Revising non-monotonic theories with sufficient and necessary conditions: the case of Defeasible Logic

FRANCESCO OLIVIERI, *Brisbane, 4000 QLD, Australia.*
E-mail: francesco.olivieriphd@gmail.com

MATTEO CRISTANI, *Department of Computer Science, University of Verona, Verona, 37134, Italy.*
E-mail: matteo.cristani@univr.it

GUIDO GOVERNATORI, *Artificial Intelligence and Cyber Futures Institute, Charles Sturt University, Bathurst, NSW, 2795, Australia.*
E-mail: ggovernatori@csu.edu.au

LUCA PASETTO, *Department of Computer Science, University of Luxembourg, Esch-sur-Alzette, 4365, Luxembourg.*
E-mail: luca.pasetto@uni.lu

ANTONINO ROTOLO, *Department of Juridical Science, University of Bologna, Bologna, 40100, Italy.*
E-mail: antonino.rotolo@unibo.it

SIMONE SCANNAPIECO, *Real T. s.r.l., Verona, 37100, Italy.*
E-mail: simone.scannapieco@realt.it

CLAUDIO TOMAZZOLI, *Department of Computer Science, University of Verona, Verona, 37134, Italy.*
E-mail: claudio.tomazzoli@univr.it

TEWABE CHEKOLE WORKNEH, *Department of Computer Science, University of Verona, Verona, 37134, Italy.*
E-mail: tewabechekole.workneh@univr.it

Abstract

In the setting of Defeasible Logic, we deal with the problem of revising and contracting a non-monotonic theory while minimizing the number of rules to be removed from the theory itself. The process is based on the notions of a set of rules being *necessary* and *sufficient* in order to prove a claim. The substantial difference among classical and non-monotonic

Vol. 35, No. 7, © The Author(s) 2024. Published by Oxford University Press.

https://doi.org/10.1093/logcom/exae044

reasoning processes makes this issue significant in order to achieve the correct revision processes. We show that the process is however computationally hard, and can be solved in polynomial time on non-deterministic machines.

Keywords: Non-monotonic reasoning, necessity, sufficiency.

1 Introduction

The idea of revising a non-monotonic theory is valued in the community of logic since some time. A non-monotonic theory is a set of knowledge tokens expressed in a logical system whose inferential apparatus is capable of generating new conclusions both when adding new knowledge tokens and when removing them.

As we shall show in Section 2, there has been a significant interest in the construction of revision mechanisms for non-monotonic systems, following a large spectrum of studies in monotonic systems. In particular, revision is the focus of a stream of studies regarding defeasible, and defeasible deontic, systems change from the pioneering study by [5] towards the study by [23], and preference revision in Defeasible Logic by [25]. Although, as shown since the beginning of the aforementioned stream of investigations, the problem of revising a defeasible theory could be easy when no constraint is superimposed, the same problem could be unfeasible when constraints are stated. Changing rules had been the first problem that has been dealt with, and it is quite straightforward when formulated as said above. Analogously, it is not straightforward to devise computationally feasible methods when the sole elements on which we can act are the *preferences* as discussed by [25].

In this paper, we propose a method to revise theories by minimal changes, while only eliminating rules. As we shall prove, this is a computationally hard task. In particular, we discuss the *contraction* and the *revision* problems.

Why are these problems interesting? To be concise, because this makes counter-argumentation more difficult, and therefore is a good strategy for advancing a statement. This has a number of applications, including persuasion dialogue management, strategic argumentation and many others. To illustrate what we actually mean, we introduce here an informal example, where we describe a case of dialogue, in which the contraction of a given assertion by one part is counter-argued by another part.

EXAMPLE 1.1

Two persons are discussing whether John, the defendant in a trial, is actually guilty or not. Alice is convinced that John is not guilty, while Bob believes that John committed the crime.

Bob initiates the dialogue by stating that he knows a person who was in a gun shop, and that person saw John there (we name this fact by 1). Therefore, John may possess a gun, and the victim was killed by a gun. Bob also knows that John was in the area where the crime was committed (3); he also heard that there is a testimony (5).

Alice, instead, knows the very same person who was in the gun shop who saw John there, and this person is short-sighted (2). Alice has heard that John works close to the place where the crime was committed (4). She also knows the victim, who was not a good person. Everybody hated him (6).

Alice and Bob agree that all the knowledge tokens they have are truthful, and that there are some priorities. In particular, they believe that 1 is more relevant than 2, and 3 is more relevant than 4. They also agree that 5 is more truthful than 4, and that 1 is more truthful than 6. If we consider the argument in an abstract way, we may be convinced that John is guilty. The arguments supporting guilt are 1, 3 and 5, the arguments supporting innocence are 2, 4 and 6. Therefore, arguments for guilt are strong enough to beat all arguments for innocence (but only when considered together).

Moreover, we can easily observe that the argument '*John possesses a gun*' is necessary for any argument able to prove guilt. Either along with '*John was in the area*' or along with '*There is a*

witness' we have enough evidence for guilt. If Alice aims at pulling apart the argument by Bob, she only has to argue against '*John possesses a gun*'. Once she convinced Bob that there is not sufficient evidence for this, the guilt of John is not proven.

In Example 2.1, we shall illustrate the concepts informally presented above, with some further extensions, in a formal way.

Based on the concepts of *minimal contraction/revision*, which grounds on the idea of *necessary rules* and *minimally sufficient rule sets* that we freshly introduce in this research, we can provide a useful tool for reasoning about the way in which one could be persuaded of a specific conclusion.

The paper is organized as follows. Section 3 introduces the formalism of Defeasible Logic. In Section 4, we provide a formalization of necessity and sufficiency in the scope of Defeasible Logic. Note that, our current investigation does not take into consideration the whole rule set, but only the 'last derivation step', as discussed above. Accordingly, notions of necessity and sufficiency determine sets of rules whose consequent is the conclusion (literal) we want to prove/refute. In that section, we prove some initial theoretical results on such notions, for instance, uniqueness of the necessary set, or properties of minimal sufficient sets.

Section 5 discusses the basic method for computing the extension of Defeasible Logic and the concepts related to this function. In Section 6, we provide algorithms to compute the necessary set (for a given conclusion, given an input theory) and to determine whether a given rule set is sufficient and minimally sufficient. There, we provide the computational results for such algorithms, and consequently and most importantly, we prove that the related problems of computing the necessary set, and determining whether a set is sufficient or minimally sufficient, can be solved in polynomial time on deterministic machines. To make the paper self-contained, we reported in full the algorithms for the computation of the extension of an input defeasible theory, and briefly described their behaviour in order to help the reader with the comprehension of the new algorithms. Such algorithms also consist of new material, as they have been modified with respect to the standard ones [29] to deal with the problem of contracting the input theory.

Lastly, in Section 7, after providing definitions and some preliminary results for *Contraction Set* and *Revision Set* in the scope of changing a theory with the notions of necessity and minimal sufficiency, we prove that (1) both problems of Minimal Contraction Set (MCSR) and Minimal Revision Set MRSR) are polynomially solvable on non-deterministic machines, and (2) the problems of Computing a Minimal Contraction (MCC) and Computing a Minimal Revision (MCR) are NP-hard. Section 8 discusses related work, and, finally, Section 9 draws some conclusions and discusses further investigations.

2 Background

As mentioned in Section 1, several approaches have been explored to represent minimal changes in the revision process, even when involving non-monotonic theories. These approaches, which in some cases solve unfeasible problems, either reveal formal properties of operators to change the theory, or focus on brute-force methods or very simple ones (when existing) to find a solution to the contraction/revision problems.

There is, generally speaking, an extensive literature on the topic of theory revision, also in non-monotonic reasoning, for instance [14] investigated the usage of answer Set Programming for revision of non-monotonic theories, and [17] deals with the revision process for Abstract Argumentation. There is, however, no method among those above that can be used to compute *minimal change* onto a defeasible theory in order to achieve a given revision goal.

A broad investigation of the topic of revision for non-monotonic theories has been conducted by [57]. Authors detail methods for minimization, based on Reiter's known result on duality of minimal inconsistent and maximal consistent sets. However, Ulbricht and co-authors do not deal with Defeasible Logic, which, as we show in this paper, has some features that differentiate the problem of minimization with respect to the other frameworks they deal with in the above mentioned research.

Revising non-monotonic theories is substantially different from revising monotonic theories: deleting rules may lead to *more* conclusions, while adding rules may lead to *less* conclusions. In this work, we focus on the formalism of sceptical non-monotonic reasoning known as *Defeasible Logic*. The reasons behind our choice are three:

- It allows us to draw meaningful conclusions in cases of partial or contrasting information.
- It has been extended to represent, and reason with, knowledge from different areas: from legal systems to business process modelling and compliance verification, from argumentation to multi-agent systems.
- It has been recently implemented in Java in a computationally efficient system named Houdini [9].

Defeasible Logic makes use of a rule-based inferential mechanism, where a rule has a set of premises and derives a single conclusion as a consequence of the premises themselves. Rules are prioritized by means of a *superiority* (or *preference*) *relation* that compares rules with opposite conclusions to determine which of the 'two opposite conclusions' is proved in a particular context. The logical apparatus is then completed with indisputable *facts* that are considered true *a priori*; from a computational point of view, facts are the 'inputs' that distinguish one context from another.

Based on the scepticism dynamics, it may appear that the revision of non-monotonic systems is unnecessary. However, there is an area of application, as we informally discussed in Section 1, where it is instead mandatory to reason with revision: *argumentation*. In argumentation, we need to provide *convincing* support to a thesis in a way that consists in stating an argument to persuade the counterpart of a discussion that we are right. For instance, when we look at the derivation process of a position in a tribunal, one party tries to convince the judge that only some parts of the admissible arguments may be considered.

[25] discussed the concept from one simple point of view: when this argumentation process takes place, there are configurations where neither the law (i.e., the rules), nor the facts, can be changed. Therefore, only preference revision is admitted for those cases.

For instance, there exists an argument in favour, and one argument against, the proof of guilt. This is enough to prove that guilt is not derived (i.e., not-guilty is derived). In the literature of the field of Defeasible Logic, this method of arguing is referred to as *strategic argumentation* [27].

In non-monotonic systems, the existence of a set of rules that apply to several, different sets of facts (scenarios) is common. Therefore, the actual notion of *revision* of facts is cabled onto the structure of non-monotonic systems' inferential apparatus. On the other hand, [25] dealt with the issue of revising the superiority relation. What about rules? Rules are (apparently) simpler cases.

Generally speaking, the revision problem has been the focus of several studies, in many systems. The original referential research by [1] discussed specific problems and methods to solve them. Revisions are classified as follows:

- *Contraction* when we aim at making the theory unable to derive something it is deriving in its unrevised form;
- *Expansion* when we aim at making the theory able to derive something it is not deriving in its unrevised form;

– *Revision*, when we aim at making the theory unable to derive something it is deriving in its unrevised form, while also introducing something it is not derived in the unrevised one.

