Künstl Intell (2017) 31:15–30
DOI 10.1007/s13218-016-0466-4

TECHNICAL CONTRIBUTION

Polynomial Algorithms for Computing a Single Preferred Assertional-Based Repair

Abdelmoutia Telli[1,3] · Salem Benferhat[1] · Mustapha Bourahla[4] · Zied Bouraoui[2] · Karim Tabia[1]

Received: 31 January 2016 / Accepted: 18 November 2016 / Published online: 23 January 2017

Abstract This paper investigates different approaches for handling inconsistent *DL-Lite* knowledge bases in the case where the assertional base is prioritized and inconsistent with the terminological base. The inconsistency problem often happens when the assertions are provided by multiple conflicting sources having different reliability levels. We propose different inference strategies based on the selection of one consistent assertional base, called a preferred repair. For each strategy, a polynomial algorithm for computing the associated single preferred repair is proposed. Selecting a unique repair is important since it allows an efficient handling of queries. We provide experimental studies showing (from a computational point of view) the benefits of selecting one repair when reasoning under inconsistency in lightweight knowledge bases.

✉ Salem Benferhat
benferhat@cril.univ-artois.fr

Abdelmoutia Telli
telli@cril.univ-artois.fr

Mustapha Bourahla
mbourahla@hotmail.com

Zied Bouraoui
BouraouiZ@cardiff.ac.uk

Karim Tabia
tabia@cril.univ-artois.fr

1 Artois University-Nord de France, CRIL CNRS UMR 8188, Artois, 62307 Lens, France

2 Cardiff University, Cardiff, UK

3 Mohamed Khider University of Biskra, BP 145 RP, 07000 Biskra, Algeria

4 University of M'sila, BP 166 RP, 28000 M'sila, Algeria

Keywords Inconsistency · Prioritized knowledge bases · Lightweight ontologies

1 Introduction

Description logics (DLs) are formal frameworks for representing and reasoning with ontologies. A DL knowledge base is built upon two distinct components: a terminological base (called *TBox*) representing generic knowledge, and an assertional base (called *ABox*) containing facts or assertions.

Recently, a particular interest was given to ontology based data access (OBDA), in which the ontological view (i.e. the *TBox*) is used to offer a better exploitation of assertions (i.e. the ABox) when querying them (e.g. [26, 33]). A crucially important problem that arises in OBDA is how to manage conflicting information. In such a setting, an ontology is usually verified and validated while the assertions can be provided in large quantities by various and unreliable sources that may be inconsistent with respect to the ontology. Moreover, it is often too expensive to manually check and validate all the assertions. This is why it is very important in OBDA to reason in the presence of inconsistency. Many works (e.g. [13, 15, 25, 27, 28]), basically inspired by the ones in the database area (e.g. [1, 12, 19]) and propositional logic approaches (e.g. [7, 8, 31]), deal with inconsistency in DLs by proposing several inconsistency-tolerant inferences, called *semantics*. These semantics are based on the notion of a maximally assertional (or ABox) repair which is closely related to the notion of a database repair [24] or a maximally consistent **subset** used in the propositional logic setting (e.g. [16, 32]). An ABox repair is simply an assertional **subset** which is consistent with an ontology.

In many applications, assertions are often provided by several and potentially conflicting sources having different reliability levels. Moreover, a given source may provide different sets of uncertain assertions with different confidence levels. Gathering such sets of assertions gives a prioritized or a stratified assertional base (i.e. ABox). The role of priorities in handling inconsistency is very important and it is largely studied in the literature within the propositional logic setting (e.g. [3, 10, 11]). Several works have also studied the notion of priority when querying inconsistent databases (e.g. [30, 35]) or DLs knowledge bases (e.g. [6, 14, 21]).

The context of this paper is the one of handling inconsistency in lightweight ontologies when the ABox is prioritized. We use *DL-Lite* [2], an important tractable fragment of DLs, as an example of lightweight ontologies which is well-suited for OBDA [25].

In the presence of conflicting information, there is always a compromise that one needs to reach between the expressiveness and computational issues. Having multiple repairs often allows to derive more conclusions than if only one repair is used. However, query answering from multiple repairs is generally more expensive than query answering from a single repair. In fact, reasoning from a single repair can be viewed as an approximation of reasoning from multiple repairs.

The main contribution of this paper is to investigate polynomial algorithms for selecting a unique preferred repair. Selecting only one preferred repair is important since, once computed, it allows an efficient query answering. It is important to note that some inference relations are specific to *DL-Lite* even if they are inspired by other formalisms such as the propositional logic setting. The polynomial algorithms proposed in this paper implement and evaluate five strategies that we have recently proposed in a conference paper [6].

The two first strategies for selecting one preferred repair are an adaptation of the well-known possibilistic inference [20] and linear-based inference mainly defined in prioritized propositional knowledge bases (e.g. [31]). The three other strategies are based on the use of the so-called non-defeated assertional-based repair and its variants obtained by adding either deductive closure or consistency criteria. Interestingly enough, many of these strategies are suitable for the *DL-Lite* setting in the sense that they allow efficient handling of inconsistencies by producing a single preferred repair. Our experimental results show the benefits of adding priorities when reasoning under inconsistency in *DL-Lite*. This journal paper is an extended version of a part of the conference paper [6].

The rest of this paper is organized as follows: Sect. 2.1 provides the needed background on *DL-Lite*. Section 3 presents some elementary concepts on inconsistency handling such as the concepts of conflicts, repairs and free assertions. Section 4 introduces the two first ways to compute one preferred repair based on possibilistic and linear-based strategies. Section 5 presents the so-called non-defeated repair. Section 6 introduces the notion of a prioritized deductive closure. Sections 7 and 8 show two variants of non-defeated repair based on the notion of consistency and prioritized closures. Section 9 provides our experimental studies and Sect. 10 concludes the paper.

2 *DL-Lite* and Prioritized Assertional Base

2.1 *DL-Lite*: A Brief Refresher

This section briefly recalls *DL-Lite* logics. For the sake of simplicity, we only consider $DL\text{-}Lite_R$ language [17] and we will simply use *DL-Lite* instead of $DL\text{-}Lite_R$. Note that the results of this paper can be extended in a straightforward way to any tractable *DL-Lite* as far as computing ABox conflicts is done in polynomial time. This is true for $DL\text{-}Lite_{core}$ (a particular case of $DL\text{-}Lite_R$) and $DL\text{-}Lite_F$. The *DL-Lite* language is defined as follows:

$$R \longrightarrow P \mid P^- \qquad E \longrightarrow R \mid \neg R$$
$$B \longrightarrow A \mid \exists R \qquad C \longrightarrow B \mid \neg B$$

where A is an atomic concept, P is an atomic role and P^- is the inverse of P. B (resp. C) is called basic (resp. complex) concept and role R (resp. E) is called basic (resp. complex) role. A knowledge base (KB) is a couple $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. A TBox includes a finite set of inclusion axioms on concepts and on roles respectively of the form: $B \sqsubseteq C$ and $R \sqsubseteq E$. The *ABox* contains a finite set of atomic concepts and role assertions respectively of the form $A(a)$ and $P(a, b)$ where a and b are two individuals.

The semantics of a *DL-Lite* knowledge base is given in terms of interpretations. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $.^{\mathcal{I}}$ that maps each individual a to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each A to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role P to $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Furthermore, the interpretation function $.^{I}$ is extended in a straightforward way for concepts and roles as follows:

$$A^I \subseteq \Delta^I$$
$$P^I \subseteq \Delta^I \times \Delta^I$$
$$(P^-)^I = \{(y, x) \in \Delta^I \times \Delta^I | (x, y) \in P^I\}$$
$$(\exists R)^I = \{x \in \Delta^I | \exists y \in \Delta^I \textit{ such that } (x, y) \in R^I\}$$
$$(\neg B)^I = \Delta^I \backslash B^I$$
$$(\neg R)^I = \Delta^I \times \Delta^I \backslash R^I$$

An interpretation I is said to be a model of a concept (resp. role) inclusion axiom, denoted by $I \vDash B \sqsubseteq C$ (resp.

