Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Survey Track

Reasoning About Inconsistent Formulas

Joao Marques-Silva[1] and **Carlos Mencía**[2]
[1]ANITI, University of Toulouse, France
[2]University of Oviedo, Spain

Abstract

The analysis of inconsistent formulas finds an ever-increasing range of applications, that include axiom pinpointing in description logics, fault localization in software, model-based diagnosis, optimization problems, but also explainability of machine learning models. This paper overviews approaches for analyzing inconsistent formulas, focusing on finding and enumerating explanations of and corrections for inconsistency, but also on solving optimization problems modeled as inconsistent formulas.

1 Introduction

In a wide range of settings, one must reason about inconsistent formulas. This is the case for example when debugging inconsistent system specifications, e.g. for understanding which constraints make the system inconsistent. This is also the case in model-based diagnosis [Reiter, 1987], where the system representation is inconsistent with its expected input-output behavior. In turn, model-based diagnosis finds a number of significant applications, that include axiom pinpointing in description logics, software fault localization, type error debugging, spreadsheet debugging, or analysis of inconsistent knowledge bases, to name a few. Inconsistency is also often used when solving optimization problems, e.g. in the case of maximum satisfiability for propositional domains.

In this paper we focus on inconsistent formulas of monotonic logics, e.g. first-order logic and its fragments. However, there is also interest in non-monotonic logics [Brewka *et al.*, 2019]. When analyzing inconsistent formulas, a number of computational problems arise. In some settings, one is interested in *explaining* one possible reason of inconsistency. In some other settings, the goal is to propose ways to *correct* the source of inconsistency. Finally, in yet some other settings, the purpose is to *enumerate* all explanations or corrections of an inconsistent formula.

The purpose of this paper is to provide an overview of recent work on reasoning about inconsistent formulas. The paper addresses the problems of explaining and correcting inconsistency, but also highlights enumeration of explanations and corrections, and briefly overviews how inconsistency is exploited when solving optimization problems. The paper also demonstrates that most of the algorithms proposed in recent years for reasoning about inconsistency, most of which originally devised in the context of analyzing propositional formulas, can be also be applied in *any* monotonic logic.

The paper is organized as follows. Section 2 introduces the definitions and notation used throughout the paper. Section 3 covers approaches for finding (or extracting) subset-minimal explanations and corrections. Section 4 overviews the relationship between reasoning about inconsistency and optimization problems. Section 5 highlights algorithms for the enumeration of both corrections and explanations. A sample of practical applications is briefly discussed in Section 6. Finally, the paper concludes in Section 7.

2 Preliminaries

Basic definitions. The paper assumes definitions standard for first-order logic (FOL) [Ben-Ari, 2012]. We will be interested in formulas expressed in some (decidable) fragment of FOL[1], and so respecting *monotonicity* of entailment. For simplicity, we will not consider the use of function symbols, but that does not affect the results in the paper. Throughout, FOL formulas will be represented in clausal form and will most often be inconsistent. In general, we consider an inconsistent FOL formula $\mathcal{F}$, such that some clauses in $\mathcal{F}$ can be *relaxed* (i.e. allowed not to be satisfied) to restore consistency, whereas others cannot. Thus, we assume that $\mathcal{F}$ is partitioned into two first-order subformulas $\mathcal{F} = \mathcal{B} \cup \mathcal{S}$, where $\mathcal{S}$ contains the *relaxable* (*soft*) clauses, and $\mathcal{B}$ contains the *non-relaxable* clauses. $\mathcal{B}$ can be viewed as background knowledge, which is assumed to be consistent and must always be satisfied.

The paper uses other standard notation. $\mathcal{P} \models \mathcal{Q}$ denotes that any model of $\mathcal{P}$ is also a model of $\mathcal{Q}$. As a result, $\mathcal{P} \models \bot$ is used to denote that $\mathcal{P}$ is inconsistent, and $\mathcal{P} \not\models \bot$ denotes that $\mathcal{P}$ is consistent. Set notation will be used to improve readability. For example, $\mathcal{P} \cup \mathcal{Q} \models \mathcal{R}$ is used to denote $\wedge_{c \in \mathcal{P} \cup \mathcal{Q}} c \models \mathcal{R}$.

Generalizations from the propositional domain. This paper covers the analysis of inconsistent formulas in very generic terms, opting not to focus exclusively on propositional formulas. A number of concepts often used in the

[1]As a result, the results presented in the paper apply in different settings, including SAT, ILP, CSP, QBF, SMT (specific theories), among others. Decidability is expected, since we will need to be able to answer decision problems both positively and negatively.

propositional domain can be easily generalized to different fragments of first-order logic. These will prove instrumental in showing how different algorithms, often described for propositional formulas, can in fact be generalized well beyond. A simple observation is that propositional variables can be viewed as predicate symbols. Thus, concepts such as reification of clauses, or constraints over propositional variables, can be used in more general settings. For example, pseudo-boolean constraints, including cardinality constraints, can be readily used in fragments of FOL. Furthermore, although QBF may seem to involve non-standard quantification (from a FOL perspective), there are well-known translations from QBF to EPR [Seidl *et al.*, 2012], and so the proposed generalizations also extend to QBF.

Subset-minimal explanations and corrections. Given an inconsistent clausal formula $\mathcal{F}$, we are interested in identifying the clauses that are responsible for unsatisfiability among those that can be relaxed, as defined next.

Definition 1 (MUS). *Let $\mathcal{F} = \mathcal{B} \cup \mathcal{S}$ be an inconsistent set of clauses ($\mathcal{F} \models \bot$). $\mathcal{M} \subseteq \mathcal{S}$ is a* Minimal Unsatisfiable Subset *(MUS) iff $\mathcal{B} \cup \mathcal{M} \models \bot$ and $\forall_{\mathcal{M}' \subsetneq \mathcal{M}}$, $\mathcal{B} \cup \mathcal{M}' \not\models \bot$.*

Informally, an MUS provides the minimal information that needs to be added to the background knowledge $\mathcal{B}$ to obtain inconsistency; thus, an MUS represents an explanation for the causes of inconsistency. Alternatively, one might be interested in correcting the formula, removing some clauses to achieve consistency.

