}$$

for all $P \in \mathcal{P}$ with $\min \emptyset = \infty$.

Example 9. If we consider $P_{11} = \{a., \neg a.\}$ we obtain $\mathcal{I}_{sd}(P_{11}) = \infty$ because no matter which set M of literals we choose, the reduct P_{11}^M will always have the inconsistent answer set $\{a, \neg a\}$. If we let $P_{12} = \{a \leftarrow \text{not } a., b \leftarrow \text{not } b., c.\}$, then for the set $\{a, b, c\}$ we obtain the reduct $P_{12}^{\{a,\,b,\,c\}} = \{c.\}$ and hence, $d_{sd}(\{a, b, c\}, \mathsf{Cl}(P_{12}^{\{a,\,b,\,c\}})) = 2$. Indeed, one can verify $\mathcal{I}_{sd}(P_{12}) = 2$ which is intuitive since this is the amount of reasons why P_{12} has no answer set. Finally, consider P_{10} from Example 7 again. In this case, $\mathcal{I}_{sd}$ behaves similar to $\mathcal{I}_{\pm}$: adding b is the most efficient way to resolve inconsistency. The reduct $P_{10}^{\{b\}}$ is the empty set and hence, $\mathcal{I}_{sd}(P_{10}) = 1$.

It seems reasonable to measure inconsistency based on the amount of contradictions a program induces. So, one conceivable approach is counting the number of complementary literals in any answer set. Since one consistent answer set is sufficient, we can safely choose the answer set with the lowest amount of contradictions to assess the program. Of course, we need to be able to distinguish inconsistent answer sets which is one reason why we allow any subset of $\mathcal{L}$ to be an answer set. This motivates the following definition.

Definition 11. *A set M of literals is called k-*inconsistent*, $k \in \mathbb{N} \cup \{0\}$, if there are exactly k atoms a s. t. $a \in M$ and $\neg a \in M$.*

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

Table 1. Compliance of inconsistency measures wrt. our rationality postulates

Further, we have to take into account that a program might be inconsistent due to having no answer set. In this case, the described approach does not work. We assign ∞ to such programs as they are a special case for this measure.

Definition 12. *Define* $\mathcal{I}_\# : \mathcal{P} \to \mathbb{R}^\infty_{\geq 0}$ *via*

$$\mathcal{I}_\#(P) = \begin{cases} \min_{M \in \mathit{Ans}(P)}\{k \mid M \text{ is } k\text{-inconsistent}\} & \text{if } \mathit{Ans}(P) \neq \emptyset, \\ \infty & \text{otherwise} \end{cases}$$

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

Example 10. We consider our running example again. Both P_8 and P_9 have the answer set $\{a_1,\ \neg a_1\}$ and hence, $\mathcal{I}_\#(P_8) = \mathcal{I}_\#(P_9) = 1$. Meanwhile, $\mathcal{I}_\#(P_{10}) = 3$ since P_{10} has $\{a_1,\ \neg a_1,\ a_2,\ \neg a_2,\ a_3,\ \neg a_3\}$ as its unique answer set.

6 Analysis

Table 1 gives an overview on the compliance of our measures with respect to the rationality postulates from Section 4 and thus summarizes Propositions 5, 6, 7, and Example 11 below. Note that, naturally, none of our measures satisfies the classical *monotonicity* postulate which is also not desired for ASP, cf. Section 3.

Proposition 5. $\mathcal{I}_\pm$ *satisfies* Consistency, CLP-Monotonicity, I-Monotonicity, Split-Monotonicity, *and* Safe-rule independence.

Proof. **Consistency** Clear due to definition of $\mathcal{I}_\pm$.
CLP-Monotonicity Note that adding classical rules to an inconsistent classical logic program will never resolve inconsistency. Let $\mathcal{I}_\pm(P \cup \{r^*\}) = k$ and let $(P \cup \{r^*\}) \setminus D$ with $|D| = k$ be consistent. Here, we can assume $A = \emptyset$ for the following reason: Due to Proposition 3, we can w. l. o. g. assume A to be a set of facts and in particular, A is a set of classical rules in this case. So, P is a classical logic program and we only delete rules and add facts. Thus, we will always consider classical programs where adding rules cannot resolve inconsistency, as already mentioned above. So, w. l. o. g. $A = \emptyset$. Now, $(P \cup \{r^*\}) \setminus D$ being consistent implies $P \setminus D$ is also consistent, because

otherwise, adding $\{r^*\}$ to $P \setminus D$ would restore consistency which is impossible. Hence, $\mathcal{I}_{\pm}(P) \leq |D| = k$.

