Strong Inconsistency in Nonmonotonic Reasoning

Gerhard Brewka\textsuperscript{1} and Matthias Thimm\textsuperscript{2} and Markus Ulbricht\textsuperscript{1}
\textsuperscript{1}Department of Computer Science, Leipzig University, Germany
{brewka.mulbricht}@informatik.uni-leipzig.de
\textsuperscript{2}Institute for Web Science and Technologies (WeST), University of Koblenz-Landau, Germany
thimm@uni-koblenz.de

Abstract

Minimal inconsistent subsets of knowledge bases play an important role in classical logics, most notably for repair and inconsistency measurement. It turns out that for nonmonotonic reasoning a stronger notion is needed. In this paper we develop such a notion, called strong inconsistency. We show that—in an arbitrary logic, monotonic or not—minimal strongly inconsistent subsets play the same role as minimal inconsistent subsets in classical reasoning. In particular, we show that the well-known classical duality between hitting sets of minimal inconsistent subsets and maximal consistent subsets generalises to arbitrary logics if the strong notion of inconsistency is used. We investigate the complexity of various related reasoning problems and present a generic algorithm for computing minimal strongly inconsistent subsets of a knowledge base. We also demonstrate the potential of our new notion for applications, focusing on repair and inconsistency measurement.

1 Introduction

Various notions which are highly useful and thus have been studied intensively in classical logic turn out to be of rather limited value when it comes to nonmonotonic reasoning based on formalisms like Reiter’s default logic [Reiter, 1980], answer set programming (ASP) [Brewka et al., 2011], or abstract argumentation [Dung, 1995]. An excellent example is the notion of equivalence. In classical logic equivalence is important as it guarantees substitutability: whenever two formulas $F$ and $F'$ are equivalent, that is, possess the same models, and $F$ is a subformula of $G$, then replacing $F$ by $F'$ in $G$ yields a formula equivalent to $G$. In nonmonotonic formalisms this is no longer the case. For instance the two logic programs $P_1 = \{c\}$ and $P_2 = \{c \leftarrow \text{not } b\}$ have the same (single) stable model $\{c\}$, but substitutability is not given.\textsuperscript{1} For example, replacing the first rule in $P_3 = \{c \leftarrow \text{not } b, \text{b}\}$ with the fact “c.” changes the semantics of $P_3$. This observation has led to a body of literature on so-called strong equivalence, a more adequate notion of equivalence for non-monotonic reasoning (see for instance [Lifschitz et al., 2001; Eiter et al., 2005; Oikarinen and Woltran, 2011]).

In this paper we study another notion, namely the notion of minimal inconsistent subsets. Again, this notion is highly interesting for classical, or more generally monotonic logics, at least for the following reasons:

- Diagnosis and repair of knowledge bases: consistency of an inconsistent knowledge base $\mathcal{K}$ can be restored by computing a minimal hitting set of the minimal inconsistent subsets of $\mathcal{K}$ and eliminating the elements of the hitting set. The result will be a maximal consistent subset of $\mathcal{K}$ [Reiter, 1987].

- Inconsistency measures: various prominent numerical measures of the degree of inconsistency of a knowledge base exist in the literature which depend on the number of minimal inconsistent subsets, see for instance [Hunter and Konieczny, 2008].

Minimal inconsistent subsets cannot play the same role for nonmonotonic formalisms. Consider the logic program $P_4 = \{a \leftarrow \text{not } a, \text{not } b, \text{b}\}$. The program is consistent and has the stable model $\{b\}$. However, it has an inconsistent subset, namely $\{a \leftarrow \text{not } a, \text{not } b\}$ which is also a minimal inconsistent subset. Yet, since $P_4$ is consistent there is no inconsistency to be resolved. Note also that one of the standard assumptions in the literature on inconsistency measurement is that the inconsistency value of a knowledge base $\mathcal{K}$ should be 0 iff $\mathcal{K}$ is consistent. The example thus also shows that the notion of minimal inconsistent subsets is of no use in defining inconsistency measures for nonmonotonic formalisms.

The goal of this paper is to develop a stronger notion of inconsistency. We will introduce so-called strongly inconsistent subsets, which generalise the classical notion adequately to the nonmonotonic case, and study the minimal ones among these sets. In particular, we show that our objects of study can indeed play the same role for nonmonotonic reasoning as regular minimal inconsistent subsets for monotonic reasoning.

The paper is organised as follows: since our main results are independent of the actual logic used, we first present in Section 2 an abstract notion of logics which is based on a similar account in the area of multi-context systems [Brewka and Eiter, 2007]. Section 3 then introduces strong inconsistency, proves a generalised duality result between strongly inconsistent and maximal consistent sets, and investigates further properties of our new notion. Computational aspects of

---

\textsuperscript{1}A brief introduction to logic programs is provided in Section 2.
strong inconsistency, including a complexity analysis and a
generic algorithm for computing strongly inconsistent sub-
sets, are studied in Section 4. Section 5 discusses applica-
tions of our new notion in diagnosis and inconsistency mea-
surement. Section 6 concludes.