The actual concept of revision defined above is known as Levi Identity, introduced by [39] and further completely formalized by [34].

In the scope of Defeasible Logic, let us focus on the cases where we can modify a theory by only changing the rules. In the contraction problem, there are two possible configurations:

1. If we can add rules, then only adding one new rule for the opposite will prevent the derivation of the undesired conclusion.
2. We can *not* add new rules but only remove them: we naively remove all supporting rules (for the undesired conclusion).

The above simple, and polynomial, solutions to the contraction problem are applicable in a similar way to the revision problem. We consider revision as the process of making the theory (i) not deriving the undesired conclusion anymore, while (ii) simultaneously obtaining the opposite conclusion.

However, now there are cases where the solution based on only removing rules is not applicable: if there are no rules for that conclusion to begin with, there is no way of deriving it without adding a new rule. When, instead, a rule for the opposite conclusion exists in the theory, we can derive it by removing all the rules for the undesired conclusion.

From the above reasoning, it results that we can decide how to change theory rules consistently with a revision goal in a polynomial time. Moreover, it is clear that we do not need to restructure the theory in its entirety: it is enough to change the *rules to conclude the desired (or undesired) conclusion or its opposite*.

It is also clear that the above operations *do not produce* a minimal change. To discharge the opponent's viewpoint, we should advance a limited number of rule changes; otherwise, it will be difficult to be persuasive; strategically, an argumentation based on the change to too many rules can be harmful. Secondly, changing too many rules can lead to an impoverished theory, which has been reduced also by unnecessary cuts to the rules and is no longer able to derive other conclusions that could have been preserved if different choices were made.

Revision and contraction procedures imply, in classical logic, elimination and substitution of axioms. In non-monotonic reasoning with Defeasible Logic, the action of removing and substituting rules is not equivalent to the same procedures in classical systems. The main reason is that retaining, or dropping, an axiom is a matter of choice that entirely depends on the ability of that axiom to prove (or disprove) a theorem considered relevant, in one of the four senses in which we mean this:

1. remove it for it proves the theorem;
2. leave it for it proves the theorem;
3. remove it for it disproves the theorem;
4. leave it for it disproves the theorem.

For non-monotonic defeasible reasoning, things are more complicated. First of all, we cannot decide what to remove or substitute, based on the ability of *one single* element to prove a result. In fact, when an axiom is used to prove a consequence, and it is necessary for that specific purpose, removing it would be enough to disprove the consequence. We need to reason on *sets of rules*, as suggested by the concept of *team defeat* [24]. Having observed this, we identified the basic brick of this construction: the notion of *sufficiency*, and its counterpart, the notion of *necessity* of rules. These ideas have been explored systematically in classical logic, but, to the best of our knowledge, minimal contraction and revision of defeasible theories by only removing *rules* have been neglected concepts so far.

The main results of this paper, presented in Section 7, regard the solution of the problem of *finding a minimal contraction* and *finding a minimal revision* for a given defeasible theory. We prove that these problems cannot be solved polynomially on deterministic machines (provided that $P \neq NP$). Before proving the aforementioned results, we introduce two notions that are employed for the definition of minimal contraction and revision computation methods: *necessity* and *sufficiency*.

A classical notion of necessity is that any *consequence* of a theorem is a necessary condition. On the other hand, any *set of premises* that can be used to prove a theorem is a sufficient condition. In Defeasible Logic, the concept of theorem is substituted by the concept of *proven literal*, whilst the relation between premises and consequences is cabled into the notion of *rule*. Accordingly, our actual questions will be the following:

- What rules are necessary to prove a given literal?
- What rules are sufficient to prove a given literal?

A rather natural exemplification field is argumentation in court.

EXAMPLE 2.1

In this example, we employ the basics of the formalism of Defeasible Logic. Literals as in classical propositional logic, with negation $\sim$ as usual. The implication $\Rightarrow$ represents a non-monotonic derivation, where antecedents are literals joined in an and by the symbol ',', and the consequent is a single literal. Each element formed in this way is named a *rule*. Rules can lead to contradictory heads, and can be related by superiority relations, represented by $\succ$, meant to represent the concept that if rules r and s are with opposite consequents, then $r \succ s$ lead to the conclusion of the rule r.

We are in court, and there is a theory, advanced by the prosecutor, which proves '*John is guilty*'. In support to this conclusion there are three evidences, admitted for the discussion by the judge: (i) A gun of his property, (ii) A picture of John in the area where the murder was committed, and (iii) A testimony who heard him speaking of the victim with hate. Along with the picture and the testimony, we need to use, specifically, the evaluation made by the investigators to establish that the evidence has been collected *legally*. Along with the possession of the gun, to establish that the possess is effective, we need the owner to be recorded. In a formal sense we claim

$$1: \mathit{John\_gun}, \mathit{registration} \Rightarrow \mathit{guilty}$$

$$3: \mathit{John\_picture}, \mathit{legal} \Rightarrow \mathit{guilty}$$

$$5: \mathit{testimony}, \mathit{legal} \Rightarrow \mathit{guilty}$$

There are some arguments that can be used to retain some of the evidence, and some that can be used to dismiss them. Jennifer, John's solicitor, is providing counter-arguments to the evidence, which is formed by these arguments. The gun has no trace of having being fired for a long period. John was in the area, but John *works* in the area, it is thus rather natural for him to be there. Other witnesses testify that many people state they hated the victim.

$$2: \mathit{John\_gun}, \mathit{registration}, \mathit{not\_fired} \Rightarrow \sim \mathit{guilty}$$

$$4: \mathit{John\_work\_place} \Rightarrow \sim \mathit{guilty}$$

$$6: \mathit{many\_hate} \Rightarrow \sim \mathit{guilty}$$

If the superiority among rules is as follows:

$$1 \succ 2 \qquad 3 \succ 4$$
$$5 \succ 4 \qquad 1 \succ 6$$

we do not need to remove all three rules $1, 3$ and 5 to disprove John's guilt: being 1 the only rule necessary to defeat 2 and 6, removing it is enough. When *necessary* rules exist the removal of these rules contract the theory with respect to guilt. Let us now consider a different priority schema:

$$1 \succ 2 \qquad 1 \succ 4$$
$$3 \succ 4 \qquad 3 \succ 6$$
$$5 \succ 2 \qquad 5 \succ 6.$$

There is not anymore a rule that is necessary to prove the guilt. We have three sets that are sufficient: $\{1, 3\}$ $\{1, 5\}$ and $\{3, 5\}$.

Minimizing the rules to dismiss is a natural task to imagine. In fact, it would be easier to convince someone with *fewer* rules, but simultaneously, we need to reduce the effort to obtain a persuasion, for the more rules we remove, the more arguments to support the removal we need to advance, where each of these arguments needs to be convincing by itself. Moreover, this minimization process also requires that the rules we manipulate are immediately recognized as needed or strong enough to produce the derivation. In fact, even though it may be fine to consider a whole set of rules deriving premises used in other points of a derivation chain, the most significant ones are those that can be used to derive the conclusion at last step, where the attack to the conclusion is most effective. The strength of an argument, in fact, lies on two opposite aspects: simplicity, which we mentioned above, and immediateness. In this paper, we value simplicity in terms of minimality and immediateness as the fact the derivation is stopped at the *last step*. The above concepts are, in a sense, subsumed by the concept of *specificity* that has been considered in argumentation theory since the beginning of the Eighties. In particular, this concept has been introduced in his seminal research by [48]. The concept is developed within argumentation theory by [53] and [55]. More recently, the notion of specificity has been discussed in a formal way by [59].

3 Defeasible Logic

Defeasible Logic (DL) [3, 41] is a simple, flexible and efficient rule-based non-monotonic formalism. Its strength lies in its constructive proof theory, as DL allows to draw meaningful conclusions from a (potentially) conflicting and incomplete knowledge base. In non-monotonic systems, more accurate conclusions can be obtained when more pieces of information become available. Throughout the years, many variants of DL have been proposed for the modelling of different application areas: from legal systems [10, 31, 44] to business process modelling and compliance verification [11, 12, 43, 45], from argumentation [28] to multi-agent systems [29].

We start with the language. Let PROP be a set of propositional atoms, and Lab be a set of arbitrary labels (the names of the rules). The set of *literals* is $\text{Lit} = \text{PROP} \cup \{\neg p \mid p \in \text{PROP}\}$. We shall use lower-case Roman letters denote literals, and lower-case Greek letters to denote rules.

The *complement* (or *opposite*) of a literal l is denoted by $\sim l$. If l is a positive literal p then $\sim l$ is $\neg p$, and if l is a negative literal $\neg p$ then $\sim l$ is p. The definition of a defeasible theory is the standard in DL by([3]).

DEFINITION 3.1 (Defeasible theory).
A *defeasible theory* D is a tuple $(\mathrm{F}, R, \succ)$, being $\mathrm{F} \subseteq$ Lit the set of facts, R the set of rules, and $\succ \subseteq R \times R$ the superiority relation.

The set of facts F denotes simple pieces of information that are considered to be always true, like 'Sylvester is a cat', formally $cat(Sylvester)$. The set of rules R contains three *types* of rules: *strict rules*, *defeasible rules* and *defeaters*. The *superiority relation*[1] $\succ$ is an irreflexive relation to solve conflicts in case of potentially contradicting information. The superiority relation is antitransitive for it connects rules with opposite conclusions, thus when $r \succ s$ and $s \succ t$, r and t have the same conclusion, and consequently cannot be in the relation $\succ$.

A defeasible theory is *finite* if the sets of facts and rules are finite.

A *rule* $\alpha \in R$ is an expression $\alpha : A(\alpha) \hookrightarrow C(\alpha)$, such that:

1. $\alpha \in$ Lab is the *unique name* of the rule;
2. $A(\alpha) \subseteq$ Lit is the (possibly empty) set of the *antecedents*[2] ;
3. $\hookrightarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$ denotes, resp., strict rules, defeasible rules and defeaters;
4. $C(\alpha) \in$ Lit is the *consequent*, a single literal.

A strict rule is a rule in the classical sense: whenever the premises are indisputable, so is the conclusion. The statement 'All Computing Scientists are humans' is formulated through the strict rule

$$\alpha : CScientist(X) \rightarrow human(X),$$

since there is no exception to it. As in ([3]), we consider only a propositional version of this logic, and we do not take into account function symbols. Every expression with variables represents the finite set of its variable-free instances.

