Strong Inconsistency

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

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

Abstract

Minimal inconsistent subsets of knowledge bases play an important role in propositional logic, most notably for diagnosis 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 a similar role as minimal inconsistent subsets in propositional logic. In particular, we show that the well-known 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 diagnosis and inconsistency measurement.

Keywords: nonmonotonic reasoning, inconsistency handling, minimal inconsistent subsets, computational complexity

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) (Gelfond and Lifschitz, 1991; Gelfond and Leone, 2002; 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, but substitutability is not given.[1] For example, replacing the first rule in $P_3 = \{c \leftarrow \text{not } b., b.\}$ 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 nonmonotonic 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 $P_4 = \{a \leftarrow \text{not } a, \text{not } b., \ b.\}$. 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 nothing to repair at all. 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 the number 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.

[1] A formal introduction to logic programs is provided in Section 2.

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 strong inconsistency, including a complexity analysis and a generic algorithm for computing strongly inconsistent subsets, are studied in Section 4. Section 5 discusses applications of our new notion in diagnosis and inconsistency measurement. Section 6 concludes.

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 $\mathsf{ACC} : \mathsf{KB} \to 2^{\mathsf{BS}}$. The analysis in this paper assumes that knowledge bases are sets of formulas. Moreover, the distinction between consistent 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 $L = (\mathsf{WF}, \mathsf{BS}, \mathsf{INC}, \mathsf{ACC})$ where WF is a set of well-formed formulas, BS is a set of belief sets, $\mathsf{INC} \subseteq \mathsf{BS}$ is an upward closed[2] set of inconsistent belief sets, and $\mathsf{ACC} : 2^{\mathsf{WF}} \to 2^{\mathsf{BS}}$ assigns a collection of belief sets to each subset of WF. A *knowledge base* $\mathcal{K}$ of L is a finite subset of WF. A knowledge base $\mathcal{K}$ is called *inconsistent* iff $\mathsf{ACC}(\mathcal{K}) \subseteq \mathsf{INC}$.

Note that a knowledge base $\mathcal{K}$ 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 will illustrate the generality of the above definition by giving instantiations for several logics in Sections 2.1, 2.2, 2.3, and 2.4 below. However, first we make precise what we mean by a *nonmonotonic* logic.

Definition 2.2. A logic $L = (\mathsf{WF}, \mathsf{BS}, \mathsf{INC}, \mathsf{ACC})$ is *weakly monotonic* whenever $\mathcal{K} \subseteq \mathcal{K}' \subseteq \mathsf{WF}$ implies

1. if $B' \in \mathsf{ACC}(\mathcal{K}')$ then $B \subseteq B'$ for some $B \in \mathsf{ACC}(\mathcal{K})$.

[2] S upward closed means $B \in S$ and $B \subseteq B'$ implies $B' \in S$.

L is *monotonic* if $\mathcal{K} \subseteq \mathcal{K}' \subseteq \mathsf{WF}$ in addition implies

2. if $B \in \mathsf{ACC}(\mathcal{K})$ then $B \subseteq B'$ for some $B' \in \mathsf{ACC}(\mathcal{K}')$,

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 guarantee that both skeptical and credulous inference based on intersection, respectively union of belief sets are monotonic. One important aspect of (weakly) monotonic logics is that supersets of inconsistent knowledge bases are always inconsistent as well.

Lemma 2.3. *Let $L = (\mathsf{WF}, \mathsf{BS}, \mathsf{INC}, \mathsf{ACC})$ be weakly monotonic and $\mathcal{K} \subseteq \mathcal{K}'$. If $\mathcal{K}$ is inconsistent then so is $\mathcal{K}'$.*

Proof. Let $\mathcal{K}$ be inconsistent, i. e., $\mathsf{ACC}(\mathcal{K}) \subseteq \mathsf{INC}$. Let $\mathcal{K} \subseteq \mathcal{K}'$. If $B' \in \mathsf{ACC}(\mathcal{K}')$, then $B \subseteq B'$ for a $B \in \mathsf{ACC}(\mathcal{K})$. However, $B \in \mathsf{ACC}(\mathcal{K})$ implies $B \in \mathsf{INC}$ and since INC is upward closed, $B' \in \mathsf{INC}$. Thus, $\mathsf{ACC}(\mathcal{K}') \subseteq \mathsf{INC}$. □

We will call a knowledge base (weakly) monotonic whenever its associated logic is. This is a slight abuse of terminology since monotonicity is a property of a logic, not of a knowledge base. However, leaving the actual logic implicit does no harm in many cases, and we prefer the simpler terminology whenever there is no risk of confusion.

In the following subsections, we provide instantiations of Definition 2.1 for classical propositional logic, Priest's logic of Paradox (Priest, 1979), answer set programming (Gelfond and Lifschitz, 1991; Gelfond and Leone, 2002; Brewka et al., 2011), and abstract argumentation frameworks (Dung, 1995). We present these instantiations to show the generality of Definition 2.1 to model a wide spectrum of logics. Furthermore, throughout the paper we will use answer set programming (Section 2.3) to illustrate our ideas.

2.1. Propositional Logic

Let A be a set of propositional atoms (a propositional signature). Any atom $a \in A$ is a well-formed formula wrt. A. If ϕ and ψ are well-formed formulas wrt. A, then $\neg\phi$, $\phi \wedge \psi$, and $\phi \vee \psi$ are also well-formed formulas wrt. A (we also assume that the usual abbreviations $\Rightarrow, \Leftrightarrow$ are defined accordingly). Let $\mathsf{WF}^{\mathrm{P}}_A$ be the set of well-formed formulas wrt. A. Let $\models$ be the classical entailment relation, i. e. $\psi \models \phi$ if all models of ψ are also models of ϕ in the classical

propositional semantics. Then define $\mathsf{ACC}^{\mathrm{P}}_A(\mathcal{K})$ for every $\mathcal{K} \subseteq \mathsf{WF}^{\mathrm{P}}_A$ to be a singleton containing only the deductive closure of $\mathcal{K}$, i. e.

$$\mathsf{ACC}^{\mathrm{P}}_A(\mathcal{K}) = \{\{\phi \mid \mathcal{K} \models \phi\}\}$$

for all $\mathcal{K} \subseteq \mathsf{WF}^{\mathrm{P}}_A$. Furthermore, $\mathsf{BS}^{\mathrm{P}}_A$ are exactly the deductively closed sets, i. e.

$$\mathsf{BS}^{\mathrm{P}}_A = \{\mathcal{K} \subseteq \mathsf{WF}^{\mathrm{P}}_A \mid \mathcal{K} = \mathsf{ACC}^{\mathrm{P}}_A(\mathcal{K})\}$$

and $\mathsf{INC}^{\mathrm{P}}_A = \{\mathsf{WF}^{\mathrm{P}}_A\}$, i. e. the only inconsistent belief set is the set of all formulas. Then the propositional logic L^{P}_A over the signature A can be defined as

$$L^{\mathrm{P}}_A = (\mathsf{WF}^{\mathrm{P}}_A, \mathsf{BS}^{\mathrm{P}}_A, \mathsf{INC}^{\mathrm{P}}_A, \mathsf{ACC}^{\mathrm{P}}_A)$$

Observe that L^{P}_A is a monotonic logic. More precisely, if $\mathcal{K}, \mathcal{K}$ are two propositional knowledge bases with $\mathcal{K} \subseteq \mathcal{K}'$ then $\mathcal{K} \models \phi$ implies $\mathcal{K}' \models \phi$ and thus both conditions 1 and 2 of Definition 2.2 are satisfied (note that the two conditions are indeed equivalent as $\mathsf{ACC}^{\mathrm{P}}_A(\mathcal{K})$ is always a singleton set).

Example 2.4. Consider the propositional signature $A_1 = \{a, b\}$ and the knowledge bases $\mathcal{K}_1, \mathcal{K}_2$ given via

$$\mathcal{K}_1 = \{a, a \Rightarrow b\} \qquad\qquad \mathcal{K}_2 = \{a, \neg a\}$$

Then

$$\mathsf{ACC}^{\mathrm{P}}_{A_1}(\mathcal{K}_1) = \{\{a, a \Rightarrow b, b, a \vee a, b \wedge b, \ldots\}\}$$
$$\mathsf{ACC}^{\mathrm{P}}_{A_1}(\mathcal{K}_2) = \{\mathsf{WF}^{\mathrm{P}}_{A_1}\}$$

Observe that $\mathcal{K}_1$ is consistent while $\mathcal{K}_2$ is inconsistent due to $\mathsf{ACC}^{\mathrm{P}}_{A_1}(\mathcal{K}_2) \subseteq \mathsf{INC}^{\mathrm{P}}_{A_1} = \{\mathsf{WF}^{\mathrm{P}}_{A_1}\}$.

2.2. Priest's Logic of Paradox

Priest's logic of paradox (Priest, 1979) is a paraconsistent logic that uses the syntax of propositional logic. So again, let A be a set of propositional atoms. The set of well-formed formulas for the logic of paradox is the same as in propositional logic, so we have $\mathsf{WF}^{\mathrm{X}}_A = \mathsf{WF}^{\mathrm{P}}_A$.

We briefly recall the semantics of (Priest, 1979) to define the function $\mathsf{ACC}^{\mathrm{X}}_A$. A three-valued interpretation υ for a signature A is a function $\upsilon : A \to \{T, F, B\}$ which assigns one of three truth values to each atom. Here, T stands for "true",

F stands for "false", and B stands for "both" (the latter is a paraconsistent truth value). These truth values are strictly ordered by a relation $\prec_X$ via $F \prec_X B \prec_X T$. Then υ can be extended to arbitrary well-formed formulas via

$$\upsilon(\phi \wedge \psi) = \min_{\prec_X}\{\upsilon(\phi), \upsilon(\psi)\}$$
$$\upsilon(\phi \vee \psi) = \max_{\prec_X}\{\upsilon(\phi), \upsilon(\psi)\}$$
$$\upsilon(\neg\phi) = \begin{cases} F & \text{if } \upsilon(\phi) = T \\ T & \text{if } \upsilon(\phi) = F \\ B & \text{otherwise} \end{cases}$$

A three-valued interpretation υ is a model of a formula $\phi \in \mathsf{WF}^{\mathrm{X}}_A$ if and only if $\upsilon(\phi) \in \{T, B\}$ and υ is a model of a set $\{\phi_1, \ldots, \phi_n\} \subseteq \mathsf{WF}^{\mathrm{X}}_A$ if and only if it is a model of $\phi_1 \wedge \ldots \wedge \phi_n$. Then we can define $\mathsf{ACC}^{\mathrm{X}}_A$ via

$$\mathsf{ACC}^{\mathrm{X}}_A(\mathcal{K}) = \{\upsilon^{-1}(T) \mid \upsilon \text{ is a model of } \mathcal{K}\}$$

for all $\mathcal{K} \subseteq \mathsf{WF}^{\mathrm{X}}_A$ and accordingly $\mathsf{BS}^{\mathrm{X}}_A = 2^A$. Note that there are no inconsistent formulas in Priest's logic (the interpretation assigning B to every atom is always a model), so we have $\mathsf{INC}^{\mathrm{X}}_A = \emptyset$. This gives us the logic

$$L^{\mathrm{X}}_A = (\mathsf{WF}^{\mathrm{X}}_A, \mathsf{BS}^{\mathrm{X}}_A, \mathsf{INC}^{\mathrm{X}}_A, \mathsf{ACC}^{\mathrm{X}}_A)$$

Observe that L^{X}_A is monotonic. More precisely, observe that for $\mathcal{K} \subseteq \mathcal{K}'$ every model of $\mathcal{K}'$ is also a model of $\mathcal{K}$ and therefore $\mathsf{ACC}^{\mathrm{X}}_A(\mathcal{K}') \subseteq \mathsf{ACC}^{\mathrm{X}}_A(\mathcal{K})$ validating both conditions of Definition 2.2.

Example 2.5. Consider the propositional signature $A_1 = \{a, b\}$ and the knowledge bases $\mathcal{K}_1, \mathcal{K}_2$ given via

$$\mathcal{K}_1 = \{a, b\} \qquad\qquad \mathcal{K}_2 = \{a, b, \neg a \vee \neg b\}$$

Then

$$\mathsf{ACC}^{\mathrm{X}}_{A_1}(\mathcal{K}_1) = \{\{a, b\}, \{a\}, \{b\}, \emptyset\}$$
$$\mathsf{ACC}^{\mathrm{X}}_{A_1}(\mathcal{K}_2) = \{\{a\}, \{b\}, \emptyset\}$$

Observe that both $\mathcal{K}_1$ and $\mathcal{K}_2$ are consistent.

2.3. Answer Set Programming

We will now consider disjunctive logic programs under the answer set semantics, cf. (Gelfond and Lifschitz, 1991; Gelfond and Leone, 2002; Brewka et al., 2011). Let again A be a set of propositional atoms and let $Lit(A) = A \cup \{\neg a \mid a \in A\}$ be the set of corresponding literals. A disjunctive rule r is a rule of the form

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

with $l_0, \ldots, l_n \in Lit(A)$ and let $\mathsf{WF}_A^{\text{ASP}}$ be the set of all such rules. For a rule r we abbreviate $head(r) = \{l_0, \ldots, l_k\}$, $pos(r) = \{l_{k+1}, \ldots, l_m\}$, and $neg(r) = \{l_{m+1}, \ldots, l_n\}$. If $m = n = k$ then r is called a fact and written $head(r)$ instead of $head(r) \leftarrow$.

A set $P \subseteq \mathsf{WF}_A^{\text{ASP}}$ is also called a logic program for short. Furthermore, a belief set is any set of literals, i. e., $\mathsf{BS}_A^{\text{ASP}} = 2^{Lit(A)}$.

If $P \subseteq \mathsf{WF}_A^{\text{ASP}}$ is a logic program without default negation—i. e. for all $r \in P$ we have $neg(r) = \emptyset$—then a model M is any set $M \in \mathsf{BS}_A^{\text{ASP}}$ such that for all $r \in P$, if $pos(r) \subseteq M$ then $head(r) \cap M \neq \emptyset$. M is a minimal model if it is a model and there is not model M' with $M' \subsetneq M$. For a logic program $P \subseteq \mathsf{WF}_A^{\text{ASP}}$ with default negation, an answer set M is any set $M \in \mathsf{BS}_A^{\text{ASP}}$ such that M is a minimal model of the logic program without default negation P^M defined via

$$P^M = \{head(r) \leftarrow pos(r) \mid head(r) \leftarrow pos(r), neg(r) \in P, neg(r) \cap M = \emptyset\}.$$

Then we define $\mathsf{ACC}_A^{\text{ASP}}$ via

$$\mathsf{ACC}_A^{\text{ASP}}(P) = \{M \mid M \text{ is an answer set of } P\}$$

for all $P \subseteq \mathsf{WF}_A^{\text{ASP}}$. Finally, inconsistent belief sets are those sets containing complementary literals

$$\mathsf{INC}_A^{\text{ASP}} = \{M \in \mathsf{BS}_A^{\text{ASP}} \mid a, \neg a \in M \text{ for some } a \in A\}.$$

This gives us the logic

$$L_A^{\text{ASP}} = (\mathsf{WF}_A^{\text{ASP}}, \mathsf{BS}_A^{\text{ASP}}, \mathsf{INC}_A^{\text{ASP}}, \mathsf{ACC}_A^{\text{ASP}}).$$

Note that L_A^{ASP} is not monotonic as the following example shows.

Example 2.6. Consider the propositional signature $A_1 = \{a, b\}$ and the logic programs P_5, P_6 given via

$$P_5 = \{b., \neg b \leftarrow \text{not } a.\}$$
$$P_6 = \{a., b., \neg b \leftarrow \text{not } a.\}$$

Then

$$\mathsf{ACC}^{\text{ASP}}_{A_1}(P_5) = \{\{b, \neg b\}\}$$
$$\mathsf{ACC}^{\text{ASP}}_{A_1}(P_6) = \{\{a, b\}\}$$

Observe that P_5 is inconsistent while P_6 is consistent. As $P_5 \subseteq P_6$ this also shows that $L^{\text{ASP}}_{A_1}$ is not monotonic.

A rule of the form

$$r: \quad a \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n, \text{not } a. \tag{2}$$

where a is an atom that does not occur elsewhere in a given program P is called a *constraint*. The intuitive meaning is that no answer set of P is allowed to contain all literals $l_1, \ldots, l_m$ and none of the literals $l_{m+1}, \ldots, l_n$. We use the established shorthand

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

for constraints of the form (2).

Now let $\mathsf{WF}^{\text{ASP}_{k=0}}_A \subseteq \mathsf{WF}^{\text{ASP}}_A$ be the set of all rules of the form (1) with $k = 0$. Then,

$$L^{\text{ASP}_{k=0}}_A = (\mathsf{WF}^{\text{ASP}_{k=0}}_A, \mathsf{BS}^{\text{ASP}}_A, \mathsf{INC}^{\text{ASP}}_A, \mathsf{ACC}^{\text{ASP}}_A)$$

is the logic corresponding to disjunction-free logic programs under answer set semantics.