Proofs of technical results can be found in an extended ver-
sion of this paper2.

2 Preliminaries
We first describe what we mean by a—potentially
nonmonotonic—logic in an abstract manner, following the
characterisation of logics in [Brewka and Eiter, 2007]. Here a
logic is specified by a set KB of knowledge bases, a set BS of
belief sets, and an acceptability function ACC : KB → 2BS.
The analysis in this paper assumes that knowledge bases are
sets of formulas. Moreover, the distinction between consist-
et and inconsistent belief sets is crucial. For this reason we
extend Brewka and Eiter’s characterisation as follows:

Definition 2.1. A logic \( L \) is a tuple \( L = (WF, BS, INC, ACC) \) where \( WF \) is a set of well-formed
formulas, \( BS \) is a set of belief sets, \( INC \subseteq BS \) is an upward closed1 set of inconsistent belief sets, and \( ACC : 2^{WF} \rightarrow 2^{BS} \)
assigns a collection of belief sets to each subset of \( WF \). A
knowledge base \( K \) of \( L \) is a finite subset of \( WF \). A knowledge
base \( K \) is called inconsistent iff \( ACC(K) \subseteq INC \).

Note that a knowledge base \( K \) can be inconsistent because it
has no belief sets, and consistent even if some (but not all)
of its belief sets are in INC. We illustrate the generality of
the above definition by giving instantiations for propositional
logic, answer set programming [Brewka et al., 2011], and abstract argumentation frameworks [Dung, 1995].

Example 2.2. Let \( A \) be a set of propositional atoms. A propositional logic \( L_P \) can be defined as \( L_P = (WF_P, BS_P, INC_P, ACC_P) \) where \( WF_P \) are the well-formed
formulas over \( A \), \( BS_P \) are the deductively closed sets of for-
mulas, \( INC_P \) has \( WF_P \) as its single element, and \( ACC_P \)
assigns to each \( K \subseteq WF_P \) the set containing its set of theorems.

Example 2.3. Let \( A \) be a set of propositional atoms. Ex-
tended logic programs under answer set semantics over \( A \) is the logic \( L_{ASP} = (WF_{ASP}, BS_{ASP}, INC_{ASP}, ACC_{ASP}) \) where \( WF_{ASP} \) is the set of all rules over \( A \). \( BS_{ASP} \) consists of the
sets of literals over \( A \). \( INC_{ASP} \) are the belief sets containing a
complementary pair of literals, and \( ACC_{ASP} \) assigns to a logic
program \( P \subseteq WF_{ASP} \) the set of all answer sets of \( P \).

Example 2.4. Abstract argumentation frameworks under
e.g. stable semantics [Dung, 1995] can be modelled as a logic
\( L_{AAF} = (WF_{AAF}, BS_{AAF}, INC_{AAF}, ACC_{AAF}) \) in the follow-
ing way: given some set of abstract arguments \( Arg \), the ele-
ments of \( WF_{AAF} \) are either elements of \( Arg \), or pairs of such
elements, called attacks, i.e., \( WF_{AAF} = Arg \cup (Arg \times Arg) \).
The former are needed to represent arguments which do not
participate in any attack. Belief sets are arbitrary sets of argu-
ments, \( INC_{AAF} \) is empty, and \( ACC_{AAF} \) assigns to each knowl-
edge base \( AF \subseteq WF_{AAF} \) the stable extensions of the argument-

---

2 http://mthimm.de/misc/btu_ijcai17_ext.pdf
3 S upward closed means \( B \subseteq S \) and \( B \subseteq B' \) implies \( B' \subseteq S \).

As the above examples show, the abstract notion is general
enough to model monotonic and nonmonotonic logics.

Definition 2.5. A logic \( L = (WF, BS, INC, ACC) \) is weakly
monotonic whenever \( K \subseteq K' \subseteq WF \) implies
1. if \( B' \in ACC(K') \) then \( B \subseteq B' \) for some \( B \in ACC(K) \).
2. if \( B \in ACC(K) \) then \( B \subseteq B' \) for some \( B' \in ACC(K') \).

Note that this definition generalises [Brewka and Eiter, 2007] where in addition \( ACC \) is required to be unique
for monotonic logics. The two conditions are needed to guar-
antee that both skeptical and credulous inference based on
intersection, respectively union of belief sets are monotonic.

It is easy to see that—as expected—propositional logic is
monotonic whereas logic programs under the answer set sem-
antics and abstract argumentation frameworks are not.

Lemma 2.6. Let \( L = (WF, BS, INC, ACC) \) be weakly
monotonic and \( K \subseteq K' \). If \( K \) is inconsistent then so is \( K' \).

We will call a knowledge base (weakly) monotonic when-
ever its associated logic is. Also, whenever there is no risk
of confusion we will leave the actual logic implicit.