$I \models R \sqsubseteq E$), if and only if $B^I \subseteq C^I$ (resp. $R^I \subseteq E^I$). Similarly, we say that an interpretation I is a model of a membership assertion $A(a)$ (resp. $P(a, b)$), denoted by $I \models A(a)$ (resp. $I \models P(a,b)$), if and only if $a^I \in A^I$ (resp. $(a^I, b^I) \in P^I$). A knowledge base $\mathcal{K}$ is called consistent if it admits at least one model, otherwise $\mathcal{K}$ is said to be inconsistent. A TBox $\mathcal{T}$ is said to be incoherent if there exists at least a concept C such that for each interpretation $\mathcal{I}$ which is a model of $\mathcal{T}$, we have $C^{\mathcal{I}} = \emptyset$.

2.2 Prioritized Assertional Bases

A prioritized assertional base (or a prioritized ABox), simply denoted by $\mathcal{A} = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$, is a tuple of sets of assertions. The sets $\mathcal{S}_i$ are called layers or strata. Each layer $\mathcal{S}_i$ contains the set of assertions having the same level of priority i and they are considered as more reliable than the ones present in a layer $\mathcal{S}_j$ when $j > i$. Hence, $\mathcal{S}_1$ contains the most important assertions while $\mathcal{S}_n$ contains the least important assertions.

Throughout this paper and when there is no ambiguity we simply use 'prioritized *DL-Lite* KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$' to refer to a *DL-Lite* KB with a prioritized ABox of the form $\mathcal{A} = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$.

Example 1 Let Student, Researcher and Teacher be three concepts that intuitively contain the set of students, the set of researchers and the set of teachers respectively. Let TeachesTo and HasSupervisor be two roles which intuitively give the list of students for a given teacher and the list of supervisors of a given student. Assume that we have the following ontology:

- Teachers are not students.
- Teachers have to give a course to at least one student.
- Individuals who receive courses from a teacher are students.
- Each student has at least a supervisor.
- Supervisors are teachers.
- Researchers are teachers.

This ontology is encoded by the following TBox $\mathcal{T}$:

$$\boxed{\begin{array}{l} Teacher \sqsubseteq \neg Student \\ Teacher \sqsubseteq \exists TeachesTo \\ \exists TeachesTo^- \sqsubseteq Student \\ Student \sqsubseteq \exists HasSupervisor \\ \exists HasSupervisor^- \sqsubseteq Teacher \\ Researcher \sqsubseteq Teacher \end{array}}$$

In addition to this TBox, assume that we have six individuals: Bill, John, Mary, Joe, Bob and Anne. We assume that there is some uncertainty regarding the status and roles of these individuals. The available factual information is encoded by the following ABox, which is assumed to be provided by five distinct sources $\mathcal{A} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5)$ such that:

$$\boxed{\begin{array}{l} \mathcal{S}_1 = \{Student(Bill), Teacher(John)\}, \\ \mathcal{S}_2 = \{Teacher(Mary), Student(Bob)\}, \\ \mathcal{S}_3 = \{HasSupervisor(Bob, Bill), Researcher(Joe)\}, \\ \mathcal{S}_4 = \{Student(Anne)\}, and \\ \mathcal{S}_5 = \{Teacher(Bill)\}. \end{array}}$$

In this example, $\mathcal{S}_1$ contains the most reliable assertions while $\mathcal{S}_5$ contains the least reliable ones. ■

In Example 1, it is easy to check that the KB is inconsistent. For instance, in the ABox we have *Student*(*Bill*) and *HasSupervisor*(*Bob*, *Bill*) as assertions. From the axiom $\exists HasSupervisor^- \sqsubseteq Teacher$ one can conclude the following fact: *Teacher*(*Bill*). This contradicts the negative axiom $Teacher \sqsubseteq \neg Student$. If a knowledge base is inconsistent then query answering is trivialised since any thing can be inferred from it. As an alternative, inconsistency-tolerant approaches try to select consistent subsets (called repairs) of the inconsistent knowledge base to meaningfully answer the queries.

This paper proposes different approaches to deal with inconsistent *DL-Lite* KB. The input of these approaches is a prioritized *DL-Lite* knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with $\mathcal{A} = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$. The output of our approaches is a standard *DL-Lite* knowledge base $\mathcal{K}' = \langle \mathcal{T}, \mathcal{R} \rangle$, where $\mathcal{R}$ is not prioritized (namely, just a set of assertions). $\mathcal{K}$ and $\mathcal{K}'$ have the same terminological base. $\mathcal{R}$ will be called a preferred repair. Then a query q is said to follow from $\mathcal{K}$ if it can be derived, using the standard *DL-Lite* inference, from $\mathcal{K}'$. Before presenting our approaches, the following section briefly recalls the main important concepts for handling inconsistency.

3 Inconsistency-Tolerant Reasoning for Prioritized *DL-Lite* Assertional Bases

3.1 The Concept of Repairs

Within the OBDA setting, we assume that $\mathcal{T}$ is coherent and hence its elements are not questionable in the presence of conflicts. Coping with inconsistency can be done by first computing the set of consistent subsets of assertions (not necessarily maximal), called repairs, then using them to perform inference (i.e. query answering). More formally, a repair is defined as follows:

Definition 1 Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized *DL-Lite* KB with $\mathcal{A} = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$.

A **subset** $\mathcal{R} \subseteq (\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n)$. is said to be a repair if $\langle \mathcal{T}, \mathcal{R} \rangle$ is consistent. And $\mathcal{R}$ is said to be a maximally inclusion-based repair of $\mathcal{K}$, denoted by *MAR*, if $\langle \mathcal{T}, \mathcal{R} \rangle$ is consistent and $\forall \mathcal{R}' \subseteq (\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n) : \mathcal{R} \subsetneq \mathcal{R}', \langle \mathcal{T}, \mathcal{R}' \rangle$ is inconsistent.

In the rest of this paper, we will use the term 'flat' to express the fact that there is no priority between different assertions of an ABox. According to the definition of *MAR*, adding any assertion f from $(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n) \setminus \mathcal{R}$ to $\mathcal{R}$ entails the inconsistency of $\langle \mathcal{T}, \mathcal{R} \cup \{f\} \rangle$. Moreover, the maximality in *MAR* is used in the sense of set inclusion. We denote by *MAR*($\mathcal{A}$) the set of *MAR* of $\mathcal{A}$ with respect to $\mathcal{T}$. The definition of *MAR* coincides with the definition of *ABox* repair proposed in [24].

Using the notion of repair, coping with inconsistency in flat *DL-Lite* knowledge bases can be done by applying standard query answering either using the whole set of repairs (universal entailment or AR-entailment [24]) or only using one repair.

Example 2 (Example 1 continued). Assume that the ABox given in Example 1 is flat. To restore consistency, one can compute two maximal assertional-based repairs:

$$\boxed{\begin{array}{l} \mathcal{R}_1 = \{\ Student(Bill),\ Teacher(John),\ Teacher(Mary), \\ \quad Student(Anne), Student(Bob), Researcher(Joe)\} \\ \\ \mathcal{R}_2 = \{Teacher(Bill), Teacher(John), Teacher(Mary), \\ \quad Student(Bob), Student(Anne), Researcher(Joe), \\ \quad HasSupervisor(Bob, Bill)\}. \end{array}}$$

Indeed, either we only ignore the assertion *Student*(*Bill*), then the remaining assertions $\mathcal{R}_2 = \mathcal{A} \setminus \{Student(Bill)\}$ is consistent with $\mathcal{T}$. Or we keep the assertion *Student*(*Bill*) and in this case we need to remove *HasSupervisor*(*Bob*, *Bill*) and *Teacher*(*Bill*) to restore the consistency of $\langle \mathcal{T}, \mathcal{A} \rangle$. Then we get $\mathcal{R}_1$. ■

3.2 Free Assertions and Conflict Sets

We now introduce the notion of a conflict. It is a minimal **subset** C of assertions of $\mathcal{A}$ such that $\mathcal{K} = \langle \mathcal{T}, C \rangle$ is inconsistent.