Definition 2 (MCS). *Let $\mathcal{F} = \mathcal{B} \cup \mathcal{S}$ be an inconsistent set of clauses ($\mathcal{F} \models \bot$). $\mathcal{C} \subseteq \mathcal{S}$ is a* Minimal Correction Subset *(MCS) iff $\mathcal{B} \cup \mathcal{S} \setminus \mathcal{C} \not\models \bot$ and $\forall_{\mathcal{C}' \subsetneq \mathcal{C}}$, $\mathcal{B} \cup \mathcal{S} \setminus \mathcal{C}' \models \bot$.*

With each MCS $\mathcal{C}$, one associates a Maximal Satisfiable Subset (MSS), given by $\mathcal{S} \setminus \mathcal{C}$. Moreover, there exists a well-known (subset-)minimal hitting set (MHS) relationship between MUSes and MCSes:

Proposition 1. *MCSes are MHSes of MUSes and vice-versa.*

The MHS relationship between MUSes and MCSes was first demonstrated in the context of model-based diagnosis [Reiter, 1987] and later investigated for propositional formulas in clausal form [Birnbaum and Lozinskii, 2003].

Recent years have witnessed the proposal of a large number of novel algorithms for the extraction and enumeration of MUSes and MCSes [Mencía *et al.*, 2015; Bacchus and Katsirelos, 2015; Liffiton *et al.*, 2016; Bacchus and Katsirelos, 2016; Mencía *et al.*, 2016; Previti *et al.*, 2018; Grégoire *et al.*, 2018; Narodytska *et al.*, 2018; Bendík *et al.*, 2018], where MUS enumeration algorithms build on the MHS relationship between MUSes and MCSes.

Tractable theories have also been studied, as Horn formulas [Marques-Silva *et al.*, 2016]. In addition, there has been interest in computing the union of all MUSes [Mencía *et al.*, 2019], which can be approximated by the so-called *lean kernel* [Kleine Büning and Kullmann, 2009; Kullmann and Marques-Silva, 2015; Peñaloza *et al.*, 2017].

Minimal sets over monotone predicates (MSMP). The similarity among solutions for a number of apparently unrelated problems motivated the proposal of a unifying paradigm, MSMP [Marques-Silva *et al.*, 2013b; Marques-Silva *et al.*, 2017], consisting in computing minimal sets over a monotone predicate. In this setting, a predicate P: $2^{\mathcal{R}} \rightarrow \{0, 1\}$, defined over a *reference set* $\mathcal{R}$, is *monotone* if whenever $P(\mathcal{R}_0)$ holds, with $\mathcal{R}_0 \subseteq \mathcal{R}$, then $P(\mathcal{R}_1)$ also holds, with $\mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \mathcal{R}$. By defining different problems as instances of MSMP, one can devise algorithms that can be used to solve different problems, many of which are related with the analysis of inconsistent formulas. Earlier work [Marques-Silva *et al.*, 2017] proposed a partition of MSMP problems into three classes, depending on the type of monotone predicate considered. (Due to space constraints, the predicate forms are only hinted at in this paper.)

Maximum satisfiability and optimization problems. Given $\mathcal{F} = \mathcal{B} \cup \mathcal{S}$, with $\mathcal{B} \cup \mathcal{S} \models \bot$, the maximum satisfiability (MaxSAT) problem asks for the largest subset $\mathcal{S}'$ of $\mathcal{S}$, such that $\mathcal{B} \cup \mathcal{S}' \not\models \bot$. A cost can be associated with each clause in $\mathcal{S}$, and so the MaxSAT problem corresponds instead to selecting the subset $\mathcal{S}'$ of the largest cost. There is a well-known relationship between MaxSAT solutions and MSSes, in that every MaxSAT solution is an MSS, and so the complement of a MaxSAT solution is an MCS. Furthermore, it is well-known that standard optimization problems can be cast as MaxSAT (where MaxSAT can be interpreted at the propositional level, but also within any fragment of FOL):

$$\begin{array}{ll} \min & \sum_i w_i \times p_i \\ \text{s.t.} & \mathcal{M} \end{array} \tag{1}$$

where $w_i \geq 0$, and each p_i can be viewed as a 0-place predicate. In this case, just let $\mathcal{B} \triangleq \mathcal{M}$, $\mathcal{S} \triangleq \{(\neg p_i)\}$ and associate with each clause c_i in $\mathcal{S}$ a cost w_i. Thus, $\mathcal{F} = \mathcal{B} \cup \mathcal{S}$ is inconsistent and an MSS (in terms of cost) of $\mathcal{F}$ is a solution to the optimization problem.

Running example. Figure 1 shows an example[2], used throughout this paper to illustrate the operation of algorithms for extracting and enumerating MUSes and MCSes. As it can be observed, the example is written in Bernays-Schönfinkel's fragment of first-order logic, also referred to as *effectively propositional logic* (EPR)[3]. It is well-known that EPR is NEXP-complete [Lewis, 1980]. In this example, we have a knowledge base (see Figure 1a) with a number of general statements about parents (P) of new newborn children (N) being proud (R), and that fathers or mothers are parents. We also indicate that there should be no proud people (with the objective of finding proud parents). Moreover, we also have statements about concrete people $\{\mathsf{aw}, \mathsf{rw}, \mathsf{hw}, \mathsf{pw}, \mathsf{mw}\}$. We declare the general statements as *hard*, denoting that we are certain that the statements are true. In contrast, the information about concrete people is declared as *soft*, denoting that we are uncertain about whether the information is true or not. The set of statements is inconsistent. Using an existing reasoner for EPR [Korovin, 2013] as an oracle, we can identify

[2] Adapted from [Nilsson and Maluszynski, 1995, Section 3.1].

[3] By using EPR, we highlight that most of the proposed algorithms generalize beyond the propositional cases and beyond oracles for NP. There are dedicated algorithms for the analysis of EPR formulas [Xie and Luo, 2016], but these are fundamentally different from the algorithms studied here.

Hard?	Knowledge base	Clause ID's
Yes	$\forall x \forall y.\, (\mathsf{P}(x,y) \wedge \mathsf{N}(y) \rightarrow \mathsf{R}(x))$ $\forall x \forall y.\, (\mathsf{F}(x,y) \rightarrow \mathsf{P}(x,y))$ $\forall x \forall y.\, (\mathsf{M}(x,y) \rightarrow \mathsf{P}(x,y))$ $\forall x.\, (\neg \mathsf{R}(x))$	$\mathcal{B}$
No	$c_1 = \mathsf{F}(\mathsf{rw},\mathsf{hw}), c_2 = \mathsf{N}(\mathsf{hw})$, $c_3 = \mathsf{F}(\mathsf{aw},\mathsf{pw}), c_4 = \mathsf{N}(\mathsf{pw})$, $c_5 = \mathsf{M}(\mathsf{mw},\mathsf{rw}), c_6 = \neg\mathsf{N}(\mathsf{rw})$	$\mathcal{S} = \{\, c_1, c_2, c_3,$ $c_4, c_5, c_6 \,\}$