Throughout the paper we use examples based on logic pro-
grams with two kinds of negation, classical negation \( \neg \) and
default negation \( not \), under answer set semantics. Such pro-
grams consist of rules of the form

\[
l_0 \leftarrow l_1, \ldots, l_m, \text{ not } l_{m+1}, \ldots, \text{ not } l_n.
\]

where the \( l_i \) are classical literals. For programs without
default negation (\( m = n \)) the unique answer set is the smallest
set of literals closed under all rules, where a set is closed un-
der a rule of form (1) iff the head literal \( l_0 \) is in the set when-
ever the body literals \( l_1, \ldots, l_m \) are. For a program \( P \) with
default negation, a set \( M \) of literals is an answer set iff \( M \)
is the answer set of the reduced program \( P^M \) obtained from \( P \)
by (i) deleting rules with not \( l_j \) in the body for some \( l_j \in M \), and
(ii) deleting default negated literals from the remaining
rules. \( P \) is inconsistent iff all of its answer sets contain a
complementary pair of literals. This includes the case where
\( P \) has no answer sets at all. Readers are referred to [Brewka
et al., 2011] for more details.

3 Strong Inconsistency
Let \( L = (WF, BS, INC, ACC) \) be a logic and \( K \subseteq WF \) a
knowledge base of \( L \). We will leave \( L \) implicit in the rest
of this section. We use \( I(K) \) to denote the collection of all
inconsistent subsets of \( K \). A set \( H \in I(K) \) is called minimal
inconsistent if \( H' \subseteq H \) implies \( H' \) is consistent. \( I_{min}(K) \) is
the set of all minimal inconsistent subsets of \( K \).

A consistent subset \( H \) of \( K \) is called maximal \( K \)-consistent
if \( H \subseteq H' \subseteq K \) implies \( H' \) is inconsistent. We let \( C(K) \) and
\( C_{max}(K) \) denote the set of all consistent and maximal
\( K \)-consistent subsets of \( K \), respectively.

In weakly monotonic logics, whenever a knowledge base \( K \)
is inconsistent, then so is \( K' \) for any knowledge base \( K \subseteq K' \).
This follows directly from weak monotonicity and upward-
closedness of the inconsistent belief sets. In nonmonotonic
logics this property does not hold in general as additional
information may resolve inconsistency.
Another property of monotonic logics is a specific duality between minimal inconsistent and maximal consistent sets. For that we need the following notion:

**Definition 3.1.** Let \( M \) be a set of sets. We call \( H \) a hitting set of \( M \) if \( H \cap M \neq \emptyset \) for each \( M \in M \). A hitting set \( H \) of \( M \) is a minimal hitting set of \( M \) if \( H' \subseteq H \) implies \( H' \) is not a hitting set of \( M \).

In the monotonic case, we have the following duality result (see [Reiter, 1987] for a proof of the first-order case in the setting of diagnosis).

**Theorem 3.2** (MinHS duality). Let \( K \) be a weakly monotonic knowledge base. Then, \( H \) is a minimal hitting set of \( I_{\min}(K) \) if and only if \( K \setminus H \in C_{\max}(K) \).

For nonmonotonic logics, this is not true anymore because a consistent knowledge base may contain inconsistent subsets. In order to generalise Theorem 3.2 to the nonmonotonic case, we need a different, stronger notion of inconsistency. Here is the definition of the central part of this paper:

**Definition 3.3.** For \( H, K \subseteq WF \) with \( H \subseteq K \), \( H \) is called strongly \( K \)-inconsistent if \( H \subseteq H' \subseteq K \) implies \( H' \) is inconsistent. The set \( H \) is called strongly inconsistent if it is strongly WF-inconsistent.

In other words, a subset of a knowledge base \( K \) is strongly \( K \)-inconsistent if all its supersets within the knowledge base \( K \) are inconsistent as well. Note that this is a generalisation of standard inconsistency: both notions coincide in the monotonic case (cf. Proposition 3.5 below).

**Definition 3.4.** For \( H, K \subseteq WF \) with \( H \subseteq K \), \( H \) is minimal strongly \( K \)-inconsistent if \( H \) is strongly \( K \)-inconsistent and \( H' \subseteq H \) implies \( H' \) is inconsistent or \( H \neq \emptyset \) is not strongly \( K \)-inconsistent.

Let \( SI(K) \) denote the set of all strongly \( K \)-inconsistent subsets of \( K \) and let \( SI_{\min}(K) \) denote the set of all minimal strongly \( K \)-inconsistent subsets of \( K \). The following results are immediate and easy to show:

**Proposition 3.5.** Let \( K \) be a knowledge base.
1. If \( K \) is weakly monotonic, then \( I(K) = SI(K) \).
2. If \( K \) is weakly monotonic, then \( I_{\min}(K) = SI_{\min}(K) \).
3. \( \neg K \) is inconsistent if and only if \( SI(K) \neq \emptyset \) if \( K \in SI(K) \).
4. If \( H \) is strongly \( K \)-inconsistent and \( H \subseteq K' \subseteq K \), then \( H \) is strongly \( K' \)-inconsistent.