2.4. Abstract Argumentation Frameworks

Our final knowledge representation formalism to be considered as a concrete instantiation for Definition 2.1 are abstract argumentation frameworks (Dung, 1995). In the original formulation, an abstract argumentation framework AF is a directed graph $AF = (A, R)$ where nodes in A represent arguments and the relation R models "attack", i. e., for $a, b \in A$, if $(a, b) \in R$ then a is a counterargument

for b and we say that a attacks b. Abstract argumentation frameworks consider the problem of argumentation only at this abstract level and do not consider the inner structure of arguments nor how the attack relation is derived. Semantics are given to an abstract argumentation framework $AF = (A, R)$ by identifying sets $E \subseteq A$ of arguments (called extensions) that can be "jointly accepted". The literature offers various approaches on how to define "jointly accepted" (Dung, 1995), but we only consider stable semantics here. A set $E \subseteq A$ is called *stable extension* if there are no arguments $a, b \in E$ with $(a, b) \in R$ and for every $c \in A \setminus E$ there is $a \in A$ with $(a, c) \in R$. Note that an abstract argumentation framework may possess none, one, or multiple stable extensions.

In order to cast abstract argumentation frameworks into our logical framework, let A be some universal set of arguments (comparable to propositional signatures used before). A well-formed formula is then either an argument $a \in A$ (stating that argument a is in the graph) or a pair of arguments $(a, b) \in A \times A$ (stating that there are arguments a and b in the in the graph and that there is an attack from argument a to argument b), i. e., $\mathsf{WF}_A^{\mathrm{AAF}} = A \cup (A \times A)$. For $S = \{a_1, \ldots, a_n, (a_{n+1}, a_{n+2}), \ldots, (a_{m-1}, a_m)\} \subseteq \mathsf{WF}_A^{\mathrm{AAF}}$ we write $AF(S) = (\{a_1, \ldots, a_m\}, \{(a_{n+1}, a_{n+2}), \ldots, (a_{m-1}, a_m)\})$ to denote the directed graph represented by S. Belief sets are then arbitrary sets of arguments, i. e., $\mathsf{BS}_A^{\mathrm{AAF}} = 2^A$, $\mathsf{INC}_A^{\mathrm{AAF}} = \emptyset$, and $\mathsf{ACC}_A^{\mathrm{AAF}}$ is defined via

$$\mathsf{ACC}_A^{\mathrm{AAF}}(S) = \{E \subseteq A \mid E \text{ is a stable extension of } AF(S)\}$$

Observe that a $S \subseteq \mathsf{WF}_A^{\mathrm{AAF}}$ is considered inconsistent if $AF(S)$ does not have any stable extensions, i. e., $\mathsf{ACC}_A^{\mathrm{AAF}}(S) = \emptyset = \mathsf{INC}_A^{\mathrm{AAF}}$. This gives us the logic

$$L_A^{\mathrm{AAF}} = (\mathsf{WF}_A^{\mathrm{AAF}}, \mathsf{BS}_A^{\mathrm{AAF}}, \mathsf{INC}_A^{\mathrm{AAF}}, \mathsf{ACC}_A^{\mathrm{AAF}})$$

Note that L_A^{AAF} is not monotonic as the following example shows.

Example 2.7. Consider the two abstract argumentation frameworks AF_1 and AF_2 over the vocabulary $A_2 = \{a, b, c, d\}$ depicted in Figures 1 and 2, respectively. These frameworks can be modelled in our general logic as sets $S_1, S_2 \subseteq \mathsf{WF}_{A_2}^{\mathrm{AAF}}$ via

$$S_1 = \{(a, b), (b, a), (c, c), (c, b)\}$$
$$S_2 = \{(a, b), (b, a), (c, c), (c, b), (d, c)\}$$

More precisely, we have $AF(S_1) = AF_1$ and $AF(S_2) = AF_2$. Then

$$\mathsf{ACC}_{A_2}^{\mathrm{AAF}}(S_1) = \emptyset$$
$$\mathsf{ACC}_{A_2}^{\mathrm{AAF}}(S_2) = \{\{a, d\}, \{b, d\}\}$$

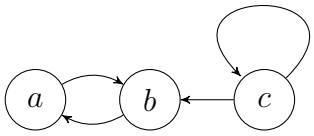

Figure 1: The argumentation framework AF_1 from Example 2.7

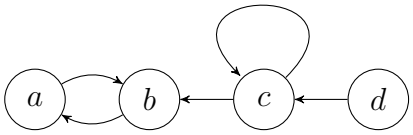

Figure 2: The argumentation framework AF_2 from Example 2.7

Observe that S_1 is inconsistent while S_2 is consistent. As $S_1 \subseteq S_2$ this also shows that $L^{\text{AAF}}_{A_2}$ is not monotonic.

3. Strong Inconsistency

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

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

3.1. Hitting Set Dualities

In this section, we prove one of the most central results of this paper. In (Reiter, 1987), Reiter points out a duality between maximal consistent and minimal inconsistent subsets of a knowledge base—the well-known hitting set duality. It is stated within the context of diagnosis and can easily be lifted to first-order logic or, more generally, any (weakly) monotonic framework. Our notion of strong inconsistency will facilitate a generalisation to *arbitrary* logics, which is the main motivation to consider this extension of ordinary inconsistency.

Of course, to generalise Reiter's hitting set duality, we need the notion of a hitting set.

Definition 3.1. Let $\mathcal{M}$ be a set of sets. We call $\mathcal{S}$ a *hitting set* of $\mathcal{M}$ if $\mathcal{S} \cap M \neq \emptyset$ for each $M \in \mathcal{M}$. A hitting set $\mathcal{S}$ of $\mathcal{M}$ is a *minimal hitting set* of $\mathcal{M}$ if $\mathcal{S}' \subsetneq \mathcal{S}$ implies $\mathcal{S}'$ is not a hitting set of $\mathcal{M}$.

In the monotonic case, we have the following duality result (see (Reiter, 1987)).

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

For nonmonotonic logics, this is not true anymore because a consistent knowledge base may contain inconsistent subsets. In order to obtain the desired generalisation of Theorem 3.2 to the nonmonotonic case, we consider an appropriate refinement of inconsistency.

Definition 3.3. For $H, \mathcal{K} \subseteq \mathsf{WF}$ with $H \subseteq \mathcal{K}$, H is called *strongly $\mathcal{K}$-inconsistent* if $H \subseteq H' \subseteq \mathcal{K}$ 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 $\mathcal{K}$ is strongly $\mathcal{K}$-inconsistent if all its supersets within the knowledge base $\mathcal{K}$ are inconsistent as well. Intuitively, one can think of a conflict within $\mathcal{K}$ that cannot be resolved by formulas in $\mathcal{K}$ itself. Due to Lemma 2.3 this condition is redundant in weakly monotonic logics and thus, mere and strong $\mathcal{K}$-inconsistency coincide (cf. Proposition 3.5 below).

Definition 3.4. For $H, \mathcal{K} \subseteq \mathsf{WF}$ with $H \subseteq \mathcal{K}$, H is *minimal strongly $\mathcal{K}$-inconsistent* if H is strongly $\mathcal{K}$-inconsistent and $H' \subsetneq H$ implies that H' is not strongly $\mathcal{K}$-inconsistent.

Let $SI(\mathcal{K})$ denote the set of all strongly $\mathcal{K}$-inconsistent subsets of $\mathcal{K}$ and let $SI_{min}(\mathcal{K})$ denote the set of all minimal strongly $\mathcal{K}$-inconsistent subsets of $\mathcal{K}$.

The following results are immediate, respectively easy to show:

Proposition 3.5. *Let $\mathcal{K}$ be a knowledge base.*

1. *If $\mathcal{K}$ is weakly monotonic, then $I(\mathcal{K}) = SI(\mathcal{K})$.*
2. *If $\mathcal{K}$ is weakly monotonic, then $I_{min}(\mathcal{K}) = SI_{min}(\mathcal{K})$.*
3. *$\mathcal{K}$ is inconsistent iff $SI(\mathcal{K}) \neq \emptyset$ iff $\mathcal{K} \in SI(\mathcal{K})$.*

4. *If H is strongly $\mathcal{K}$-inconsistent and $H \subseteq \mathcal{K}' \subseteq \mathcal{K}$, then H is strongly $\mathcal{K}'$-inconsistent.*

Now, we can generalise Theorem 3.2 to the nonmonotonic case.

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

Proof. "⇒": Let $\mathcal{S}$ be a minimal hitting set for $SI_{min}(\mathcal{K})$ and let $H = \mathcal{K} \setminus \mathcal{S}$. First, we show that any set H' with $H \subsetneq H' \subseteq \mathcal{K}$ is inconsistent. Such H' is of the form

$$H' = H \cup M \text{ with } \emptyset \neq M \subseteq \mathcal{K} \setminus H.$$

Hence, it holds that $\emptyset \neq M \subseteq \mathcal{S}$. Since $\mathcal{S}$ was assumed to be a minimal hitting set of $SI_{min}(\mathcal{K})$, $\mathcal{S} \setminus M$ is not a hitting set of $SI_{min}(\mathcal{K})$. Hence,

$$H' = H \cup M = (\mathcal{K} \setminus \mathcal{S}) \cup M = \mathcal{K} \setminus (\mathcal{S} \setminus M)$$

contains a strongly $\mathcal{K}$-inconsistent set. Thus, H' is inconsistent. So, we have

$$H \subsetneq H' \subseteq \mathcal{K} \Rightarrow H' \text{ is inconsistent.} \tag{3}$$

Now we show $H \in C(\mathcal{K})$. Then, maximality follows from (3). Let us assume H is inconsistent. Then, H is strongly $\mathcal{K}$-inconsistent due to (3). Hence, H contains a minimal strongly $\mathcal{K}$-inconsistent subset $H'' \subseteq H$. Due to

$$H'' \subseteq H = \mathcal{K} \setminus \mathcal{S}$$

we have $\mathcal{S} \cap H'' = \emptyset$ yielding a contradiction as $\mathcal{S}$ was assumed to be a hitting set of $SI_{min}(\mathcal{K})$. Hence, $H \in C(\mathcal{K})$. Together with (3), we obtain that $H \in C_{max}(\mathcal{K})$.

"⇐": Let $H \subseteq \mathcal{K}$ be a maximal consistent set. Let $H = \mathcal{K} \setminus \mathcal{S}$. If $\mathcal{S}$ is no hitting set of $SI_{min}(\mathcal{K})$, then we see as above that H contains a strongly $\mathcal{K}$-inconsistent set, yielding a contradiction. Hence, $\mathcal{S}$ is a hitting set of $SI_{min}(\mathcal{K})$. Now assume that there is a set $\mathcal{S}' \subsetneq \mathcal{S}$ that is a hitting set of $SI_{min}(\mathcal{K})$ as well. Assume w. l. o. g. that $\mathcal{S}'$ is minimal. Then, $\mathcal{K} \setminus \mathcal{S}' \in C(\mathcal{K})$ as already shown above. However, $\mathcal{S}' \subsetneq \mathcal{S}$ implies $\mathcal{K} \setminus \mathcal{S} \subsetneq \mathcal{K} \setminus \mathcal{S}'$. Hence, $\mathcal{K} \setminus \mathcal{S}$ is not maximal in $C(\mathcal{K})$, a contradiction. □

Theorem 3.6 suggests that strong inconsistency can indeed play a similar role in nonmonotonic frameworks as classical inconsistency does in monotonic ones. Even though restoring consistency in one part of $\mathcal{K}$ by removing a formula could potentially render another part of $\mathcal{K}$ inconsistent, we can resolve inconsistency as in the monotonic case, using the notion of strong inconsistency.

Example 3.7. We give three examples to illustrate the theorem in monotonic and nonmonotonic frameworks.

(a) Consider the propositional knowledge base $\mathcal{K} = \{a,\ a \Rightarrow b,\ \neg b,\ c,\ \neg c\}$. We have $SI_{min}(\mathcal{K}) = I_{min}(\mathcal{K}) = \{\{a,\ a \Rightarrow b,\ \neg b\},\ \{c,\ \neg c\}\}$. It is straightforward to see the 6 possible minimal hitting sets of $SI_{min}(\mathcal{K})$ and to verify that they correspond to the 6 maximal consistent subsets of $\mathcal{K}$.

(b) Consider again the logic program $P_4 = \{a \leftarrow \text{not } a, \text{not } b.,\ \ b.\}$ from before. The single (and thus also minimal) inconsistent subset is $H = \{a \leftarrow \text{not } a, \text{not } b.\}$. 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.

(c) Consider the slightly more involved program P_7 given as follows:

$$\begin{array}{lcc} P_7: & a \text{ or } b. & a \leftarrow b. \\ & c \leftarrow \text{not } b. & \neg c \leftarrow \text{not } b. \end{array}$$

Program P_7 includes an interesting nonmonotonic mechanism: Even though the disjunction "a or b." could actually resolve the conflict $\{c \leftarrow \text{not } b., \neg c \leftarrow \text{not } b.\}$, this is prevented by the rule "$a \leftarrow b$.": if a set of atoms contains b, then it also needs to contain a in order to be an answer set. However, $\{a,\ b\}$ is not a minimal model of the reduct $P_7^{\{a,\ b\}} = \{a \text{ or } b.,\ a \leftarrow b.\}$.

Thus, the maximal consistent subprograms of P_7 are obtained by either removing one of the rules $\{c \leftarrow \text{not } b.,\ \neg c \leftarrow \text{not } b.\}$ or "$a \leftarrow b$.". We see

$$\begin{aligned} C_{max}(P_7) = \{&\{a \text{ or } b.,\ a \leftarrow b.,\ c \leftarrow \text{not } b.\}, \\ &\{a \text{ or } b.,\ a \leftarrow b.,\ \neg c \leftarrow \text{not } b.\}, \\ &\{a \text{ or } b.,\ c \leftarrow \text{not } b.,\ \neg c \leftarrow \text{not } b.\}\}. \end{aligned}$$

We verify the duality by computing $SI_{min}(P_7)$. It is clear that any inconsistent subset of P_7 contains $\{c \leftarrow \text{not } b., \neg c \leftarrow \text{not } b.\}$. However, this is not a

strongly P_7-inconsistent subset since adding "a or b." restores consistency. Thus, we also need to add "$a \leftarrow b$.". Hence,

$$SI_{min}(P_7) = \{\{c \leftarrow \text{not } b.,\ \neg c \leftarrow \text{not } b.,\ a \leftarrow b.\}\}.$$

We see that the three minimal hitting sets of $SI_{min}(P_7)$ correspond to $C_{max}(P_7)$.

In (Birnbaum and Lozinskii, 2003), Theorem 4.5, item (d), a similar duality result is given. Here, the minimal inconsistent subsets of a knowledge base are considered rather than hitting sets of them. They are obtained by removing a minimal hitting set of $coC_{max}(\mathcal{K})$:

Definition 3.8. A set $\overline{M} \subseteq \mathcal{K}$ is in $coC_{max}(\mathcal{K})$ if there is a set $M \in C_{max}(\mathcal{K})$ such that $\overline{M} = \mathcal{K} \setminus M$.

The following result is given (see (Birnbaum and Lozinskii, 2003)):

Theorem 3.9. *Let $\mathcal{K}$ be a weakly monotonic knowledge base. Then, $\mathcal{S}$ is a minimal hitting set of $coC_{max}(\mathcal{K})$ if and only if $\mathcal{S} \in I_{min}(\mathcal{K})$.*

This Theorem can also be lifted to arbitrary logics. Again, we only need to replace I_{min} by SI_{min}.

Theorem 3.10. *Let $\mathcal{K}$ be a knowledge base. Then, $\mathcal{S}$ is a minimal hitting set of $coC_{max}(\mathcal{K})$ if and only if $\mathcal{S} \in SI_{min}(\mathcal{K})$.*

Proof. "$\Rightarrow$": Let $\mathcal{S}$ be a minimal hitting set of $coC_{max}(\mathcal{K})$. Assume for the sake of contradiction that $\mathcal{S}$ is not in $SI(\mathcal{K})$. Hence, there is a consistent set H with $\mathcal{S} \subseteq H$. We obtain

$$\mathcal{S} \cap (\mathcal{K} \setminus H) = \emptyset.$$

Since we can w. l. o. g. assume $H \in C_{max}(\mathcal{K})$, $\mathcal{S}$ is not a hitting set of $coC_{max}(\mathcal{K})$. This is a contradiction.

Now assume $\mathcal{S}$ is strongly $\mathcal{K}$-inconsistent, but not minimal. In this case, there is a set $H \subsetneq \mathcal{S}$ with $H \in SI_{min}(\mathcal{K})$. We show that H is a hitting set of $coC_{max}(\mathcal{K})$ as well, implying that $\mathcal{S}$ is not minimal: If H is not a hitting set of $coC_{max}(\mathcal{K})$, then there is a maximal consistent set $M \in C_{max}(\mathcal{K})$ with $H \cap (\mathcal{K} \setminus M) = \emptyset$ and thus, $H \subseteq M$ which contradicts the assumption that H is strongly $\mathcal{K}$-inconsistent.