(a) Example knowledge base

Type	Computed sets
MUSes	$\{\{\mathsf{F}(\mathsf{rw},\mathsf{hw}), \mathsf{N}(\mathsf{hw})\}, \{\mathsf{F}(\mathsf{aw},\mathsf{pw}), \mathsf{N}(\mathsf{pw})\}\}$
MCSes	$\{\{\mathsf{F}(\mathsf{rw},\mathsf{hw}), \mathsf{F}(\mathsf{aw},\mathsf{pw})\}, \{\mathsf{F}(\mathsf{rw},\mathsf{hw}), \mathsf{N}(\mathsf{pw})\},$ $\{\mathsf{F}(\mathsf{aw},\mathsf{pw}), \mathsf{N}(\mathsf{hw})\}, \{\mathsf{N}(\mathsf{hw}), \mathsf{N}(\mathsf{pw})\}\}$

(b) MUSes & MCSes for running example

Figure 1: Running example

Algorithm 1: Deletion-based minimal set computation

Function DELETION (P, $\mathcal{B}$, $\mathcal{S}$, $\mathcal{R}$)

1 $\mathcal{M} \leftarrow \mathcal{R}$;
2 **foreach** $u \in \mathcal{M}$ **do** // Inv: $P(\mathcal{B}, \mathcal{S}, \mathcal{M})$
3 **if** $P(\mathcal{B}, \mathcal{S}, \mathcal{M} \setminus \{u\})$ **then**
4 $\mathcal{M} \leftarrow \mathcal{M} \setminus \{u\}$;
5 **return** $\mathcal{M}$;

two minimal explanations (MUSes) of inconsistency as well as four minimal corrections (MCSes) for eliminating inconsistency. These sets are shown in Figure 1b. The justification for the inconsistencies serve to convey the facts that aw and rw can be inferred to be proud parents, and this is inconsistent with the requirement of no one being proud. As can be observed, the MUSes are MHSes of the MCSes and vice-versa.

3 Subset-Minimal Sets

This section overviews approaches for extracting subset-minimal explanations and corrections of inconsistency[4].

Subset-minimal explanations. Given an inconsistent formula $\mathcal{F} = \mathcal{B} \cup \mathcal{S}$, there is a fairly straightforward algorithm for finding one MUS. Starting from a set of clauses $\mathcal{M} \triangleq \mathcal{S}$, iteratively remove one clause u from $\mathcal{M}$. If $\mathcal{B} \cup (\mathcal{M} - \{u\})$ is inconsistent, then we can safely discard u. Otherwise, we *must* keep u, to preserve inconsistency. After analyzing all clauses of $\mathcal{S}$, the resulting set $\mathcal{M}$ is an MUS of $\mathcal{F}$. Variants of this algorithm have been studied since at least the early 90s [Chinneck and Dravnieks, 1991; Bakker *et al.*, 1993; Dershowitz *et al.*, 2006]. Algorithm 1 represents a possible instantiation of this approach for computing MUSes, but which also exploits the MSMP framework. As a result, one must specify which predicate to use for MUS extraction. Following [Marques-Silva *et al.*, 2013b; Marques-Silva *et al.*, 2017], we use the predicate $P(\mathcal{B}, \mathcal{S}, \mathcal{W}) \triangleq \neg\mathsf{SAT}(\mathcal{B} \wedge \mathcal{W})$, with $\mathcal{W} \subseteq \mathcal{R}$, and $\mathcal{R} \triangleq \mathcal{S}$. The SAT test for the running example needs to decide the satisfiability of an EPR formula. In general, SAT must solve the decision problem for the logic used to represent $\mathcal{B}$ and $\mathcal{R}$. Figure 2 summarizes a possible execution of Algorithm 1 on the formula from Figure 1. Clearly, the computed MUS depends on the order of clauses considered. In this case, the computed MUS is $\{c_3, c_4\} \triangleq \{\mathsf{F}(\mathsf{aw},\mathsf{pw}), \mathsf{N}(\mathsf{pw})\}$ which, together with $\mathcal{B}$, is inconsistent and minimal.

[4]We consider clausal representations. Analysis of non-clausal formulas has also been investigated in the past [Liffiton and Sakallah, 2008; Belov and Marques-Silva, 2011].

$\mathcal{M}$ (line 2)	u	$P(\mathcal{B}, \mathcal{S}, \mathcal{M} \setminus \{u\})$	Action
$\{c_1, c_2, c_3, c_4, c_5, c_6\}$	c_1	1	Drop c_1
$\{c_2, c_3, c_4, c_5, c_6\}$	c_2	1	Drop c_2
$\{c_3, c_4, c_5, c_6\}$	c_3	0	Keep c_3
$\{c_3, c_4, c_5, c_6\}$	c_4	0	Keep c_4
$\{c_3, c_4, c_5, c_6\}$	c_5	1	Drop c_5
$\{c_3, c_4, c_6\}$	c_6	1	Drop c_6
$\{c_3, c_4\}$	–	–	–

Figure 2: Finding MUS with Algorithm 1 on formula from Figure 1

Algorithm	Oracle Calls	Reference
Insertion	$\mathcal{O}(k\,m)$	[de Siqueira N. and Puget, 1988]
MCS_MUS	$\mathcal{O}(k\,m)$	[Bacchus and Katsirelos, 2015]
Deletion	$\mathcal{O}(m)$	[Chinneck and Dravnieks, 1991]
Linear ins.	$\mathcal{O}(m)$	[Belov *et al.*, 2012]
Dichotomic	$\mathcal{O}(k\,\log(m))$	[Hemery *et al.*, 2006]
QuickXplain	$\mathcal{O}(k + k\,\log(\frac{m}{k}))$	[Junker, 2004]
Progression	$\mathcal{O}(k\,\log(1 + \frac{m}{k}))$	[Marques-Silva *et al.*, 2013b]

Table 1: Query complexity of selected MUS algorithms

It is simple to conclude that Algorithm 1 requires $\Theta(|\mathcal{R}|)$ predicate tests. There is no known approach for computing one MUS that improves asymptotically on this bound. A number of algorithms for MUS extraction have been proposed over the years e.g. [Junker, 2004; Marques-Silva *et al.*, 2013b] but without exception, in the worst-case, the number of predicate calls is $\Omega(|\mathcal{S}|)$. Depending on the target decision problem, different optimizations can be envisioned for MUS extraction, e.g. [Belov *et al.*, 2012; Wieringa, 2012; Lonsing and Egly, 2015]. Table 1 summarizes the query complexity of well-known algorithms for extracting a single MUS, where m and k denote the number of clauses in $\mathcal{S}$ and the size of the largest minimal set respectively.