Defeasible rules represent statements that can be defeated by contrary evidence: the statement 'Computing scientists travel to the city of the conference' is thus represented with the defeasible rule

$$\beta : CScientist, PaperAccepted \Rightarrow TravelConference.$$

Defeaters are special rules whose only purpose is to prevent the derivation of the opposite conclusion; the statement 'During pandemic travels might be prohibited' is hence better represented by the defeater

$$\gamma : Pandemic \rightsquigarrow \neg TravelConference$$

than a defeasible rule, as we simply want to 'block' the derivation of travelling.

We use some conventional abbreviations. The set of strict rules is R_{s}, the set of strict and defeasible rules is R_{sd}. $R[l]$ is the rule set whose consequent is literal l. We say that a literal l *appears in* D, if there exists a rule such that l is either one of its antecedents, or the consequent.

A *conclusion of* a defeasible theory D is a *tagged formula*, an expression of the form $\pm\#l$, $l \in$ Lit and $\# \in \{\Delta, \partial\}$, with the following meanings:

- $+\Delta l$ means that l is *strictly proved* in D, i.e. there is a strict proof of l in D;
- $-\Delta l$ means that l is *strictly refuted*, i.e. a strict proof of l does not exist in D;
- $+\partial l$ means that l is *defeasibly proved* in D, i.e. there is a defeasible proof of l in D;

[1] Also referred to as *priority* or *preference relation*.
[2] We shall interchangeably use *antecedent* to refer to $A(\alpha)$.

– $-\partial l$ means that l is *defeasibly refuted*, i.e. a defeasible proof of l does not exist in D.

We refer to $\pm\Delta$ and $\pm\partial$ as *proof tags*. The definition of Δ describes forward chaining of strict rules, while ∂ represents the non-monotonic part of the logic.

Given a defeasible theory D, a *proof P of length n in D* is a finite sequence $P(1), P(2), \ldots, P(n)$ of tagged formulas of the type $+\Delta l$, $-\Delta l$, $+\partial l$, $-\partial l$, where the proof conditions defined hereafter hold; $P(1..n)$ denotes the first n steps of P. Proofs are based on the notions of a rule being *applicable* or *discarded*. A rule is *applicable* when all the elements in its antecedent have been proved; a rule is *discarded* if at least one of such elements has been refuted.

DEFINITION 3.2 (Applicable & Discarded).
Given a defeasible theory $D = (\mathrm{F}, R, \succ)$, a rule $\alpha \in R$ is

– *#-applicable*, $\# \in \{\Delta, \partial\}$ at $P(n+1)$ iff $\forall a_i \in A(\alpha).\ +\#l \in P(1..n)$;
– *#-discarded*, $\# \in \{\Delta, \partial\}$ at $P(n+1)$ iff $\exists a_i \in A(\alpha).\ -\#l \in P(1..n)$.

When not needed or clear from the context, we will omit 'Δ/∂-' and will use the shorten version 'rule is applicable/discarded'. Moreover, if no confusion arises we only say 'Δ/∂-'applicable (or applicable) and omit 'at $P(n+1)$'. The proof conditions hereafter are also standard in DL; we start with strict provability.

A literal is *strictly proved* if (1) it is a fact, or (2) there exists a Δ-applicable strict rule for it. Negative proof tags are omitted as they are obtained by applying the *strong negation principle* to their positive counter-parts [30], which applies the function that simplifies a formula by moving all negations to an innermost position in the resulting formula, replacing the positive tags with the respective negative tags and vice versa. Follows the definition of defeasible provability.

A literal is *defeasibly proved* if (1) it has been strictly proved, or (2.1) the opposite has not been strictly proved and (2.2) there exists a ∂-applicable rule (α for l) such that every attack (β for the opposite $\sim l$) is either (2.3.1) ∂-discarded or (2.3.2) defeated by a ∂-applicable (ζ also for l).

Note that α cannot be a defeater, whereas ζ and β can, as defeaters cannot 'directly' draw conclusions but only support, see Example 4.1 at the beginning of Section 4. This will play a significant role in determining necessary, and (minimally) sufficient, sets of rules.

The last notions of this section are those of extension of a theory, and of equivalence of theories. Informally, an *extension* is everything that is derived and refuted; two theories are *equivalent* when they have the same extension.

DEFINITION 3.3 (Extension and theory equivalence).
Assume two defeasible theories D and D'. The set of positive and negative conclusions of D is the *extension* $E(D) = (+\Delta, -\Delta, +\partial, -\partial)$, where $\pm\# = \{l \mid l \text{ appears in } D \text{ and } D \vdash \pm\#l\}$, $\# \in \{\Delta, \partial\}$, and we use symbol $\vdash$ to indicate that there is a derivation of l from D.

D and D' are *equivalent*, $E(D) \equiv E(D')$, iff they have the same extension.

4 Necessity and sufficiency in Defeasible Logic

From a computational perspective, rules and superiority[3] can be seen as the 'fixed' part of a theory, whereas the set of facts acts as the *input*: two theories with the same rule set and superiority, but with

different facts, may have significantly different extensions, as illustrated in Example 4.1. From now on, we adopt the following convention. For all rules with a positive head we employ labels formed by just *odd numbers*, for rules with a negative head we employ labels formed by *even numbers*.

EXAMPLE 4.1
Consider theory $D = (\mathrm{F}, R, \succ)$, where

$$R = \{1: a \Rightarrow b \qquad 3: b \Rightarrow p \qquad 5: \cdots \rightsquigarrow p$$
$$7: a \rightsquigarrow p \qquad 2: c \rightsquigarrow \neg p \qquad 4: \cdots \rightsquigarrow \neg p\},$$

$$\succ = \{(7 \succ 2), (5 \succ 4), (2 \succ 3), (2 \succ 5)\}.$$

Firstly, note that 4 and 5 are vacuously applicable.

Assume $\mathrm{F} = \{a, c\}$. Then, $D \vdash +\#a, c, \# \in \{\Delta, \partial\}$, by Conditions 1 of both positive proof tags; thus, all remaining rules are ∂-applicable. Given that we have no rules for $\neg b$, then $D \vdash +\partial b$. We conclude $D \vdash +\partial p$ for two reasons: (i) all applicable rules supporting $\neg p$ are defeated, specifically 2 by 7 and 4 by 5, and (ii) one of the applicable rules supporting p is *not* a defeater, i.e. rule 3. The extension $E(D)$ is

$$+\Delta = \{a, c\}, \qquad -\Delta = \{\neg a, b, \neg b, \neg c, e, \neg e, p, \neg p\},$$

$$+\partial = \{a, b, c, p\}, \qquad -\partial = \{\neg a, \neg b, \neg c, e, \neg e, \neg p\}.$$

For the sake of completeness, we have reported the full extension; from now on, we will only show literals appearing in the theory, e.g., omitting $\neg b$ in $-\Delta$ and $-\partial$.

Assume now a set of facts containing only $\mathrm{F} = \{c\}$; our focus is again on p and $\neg p$. Contrary to before, rules 1, 3 and 7 are now discarded; 5 for p is still applicable, and so are 2 and 4 for $\neg p$. We still conclude $D \vdash -\partial\neg p$: it does not matter that 5 is defeated by 2, as there are no strict nor defeasible rules for $\neg p$ and the only supporting active rules are defeaters.

A rule is *necessary* in proving a p when without such a rule the system is no longer able to prove p; the necessary set is made of all necessary rules.

DEFINITION 4.2 (Necessity).
Assume a defeasible theory $D = (\mathrm{F}, R, \succ)$ and a literal $p \in \mathrm{Lit}$, such that $D \vdash +\partial p$. A rule $\alpha \in R[p]$ is *necessary for* p if $D' \vdash -\partial p$, with $D' = (\mathrm{F}, R \setminus \{\alpha\}, \succ)$. Set $\mathrm{Nec}(p) = \{\alpha \in R \mid \alpha \textit{ is necessary for} p\}$ is the *necessary set for* p.

Intuitively, a set of rules is *sufficient* in proving p when (i) '*alone*' it defeats all 'applicable' attacks, and (ii) one of such, applicable, rules is not a defeater.

DEFINITION 4.3 (Sufficiency).
Assume a defeasible theory $D = (\mathrm{F}, R, \succ)$ and a literal $p \in \mathrm{Lit}$, such that $D \vdash +\partial p$. A set $\mathrm{Suf} \subseteq R[p]$ is *sufficient for* p if $D^{\mathrm{Suf}} \vdash +\partial p$, with $D^{\mathrm{Suf}} = (\mathrm{F}, (R \setminus R[p]) \cup \mathrm{Suf}, \succ)$. A set Min is *minimally*

[3] Recent works pointed out that there are many instances where it should actually be dynamically computed [2], as [25] proved that the problem of changing the superiority relation is NP-hard.

sufficient for p if (1) Min is sufficient for p, and (2) there exists no set X sufficient for p such that $X \subset$ Min.

The concepts of *necessary* and *sufficient* correspond (but are defined in an abstract way, on the one hand, and associated to defeasible inference mechanism on the other hand) to the notions of *incision* and kernel in base belief revision as discussed by [49].

When clear from the context, we shall simply refer to α as a necessary rule, and to Nec(p) and Suf(p) as a necessary or a sufficient set.

Consider theory D of Example 4.1 with $\mathrm{F} = \{a, c\}$. Rule 7 is necessary to prove p as no other applicable rule for p defeats 2; for the same reason, 5 is necessary as no other rule defeats 4. $\{5, 7\}$ is a necessary set but it is not a sufficient set to prove p as both rules are defeaters: we need rule 3 as the only applicable defeasible rule. Therefore, a necessary, and minimally sufficient, set for p is $\{3, 5, 7\}$.

Consider now $\mathrm{F} = \{a\}$. Being rule 2 now discarded, 7 is no longer needed to defeat it. Therefore, a necessary, and minimally sufficient, set for p is $\{3, 5\}$.

LEMMA 4.4
The intersection of all sufficient sets for a literal p is the necessary set for p.

PROOF. The claim can be rewritten as follows: a rule α belongs to every sufficient set for a literal p *iff* α is necessary for p. Assume theory $D = (\mathrm{F}, R, \succ)$ such that $D \vdash +\partial p$.

($\Rightarrow$) Consider a rule α that is in every sufficient set for p. Now, since α belongs to *each* sufficient set, it also belongs to any minimally sufficient set. Now consider the theory $D' = (\mathrm{F}, R \setminus \{\alpha\}, \succ)$. Since every set that was minimally sufficient in D lacks α in D', we have that $D' \vdash -\partial p$. Therefore, rule α is necessary for p.