Definition 2 Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite* KB. A **subset** $C \subseteq \mathcal{A}$ is said to be an assertional conflict of $\mathcal{K}$ iff $\langle \mathcal{T}, C \rangle$ is inconsistent and $\forall f \in C, \langle \mathcal{T}, C \setminus \{f\} \rangle$ is consistent.

From Definition 2, removing any fact f from C restores the consistency of $\langle \mathcal{T}, C \rangle$. In *DL-Lite*, when the TBox is coherent, a conflict involves exactly two assertions [18]. We denote by $\mathcal{C}(\mathcal{A})$ the set of conflicts in $\mathcal{A}$.

Example 3 (Example 1 continued). The set of conflicts is:

$$\boxed{\begin{array}{l} \mathcal{C}(\mathcal{A}) = \{\{\ Student(Bill), Teacher(Bill)\ \}, \\ \quad \{\ HasSupervisor(Bob, Bill), Student(Bill)\ \}\} \end{array}}$$

■

In the rest of this paper, we will use the term '*Conf*' to express the computational complexity of computing the set of conflicts $\mathcal{C}(\mathcal{A})$ of a standard *DL-Lite* KB. Note that checking whether $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent or not, comes down to check whether $\mathcal{C}(\mathcal{A})$ is empty or not. Hence, in the following *Conf* will also refer to the computational complexity of consistency checking of a standard *DL-Lite* Knowledge base.

A nice feature of *DL-Lite* is that computing the set of conflicts is done in polynomial time [5].

We now introduce the notion of non-conflicting or free elements.

Definition 3 Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be *DL-Lite* KB. An assertion $f \in \mathcal{A}$ is said to be *free* if and only if $\forall c \in \mathcal{C}(\mathcal{A}) : f \notin c$.

Intuitively, *free* assertions correspond to assertions that are not involved in any conflict. We denote by *free*($\mathcal{A}$) the set of *free* assertions in $\mathcal{A}$. The notion of *free* elements is originally proposed in [9] in a propositional logic setting. Within a *DL-Lite* setting, *free*($\mathcal{A}$) is computed in polynomial time thanks to the fact that computing conflicts is done in polynomial time.

Example 4 (Example 1 continued). The set of free elements for the Abox of Example 1 is:

$$\boxed{\begin{array}{l} free(\mathcal{A}) = \{Teacher(John), Teacher(Mary), \\ \quad Student(Anne),\ Researcher(Joe), \\ \quad Student(Bob)\}. \end{array}}$$

■

The folowing Lemma 1 rephrases the set of *free* assertions using the set of maximally inclusion-based repairs:

Lemma 1 *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite KB. Then:*

$$free(\mathcal{A}) = \bigcap_{X \in MAR(\mathcal{A})} X.$$

For flat *DL-Lite* knowledge bases, the *free*-entailment (entailment based on free assertions) is equivalent to the *IAR*-entailment proposed in [24]. In fact, in the context of propositional logic, the concept of a maximal consistent **subset** has been introduced before the concept of repairs [32]. Besides, the concept of *free* entailment has been introduced in [9]. In the rest of this paper, we will only use the notation *free*($\mathcal{A}$) to denote the set of assertions that are not **responsible for** conflicts in $\langle \mathcal{T}, \mathcal{A} \rangle$.

The following function computes the set of *free* elements in a set of assertions X (given a set of conflicts $\mathscr{C}$). This function will be used by algorithms developed in this paper.

Algorithm 1 Set of free elements

```
function FREE(X, 𝒞)
Input: X : A set of assertion
       𝒞: List of conflicts
Output: List of free elements in X
return (X\{f : f ∈ X, ∃g ∈ X such that (f, g) ∈ 𝒞})
```

The next sections of the paper describe the main strategies for computing repairs that are suitable for the *DL-Lite* setting when the assertional base is prioritized.

4 Possibilistic and Linear-Based Repairs

This section presents two approaches for dealing with inconsistent *DL-Lite* KB that have been originally proposed in a weighted propositional logic setting. These two approaches need a slight adaptation to be used within *DL-Lite* setting.

4.1 Possibilistic-Based Repair

Possibility theory [22] and possibilistic logic [20] are natural and intuitive frameworks for representing uncertain, incomplete, qualitative and prioritized information. One of the interesting aspects of possibilistic logic is its ability of reasoning with partially inconsistent knowledge [23]. As shown in [4], the entailment in possibilistic *DL-Lite*, an adaptation of *DL-Lite* entailment within a possibility theory setting, is based on the selection of one consistent, but not necessarily maximal, **subset** of $\mathcal{K}$. This **subset** is induced by a level of priority called here the consistency rank. The following gives the definition of consistency rank for prioritized *DL-Lite* assertional bases.

Definition 4 Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized *DL-Lite* KB. The consistency rank of $\mathcal{K}$, denoted by:

Consrank($\mathcal{K}$), is defined as follows:

$$Consrank(\mathcal{K}) = \begin{cases} 0 & \text{if } \langle \mathcal{T}, \mathcal{S}_1 \rangle \text{ is inconsistent .} \\ max\{i : \langle \mathcal{T}, (\mathcal{S}_1, \ldots, \mathcal{S}_i) \rangle \text{ is consistent } \} & \text{Otherwise.} \end{cases}$$

The notion of consistency rank given in Definition 4 is related to the notion of inconsistency degree used in possibilistic logic (where degrees are encoded using the unit interval [0, 1] instead of a stratification using positive integers). The **subset** $\pi(\mathcal{A})$ is made of the assertions having priority levels that are less or equal to *Consrank*($\mathcal{K}$). If $\mathcal{K}$ is consistent then we simply let $\boldsymbol{\pi(\mathcal{A}) = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n}$. Algorithm 1 implements Definition 4 and returns the possibilistic-based repair. It is the counterpart of the algorithm proposed in [20] (resp. [34]) in the propositional logic (resp. description logic) setting.

Algorithm 2 Possibilistic Based Repair

```
Input: 𝒦 = ⟨𝒯, 𝒜⟩ where 𝒜 = (S_1, ··· , S_n)
Output: A flat assertional base π(𝒜)
if 𝒦 = ⟨𝒯, 𝒜⟩ is consistent then
    return π(𝒜) = S_1 ∪ ··· ∪ S_n
else
    if 𝒦 = ⟨𝒯, S_1⟩ is inconsistent then
        return π(𝒜) = ∅
    else
        ℓ ⟵ 1
        u ⟵ n
        while (ℓ < u ) do
            α ⟵ ⌊(ℓ+u)/2⌋
            if ⟨T, S_1 ∪ ··· ∪ 𝒮_α⟩ is consistent then
                ℓ ⟵ α + 1
            else
                u ⟵ α
    return π(𝒜) = S_1 ∪ ··· ∪ 𝒮_{α−1}
```

As in the propositional logic setting, Algorithm 2 needs $log_2(n)$ consistency tests, where n is the number of different strata in $\mathcal{A}$. Note that computing repairs in a standard propositional logic setting is a hard task while it is polynomial in the *DL-Lite* setting [4]. Hence, Algorithm 2 returns the possibilistic-based repair in polynomial time.