"$\Leftarrow$": Now let $\mathcal{S} \in SI_{min}(\mathcal{K})$. We just argued that $\mathcal{S}$ is a hitting set of $coC_{max}(\mathcal{K})$, since otherwise, $\mathcal{S}$ would not be strongly $\mathcal{K}$-inconsistent. We have left to show that $\mathcal{S}$ minimal. So assume $H \subsetneq \mathcal{S}$ is a hitting set of $coC_{max}(\mathcal{K})$. However, as already argued, $H \in SI(\mathcal{K})$ in this case. Hence, $\mathcal{S}$ is not in $SI_{min}(\mathcal{K})$, a contradiction. □

Example 3.11. Consider the program P_7

$$\begin{array}{lll} P_7: & a \text{ or } b. & a \leftarrow b. \\ & c \leftarrow \text{not } b. & \neg c \leftarrow \text{not } b. \end{array}$$

again. As already pointed out, we have

$$\begin{array}{ll} C_{max}(P_7): & coC_{max}(P_7): \\ \{a \text{ or } b.,\ a \leftarrow b.,\ c \leftarrow \text{not } b.\} & \{\neg c \leftarrow \text{not } b.\} \\ \{a \text{ or } b.,\ a \leftarrow b.,\ \neg c \leftarrow \text{not } b. & \{c \leftarrow \text{not } b.\} \\ \{a \text{ or } b.,\ c \leftarrow \text{not } b.,\ \neg c \leftarrow \text{not } b.\} & \{a \leftarrow b.\} \end{array}$$

Indeed, the unique minimal hitting set of $coC_{max}(P_7)$ is the only set in

$$SI_{min}(P_7) = \{\{c \leftarrow \text{not } b.,\ \neg c \leftarrow \text{not } b.,\ a \leftarrow b.\}\}.$$

3.2. On the Notion of Free Formulas

In the literature on classical inconsistency handling the notion of *free formulas* plays a special role (Hunter and Konieczny, 2008). There, a free formula is any formula which is not contained in any minimal inconsistent subset. As it turns out, there are at least two contemplable ways to generalise this notion to nonmonotonic logics. We are going to present them and investigate their properties.

Consider a classical logic. Since the minimal inconsistent subsets of a knowledge base $\mathcal{K}$ are interpreted as the "raw" conflicts in $\mathcal{K}$, a formula which is not a member of any of them should not play a role regarding consistency.

Definition 3.12. Let $\mathcal{K}$ be a weakly monotonic knowledge base. A formula $\alpha \in \mathcal{K}$ is called *free* if

$$\alpha \in \mathcal{K} \setminus \bigcup_{H \in I_{min}(\mathcal{K})} H. \tag{4}$$

Let $Free(\mathcal{K})$ be the set of all free formulas of $\mathcal{K}$.

Another way to define free formulas is requiring that α is contained in all maximal consistent subsets of $\mathcal{K}$. This is equivalent since

$$\mathcal{K} \setminus \bigcup_{H \in I_{min}(\mathcal{K})} H = \bigcap_{H \in C_{max}(\mathcal{K})} H \tag{5}$$

holds if $\mathcal{K}$ is (weakly) monotonic. However, (5) follows directly from Reiter's hitting set duality. We thus obtain a similar result:

Corollary 3.13. *Let $\mathcal{K}$ be a knowledge base. Then*

$$\mathcal{K} \setminus \bigcup_{H \in SI_{min}(\mathcal{K})} H = \bigcap_{H \in C_{max}(\mathcal{K})} H.$$

Proof. "$\subseteq$": Let $\alpha \in \mathcal{K} \setminus \bigcup_{H \in SI_{min}(\mathcal{K})} H$. Hence, α does not occur in any minimal hitting set of $SI_{min}(\mathcal{K})$ and thus, due to Theorem 3.6, it occurs in all maximal consistent sets $M \in C_{max}(\mathcal{K})$.
"$\supseteq$": Now assume $\alpha \notin \mathcal{K} \setminus \bigcup_{H \in SI_{min}(\mathcal{K})} H$, i. e., $\alpha \in H$ for a minimal strongly $\mathcal{K}$-inconsistent set $H \in SI_{min}(\mathcal{K})$. Hence, $H \setminus \{\alpha\} \notin SI_{min}(\mathcal{K})$ and thus, there is a maximal consistent set H' with $H \setminus \{\alpha\} \subseteq H'$. Now, $\alpha \notin H'$ because otherwise, H' would contain a strongly $\mathcal{K}$-inconsistent set. It follows that $\alpha \notin \bigcap_{H \in C_{max}(\mathcal{K})} H$. □

Corollary 3.13 motivates a very natural generalisation of free formulas, induced by replacing "I_{min}" with "SI_{min}":

Definition 3.14. Let $\mathcal{K}$ be a knowledge base. A formula $\alpha \in \mathcal{K}$ is called *free with respect to strong inconsistency* (or *SI-free* or simply *free* if there is no risk of confusion) if

$$\alpha \in \mathcal{K} \setminus \bigcup_{H \in SI_{min}(\mathcal{K})} H$$

We let $Free_{SI}(\mathcal{K})$ denote the set of all SI-free formulas of $\mathcal{K}$.

Observe that $Free(\mathcal{K})$ and $Free_{SI}(\mathcal{K})$ coincide in the monotonic case.

Proposition 3.15. *Let $\mathcal{K}$ be a weakly monotonic knowledge base. Then, $Free(\mathcal{K}) = Free_{SI}(\mathcal{K})$.*

Proof. Due to Proposition 3.5, Item 2, we have $I_{min}(\mathcal{K}) = SI_{min}(\mathcal{K})$. In particular,

$$Free(\mathcal{K}) = \mathcal{K} \setminus \bigcup_{H \in I_{min}(\mathcal{K})} H = \mathcal{K} \setminus \bigcup_{H \in SI_{min}(\mathcal{K})} H = Free_{SI}(\mathcal{K}).$$

□

We name some equivalent properties in order for α to be in $Free_{SI}$.

Proposition 3.16. *Let $\mathcal{K}$ be a knowledge base. Let $\alpha \in \mathcal{K}$. The following are equivalent:*

1. *α is free, i. e.,*

$$\forall H \subseteq \mathcal{K} : \; H \notin SI_{min}(\mathcal{K}) \Rightarrow H \cup \{\alpha\} \notin SI_{min}(\mathcal{K}), \tag{6}$$

2. *α does not introduce strong $\mathcal{K}$-inconsistency in general, i. e.,*

$$\forall H \subseteq \mathcal{K} : \; H \notin SI(\mathcal{K}) \Rightarrow H \cup \{\alpha\} \notin SI(\mathcal{K}), \tag{7}$$

3. *α is contained in every maximal consistent subset, i. e.,*

$$\forall H \subseteq \mathcal{K} : \; H \in C_{max}(\mathcal{K}) \Rightarrow \alpha \in H.$$

Proof. "1. $\Leftrightarrow$ 3." holds due to Corollary 3.13. Furthermore, 2. trivially implies 1. We show "1. $\Rightarrow$ 2.": Assume (7) is wrong, i. e., there is a set $H \notin SI(\mathcal{K})$ with $H \cup \{\alpha\} \in SI(\mathcal{K})$. $H \cup \{\alpha\}$ contains a minimal $\mathcal{K}$-inconsistent set H'. Observe that $\alpha \in H'$, because otherwise, H' has a consistent superset as it is the case for H. Since H' is minimal, it holds that $H' \setminus \{\alpha\} \notin SI(\mathcal{K})$, but $H' \in SI(\mathcal{K})$ and thus, α does not satisfy (6). □

Example 3.17. Consider program P_7 from above again:

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

As already discussed,

$$SI_{min}(P_7) = \{c \leftarrow \text{not } b.,\ \neg c \leftarrow \text{not } b.,\ a \leftarrow b.\}.$$

We obtain

$$Free_{SI}(P_7) = \{a \text{ or } b.\}.$$

Furthermore, recall

$$\begin{aligned} C_{max}(P_7) = \{&\{a \text{ or } b.,\ a \leftarrow b.,\ c \leftarrow \text{not } b.\},\\ &\{a \text{ or } b.,\ a \leftarrow b.,\ \neg c \leftarrow \text{not } b.\},\\ &\{a \text{ or } b.,\ c \leftarrow \text{not } b.,\ \neg c \leftarrow \text{not } b.\}\}. \end{aligned}$$

Since "a or b." is the only formula appearing in all maximal consistent sets, we obtain

$$\bigcap_{H \in C_{max}(P_7)} H = \{a \text{ or } b.\} = \mathcal{K} \setminus \bigcup_{H \in SI_{min}(P_7)} H$$

as stated in Corollary 3.13.

We will now turn to another generalisation of free formulas, namely *neutral formulas*. Even though considering the set

$$Free_{SI}(\mathcal{K}) = \bigcap_{H \in C_{max}(\mathcal{K})} H$$

seems quite natural, it has some undesired properties in nonmonotonic frameworks. The following example shows that $Free_{SI}(\mathcal{K})$ is not necessarily consistent, making it a debatable notion of "free formulas".

Example 3.18. Consider the logic program

$$P_8 : \qquad a. \qquad \neg a. \qquad \leftarrow \text{not } a, \text{not } \neg a.$$

One can verify that

$$SI_{min}(P) = \{\{a.,\ \neg a.\}\}.$$

Hence,

$$Free_{SI}(\mathcal{K}) = \{\leftarrow \text{not } a, \text{not } \neg a.\}$$

is inconsistent.

The reason why such a situation may occur is that we consider inconsistency only within the context of all subsets of a given knowledge base $\mathcal{K}$.

Instead of defining free formulas in a way that they "do not introduce strong $\mathcal{K}$-inconsistency" one could also formalise that they should be "irrelevant regarding (in)consistency". So consider a (weakly) monotonic knowledge base $\mathcal{K}$. One can see that α is free if

$$\forall H \subseteq \mathcal{K} :\ H \in C(\mathcal{K}) \Rightarrow H \cup \{\alpha\} \in C(\mathcal{K}). \tag{8}$$

In a nonmonotonic framework, (8) does not necessarily mean that α is irrelevant regarding consistency of $\mathcal{K}$, because α could *restore* consistency. Hence, for our notion, we explicitly require an equivalence.

Definition 3.19. Let $\mathcal{K}$ be a knowledge base. A formula $\alpha \in \mathcal{K}$ is called *neutral* if it satisfies

$$\forall H \subseteq \mathcal{K} :\ H \in C(\mathcal{K}) \Leftrightarrow H \cup \{\alpha\} \in C(\mathcal{K}). \tag{9}$$

A formula $\alpha \in \mathcal{K}$ is called *consistency restoring* if it satisfies (8), but not (9). The neutral and the consistency restoring formulas in $\mathcal{K}$ are denoted $Ntr(\mathcal{K})$ and $Res(\mathcal{K})$, respectively.

Of course, in the monotonic case, we do not need to explicitly require the equivalence in (9) and hence, Free and Ntr coincide.

Proposition 3.20. *If $\mathcal{K}$ is weakly monotonic, then* (8) *and* (9) *coincide and hence, $\mathit{Ntr}(\mathcal{K}) = \mathit{Free}(\mathcal{K})$ and $\mathit{Res}(\mathcal{K}) = \emptyset$.*

Proof. That is clear by definition. □

Note that in general, this is a stronger notion than Free_{SI}.

Proposition 3.21. *Let $\mathcal{K}$ be a knowledge base. Then*

$$\mathit{Ntr}(\mathcal{K}) \cup \mathit{Res}(\mathcal{K}) \subseteq \mathit{Free}_{SI}(\mathcal{K}).$$

Proof. Let $\alpha \in \mathit{Ntr}(\mathcal{K}) \cup \mathit{Res}(\mathcal{K})$. Due to (8), it can be added to any set $H \subseteq \mathcal{K}$ without introducing inconsistency. Hence, $\alpha \in \bigcap_{H \in C_{max}(\mathcal{K})} H = \mathit{Free}_{SI}(\mathcal{K})$. □

In contrast to SI-free formulas, the neutral ones do not make use of strong $\mathcal{K}$-inconsistency. Even though the hitting set duality from Theorem 3.6 suggests to utilize this notion, the neutral formulas are still quite well-behaving. The reason is that neutral formulas do not depend as much on the structure of the rest of the knowledge base. For example, here we have (cf. Example 3.18):

Proposition 3.22. *If L is a logic such that $\emptyset$ is consistent and $\mathcal{K}$ a knowledge base of L, then $\mathit{Ntr}(\mathcal{K})$ is consistent.*

Proof. That is clear. □

The neutral formulas also do not influence the structure of $\mathit{SI}_{min}(\mathcal{K})$.

Proposition 3.23. *Let $\mathcal{K}$ be a knowledge base and let $\alpha \in \mathit{Ntr}(\mathcal{K})$. Then,*

$$\mathit{SI}_{min}(\mathcal{K}) = \mathit{SI}_{min}(\mathcal{K} \setminus \{\alpha\}).$$

Proof. For any set $H \subseteq \mathcal{K} \setminus \{\alpha\}$ it holds that $H \in \mathit{SI}(\mathcal{K})$ if and only if $H \in \mathit{SI}(\mathcal{K} \setminus \{\alpha\})$, because α is neutral. Hence, the claim follows if we can prove that no set $H \in \mathit{SI}_{min}(\mathcal{K})$ contains α. This, however, is clear: H being strongly $\mathcal{K}$-inconsistent means any H' with $H \subseteq H' \subseteq \mathcal{K}$ is inconsistent as well. Since α is neutral, this is also the case if we replace H' with $H' \setminus \{\alpha\}$. Hence, $H \setminus \{\alpha\}$ is strongly $\mathcal{K}$-inconsistent. So, if $\alpha \in H$, then H cannot be minimal in $\mathit{SI}(\mathcal{K})$. □

It is easy to see that a formula $\alpha \in \mathit{Free}_{SI}(\mathcal{K})$ does not need to have this property. For example, if $\mathcal{K}$ is minimal strongly $\mathcal{K}$-inconsistent itself, but $\mathcal{K} \cup \{\alpha\}$ is consistent.

We can, however, strengthen the notion of free formulas in order to obtain a similar result. Consider

$$\forall H \subseteq \mathcal{K} : \; H \notin \mathit{SI}(\mathcal{K}) \Rightarrow H \cup \{\alpha\} \notin \mathit{SI}(\mathcal{K}) \tag{7}$$

again. Replacing "$\Rightarrow$" with "$\Leftrightarrow$"—as we have done it in (9)—obviously does not make sense, because due to the use of strong $\mathcal{K}$-inconsistency, this would be equivalent. To obtain a notion ensuring that α does not "resolve strong $\mathcal{K}$-inconsistency", we need to consider $\mathcal{K}$ *without* the formula α. Otherwise, α inherently never resolves strong $\mathcal{K}$-inconsistency.

Definition 3.24. Let $\mathcal{K}$ be a knowledge base. A formula $\alpha \in \mathcal{K}$ is called *nonparticipating* if

$$\forall H \subseteq \mathcal{K} : \; H \in \mathit{SI}(\mathcal{K} \setminus \{\alpha\}) \Rightarrow H \in \mathit{SI}(\mathcal{K}).$$

We denote the set of all nonparticipating formulas of $\mathcal{K}$ by $NP(\mathcal{K})$

Example 3.25. Consider the program P_4 again:

$$P_4 : \; a \leftarrow \text{not } a, \text{not } b. \qquad b.$$

The rule "$b.$" is not in $NP(P_4)$, because it restores consistency by satisfying the constraint "$a \leftarrow \text{not } a, \text{not } b.$": We have

$$P_4 \setminus \{b.\} = \{a \leftarrow \text{not } a, \text{not } b.\}$$

and thus for $H = \{a \leftarrow \text{not } a, \text{not } b.\}$ it holds that

$$H \in \mathit{SI}(P_4 \setminus \{b.\}), \qquad\qquad H \notin \mathit{SI}(P_4).$$

Now, we can characterise when addition or removal of α does not change $\mathit{SI}_{min}(\mathcal{K})$.

Proposition 3.26. *Let $\mathcal{K}$ be a knowledge base. Let $\alpha \in \mathcal{K}$. Then,*

$$\mathit{SI}_{min}(\mathcal{K}) = \mathit{SI}_{min}(\mathcal{K} \setminus \{\alpha\}) \qquad \Leftrightarrow \qquad \alpha \in \mathit{Free}_{SI}(\mathcal{K}) \cap NP(\mathcal{K}).$$

Proof. "⇒":
We show $\alpha \in \mathit{Free}_{SI}(\mathcal{K})$: Assume $\alpha \notin \mathit{Free}_{SI}(\mathcal{K})$. Then there is a set $H \subseteq \mathcal{K}$ with

$$H \notin SI(\mathcal{K}), \qquad H \cup \{\alpha\} \in SI(\mathcal{K}).$$

Note that $H \cup \{\alpha\}$ contains a minimal strongly $\mathcal{K}$-inconsistent set $H' \in SI_{min}(\mathcal{K})$. Since H is not strongly $\mathcal{K}$-inconsistent itself, $\alpha \in H'$. This, however, implies

$$H' \not\subseteq \mathcal{K} \setminus \{\alpha\}$$

and in particular,

$$H' \notin SI_{min}(\mathcal{K} \setminus \{\alpha\}).$$

We obtain $SI_{min}(\mathcal{K}) \neq SI_{min}(\mathcal{K} \setminus \{\alpha\})$.
We show $\alpha \in NP(\mathcal{K})$: Assume $\alpha \notin NP(\mathcal{K})$. Consider

$$H \in SI(\mathcal{K} \setminus \{\alpha\})$$

with

$$H \notin SI(\mathcal{K}).$$

Consider a consistent set H^* with

$$H \subseteq H^* \subseteq \mathcal{K}.$$

Since H is strongly $\mathcal{K} \setminus \{\alpha\}$-inconsistent, $\alpha \in H^*$. Moreover, let $H' \subseteq H$ be minimal, i. e.,

$$H' \in SI_{min}(\mathcal{K} \setminus \{\alpha\}).$$

Due to $\alpha \in H^*$, it holds that $H' \cup \{\alpha\} \subseteq H^*$ and hence $H' \cup \{\alpha\} \notin SI_{min}(\mathcal{K})$. Thus we obtain $SI_{min}(\mathcal{K}) \neq SI_{min}(\mathcal{K} \setminus \{\alpha\})$ again.