($\Leftarrow$) Assume α is a necessary rule, and that α does not belong to a sufficient set S. By definition of sufficient set, we have $D' \vdash +\partial p$, with $D' = (\mathrm{F}, (R \setminus R[p]) \cup S, \succ)$, such that $\alpha \in R[p] \setminus S$. Thus, α is not a necessary rule for p, a contradiction. □

[8] investigate the ways in which non-monotonic theories could be treated in terms of necessity and sufficiency. The concepts expressed in this paper are applicable, in general, to Defeasible Logic as well. However, non-trivial differences exist. In particular, we mark the basic difference: sufficiency is, in the context of defeasible logic, a monotonic property.

OBSERVATION 4.5
Assume theory $D = (\mathrm{F}, R, \succ)$ such that $D \vdash +\partial p$. If the empty set $\emptyset$ is minimally sufficient for p, then $p \in \mathrm{F}$.

As a consequence of Observation 4.5, we can claim the following.

OBSERVATION 4.6
Assume theory $D = (\mathrm{F}, R, \succ)$ and literal $p \notin \mathrm{F}$. Every sufficient set for p contains a non-empty minimally sufficient set.

PROOF. The proof is by contradiction. Assume that there is a set S that is sufficient for p, and that every subset of S is not minimally sufficient. Then, also singleton subsets of S would be so. Since the only subset of a collection of singletons is the empty set, by Observation 4.5 we have $l \in \mathrm{F}$. This is against the hypothesis. □

LEMMA 4.7
Given a theory D, a literal p and the set Nec(p), such that $D \vdash +\partial p$ and $D \vdash -\Delta p$. For each applicable $\beta \in R[\sim p]$ there exists an applicable $\alpha \in \text{Nec}(p)$ such that $\alpha \succ \beta$.

PROOF. Assume, by contradiction, that there exists $\beta \in R[\sim p]$ such that no α in Nec(p) defeats β. Since $D \vdash +\partial p$ by hypothesis, then by definition of $+\partial$ either β is discarded (Condition 2.3.1) which cannot be the case as against the hypothesis, or by Condition 2.3.2 there would exist an applicable rule $\zeta \in R[p]$ such that $\zeta \succ \beta$ but ζ is not in Nec(p). By definition, every rule that is needed to beat a rule for the opposite of p is in Nec(p) (namely it is necessary). Thus, ζ must belong to Nec(p). □

5 Algorithms for computing the extension of a defeasible theory

Before advancing algorithms and computational results regarding necessity and sufficiency, we report in this section the algorithms for the computation of the extension of a theory given as input. These algorithms are the standard in DL literature [29], and reported here for the sake of self-containment. For space reasons, the algorithms presented in this work do not compute the strict part of the extension: to include that is a mundane task as all the defeasibility checks are not taken into consideration.

Algorithm 1: Algorithm for computing extension.

Input: Defeasible theory D
Output: The defeasible extension $E(D)$

$\pm\partial \leftarrow \emptyset$;
`InitialiseHerbrandBase(`HB`)`;
for $l \in HB$ **do**
  $R[l]_{supp} \leftarrow \emptyset$;
  $R[l]_{infd} \leftarrow \emptyset$;
  $Defeaters[l] \leftarrow \{\alpha \in R[l] | \, \alpha \notin (R_s \cup R_{\text{sd}})\}$;
for $l \in \text{F}$ **do**
  `Prove(`l`)`;
  `Refute(`$\sim l$`)`;
repeat
  $\partial^{\pm} \leftarrow \emptyset$;
  for $l \in HB$ **do**
    if $R[l] = \emptyset$ **then** `Refute(`l`)`;
    if $\exists \alpha \in R[l].\, A(\alpha) = \emptyset$ **then**
      $R[l]_{supp} \leftarrow R[l]_{supp} \cup \{\alpha\}$;
      $R[\sim l]_{infd} \leftarrow R[\sim l]_{infd} \cup \{\beta \in R[\sim l] \, | \, \alpha \succ \beta\}$;
      if $\{\beta \in R[\sim l] | \, \beta \succ \alpha\} = \emptyset$ **then**
        `Refute(`$\sim l$`)`;
        if $(R[\sim l] \setminus R[\sim l]_{infd} = \emptyset) \wedge (R[l]_{supp} \setminus Deafeters[l] \neq \emptyset)$ **then**
          `Prove(`l`)`;
  $\pm\partial \leftarrow \pm\partial \cup \partial^{\pm}$;
until $\partial^{+} = \emptyset$ *and* $\partial^{-} = \emptyset$;
return $E(D) = (\pm\partial)$

Procedure Prove

Input: $l \in \text{Lit}$

1 $\partial^+ \leftarrow \partial^+ \cup \{l\}$;
2 $HB \leftarrow HB \setminus \{l\}$;
3 $R \leftarrow \{A(\zeta) \setminus \{l\} \hookrightarrow C(\zeta) | \zeta \in R\}$;
4 $\succ \leftarrow \succ \setminus \{(\zeta, \psi), (\psi, \zeta) \in \succ | \sim l \in A(\zeta)\}$;

Procedure Refute

Input: $l \in \text{Lit}$

1 $\partial^- \leftarrow \partial^-_\Box \cup \{l\}$;
2 $HB \leftarrow HB \setminus \{l\}$;
3 $R \leftarrow R \setminus \{\zeta \in R | l \in A(\zeta)\}$;
4 $\succ \leftarrow \succ \setminus \{(\zeta, \psi), (\psi, \zeta) \in \succ | l \in A(\zeta)\}$;

Let us now delve in more detail into how Algorithm 1 works. It begins by populating the Herbrand Base and, for every literal l, it initializes: (1) the support set $R[l]_{infd}$ that will contain those rules for l defeated by applicable rules for the opposite, (2) the support set $R[l]_{supp}$ that will contain all the applicable rules for l and (3) the set of defeaters *Defeaters* for l.

Every literal in the set of facts is proved, whilst the opposite is refuted (**for** at Lines 7–9).

The algorithm now enters the main cycle **Repeat-Until** at Lines 10–22. For every literal l in HB, we first verify whether there is any rule supporting it: if not, we refute it (Line 13). Otherwise, if there exists an applicable rule α supporting it (**if** at Lines 14–20), we update the set of applicable rules for it (Line 15), and then the set of opposite rules *defeated* by α itself ($R[\sim l]_{infd}$ at Line 16). The next step is to verify whether there actually exists any rule supporting $\sim l$ stronger than α: if not, $\sim l$ can be refuted (Line 18).

The idea behind the process of computing whether a certain literal is actually provable is implemented in the **if** at Lines 19–20, and is the following: if $D \vdash +\partial l$, eventually the **repeat-until** cycle will have added to $R[\sim l]_{infd}$ enough rules to defeat all opposite supports (and this happens when $R[\sim l] \setminus R[\sim l]_{infd}$ is empty). When this is the case and if there exists at least one applicable supporting rule that is not a defeater (i.e., when $R[l]_{supp} \setminus \textit{Defeaters}[l]$ is *not* empty), we invoke **Prove** on l. The algorithm does not have to check whether the defeater rules for $\sim l$ are applicable or discarded (check at Line 19), but only care that all such rules are defeated. In fact, if a defeated rule for $\sim l$ is actually discarded, it does not affect the provability of l. For this reason, when Algorithm 1 invokes Procedure Prove for a literal l, set $R[l]_{supp}$ is l-sufficient but (possibly) *not minimally* l-sufficient.

At the end of the execution of the **repeat-until** cycle, the extension has been computed and, the following conditions hold true (by design of the algorithm):

- All applicable rules have empty antecedent;
- All discarded rules have been eliminated from the set of rules.

Note that, by implementing hash tables with pointers to rules where a given literal occurs, each rule can be accessed in constant time. We also implement hash tables for the tuples of the superiority relation where a given rule appears as either of the two elements, and thus even those can be accessed in constant time. The result of this implementation leads to the computational analysis reported below and amply discussed and formalized by [3]. We do not report the proofs here.

In order to discuss termination and computational complexity, we first define what is the *size* of a theory D, which is denoted by as $\Sigma(D)$ and is: the number of the occurrences of literals, plus the number of occurrences of rules, plus 1 for every tuple in the superiority relation. We omit D when implied by the context, and just write Σ.

THEOREM 5.1
Algorithm 1 terminates and its complexity is $O(\Sigma)$.

THEOREM 5.2
Algorithm 1 is correct and complete, i.e.

1. $D \vdash +\partial p$ iff $p \in +\partial$ of $E(D), p \in$ Lit;
2. $D \vdash -\partial p$ iff $p \in -\partial$ of $E(D), p \in$ Lit.

Proofs are omitted for readability reasons, and can be found in [29].

6 Methods to process minimal revisions with rules in Defeasible Logic

We now proceed in advancing: Algorithm 2 that computes the necessary set for an input literal p, and Algorithm 3 that determines whether a given input set is sufficient, and/or minimally sufficient, for an input literal p. We assume that both algorithms are run after Algorithm 1, and with the same input defeasible theory.

The results reported here are not direct consequence of the linear complexity of the computation of the extension of a defeasible theory, as discussed in the proofs of Theorem 6.1 and Theorem 6.2.

Algorithm 2: Computing the necessary set

Input: Defeasible theory $D = (\mathrm{F}, R, \succ)$ and a literal p
Output: The necessary set $\mathrm{Nec}(p)$

1 $\mathrm{Nec}(p) \leftarrow \emptyset$;
2 `Algorithm 1`(D);
3 **for** $\alpha \in R_{infd}[\neg p]$ **do**
4 | `Algorithm 1`$(D' = (\mathrm{F}, R \setminus \{\alpha\}, \succ))$;
5 | **if** $p \in -\partial'$ **then** $\mathrm{Nec}(p) \leftarrow \mathrm{Nec}(p) \cup \{\alpha\}$;
6 **if** $Nec(p) \cap R_{\mathrm{sd}} = \emptyset$ **then**
7 | $\mathrm{Nec}(p) \leftarrow \mathrm{Nec}(p) \cup \{\zeta\}$ s.t. $\zeta \in R_{infd}[\neg p] \cap R_{\mathrm{sd}}$;

After Algorithm 1 has finished, $R_{infd}[\neg p]$ stores all the applicable rules for p that defeat at least one applicable rule for $\neg p$. To determine if a rule α is necessary, we simply invoke again Algorithm 1 with the same input theory except for α that is now missing from the set of rules. If, in line with Definition 4.2, p is no longer proved, then α is necessary. Finally, as defeaters cannot alone prove a conclusion, we need to verify that at least one of such rules is a strict or defeasible rule. If not (Line 6), we choose one from the applicable ones in $R_{infd}[\neg p]$ (Line 7).First, the algorithm checks whether the target literal is actually proved with the input theory (Line 2). If so, Algorithm 1 is invoked with set of rules R^{Suf} equals to the rules in R without all the rules in $R_{infd}[\neg p]$ that are not in Suf. If the theory no longer proves p (i.e., $p \in -\partial^{\mathrm{Suf}}$ at Line 5) then Suf is not a sufficient set to prove p, and we exit. Otherwise, we can move on in verifying whether Suf is also minimally sufficient to prove p: it is so if there is no rule α in Suf such that the theory invoked without it (Line 8) still proves p.