We are now in a position to present one of our main results.

**Theorem 3.6** (Generalised MinHS duality). Let \( K \) be a knowledge base. Then, \( H \) is a minimal hitting set of \( SI_{\min}(K) \) if and only if \( K \setminus H \in C_{\max}(K) \).

**Example 3.7.** Consider again the logic program \( P_4 = \{ a \leftarrow not \ a, not \ b \} \). From before, the single and (thus also minimal) inconsistent subset is \( H = \{ a \leftarrow not \ a, not \ b \} \). Since it contains one rule only, it coincides with the minimal hitting set. \( P_4 \setminus H \) is consistent, but not maximal consistent as \( P_4 \) itself is consistent. Using strong inconsistency leads to the intended result: \( H \) is inconsistent, but not strongly \( P_4 \)-inconsistent. In fact, \( SI_{\min}(P_4) \) is empty, and so is the minimal hitting set. The single maximal consistent subset is \( P_4 \), as stated in Theorem 3.6.

Theorem 3.6 suggests that strong inconsistency is indeed an adequate notion that allows us to generalise results from monotonic logics to arbitrary ones. Even though fixing one part of \( K \) by removing a formula could potentially render another part of \( K \) inconsistent, we can restore consistency as in the monotonic case, using the notion of strong inconsistency.

In the literature on classical inconsistency handling the notion of free formulas plays a special role [Hunter and Konieczny, 2008] since such a formula has no influence whatsoever regarding consistency. We will next investigate a similar notion for our general setting.

**Definition 3.8.** Let \( K \) be a monotonic knowledge base. A formula \( \alpha \in K \) is called free if

\[
\alpha \in K \setminus \bigcup_{H \in I_{\min}(K)} H = \bigcap_{H \in C_{\max}(K)} H. \tag{2}
\]

Due to Theorem 3.6, an equality similar to the one in (2) also holds in nonmonotonic logics.

**Corollary 3.9.** Let \( K \) be a knowledge base. Then

\[
K \setminus \bigcup_{H \in SI_{\min}(K)} H = \bigcap_{H \in C_{\max}(K)} H. \tag{3}
\]

Consider a monotonic knowledge base \( K \). Since a free formula \( \alpha \in K \) is contained in any maximal consistent set \( H \subseteq K \), we see that for a free formula \( \alpha \) the following implication holds.

\[
\forall H \subseteq K : H \in C(K) \Rightarrow H \cup \{ \alpha \} \in C(K) \tag{4}
\]

However, in a nonmonotonic framework, (3) does not necessarily mean that \( \alpha \) is irrelevant regarding consistency of \( K \). For example, \( K \) could be consistent while \( K \setminus \{ \alpha \} \) is not. Hence, in order to obtain a similar notion of free formulas we need to strengthen (3).

**Definition 3.10.** Let \( K \) be a knowledge base. A formula \( \alpha \in K \) is called neutral if it satisfies

\[
\forall H \subseteq K : H \in C(K) \Leftrightarrow H \cup \{ \alpha \} \in C(K). \tag{5}
\]

A formula \( \alpha \in K \) is called consistency restoring if it satisfies (3) and (5), but not (4). The neutral and the consistency restoring formulas in \( K \) are denoted \( Ntr(K) \) and \( Res(K) \), respectively.

**Proposition 3.11.** If \( K \) is weakly monotonic, then (3) and (4) coincide and hence, \( Ntr(K) = Free(K) \) and \( Res(K) = \emptyset \).

**Proposition 3.12.** Let \( K \) be a knowledge base. Then

\[
Ntr(K) \cup Res(K) \subseteq \bigcap_{H \in C_{\max}(K)} H. \tag{6}
\]

In the Introduction we mentioned equivalence as another notion that has been strengthened for nonmonotonic logics. We conclude this section with a connection between strong equivalence and strong inconsistency. Strong equivalence, mainly studied in logic programming and argumentation, can be generalised to arbitrary logics in the following way: let \( L = \{ WF, BS, INC, ACC \} \) be a logic. The knowledge bases \( K \) and \( K' \) are strongly equivalent iff \( ACC(K \cup H) = ACC(K' \cup H) \) for each \( H \subseteq WF \). We obtain the following:

**Proposition 3.13.** Let \( K \) and \( K' \) be knowledge bases. If \( K \) and \( K' \) are strongly equivalent, then \( K \) is strongly \( K \cup H \)-inconsistent if \( K' \) is strongly \( K' \cup H \)-inconsistent.