"⇐": We show

$$SI_{min}(\mathcal{K}) = SI_{min}(\mathcal{K} \setminus \{\alpha\}).$$

"⊆": Let $H \in SI_{min}(\mathcal{K} \setminus \{\alpha\})$. In particular,

$$H \in SI(\mathcal{K} \setminus \{\alpha\}).$$

Now, $\alpha \in NP(\mathcal{K})$ implies $H \in SI(\mathcal{K})$. Thus, we have left to show minimality of H: Assume $H' \subsetneq H$ exists with $H' \in SI_{min}(\mathcal{K})$. Due to $H' \subseteq H$ and $H \in SI(\mathcal{K} \setminus \{\alpha\})$, we have $\alpha \notin H'$. Hence, $H' \in SI(\mathcal{K} \setminus \{\alpha\})$ as well, which is a contradiction since H was assumed to be in $SI_{min}(\mathcal{K} \setminus \{\alpha\})$.
"⊇": If $H \in SI_{min}(\mathcal{K})$, then $\alpha \notin H$ since $\alpha \in \mathit{Free}_{SI}(\mathcal{K})$. Hence $H \in SI_{min}(\mathcal{K} \setminus \{\alpha\})$ as well. □

Example 3.27. Let us consider the previous example again:

$$P_4 : \ a \leftarrow \text{not } a, \text{not } b. \qquad b.$$

We already pointed out that $b. \notin NP(P_4)$ and in fact, removing it changes the structure of $SI_{min}(P_4)$. We have

$$SI_{min}(P_4) = \emptyset$$

as P_4 itself is consistent, but

$$SI_{min}(P_4 \setminus \{b.\}) = \{a \leftarrow \text{not } a, \text{not } b.\}$$

because without "$b.$", the constraint is not satisfied anymore. In contrast, "$a \leftarrow$ not a, not $b.$" is clearly free, which is easy to see because P_4 is consistent. Moreover, one can see that $a \leftarrow \text{not } a, \text{not } b. \in NP(P_4)$ as well. So, the constraint is in $Free_{SI}(P_4) \cap NP(P_4)$. In fact,

$$SI_{min}(P_4) = \emptyset = SI_{min}(P_4 \setminus \{a \leftarrow \text{not } a, \text{not } b.\})$$

as stated in Proposition 3.26.

Recall that in Proposition 3.23 we pointed out that

$$SI_{min}(\mathcal{K}) = SI_{min}(\mathcal{K} \setminus \{\alpha\}) \tag{10}$$

holds for $\alpha \in Ntr(\mathcal{K})$. However, we did not state an equivalence. In fact, we found the equivalence of (10) and $\alpha \in Free_{SI}(\mathcal{K}) \cap NP(\mathcal{K})$ in Proposition 3.26. Thus, we have

$$\alpha \in Ntr(\mathcal{K}) \quad \Rightarrow \quad SI_{min}(\mathcal{K}) = SI_{min}(\mathcal{K} \setminus \{\alpha\}) \quad \Leftrightarrow \quad \alpha \in Free_{SI}(\mathcal{K}) \cap NP(\mathcal{K})$$

which raises the question whether the first implication is an equivalence and thus, $Ntr(\mathcal{K}) = Free_{SI}(\mathcal{K}) \cap NP(\mathcal{K})$ holds. This, however, is *not* the case as the following example shows.

Example 3.28. We consider P_9—a slightly different program than P_4—given as follows:

$$P_9 : \ \leftarrow \text{not } a, \text{not } b. \qquad a. \qquad b.$$

We see $a. \in NP(P_9)$, because due to "$b.$" the constraint is satisfied even without "$a.$". However, "$a.$" is not neutral, because

$$\{a., \leftarrow \text{not } a, \text{not } b.\} \in C(P_9), \qquad \{\leftarrow \text{not } a, \text{not } b.\} \notin C(P_9).$$

In order to finish our discussion on different notions of free formulas, we make one more quite simple observation. Recall from Definition 3.19 that $\alpha \in \mathcal{K}$ is called consistency restoring if it satisfies

$$\forall H \subseteq \mathcal{K} : \ H \in C(\mathcal{K}) \Rightarrow H \cup \{\alpha\} \in C(\mathcal{K}) \tag{8}$$

but not

$$\forall H \subseteq \mathcal{K} : \ H \in C(\mathcal{K}) \Leftrightarrow H \cup \{\alpha\} \in C(\mathcal{K}). \tag{9}$$

In addition to the observation from Proposition 3.21 the following example shows that $\alpha \in Res(\mathcal{K})$ can be either be in $NP(\mathcal{K})$ or not.

Example 3.29. Consider the programs

$$\begin{array}{lllll} P_4: & \leftarrow \text{not } a, \text{not } b. & b. & & \\ P_9: & \leftarrow \text{not } a, \text{not } b. & b. & a. & \end{array}$$

We have $b. \in Res(P_4)$ and $b. \in Res(P_9)$, but $b. \notin NP(P_4)$, while $b. \in NP(P_9)$

This finishes our discussion on the structure of a knowledge base $\mathcal{K}$ with respect to the four subsets we introduced:

- $Free_{SI}(\mathcal{K})$,
- $Ntr(\mathcal{K})$,
- $Res(\mathcal{K})$,
- $NP(\mathcal{K})$.

Figure 3 depicts the relationship between the sets we considered.

3.3. Strong Inconsistency and Strong Equivalence

In the Introduction, we mentioned equivalence as another notion that has been strengthened for nonmonotonic logics. We consider now 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:

Definition 3.30. Let $L = (\mathsf{WF}, \mathsf{BS}, \mathsf{INC}, \mathsf{ACC})$ be a logic. The knowledge bases $\mathcal{K}$ and $\mathcal{K}'$ are *strongly equivalent* iff $\mathsf{ACC}(\mathcal{K} \cup H) = \mathsf{ACC}(\mathcal{K}' \cup H)$ for each $H \subseteq \mathsf{WF}$.

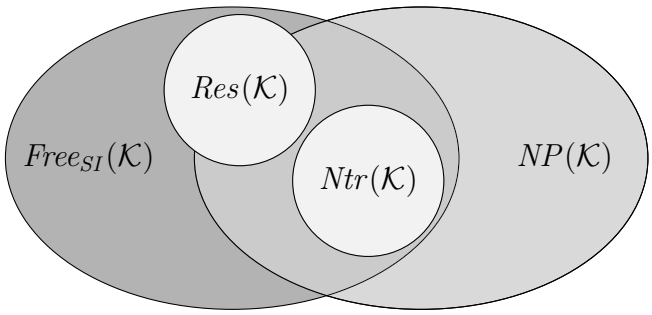

Figure 3: Venn Diagram of the considered subsets of $\mathcal{K}$

Example 3.31. As expected, the programs P_{10} and P_{11} given as

$$\begin{array}{rr} P_{10}: & b \leftarrow \text{not } a. \\ & \neg\, b \leftarrow \text{not } a. \end{array} \qquad\qquad \begin{array}{rr} P_{11}: & b. \\ & \neg\, b. \end{array}$$

are not strongly equivalent, even though they have the same answer set. Moreover, if we view them as subprograms of P_{12} given via

$$\begin{array}{rr} P_{12}: & b \leftarrow \text{not } a. \\ & \neg\, b \leftarrow \text{not } a. \end{array} \qquad\qquad \begin{array}{r} b. \\ \neg\, b. \end{array} \qquad\qquad a.$$

then P_{11} is strongly P_{12}-inconsistent while P_{10} is not.

The previous example contained two knowledge bases that are *not* strongly equivalent. However, as the following proposition shows, strong inconsistency is compatible with strong equivalence.

Proposition 3.32. *Let $\mathcal{K}$, $\mathcal{K}'$ and H be knowledge bases. If $\mathcal{K}$ and $\mathcal{K}'$ are strongly equivalent, then $\mathcal{K}$ is strongly $\mathcal{K} \cup H$-inconsistent iff $\mathcal{K}'$ is strongly $\mathcal{K}' \cup H$-inconsistent.*

Proof. The knowledge base $\mathcal{K}$ is strongly $\mathcal{K} \cup H$-inconsistent if and only if $\mathcal{K} \cup G$ is inconsistent for any $G \subseteq H$. Since $\mathcal{K}$ and $\mathcal{K}'$ are strongly equivalent, this is the case if and only if $\mathcal{K}' \cup G$ is inconsistent for any $G \subseteq H$, which is the definition of $\mathcal{K}'$ being strongly $\mathcal{K}' \cup G$-inconsistent. □

3.4. Some Remarks on Strengthening Consistency

The analysis so far was based on inconsistency. Instead, one could also think of a refined notion of *consistency*. We will now discuss such a refinement, obtaining rather similar results. In a nutshell, we replace "consistency" with "strong consistency" this time, leading to a second possibility to generalise the results from monotonic frameworks.

Not only the following results are rather similar to the previous ones, but also their proofs. We will thus omit them for ease of presentation.

Definition 3.33. For $H, \mathcal{K} \subseteq \mathsf{WF}$ with $H \subseteq \mathcal{K}$, H is called *strongly $\mathcal{K}$-consistent* if $H' \subseteq H$ implies H' is consistent. Furthermore, H is called *maximal strongly $\mathcal{K}$-consistent subset* if $H \subsetneq H' \subseteq \mathcal{K}$ implies H' is not strongly $\mathcal{K}$-consistent. Let $SC(\mathcal{K})$ ($SC_{max}(\mathcal{K})$) denote the (maximal) strongly consistent subsets of $\mathcal{K}$.

Proposition 3.34. *Let $\mathcal{K}$ be a knowledge base.*

1. *If $\mathcal{K}$ is weakly monotonic, then $C(\mathcal{K}) = SC(\mathcal{K})$.*
2. *If $\mathcal{K}$ is weakly monotonic, then $C_{max}(\mathcal{K}) = SC_{max}(\mathcal{K})$.*

Definition 3.35. A set $\overline{M} \subseteq \mathcal{K}$ is in $coSC_{max}(\mathcal{K})$ if there is a set $M \in SC_{max}(\mathcal{K})$ such that $\overline{M} = \mathcal{K} \setminus M$.

We give the analogous duality results. Note that classical inconsistency is used rather than strong $\mathcal{K}$-inconsistency.

Theorem 3.36. *Let $\mathcal{K}$ be a knowledge base.*

1. *$\mathcal{S}$ is a minimal hitting set of $I_{min}(\mathcal{K})$ if and only if $\mathcal{K} \setminus \mathcal{S} \in SC_{max}(\mathcal{K})$.*
2. *$\mathcal{S}$ is a minimal hitting set of $coSC_{max}(\mathcal{K})$ if and only if $\mathcal{S} \in I_{min}(\mathcal{K})$.*

Corollary 3.37. *Let $\mathcal{K}$ be a knowledge base. Then*

$$\bigcap_{H \in SC_{max}(\mathcal{K})} H = \mathcal{K} \setminus \bigcup_{H \in I_{min}(\mathcal{K})} H.$$

We turn to free formulas. As the reader might expect already, we define:

Definition 3.38. Let $\mathcal{K}$ be a knowledge base. A formula $\alpha \in \mathcal{K}$ is called *free with respect to strong consistency* (or *SC-free* or simply *free* if there is no risk of confusion) if

$$\forall H \subseteq \mathcal{K} : \; H \in SC(\mathcal{K}) \Rightarrow H \cup \{\alpha\} \in SC(\mathcal{K}).$$

We let $Free_{SC}(\mathcal{K})$ denote the set of all SC-free formulas of $\mathcal{K}$.

Again, the expected result:

Proposition 3.39. *Let $\mathcal{K}$ be a knowledge base. Then*

$$Free_{SC}(\mathcal{K}) = \bigcap_{H \in SC_{max}(\mathcal{K})} H = \mathcal{K} \setminus \bigcup_{H \in I_{min}(\mathcal{K})} H.$$

Note that here, the classical notion of inconsistency and strong consistency are used. It is interesting that these complementary results can be obtained by considering strong consistency instead of strong inconsistency. However, we believe that in nonmonotonic reasoning, strong inconsistency and usual consistency as used for the previous results are more appropriate. For example, inconsistent subsets of a knowledge base should not be an issue as long as the knowledge base itself is consistent. That is why we continue our investigation focusing on strong inconsistency rather then strong consistency.

4. Computational Complexity

The notion of strong $\mathcal{K}$-inconsistency includes 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 P, NP and coNP. Furthermore, we consider the polynomial hierarchy as usual: we let $\Sigma_0^p = \Pi_0^p = \mathsf{P}$ and for $m \geq 1$, Σ_m^p is the class of all languages L such that there is a polynomial time Turing machine M (with output $M(x, X_1, ..., X_m)$) and a polynomial p such that $x \in L$ if and only if

$$\exists X_1 \ldots Q_m X_m : \ M(x, X_1, \ldots, X_m) = 1 \tag{11}$$

with $|X_1|, ..., |X_m| \leq p(|x|)$ and $Q_i \in \{\exists, \ \forall\}$ for $2 \leq i \leq m$. The class Π_m^p is defined analogously, but the expression in (11) starts with a universal quantifier rather then an existential one. Note that $\Sigma_1^p = \mathsf{NP}$ and $\Pi_1^p = \mathsf{coNP}$.

An equivalent formalisation makes use of oracle machines. Let $\mathcal{C}^{\mathcal{D}}$ be the class of decision problems solvable in $\mathcal{C}$ having access to an oracle for some problem that is complete in $\mathcal{D}$. Now, $\Sigma_{m+1}^p = \mathsf{NP}^{\Sigma_m^p}$ and $\Pi_{m+1}^p = \mathsf{coNP}^{\Sigma_m^p}$ for $m \geq 0$.

A *quantified Boolean formula* (QBF) Φ is a formula

$$\Phi = Q_1 X_1 \ldots Q_m X_m \, \phi$$

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 ϕ over the variables $X_1 \cup \ldots \cup X_m$. A QBF Φ is *true* if ϕ evaluates to true with respect to the quantifiers, e. g., $\forall x_1 \exists x_2 (x_1 \vee \neg x_2)$ is true as for every truth value of x_1 one can find a truth value of x_2 such that $x_1 \vee \neg x_2$ evaluates to true. A QBF Φ 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 Φ with m alternating quantifiers starting with $\exists$ (resp. starting with $\forall$) is true is the canonical Σ_m^p-complete (resp. Π_m^p-complete) problem (Papadimitriou, 1994).

We also make use of the differences classes D_m^p (Papadimitriou, 1994). They were introduced to capture threshold problems like "Is it true that a given graph G has a Hamiltonian cycle, but if one removes an arbitrary edge, the resulting graph does not?" Formally,

$$\mathsf{D}_1^p = \{L_1 \setminus L_2 \mid L_1, L_2 \in \mathsf{NP}\}$$

or, equivalently,

$$\mathsf{D}_1^p = \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP},\ L_2 \in \mathsf{coNP}\}.$$

For example, the generic D_1^p-complete problem is SAT-UNSAT, where we are given two propositional formulas ϕ_1 and ϕ_2, and have to decide whether ϕ_1 is satisfiable while ϕ_2 is not. The natural generalisation of D_1^p is

$$\mathsf{D}_m^p = \{L_1 \cap L_2 \mid L_1 \in \Sigma_m^p,\ L_2 \in \Pi_m^p\}.$$

Moreover, we consider PSPACE, i. e., the class of all languages L that can be computed using polynomial space. In contrast to Σ_m^p and Π_m^p, where we can decide the truth value of a QBF with m alternating quantifiers, the generic PSPACE-complete problem is deciding the truth value of an *arbitrary* QBF.

While examining the structure of a knowledge base $\mathcal{K}$, one might be interested in the number of minimal strongly $\mathcal{K}$-inconsistent subsets. We will thus consider counting complexity classes (cf. (Valiant, 1979)). They are defined using witness functions w that assign words from an input alphabet Σ to finite subsets of an alphabet Γ. Given a string x from the alphabet Σ, the task is to return $|w(x)|$, i. e., the number of witnesses. Given a class $\mathcal{C}$ of decision problems, by $\#\cdot\mathcal{C}$ we denote the class of counting problems such that

- for every input string x, each $y \in w(x)$ is polynomially bounded,
- the decision problem "Is $y \in w(x)$?" is in $\mathcal{C}$.

Thus, verifying a potential solution is in $\mathcal{C}$ and the length of it is polynomially bounded. For example, the generic $\#\cdot\Pi_2^p$-complete problem is counting the number of truth assignments to the X-variables rendering a formula of the form $\Phi = \forall Y \phi(X, Y)$ true. Here, $\mathsf{Mod}(\Phi)$ is the witness function assigning to a given formula the corresponding models. As required, each truth assignment is polynomial bounded and given an assignment to the X-variables, the decison problem whether $\forall Y \phi(X, Y)$ holds is in coNP.

4.1. Minimal Unsatisfiability for QBFs

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 D_1^p-complete. MU is the following problem: "Given a propositional formula ϕ 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.

For this we let QBF-MU$(Q_1, ..., Q_m)$ be the following problem:

> Given a QBF $\Phi = Q_1 X_1 \ldots Q_m X_m \ \phi$ in prenex normal form with $\phi = C_1 \wedge \ldots \wedge C_r$ and formulas $C_1, \ldots, C_r$, is it true that Φ is false, but removing any conjunct C_k from ϕ renders Φ true?

Since ϕ is a conjunction, Φ is true if and only if all conjuncts $C_1, \ldots, C_r$ evaluate to true (with respect to the quantifiers). Note that Φ evaluates to true if ϕ is the empty conjunction.