I-Monotonicity Let $\mathcal{I}_{\pm}(P \cup \{r^*\}) = k$. Let $(P \cup \{r^*\} \cup A) \setminus D$ be consistent with $|A| + |D| = k$. Let M be a consistent answer set of $(P \cup \{r^*\} \cup A) \setminus D$. Using Proposition 3, we assume that A is a set of facts. So, since

$$\mathsf{Path}(P, head(r^*)) \cap neg(P) = \emptyset,$$

we also obtain

$$\mathsf{Path}((P \cup A) \setminus D, head(r^*)) \cap neg(P) = \emptyset,$$

because adding facts (and deleting rules) does not extend the dependency graph. Furthermore, we have

$$neg((P \cup A) \setminus D) = neg(P \setminus D) \subseteq neg(P)$$

and hence,

$$\mathsf{Path}((P \cup A) \setminus D, head(r^*)) \cap neg(P) = \emptyset$$

also implies

$$\mathsf{Path}((P \cup A) \setminus D, head(r^*)) \cap neg((P \cup A) \setminus D) = \emptyset.$$

So, we can apply Lemma 2 to the program $(P \cup A) \setminus D$ and the additional rule r^*: Since M was assumed to be an answer set of$(P \cup \{r^*\} \cup A) \setminus D$, we obtain that $(P \cup A) \setminus D$ has an answer set X with $X \subseteq M$. With M being consistent, X is consistent, too. Likewise, $(P \cup A) \setminus D$ is consistent. Thus, $\mathcal{I}_{\pm}(P) \leq |A| + |D| = k = \mathcal{I}_{\pm}(P \cup \{r^*\})$.

Split-Monotonicity Let P be a program and U a splitting set. Let $\mathcal{I}_{\pm}(P) = k$ and let $(P \cup A) \setminus D$ be consistent with $|A| + |D| = k$. As usual, we use Proposition 3 to assume that A is a set of facts. Thus, $(P \cup A) \setminus D$ contains no additional edges in the dependency graph compared to P which implies that U is also a splitting set of $(P \cup A) \setminus D$. In particular, if we let A_U be the subset of A such that $r \in A_U$ if and only if $head(r) \in U$, then $(bot_U(P)) \cup A_U) \setminus D$ is the corresponding bottom program. Now let M be a consistent answer set of $(P \cup A) \setminus D$. Due to Theorem 1, there is a subset $X \subseteq M$ s. t. X is an answer set of $(bot_U(P)) \cup A_U) \setminus D$. As a subset of the consistent set M of literals, X is consistent. Therefore, $(bot_U(P)) \cup A_U) \setminus D$ is consistent. Hence,

$$\mathcal{I}_{\pm}((bot_U(P)) \cup A_U) \setminus D) \leq |D| + |A_U| \leq |D| + |A| = k.$$

Safe-Rule Independence $\mathcal{I}_{\pm}(P) \leq \mathcal{I}_{\pm}(P \cup \{r^*\})$ follows from Split-Monotonicity since P can be seen as bottom part of $P \cup \{r^*\}$. On the other hand, let $\mathcal{I}_{\pm}(P) = k$ and let $(P \cup A) \setminus D$ be consistent with $|A| + |D| = k$. Clearly, we can assume $(A \cup D) \cap \{head(r^*)\} = \emptyset$. Now, if M is a consistent answer set of $(P \cup A) \setminus D$, then either M itself or $M \cup \{head(r^*)\}$ is also an answer set of $(P \cup A \cup \{r^*\}) \setminus D$, depending on whether $\{r^*\}^M \neq \emptyset$ and $M \vDash body(\{r^*\}^M)$ or not. But since r^* is safe, both M and $M \cup \{head(r^*)\}$ are consistent. So, $(P \cup A \cup \{r^*\}) \setminus D$ is consistent and hence, $\mathcal{I}_{\pm}(P \cup \{r^*\}) \leq \mathcal{I}_{\pm}(P)$. □

The following example shows that $\mathcal{I}_{\pm}$ does not satisfy CLP-Dominance. Note that this also shows that CLP-Monotonicity does not imply CLP-Dominance, since $\mathcal{I}_{\pm}$ satisfies CLP-Monotonicity. Consequently, this confirms that the converse of Proposition 2 is not true, indeed.