The result does not hold if equivalence is used rather than strong equivalence.
4 Computational Complexity

Theorem 3.6 suggests that strong $\mathcal{K}$-inconsistency (Definition 3.3) naturally generalizes inconsistency to nonmonotonic frameworks. However, this notion requires consideration of all supersets of a given set, which is apparently more involved than considering inconsistency in monotonic logics. So, we are interested in the computational complexity of deciding (minimal) strong $\mathcal{K}$-inconsistency and in particular the difference between monotonic and nonmonotonic logics.

We assume the reader to be familiar with the classes $\Sigma^p_m$ and $\Pi^p_m$, $m \geq 0$, of the polynomial hierarchy. We also make use of the classes $\mathcal{D}^p_m$, which are the classes of languages that are intersections of a language in $\Sigma^p_m$ and a language in $\Pi^p_m$ [Papadimitriou, 1994].

In order to assess the computational complexity of deciding (minimal) strong inconsistency, we compare it to the classical setting: in [Papadimitriou and Wolfe, 1988], it has been shown that Minimal Unsatisfiability (MU) is $\mathcal{D}^p$-complete.

MU is the following problem: “Given a propositional formula $\phi$ in CNF, is it true that it is unsatisfiable, but removing an arbitrary clause renders it satisfiable?” Our first observation in this section is a generalisation of this result to higher levels of the polynomial hierarchy.

A quantified Boolean formula (QBF) $\Phi$ is a formula

$$\Phi = Q_1X_1 \ldots Q_mX_m \phi$$

with quantifiers $Q_1, \ldots, Q_m \in \{\forall, \exists\}$, pair-wise disjoint sets of variables $X_1, \ldots, X_m$, and a propositional formula $\phi$ over the variables $X_1 \cup \ldots \cup X_m$. A QBF $\Phi$ is true if $\phi$ evaluates to true with respect to the quantifiers, e.g., $\forall x_1 \exists x_2 (x_1 \lor \neg x_2)$ is true as for every truth value of $x_1$ one can find a truth value of $x_2$ such that $x_1 \lor \neg x_2$ evaluates to true. A QBF $\Phi$ is in prenex normal form if the quantifiers $Q_1, \ldots, Q_m$ alternate between $\forall$ and $\exists$. The problem of deciding whether a QBF $\Phi$ with $m$ alternating quantifiers starting with $\exists$ (resp. starting with $\forall$) is true is the canonical $\Sigma^p_m$-complete (resp. $\Pi^p_m$-complete) problem [Papadimitriou, 1994].

Let now QBF-MU$(Q_1, \ldots, Q_m)$ be the following problem:

Given a QBF $\Phi = Q_1X_1 \ldots Q_mX_m \phi$ in prenex normal form with $\phi = C_1 \land \ldots \land C_r$ and formulas $C_1, \ldots, C_r$, is it true that $\phi$ is false, but removing any conjunct $C_k$ from $\phi$ renders $\phi$ true?

For this problem, we obtain a similar result as [Papadimitriou and Wolfe, 1988], which, to our knowledge, has not been stated explicitly before.

**Theorem 4.1.** If $m \geq 2$, then QBF-MU$(Q_1, \ldots, Q_m)$ is $\mathcal{D}^p_m$-complete.

Combining Theorem 4.1 and the result in [Papadimitriou and Wolfe, 1988] (i.e., the case where $m = 1$ and $Q_1 = \exists$), one can observe that the case where $\Phi$ is of the form $\Phi = \forall X \phi$ is missing. Indeed, it turns out to be easier.

**Proposition 4.2.** QBF-MU$(\forall)$ is $\text{NP}$-complete.

**Remark 4.3.** We can cast the logic of quantified Boolean formulas into our general logical framework as well. Given quantifiers $Q_1, \ldots, Q_m$ and sets of variables $X_1, \ldots, X_m$, we define a corresponding logic $L = L(Q_1, \ldots, Q_m, X_1, \ldots, X_m) = (\text{WF}, \text{BS}, \text{INC}, \text{ACC})$ as follows: $\text{WF} = \text{WF}(X_1 \cup \ldots \cup X_m)$ is the set of all well-formed Boolean formulas over the atoms in $X_1 \cup \ldots \cup X_m$. $\text{BS} = \{\bot, \top\}$, $\text{INC} = \{\bot\}$ and for a knowledge base $\mathcal{K} = \{C_1, \ldots, C_r\}$ we let $\phi(\mathcal{K}) = C_1 \land \ldots \land C_r$ and define $\text{ACC} = \text{ACC}(Q_1, \ldots, Q_m)$ via

$$\text{ACC}(\mathcal{K}) = \begin{cases} \{\bot\}, & \text{if } Q_1X_1 \ldots Q_mX_m \phi(\mathcal{K}), \\ \{\bot, \top\}, & \text{otherwise}. \end{cases}$$

Now, deciding whether $\Phi$ is a “yes” instance of QBF-MU$(Q_1, \ldots, Q_m)$ corresponds to checking whether $\mathcal{K}$ is minimal (strongly) inconsistent.