Algorithm 3: Deciding whether a given set is sufficient, or minimally sufficient

Input: Defeasible theory $D = (\mathrm{F}, R, \succ)$, a literal p, a set of rules Suf
Output: Verifies if Suf is a, minimally, sufficient set for p

```
1  Algorithm 1(D);
2  if p ∈ −∂^D then Exit;
3  R^Suf ← R \ R_infd[¬p] ∪ Suf;
4  Algorithm 1(D^Suf = (F, R^Suf, ≻));
5  if p ∈ −∂^Suf then
6  |  return Set Suf is not a sufficient set for p;
7  for α ∈ Suf do
8  |  Algorithm 1(D^M = (F, R^Suf \ {α}, ≻));
9  |  if p ∈ +∂^M then
10 |  |  return Set Suf is a sufficient, but not
   |  |    minimal, set for p;
11 return Set Suf is a minimally sufficient set for p;
```

First of all, we discuss the complexity of the problems solved by the algorithms introduced above. In particular, Algorithm 2 is claimed to be correct and complete and to perform polynomially in Theorem 6.1.

THEOREM 6.1
The problem of computing the necessary set Nec(p) of a given defeasible theory of size Σ for a given literal p is $\mathrm{O}(\Sigma^2)$.

The proof is straightforward and therefore omitted for the sake of conciseness.

On the other hand, we cannot compute *all* sufficient (and not even all minimally sufficient) sets in a feasible time, for they constitute an exponential (in the worst case) set of rule sets. However, we can solve polynomially the problem of deciding whether a given set is sufficient, or minimally sufficient, by Algorithm 3.

THEOREM 6.2
The problems of deciding whether a given set of rules with consequent p are sufficient or minimally sufficient is $O(\Sigma^2)$ in the size of the theory Σ.

Algorithms 2 and 3 are employed to build revision methods in Section 7. They compute necessary, and decide about (minimally) sufficient, sets. They do so by computing the extension of the input theory. We remind the reader that: (a) the extension of a defeasible theory is, in a sketch, all that which the theory proves and refutes, and (b) two theories are equivalent when they have the same extension. Such an extension is computed via the following ideas. At each iteration step, the algorithms obtain a *simpler*, *but equivalent* theory than the theory at the previous step. By simpler, we mean that, by proving and refuting literals, we can progressively simplify the rules and the theory itself as follows:

- Every time we prove a literal, we can *remove such a literal* from all the rules where it appears as antecedent.
- Every time we refute a literal, we can *remove every rule* that has such a literal in their antecedent, as they become discarded (Definition 3.2).

Important for this goal is to note that, trivially, a rule with empty antecedent is vacuously applicable. These operations are performed by Procedure Prove at Line , and by Procedure Refute at Line . For a better understanding, consider the following example.

EXAMPLE 6.3
Let $D = (\mathrm{F} = \{a, c\}, R, \succ = \{(2, 3)\})$ be the defeasible theory such that

$$R = \{1 : a \Rightarrow b \qquad 3 : a, c \Rightarrow d$$
$$2 : \neg a \Rightarrow \neg d \qquad 4 : c \Rightarrow \neg d\}.$$

At a certain step, the algorithm will prove a (as it is a fact) and refute $\neg a$. Consequently, we can remove a from $A(1)$ and $A(3)$, as from now on 1's and 3's applicability will no longer depend on a. At an orthogonal level, we can remove 2 from R: since $\neg a \in A(2)$, we conclude that rule 2 is discarded; this leaves rule 4 the only remaining rule for $\neg d$. The simplified theory result is proposed below:[4]

$$R' = \{1 : \emptyset \Rightarrow b \qquad 3 : c \Rightarrow d \qquad 4 : c \Rightarrow \neg d\}.$$

Those two procedures also take care to update the superiority relation, as all discarded rules play no part in the team defeater fight. The superiority is actually fixed, but the algorithm computes, at every iteration, a simpler theory than the one in the previous step, while maintaining the same extension. Thus, the algorithm removes $2 \succ 3$ from the superiority relation.

7 Contraction and revision with Defeasible Rules

In this section, we discuss two problems that are of interest when changing defeasible theories: the *contraction* and the *revision* of literals. The contraction is the process in which we modify a theory D that is able to derive a given literal in a positive way ($D \vdash +\partial p$), in such a way that the new theory D' obtained by the operation of contraction would not derive p anymore in a positive way (i.e., $D' \vdash -\partial p$). The revision operator that subsumes the contraction one, executes the same process, but it terminates with the positive derivation of the opposite literal (i.e., $D' \vdash +\partial \sim p$).

As we shall see below, it is rather easy to show that both these problems, with some very reasonable limitations, can be solved polynomially, if we do not require the contraction/revision to be *minimal*. On the other hand, when we aim at obtaining the desired change in the theory by changing a *minimal set of rules*, the problems are significantly more difficult.

As already discussed widely by [22], while in monotonic systems the *expansion* (namely the process of making a theory derive a new literal which was not derived before) cannot be obtained by contrapositive operation of contracting the opposite, this is possible in non-monotonic systems in general and, in particular, in Defeasible Logic. This means that also the actual revision operator (that can be seen as a contraction followed by an expansion) is not corresponding to a sequence formed by a contraction followed by a contraction of the opposite.

We define below one key concept, the contraction set, which is further used to devise a method for computing minimal contractions.

[4]The proof that the transformations used by the algorithm preserve theory equivalence is omitted for space reason, but the proof are analogous with immediate modifications to those presented in [26].

DEFINITION 7.1 (Contraction Set).
Assume a theory $D = (F, R, \succ)$ and a literal p such that $D \vdash +\partial p$. Set $S_C \subseteq R[p]$ is called a *p-contraction* if $D' \vdash -\partial p$, with $D' = (F, R \setminus S_C, \succ)$. A p-contraction set S_C is said to be *minimal* when no p-contraction subset of S_C can be found.

Rather naturally, a very obvious contraction set can be defined as in the observation below.

OBSERVATION 7.2
Given a theory $D = (F, R, \succ)$ and a literal $p \in$ Lit such that $D \vdash +\partial p$ and $D \vdash -\Delta p$, set $R[p]$ is a p-contraction set.

A slightly more sophisticated contraction is made by the set of applicable rules for p that defeat an applicable rule for $\sim p$.

OBSERVATION 7.3
Given a theory $D = (F, R, \succ)$ and a literal $p \notin F$ such that $D \vdash +\partial p$, the set

$$\{\alpha \in R[p] | \exists \beta \in R[\sim p] \text{ s.t. } \alpha, \beta \text{ are applicable and } \alpha \succ \beta\}$$

is a p-contraction set.

Note that, when considering a minimal contraction, we are not counting the amount of rules appearing in the contraction itself. We are, instead, considering minimality with respect to set inclusion, and, therefore, there could be cases in which a minimal contraction has a larger number of elements than another contraction, potentially not minimal. This is shown in the following example.

EXAMPLE 7.4
Consider a literal p that can be proven by one among three possible minimally sufficient sets M_1, M_2 and M_3, respectively formed by the following rules:

- $M_1 = \{1, 3, 5, 7\}$
- $M_2 = \{1, 5, 9, 11\}$
- $M_3 = \{7, 13, 15, 17\}$

The three sets have no rule in common, but M_1 and M_2 have at least one rule in common. Choosing to drop a rule that belongs to both M_1 and M_2 and one rule belonging to M_3 is enough to disprove the literal p and this would result to be a minimal contraction. For instance, the contraction set

$$C = \{1, 13\}$$

is minimal. However, even if we choose one rule for M_1, one for M_2 (not belonging to the intersection of M_1 and M_2) and one for M_3 we compute in this way a minimal contraction, which has three elements instead of the two elements of the contraction computed above; this is true for

$$C = \{3, 9, 15\},$$

which is a minimal contraction set.

Accordingly, the problem of computing a minimal contraction lies in devising a subset that works appropriately, not in finding the minimum set in terms of cardinality.

Let us now consider the situation that occurs when the set Nec(p) for a p-contraction is not empty. By definition of necessary rules, we can just remove *one* rule chosen at random from *Nec(p)* and

consequently prevent the theory from further deriving the undesired literal. The reasoning above leads to the following:

OBSERVATION 7.5
Given a literal p and a theory D where $D \vdash +\partial p$, when $\text{Nec}(p) \neq \emptyset$, the problem of computing a minimal contraction set is polynomially solvable on deterministic machines.

In fact, removing one single element from the set $\text{Nec}(p)$ is a polynomially solvable task by Theorem 6.1. Once we determine the necessary set, we remove one element from it and the system will not be able to derive p anymore. This revision is formed by one single literal, and therefore, based on Definition 7.1 it is a contraction set and it is minimal, because for any other contraction set C, either C is not comparable or it contains the set formed by the singleton of the removed rule.

We are now able to introduce a connection between minimally sufficient sets for a literal p and minimal contractions.

LEMMA 7.6
Given a theory $D = (F, R, \succ)$, a literal $p \notin F$ such that $D \vdash +\partial p$ and a p-contraction set S_C, for every minimally p-sufficient set $M \subseteq R[p]$ there exists one rule $\alpha \in M$ such that $\alpha \in S_C$.

Similarly, we introduce the notion of minimal revision, starting with the notion of revision set.

DEFINITION 7.7 (Revision Set).
Assume a theory $D = (F, R, \succ)$ and a literal p such that $D \vdash +\partial p$. Set $S_R \subseteq R[p]$ is named a *p-revision* when, for $D' = (F, R \setminus S_R, \succ)$, $D' \vdash +\partial \sim p$. A p-revision set S_R is said to be *minimal* when no p-revision subset of S_R can be found.

With some limitations, we can extend the result of Observation 7.2 to minimal revisions.

OBSERVATION 7.8
Given a theory $D = (F, R, \succ)$ and a literal $p \in \text{Lit}$ such that $D \vdash +\partial p$ and $D \vdash -\Delta p$, set $R[p]$ is a p-revision if (1) $R_{\text{sd}}[\sim p] \neq \emptyset$, and (2) one of such (strict or defeasible) rules $\beta \in R[\sim p]$ is applicable.

Note that an immediate consequence of the above results is that computing a contraction or a revision is a very simple task to achieve. We may just check the mere existence of a rule for the opposite literal, and, in case of the revision, verify that the rule is applicable. When one of the above conditions occurs (as in Definition 7.1 and 7.7), the set $R[p]$ of the rules for p is a contraction or a revision.