Subset-minimal corrections. Given an inconsistent formula $\mathcal{F} = \mathcal{B} \cup \mathcal{S}$, the following approach can be used for finding one MCS. Starting from a set of clauses $\mathcal{M} \triangleq \emptyset$, iteratively consider one clause $u \in \mathcal{S}$. If $\mathcal{B} \cup \mathcal{M} \cup \{u\}$ is consistent, add u to $\mathcal{M}$. After considering all clauses in $\mathcal{S}$, the clauses that could not be added to $\mathcal{M}$ denote an MCS of $\mathcal{F}$. Such an algorithm has been used since at least the mid 00s [Bailey and Stuckey, 2005], and has more recently

$\mathcal{M}$ (line 2)	u	$P(\mathcal{B}, \mathcal{S}, \mathcal{M} \setminus \{u\})$	Action
$\{c_1, c_2, c_3, c_4, c_5, c_6\}$	c_1	1	Drop c_1
$\{c_2, c_3, c_4, c_5, c_6\}$	c_2	0	Keep c_2
$\{c_2, c_3, c_4, c_5, c_6\}$	c_3	1	Drop c_3
$\{c_2, c_4, c_5, c_6\}$	c_4	0	Keep c_4
$\{c_2, c_4, c_5, c_6\}$	c_5	1	Drop c_5
$\{c_2, c_4, c_6\}$	c_6	1	Drop c_6
$\{c_2, c_4\}$	–	–	–

Figure 3: Finding MCS with Algorithm 1 on formula from Figure 1

Algorithm 2: Clause D minimal set computation

Function CLD (P, $\mathcal{B}$, $\mathcal{S}$, $\mathcal{R}$)

1 $X \leftarrow \emptyset$;
2 $(\mathcal{R}', Y) \leftarrow \texttt{ReifyClauses}(\mathcal{R}, X)$;
3 $D \leftarrow \text{DCLAUSE}(Y)$;
4 **while** true **do**
5 $(st, M) \leftarrow P(\mathcal{B}, \mathcal{S}, \mathcal{R}' \land D)$;
6 **if** $\neg st$ **then return**
$\mathcal{M} \leftarrow \texttt{SetElements}(\mathcal{R}, Y)$;
7 $X \leftarrow \texttt{PickSatisfiedRVars}(M, Y)$;
8 $(\mathcal{R}', Y) \leftarrow \texttt{ReifyClauses}(\mathcal{R}, X)$;
9 $D \leftarrow \text{DCLAUSE}(Y)$;

been referred to as *basic linear search* (BLS) [Marques-Silva *et al.*, 2013a]. As implicit in earlier work [Marques-Silva *et al.*, 2013b; Marques-Silva *et al.*, 2017], by changing the predicate used, Algorithm 1 corresponds to BLS, as outlined above. For this case, the predicate to use is defined as follows: $P(\mathcal{B}, \mathcal{S}, \mathcal{W}) \triangleq \mathsf{SAT}(\mathcal{B} \land (\mathcal{S} \setminus \mathcal{W}))$, with $\mathcal{W} \subseteq \mathcal{R}$ and $\mathcal{R} \triangleq \mathcal{S}$. Figure 3 illustrates the execution of Algorithm 1 for computing an MCS. The computed set is $\{c_2, c_4\} \triangleq \{\mathsf{N}(\mathsf{hw}), \mathsf{N}(\mathsf{pw})\}$. A number of optimizations to the BLS algorithm have been proposed in earlier work [Marques-Silva *et al.*, 2013a; Mencía *et al.*, 2015]. These include, among others, exploiting models to save a number of predicate tests.

BLS is one of a number of algorithms that have been proposed for extracting MCSes. A conceptually simple solution is to find an MCS using maximum satisfiability [Liffiton and Sakallah, 2008]. In practice, solving MaxSAT, even if requiring fewer oracle calls in the worst case, is not as effective as other alternatives. One of the most widely used methods for the extraction of MCSes is the Clause D (CLD) algorithm [Marques-Silva *et al.*, 2013a]. For some kinds of MSMP problems, the CLD algorithm can be used.

Algorithm 2 summarizes the main steps of CLD. One starts by *reifying* each clause $c_i \in \mathcal{S}$ with a fresh *relaxation* variable (or 0-place predicate): $(r_i \lor c_i)$, thus creating a replacement $\mathcal{R}'$ for the original $\mathcal{R}$. Moreover, a disjunction is created with a literal $\neg r_i$ for each r_i variable (this is the D clause). Then, the algorithm iteratively tests $\mathcal{B} \cup \mathcal{R}' \cup \{D\}$. (Observe that this requires at least one r_i to be assigned value 0.) If this formula is satisfiable, then some soft clause(s) c_i can be satisfied without assigning r_i to 1. Thus, we recreate the formula by dropping any r_i assigned value 0 and recreate the D clause without the corresponding literals. The process is repeated until the D clause contains a set of literals that, given the other clauses, cannot be satisfied. These literals identify an MCS[5]. In contrast to the deletion-based approach, as well as to many other algorithms for MUS extraction, CLD expects the oracle call to return a model, from which it can identify the clauses that will no longer be reified. Thus, the predicate call must test for *consistency* and not for *inconsistency*. Nevertheless, as shown in earlier work, a large number of minimal set problems are defined by testing for consistency.

Algorithm	Oracle Calls	Reference
Linear search	$\mathcal{O}(m)$	[Bailey and Stuckey, 2005]
MaxSAT	$\mathcal{O}(\log m)$	[Liffiton and Sakallah, 2008]
Clause D	$\mathcal{O}(m-k)$ [6]	[Marques-Silva *et al.*, 2013a]
FastDiag	$\mathcal{O}(k + k \log \frac{m}{k})$	[Felfernig *et al.*, 2012]
Progression	$\mathcal{O}(k \log(1 + \frac{m}{k}))$	[Marques-Silva *et al.*, 2013b]
LBX[7]	$\mathcal{O}(n)$	[Mencía *et al.*, 2015]
CMP	$\mathcal{O}(m^2)$	[Grégoire *et al.*, 2014]
UCD	$\mathcal{O}(k + k \log(\frac{m}{k}))$	[Mencía *et al.*, 2016]
UBS, LOPZ	$\mathcal{O}(\sqrt{m} \log m)$	[Mencía *et al.*, 2016]

Table 2: Query complexity of selected MCS algorithms

Table 2 summarizes the query complexity of well-known algorithms for extracting a single MCS, where n is the number of variables in $\mathcal{S}$, and m and k have the same meaning as before, with the exception of CLD, where k is the size of the smallest MCS. Moreover, there are algorithms that compute an MCS by modifying the oracle, imposing a fixed branching preference [Rosa *et al.*, 2010; Bacchus *et al.*, 2014].