We now turn to the general discussion on the computational complexity of problems related to strong inconsistency. For that, we assume an arbitrary but fixed logic $L = (\text{WF}, \text{BS}, \text{INC}, \text{ACC})$ for the remainder of this section. To be able to assess how difficult it is to check whether a subset $H \subseteq \mathcal{K}$ of a knowledge base $\mathcal{K}$ is (minimal) strongly $\mathcal{K}$-inconsistent, we consider checking satisfiability of $\mathcal{K}$ as the basis of our investigation. More precisely, we consider the following decision problems:

- **SAT$_\mathcal{K}$**
  - **Input**: $\mathcal{K} \subseteq \text{WF}$
  - **Output**: TRUE iff $\mathcal{K}$ is consistent

- **S-INC$_\mathcal{K}(H)$**
  - **Input**: $\mathcal{K} \subseteq \text{WF}, H \subseteq \mathcal{K}$
  - **Output**: TRUE iff $H \in SI(\mathcal{K})$

- **MIN-S-INC$_\mathcal{K}(H)$**
  - **Input**: $\mathcal{K} \subseteq \text{WF}, H \subseteq \mathcal{K}$
  - **Output**: TRUE iff $H \in SI_{\min}(\mathcal{K})$

In other words, SAT$_\mathcal{K}$ is the generalisation of the satisfiability problem in our general logic $L$. S-INC$_\mathcal{K}(H)$ is about deciding whether $H$ is strongly $\mathcal{K}$-inconsistent, and MIN-S-INC$_\mathcal{K}(H)$ is about deciding whether $H$ is a minimal strongly $\mathcal{K}$-inconsistent set. If $L$ is monotonic and SAT$_\mathcal{K} \in \mathcal{C}$ for some class $\mathcal{C}$, then S-INC$_\mathcal{K}(H)$ is in co-$\mathcal{C}$. However, in a nonmonotonic framework, checking whether a given subset $H \subseteq \mathcal{K}$ is strongly $\mathcal{K}$-inconsistent involves considering all sets $H'$ with $H \subseteq H' \subseteq \mathcal{K}$ and corresponding consistency checks. This may increase computational complexity in some cases, but, interestingly, not always as the following result shows.

**Theorem 4.4.** Let $\mathcal{K}$ be a knowledge base. Let $m \geq 1$. If the decision problem SAT$_\mathcal{K}$ is in

(a) $\Sigma^p_m$, then S-INC$_\mathcal{K}(H)$ is in $\Pi^p_m$.
(b) $\Pi^p_m$, then S-INC$_\mathcal{K}(H)$ is in $\Pi^p_{m+1}$.
(c) $\Pi^p_m$ and $\mathcal{K}$ is weakly monotonic, then S-INC$_\mathcal{K}(H)$ is in $\Sigma^p_m$.

Theorem 4.1 already showed how difficult MIN-S-INC$_\mathcal{K}(H)$ is compared to the decision problem SAT$_\mathcal{K}$ in the generic framework of QBFs (cf. Remark 4.3). As stated in Theorem 4.4, checking strong inconsistency is in general more difficult in nonmonotonic frameworks and we obtain a similar result in the case of MIN-S-INC$_\mathcal{K}(H)$. However, the increase of the computational complexity stems from checking the “strong” part in “strong minimal inconsistency” rather than the “minimal” part. For that reason and as the following result shows, moving from the problem S-INC$_\mathcal{K}(H)$ to the problem MIN-S-INC$_\mathcal{K}(H)$—i.e., additionally asking for minimality—does not involve...
Theorem 4.5. Let $K$ be a knowledge base. Let $m \geq 1$. If the decision problem $\text{SAT}_{\text{K}}$ is in

(a) $\Sigma^p_m$, then $\text{MIN-S-INC}_K(H)$ is in $\mathcal{D}^p_m$,
(b) $\Pi^p_m$, then $\text{MIN-S-INC}_K(H)$ is in $\mathcal{D}^p_{m+1}$,
(c) $\Pi^p_m$ and $K$ is weakly monotonic, then $\text{MIN-S-INC}_K(H)$ is in $\mathcal{D}^p_m$.

In Theorems 4.4 and 4.5 only membership statements are given. Thus, they leave open whether $\text{S-INC}_K(H)$ and $\text{MIN-S-INC}_K(H)$ can be strictly less difficult than $\Pi^p_m$ and $\mathcal{D}^p_{m+1}$, respectively, if $K$ is nonmonotonic and $\text{SAT}_K$ in $\Pi^p_m$. However, we obtain completeness for a specific (artificial) nonmonotonic logic, showing that the bounds from both cases (b) from Theorems 4.4 and 4.5 are tight in general.

Theorem 4.6. There is a logic $L = (WF, BS, INC, ACC)$ such that $\text{SAT}_K$ is in $\text{coNP}$ if and

(a) $\text{S-INC}_K(H)$ is $\Pi^p_2$-complete and
(b) $\text{MIN-S-INC}_K(H)$ is $\mathcal{D}^p_2$-complete.