We can now start to discuss the computational issues related to minimal contractions and minimal revisions. We start by naming the problems of establishing whether a given set S is a minimal contraction or a minimal revision set, respectively MCSR and MRCR.

The above two problems are proven to be polynomially solvable in Lemmata 7.9 and 7.10 that make use of Algorithm 4. In particular Lemma 7.9 proves that contractions and revisions can be detected in polynomial time, while Lemma 7.10 proves that also minimal contractions and revisions are detected in polynomial time.

LEMMA 7.9
Given a theory $D = (F, R, \succ)$, a literal p such that $D \vdash +\partial p$ and a set $S \subseteq R[p]$, checking whether S is a p-contraction set, or a p-revision set, takes polynomial time.

PROOF.In polynomial time, we can compute the extension of $D' = (F, R \setminus S, \succ)$. Then, S is a p-contraction set iff $D' \vdash -\partial p$, and S is a p-revision set iff $D' \vdash +\partial \sim p$. □

LEMMA 7.10
Given a theory $D = (F, R, \succ)$, a literal p such that $D \vdash +\partial p$ and a p-contraction (or p-revision) set $S \subseteq R[p]$, checking whether S is a minimal p-contraction (or p-revision) set takes polynomial time.

PROOF. The proof works by iteratively removing one element from the set S. Consider the set $S' = S \setminus \{\alpha \mid \alpha \in S\}$ and the defeasible theory $D' = (F, R \setminus S', \succ)$. In polynomial time, we can compute the extension of each D'. If S is a p-contraction set and there exists at least one rule $\alpha \in S$ such that $D' \vdash -\partial p$, then S is not minimal; and if S is a p-revision set and there exists at least one rule $\alpha \in S$ such that $D' \vdash +\partial \sim p$, then S is not minimal. Otherwise, if for every $\alpha \in S$ we have that $D' \vdash +\partial p$, then S is minimal. In the worst case scenario, S is minimal and the algorithm computes the extension of a number of theories corresponding to the number of elements in S. Since computing the extension of a theory takes linear time and the cardinality of S is linear, the entire algorithm takes polynomial time. □

What about, instead, determining a *minimal* contraction/revision? The task appears intuitively more difficult, for the potential need, as proven in Lemma 7.6, to compute all the minimally sufficient sets, in order to extract one rule from each of these, and use them to define the contraction/revision.

Rather naturally, when the set of *necessary* rules is not empty, since every element of the set Nec(p) also belongs to all the minimally p-sufficient sets, then computing a minimal contraction/revision is polynomial, and consists in removing one rule alone from the rules for p that belongs to Nec(p).

We define the following problems:

- The problem of deciding whether a given set of rules for a conclusion p is a minimal contraction within a theory D is named $MCRS(p, D)$.
- The problem of deciding whether a given set of rules for a conclusion p is a minimal revision within a theory D is named $MRRS(p, D)$.
- The problem of computing a minimal contraction within a theory D for a conclusion p is a $MCC(p, D)$.
- The problem of computing a minimal revision within a theory D for a conclusion p is a $MCR(p, D)$.

We need now to make some observations on the nature of the above defined problems. In particular, under the constraint of not performing changes other than removing rules, whilst *MCC* is always solvable, *MCR* is not. Therefore, the first is a pure search problem, and the second is a mixed problem, where we ask to solve the following: Decide whether in a given theory D it is possible to revise the literal p by only removing rules in D, and if it possible, exhibit a minimal change.

We should however notice that the problem of deciding whether a given revision is possible is a polynomial one on deterministic machines, as it corresponds to deciding whether the theory contains at least one rule for the opposite of the undesired literal.

We omit the proof of Lemma 7.11 that is a straightforward consequence of Lemmata 7.9 and 7.10, and the structure of Algorithm 4.

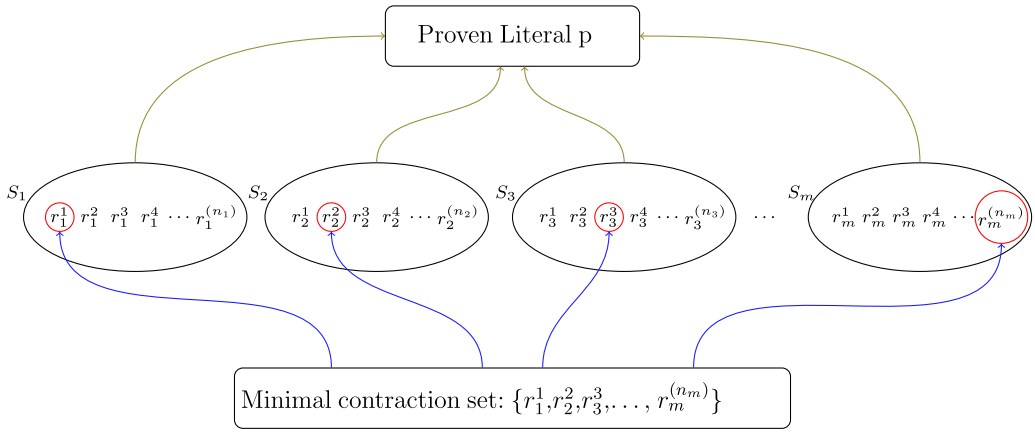

FIGURE 1. Schema of the minimization process for contraction sets.

Algorithm 4: Algorithm to solve MCSR and MRSR problems.

Input: $S \subseteq R[p]$ and a flag toggling C/R for contraction or revision
Output: Yes if S is a minimal contraction/revision, No otherwise

```
for α ∈ S do
    Compute S' = S \ α;
    Decide whether S' is a contraction (revision
      depending on the toggling flag);
if None of the sets is a contraction (revision) then
    Return Yes;
else
    Return No ;
```

LEMMA 7.11
Algorithm 4 correctly decides for every set of rules whether that set is a minimal contraction/revision in $O(n^2)$ with n number of rules.

We are now able to determine the complexity of the problems defined above.

A schema of the way in which the problem of minimizing a contraction set is shown in Figure 1. This schema represents the case of minimally sufficient sets that are pairwise disjoint, which, as we see below, is actually the subset of MCC and MCR problems to which we reduce a known NP-hard problem.

THEOREM 7.12
MCRS and MRRS are polynomially solvable on non-deterministic machines.

PROOF. The contraction/revision process is solved by a non-deterministic program with oracle, which after guessing a candidate set (by such an oracle), applies Algorithm 4 and establishes whether this is a minimal contraction/revision. □

We are finally able to provide a reduction for the problem of *computing a minimal contraction/revision* (MCC, MRC) for a literal p in Defeasible Logic. The method we apply is a many-to-one reduction to the problem of computing an *Exact Hitting Set* for a collection of subsets. The first of the two main results is therefore claimed in Theorem 7.13. To be precise, the reduction is from the *exact hitting problem* to a sub-problem of the minimal contraction that is devised as the problem of finding such a contraction when the minimally sufficient sets are pairwise disjoint. This assumption is not used in the proof, for we do not need it to solve correctly the exact hitting problem. However, we should observe that this is not true on the inverse way. In particular, a minimal contraction can be obtained in a specific way that *does not* correspond to the exact hitting problem, for such a contraction would possibly be non-minimal. In fact, consider the behaviour of the Example 7.4. If we choose an element $x \in M_1$ that is an element of M_2 as well, and then choose, for M_2 an element y that is not in M_1, and finally an element z of M_3 at random, the obtained contraction $\{x, y, z\}$ is not minimal, for the subset $\{x, z\}$ would be minimal as well. Note that, however, finding a minimal contraction for the cases in which it is not true that the minimally sufficient sets are all pairwise disjoint, is, in the worst case, worse than the case in which they are so. Therefore, since the solution that we have guarantees that the problem can be solved polynomially on non-deterministic machines, we have a correct reduction for the above mentioned problem.

THEOREM 7.13
MCC and MCR are NP-hard.

PROOF. Let us consider the *Exact Hitting Problem*, which is formulated as follows: given a universe U and a collection S of subsets of U covering U, namely such that $\bigcup_{x \in S} x = U$, compute a subset H of U such that each subset in the collection S contains exactly one element in H. The problem is known to be NP-hard—see, for instance, the study by [18].

Consider an instance of the Exact Hitting Problem, where we have a collection C of subsets of a universe U. Elements of the universe are enumerated by naturals $1, 2, ..., n$, and elements of the collection are enumerated $c_1, c_2, ..., c_n$. We shall reduce it to the minimal contraction problem (and omit for the sake of space the reduction to the corresponding revision problem that is straightforward). We will construct a defeasible theory from it in a way that will allow us to consequently compute a minimal contraction for a special literal, and then convert this to a solution of the Exact Hitting Problem instance we started with.

Without any loss of generality, we assume that elements of the collection C given as input of the problem are such that for each $x \in C$ there is no $y \in C$ such that $x \subseteq y$. Given a generic input of the Exact Hitting Problem, we can limit ourselves to this sub-case of the problem by checking each element for the above condition, and, in case we find a pair such that $x \subseteq y$, we remove y from the collection. In this correspondence, for the sake of simplicity, we adopt the notation of rules to be represented by lowercase Greek letters indexed by naturals.

We build the defeasible theory $(F, D, \succ)$ in the following way:

(i) For each element $1 \leq k \leq n$ in U, we add rules $\alpha_k :\Rightarrow p$ and $\beta_k :\Rightarrow \sim p$;
(ii) For each k in U, we add the superiority $\alpha_k \succ \beta_k$;
(iii) For each singleton element c_i of the collection C, with $c_i = \{m\}$, we add $\alpha_m \succ \beta_{m'}$ for every $m' \neq m$;
(iv) For each non-singleton element c_i of the collection C, we divide in a random way the rules (for instance by turning left to right on the elements of c_i) and build superiority pairs letting each element β_k, where $k \notin c_i$, be beaten by exactly one rule corresponding to elements in c_i.

Basically, we mapped each subset of the collection C onto a minimally sufficient set, and force the algorithm that would hypothetically solve the problem of computing a minimal contraction on the obtained defeasible theory to determine exactly one element of each subset of the collection that, as translated in the theory, makes unusable that specific minimally sufficient set, by executing the choice for each minimally sufficient set. In practice, the theory is such that $D \vdash +\partial p$ and we compute a minimal contraction to obtain theory D' such that $D' \vdash -\partial p$. Such a contraction set is a set of rules of the form $\alpha_k :\Rightarrow p$ that can be mapped back into a set of naturals of the form k: this is a solution for the instance of the Exact Hitting Problem.

The above defined translation is clearly polynomial, and therefore we have proved that the problem is NP-hard. □

For instance, consider the following situation:

- The universe is $U = \{1, 2, 3, 4, 5, 6, 7, 8\}$;
- The collection $C = \{\{1, 4\}, \{4, 5, 7\}, \{3, 5, 6\}, \{2, 7\}, \{8\}\}$.