Example 11. Consider the program H_1 given via

$$H_1 : \qquad \neg a. \qquad d. \qquad \neg a \leftarrow d. \qquad e. \qquad \neg a \leftarrow e.$$

Note that in order to prevent H_1 from entailing $\neg a$ at least three rules have to be deleted. Now consider P_{13} and P_{14} given as follows:

$$\begin{array}{lccc} P_{13} : & a. & b. & c. \\ P_{14} : & a. & b. & a \leftarrow b. \end{array}$$

Clearly, both P_{13} and P_{14} are consistent and all literals entailed by P_{14} are also entailed by P_{13} (and not the other way round). To prevent $P_{13} \cup H_1$ from entailing a contradiction, it is sufficient to delete "a.", while for $P_{14} \cup H_1$ one needs to delete two rules, e. g., "a." and "b.". So, $\mathcal{I}_{\pm}(P_{13} \cup H_1) = 1 < 2 = \mathcal{I}_{\pm}(P_{14} \cup H_1)$. Thus, $\mathcal{I}_{\pm}$ does not satisfy CLP-Dominance.

Proposition 6. *$\mathcal{I}_{\#}$ satisfies* Consistency, CLP-Monotonicity, I-Montonicity, Split-Monotonicity, Safe-Rule Independence, *and* CLP-Dominance.

Proof. **Consistency** Clear due to definition.
CLP-Monotonicity CLP-Dominance and Proposition 2.
I-Monotonicity Let $\mathcal{I}_{\#}(P \cup \{r^*\}) = k^*$ and let M be a k^*-inconsistent answer set of $P \cup \{r^*\}$. By Lemma 2, there is an answer set X of P with $X \subseteq M$. So, if X is k-consistent, then $k \leq k^*$ and we obtain $\mathcal{I}_{\#}(P) \leq k \leq k^* = \mathcal{I}_{\#}(P \cup \{r^*\})$.
Split-Monotonicity Let P be a program and U a splitting set. Let X^* be the answer set of $bot_U(P)$ with a minimal amount of complementary literals, say $2k$. Thus, $\mathcal{I}_{\#}(bot_U(P)) = k$. Now let M be an answer set of P. Due to Theorem 1, $X \subseteq M$ for an answer set X of $bot_U(P)$. Thus, M contains at least as many complementary literals as X, which itself contains at least as many as X^*. Hence, $\mathcal{I}_{\#}(P) \geq k$.
Safe-Rule Independence Since $\mathcal{A}(head(r^*)) \cap \mathcal{A}(P \setminus \{r^*\}) = \emptyset$ and $pos(r^*) \cup neg(r^*) \subseteq \mathcal{L}(P)$, the correspondence between the answer sets of P and $P \cup \{r^*\}$ is straightforward and the amount of complementary literals in any answer set remains unchanged.
CLP-Dominance Due to monotonicity of classical logic programs, $\mathsf{Cl}(P') \subseteq \mathsf{Cl}(P)$ implies $\mathsf{Cl}(P' \cup H) \subseteq \mathsf{Cl}(P \cup H)$. So, the latter contains at least as many complementary literals as the former. By definition of $\mathcal{I}_{\#}$, we obtain $\mathcal{I}_{\#}(P' \cup H) \leq \mathcal{I}_{\#}(P \cup H)$ □

Proposition 7. *$\mathcal{I}_{sd}$ satisfies* Consistency, CLP-Monotonicity, I-Monotonicity, Split-Monotonicity, Safe-Rule independence, *and* CLP-Dominance.