We give two more hardness results for case (a) of Theorem 4.5 for the framework of logic programming. Be reminded that deciding whether a given logic program $P$ is consistent is $\text{NP}$-complete [Eiter and Gottlob, 1995].

Theorem 4.7. The problem $\text{MIN-S-INC}_P(H)$ is $\mathcal{D}^p_2$-complete for logic programs.

As a second example, we consider disjunctive logic programs, i.e., logic programs with rules such as (1) but that may have disjunctions in the head rather than a single literal. Due to [Eiter and Gottlob, 1995], deciding whether a given disjunctive program is consistent is $\Sigma^p_2$-complete.

Theorem 4.8. The problem $\text{MIN-S-INC}_P(H)$ is $\mathcal{D}^p_2$-complete for disjunctive logic programs.

To conclude this discussion on computational complexity, we present a generic algorithm for computing $\text{SI}(K)$. Algorithm 1 computes strongly $K$-inconsistent subsets in the order of decreasing cardinality, starting with $K$. It is based on the observation that a proper subset $S$ of $K$ can only be strongly $K$-inconsistent if all subsets of $K$ which contain one additional element are also strongly $K$-inconsistent (this property is checked during the computation of $New$). This additional check presumably reduces the search space in many cases, but a detailed evaluation of this algorithm is left for future work. The algorithm is somewhat reminiscent of the Apriori algorithm for computing frequent sets in data mining [Agrawal and Srikant, 1994], but rather than working bottom up from smaller to bigger sets, it works in the opposite direction. The algorithm can easily be turned into one for $\text{SI}_{\text{min}}(K)$ by deleting non-minimal elements whenever $New$ is added to $H'$.

Proposition 4.9. Algorithm 1 is sound, complete, and has runtime $O(2^n \ast n \ast f(n))$ where $f(n)$ is the runtime of an algorithm for checking consistency in the given logic.

We expect that for specific logics one can do better. For instance, for logic programs without classical negation it is well-known that inconsistency can only arise if there are certain negative loops in the dependency graph. The analysis of such loops may lead to more direct algorithms. This topic is currently under investigation.

5 Applications

We will now discuss some of the potential applications of strong inconsistency in nonmonotonic reasoning, namely knowledge base repair and inconsistency measurement.

5.1 Diagnosis and Repair

Theorem 3.6 already shows how consistency of a knowledge base $K$ can be restored by deleting a minimal subset of formulas. As in the classical case, the key is to compute certain inconsistent subsets of the knowledge base. The hitting sets of these subsets then are the candidates for deletion. Unlike in monotonic logics, in the general case one has to compute hitting sets of minimal strongly inconsistent subsets. Our theorem shows that this guarantees minimality of the modification performed on $K$.

Example 5.1. Consider the following logic program $P_5$:

\begin{align*}
P_5 : & \quad a \leftarrow \neg b. \quad b \leftarrow \neg c. \quad a \leftarrow \neg d. \quad e. \\
& \quad c \leftarrow \neg a. \quad d. \quad \neg e.
\end{align*}

Minimal inconsistent subsets of $P_5$ are $H_1 = \{a \leftarrow \neg b, b \leftarrow \neg c, c \leftarrow \neg a\}$ and $H_2 = \{e, \neg e\}$. Whereas $H_3$ is also minimal strongly $P_5$-inconsistent, $H_1$ is not as adding “$a \leftarrow \neg d$” resolves inconsistency. The second minimal strongly $P_5$-inconsistent subset is $H_3 = H_1 \cup \{d\}$. Minimal hitting sets consist of one element of $H_3$ and one of $H_1$. The program can be repaired by deleting the rules in any of the hitting sets.

It is worth mentioning that in the nonmonotonic case, this is not the only way of repairing a knowledge base, as adding formulas may also lead to consistency. However, deletion-based repair is also important for nonmonotonic knowledge bases for various reasons. First of all, in many cases it is far from clear how to select and justify the added formulas. Second, there are situations where modelling errors are more
probable than modelling gaps, and where identifying such errors simply is the better option. And finally, there are cases where there is simply no choice as some inconsistencies cannot be repaired by additions alone.

The results of this paper are not only relevant for knowledge base repair, but also for model-based diagnosis of technical systems along the lines of [Reiter, 1987; de Kleer et al., 1992]. In this approach a system description SD is given in terms of first-order logic. SD describes the correct behaviour of a set of components Comp and uses ab predicates for this purpose. The idea is then to identify minimal sets of components C such that SD ∪ Obs ∪ {ab(c) | c ∈ C} is inconsistent. The results of this paper allow us to capture system descriptions expressed in more general logics. All we have to do is replace the inconsistency check with a strong inconsistency check. A more detailed analysis of this generalisation of model-based diagnosis is beyond the scope of this paper and will be discussed elsewhere.