For the problem QBF-MU$(Q_1, ..., Q_m)$, 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$(Q_1, ..., Q_m)$ *is D_m^p-complete.*

Proof. (Membership) Given a QBF

$$\Phi = Q_1 X_1 \ldots Q_m X_m \ \phi$$

with $\phi = C_1 \wedge \ldots \wedge C_r$ we can verify that it is a "yes" instance of QBF-MU$(Q_1, ..., Q_m)$ by

- checking that Φ is false which is in Π_m^p if $Q_1 = \exists$ and in Σ_m^p if $Q_1 = \forall$ and

- checking that for each $k = 1, ..., r$ the formula

$$Q_1 X_1 \ldots Q_m X_m \, C_1 \wedge \ldots \wedge C_{k-1} \wedge C_{k+1} \wedge \ldots \wedge C_r$$

  is true which is in Σ_m^p if $Q_1 = \exists$ and in Π_m^p if $Q_1 = \forall$.

Since the latter consists of only linearly many checks, we obtain membership in D_m^p.

(Hardness) We consider the generic D_m^p-complete problem which, given two QBFS $\Phi_i = Q_1 X_1 \ldots Q_m X_m \phi_i$, $i = 1, 2$, asks whether Φ_1 is false while Φ_2 is true. First, we give a (polynomial) construction to obtain a formula Ψ which is a "yes" instance of QBF-MU$(Q_1, ..., Q_m)$ if and only if Φ_1 is false. A minor adjustment will lead to a second formula Ψ' which is a "yes" instance of QBF-MU$(Q_1, ..., Q_m)$ if and only Φ_2 is true. Combining both constructions will yield D_m^p-hardness. Since both reductions are considered independently, we omit the indices 1 and 2 for ease of presentation and denote the formula by $\Phi = Q_1 X_1 \ldots Q_m X_m \, \phi$ in both steps.

So let $\phi = C_1 \wedge \ldots \wedge C_r$ be a conjunction and let

$$\Phi = Q_1 X_1 \ldots Q_m X_m \, \phi$$

be a QBF in prenex normal form where $m \geq 2$. We distinguish two cases, depending on the *final* quantifier.

Case 1: $Q_m = \exists$.

Here, we can assume that ϕ is in 3-CNF, i. e., all C_k are disjunctions containing at most three literals, say $C_k = x_{k,1} \vee \ldots \vee x_{k,3}$ for $k = 1, ..., r$. Our proof is similar to the one given in Papadimitriou and Wolfe (1988), Theorem 1. The reader may be referred to this proof since we do not give as many details. We use a similar construction, where we, roughly speaking, ignore variables that occur in the scope of an universal quantifier for the most part.

For any $k \in \{1, \ldots, r\}$, we assume w. l. o. g. that the disjunction C_k is not a tautology, i. e., does not contain both a and $\neg a$ for an atom a.

Let $\mathcal{X}$ be the set of all variables in Φ that occur in the scope of an existential quantifier. Let $y_1, \ldots, y_r$ be fresh atoms not appearing in Φ and define

$$Y = y_1 \vee \ldots \vee y_r.$$

Let

$$Y - y_k := y_1 \vee \ldots \vee y_{k-1} \vee y_{k+1} \vee \ldots \vee y_r$$

for each $k = 1, \ldots, r$. We construct a formula ψ containing the following conjuncts.

- If C_k occurs in ϕ, then

$$D_k = C_k \vee (Y - y_k)$$

  is a conjunct of ψ.

- Let $C_k = x_{k,1} \vee \ldots \vee x_{k,3}$. Then, for $j = 1, 2, 3$ and $x_{k,j} \in \mathcal{X}$

$$E_{k,j} = \neg x_{k,j} \vee (Y - y_k) \vee \neg y_k$$

  occurs in ψ as a conjunct.

- For each $i, j = 1, \ldots, r$ with $i \neq j$

$$H_{i,j} = \neg y_i \vee \neg y_j$$

  is a conjunct of ψ.

Consider

$$\Psi = Q_1 X_1 \ldots Q_{m-1} X_{m-1} \exists X_m \cup \{y_1, \ldots, y_r\}\ \psi.$$

We claim that Φ is false if and only if Ψ is a "yes" instance of QBF-MU$(Q_1, ..., Q_m)$.

"$\Rightarrow$": We start by showing that Ψ is false. Afterwards, we prove minimality. The first step is to argue that all y-variables would need to be false in order for Ψ to be true. Due to the $H_{i,j}$ conjuncts, at most one y-variable can be true, say y_k. We can assume that at least one $E_{k,j}$ ($j = 1, 2, 3$) exists, i. e., C_k contains variables in $\mathcal{X}$. Otherwise, all variables in C_k occur in the scope of an universal quantifier. Since we assumed no tautological conjunct exists, there is an assignment to those variables rendering C_k false. Thus, in order for Ψ to be true, there has to be a $k' \neq k$ such that $y_{k'}$ is true (because of the conjunct D_k), which is not possible, as argued above.

Now consider $E_{k,j}$. To render it true, $x_{k,j}$ needs to be false ($j = 1, 2.3$). However, we also need to consider the conjunct

$$D_k = x_{k,1} \vee x_{x,2} \vee x_{k,3} \vee (Y - y_k).$$

If all three variables $x_{k,1}, \ldots, x_{k,3}$ are in $\mathcal{X}$, then D_k is false. Otherwise, the remaining variables occur in the scope of an universal quantifier. Again, there is a assignment to those variables rendering D_k false.

Hence, if one y-variable is true, then Ψ is false. This, however, finishes the first step already: Since Φ is false by assumption and all y-variables need to be false, Ψ is false as well.

We turn to minimality and argue that removal of any conjunct in ψ renders Ψ true.

- If we remove $H_{i,j}$ then letting y_i and y_j be true ensures satisfaction of all conjuncts. Of course, the other y-variables need to be false.
- If $E_{k,j}$ is removed (assuming it exists and hence $x_{k,j} \in \mathcal{X}$), then let $x_{k,j}$ and y_k be true, the other y-variables false. If $E_{k,j'}$ exists for $j' \neq j$, then $x_{k,j'}$ needs to be false. This assignment ensures satisfaction of all conjuncts.
- If we remove D_k, then all conjuncts are satisfied if y_k is true and $x_{k,1}, \ldots, x_{k,3}$ are false (for the j such that $x_{k,j} \in \mathcal{X}$). The latter are needed for the $E_{k,j}$ (which exists for the j such that $x_{k,j} \in \mathcal{X}$).

"$\Leftarrow$": Assume Ψ is false. In particular, this means that Ψ is false even if all y-variables are false. Thus, Φ is false.

So, Ψ is a "yes" instance of QBF-MU$(Q_1, ..., Q_m)$ if and only if Φ is false.

As mentioned above, we need a second reduction to obtain a formula Ψ' that is a "yes" instance of QBF-MU$(Q_1, ..., Q_m)$ if and only if Φ is true. To obtain such a formula, use the same construction and add $y_1 \vee \ldots \vee y_r$ as a conjunct to ψ. Then, Ψ' is definitely false and minimal if and only if Φ is true, as similar considerations show.

Hence, we obtain the two desired formulas Ψ and Ψ'. We describe below how they are combined, which does not depend on the cases we distinguish.

Case 2: $Q_m = \forall$.

We may assume ϕ to be in 3-DNF here, i. e., $r = 1$ and $C := C_1$ is in 3-DNF. Let $C = D_1 \vee \ldots \vee D_n$ with $D_i = x_{i,1} \wedge x_{i,2} \wedge x_{i,3}$ for $i = 1, \ldots, n$. We see that $\Psi = \Phi$ is a "yes" instance of QBF-MU$(Q_1, ..., Q_m)$ if and only if the formula Φ is false, since removing the only conjunct (i. e., the whole formula ϕ) renders Φ true, because the empty conjunct is true by definition. Moreover,

$$\Psi' = Q_1 X_1 \ldots Q_{m-2} X_{m-2} \exists X_{m-1} \cup \{y\} \forall X_m \, \psi'$$

with

$$\psi' = ((D_1 \wedge y) \vee \ldots \vee (D_n \wedge y)) \wedge (\neg y)$$

is a "yes" instance of QBF-MU$(Q_1, ..., Q_m)$ if and only if Φ is true.

Combining:

In both cases above, we obtain two formulas Ψ and Ψ' that are both positive instances of QBF-MU$(Q_1, ..., Q_m)$. To finish our proof for hardness, we make *one* formula out of them. We assume w. l. o. g. that Ψ and Ψ' are QBFs over disjoint sets of variables. Given these two QBFs, we construct one formula Θ as it is done in (Papadimitriou and Wolfe, 1988), Lemma 3. Let

$$\Psi = Q_1 X_1 \dots Q_m X_m \, \psi, \qquad \Psi' = Q_1 X'_1 \dots Q_m X'_m \, \psi'$$

with

$$\psi = C_1 \wedge \ldots \wedge C_r, \qquad \psi' = C'_1 \wedge \ldots \wedge C'_{r'}.$$

Now, Θ contains of all possible pairs of clauses, one from ψ and one from ψ', i. e., for

$$\theta_{p,q} = C_p \wedge C'_q, \quad p = 1, \ldots, r, q = 1, \ldots, r'$$

we let

$$\Theta = Q_1 X_1 \cup X'_1 \dots Q_m X_m \cup X'_m \bigwedge_{p=1,\ldots,r, q=1,\ldots,r'} \theta_{p,q}$$

Now, removing $\theta_{p,q}$ from Θ corresponds to removing C_p from Ψ and C'_q from Ψ'. Hence, Θ is a "yes" instance of QBF-MU$(Q_1, ..., Q_m)$ if and only if both Ψ and Ψ' are.

To summarize, Θ is a positive instance of QBF-MU$(Q_1, ..., Q_m)$ if and only if Φ_1 is false while Φ_2 is true. Hardness in D^p_m follows. □

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 Φ is of the form $\Phi = \forall X \phi$ is missing. Indeed, it turns out to be easier.

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

Proof. Let Φ be the sentence

$$\Phi = \forall X \phi$$

with $\phi = C_1 \wedge \ldots \wedge C_r$. Verifying that Φ is false is obviously NP-complete in general. However, there is nothing to check beyond that, because removal of an arbitrary conjunct renders Φ true if and only if $r = 1$, i. e., ϕ consists of one conjunct only. That is easy to see: Assume $r \geq 2$ and assume Φ is false. Then,

there is a k and a truth assignment to the variables such that C_k is false. If we remove k' for any $k' \neq k$, then C_k is still false for the same truth assignment and thus,

$$\forall X \ (C_1 \wedge \ldots \wedge C_{k'-1} \wedge C_{k'+1} \wedge \ldots \wedge C_r)$$

is still false, i. e., Φ is a "no" instance of QBF-MU($\forall$). □

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 $L = L(Q_1, \ldots, Q_m, X_1, \ldots, X_m) = (\mathsf{WF}, \mathsf{BS}, \mathsf{INC}, \mathsf{ACC})$ as follows: $\mathsf{WF} = \mathsf{WF}(X_1 \cup \ldots \cup X_m)$ is the set of all well-formed Boolean formulas over the atoms in $X_1 \cup \ldots \cup X_m$, $\mathsf{BS} = \{\bot, \top\}$, $\mathsf{INC} = \{\bot\}$ and for a knowledge base $\mathcal{K} = \{C_1, \ldots, C_r\}$ we let $\phi(\mathcal{K}) = C_1 \wedge \ldots \wedge C_r$ and define $\mathsf{ACC} = \mathsf{ACC}(Q_1, \ldots, Q_m)$ via

$$\mathsf{ACC}(\mathcal{K}) = \begin{cases} \{\top\}, & \text{if } Q_1 X_1 \ldots Q_m X_m \ \phi(\mathcal{K}), \\ \{\bot\}, & \text{otherwise.} \end{cases}$$

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

4.2. (Minimal) Strong Inconsistency in General

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 $L = (\mathsf{WF}, \mathsf{BS}, \mathsf{INC}, \mathsf{ACC})$ 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$_L$	**Input**:	$\mathcal{K} \subseteq \mathsf{WF}$
	Output:	TRUE iff $\mathcal{K}$ is consistent
S-INC$_L$	**Input**:	$\mathcal{K} \subseteq \mathsf{WF}$, $H \subseteq \mathcal{K}$
	Output:	TRUE iff $H \in SI(\mathcal{K})$
MIN-S-INC$_L$	**Input**:	$\mathcal{K} \subseteq \mathsf{WF}$, $H \subseteq \mathcal{K}$
	Output:	TRUE iff $H \in SI_{min}(\mathcal{K})$

In other words, SAT$_L$ is the generalisation of the satisfiability problem in our general logic L, S-INC$_L$ is about deciding whether H is strongly $\mathcal{K}$-inconsistent, and

MIN-S-INC$_L$ is about deciding whether H is a minimal strongly $\mathcal{K}$-inconsistent set. If L is (weakly) monotonic and SAT$_L \in \mathcal{C}$ for some class $\mathcal{C}$, then S-INC$_L$ 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 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$_L$ *is in*

(a) Σ^p_m, then S-INC$_L$ *is in Π^p_m,*

(b) Π^p_m, then S-INC$_L$ *is in Π^p_{m+1},*

(c) Π^p_m and L is weakly monotonic, then S-INC$_L$ *is in Σ^p_m.*

Proof. See proof of Theorem 4.5. □

Theorem 4.1 already showed how difficult MIN-S-INC$_L$ is compared to the decision problem SAT$_L$ 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$_L$. 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$_L$ to the problem MIN-S-INC$_L$—i. e., additionally asking for minimality—does not involve going up an additional level in the polynomial hierarchy but only moving to the corresponding D^p_m class.

Theorem 4.5. *Let $\mathcal{K}$ be a knowledge base. Let $m \geq 1$. If the decision problem* SAT$_L$ *is in*

(a) Σ^p_m, then MIN-S-INC$_L$ *is in D^p_m,*

(b) Π^p_m, then MIN-S-INC$_L$ *is in D^p_{m+1},*

(c) Π^p_m and L is weakly monotonic, then MIN-S-INC$_L$ *is in D^p_m.*

Proof. (a) We need to show that it is sufficient to solve one problem in Σ^p_m and one in Π^p_m. To check whether H is strongly $\mathcal{K}$-inconsistent, we need to check that H' is inconsistent for each $H \subseteq H' \subseteq \mathcal{K}$. Checking that H' is

a "no" instance of SAT_L is in Π_m^p. Since there are only exponentially many $H' \subseteq \mathcal{K}$, we can also decide in Π_m^p whether

$$\forall H \subseteq H' \subseteq \mathcal{K} : \ H' \text{ is inconsistent}$$

is true. The minimality of H can be written as

$$\forall G \subsetneq H \ \exists G' : \ G \subseteq G', \ G' \text{ is consistent} \tag{12}$$

stating that each proper subset G of H has a consistent superset G', which ensures that G is not strongly $\mathcal{K}$-inconsistent. However, it is sufficient to check $|H|$ subsets of H: Let $H = \{\alpha_1, \ldots, \alpha_k\}$ and let $H_i = H \setminus \{\alpha_i\}$, i. e., we consider the $|H|$ possible subsets of size $|H| - 1$. Now consider an arbitrary set $G \subsetneq H$. Clearly, there is one i such that $G \subseteq H_i$. Hence, if H_i has a consistent superset H_i', then so has G. Thus, all proper subset of H do have a consistent superset if and only if this is the case for $H_1, ..., H_k$. Hence, we only need to check these subsets of H and therefore, deciding whether

$$\exists H_i \subseteq H_i' : \ H_i' \text{ is consistent}$$

is true (with $H_1, ..., H_k$ as described) is sufficient. Since there are only linearly many H_i and only exponentially many potential sets $H_i \subseteq H_i'$, this is in Σ_m^p if SAT_L is in Σ_m^p. Now, the claim follows due to the definition of D_m^p.

(b) This follows from (a) since $\Pi_m^p \subseteq \Sigma_{m+1}^p$.

(c) If L is weakly monotonic and SAT_L in Π_m^p, then MIN-S-INC_L corresponds to

- verifying that H is inconsistent (which is in Σ_m^p) and
- verifying that $H_1, ..., H_k$ as in (a) are consistent (which is in Π_m^p).

Thus, MIN-S-INC_L is in D_m^p.

□

In the above theorems we required $m \geq 1$, i. e., we did not consider logics where SAT_L is in P. Indeed, the statements are not true anymore for $m = 0$ as pointed out in Theorem 4.11 below.

So far, the decision problem SAT_L was assumed to be in a fixed level of the polynomial hierarchy, where the generic complete problem is deciding the truth

value of a QBF with a fixed amount of quantifiers. As mentioned above, deciding the truth value of *arbitrary* QBFs is the generic complete problem for PSPACE. Thus, it seems natural to continue our investigation here. However, PSPACE is already "too big" in order to be interesting for us, because one can just enumerate all subsets of $\mathcal{K}$. Then, the decision problems we considered so far depend on deciding (un)satisfiability only. The minimality check as well as considering *all* supersets of $H \subseteq \mathcal{K}$ are rendered trivial.