Proof. **Consistency** Assume $\mathcal{I}_{sd}(P) = 0$. In this case, a set M of literals can be found such that M is consistent and coincides with $\mathsf{Cl}(P^M)$. Thus, M is a consistent answer set of P implying that P is consistent. Now let us assume P is consistent and M is a

consistent answer set of P. By definition, $d_{sd}(M, \mathsf{Cl}(P^M)) = 0$ and hence, $\mathcal{I}_{sd}(P) = 0$.

CLP-Monotonicity CLP-Dominance and Proposition 2.

I-Monotonicity The case $\mathcal{I}_{sd}(P \cup \{r^*\}) = \infty$ is clear. So, we let $\mathcal{I}_{sd}(P \cup \{r^*\}) = k < \infty$. Let M be a consistent set of literals such that $\mathsf{Cl}((P \cup \{r^*\})^M) = M'$ is consistent and let $d_{sd}(M, M') = k$.

Consider P^M. As seen in the proof of Lemma 1, if we let $\mathsf{Cl}(P^M) = X'$, then $\mathsf{Cl}((P \cup \{r^*\})^M)$ is of the form $X' \cup Y'$ for a set Y' of literals that is disjoint from X' with $P^M = P^{M \setminus Y'}$.

Now consider $X = M \setminus Y'$. We have $X' = \mathsf{Cl}(P^M) = \mathsf{Cl}(P^{M \setminus Y'})$ and

$$d_{sd}(X, X') = d_{sd}(M \setminus Y', M' \setminus Y') \leq d_{sd}(M, M') = k.$$

This implies $\mathcal{I}_{sd}(P) \leq k$.

Split-Monotonicity Let P be an extended logic program, U a splitting set and $bot_U(P)$ the corresponding bottom program. Again, we only have to consider the case $\mathcal{I}_{sd}(P) = k < \infty$. Let M be a consistent set of literals with $d_{sd}(M, \mathsf{Cl}(P^M)) = k$, such that $\mathsf{Cl}(P^M)$ is consistent. Note that U is a splitting set of P^M and $(bot_U(P))^M$ is the corresponding bottom part. So, due to Theorem 1, $\mathsf{Cl}(P^M) =: M'$ is of the form $X' \cup Y'$ with $X' = \mathsf{Cl}((bot_U(P))^M)$. We can w. l. o. g. assume $X' \cap Y' = \emptyset$.

Now consider $X = M \cap U$ and $Y = M \setminus X$. Then, similar to X' and Y' we have $X \cup Y = M$ and $X \cap Y = \emptyset$.

We obtain $(bot_U(P))^M = (bot_U(P))^X$ due to the construction of the reduct and U being a splitting set. Thus, $X' = \mathsf{Cl}((bot_U(P))^X)$.

The last step argues that the constructed sets are disjoint: We have $X, X' \subseteq U$ and $Y \cap U = Y' \cap U = \emptyset$. So, $X \cap Y' = X' \cap Y = \emptyset$. Furthermore, $X \cap Y = X' \cap Y' = \emptyset$ was already mentioned. Thus, we can calculate

$$d_{sd}(X, X') \leq d_{sd}(X \cup Y, X' \cup Y') = d_{sd}(M, M') = k.$$

To summarize, we found a set X of literals with $X' = \mathsf{Cl}((bot_U(P))^X)$ and $d_{sd}(X, X') \leq k$. Hence, $\mathcal{I}_{sd}(bot_U(P)) \leq k$.

Safe-Rule independence $\mathcal{I}_{sd}(P) \leq \mathcal{I}_{sd}(P \cup \{r^*\})$ follows from Split-Monotonicity since P can be seen as bottom part of $P \cup \{r^*\}$.