4 Cardinality- & Preferred Minimal Sets

Solving maximum satisfiability. MaxSAT is solved by finding smallest (cost) MCSes. A well-known approach for solving MaxSAT is branch-and-bound search (B&B) [Li and Manyà, 2009]. Since the mid 00s, experimental evidence from a number of applications showed that B&B does not scale in practice [Morgado *et al.*, 2013; Ansótegui *et al.*, 2013]. An alternative approach consists in reifying all the soft clauses, i.e. replace (c_i), with $c_i \in \mathcal{S}$, with $(r_i \lor c_i)$, obtaining $\mathcal{S}_r$; and then defining a constraint $\mathcal{X} \triangleq \sum_i w_i \times r_i \leq \tau$, where τ ranges from 0 to $\sum_i w_i$. As a result, finding the smallest value of τ, such that $\mathcal{B} \land \mathcal{S}_r \land \mathcal{X}$ is consistent, can then be achieved with linear search [Le Berre and Parrain, 2010], by refining lower or upper bounds on the value of τ, or with binary search [Koshimura *et al.*, 2012]. When the number of soft clauses is large, the representation or handling of $\mathcal{X}$ can be a problem. Motivated by this drawback, two approaches for solving MaxSAT using SAT oracles have been proposed since the mid 00s, both of which reason about inconsistency. Core-guided approaches relax soft clauses

[5] The propositional version admits optimizations, including the fact that reification is unnecessary. Some optimizations are not restricted to the propositional version.

[6] In the propositional case one obtains $\mathcal{O}(\min(n, m-k))$.

[7] LBX can only be used with non-quantified formulas.

on demand (see [Morgado *et al.*, 2013; Ansótegui *et al.*, 2013] for surveys of earlier work and [Martins *et al.*, 2014; Morgado *et al.*, 2014; Alviano *et al.*, 2015] for recent improvements). An alternative is based on the iterative computation of minimum hitting sets (MinHS) on clauses describing (minimal) unsatisfiable cores [Davies and Bacchus, 2011]. Core-guided approaches reify clauses on demand given identified (minimal) unsatisfiable subsets. Cardinality (for unweighted problems) or pseudo-boolean constraints (for weighted problems) are then specified using the relaxation variables and existing lower or upper bounds. MinHS approaches do not reify clauses on demand. Instead, computed (minimal) unsatisfiable subsets serve as sets that must be hit when selecting the next minimum cost subset of clauses whose satisfiability is to be tested. Not surprisingly, both approaches generalize beyond the propositional case.

Smallest explanations. The previous section argued that existing algorithms for computing an MUS require $\Theta(|\mathcal{S}|)$ calls to a suitable oracle. In contrast, computing a cardinality-minimal (or smallest) MUS (SMUS) seems to require solving a computationally harder problem. For the case of propositional logic, deciding whether there exists an MUS of size not greater than some k is complete for Σ_2^{P} [Liberatore, 2005], and so computing a smallest MUS requires a logarithmic number of calls to a Σ_2^{P} oracle. The state of the art in computing smallest explanations is FORQES [Ignatiev *et al.*, 2015]. As with other methods studied in this paper, the FORQES algorithm generalizes beyond the propositional case. Earlier work proposed worst-case exponential propositional encodings for this problem [Lynce and Marques-Silva, 2004].

Computing preferred sets. In propositional logic, preferences of inclusion of clauses in MUSes and MCSes are hard for the second level of the polynomial hierarchy [Marques-Silva and Previti, 2014]. Preferences of non-inclusion of clauses in MUSes and MCSes are in FP^{NP}. Similar increases in complexity are expected to hold beyond the propositional case.

5 Enumeration of Minimal Sets

Enumeration of minimal corrections. Enumeration of MCSes can be achieved by enumerating MaxSAT solutions [Liffiton and Sakallah, 2008]. More recent work proposed dedicated algorithms for MCS enumeration [Marques-Silva *et al.*, 2013a; Bacchus *et al.*, 2014; Previti *et al.*, 2017; Previti *et al.*, 2018; Grégoire *et al.*, 2018; Narodytska *et al.*, 2018]. To implement enumeration of MCSes, one needs to prevent the same MCS from being computed again. This can be done by blocking the selection of the set of clauses corresponding to an MCS. Most recent work focuses on how to optimize the enumeration process.

Enumeration of minimal explanations. To our best knowledge, enumeration of minimal explanations is harder than the enumeration of minimal corrections, and cannot be obtained directly by iterative extraction of MUSes[8]. One well-known solution is the enumeration of all MCSes, from which all the MUSes can be obtained by computing all the minimal hitting sets. This was implemented for example in the CAMUS tool [Liffiton and Sakallah, 2008]. A difficulty of these approaches is that MUSes can be computed only after *all* the MCSes have been computed, which can be exponential in number. A recent alternative is the MARCO algorithm [Liffiton *et al.*, 2016], which iteratively enumerates both MUSes and MCSes, and can be configured such that only one kind of minimal set needs to be extracted.

[8]Implicit enumeration of all possible sets [Reiter, 1987; Bailey and Stuckey, 2005] could be considered, but these approaches do not scale in practice [Liffiton and Sakallah, 2008].

Algorithm 3: Generic MUS/MCS enumeration

Function MINSETENUM (P_{XYZ}, $\mathcal{B}$, $\mathcal{S}$)

$(\mathcal{N}, \mathcal{P}) \leftarrow (\emptyset, \emptyset)$;
while true **do**
  $(st_\lambda, \lambda) \leftarrow \texttt{FindMHS}(\mathcal{N}, \mathcal{P})$;
  if $\neg st_\lambda$ **then break**;
  $(st_\rho, \rho) \leftarrow \texttt{Chk}_{\text{XYZ}}(\mathcal{B}, \mathcal{S}, \texttt{Cls}(\lambda))$;
  if st_ρ **then**
    $\tau \leftarrow \texttt{GetXYZ}(P_{\text{XYZ}}, \mathcal{B}, \mathcal{S}, \mathcal{S} \setminus \texttt{Cls}(\lambda))$;
    $\texttt{ReportXYZ}(\tau)$;
    $\mathcal{N} \leftarrow \mathcal{N} \cup \texttt{NegClIDs}(\tau)$;
  else
    $\texttt{ReportABC}(\texttt{Cls}(\lambda))$;
    $\mathcal{P} \leftarrow \mathcal{P} \cup \texttt{PosClIDs}(\texttt{Cls}(\lambda))$;