Example 5 (Example 1 continued). According to Algorithm 2, we have:

$$\boxed{\begin{aligned} \pi(\mathcal{A}) =& \mathcal{S}_1 \cup \mathcal{S}_2 \\ =& \{Student(Bill), Teacher(John), \\ & Teacher(Mary), Student(Bob)\}. \end{aligned}}$$

Indeed, one can check that $\langle \mathcal{T}, \mathcal{S}_1 \cup \mathcal{S}_2 \rangle$ is consistent, while $\langle \mathcal{T}, \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \rangle$ is inconsistent. ■

free entailment and possibilistic-based repair can be viewed as a safe way to deal with inconsistency. The term *safe* is used by opposition to the term *risky* or *adventurous* with respect to the derived conclusions. Similarly, possibilistic conclusions are also considered safe since the possibilistic-based repair algorithm stops at the first layer where the inconsistency is introduced. Hence, only assertions having a rank less or equal than the one of inconsistency rank are taken into account for deriving conclusions. However, assertions having priority levels strictly greater than the consistency rank are simply inhibited [7] even if they are not involved in any conflict. To overcome such a limitation and provide more productive or larger repairs, a linear-based approach can be used.

4.2 Linear-Based Repair

One way to recover the inhibited assertions by the possibilistic-based repair is to use the linear-based repair from $\mathcal{A}$. Linear entailment has also been used in a propositional logic setting in [31] and has been applied for the DL setting (e.g. [34]).

Definition 5 Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized *DL-Lite* KB. The linear-based repair of $\mathcal{A}$, denoted by:

$\ell(\mathcal{A}) = \mathcal{S}'_1 \cup \cdots \cup \mathcal{S}'_n$, is defined as follows:

(i) If $i = 1$:

$$\mathcal{S}'_1 = \begin{cases} \mathcal{S}_1 & \text{if } \langle \mathcal{T}, \mathcal{S}_1 \rangle \text{ is consistent} \\ \emptyset & \text{Otherwise} \end{cases}$$

(ii) For $i = 2, \ldots, n$

$$\mathcal{S}'_i = \begin{cases} \mathcal{S}_i & \text{if } \langle \mathcal{T}, \mathcal{S}'_1 \cup \cdots \cup \mathcal{S}'_{i-1} \cup \mathcal{S}_i \rangle \text{ is consistent .} \\ \emptyset & \text{Otherwise.} \end{cases}$$

Clearly, $\ell(\mathcal{A})$ is obtained by discarding a layer $\mathcal{S}_i$ when its facts conflict with the ones involved in the previous layers. Algorithm 3 implements Definition 5. The **subset** $\ell(\mathcal{A})$ is unique and it is consistent with $\mathcal{T}$. The time complexity of computing $\ell(\mathcal{A})$ is in P. Indeed, from Algorithm 3, the computational complexity of computing $\ell(\mathcal{A})$ needs n times the computional complexity of checking the consistency (namely *Conf*) of a standard *DL-Lite*.

Algorithm 3 Linear-Based Repair

```
Input: K = ⟨T, A⟩ where A = (S_1, ..., S_n)
Output: A flat assertional base ℓ(A)
  ℓ(A) ← ∅
  for i = 1 to n do
    if ⟨T, ℓ(A) ∪ S_i⟩ is consistent then
      ℓ(A) ← ℓ(A) ∪ S_i
  return ℓ(A)
```

Example 6 (Example 1 continued). According to Definition 5, we have:

$$\boxed{\begin{array}{l} \mathcal{S}'_1 = \{Student(Bill), Teacher(John)\}, \\ \mathcal{S}'_2 = \{Teacher(Mary), Student(Bob)\}, \\ \mathcal{S}'_3 = \emptyset \\ \mathcal{S}'_4 = \{Student(Anne)\}, and \\ \mathcal{S}'_5 = \emptyset. \\ \text{Hence, } \boldsymbol{\ell(\mathcal{A})} = \mathcal{S}'_1 \cup \mathcal{S}'_2 \cup \mathcal{S}'_3 \cup \mathcal{S}'_4 \cup \mathcal{S}'_5. \\ \qquad\qquad = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_4. \end{array}}$$

Indeed $\langle \mathcal{T}, \mathcal{S}_1 \cup \mathcal{S}_2 \rangle$ and $\langle \mathcal{T}, \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_4 \rangle$ are consistent. However $\langle \mathcal{T}, \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \rangle$ and $\langle \mathcal{T}, \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_4 \cup \mathcal{S}_5 \rangle$ are both inconsistent. ■

In the following sections, we will present three new strategies that only select one preferred repair.

5 Non-Defeated Repair

Another way to get one preferred repair is to iteratively retrieve, layer per layer, the set of *free* elements. More precisely:

Definition 6 Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized *DL-Lite* KB. We define the non-defeated repair, denoted by $nd(\mathcal{A}) = \mathcal{S}'_1 \cup \cdots \cup \mathcal{S}'_n$, as follows:

$\forall i = 1, \ldots, n: \mathcal{S}'_i = free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i)$.

Namely, $nd(\mathcal{A}) =$

$free(\mathcal{S}_1) \cup free(\mathcal{S}_1 \cup \mathcal{S}_2) \cup \cdots \cup free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n)$.

The definition of non-defeated **subset** is an adaptation of the definition proposed in [11] within a propositional logic setting. However, contrarily to the propositional setting, as we will see later, the non-defeated repair can be applied on $\mathcal{A}$ or its deductive closure $c\ell(\mathcal{A})$ which leads to two different ways to select a single preferred repair. Besides the non-defeated repair is computed in polynomial time in a *DL-Lite* setting while its computation is hard in a propositional logic setting.

Algorithm 4 gives the computation of the non-defeated repair. Algorithm 4 first computes the set of conflicts (step 1). Step 2 simply initializes $nd(\mathcal{A})$ to an empty set. The expression:

$\{f{:}f \in \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i \text{ and } \exists g \in \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i \text{ such that } \{f, g\} \in \mathcal{C}\}$

represents the set of conflicting elements in $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i$. Hence, Step 4 computes the set of free elements in

$\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i$. Step 5 adds this result to $nd(\mathcal{A})$. Clearly, Algorithm 4 straightforwardly implements Definition 6. In Algorithm 4 the set of conflicts is computed once. Hence, the complexity of Algorithm 4 is *Conf* (step 1) plus $\mathcal{O}(n)$ (steps 2–6), where n is the number of strata in the *DL-Lite* knowledge base $\mathcal{K}$.

The following proposition shows that the non-defeated repair is consistent and its computation is in polynomial time.

Proposition 1 *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized DL-Lite KB. Let $nd(\mathcal{A})$ be its non-defeated repair. Then:*

(i) *$\langle \mathcal{T}, nd(\mathcal{A}) \rangle$ is consistent, and*
(ii) *The complexity of computing $nd(\mathcal{A})$ is in P.*

Proof Recall that: $nd(\mathcal{A}) =$

$$free(\mathcal{S}_1) \cup free(\mathcal{S}_1 \cup \mathcal{S}_2) \cup \cdots \cup free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n).$$

Recall also that if C is a conflict then either a C is singleton or C is a doubleton. Now, assume that $nd(\mathcal{A})$ is inconsistent. Then this means that there exists a conflict C of $\langle \mathcal{T}, \mathcal{A} \rangle$ such that $C \subseteq nd(\mathcal{A})$.

- Assume that $C = \{f\}$ is a singleton and $\mathcal{S}_i$ is the first layer where $f \in \mathcal{S}_i$ (namely, $\forall j < i, f \notin \mathcal{S}_j$). This means that:
  - $\forall\, j < i, f \notin free(\mathcal{S}_1 \cup \cdots \cup free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_j)$ (since $f \notin \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{i-1}$),
  - $\forall j \geq i, f \notin free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_j)$ (since *free* only contains non-conflicting information).