It is easy to see that the set $\{1, 2, 5, 8\}$ is a hitting set for C. In fact, 1 hits $c_1 = \{1, 4\}$, while 2 hits $c_4 = \{2, 7\}$, 5 hits $c_2 = \{4, 5, 7\}$ and $c_3 = \{3, 5, 6\}$ and finally 8 hits c_5. Analogously, the set $\{4, 3, 2, 8\}$ is hitting, since 4 hits c_1 and c_2, 3 hits c_3, 2 hits c_4 and 8 hits c_5.

Step (i): We build the basic rules $\alpha_j :\Rightarrow p$ and the opposite $\beta_k :\Rightarrow \sim p$, for every j with $1 \leq j \leq 8$.

Step (ii): We construct the superiority pairs $\alpha_j \succ \beta_j$, e.g., $\alpha_1 \succ \beta_1$.

Step (iii): We construct the superiority pairs $\alpha_8 \succ \beta_j$, for any $1 \leq j \leq 7$.

Step (iv): Consider now the elements in each set in the collection C, and in turn, left to right, assign the elements to the superiority pairs. When starting with the set $c_1 = \{1, 4\}$, we divide the rules in c_1 and $R \setminus \{1, 4\}$. This means that $\alpha_1 \succ \beta_2$, $\alpha_4 \succ \beta_3$, $\alpha_1 \succ \beta_5$, $\alpha_4 \succ \beta_6$, $\alpha_1 \succ \beta_7$ and lastly $\alpha_4 \succ \beta_8$. The same procedure is applied to the set $\{4, 5, 7\}$ (e.g., $\alpha_4 \succ \beta_1$, $\alpha_5 \succ \beta_2$), and further on $\{3, 5, 6\}$, to finally $\{2, 7\}$.

This concludes the reduction. Now, let us go onto the execution of the minimal contraction algorithm. The rules that need to be removed to contract p are α_4, α_3, α_2 and α_8, which let us discard rules corresponding one at a time to the subsets $\{4, 5, 7\}, \{3, 5, 6\}, \{2, 7\}$. The above set, corresponding to the contraction, is the hitting set $\{4, 3, 2, 8\}$.

A further observation should be made at this point. Some equivalent formulations of the Exact Hitting Problem can be stated with the requirement that the number of elements in the solution is the smallest possible number. This is not the case for the above computed solution $\{4, 3, 2, 8\}$, or for the solution $\{1, 2, 5, 8\}$ mentioned before.

That is a minimal contraction too. It is easy to see that the two problems are also equivalent on Defeasible Logic. We can simply require the oracle to guess a minimal contraction. This is a polynomial task, for it simply consists in determining all the possible minimal contractions (again polynomial on non-deterministic machines) and choose the last one satisfying the requirement, after having computed each of the minimal ones. This can be obviously employed to solve the exact hitting set problem variant requiring minimum number of elements with the very same method. Analogous extensions can be obtained for revisions.

8 Related work

Mapping the literature on necessity and sufficiency is a titanic effort, which could appear also rather assumptive for a specific study in non-monotonic reasoning. We therefore more modestly look at possible proximal results, both in terms of temporal vicinity and in terms of resemblance

of methods as based on rules, with the specific purpose of providing readers with a landscape of research pathways, and clarify our standpoint and perspectives on this respect.

The starting point of revision methods for logical systems is obviously the study on revision operators' postulates by [1]. In their view, authors of this foundational work pose a general issue: how can we preserve some coherence of the axiomatic structure of a theory when modifying it at the same time? In order to answer this, they provide a list of properties that are known as the AGM postulates.

Other approaches to non-monotonicity that have been pursued in the community of formal logic and automated reasoning, which incorporate notions of sceptical inference are the prioritized Logic Programming, an ample field whose milestone has been posed by [52]. Similarly, aspects related to the development of hypotheses and verification of them are proposed in the studies on Answer Set Programming. In general, [7] dealt with this problem in a systematic way. More details on the aspects of the revision process by itself in this domain derive by the research by [46]

Recent advancements [4] on these aspects have also shown that it is possible to devise a general strategy of belief revision only when some truth preservation is obtained, which requires specific properties of the operators, not necessarily just compliance to AGM postulates. This study is in the path defined by Tennant in his research of founding the belief revision on a different base [56].

Specific investigation in the field is the one on conditional belief revision. This field can be considered initiated by [13], and further developed by [37]. The basic notion investigated in these studies is that, quoting directly 'the propositional paradigm of minimal change guiding the AGM-postulates of belief revision proved to be inadequate for preserving conditional beliefs under revision'. In other terms, the basic issue is to guarantee the preservation of some relevant properties of the revised system.

A wide attention has been posed on the theme of *minimal change*, starting with the investigation of [6] towards the new applications and further results obtained in the last two decades by [38] and [50]. The notion of minimal change as a reference point for this research has already been discussed in Section 1, in particular for what concerns [50].

Belief revision is more complicated when the logical framework we are looking at is not monotonic. Some investigations have been carried out on this aspect in Argumentation Theory [20, 36, 54]. More widely, some of the authors of this article have explored the topic [25] in the context of the revision of defeasible preferences.

The notion of preservation of coherence that we may consider foundational of revision in non-monotonic systems is presented by [58]. That concept of minimality recalls the notion of prime implicants as devised by [35], whom we mentioned in the Introduction section. To identify claims to be processed in a theory, Van Benthem discusses an approach that looks backward from a conclusion to draw which premises are needed in that specific proof. In other terms, the method looks at *ways* of proving a claim as starting point of a revision process. This is exactly what we need to do in non-monotonic systems. The coherence we need to preserve, when devising a revision, consists in amending the theory in the elements that bring to the conclusion and not amending it where not needed. We need to *isolate* the elements of the theory that are needed to provide the proof of a token to discharge it, and possibly invert the superiority relation when making a claim opposite to the starting one.

There are also other literature areas that need to be considered. In the case of minimization of theories, in terms of axioms. In particular, we have considered axiom set minimizations in rough and fuzzy systems, as investigated by [60] and further by [61]. In both these studies, minimization of axiom sets is complicated by the intrinsic irreducibility of fuzzy and rough sets to classical theories. A generalization of results on belief revision as applied to classic and non-classic systems has been the focus of the study by [16], which in turn is based on the results by [15].

The authors of [40] study methods to change defeasible logic programs (de.l.p.s) which are the knowledge bases used by the Defeasible Logic Programming (DeLP) interpreter. DeLP is an argumentation formalism that allows to reason over potentially inconsistent de.l.p.s. Argument Theory Change (ATC) [51]. Abstract arguments are rendered concrete by using the particular rule-based defeasible logic adopted by DeLP. ATC applies belief revision concepts to the field of abstract argumentation [19]. The main contribution provided by ATC is a revision operator at argument level that revises a theory by an argument seeking for its warrant. A warrant is identified as an argument's acceptance criterion corresponding to the adopted argumentative semantics. To such end, the theory—and thus the set of arguments obtained from it—is modified in order to guarantee success: the new argument should be accepted by the argumentation semantics. In the article, the authors defined prioritized argument revision operators *à la* ATC for de.l.p.s, in such a way that the newly inserted argument ends up undefeated after the revision, thus warranting its conclusion. In order to ensure this warrant, the de.l.p. has to be changed in concordance according to some minimal change principle. Diverse notions of minimality are given: (1) preserve the tree structure, (2) deactivate less defeasible rules as possible, (3) minimizing the number of incisions (arguments to be removed), and (4) profitability, which requires collateral incisions to affect selections in lines belonging to the attacking set (collateral means that the deactivation of some arguments may lead to the unwilling deactivation of others). Note that, warranting of the new argument is achieved by never adding but only removing elements. We believe that the closest notion to our investigation is less number of incisions as both the problems we studied, MCC and MRC, only removes rules from the input theory, and, with respect to our framework, we can consider an argument as the reasoning chain of rules leading to its last (root) conclusion. In the light of the results of NP-hardness we advanced and the motivations behind them, we do not think it is worth studying different minimality criteria than the one we considered; others could be (i) preserve the positive extension, (ii) preserve the negative extension, (iii) preserve both extensions, (iv) depending on (i)–(iii) the number of rules applicable/discarded in the modified theory with respect to the ones in the original theory. The motivation behind our choice is that the related decision problems would be, at least, as hard to solve, we suspect in a higher computational class.

As we also discussed in Section 1, we could consider as a general reference for non-monotonic revision problems the research by [57]. Inspired by their study, further researches have been carried out, in particular in the field of database revision by [32, 47] and for the definition of *general information space*, a concept that generalizes the idea of inconsistency in logical apparatuses by [33].

9 Conclusions and future investigations

In this paper, we have discussed the problem of identifying minimal contraction and revision of defeasible theories to drop results or to conclude for the opposite. We have shown that the two above mentioned problems are NP-hard. We have however shown that there is a heuristics that can be applied polynomially in a subset of cases, when *necessary rules* exist.

In fact, in the above mentioned methods we employed the notions of *necessity* and *sufficiency* of rules that is a novel concept for the specific context of Defeasible Logic, though similar concepts have been widely analysed with respect to classical logic.

We obtain applicable results on these concepts that are indeed used specifically in the above context of contraction and revision, but serve as a base for other possible uses as shown by some examples we introduced in the paper.

Further investigations shall focus on other possible ways of applying the powerful concepts of necessity and sufficiency to other aspects, in particular to the superiority relation, on how to make these notions interfere in the process of deriving metarules [21, 42], and how to develop more extended heuristics to improve computational effectiveness in these contraction and revision processes.

There is a specific interaction that is also worth investigating, among facts and rules. Facts are limiting strongly the concepts of necessity and sufficiency, and we are interested in understanding better this dependency, to state when, for instance, a rule depends on the facts to be sufficient (or necessary). This is certainly the case for what concerns the rules as related to the facts that activate or discharge a rule, but what about other facts? A method to devise solutions for these cases is worth investigating.

Finally, we shall investigate the problems related to the usage of rules whose head is actually another rule. In this case, the *dynamics* of rule change could be very different than it is in the basic case. Recently [44] investigated the notion of *deontic* meta-rules, which include defeasible meta-rules as well. This will be the starting point of the deepening mentioned above.

Acknowledgements

The authors Francesco Olivieri, Matteo Cristani, Guido Governatori and Luca Pasetto gratefully thank the Ministry of Justice of the Italian Republic for the financial support under grant UNI4JUSTICE PON (prot. n. m_dg_DGCPC.05/01/2022.0000016.ID).
The author Tewabe Chekole Workneh gratefully thanks the Ministry of Education of the Italian Republic for the financial support under PON Green AI Doctoral Program ‘Green AI as a tool for carbon footprint reduction and environmental sustainability’, with the collaboration of CSQA s.r.l.