5.2 Measuring Inconsistency
An inconsistency measure Inc is a function that maps knowledge bases to non-negative real numbers. The intuition behind such functions is that larger values indicate severe inconsistencies in the knowledge base and the value 0 indicates minimal inconsistency, i.e., consistency. Different approaches to measuring inconsistency have been proposed in the literature, mostly for classical propositional logic, see [Thimm, 2017] for a recent survey. In this context, a simple but popular approach to measure inconsistency is to take the number of minimal inconsistent subsets [Hunter and Konieczny, 2008], i.e., to define Incℳ(K) = |Imin(K)| for a classical knowledge base K. This measure already complies with some basic ideas of inconsistency measurement, in particular Incℳ(K) = 0 iff K is consistent. By also taking the size and the relationships of minimal inconsistent subsets into account, a wide variety of different inconsistency measures can be defined on top of that idea, see e.g. [Hunter and Konieczny, 2008; Jabbour et al., 2016; Jabbour and Sais, 2016].

Measuring inconsistency in nonmonotonic logics has only recently gained some attention [Ulbricht et al., 2016] and a thorough study is still needed. In this setting, a measure such as Incℳ is not applicable as a consistent nonmonotonic knowledge base K may contain minimal inconsistent subsets, recall the logic program P4 = \{a ← not a, not b, b} from the introduction. However, using our notion of strong inconsistency the wide spectrum of measures based on minimal inconsistent subsets can be lifted to the general case. Here, we only consider the measure Incℳ.

Definition 5.2. Define Incℳ(K) via Incℳ(K) = |SImin(K)| for every knowledge base K.

If K is monotonic then Incℳ(K) = Incℳ(K) due to Proposition 3.5, item 2. So the measure Incℳ faithfully generalises Incℳ to all kinds of logics.

Example 5.3. For the logic program P4 = \{a ← not a, not b, b\} we obtain Incℳ(P4) = 0, despite the fact that P1 contains a (classical) minimal inconsistent subset. For P5 = \{a, ¬a, b ← not b\} we have Incℳ(P5) = 2.

The field of inconsistency measurement is driven by rationality postulates, i.e., the development of general properties that should hold for an inconsistency measure, cf. [Thimm, 2017]. Many of them specify desirable behaviour in terms of minimal inconsistent subsets and can thus easily be lifted to the general case. The following result shows the compliance of our generalised measure with some important properties.

Theorem 5.4. Let K be a (monotonic or nonmonotonic) knowledge base.

Consistency Incℳ(K) = 0 if and only if K is consistent.

Independence If α ∈ Ntr(K) then Incℳ(K) = Incℳ(K \ \{α\}).

Separability If SImin(K1) ∪ SImin(K2) = SImin(K1) ∪ SImin(K2) and SImin(K1) ∩ SImin(K2) = ∅ then Incℳ(K1 ∪ K2) = Incℳ(K1) + Incℳ(K2).

The measure Incℳ violates one important property though, which is usually demanded for classical measures: the monotonicity postulate. This postulate requires Inc(K) ≤ Inc(K′) whenever K ⊆ K′ and formalises the intuition that inconsistency can only increase when adding new information. However, this intuition is inadequate for nonmonotonic logics as the addition of new information may resolve inconsistencies. Therefore, satisfaction of the monotonicity postulate is indeed not desirable in general, see [Ulbricht et al., 2016] for a discussion on this topic.

In the same vein, other approaches that utilise minimal inconsistent sets for inconsistency measurement [Hunter and Konieczny, 2008; Jabbour et al., 2016; Jabbour and Sais, 2016] can also be lifted to the general case. We leave a deeper investigation of this topic for future work.

6 Conclusions
In this paper we studied inconsistency in an abstract setting covering arbitrary logics, including nonmonotonic ones. We showed that in the general case the standard notion of inconsistency is unable to play the same role it does in monotonic reasoning. Our main contribution is the identification of an adequate strengthening of inconsistency. One of our main results shows that the duality between minimal inconsistent subsets and maximal consistent subsets of a knowledge base, which does not hold for nonmonotonic logics, can be restored when minimal strongly inconsistent subsets are used. We established encouraging complexity results for problems related to strong inconsistency, presented a generic algorithm for computing (minimal) strongly inconsistent subsets, and demonstrated possible applications of our new notion in diagnosis/repair and inconsistency measurement.

Although there is a rich literature on inconsistency handling (see [Bertossi et al., 2005] for an introduction and [Bienvenida et al., 2016] for a recent approach), we are not aware of any work addressing the issues we studied in this paper.

In future work we will investigate algorithms for specific nonmonotonic logics, elaborate the use of strong inconsistency in model-based diagnosis and continue the study of inconsistency measures based on strong inconsistency.

Acknowledgements
This work was partially funded by DFG (Research Training Group 1763; project BR 1817/7-2).
References