Hence, $f \notin free(\mathcal{S}_1) \cup \cdots \cup free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n)$.

Namely, $f \notin nd(\mathcal{A})$. Now, assume that $C = \{f, g\}$ is a doubleton. Let $\mathcal{S}_i$ (resp. $\mathcal{S}_j$) be the first layer containing f (resp. g). Let us assume that $i \leq j$. Then **clearly**

$$C \nsubseteq free(\mathcal{S}_1) \cup \cdots \cup free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{j-1})$$

since $f \nsubseteq \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{j-1}$.

Besides, for all $k \geq j$, we have:

$$free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_k) \cap C = \emptyset.$$

Hence, using the definition of *free* assertion, we get:

$$C \nsubseteq free(\mathcal{S}_1) \cup \cdots \cup free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n).$$

Hence $nd(\mathcal{A})$ contains no conflict and it is consistent. Regarding the computational complexity, recall that computing conflicts is done in a polynomial time. Since, the set of free assertions can be obtained in a linear time with respect to the set of conflicts, then the whole computation of $nd(\mathcal{A})$is also done in polynomial time. □

Algorithm 4 Non-Defeated Repair

Input: $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{A} = (\mathcal{S}_1, \cdots, \mathcal{S}_n)$
Output: A flat assertional base $nd(\mathcal{A})$
1: $\mathscr{C} \leftarrow \boldsymbol{\mathcal{C}(\mathcal{A})}$ { List of conflicts}
2: $nd(\mathcal{A}) \ \leftarrow \emptyset$
3: **for** $i = 1$ to n **do**
4: $\quad \boldsymbol{nd(\mathcal{A})} \leftarrow \boldsymbol{nd(\mathcal{A})} \cup$ FREE$(\boldsymbol{\mathcal{S}_1} \cup \cdots \cup \boldsymbol{\mathcal{S}_i}, \mathscr{C})$
5: **return** $nd(\mathcal{A})$

Example 7 (Example 1 continued). Following Algorithm 4, we have:

- $free(\mathcal{S}_1) = \{Student(Bill), Teacher(John)\}$,
- $free(\mathcal{S}_1 \cup \mathcal{S}_2) = \{Student(Bill), Teacher(John), Student(Bob), Teacher(Mary)\}$,
- $free(\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3) = \{Student(Bob), Teacher(John), Teacher(Mary), Researcher(Joe)\}$,
- $free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_4) = \{Student(Bob), Teacher(John), Teacher(Mary), Student(Anne), Researcher(Joe)\}$, and
- $free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_5) = \{Teacher(John), Teacher(Mary), Student(Anne), Researcher(Joe), Student(Bob)\}$.

Therefore: $nd(\mathcal{A}) = \{Student(Bill), Teacher(John), Researcher(Joe), Student(Bob), Teacher(Mary), Student(Anne)\}$. ■

Clearly, we have $\pi(\mathcal{A}) \subseteq \ell(\mathcal{A})$ and $\pi(\mathcal{A}) \subseteq nd(\mathcal{A})$ but $\ell(\mathcal{A})$ and $nd(\mathcal{A})$ are in general incomparable, as it is illustrated by the following propositions and examples.

Proposition 2 *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent and prioritized DL-Lite knowledge base, then:*

(i) *$\pi(\mathcal{A}) \subseteq \ell(\mathcal{A})$, and*
(ii) *$\pi(\mathcal{A}) \subseteq nd(\mathcal{A})$.*

Proof If $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent then trivially using the definition of possibilistic-based repair, linear-based repair and non-defeated based repair, we have:

$$\pi(\mathcal{A}) = \ell(\mathcal{A}) = nd(\mathcal{A}) = \boldsymbol{\mathcal{S}_1} \cup \cdots \cup \boldsymbol{\mathcal{S}}_i.$$

Now, assume that $\langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent. This means that there exists a rank i such that $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i$ is consistent but $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{i+1}$ is inconsistent (namely, $Consrank(\mathcal{K}) = i$). Then by definition of possibilistic-based repair, we have:

$$\pi(\mathcal{A}) = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i.$$

Clearly, using the definition of linear-based repair we also have:

$\pi(\mathcal{A}) \subseteq \ell(\mathcal{A})$.

Similarly, since $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i$ is consistent then:

$free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i) = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i = \pi(\mathcal{A})$.

Now since $nd(\mathcal{A})$ is defined by:

$nd(\mathcal{A}) free(S_1) \cup free(S_1 \cup S_2) \cup \cdots \cup free(S_1 \cup \cdots \cup S_n)$,

then we trivially have: $\pi(\mathcal{A}) \subseteq nd(\mathcal{A})$. □

The following example shows a situation where $\ell(\mathcal{A})$ and $nd(\mathcal{A})$ are not included in $\pi(\mathcal{A})$.

Example 8 Let us consider the following KB:

$\mathcal{T} = \{A \sqsubseteq \neg B\}$ and
$\mathcal{A} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$,

where:

$\mathcal{S}_1 = \{A(a)\}$,
$\mathcal{S}_2 = \{B(a)\}$, and
$\mathcal{S}_3 = \{C(a)\}$.

Using the definitions of possibilistic-based repair, linear-based repair and non-defeated repair, we have:

$\pi(\mathcal{A}) = \{A(a)\}, nd(\mathcal{A}) = \ell(\mathcal{A}) = \{A(a), C(a)\}$.

Clearly, $C(a) \in \ell(\mathcal{A}), C(a) \in nd(\mathcal{A})$ but $C(a) \notin \pi(\mathcal{A})$. ■

6 The Notion of Prioritized Closure

We now introduce the concept of a prioritized closure then check which among the above approaches (possibilistic-based repair, linear-based repair and non-defeated repair) are sensitive to the use of the deductive closure. In fact, the three preferred repairs given in the previous sections can be either defined on $\langle \mathcal{T}, \mathcal{A} \rangle$ or on $\langle \mathcal{T}, c\ell(\mathcal{A}) \rangle$ where $c\ell(\mathcal{A})$ denotes the deductive closure of a set of assertions $\mathcal{A}$ and it is defined as follows.

Definition 7 *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a flat DL-Lite KB. Let $\mathcal{T}_p$ be the set of all positive inclusion axioms of $\mathcal{T}$.*[1] *We define the deductive closure of $\mathcal{A}$ with respect to $\mathcal{T}$ as follows: $cl(\mathcal{A}) = \{B(a)$: $\langle \mathcal{T}_p, \mathcal{A} \rangle \models B(a)$ where, B is a concept of $\mathcal{T}$ and a is an individual of $\mathcal{A}\} \cup \{R(a,b)$: $\langle \mathcal{T}_p, \mathcal{A} \rangle \models R(a, b)$, where R is a role of $\mathcal{T}$ and a,b* are individuals of $\mathcal{A}\}$.

[1] Positive inclusion axioms are of the form $B_1 \sqsubseteq B_2$.

The use of a deductive closure of an ABox fully makes sense in DL languages. Indeed, one of the specificities of a DL settings is the separation between positive knowledge (positive axioms and facts) and negative knowledge (negative axioms). Therefore, even if the *DL-Lite* base is inconsistent, it is still possible to define a non-trivial deductive closure (using only positive knowledge) that does not produce the whole language. Of course, the concept of a deductive closure would also make sense in a propositional setting if only the positive Horn clauses were retained, which is essentially what Definition 7 does or if there is a way to separate positive from negative knowledge. The following definition extends Definition 7 to the prioritized case.