Proposition 4.6. *Let $\mathcal{K}$ be a knowledge base. If the decision problem* SAT_L *is in* PSPACE*, then*

(a) S-INC_L *is in* PSPACE *and*

(b) MIN-S-INC_L *is in* PSPACE*.*

Proof. In PSPACE, we can successively enumerate all subsets of $\mathcal{K}$ and then perform the corresponding consistency checks. It is then clear whether or not $(\mathcal{K}, H)$ is a "yes" instance of S-INC_L resp. MIN-S-INC_L. □

We briefly discuss the natural counting problem corresponding to MIN-S-INC_L:

#MIN-S-INC(L) **Input**: $\mathcal{K} \subseteq$ WF
Output: $|SI_{min}(\mathcal{K})|$

The result regarding membership is similar to Theorem 4.5. Recall that we considered the same three cases in Theorem 4.5 and note that (b) is the most difficult one again.

Theorem 4.7. *Let $\mathcal{K}$ be a knowledge base. Let $m \geq 1$. If the decision problem* SAT_L *is in*

(a) Σ_m^p*, then* #S-INC(L) *and* #MIN-S-INC(L) *are in* $\#\cdot\Pi_m^p$*,*

(b) Π_m^p*, then* #S-INC(L) *and* #MIN-S-INC(L) *are in* $\#\cdot\Pi_{m+1}^p$*,*

(c) Π_m^p *and L is weakly monotonic, then* #S-INC(L) *and* #MIN-S-INC(L) *are in* $\#\cdot\Pi_m^p$*.*

Proof. We make use of the observation that

$$\#\cdot\Delta_{m+1}^p = \#\cdot\Pi_m^p$$

(see (Hemaspaandra and Vollmer, 1995)). Furthermore, it is clear that

$$\mathsf{D}_m^p \subseteq \Delta_{m+1}^p.$$

So, we use Theorems 4.4 and 4.5 and obtain:

(a) If SAT_L is in Σ^p_m, then S-INC_L is in Π^p_m and thus, $\#\text{S-INC}(L)$ is in $\#\cdot\Pi^p_m$. Moreover, MIN-S-INC_L is in D^p_m and hence, $\#\text{MIN-S-INC}(L)$ is in $\#\cdot\Delta^p_m = \#\cdot\Pi^p_m$.

Analogously, (b) and (c) follow from Theorems 4.4 and 4.5. □

4.3. Hardness Results for Logic Programs and Abstract Argumentation

In Theorems 4.4 and 4.5, only membership statements are given. But bear in mind that these results are valid for *every* logic that can be phrased via Definition 2.1. One cannot expect to obtain similar general hardness results as reductions may work very differently for different logics (and they do for the cases we are going to consider). Nonetheless, we are now going to give some concrete hardness results for specific cases. First, we construct an arbitrary logic witnessing that the result in item (b) of the two theorems cannot be improved in general. Moreover, we consider abstract argumentation frameworks and logic programming. Those frameworks belong to case (a) of Theorems 4.4 and 4.5 and hardness in the belonging class will be proven.

Theorem 4.8. *For any $m \geq 1$, there is a logic $L_{\Pi^p_m} = (\mathit{WF}_{\Pi^p_m}, \mathit{BS}_{\Pi^p_m}, \mathit{INC}_{\Pi^p_m}, \mathit{ACC}_{\Pi^p_m})$ such that $\text{SAT}_{L_{\Pi^p_m}}$ is in Π^p_m and*

(a) $\text{S-INC}_{L_{\Pi^p_m}}$ is Π^p_{m+1}-complete and

(b) $\text{MIN-S-INC}_{L_{\Pi^p_m}}$ is D^p_{m+1}-complete.

Proof. For this proof, we make use of the following notation: Assume we are given a set X of atoms and a formula ϕ over X. Consider a (partial) assignment $\omega : X \to \{0, 1\}$. Now, we let $\phi[\omega]$ be the formula where the atoms are evaluated according to ω, i. e., $a \in X$ is substituted by 1 if $\omega(a) = 1$, by 0 if $\omega(a) = 0$ and remains unchanged if ω is not defined on a.

Now let $m \geq 1$. Consider quantifiers $Q_1 = \forall, \ldots, Q_m$ and sets of variables $X_1, \ldots, X_m$. We define a logic $L_{\Pi^p_m} = L_{\Pi^p_m}(Q_1, \ldots, Q_m, X_1, \ldots, X_m) = (\mathsf{WF}_{\Pi^p_m}, \mathsf{BS}_{\Pi^p_m}, \mathsf{INC}_{\Pi^p_m}, \mathsf{ACC}_{\Pi^p_m})$. The set $\mathsf{WF}_{\Pi^p_m}$ consists of tuples of the form (ϕ, L), where ϕ is a boolean formula over the variables in $X_1 \cup \ldots \cup X_m$ and L is a set of literals over $X_1 \cup \ldots \cup X_m$. Let $\mathsf{BS}_{\Pi^p_m} = \{\bot, \top\}$ and $\mathsf{INC}_{\Pi^p_m} = \{\bot\}$. The mapping $\mathsf{ACC}_{\Pi^p_m}$ works similar as in Remark 4.3, except formulas are tuples (ϕ, L) now. In a nutshell, the set L in (ϕ, L) defines a partial assignment. So given a knowledge base

$$\mathcal{K} = \{(C_1, L_1), \ldots, (C_r, L_r)\}$$

we let $L_{\mathcal{K}} = L = L_1 \cup \ldots \cup L_r$. If L contains two complementary literals, then $\mathsf{ACC}_{\Pi^p_m}(\mathcal{K}) = \{\bot\}$, rendering $\mathcal{K}$ inconsistent (the reason is that L shall correspond to a (partial) assignment to the variables; hence, it cannot contain a complementary pair of literals). Otherwise, let $\omega_{\mathcal{K}} = \omega : X_1 \cup \ldots \cup X_m \to \{0,1\}$ be the (partial) assignment corresponding to L, i. e.,

$$\omega(l) = \begin{cases} 1 & \text{if } l \in L, \\ 0 & \text{if } l \notin L. \end{cases}$$

The mapping $\mathsf{ACC}_{\Pi^p_m}$ treats the formulas $C_1, \ldots, C_r$ in $\mathcal{K}$ as conjuncts while respecting the (partial) assignment ω, i. e., we let $\phi_{\mathcal{K}} = \phi = C_1 \wedge \ldots \wedge C_r$ and

$$\mathsf{ACC}_{\Pi^p_m}(\mathcal{K}) = \begin{cases} \{\top\} & \text{if } \forall X_1 \ldots Q_m X_m \, \phi[\omega], \\ \{\bot\} & \text{otherwise.} \end{cases}$$

This mechanism introduced by ω is important for us due to two different reasons: First, we can introduce formulas that correspond to partial assignments to "neutralise" a universal quantifier, which gives our logic the nonmonotonic layer we need. Second, we can construct knowledge bases such that supersets correspond to assignments. As strong $\mathcal{K}$-inconsistency considers *all* supersets of a given $H \subseteq \mathcal{K}$, this facilitates simulation of an additional universal quantifier.

We now show that $L_{\Pi^p_m}$ has the properties we claim, i. e., $\textsc{Sat}_{L_{\Pi^p_m}}$ is in Π^p_m, $\text{S-INC}_{L_{\Pi^p_m}}$ is Π^p_{m+1}-complete and $\textsc{Min}\text{-S-INC}_{L_{\Pi^p_m}}$ is D^p_{m+1}-complete.

(Membership) Since we consider m quantifiers, where the first one is universal, it is rather easy to see that $\textsc{Sat}_{L_{\Pi^p_m}}$ is in Π^p_m. Then $\text{S-INC}_{L_{\Pi^p_m}} \in \Pi^p_{m+1}$ and $\textsc{Min}\text{-S-INC}_{L_{\Pi^p_m}} \in \mathsf{D}^p_{m+1}$ follow from Theorems 4.4 and 4.5.

(Hardness) Assume we are given a formula

$$\Phi = \exists Z \forall X_2 Q_3 X_3 \ldots Q_{m+1} X_{m+1} \, \phi. \tag{13}$$

Note that Φ contains $m+1$ quantifiers, i. e., deciding whether it is true is Σ^p_{m+1}-complete in general. We assume ϕ is an arbitrary conjunction, i. e., $\phi = C_1 \wedge \ldots \wedge C_r$. We consider the logic

$$L_{\Pi^p_m} = L_{\Pi^p_m}(\forall, Q_3, \ldots, Q_{m+1}, Z \cup X_2, X_3, \ldots, X_{m+1}).$$

For (a), we show that there is a knowledge base $\mathcal{K}$ with a subset H that is a "yes" instance of $\text{S-INC}_{L_{\Pi^p_m}}$ if and only if Φ is false. The statement (b) follows from (a) utilizing a formula similar to Θ we constructed in the proof of Theorem 4.1.

(a) We prove the following statement: Given the formula

$$\Phi = \exists Z \forall X_2 Q_3 X_3 \dots Q_{m+1} X_{m+1}\, \phi \tag{13}$$

where $\phi = C_1 \wedge \ldots \wedge C_r$ is an arbitrary conjunction, the pair $(\mathcal{K}, H)$ with

$$\mathcal{K} = \{(C_1, \emptyset), \dots, (C_r, \emptyset), (\top, z_1), (\top, \neg z_1), \dots, (\top, z_n), (\top, \neg z_n)\},$$
$$H = \{(C_1, \emptyset), \dots, (C_r, \emptyset)\}$$

is a "yes" instance of S-INC$_{L_{\Pi_m^p}}$ if and only if Φ is false.

Note that we can identify H with $\{\phi, \emptyset\}$ due to the definition of our logic. The subset H is considered consistent if

$$\forall Z \cup X_2 Q_3 X_3 \dots Q_{m+1} X_{m+1}\, \phi$$

is valid and hence, inconsistent if

$$\exists \{Z \cup X_2\} \overline{Q_3} X_3 \dots \overline{Q_{m+1}} X_{m+1}\, \neg\phi \tag{14}$$

holds, where $\overline{Q_i}$ is the complementary quantifier. Note that the formulas besides the $(C_i, \emptyset)$ are all of the form $(\top, z_k)$ resp. $(\top, \neg z_k)$. Hence, all they do is fixing z-variables. So, naturally, any set H' with $H \subseteq H' \subseteq \mathcal{K}$ corresponds to the formula Φ with respect to a partial assignment on Z. This motivates the following notation: Given a (partial) assignment $\omega : Z \to \{0, 1\}$, we let H_ω be the set of formulas of the form $(\top, z_k)$ resp. $(\top, \neg z_k)$ that naturally corresponds to the assignment, i. e., if $\omega(z_k) = 1$, then $(\top, z_k)$ occurs in H_ω and if $\omega(z_k) = 0$, then $(\top, \neg z_k)$ occurs in H_ω.

Now, we show: Φ is false if and only if H is strongly $\mathcal{K}$-inconsistent.

"$\Leftarrow$": Assume H is strongly $\mathcal{K}$-inconsistent. Consider an assignment $\omega : Z \to \{0, 1\}$ and the set

$$H' = H \cup H_\omega.$$

Since H is strongly $\mathcal{K}$-inconsistent, H' is inconsistent. The formulas H_ω augment ϕ with conjuncts "$\top$" only, which can be ignored. Additionally, the z-variables are fixed according to ω, making the consideration of $\phi[\omega]$ rather than ϕ itself necessary. Thus, H' being inconsistent means

$$\exists \{Z \cup X_2\} \overline{Q_3} X_3 \dots \overline{Q_{m+1}} X_{m+1}\, \neg\phi[\omega]$$

holds. Since ω fixes the z-variables, this is equivalent to

$$\exists X_2 \overline{Q_3} X_3 \ldots \overline{Q_{m+1}} X_{m+1}\, \neg\phi[\omega].$$

Since ω was an arbitrary assignment, we obtain that

$$\forall Z \exists X_2 \overline{Q_3} X_3 \ldots \overline{Q_{m+1}} X_{m+1}\, \neg\phi$$

holds. Hence,

$$\Phi = \exists Z \forall X_2 Q_3 X_3 \ldots Q_{m+1} X_{m+1}\, \phi \tag{13}$$

is false.

"$\Rightarrow$": Now assume Φ is false. Hence,

$$\forall Z \exists X_2 \overline{Q_3} X_3 \ldots \overline{Q_{m+1}} X_{m+1}\, \neg\phi \tag{15}$$

holds. Again, consider an assignment $\omega : Z \to \{0,1\}$. Since (15) is true, we obtain that in particular,

$$\exists X_2 \overline{Q_3} X_3 \ldots \overline{Q_{m+1}} X_{m+1}\, \neg\phi[\omega]$$

is true. Since the z-variables are fixed anyway,

$$\exists \{Z \cup X_2\} \overline{Q_3} X_3 \ldots \overline{Q_{m+1}} X_{m+1}\, \neg\phi[\omega]$$

holds as well. Due to the definition of our logic, this means that the set

$$H' = H \cup H_\omega$$

is inconsistent. Now consider any set H^* with $H \subseteq H^* \subseteq H'$. Such H^* naturally corresponds to an assignment $\omega^* : Z^* \to \{0,1\}$ with $Z^* \subseteq Z$ and $\omega_{|Z^*} = \omega^*$. It is an easy observation that

$$\exists \{Z \cup X_2\} \overline{Q_3} X_3 \ldots \overline{Q_{m+1}} X_{m+1}\, \neg\phi[\omega]$$

being true now implies that

$$\exists \{Z \cup X_2\} \overline{Q_3} X_3 \ldots \overline{Q_{m+1}} X_{m+1}\, \neg\phi[\omega^*]$$

holds as well, because the latter is the same formula with less z-variables being fixed. Again by definition of our logic, H^* is inconsistent. Since ω and ω^* are arbitrary, we obtain that any

$$H' = H \cup H_\omega$$

where ω is a (partial) assignment is inconsistent. Now, the remaining sets H' with $H \subseteq H' \subseteq \mathcal{K}$ do not correspond to a (partial) assignment. Hence, all of them contain at least one pair of the form $\{(\top, z_k), (\top, \neg z_k)\}$ rendering them inconsistent (recall the definition of $\mathsf{ACC}_{\Pi_m^p}$ if the set L contains a complementary pair of literals). Hence, H is strongly $\mathcal{K}$-inconsistent.

This completes (a).

(b) For (b), the technical work is already done. Assume we are given the generic D_{m+1}^p-complete problem, i. e., two formulas Φ_1 and Φ_2 of the form

$$\Phi_i = \exists Z \forall X_2 Q_3 X_3 \ldots Q_{m+1} X_{m+1}\, \phi_i \tag{13}$$

and have to decide whether Φ_1 is false while Φ_2 is true. We construct the formula

$$\Theta = \exists Z \forall X_2 Q_3 X_3 \ldots Q_{m+1} X_{m+1}\, \theta$$

as it is done in the proof of Theorem 4.1, within the last step. Thus, θ is a conjunction and for notational convenience, we assume $\theta = C_1 \wedge \ldots \wedge C_r$. Recall that Θ is a "yes" instance of QBF-MU$(Q_1, ..., Q_{m+1})$ if and only if Φ_1 is false while Φ_2 is true. Consider $(\mathcal{K}, H)$ given as in (a), i. e.,

$$\mathcal{K} = \{(C_1, \emptyset), \ldots, (C_r, \emptyset), (\top, z_1), (\top, \neg z_1), \ldots, (\top, z_n), (\top, \neg z_n)\},$$
$$H = \{(C_1, \emptyset), \ldots, (C_r, \emptyset)\}.$$

Now $(\mathcal{K}, H)$ is a "yes" instance of MIN-S-INC$_{L_{\Pi_m^p}}$ if and only if H is strongly $\mathcal{K}$-inconsistent, but no proper subset of H. Due to (a), this is the case if and only if Θ is false, but removing any conjunct from θ renders it true. By construction of Θ, this is the case if and only if Φ_1 is false, while Φ_2 is true. This yields hardness in D_{m+1}^p.

□

We give two more hardness results for case (a) of Theorem 4.5 for the framework of logic programming, i. e., the logics L_A^{ASP} and $L_A^{\text{ASP}_{k=0}}$ we introduced in Section 2.3. Be reminded that deciding whether a given disjunction-free logic program P is consistent is NP-complete (Eiter and Gottlob, 1995).

Theorem 4.9. *It holds that*

(a) the problem S-INC$_{L_A^{ASP_{k=0}}}$ *is* Π_1^p*-complete,*

(b) the problem MIN-S-INC$_{L_A^{ASP}{}_{k=0}}$ *is* D_1^p*-complete.*

Proof. (Membership) The membership statements follow from Theorems 4.4 and 4.5 since SAT$_{L_A^{\text{ASP}}{}_{k=0}}$ is in NP.

(Hardness) As mentioned above, the problem MU, i. e., checking whether a given formula in CNF is unsatisfiable, but removing one clause renders it satisfiable, is D_1^p-complete (Papadimitriou and Wolfe, 1988). Given such a formula, we construct a program P and a subprogram H that is minimal strongly P-inconsistent if and only if Φ is a "yes" instance of MU.

Let Φ be in 3-CNF, i. e., the conjunction of $C_1, ..., C_r$ with $C_k = l_{k,1} \vee \ldots \vee l_{k,3}$. Let $a_1, ..., a_n$ be the atoms occurring in Φ. Let σ be the mapping translating classical negation into default negation, i. e.,

$$\sigma(l) = \begin{cases} a_i & \text{if } l = a_i \in \{a_1, ..., a_n\}, \\ \text{not } a_i & \text{if } l = \neg a_i \in \{\neg a_1, ..., \neg a_n\}. \end{cases}$$

Let P and H be the following programs:

$$\begin{array}{ll} P: & \\ a_1., \ldots, a_n. & \\ w_{C_k} \leftarrow \sigma(l_{k,1}). & k = 1, \ldots, r \\ w_{C_k} \leftarrow \sigma(l_{k,2}). & k = 1, \ldots, r \\ w_{C_k} \leftarrow \sigma(l_{k,3}). & k = 1, \ldots, r \\ \leftarrow \text{not } w_{C_k}. & k = 1, \ldots, r \\ H: & \\ \leftarrow \text{not } w_{C_k}. & k = 1, \ldots, r \end{array}$$

Intuitively, for each k the atom w_{C_k} shall witness that the clause C_k is true. That is why it can be included in an answer set whenever one of the literals $\sigma(l_{k,1}), \ldots, \sigma(l_{k,3})$ is. Note that the construction of P is polynomial. We prove the claimed hardness results, starting with the second one.