We show "$\geq$": Let $\mathcal{I}_{sd}(P) = k < \infty$. Let M be consistent, $M' = \mathsf{Cl}(P^M)$ and $d_{sd}(M, M') = k$. If $M' \not\models body(\{r^*\}^M)$, then $M' = \mathsf{Cl}((P \cup \{r^*\})^M)$ and the claim clearly holds. Otherwise, consider $M \cup \{head(r^*)\}$ and $M' \cup \{head(r^*)\}$. Since $\mathcal{A}(head(r^*)) \cap \mathcal{A}(P) = \emptyset$, we have $(P \cup \{r^*\})^M = (P \cup \{r^*\})^{M \cup \{head(r^*)\}}$ and for the same reason, M' being the minimal model of $P^M = P^{M \cup \{head(r^*)\}}$ implies that $M' \cup \{head(r^*)\}$ is the minimal model of $(P \cup \{r^*\})^{M \cup \{head(r^*)\}}$. Further, augmenting both sets with the additional literal does not change their distance, i. e.,

$$d_{sd}(M' \cup \{head(r^*)\}, M \cup \{head(r^*)\}) = d_{sd}(M', M) = k.$$

Hence, $\mathcal{I}_{sd}(P) = k \geq \mathcal{I}_{sd}(P \cup \{r^*\})$.

CLP-Dominance In this case, $\mathcal{I}_{sd}(P)$ depends only on whether the unique answer set of a classical program is consistent or not. Due to monotonicity of classical logic programs, we have $\mathcal{I}_{sd}(H \cup P') \leq \mathcal{I}_{sd}(H \cup P)$. □

Our analysis shows that the measures we considered here are indeed reasonable. The proposed postulates are adapted from existing ones for classical logics, and the most interesting ones are satisfied by our measures. Monotonicity is not a desirable property for any inconsistency measure in the context of ASP anyway. Only $\mathcal{I}_\pm$ violates one postulate which may be desired in a specific context. Our results thus provide the basis for an informed choice of a suitable measure in a particular application scenario.

We already showed that $\mathcal{I}_\pm$ is equivalent to $\mathcal{I}^\pm$ but it is still not clear whether there exist some other relationships between our new measures. In order to compare inconsistency measures we can use the following *refinement order*, cf. [17]. For two inconsistency measures $\mathcal{I}_1$ and $\mathcal{I}_2$, the refinement order $\sqsubseteq$ is defined as $\mathcal{I}_1 \sqsubseteq \mathcal{I}_2$ iff $\mathcal{I}_2(P) \geq \mathcal{I}_2(P')$ implies $\mathcal{I}_1(P) \geq \mathcal{I}_1(P')$ for all $P, P' \in \mathcal{P}$. $\mathcal{I}_1 \sqsubseteq \mathcal{I}_2$ means that $\mathcal{I}_2$ is a refinement of $\mathcal{I}_1$. If both $\mathcal{I}_1 \sqsubseteq \mathcal{I}_2$ and $\mathcal{I}_2 \sqsubseteq \mathcal{I}_1$, then $\mathcal{I}_1$ and $\mathcal{I}_2$ are *order-compatible*, i. e., they induce the same ranking on programs without necessarily assigning the same inconsistency values. Indeed, all our measures are incompatible, thus truly novel, and provide different points of view on inconsistency.

Proposition 8. *For every $\mathcal{I}_1, \mathcal{I}_2 \in \{\mathcal{I}_\pm, \mathcal{I}_\#, \mathcal{I}_{sd}\}$ with $\mathcal{I}_1 \neq \mathcal{I}_2$, $\mathcal{I}_1 \not\sqsubseteq \mathcal{I}_2$.*

Proof. $\mathcal{I}_\pm$ and $\mathcal{I}_\#$: Consider P_9 and P_{10} from Example 7 again:

$$P_9: \quad a_1 \leftarrow \text{not } b. \quad \neg a_1 \leftarrow \text{not } b. \quad a_1 \leftarrow \text{not } c. \quad \neg a_1 \leftarrow \text{not } c. \quad a_1 \leftarrow \text{not } d. \quad \neg a_1 \leftarrow \text{not } d.$$

and

$$P_{10}: \quad a_1 \leftarrow \text{not } b. \quad \neg a_1 \leftarrow \text{not } b. \quad a_2 \leftarrow \text{not } b. \quad \neg a_2 \leftarrow \text{not } b. \quad a_3 \leftarrow \text{not } b. \quad \neg a_3 \leftarrow \text{not } b.$$

We already saw that $\mathcal{I}_\pm(P_9) = 3$ and $\mathcal{I}_\pm(P_{10}) = 1$, but $\mathcal{I}_\#(P_9) = 1$ and $\mathcal{I}_\#(P_{10}) = 3$. So, $\mathcal{I}_\pm \not\sqsubseteq \mathcal{I}_\#$ and $\mathcal{I}_\# \not\sqsubseteq \mathcal{I}_\pm$.