ABC	XYZ	$\texttt{Chk}_{\text{XYZ}}$	GetXYZ
MCS	MUS	$\neg\texttt{SAT}(\mathcal{B} \wedge \mathcal{S} \setminus \texttt{Cls}(\lambda))$	MUS
MUS	MCS	$\texttt{SAT}(\mathcal{B} \wedge \texttt{Cls}(\lambda))$	MCS

Table 3: Configuration of Algorithm 3

Algorithm 3 outlines a modified MARCO algorithm, where the preference for a specific kind of minimal set is made explicit. The two configurations are shown in Table 3. Let us focus on the one that gives preference to finding MUSes (the other one is analogous). In this case, XYZ is set to MUS, and ABC to MCS. The algorithm maintains two sets of sets of indices (where index i_j is associated to clause $c_j \in \mathcal{S}$): $\mathcal{N}$, with sets blocking all MUSes found; and $\mathcal{P}$, with sets blocking all MCSes found. Iteratively, a minimal hitting set (MHS) λ of $\mathcal{N}$ subject to $\mathcal{P}$ is computed, which induces the set of clauses $\texttt{Cls}(\lambda)$. Then, if $\neg\texttt{SAT}(\mathcal{B} \wedge \mathcal{S} \setminus \texttt{Cls}(\lambda))$ holds, an MUS is extracted from $\mathcal{S} \setminus \texttt{Cls}(\lambda)$, which is blocked in $\mathcal{N}$. Otherwise, $\texttt{Cls}(\lambda)$ represents an MCS, which is blocked in $\mathcal{P}$. The process continues until no MHS exists, guaranteeing that all MUSes and MCSes have been reported. Figure 4 illustrates the execution of Algorithm 3 on the running example. Improvements to MARCO have been proposed in recent years [Bacchus and Katsirelos, 2016; Narodytska *et al.*, 2018; Bendík *et al.*, 2018]. Besides, MARCO and variants have a growing number of applications [Polikarpova *et al.*, 2016; Rothenberg and Grumberg, 2016; Brandt *et al.*, 2018].

$\mathcal{N}$	$\mathcal{P}$	(st_λ, λ)	st_ρ	MUS/MCS	$\mathcal{N}$ update	$\mathcal{P}$ update
$\emptyset$	$\emptyset$	$(1, \emptyset)$	1	$\{c_1, c_2\}$	$\{\neg i_1, \neg i_2\}$	–
$\{\{\neg i_1, \neg i_2\}\}$	$\emptyset$	$(1, \{i_1\})$	1	$\{c_3, c_4\}$	$\{\neg i_3, \neg i_4\}$	–
$\{\{\neg i_1, \neg i_2\}, \{\neg i_3, \neg i_4\}\}$	$\emptyset$	$(1, \{i_1, i_3\})$	0	$\{c_1, c_3\}$	–	$\{i_1, i_3\}$
$\{\{\neg i_1, \neg i_2\}, \{\neg i_3, \neg i_4\}\}$	$\{\{i_1, i_3\}\}$	$(1, \{i_1, i_4\})$	0	$\{c_1, c_4\}$	–	$\{i_1, i_4\}$
$\{\{\neg i_1, \neg i_2\}, \{\neg i_3, \neg i_4\}\}$	$\{\{i_1, i_3\}, \{i_1, i_4\}\}$	$(1, \{i_2, i_3\})$	0	$\{c_2, c_3\}$	–	$\{i_2, i_3\}$
$\{\{\neg i_1, \neg i_2\}, \{\neg i_3, \neg i_4\}\}$	$\{\{i_1, i_3\}, \{i_1, i_4\}, \{i_2, i_3\}\}$	$(1, \{i_2, i_4\})$	0	$\{c_2, c_4\}$	–	$\{i_2, i_4\}$
$\{\{\neg i_1, \neg i_2\}, \{\neg i_3, \neg i_4\}\}$	$\{\{i_1, i_3\}, \{i_1, i_4\}, \{i_2, i_3\}, \{i_2, i_4\}\}$	$(0, -)$	–	–	–	–

Figure 4: Enumerating MUSes&MCSes with Algorithm 3 on formula from Figure 1, with XYZ set to MUS and ABC set to MCS

6 Example Applications

This section provides a brief glimpse of the applications of reasoning about inconsistency. Optimizing linear cost functions subject to sets of constraints is ubiquitous in a growing range of domains. One well-known application domain is program analysis [Si *et al.*, 2017]. Additional examples include Pseudo-Boolean Optimization but also optimization subject to quantified constraints and multi-objective optimization [Terra-Neves *et al.*, 2017]. The relationship between the analysis of inconsistent formulas and model-based diagnosis (MBD) can be traced to the seminal work of Reiter [Reiter, 1987]. Approaches based on solving MaxSAT or computing MCSes have been proposed [Metodi *et al.*, 2014; Marques-Silva *et al.*, 2015; Ignatiev *et al.*, 2019a]. A concrete instantiation of MBD is axiom pinpointing. In this respect, several of the algorithms presented in this paper have been applied to axiom pinpointing in $\mathcal{EL}^+$ ontologies [Sebastiani and Vescovi, 2009; Arif *et al.*, 2015]. In addition, MUS extraction finds application in formal verification and model checking [McMillan and Amla, 2003; Nadel, 2010]. Explainability is arguably a strategic area in Machine Learning (ML). Most existing approaches for explaining ML models are heuristic, with recent results raising concerns about the global validity of the computed explanations [Ignatiev *et al.*, 2019d]. Recent work relates explanations with prime implicants, and so with analyzing inconsistent formulas [Ignatiev *et al.*, 2019b]. Furthermore, recent work has also shown important links between explanations, adversarial examples and the seminal work of Reiter [Ignatiev *et al.*, 2019c].

7 Conclusions & Research Directions

This paper overviews solutions for the analysis of inconsistent formulas, including the identification of minimal corrections and minimal explanations, but also their enumeration. The paper highlights that most of the algorithms proposed for reasoning about inconsistent formulas generalize beyond the propositional case, being applicable in more expressive logics. Motivated by the many existing applications, the recent fast pace of improvement of algorithms for inconsistency analysis is expected to continue in the near future.

Acknowledgements

This work is supported by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future – PIA3" under Grant agreement n^o ANR-19-PI3A-0004, by the Spanish Government under project TIN2016-79190-R and by the Principality of Asturias under grant IDI/2018/000176.

References

[Alviano *et al.*, 2015] Mario Alviano, Carmine Dodaro, and Francesco Ricca. A MaxSAT algorithm using cardinality constraints of bounded size. In *IJCAI*, pages 2677–2683, 2015.