(b) We claim that H is minimal strongly P-inconsistent if and only if Φ is a "yes" instance of MU.

"$\Rightarrow$": Assume $H \in SI_{min}(P)$. Hence, no program H' with $H \subseteq H' \subseteq P$ is consistent.

We argue that Φ must be unsatisfiable: For the sake of contradiction, assume there is a satisfying assignment ω to the variables $a_1, \ldots, a_n$. Augment H with the fact "a_i" for the i such that $\omega(a_i) = 1$. Moreover, add all rules of the form "$w_{C_k} \leftarrow \sigma(l_{k,i})$". Since ω renders Φ true, all constraints in H are satisfied. Thus, H' is consistent—a contradiction since H was assumed to be strongly inconsistent. Similarly, minimality of H in $SI(P)$ ensures that removing any conjunct from Φ renders it satisfiable.

"$\Leftarrow$": Now assume Φ is a "yes" instance of MU. Then, the construction as described above does not work; no super-program of H is consistent. Hence, H is strongly inconsistent. Minimality is similar, again.

As we already mentioned, MU is D_1^p-hard. Hence, we obtain hardness of MIN-S-INC$_{L_A^{\text{ASP}_{k=0}}}$ in D_1^p as well.

(a) Φ is an arbitrary formula, i. e., deciding whether it is unsatisfiable is $\Pi_1^p =$ coNP-complete. As argued above, H is strongly P-inconsistent if and only if Φ is unsatisfiable. Hardness in coNP follows.

□

Consider again the program P from the previous proof. Except the constraints "$\leftarrow$ not w_{C_k}." the program is even stratified (Apt et al., 1988). This is particularly interesting since deciding consistency of stratified programs is in P. However, as seen in the proof, the construction suffices to show hardness of S-INC$_{L_A^{\text{ASP}_{k=0}}}$ and MIN-S-INC$_{L_A^{\text{ASP}_{k=0}}}$ in Π_1^p and D_1^p, respectively. Thus, we see that Theorems 4.4 and 4.5 are not applicable for $m = 0$, as we already mentioned earlier. For that, we recall the notion of stratification (Apt et al., 1988).

Definition 4.10. A logic program P over a set A of atoms is called *stratified* if there is a mapping $\|\cdot\| : A \to \mathbb{N}$ such that for any rule

$$r: \quad l_0 \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n.$$

from P it holds that

- $k(l_0) \geq k(l_i)$ for $i = 1, ..., m$,
- $k(l_0) > k(l_j)$ for $j = m + 1, ..., n$.

Now, we construct a simple logic that rejects any program which is not stratified. A knowledge base is going to consist of rules and integers corresponding to strata for the atoms. It is thus quite easy to see that the satisfiability check of this logic is in P. We utilize the same construction as in the proof of Theorem 4.9 for the hardness results.

Theorem 4.11. *There is a logic* $L_A^{P_{Strat}} = (\mathit{WF}_A^{P_{Strat}}, \mathit{BS}_A^{P_{Strat}}, \mathit{INC}_A^{P_{Strat}}, \mathit{ACC}_A^{P_{Strat}})$ *such that* $\mathrm{SAT}_{L_A^{P_{Strat}}}$ *is in* $\mathsf{P} = \Sigma_0^p$ *and*

(a) $\text{S-INC}_{L_A^{P_{Strat}}}$ *is* coNP*-complete and*

(b) $\text{MIN-S-INC}_{L_A^{P_{Strat}}}$ *is* D_1^p*-complete.*

Proof. Our logic

$$L_A^{P_{Strat}} = (\mathsf{WF}_A^{P_{Strat}}, \mathsf{BS}_A^{P_{Strat}}, \mathsf{INC}_A^{P_{Strat}}, \mathsf{ACC}_A^{P_{Strat}})$$

is similar to

$$L_A^{\mathrm{ASP}_{k=0}} = (\mathsf{WF}_A^{\mathrm{ASP}_{k=0}}, \mathsf{BS}_A^{\mathrm{ASP}}, \mathsf{INC}_A^{\mathrm{ASP}}, \mathsf{ACC}_A^{\mathrm{ASP}})$$

with some minor modifications: in addition to rules r, $\mathsf{WF}_A^{P_{Strat}}$ contains tuples $(l, k(l))$ where l is a literal and $k(l)$ a nonnegative integer. The integer $k(l)$ is the stratum corresponding to l. Hence, formally a knowledge base $\mathcal{K}$ is of the form $\mathcal{K} = P \cup S$ where P is a set of rules of the form

$$r: \quad l_0 \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n.$$

and S (S for *strata*) is a set of tuples of the form $(l, k(l))$ with $l \in A$. Now given a knowledge base $\mathcal{K} = P \cup S$ we check in polynomial time that for every literal l occurring in P, exactly one tuple $(l, k(l))$ is contained in S. Then, for every rule (except for the constraints occurring in P)

$$r: \quad l_0 \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n.$$

we check in polynomial time that

- $\|l_0\| \geq \|l_i\|$ for $i = 1, ..., m$,
- $\|l_0\| > \|l_j\|$ for $j = m + 1, ..., n$,

i. e., that the program is stratified and the integers $k(l)$ correspond to appropriate strata for the atoms. If there is a literal l occurring in the program such that $(l, k(l))$ is not contained or occurring twice for two different integers $k_1(l)$ and $k_2(l)$ in S, then $\mathcal{K}$ is considered inconsistent. Otherwise, $\mathcal{K}$ is consistent if and only if the (stratified) program P has an answer set, i. e., all constraints are satisfied by the unique answer set of P. This check is done in polynomial time as well (Apt et al., 1988) and hence $\text{SAT}_{L_A^{P_{Strat}}}$ is in $\mathsf{P} = \Sigma_0^p$.

Now we utilize the construction given in the proof of Theorem 4.9 to see the hardness results. □

As a second example, we consider disjunctive logic programs i. e., the logic L_A^{ASP} as introduced in Section 2.3. Due to (Eiter and Gottlob, 1995), deciding whether a given disjunctive program is consistent is Σ_2^p-complete.

Theorem 4.12. *It holds that*

(a) the problem S-INC$_{L_A^{ASP}}$ *is* Π_2^p*-complete,*

(b) the problem MIN-S-INC$_{L_A^{ASP}}$ *is* D_2^p*-complete.*

Proof. (Membership) The membership statements follow from Theorems 4.4 and 4.5 since $\text{SAT}_{L_A^{\text{ASP}}}$ is in Σ_2^p.

(Hardness) In (Eiter and Gottlob, 1995), it has been shown that deciding whether a disjunctive logic program is consistent is Σ_2^p-complete. We sketch the proof since we are going to make use of the construction.

Assume we are given a QBF

$$\Phi = \exists X \forall Y \, \phi$$

where ϕ is in 3-DNF, i. e., $\phi = C_1 \vee \ldots \vee C_r$ with $C_k = l_{k,1} \wedge \ldots \wedge l_{k,3}$. Let $X = \{x, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$. We introduce fresh atoms $\{x'_1, \ldots, x'_n\}$ and $\{y'_1, \ldots, y'_m\}$ corresponding to the classical negation of the atoms. We let σ be the appropriate mapping, i. e.,

$$\sigma(l) = \begin{cases} l & \text{if } l \text{ is of the form } x_i \text{ or } y_j, \\ x'_i & \text{if } l \text{ is of the form } \neg x_i, \\ y'_j & \text{if } l \text{ is of the form } \neg y_j. \end{cases}$$

We consider the following program P.

$$
\begin{array}{ll}
P: & \\
x_i \vee x'_i. & i = 1, ..., n \\
y_j \vee y'_j. & j = 1, ..., m \\
y_j \leftarrow w. & j = 1, ..., m \\
y'_j \leftarrow w. & j = 1, ..., m \\
w \leftarrow \sigma(l_{k,1}), \sigma(l_{k,2}), \sigma(l_{k,3}). & k = 1, ..., r \\
\leftarrow \text{not } w. &
\end{array}
$$

Now, in order for a set M to be an answer set of P, w needs to be contained in M. Since $w \in M$, all y-variables have to be in M as well. Thus, in order for M to be a minimal model, it needs to be possible to entail w for *any* choice of the y-variables (that is the translation of the universal quantifier). We refer the reader to (Eiter and Gottlob, 1995) for more details.

Now, for (b), i. e., the hardness of MIN-S-INC$_{L_A^{\text{ASP}}}$ in D_2^p, we proceed as in the proof of Theorem 4.1. We assume we are given an instance of the generic D_2^p-complete problem, i. e., two formulas

$$\Phi_i = \exists X_i \forall Y_i \, \phi_i, \qquad i = 1, 2$$

where we have to decide whether it holds that Φ_1 is false while Φ_2 is true. We construct programs (P_1, H_1) that are a "yes" instance of MIN-S-INC$_{L_A^{\text{ASP}}}$ if and only if Φ_1 is false, programs (P_2, H_2) that are a "yes" instance of MIN-S-INC$_{L_A^{\text{ASP}}}$ if and only if Φ_2 is true and then we show how to combine them to one instance (Q, G) of MIN-S-INC$_{L_A^{\text{ASP}}}$ which is a positive one if and only if both (P_1, H_1) and (P_2, H_2) are. Utilizing the construction of (P_1, H_1) yields (a).

As in the proof of Theorem 4.1, we omit the indices and denote the formula by Φ for both constructions. We assume w. l. o. g. that both formulas are over disjoint sets of variables. So consider

$$\Phi = \exists X \forall Y \, \phi$$

as above. The reason why we have to adjust the construction given in (Eiter and Gottlob, 1995) is the translation of the universal quantifier. The rules "$y_j \leftarrow w$." and "$y'_j \leftarrow w$." make sure that the formula is true for any choice of the y-variables. However, given H, one might construct H' with $H \subseteq H' \subseteq P$ such that H' does

not contain all rules of this form. Then, the universal quantifier is not translated correctly.

To solve this issue, we allow entailment of w only if all y_j and y'_j are true. Thus, we introduce the atom w^* as a tool to obtain the y-variables. They, in turn, allow entailment of w. So we construct P_1 and H_1 as follows.

$$
\begin{array}{ll}
P_1: & \\
x_i \vee x'_i. & i = 1, ..., n \\
y_j \vee y'_j. & j = 1, ..., m \\
y_j \leftarrow w^*. & j = 1, ..., m \\
y'_j \leftarrow w^*. & j = 1, ..., m \\
w^* \leftarrow \sigma(l_{k,1}), \sigma(l_{k,2}), \sigma(l_{k,3}). & k = 1, ..., r \\
w \leftarrow y_1,\ y'_1, \ldots, y_m,\ y'_m. & \\
\leftarrow \text{not } w. & \\
H_1: & \\
\leftarrow \text{not } w. &
\end{array}
$$

We can already argue for hardness of S-INC$_{L_A^{\text{ASP}}}$ in Π_2^p.

(a) Deciding whether Φ is true is Σ_2^p-complete in general. Since P_1 is strongly P_1-inconsistent if and only if it is inconsistent itself, we see (as in (Eiter and Gottlob, 1995)): Φ is false if and only if (P_1, P_1) is a "yes" instance of S-INC$_{L_A^{\text{ASP}}}$. Hence, the latter is Π_2^p-hard in general.

(b) We claim that Φ is false if and only if H_1 is minimal strongly P_1-inconsistent.

"$\Rightarrow$": Minimality is clear since H_1 consists only of one rule. Now assume H_1 is *not* strongly P_1-inconsistent. Then, there is a program $H_1 \subseteq H'_1 \subseteq P_1$ that is consistent. We argue that in this case, P_1 itself is consistent. The program H'_1 must contain "$w \leftarrow y_1,\ z_1, \ldots, y_m,\ z_m$.". One can see that any answer set M of H'_1 has to contain $y_1, y'_1, \ldots, y_m, y'_m$. Hence, the rules of the form "$y_j \leftarrow w$." and "$y'_j \leftarrow w$." can all assumed to be contained in H'_1. It is easy to see that the remaining rules do not introduce inconsistency. Hence, P_1 is consistent. Thus, Φ is true (cf. (Eiter and Gottlob, 1995)), a contradiction.

"$\Leftarrow$": Assume H_1 is minimal strongly inconsistent. Then, P_1 itself is inconsistent. Hence, Φ is false.

As already mentioned, we need a second program. As stipulated above, we denote the given QBF by Φ again. We make use of the same notations as above. Recall, however, that we assume the formulas to be over disjoint sets of variables. Also note that our goal is now to obtain programs (P_2, H_2) that are a positive instance of MIN-S-INC$_{L_A^{\text{ASP}}}$ if and only if Φ is true. Consider:

$$
\begin{array}{ll}
P_2: & \\
x_i \vee x_i'. & i = 1, ..., n \\
y_j \vee y_j'. & j = 1, ..., m \\
y_j \leftarrow v^*. & j = 1, ..., m \\
y_j' \leftarrow v^*. & j = 1, ..., m \\
v^* \leftarrow \sigma(l_{k,1}), \sigma(l_{k,2}), \sigma(l_{k,3}). & k = 1, ..., r \\
v \leftarrow y_1,\ y_1', \ldots, y_m,\ y_m'. & \\
\leftarrow \text{not } v. & \\
\leftarrow v. & \\
H_2: & \\
\leftarrow \text{not } v. & \\
\leftarrow v. &
\end{array}
$$

The situation is similar except that now, H_2 is strongly P_2-inconsistent in any case. Furthermore, the subprogram $\{\leftarrow v.\}$ is consistent. So, we have:

$$
\begin{aligned}
& H_2 \textit{ minimal} \text{ strongly } P_2\text{-inconsistent} \\
\Leftrightarrow\ & \{\leftarrow \text{not } v.\} \textit{ not} \text{ strongly } P_2\text{-inconsistent} \\
\Leftrightarrow\ & P_2 \setminus \{\leftarrow v.\} \text{ consistent} \\
\Leftrightarrow\ & \Phi \text{ true.}
\end{aligned}
$$

Combining:

Now assume we are given two formulas Φ_1 and Φ_2 as above over disjoint sets of variables. Construct P_1 and P_2 as above, except the occurring constraints. Let P be the union of both programs, i. e.,

$$P = (P_1 \cup P_2) \setminus \{\leftarrow \text{not } w., \leftarrow \text{not } v., \leftarrow v.\}$$

Consider Q and G given as follows.

Q :

P

$\leftarrow$ not v, not w. $\qquad\qquad$ $\leftarrow v$, not w.

G :

$\leftarrow$ not v, not w. $\qquad\qquad$ $\leftarrow v$, not w.

Using the considerations above, we can easily verify that G is minimal strongly Q-inconsistent if and only if Φ_1 (responsible for entailment of w) is false while Φ_2 (responsible for entailment of v) is true.

To summarize, (Q, G) is a "yes" instance of MIN-S-INC$_{L_A^{\text{ASP}}}$ if and only if Φ_1 is false while Φ_2 is true. We thus obtain the desired hardness result.

□

We turn to argumentation frameworks under stable model semantics, i. e., the logic L_A^{AAF} introduced in Section 2.4. It turns out that we obtain similar results. Recall that deciding whether a stable extension exists is NP-complete in general (Dunne and Wooldridge, 2009).

Theorem 4.13. *It holds that*

(a) the problem S-INC$_{L_A^{AAF}}$ *is coNP-complete and*

(b) the problem MIN-S-INC$_{L_A^{AAF}}$ *is* D_1^p*-complete.*

Proof. Membership follows from Theorems 4.4 and 4.5, so we need to show hardness.