$\mathcal{I}_\pm$ and $\mathcal{I}_{sd}$: For $P_{11} = \{a., \neg a.\}$ and $P_{12} = \{a \leftarrow \text{not } a., b \leftarrow \text{not } b., c.\}$ we obtain $\mathcal{I}_\pm(P_{11}) = 1$ and $\mathcal{I}_\pm(P_{12}) = 2$, but $\mathcal{I}_{sd}(P_{11}) = \infty$ and $\mathcal{I}_{sd}(P_{12}) = 2$.

$\mathcal{I}_{sd}$ and $\mathcal{I}_\#$: We have $\mathcal{I}_{sd}(P_9) = 3$ (use $M = \{a, , b , c\}$ and $P_9^M = \emptyset$) and $\mathcal{I}_{sd}(P_{10}) = 1$ as already pointed out. On the contrary, $\mathcal{I}_\#(P_9) = 1$ and $\mathcal{I}_\#(P_{10}) = 3$.

This completes the proof. □

Of course, it holds $\mathcal{I}_\pm \sqsubseteq \mathcal{I}^\pm$ and $\mathcal{I}^\pm \sqsubseteq \mathcal{I}_\pm$ due to $\mathcal{I}_\pm = \mathcal{I}^\pm$.

7 Summary and Discussion

In this paper, we addressed the challenge of measuring inconsistency in ASP by critically reviewing the classical framework of inconsistency measurement and taking non-monotonicity 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. We showed that our new measures comply with many of our rationality postulates and illustrated their usage.

To the best of our knowledge, measuring inconsistency in extended logic programs under the answer set semantics has not been addressed before. The closest related works are by Madrid and Ojeda-Aciego, see e. g. [26, 27], who address inconsistencies in residuated logic programs under fuzzy answer set semantics. In their setting, rules such as (1) are augmented with fuzzy values in $[0, 1]$ (or some arbitrary lattice) and inconsistency is measured by considering minimal changes in the values to restore the existence of fuzzy stable models. However, Madrid and Ojeda-Aciego do not discuss the classical case and rationality postulates.

In future work, we would like to extend our analysis to more general classes of logic programs, e. g., disjunctive logic programs which allow for disjunctions in the heads of rules, programs with choice rules, weight constraints and aggregates. For an overview on these extensions see [11]. It would also be interesting to see whether our measures, or similar ones, can be applied to other non-monotonic formalisms, like default logic [31] or autoepistemic logic [28].