References

[1] C. E. Alchourrón, P. Gärdenfors and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, **50**, 510–530, 1985.
[2] G. Antoniou. Defeasible logic with dynamic priorities. *International Journal of Intelligent Systems*, **19**, 463–472, 2004.
[3] G. Antoniou, D. Billington, G. Governatori and M. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, **2**, 255–287, 2001.
[4] A. Baltag, N. Gierasimczuk and S. Smets. Truth-tracking by belief revision. *Studia Logica*, **107**, 917–947, 2019.
[5] D. Billington, G. Antoniou, G. Governatori and M. Maher. Revising nonmonotonic theories: the case of defeasible logic. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **1701**, 101–112, 1999.
[6] C. Boutilier. Iterated revision and minimal change of conditional beliefs. *Journal of Philosophical Logic*, **25**, 263–305, 1996.
[7] G. Brewka, T. Eiter and M. Truszczyński. Answer set programming at a glance. *Communications of the ACM*, **54**, 92–103, 2011.
[8] G. Brewka, M. Thimm and M. Ulbricht. Strong inconsistency. *Artificial Intelligence*, **267**, 78–117, 2019.
[9] M. Cristani, G. Governatori, F. Olivieri, L. Pasetto, F. Tubini, C. Veronese, A. Villa and E. Zorzi. Houdini (unchained): an effective reasoner for defeasible logic. In *CEUR Workshop Proceedings*, vol. 3354, 2022.

[10] M. Cristani, F. Olivieri and A. Rotolo. Changes to temporary norms. In *ICAIL*, pp. 39–48. ACM, 2017.
[11] M. Cristani, F. Olivieri and C. Tomazzoli. Automatic synthesis of best practices for energy consumptions. In *IMIS*, pp. 154–161. IEEE Computer Society, 2016.
[12] M. Cristani, C. Tomazzoli, E. Karafili and F. Olivieri. Defeasible reasoning about electric consumptions. In *30th IEEE International Conference on Advanced Information Networking and Applications*, pp. 885–892. IEEE Computer Society, 2016.
[13] A. Darwiche and J. Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, **89**, 1–29, 1997.
[14] J. Delgrande, P. Peppas and S. Woltran. Agm-style belief revision of logic programs under answer set semantics. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8148 LNAI*, pp. 264–276, 2013.
[15] J. P. Delgrande and P. Peppas. Belief revision in horn theories. *Artificial Intelligence*, **218**, 1–22, 2015.
[16] J. P. Delgrande, P. Peppas and S. Woltran. General belief revision. *Journal of the ACM*, **65**, 1–34, 2018.
[17] M. Diller, A. Haret, T. Linsbichler, S. Rümmele and S. Woltran. An extension-based approach to belief revision in abstract argumentation. *International Journal of Approximate Reasoning*, **93**, 395–423, 2018.
[18] L. Drori and D. Peleg. Faster exact solutions for some np-hard problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **1643**, 450–461, 1999.
[19] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, **77**, 321–357, 1995.
[20] M. A. Falappa, A. J. García, G. Kern-Isberner and G. R. Simari. Stratified belief bases revision with argumentative inference. *Journal of Philosophical Logic*, **42**, 161–193, 2013.
[21] G. Governatori, F. Olivieri, A. Rotolo, A. Sattar and M. Cristani. Computing private international law. *Frontiers in Artificial Intelligence and Applications*, **346**, 181–190, 2021.
[22] G. Governatori, F. Olivieri, S. Scannapieco and M. Cristani. Superiority based revision of defeasible theories. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6403 LNCS*, pp. 104–118, 2010.
[23] G. Governatori and A. Rotolo. Changing legal systems: legal abrogations and annulments in defeasible logic. *Logic Journal of the IGPL*, **18**, 157–194, 2009.
[24] G. Governatori, M. J. Maher, D. Billington and G. Antoniou. Argumentation semantics for defeasible logics. *Journal of Logic and Computation*, **14**, 675–702, 2004.
[25] G. Governatori, F. Olivieri, M. Cristani and S. Scannapieco. Revision of defeasible preferences. *International Journal of Approximate Reasoning*, **104**, 205–230, 2019.
[26] G. Governatori, F. Olivieri, A. Rotolo and S. Scannapieco. Computing strong and weak permissions in defeasible logic. *Journal of Philosophical Logic*, **42**, 799–829, 2013.
[27] G. Governatori, F. Olivieri, S. Scannapieco, A. Rotolo and M. Cristani. Strategic argumentation is np-complete. In *21st European Conference on Artificail Intelligence*, pp. 399–404. IOS Press, 2014.
[28] G. Governatori, F. Olivieri, S. Scannapieco, A. Rotolo and M. Cristani. Strategic argumentation is np-complete. In *21st European Conference on Artificial Intelligence*. Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 399–404. IOS Press, 2014.
[29] G. Governatori, F. Olivieri, S. Scannapieco, A. Rotolo and M. Cristani. The rationale

behind the concept of goal. *Theory and Practice of Logic Programming*, **16**, 296–324, 2016.
[30] G. Governatori, V. Padmanabhan, A. Rotolo and A. Sattar. A defeasible logic for modelling policy-based intentions and motivational attitudes. *Logic Journal of the IGPL*, **17**, 227–265, 2009.
[31] G. Governatori, A. Rotolo, F. Olivieri and S. Scannapieco. Legal contractions: a logical analysis. In *International Conference on Artificial Intelligence and Law*, pp. 63–72. ACM, 2013.
[32] J. Grant. Postulate satisfaction for inconsistency measures in monotonic logics and databases. *Journal of Applied Non-Classical Logics*, **33**, 537–560, 2023.
[33] J. Grant and F. Parisi. General information spaces: measuring inconsistency, rationality postulates, and complexity. *Annals of Mathematics and Artificial Intelligence*, **90**, 235–269, 2022.
[34] P. Gärdenfors. An epistemic approach to conditionals. *American Philosophical Quarterly*, **18**, 203–211, 1981.
[35] P. L. Hammer, E. L. Johnson and U. N. Peled. Facet of regular 0-1 polytopes. *Mathematical Programming*, **8**, 179–206, 1975.
[36] J. Heyninck, G. Kern-Isberner, T. Rienstra, K. Skiba and M. Thimm. Revision, defeasible conditionals and non-monotonic inference for abstract dialectical frameworks. *Artificial Intelligence*, **317**, 103876, 2023.
[37] G. Kern-Isberner. A thorough axiomatization of a principle of conditional preservation in belief revision. *Annals of Mathematics and Artificial Intelligence*, **40**, 127–164, 2004.
[38] S. Konieczny, M. M. Grespan and R. P. Pérez. Taxonomy of improvement operators and the problem of minimal change. *Proceedings of the International Conference on Knowledge Representation and Reasoning*, 161–170, 2010.
[39] I. Levi. Subjunctives, dispositions and chances. *Synthese*, **34**, 423–455, 1977.
[40] M. O. Moguillansky, N. D. Rotstein, M. A. Falappa, A. J. García and G. R. Simari. Dynamics of knowledge in DeLP through argument theory change. *Theory and Practice of Logic Programming*, **13**, 893–957, 2013.
[41] D. Nute. Defeasible logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 3. Oxford University Press, 1987.
[42] F. Olivieri, G. Governatori, M. Cristani and A. Sattar. Computing defeasible meta-logic. *Lecture Notes in Computer Science*, **12678**, 2021.
[43] F. Olivieri, M. Cristani and G. Governatori. Compliant business processes with exclusive choices from agent specification. In *PRIMA*. LNCS, vol. 9387, pp. 603–612. Springer, 2015.
[44] F. Olivieri, G. Governatori, M. Cristani, A. Rotolo and A. Sattar. Deontic meta-rules. *Journal of Logic and Computation*, **34**, 261–314, 2024.
[45] F. Olivieri, G. Governatori, M. Cristani, N. van Beest and S. C. Tosatto. Resource-driven substructural defeasible logic. In *PRIMA*. LNCS, vol. 11224, pp. 594–602. Springer, 2018.
[46] M. Osorio and V. Cuevas. Updates in answer set programming: an approach based on basic structural properties. *Theory and Practice of Logic Programming*, **7**, 451–479, 2007.
[47] F. Parisi and J. Grant. On measuring inconsistency in definite and indefinite databases with denial constraints. *Artificial Intelligence*, **318**, 103884, 2023.
[48] D. L. Poole. On the comparison of theories: preferring the most specific explanation. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)*, 1985.

[49] J. S. Ribeiro. Kernel contraction and the order of relevance. In *19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022*, pp. 299–308, 2022.
[50] M. M. Ribeiro, R. Wassermann, G. Flouris and G. Antoniou. Minimal change: relevance and recovery revisited. *Artificial Intelligence*, **201**, 59–80, 2013.
[51] N. D. Rotstein, M. O. Moguillansky, M. A. Falappa, A. J. García and G. R. Simari. Argument theory change: revision upon warrant. In *Computational Models of Argument*. Frontiers in Artificial Intelligence and Applications, vol. 172, pp. 336–347. IOS Press, 2008.
[52] C. Sakama and K. Inoue. Prioritized logic programming and its application to commonsense reasoning. *Artificial Intelligence*, **123**, 185–222, 2000.
[53] G. R. Simari and R. P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, **53**, 125–157, 1992.
[54] M. Snaith and C. Reed. Argument revision. *Journal of Logic and Computation*, **27**, 2089–2134, 2017.
[55] F. Stolzenburg, A. J. Garća, C. I. Chesñevar and G. R. Simari. Computing generalized specificity. *Journal of Applied Non-Classical Logics*, **13**, 87–113, 2003.
[56] N. Tennant. New foundations for a relational theory of theory-revision. *Journal of Philosophical Logic*, **35**, 489–528, 2006.
[57] M. Ulbricht, M. Thimm and G. Brewka. Handling and measuring inconsistency in non-monotonic logics. *Artificial Intelligence*, **286**, 103344, 2020.
[58] J. Van Benthem. Minimal predicates, fixed-points, and definability. *Journal of Symbolic Logic*, **70**, 696–712, 2005.
[59] C.-P. Wirth and F. Stolzenburg. A series of revisions of david poole's specificity. *Annals of Mathematics and Artificial Intelligence*, **78**, 205–258, 2016.
[60] X.-P. Yang and T.-J. Li. The minimization of axiom sets characterizing generalized approximation operators. *Information Sciences*, **176**, 887–899, 2006.
[61] Y.-L. Zhang and M.-K. Luo. On minimization of axiom sets characterizing covering-based approximation operators. *Information Sciences*, **181**, 3032–3042, 2011.

Received 19 March 2024