[Ansótegui *et al.*, 2013] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artif. Intell.*, 196:77–105, 2013.

[Arif *et al.*, 2015] M. Fareed Arif, Carlos Mencía, and Joao Marques-Silva. Efficient MUS enumeration of Horn formulae with applications to axiom pinpointing. In *SAT*, pages 324–342, 2015.

[Bacchus and Katsirelos, 2015] Fahiem Bacchus and George Katsirelos. Using minimal correction sets to more efficiently compute minimal unsatisfiable sets. In *CAV*, pages 70–86, 2015.

[Bacchus and Katsirelos, 2016] Fahiem Bacchus and George Katsirelos. Finding a collection of MUSes incrementally. In *CPAIOR*, pages 35–44, 2016.

[Bacchus *et al.*, 2014] Fahiem Bacchus, Jessica Davies, Maria Tsimpoukelli, and George Katsirelos. Relaxation search: A simple way of managing optional clauses. In *AAAI*, pages 835–841, 2014.

[Bailey and Stuckey, 2005] James Bailey and Peter J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *PADL*, pages 174–186, 2005.

[Bakker *et al.*, 1993] R. R. Bakker, F. Dikker, F. Tempelman, and P. M. Wognum. Diagnosing and solving over-determined constraint satisfaction problems. In *IJCAI*, pages 276–281, 1993.

[Belov and Marques-Silva, 2011] Anton Belov and Joao Marques-Silva. Minimally unsatisfiable boolean circuits. In *SAT*, pages 145–158, 2011.

[Belov *et al.*, 2012] Anton Belov, Inês Lynce, and Joao Marques-Silva. Towards efficient MUS extraction. *AI Commun.*, 25(2):97–116, 2012.

[Ben-Ari, 2012] Mordechai Ben-Ari. *Mathematical Logic for Computer Science*. Springer, 2012.

[Bendík *et al.*, 2018] Jaroslav Bendík, Ivana Cerná, and Nikola Benes. Recursive online enumeration of all minimal unsatisfiable subsets. In *ATVA*, pages 143–159, 2018.

[Birnbaum and Lozinskii, 2003] Elazar Birnbaum and Eliezer L. Lozinskii. Consistent subsets of inconsistent systems: structure and behaviour. *J. Exp. Theor. Artif. Intell.*, 15(1):25–46, 2003.

[Brandt *et al.*, 2018] Felix Brandt, Christian Saile, and Christian Stricker. Voting with ties: Strong impossibilities via SAT solving. In *AAMAS*, pages 1285–1293, 2018.

[Brewka *et al.*, 2019] Gerhard Brewka, Matthias Thimm, and Markus Ulbricht. Strong inconsistency. *Artif. Intell.*, 267:78–117, 2019.

[Chinneck and Dravnieks, 1991] John W. Chinneck and Erik W. Dravnieks. Locating minimal infeasible constraint sets in linear programs. *ORSA Journal on Computing*, 3(2):157–168, 1991.

[Davies and Bacchus, 2011] Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *CP*, pages 225–239, 2011.

[de Siqueira N. and Puget, 1988] J. L. de Siqueira N. and Jean-Francois Puget. Explanation-based generalisation of failures. In *ECAI*, pages 339–344, 1988.

[Dershowitz *et al.*, 2006] Nachum Dershowitz, Ziyad Hanna, and Alexander Nadel. A scalable algorithm for minimal unsatisfiable core extraction. In *SAT*, pages 36–41, 2006.

[Felfernig *et al.*, 2012] Alexander Felfernig, Monika Schubert, and Christoph Zehentner. An efficient diagnosis algorithm for inconsistent constraint sets. *AI EDAM*, 26(1):53–62, 2012.

[Grégoire *et al.*, 2014] Éric Grégoire, Jean-Marie Lagniez, and Bertrand Mazure. An experimentally efficient method for (MSS, CoMSS) partitioning. In *AAAI*, pages 2666–2673, 2014.

[Grégoire *et al.*, 2018] Éric Grégoire, Yacine Izza, and Jean-Marie Lagniez. Boosting MCSes enumeration. In *IJCAI*, pages 1309–1315, 2018.

[Hemery *et al.*, 2006] Fred Hemery, Christophe Lecoutre, Lakhdar Sais, and Frédéric Boussemart. Extracting MUCs from constraint networks. In *ECAI*, pages 113–117, 2006.

[Ignatiev *et al.*, 2015] Alexey Ignatiev, Alessandro Previti, Mark H. Liffiton, and Joao Marques-Silva. Smallest MUS extraction with minimal hitting set dualization. In *CP*, pages 173–182, 2015.

[Ignatiev *et al.*, 2019a] Alexey Ignatiev, António Morgado, Georg Weissenbacher, and Joao Marques-Silva. Model-based diagnosis with multiple observations. In *IJCAI*, pages 1108–1115, 2019.

[Ignatiev *et al.*, 2019b] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019.

[Ignatiev *et al.*, 2019c] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. On relating explanations and adversarial examples. In *NeurIPS*, pages 15857–15867, 2019.

[Ignatiev *et al.*, 2019d] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019.

[Junker, 2004] Ulrich Junker. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In *AAAI*, pages 167–172, 2004.

[Kleine Büning and Kullmann, 2009] Hans Kleine Büning and Oliver Kullmann. Minimal unsatisfiability and autarkies. In *Handbook of Satisfiability*, pages 339–401. 2009.

[Korovin, 2013] Konstantin Korovin. Inst-Gen - A modular approach to instantiation-based automated reasoning. In *Programming Logics - Essays in Memory of Harald Ganzinger*, pages 239–270, 2013.

[Koshimura *et al.*, 2012] Miyuki Koshimura, Tong Zhang, Hiroshi Fujita, and Ryuzo Hasegawa. QMaxSAT: A partial Max-SAT solver. *JSAT*, 8(1/2):95–100, 2012.

[Kullmann and Marques-Silva, 2015] Oliver Kullmann and Joao Marques-Silva. Computing maximal autarkies with few and simple oracle queries. In *SAT*, pages 138–155, 2015.

[Le Berre and Parrain, 2010] Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *J. Satisf. Boolean Model. Comput.*, 7(2-3):59–6, 2010.

[Lewis, 1980] Harry R. Lewis. Complexity results for classes of quantificational formulas. *J. Comput. Syst. Sci.*, 21(3):317–353, 1980.

[Li and Manyà, 2009] Chu Min Li and Felip Manyà. MaxSAT, hard and soft constraints. In *Handbook of Satisfiability*, pages 613–631. 2009.