References

1. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. The Knowledge Engineering Review 26(4), 365–410 (2011)
2. Benferhat, S., Dubois, D., Prade, H.: A local approach to reasoning under inconsisteny in stratified knowledge bases. In: Proceedings of the Third European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'95) (1995)
3. 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)
4. Béziau, J.Y., Carnielli, W., Gabbay, D. (eds.): Handbook of Paraconsistency. College Publications, London (2007)
5. Blair, H.A., Subrahmanian, V.: Paraconsistent logic programming. Theoretical Computer Science 68(2), 135 – 154 (1989), http://www.sciencedirect.com/science/article/pii/0304397589901266
6. 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
7. Delgrande, J.P., Schaub, T., Tompits, H., Woltran, S.: Belief revision of logic programs under answer set semantics. In: Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning. pp. 411–421 (2008)
8. Dubois, D., Lang, J., Prade, H.: Inconsistency in possibilistic knowledge bases: To live with it or not live with it. In: Zadeh, L., Kacprzyk, J. (eds.) Fuzzy Logic for the Management of Uncertainty , pp. 335–351. Wiley, New York (1992)
9. Dubois, D., Prade, H.: A possibilistic analysis of inconsistency. In: Scalable Uncertainty Management - 9th International Conference, SUM 2015, Québec City, QC, Canada, September 16-18, 2015. Proceedings. pp. 347–353 (2015), http://dx.doi.org/10.1007/978-3-319-23540-0_23
10. Gabbay, D.M., Giordano, L., Martelli, A., Olivetti, N.: Hypothetical updates, priority and inconsistency in a logic programming language. In: Logic Programming and Nonmonotonic Reasoning, Third International Conference, LPNMR'95, Lexington, KY, USA, June 26-28, 1995, Proceedings. pp. 203–216 (1995)
11. Gebser, M., Schaub, T.: Modeling and language extensions. AI Magazine: Special Issue on Answer set Programming to appear Fall 2016 (2016)
12. Gebser, M., Schaub, T., Thiele, S., Veber, P.: Detecting inconsistencies in large biological networks with answer set programming. Theory and Practice of Logic Programming 11(2-3), 323–360 (2011)
13. Gelfond, M., Leone, N.: Logic programming and knowledge representation – the A-Prolog perspective. Artificial Intelligence 138(1–2), 3–38 (2002)
14. 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
15. Grant, J., Hunter, A.: Measuring Inconsistency in Knowledgebases. Journal of Intelligent Information Systems 27, 159–184 (2006)
16. Grant, J., Hunter, A.: Analysing Inconsistent First-Order Knowledgebases. Artificial Intelligence 172(8-9), 1064–1093 (May 2008)
17. Grant, J., Hunter, A.: Analysing inconsistent information using distance-based measures. International Journal of Approximate Reasoning In press (2016)
18. Hansson, S.O.: A Textbook of Belief Dynamics. Kluwer Academic Publishers, Norwell, MA, USA (2001)

19. 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)
20. Hunter, A., Konieczny, S.: Measuring inconsistency through minimal inconsistent sets. In: Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2008). pp. 358–366. AAAI Press (2008)
21. Hunter, A., Konieczny, S.: On the measure of conflicts: Shapley inconsistency values. Artificial Intelligence 174(14), 1007–1026 (July 2010)
22. Konieczny, S., Lang, J., Marquis, P.: Reasoning under inconsistency: the forgotten connective. In: Proceedings of IJCAI-2005. pp. 484–489 (2005)
23. Konieczny, S., Perez, R.P.: Logic based merging. Journal of Philosophical Logic 40, 239–270 (2011)
24. Lang, J., Marquis, P.: Reasoning under inconsistency: A forgetting-based approach. Artificial Intelligence 174(12–13), 799–823 (July 2010)
25. 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)
26. Madrid, N., Ojeda-Aciego, M.: Measuring instability in normal residuated logic programs: adding information. In: Proceedings of the 19th IEEE International Conference on Fuzzy Systems (2010)
27. Madrid, N., Ojeda-Aciego, M.: Measuring inconsistency in fuzzy answer set semantics. IEEE Transactions on Fuzzy Systems 19(4), 605–622 (2011)
28. Moore, R.C.: Autoepistemic logic revisited. Artif. Intell. 59(1-2), 27–30 (1993), http://dx.doi.org/10.1016/0004-3702(93)90165-8
29. Mu, K., Wang, K., Wen, L.: Approaches to Measuring Inconsistency for Stratified Knowledge Bases. International Journal of Approximate Reasoning 55, 529–556 (2014)
30. Potyka, N.: Linear Programs for Measuring Inconsistency in Probabilistic Logics. In: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14). pp. 568–577 (2014)
31. Reiter, R.: A logic for default reasoning. Artif. Intell. 13(1-2), 81–132 (1980), http://dx.doi.org/10.1016/0004-3702(80)90014-4
32. 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)
33. Thimm, M.: Inconsistency Measures for Probabilistic Logics. Artificial Intelligence 197, 1–24 (2013)
34. Thimm, M.: On the expressivity of inconsistency measures. Artificial Intelligence 234, 120–151 (2016)
35. Zhou, L., Huang, H., Qi, G., Ma, Y., Huang, Z., Qu, Y.: Measuring Inconsistency in DL-Lite Ontologies. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'09). pp. 349–356. IEEE Computer Society, Washington, DC, USA (2009)