Definition 8 Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized *DL-Lite* KB. We define a prioritized closure of $\mathcal{A}$ with respect to $\mathcal{T}$, simply denoted by $c\ell(\mathcal{A})$, as follows: $c\ell(\mathcal{A}) = (\mathcal{S}'_1, \ldots, \mathcal{S}'_n)$ where:

$\forall i = 1, \ldots, n$:$\mathcal{S}'_i = c\ell(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{i-1} \cup \mathcal{S}_i)$.

Example 9 (Example 1 continued). Using Definition 8, we have: $c\ell(\mathcal{A}) = \mathcal{S}'_1 \cup \cdots \cup \mathcal{S}'_5$ where:

$$
\boxed{
\begin{aligned}
\mathcal{S}'_1 &= \{Student(Bill), Teacher(John)\},\\
\mathcal{S}'_2 &= \{Student(Bill), Teacher(John),\\
&\quad Student(Bob), Teacher(Mary)\},\\
\mathcal{S}'_3 &= \{Student(Bill), Teacher(John),\\
&\quad Student(Bob), Teacher(Mary),\\
&\quad HasSupervisor(Bob, Bill), Researcher(Joe),\\
&\quad Teacher(Joe), Teacher(Bill)\},\\
\mathcal{S}'_4 &= \{Student(Bill), Teacher(John),\\
&\quad Student(Bob), Teacher(Mary),\\
&\quad HasSupervisor(Bob, Bill), Researcher(Joe),\\
&\quad Teacher(Joe), Teacher(Bill), Student(Anne)\},\\
\mathcal{S}'_5 &= \{Student(Bill), Teacher(John),\\
&\quad Student(Bob), Teacher(Mary),\\
&\quad HasSupervisor(Bob, Bill), Researcher(Joe),\\
&\quad Teacher(Joe), Teacher(Bill), Student(Anne)\}.
\end{aligned}
}
$$

■

The first motivation of Definition 8 is that if an assertion f is derived from $\langle \mathcal{K}, \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n \rangle$ then f should belong to $c\ell(\mathcal{A})$. The second motivation is that if an assertion f is believed to some rank i then it should also be believed to all ranks that are higher than i. Namely, if f is derived from $\langle \mathcal{K}, \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i \rangle$ then $\forall j \geqslant i, f \in \mathcal{S}'_j$.

There exists another alternative definition to Definition 8 that avoids duplicating the set of derived conclusions. The idea is that a derived assertion f is added to rank i if and only if it can be obtained solely from $\mathcal{S}_i$. More precisely, we define the local prioritized closure of $\mathcal{A}$ with respect to $\mathcal{T}$, denoted by $\ell c\ell(\mathcal{A}) = (\mathcal{S}'_1, \dots, \mathcal{S}'_n)$, as follows:

$\forall i = 1, \dots, n, \mathcal{S}'_i = c\ell(\mathcal{S}_i)$.

An important feature of possibilistic and linear-based repairs is that they are insensitive to the local prioritized deductive closure. Namely, if one first uses the possibilistic-based repair (resp. linear-based repair) then applies local prioritized deductive closure will give the same result as if one first applies the local prioritized deductive closure then uses possibilistic-based repair (resp. linear-based repair). The situation is different if Definition 8 were used. In this case, possibilistic-based repair is still insensitive to the deductive closure, which is not the case with linear-based repair as it is summarized by the following proposition.

Proposition 3 *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized DL-Lite knowledge base. Then:*

(i) $\langle \mathcal{T}, c\ell(\pi(\mathcal{A})) \rangle = \langle \mathcal{T}, \pi(c\ell(\mathcal{A})) \rangle$
(ii) $\langle \mathcal{T}, \ell(c\ell(\mathcal{A})) \rangle \subseteq \langle \mathcal{T}, c\ell(\ell(\mathcal{A})) \rangle$.
(iii) $\langle \mathcal{T}, \ell(\ell c\ell(\mathcal{A})) \rangle = \langle \mathcal{T}, \ell c\ell(\ell(\mathcal{A})) \rangle$.

Proof Recall first that in standard *DL-Lite*, $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent if and only if $\langle \mathcal{T}, c\ell(\mathcal{A}) \rangle$ (resp. $\langle \mathcal{T}, \ell c\ell(\mathcal{A}) \rangle$) is consistent.

(i) If $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent, then trivially: $\boldsymbol{\pi(\mathcal{A}) = \mathcal{S}_1} \cup \dots \cup \boldsymbol{\mathcal{S}_n}$ and $\pi(c\ell(\mathcal{A})) = c\ell(\mathcal{A})$. Hence, $\pi(c\ell(\mathcal{A})) = c\ell(\pi(\mathcal{A}))$. Now, assume that $\langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent. This means that there exists an integer i such that: $\pi(\mathcal{A}) = \mathcal{S}_1 \cup \dots \cup \mathcal{S}_i$ is consistent but: $\pi(\mathcal{A}) \cup \mathcal{S}_{i+1}$ is inconsistent. Recall that $c\ell(\mathcal{A}) = (\mathcal{S}'_1, \dots, \mathcal{S}'_n)$ is such that: $\mathcal{S}'_1 = c\ell(\mathcal{S}_1)$, and

$\forall j = 2, \dots, n\ \mathcal{S}'_j = c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_{j-1} \cup \mathcal{S}_j)$.

Since $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_i$ is consistent and $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_{i+1}$ is inconsistent, then: $\mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_i$ is consistent and $\mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_{i+1}$ is inconsistent. Therefore

$\pi(c\ell(\mathcal{A})) = \mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_i$.

Besides, it is easy to check that:

$\mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_i = c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_i) = c\ell(\pi(\mathcal{A}))$.

Hence, $c\ell(\pi(\mathcal{A})) = \pi(c\ell(\mathcal{A}))$.

(ii) To see the proof it is enough to show that: $\ell(c\ell(\mathcal{A})) = \pi(c\ell(\mathcal{A}))$.

If $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent, then: $\ell(\mathcal{A}) = \mathcal{S}_1 \cup \dots \cup \mathcal{S}_n$ and $\ell(c\ell(\mathcal{A})) = c\ell(\mathcal{A})$. Hence trivially, we have:

$\ell(c\ell(\mathcal{A})) = c\ell(\ell(\mathcal{A})) = c\ell(\pi(\mathcal{A})) = \pi(c\ell(\mathcal{A}))$.

Assume that $\langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent. And again let i be such that $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_i$ is consistent but $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_{i+1}$ is inconsistent. This means that:

$\pi(c\ell(\mathcal{A})) = c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_i) = (\mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_i)$.

Recall that $c\ell(\mathcal{A})$ is such that:

$\mathcal{S}'_1 = c\ell(\mathcal{S}_1)$

and $\forall j = 2, \dots, n\ \mathcal{S}_j = c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_{j-1} \cup \mathcal{S}_j)$. Since, $\mathcal{S}_1 \cup \dots \cup$ $\boldsymbol{\mathcal{S}_{\geqq + \nmid}}$ is inconsistent. Then: $\forall j > i$, $\mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_j$ also is inconsistent. Therefore,

$ell(c\ell(\mathcal{A})) = \pi(c\ell(\mathcal{A}))$.

Now, it is easy to see that:

$\ell(c\ell(\mathcal{A})) = \pi(c\ell(\mathcal{A}))$.

(iii) Recall that $\ell c\ell(\mathcal{A}) = \mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_i$ is such that:

$\forall i = 1, \dots, \mathcal{S}'_i = c\ell(\mathcal{S}_i)$.

If $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent then $\pi(\mathcal{A}) = \mathcal{S}_1 \cup \dots \cup \mathcal{S}_n$. Since $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_n$ is **consistent**, then $c\ell(\mathcal{S}_1) \cup \dots \cup c\ell(\mathcal{S}_n) = \mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_n$ is also consistent. Hence,

$\pi(\ell c\ell(\mathcal{A})) = \mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_n = \ell c\ell(\pi(\mathcal{A}))$.