[Liberatore, 2005] Paolo Liberatore. Redundancy in logic I: CNF propositional formulae. *Artif. Intell.*, 163(2):203–232, 2005.

[Liffiton and Sakallah, 2008] Mark H. Liffiton and Karem A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40(1):1–33, 2008.

[Liffiton *et al.*, 2016] Mark H. Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva. Fast, flexible MUS enumeration. *Constraints*, 21(2):223–250, 2016.

[Lonsing and Egly, 2015] Florian Lonsing and Uwe Egly. Incrementally computing minimal unsatisfiable cores of QBFs via a clause group solver API. In *SAT*, pages 191–198, 2015.

[Lynce and Marques-Silva, 2004] Inês Lynce and Joao Marques-Silva. On computing minimum unsatisfiable cores. In *SAT*, 2004.

[Marques-Silva and Previti, 2014] Joao Marques-Silva and Alessandro Previti. On computing preferred MUSes and MCSes. In *SAT*, pages 58–74, 2014.

[Marques-Silva *et al.*, 2013a] Joao Marques-Silva, Federico Heras, Mikolás Janota, Alessandro Previti, and Anton Belov. On computing minimal correction subsets. In *IJCAI*, pages 615–622, 2013.

[Marques-Silva *et al.*, 2013b] Joao Marques-Silva, Mikolás Janota, and Anton Belov. Minimal sets over monotone predicates in boolean formulae. In *CAV*, pages 592–607, 2013.

[Marques-Silva *et al.*, 2015] Joao Marques-Silva, Mikolás Janota, Alexey Ignatiev, and António Morgado. Efficient model based diagnosis with maximum satisfiability. In *IJCAI*, pages 1966–1972, 2015.

[Marques-Silva *et al.*, 2016] Joao Marques-Silva, Alexey Ignatiev, Carlos Mencía, and Rafael Peñaloza. Efficient reasoning for inconsistent Horn formulae. In *JELIA*, pages 336–352, 2016.

[Marques-Silva *et al.*, 2017] Joao Marques-Silva, Mikolás Janota, and Carlos Mencía. Minimal sets on propositional formulae. problems and reductions. *Artif. Intell.*, 252:22–50, 2017.

[Martins *et al.*, 2014] Ruben Martins, Saurabh Joshi, Vasco M. Manquinho, and Inês Lynce. Incremental cardinality constraints for MaxSAT. In *CP*, pages 531–548, 2014.

[McMillan and Amla, 2003] Kenneth L. McMillan and Nina Amla. Automatic abstraction without counterexamples. In *TACAS*, pages 2–17, 2003.

[Mencía *et al.*, 2015] Carlos Mencía, Alessandro Previti, and Joao Marques-Silva. Literal-based MCS extraction. In *IJCAI*, pages 1973–1979, 2015.

[Mencía *et al.*, 2016] Carlos Mencía, Alexey Ignatiev, Alessandro Previti, and Joao Marques-Silva. MCS extraction with sublinear oracle queries. In *SAT*, pages 342–360, 2016.

[Mencía *et al.*, 2019] Carlos Mencía, Oliver Kullmann, Alexey Ignatiev, and Joao Marques-Silva. On computing the union of MUSes. In *SAT*, pages 211–221, 2019.

[Metodi *et al.*, 2014] Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. A novel SAT-based approach to model based diagnosis. *J. Artif. Intell. Res.*, 51:377–411, 2014.

[Morgado *et al.*, 2013] António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and Joao Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.

[Morgado *et al.*, 2014] António Morgado, Carmine Dodaro, and Joao Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In *CP*, pages 564–573, 2014.

[Nadel, 2010] Alexander Nadel. Boosting minimal unsatisfiable core extraction. In *FMCAD*, pages 221–229, 2010.

[Narodytska *et al.*, 2018] Nina Narodytska, Nikolaj Bjørner, Maria-Cristina V. Marinescu, and Mooly Sagiv. Core-guided minimal correction set and core enumeration. In *IJCAI*, pages 1353–1361, 2018.

[Nilsson and Maluszynski, 1995] Ulf Nilsson and Jan Maluszynski. *Logic, programming and Prolog*. Wiley, 1995.

[Peñaloza *et al.*, 2017] Rafael Peñaloza, Carlos Mencía, Alexey Ignatiev, and Joao Marques-Silva. Lean kernels in description logics. In *ESWC*, pages 518–533, 2017.

[Polikarpova *et al.*, 2016] Nadia Polikarpova, Ivan Kuraj, and Armando Solar-Lezama. Program synthesis from polymorphic refinement types. In *PLDI*, pages 522–538, 2016.

[Previti *et al.*, 2017] Alessandro Previti, Carlos Mencía, Matti Järvisalo, and Joao Marques-Silva. Improving MCS enumeration via caching. In *SAT*, pages 184–194, 2017.

[Previti *et al.*, 2018] Alessandro Previti, Carlos Mencía, Matti Järvisalo, and Joao Marques-Silva. Premise set caching for enumerating minimal correction subsets. In *AAAI*, pages 6633–6640, 2018.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[Rosa *et al.*, 2010] Emanuele Di Rosa, Enrico Giunchiglia, and Marco Maratea. Solving satisfiability problems with preferences. *Constraints*, 15(4):485–515, 2010.

[Rothenberg and Grumberg, 2016] Bat-Chen Rothenberg and Orna Grumberg. Sound and complete mutation-based program repair. In *FM*, pages 593–611, 2016.

[Sebastiani and Vescovi, 2009] Roberto Sebastiani and Michele Vescovi. Axiom pinpointing in lightweight description logics via Horn-SAT encoding and conflict analysis. In *CADE*, pages 84–99, 2009.

[Seidl *et al.*, 2012] Martina Seidl, Florian Lonsing, and Armin Biere. qbf2epr: A tool for generating EPR formulas from QBF. In *PAAR@IJCAR*, pages 139–148, 2012.

[Si *et al.*, 2017] Xujie Si, Xin Zhang, Radu Grigore, and Mayur Naik. Maximum satisfiability in software analysis: Applications and techniques. In *CAV*, pages 68–94, 2017.

[Terra-Neves *et al.*, 2017] Miguel Terra-Neves, Inês Lynce, and Vasco M. Manquinho. Introducing Pareto minimal correction subsets. In *SAT*, pages 195–211, 2017.

[Wieringa, 2012] Siert Wieringa. Understanding, improving and parallelizing MUS finding using model rotation. In *CP*, pages 672–687, 2012.

[Xie and Luo, 2016] Huiyuan Xie and Jie Luo. An algorithm to compute minimal unsatisfiable subsets for a decidable fragment of first-order formulas. In *ICTAI*, pages 444–451, 2016.