Assume we are given a formula Φ in 3-CNF, i. e., the conjunction of $C_1, ..., C_r$ with $C_k = l_{k,1} \vee \ldots \vee l_{k,3}$. Let $a_1, ..., a_n$ be the atoms occurring in Φ. We construct a framework $AF = (A, R)$ as follows, cf. similar constructions in (Dimopoulos and Torres, 1996). A contains an element for each literal and each clause in ϕ, i. e.,

$$A = \{a_1, \ldots, a_n, \neg a_1, \ldots, \neg a_n, C_1, \ldots, C_r\}.$$

The relation R shall ensure that a stable extension corresponds to a satisfying assignment. Hence, we include

$$(a_1, \neg a_1), \ldots, (a_n, \neg a_n), (\neg a_1, a_1), \ldots, (\neg a_n, a_n)$$

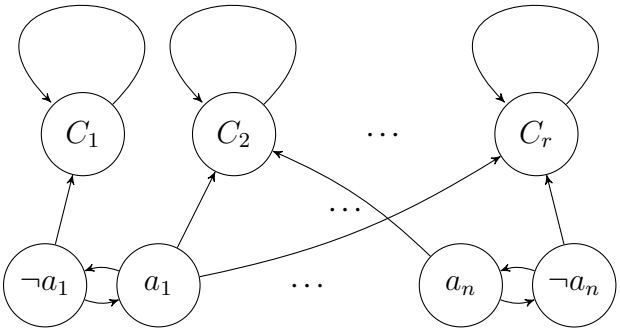

Figure 4: Construction for Arg

to make sure each stable model contains either a_i or $\neg a_i$ for any stable model. Moreover, all clauses need to be satisfied which we translate as "need to be attacked", i. e., we include the attacks

$$(C_1, C_1), \ldots, (C_r, C_r).$$

Now, each C_k attacks itself ($k = 1, \ldots, r$). Hence, in any stable model, every C_k is "attacked" from another argument. So, finally, we include that the literals $l_{k,1}, \ldots, l_{k,3}$ are the ones rendering C_k true ("attacking C_k"):

$$(l_{k,1}, C_k), \ldots, (l_{k,3}, C_k), \quad k = 1, \ldots, r.$$

To summarize, R is given as follows.

$$\begin{aligned} R \ &= \{(a_1, \neg a_1), \ldots, (a_n, \neg a_n), (\neg a_1, a_1), \ldots, (\neg a_n, a_n)\} && \text{(assignment)} \\ &\cup \{(C_1, C_1), \ldots, (C_r, C_r)\} && \text{(each } C_i \text{ has to be attacked)} \\ &\cup \{(l_{1,1}, C_1), \ldots, (l_{1,3}, C_1), \ldots, (l_{r,1}, C_r), \ldots, (l_{r,3}, C_r)\} && \text{(attacked if true)} \end{aligned}$$

Observe that Φ is satisfiable if and only if AF has a stable extension (Dimopoulos and Torres, 1996), i. e., if AF is consistent wrt. stable semantics. Consider $AF_H = (A_H, R_H) \subseteq AF = (A, R)$ given as $A_H = \{C_1, \ldots, C_r\}$ and $R_H = \{(C_1, C_1), \ldots, (C_r, C_r)\}$. Note that AF_H is inconsistent (wrt. stable semantics).

(a) It is easy to verify that AF_H is *not* strongly AF-inconsistent iff there is a satisfying assignment to Φ. Hence, we obtain hardness for S-INC$_{L_A^{\text{AAF}}}$ in coNP.

Input: a knowledge base $\mathcal{K}$
Result: $SI(\mathcal{K})$
$n := |\mathcal{K}|$; $H := \emptyset$; $H' := \emptyset$;
if $\mathcal{K}$ inconsistent **then** $H' := \{\mathcal{K}\}$;
while $H \neq H'$ **do**
  $n := n - 1$; $H := H'$; $New := \emptyset$;
  for *each* $S \in H$ *with* $|S| = n + 1$ **do**
    for *each* $S' \subseteq S$ *with* $|S'| = n$ **do**
      if S' inconsistent and
      $S' \cup \{\phi\} \in H$ for each $\phi \in \mathcal{K} \setminus S'$
      then $New := New \cup \{S'\}$;
    end
  end
  $H' := H' \cup New$;
end
return H.

Algorithm 1: A generic algorithm for computing $SI(\mathcal{K})$

(b) It is also straightforward to see that (AF, AF_H) is a "yes" instance of MIN-S-INC$_{L_A^{\text{AAF}}}$ iff Φ is one for MU. We thus obtain D_1^p-hardness. □

4.4. A Generic Algorithm

To conclude this discussion on computational complexity, we present a generic algorithm for computing $SI(\mathcal{K})$. Algorithm 1 computes strongly $\mathcal{K}$-inconsistent subsets in the order of decreasing cardinality, starting with $\mathcal{K}$. It is based on the observation that a proper subset S of $\mathcal{K}$ can only be strongly $\mathcal{K}$-inconsistent if all subsets of $\mathcal{K}$ which contain one additional element are also strongly $\mathcal{K}$-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 $SI_{min}(\mathcal{K})$ by deleting non-minimal elements whenever New is added to H'.

Proposition 4.14. *Algorithm 1 is sound, complete, and has runtime* $O(2^n * n * f(n))$ *where* $f(n)$ *is the runtime of an algorithm for checking consistency in the given logic.*

Proof. In order to prove soundness let H be the result of applying Algorithm 1 on $\mathcal{K}$ and let $S \in H$. We have to show that $S \in SI(\mathcal{K})$. If $S = \mathcal{K}$ then S has been added to H before the **while**-loop because $\mathcal{K}$ is inconsistent. By definition it follows $S \in SI(\mathcal{K})$. If $S \neq \mathcal{K}$ then S has been added to H at the end of the **while**-loop. This is due to the fact that S is inconsistent and, by induction, each union of S with another formula is strongly $\mathcal{K}$-inconsistent (second **if**-statement). It follows that S is strongly $\mathcal{K}$-inconsistent as well. For completeness, let $T \in SI(\mathcal{K})$. Then there is a chain $T = T_0 \subsetneq T_1 \subsetneq \ldots \subsetneq T_{k-1} \subsetneq T_k = \mathcal{K}$ such that $T_0, \ldots, T_k \in SI(\mathcal{K})$ (note that this statement actually holds for all such chains). As $T_k = \mathcal{K}$ is strongly $\mathcal{K}$-inconsistent it is inconsistent as well and added to H before the **while**-loop. By induction, each T_i (in reverse order) is found in the following **while**-loops as all subsets of the given cardinality are tested for inconsistency.

Let now $f(n)$ be the runtime of an algorithm for checking consistency. First, observe that the worst-case runtime of Algorithm 1 is attained when $SI(\mathcal{K}) = 2^{\mathcal{K}} \setminus \{\emptyset\}$, i. e., all subsets of $\mathcal{K}$ (except the empty set) are strongly $\mathcal{K}$-inconsistent. Then the first **for**-loop is iterated exactly once for each $S \in SI(\mathcal{K})$—i. e. $2^{|\mathcal{K}|}-1 = 2^n - 1$ times— during the execution of the algorithm as it considers all sets with decreasing cardinality (note that the actual number of iterations of the outer **while**-loop is thus irrelevant for the runtime analysis). For each $S \in SI(\mathcal{K})$ we then consider each subset of S with cardinality $|S| - 1$ of which there are at most $|\mathcal{K}| = n$ many. For each of those one consistency check with runtime $f(n)$ is executed and at most $|\mathcal{K}| = n$ many member checks (of constant runtime) are performed. In total we have that Algorithm 1 has runtime $O(2^n * n * f(n))$. □

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 diagnosis and inconsistency measurement.

5.1. Diagnosis and Repair

Theorem 3.6 already shows how consistency of a knowledge base $\mathcal{K}$ 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 $\mathcal{K}$.

Example 5.1. Consider the following logic program P_{13}:

$$\begin{array}{lllll} P_{13}: & a \leftarrow \text{not } b. & b \leftarrow \text{not } c. & a \leftarrow \text{not } d. & e. \\ & c \leftarrow \text{not } a. & & d. & \neg e. \end{array}$$

Minimal inconsistent subsets of P_{13} are $M_1 = \{a \leftarrow \text{not } b., b \leftarrow \text{not } c., c \leftarrow \text{not } a.\}$ and $M_2 = \{e., \neg e.\}$. Whereas M_2 is also minimal strongly inconsistent, M_1 is not as adding the rule "$a \leftarrow \text{not } d.$" resolves inconsistency. The second minimal strongly inconsistent subset is $M_3 = M_1 \cup \{d\}$. Hitting sets consist of one element of M_3 and one of M_2. The program P_{13} can be repaired by deleting the rules in any of these 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 restore consistency (as long as the knowledge base does not contain a general strongly inconsistent subset). 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. Secondly, 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 $SD \cup Obs \cup \{ab(c) \mid c \in C\}$ 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.

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, 2016) 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 $Inc_{\mathsf{MI}}(\mathcal{K}) = |I_{min}(\mathcal{K})|$ for a classical knowledge base $\mathcal{K}$. This measure already complies with some basic ideas of inconsistency measurement, in particular $Inc_{\mathsf{MI}}(\mathcal{K}) = 0$ iff $\mathcal{K}$ 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 Inc_{MI} is not applicable as a consistent nonmonotonic knowledge base $\mathcal{K}$ may contain minimal inconsistent subsets, recall the logic program $P_4 = \{a \leftarrow \text{not } a, \text{not } b., b.\}$ 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 Inc_{MI}.

Definition 5.2. Define Inc_{MSI} via $Inc_{\mathsf{MSI}}(\mathcal{K}) = |SI_{min}(\mathcal{K})|$ for every knowledge base $\mathcal{K}$.

If $\mathcal{K}$ is weakly monotonic then $Inc_{\mathsf{MSI}}(\mathcal{K}) = Inc_{\mathsf{MI}}(\mathcal{K})$ due to Proposition 3.5, item 2. So the measure Inc_{MSI} faithfully generalises Inc_{MI} to all kinds of logics.

Example 5.3. For the logic program $P_4 = \{a \leftarrow \text{not } a, \text{not } b., b.\}$ we obtain $Inc_{\mathsf{MSI}}(P_4) = 0$, despite the fact that P_4 contains a (classical) minimal inconsistent subset. For $P_{14} = \{a., \neg a., b \leftarrow \text{not } b.\}$ we have $Inc_{\mathsf{MSI}}(P_{14}) = 2$.

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, 2016). 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 $\mathcal{K}$ be a (monotonic or nonmonotonic) knowledge base.*

Consistency $Inc_{\mathsf{MSI}}(\mathcal{K}) = 0$ *if and only if* $\mathcal{K}$ *is consistent.*

Independence *If* $\alpha \in Ntr(\mathcal{K})$ *then* $Inc_{\mathsf{MSI}}(\mathcal{K}) = Inc_{\mathsf{MSI}}(\mathcal{K} \setminus \{\alpha\})$.

Separability *If* $SI_{min}(\mathcal{K}_1 \cup \mathcal{K}_2) = SI_{min}(\mathcal{K}_1) \cup SI_{min}(\mathcal{K}_2)$ *and* $SI_{min}(\mathcal{K}_1) \cap SI_{min}(\mathcal{K}_2) = \emptyset$ *then* $Inc_{\mathsf{MSI}}(\mathcal{K}_1 \cup \mathcal{K}_2) = Inc_{\mathsf{MSI}}(\mathcal{K}_1) + Inc_{\mathsf{MSI}}(\mathcal{K}_2)$.

Proof. Consistency follows directly from Proposition 3.5, item 3. *Independence* holds since for any $H \subseteq \mathcal{K}$, H is minimal strongly $\mathcal{K}$-inconsistent if and only if $H \setminus \{\alpha\}$ is (because α cannot introduce or resolve inconsistency). *Separability* follows from the definition of Inc_{MSI}. □

The measure Inc_{MSI} violates one important property though, which is usually demanded for classical measures: the *monotonicity postulate*. This postulate requires $Inc(\mathcal{K}) \leq Inc(\mathcal{K}')$ whenever $\mathcal{K} \subseteq \mathcal{K}'$ 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.

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 rather encouraging complexity results for problems related to strong inconsistency, presented a generic algorithm for computing (minimal) strongly inconsistent subsets of a knowledge base, 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 (Bienvenu et al., 2016) for a recent approach),

we are not aware of any work addressing the issues we studied in this paper – with one notable exception: in (Eiter et al., 2014) Thomas Eiter and colleagues have studied ways of restoring consistency in multi-context systems (Brewka and Eiter, 2007). They focus on the case where the source of inconsistency can be attributed to the bridge rules of a multi-context system. These nonmonotonic rules are similar to rules in normal logic programs, but may refer to beliefs held in other contexts. They are thus able to facilitate the information exchange between different contexts. Inconsistencies originating from bridge rules are repaired in two different ways: by rule deletions and by eliminations of rule bodies, which amounts to treating rule heads as unconditional facts. Since we are not dealing with modifications of formulas in knowledge bases nor with inconsistency handling via new formulas outside the knowledge base, the notion most relevant to our work is *deletion-explanation* (Eiter et al., 2014, Def. 7). A subset E of the bridge rules is a deletion-explanation if the multi-context system at hand is inconsistent for each subset of its bridge rules containing E. This is obviously strong inconsistency in a restricted setting. In a nutshell, our work generalizes what Eiter and colleagues have done for bridge rules in multi-context systems to arbitrary logics, albeit neglecting the possibility of modifying rather than deleting formulas in the knowledge base.

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

Agrawal, R., Srikant, R., 1994. Fast algorithms for mining association rules in large databases. In: VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases. pp. 487–499.
URL http://www.vldb.org/conf/1994/P487.PDF

Apt, K. R., Blair, H. A., Walker, A., 1988. Towards a theory of declarative knowledge. In: Foundations of Deductive Databases and Logic Programming. Morgan Kaufmann, pp. 89–148.

Bertossi, L. E., Hunter, A., Schaub, T. (Eds.), 2005. Inconsistency Tolerance. Vol. 3300 of Lecture Notes in Computer Science. Springer.

Bienvenu, M., Bourgaux, C., Goasdoué, F., 2016. Query-driven repairing of inconsistent dl-lite knowledge bases. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016. pp. 957–964.
URL http://www.ijcai.org/Abstract/16/140

Birnbaum, E., Lozinskii, E. L., 2003. Consistent subsets of inconsistent systems: structure and behaviour. Journal of Experimental & Theoretical Artificial Intelligence 15 (1), 25–46.

Brewka, G., Eiter, T., 2007. Equilibria in heterogeneous nonmonotonic multicontext systems. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada. pp. 385–390.

Brewka, G., Eiter, T., Truszczynski, M., 2011. Answer set programming at a glance. Commun. ACM 54 (12), 92–103.
URL http://doi.acm.org/10.1145/2043174.2043195

de Kleer, J., Mackworth, A. K., Reiter, R., 1992. Characterizing diagnoses and systems. Artif. Intell. 56 (2-3), 197–222.
URL http://dx.doi.org/10.1016/0004-3702(92)90027-U

Dimopoulos, Y., Torres, A., 1996. Graph theoretical structures in logic programs and default theories. Theoretical Computer Science 170 (1–2), 209–244.

Dung, P. M., 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. 77 (2), 321–358.
URL http://dx.doi.org/10.1016/0004-3702(94)00041-X

Dunne, P. E., Wooldridge, M., 2009. Complexity of Abstract Argumentation. Springer US, Boston, MA, Ch. 5, pp. 85–104.

Eiter, T., Fink, M., Schüller, P., Weinzierl, A., 2014. Finding explanations of inconsistency in multi-context systems. Artif. Intell. 216, 233–274.
URL https://doi.org/10.1016/j.artint.2014.07.008

Eiter, T., Fink, M., Tompits, H., Woltran, S., 2005. Strong and uniform equivalence in answer-set programming: Characterizations and complexity results for the non-ground case. In: Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA. pp. 695–700.
URL http://www.aaai.org/Library/AAAI/2005/aaai05-109.php

Eiter, T., Gottlob, G., 1995. On the computational cost of disjunctive logic programming: Propositional case. Annals of Mathematics and Artificial Intelligence 15 (3-4), 289–323.

Gelfond, M., Leone, N., 2002. Logic programming and knowledge representation – the A-Prolog perspective. Artificial Intelligence 138 (1–2), 3–38.

Gelfond, M., Lifschitz, V., 1991. Classical negation in logic programs and disjunctive databases. New Generation Comput. 9 (3/4), 365–386.
URL http://dx.doi.org/10.1007/BF03037169

Hemaspaandra, L. A., Vollmer, H., 1995. The satanic notations: counting classes beyond# p and other definitional adventures. ACM SIGACT News 26 (1), 2–13.

Hunter, A., Konieczny, S., 2008. Measuring inconsistency through minimal inconsistent sets. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008. pp. 358–366.
URL http://www.aaai.org/Library/KR/2008/kr08-035.php

Jabbour, S., Ma, Y., Raddaoui, B., Sais, L., Salhi, Y., 2016. A MIS Partition Based Framework for Measuring Inconsistency. In: Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR'16). pp. 84–93.

Jabbour, S., Sais, L., 2016. Exploiting MUS Structure to Measure Inconsistency of Knowledge Bases. In: Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI'16). pp. 991–998.

Lifschitz, V., Pearce, D., Valverde, A., 2001. Strongly equivalent logic programs. ACM Trans. Comput. Log. 2 (4), 526–541.
URL http://doi.acm.org/10.1145/502166.502170

Oikarinen, E., Woltran, S., 2011. Characterizing strong equivalence for argumentation frameworks. Artif. Intell. 175 (14-15), 1985–2009.
URL http://dx.doi.org/10.1016/j.artint.2011.06.003

Papadimitriou, C., 1994. Computational Complexity. Addison-Wesley.

Papadimitriou, C. H., Wolfe, D., 1988. The complexity of facets resolved. Journal of Computer and System Sciences 37 (1), 2–13.

Priest, G., 1979. Logic of Paradox. Journal of Philosophical Logic 8, 219–241.

Reiter, R., 1980. A logic for default reasoning. Artif. Intell. 13 (1-2), 81–132.
URL http://dx.doi.org/10.1016/0004-3702(80)90014-4

Reiter, R., 1987. A theory of diagnosis from first principles. Artif. Intell. 32 (1), 57–95.
URL http://dx.doi.org/10.1016/0004-3702(87)90062-2

Thimm, M., August 2016. On the compliance of rationality postulates for inconsistency measures: A more or less complete picture. Künstliche Intelligenz.

Ulbricht, M., Thimm, M., Brewka, G., November 2016. Measuring Inconsistency in Answer Set Programs. In: Proceedings of the 15th European Conference on Logics in Artificial Intelligence (JELIA'16). pp. 577–583.

Valiant, L. G., 1979. The complexity of computing the permanent. Theoretical computer science 8 (2), 189–201.