Assume that $\langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent. Let i be an integer such that $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_i$ is consistent but $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_{i+1}$ is inconsistent. This means that $\pi(\mathcal{A}) = \mathcal{S}_1 \cup \dots \cup \mathcal{S}_i$. This also means that $\mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_i$ is consistent but $\mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_{i+1}$ is inconsistent. Namely,

$$\pi(\ell c\ell(\mathcal{A})) = \mathcal{S}'_1 \cup \dots \cup \mathcal{S}'_i = \ell c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_i) \\ = \ell c\ell(\pi(\mathcal{A})).$$

□

The following is a counterexample of the converse of item (*ii*) of Proposition 3.

Example 10 Let us use the following knowledge base:

$\mathcal{T} = \{A \sqsubseteq \neg B, D \sqsubseteq E\}$,
$\mathcal{A} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$.

where:

$\mathcal{S}_1 = \{A(a)\}$,
$\mathcal{S}_2 = \{B(a), C(a)\}$, and
$\mathcal{S}_3 = \{D(a)\}$.

The prioritized closure of $\mathcal{A}$ is: $c\ell(\mathcal{A}) = (\mathcal{S}'_1, \mathcal{S}'_2, \mathcal{S}'_3)$ where:

$\mathcal{S}'_1 = \{A(a)\}$,
$\mathcal{S}'_2 = \{B(a), C(a), A(a)\}$, and
$\mathcal{S}'_3 = \{D(a), E(a), B(a), C(a), A(a)\}$.

The linear-base repair of $\langle \mathcal{T}, \mathcal{A} \rangle$ is:

$\ell(\mathcal{A}) = \{A(a), D(a)\}, \ell(c\ell(\mathcal{A})) = \{A(a)\}$

Besides, the closure of $\ell(\mathcal{A})$ is:

$c\ell(\ell(\mathcal{A})) = \{A(a), D(a), E(a)\}$.

Clearly, $E(a)$ can be obtained from $c\ell(\ell(\mathcal{A}))$ while it cannot be obtained from $\ell(c\ell(\mathcal{A})$. ■

The next section shows that the non-defeated repair is sensitive to the prioritized deductive closure which then leads to a new strategy for selecting repairs.

7 Adding Deductive Closure to Non-Defeated Repair

The non-defeated repair, when defined on $\mathcal{A}$, is safe since it only uses elements of $\mathcal{A}$ which are not involved in any conflict. One way to get a larger consistent repair is to use $c\ell(\mathcal{A})$ instead of $\mathcal{A}$. Namely, we define a closed non-defeated repair, denoted by $c\ell nd(\mathcal{A})$, as $S'_1 \cup \dots \cup S'_n$, such that:

$\forall i = 1, \dots, n$: $\mathcal{S}'_i = free(c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_i))$.

Let us illustrate this new strategy on our running example.

Example 11 (Example 1 continued). We have:

- $free(c\ell(\mathcal{S}_1)) = \{Student(Bill), Teacher(John)\}$,
- $free(c\ell(\mathcal{S}_1 \cup \mathcal{S}_2)) = \{Student(Bill), Teacher(John), Student(Bob), Teacher(Mary)\}$,
- $free(c\ell(\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3)) = \{Teacher(Mary), Teacher(John), Student(Bob), Researcher(Joe), Teacher(Joe)\}$,
- $free(c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_4)) = \{Teacher(Mary), Teacher(John), Student(Bob), Researcher(Joe), Teacher(Joe), Student(Anne)\}$, and
- $free(c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_5)) = \{Teacher(Mary), Teacher(John), Student(Bob), Researcher(Joe), Teacher(Joe), Student(Anne)\}$

Then $c\ell nd(\mathcal{A}) = \{Student(Bill), Teacher(John), Student(Bob), Teacher(Mary), Researcher(Joe), Teacher(Joe), Student(Anne)\}$.

One can see that $Teacher(Joe)$ is a new assertion that was not part of the non-defeated repair given in Example 7. ■

Note that strictly speaking $c\ell nd(\mathcal{A})$ is not a repair of $\langle \mathcal{T}, \mathcal{A} \rangle$ if one refers to Definition 1. Indeed, $c\ell nd(\mathcal{A})$ may be not included in $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_n$ since it may contain elements which are not explicitly stated in $\mathcal{A}$. However, to avoid introducing new notations and new concepts, we prefer to continue using the term repair when mentioning $c\ell nd(\mathcal{A})$. In fact, $c\ell nd(\mathcal{A})$ is a repair of $\langle \mathcal{T}, c\ell(\mathcal{A}) \rangle$, since $c\ell nd(\mathcal{A}) \subseteq c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_n)$.

Contrarily to the possibilistic and linear-based repairs, the following proposition shows that non-defeated repair is sensitive to the use of the deductive closure.

Proposition 4 *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized DL-Lite KB. Then*

(i) $nd(\mathcal{A}) \subseteq c\ell nd(\mathcal{A})$. *The converse is false.*
(ii) $c\ell(nd(\mathcal{A})) \subseteq c\ell nd(\mathcal{A})$. *The converse is false.*

Proof The proof follows from the fact that $\forall i = 1, \dots, n$ we have $free(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_i) \subseteq free(c\ell(\mathcal{S}_1 \cup \dots \cup \mathcal{S}_i))$. For the converse it is enough to consider the Example 12. □

Example 12 Let us the following knowledge base where: $\mathcal{T} = \{A \sqsubseteq \neg B, B \sqsubseteq C, A \sqsubseteq C\}$ and $\mathcal{A} = \mathcal{S}_1 = \{A(a), B(a)\}$.

One can check that $nd(\mathcal{A}) = \emptyset$, $cl(nd(\mathcal{A})) = \emptyset$ while $nd(c\ell(\mathcal{A})) = c\ell nd(\mathcal{A}) = \{C(a)\}$. ■

Algorithm 5 gives the result of adding the deductive closure to the non-defeated repair (which is obtained from Algorithm 4 by replacing $\mathcal{A}$ by its closure $c\ell(\mathcal{A})$).

Algorithm 5 Adding the Deductive Closure to Non-Defeated Repair

Input: $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{A} = (\mathcal{S}_1, \cdots, \mathcal{S}_n)$
Output: A flat assertional base $c\ell nd(\mathcal{A})$
1: $\mathscr{C} = \boldsymbol{C(c\ell(\mathcal{A}))}$ {The set of conflicts in $\langle \mathcal{T}, c\ell(\mathcal{A}) \rangle$}
2: $c\ell nd(\mathcal{A}) \leftarrow \emptyset$
3: **for** $i = 1$ to n **do**
4: $\boldsymbol{c\ell nd(\mathcal{A})} \leftarrow \boldsymbol{c\ell nd(\mathcal{A})} \cup$ FREE$(\boldsymbol{c\ell(\mathcal{S}_1} \cup \dots \cup \boldsymbol{\mathcal{S}_i), \mathscr{C}})$
5: **return** $c\ell nd(\mathcal{A})$

8 Adding Consistency to Non-Defeated Repair

We now present another new way to select a single preferred repair. It consists in slightly improving both linear-based repair and non-defeated repair. The idea is that in linear-based repair instead of ignoring a whole stratum in case of inconsistency, one can only ignore conflicting elements. The linear-based non-defeated repair, denoted by $\ell nd(\mathcal{A})$, is given by Definition 9.

Definition 9 Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized knowledge base. We define a linear-based non-defeated repair, denoted by $\ell nd(\mathcal{A}) = \mathcal{S}'_1 \cup \cdots \cup \mathcal{S}'_n$ as follows:

$$\mathcal{S}'_1 = free(\mathcal{S}_1),$$

for $i = 2, \ldots, n$ $\mathcal{S}'_i = free(\mathcal{S}'_1 \cup \cdots \cup \mathcal{S}'_{i-1} \cup \mathcal{S}_i)$.

Definition 9 is straightforwardly restated in Algorithm 6.

Algorithm 6 Linear-Based Non-Defeated Repair

Input: $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{A} = (\mathcal{S}_1, \cdots, \mathcal{S}_n)$
Output: A flat assertional base $\ell nd(\mathcal{A})$

```
1: 𝒞 ← C(A) { List of conflicts }
2: ℓnd(A) ← ∅
3: for i = 1 to n do
4:     ℓnd(A) ← ℓnd(A)∪ FREE(S_i ∪ ℓnd(A), 𝒞)
5: return ℓnd(A)
```

Example 13 (Example 1 continued). We have:

$$\begin{aligned}
\mathcal{S}'_1 &= free(\mathcal{S}_1) = \{Student(Bill), Teacher(John)\}\\
\mathcal{S}'_2 &= free(\mathcal{S}'_1 \cup \mathcal{S}_2)\\
&= \{Student(Bill), Teacher(John),\\
&\quad Teacher(Mary), Student(Bob)\}\\
\mathcal{S}'_3 &= free(\mathcal{S}'_1 \cup \mathcal{S}'_2 \cup \mathcal{S}_3)\\
&= \{Teacher(John), Teacher(Mary),\\
&\quad Researcher(Joe)\}\\
\mathcal{S}'_4 &= free(\mathcal{S}'_1 \cup \mathcal{S}'_2 \cup \mathcal{S}'_3 \cup \mathcal{S}_4)\\
&= \{Teacher(John), Teacher(Mary),\\
&\quad Researcher(Joe),\\
&\quad Student(Anne)\}\\
\mathcal{S}'_5 &= free(\mathcal{S}'_1 \cup \mathcal{S}'_2 \cup \mathcal{S}'_3 \cup \mathcal{S}'_4 \cup \mathcal{S}_5)\\
&= \{Teacher(John), Teacher(Mary),\\
&\quad Researcher(Joe), Student(Anne)\}\\
\ell nd(\mathcal{A}) &= \{Student(Bill), Teacher(John),\\
&\quad Teacher(Mary), Researcher(Joe),\\
&\quad Student(Anne), Student(Bob)\}.
\end{aligned}$$

■

Clearly, $\ell nd(\mathcal{A})$ extends $nd(\mathcal{A})$ by only focusing on elements in $\mathcal{S}_i$ that are consistent with $\mathcal{S}'_1 \cup \cdots \cup \mathcal{S}'_{i-1}$ (rather than with $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{i-1}$). The nice feature of ℓnd-repair is that the combination of ℓ-repair and nd-repair is done without extra computational cost. More precisely, computing $\ell nd(\mathcal{A})$ is in P. Clearly $\ell nd(\mathcal{A})$ is consistent and it is larger than $\pi(\mathcal{A})$ and $nd(\mathcal{A})$, namely

$\pi(\mathcal{A}) \subseteq \ell nd(\mathcal{A})$ and $nd(\mathcal{A}) \subseteq \ell nd(\mathcal{A})$.

But it remains incomparable with the other approaches presented above. Properties of linear-based non defeated repair are summarized in the proposition 5. Property (*iv*) states that the use of linear-based non-defeated repair (ℓnd) on prioritized closure $c\ell(\mathcal{A})$ simply leads to the definition of the closure non-defeated repair.

Proposition 5 *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a prioritized DL-Lite knowledge base. Then:*

(i) *$\ell nd(\mathcal{A})$ is consistent.*
(ii) $\pi(\mathcal{A}) \subseteq \ell nd(\mathcal{A})$.
(iii) $nd(\mathcal{A}) \subseteq \ell nd(\mathcal{A})$.
(iv) $\ell nd(c\ell(\mathcal{A}))= c\ell nd(\mathcal{A})$.

Proof The proof of item (i) follows exactly the same steps as the one used to establish the proof of item (i) of Proposition 1. Les us show the proof of item (ii). If $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent, then $\pi(\mathcal{A}) = \ell nd(\mathcal{A})$. Hence item (ii) is satisfied for consistent *DL-Lite* knowledge bases. Assume now that $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent. Recall that $\ell nd(\mathcal{A}) = \mathcal{S}'_1 \cup \cdots \cup \mathcal{S}'_n$ such that $\mathcal{S}'_1 = free(\mathcal{S}_1)$ and for $i = 2, \ldots, n$ $\mathcal{S}'_i = free(\mathcal{S}'_1 \cup \cdots \cup \mathcal{S}'_{i-1} \cup \mathcal{S}_i)$. Let i be such that $\pi(\mathcal{A}) = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i$ is consistent but $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i \cup \mathcal{S}_{i+1}$ is inconsistent. We have:

$\mathcal{S}'_1 = free(\mathcal{S}_1) = \mathcal{S}_1$,
$\mathcal{S}'_2 = free(\mathcal{S}_2 \cup \mathcal{S}'_1) = \mathcal{S}_1 \cup \mathcal{S}_2$, and
$\mathcal{S}'_i = free(\mathcal{S}_1 \cup \cdots . \boldsymbol{\mathcal{S}_i} \cup \mathcal{S}'_{i-1}) = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i$. Hence

$\pi(\mathcal{A}) \subseteq \ell nd(\mathcal{A})$.

Let us now give the proof of item (iii). Recall that $nd(\mathcal{A}) = \mathcal{S}''_1 \cup \cdots \cup \mathcal{S}''_n$ is such that:

$$\forall i = 1, \ldots, n\ \mathcal{S}''_i = free(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i).$$

Clearly, $\forall i = 1, \ldots, n$ $\mathcal{S}''_i \subseteq \mathcal{S}'_i$. Hence,

$$nd(\mathcal{A}) \subseteq \ell nd(\mathcal{A}).$$

Let us now show item (*iv*).

Let $c\ell(A) = (\mathcal{S}'_1, \ldots, \mathcal{S}'_n)$ be the prioritized closure of $\mathcal{A}$. Recall that by the definition of the prioritized closure we have:

$$\forall i = 1, \ldots, n, \mathcal{S}'_i = c\ell(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i).$$

Let $c\ell nd(\mathcal{A}) = \mathcal{S}_1'' \cup \cdots \cup \mathcal{S}_n''$ be the closed non-defeated repair defined by:

$$\mathcal{S}_1'' = free(c\ell(\mathcal{S}_1)) = free(\mathcal{S}_1')$$

and

$$\forall \boldsymbol{i} = 2, \ldots, \boldsymbol{n}, \mathcal{S}_i'' = \boldsymbol{free}(\boldsymbol{c\ell}(\mathcal{S}_\mathbf{1} \cup \cdots \cup \mathcal{S}_i)) = \boldsymbol{free}(\mathcal{S}_i').$$

Let $\ell nd(c\ell(\mathcal{A})) = \mathcal{S}_1''' \cup \cdots \cup \mathcal{S}_n'''$ be the repair obtained by applying linear-based non-defeated repair on $c\ell(\mathcal{A})$. By definition of ℓnd repair we have: $\mathcal{S}_1''' = free(\mathcal{S}_1') = \mathcal{S}_1''$, and

$$\forall \boldsymbol{i} = \mathbf{2}, \ldots, \boldsymbol{n}, \mathcal{S}_i''' = \boldsymbol{free}(\mathcal{S}_\mathbf{1}''' \cup \cdots \cup \mathcal{S}_{\boldsymbol{i-1}}''' \cup \mathcal{S}_i') = free(\mathcal{S}_i') = \mathcal{S}_i''.$$

Indeed, $\mathcal{S}_i' = c\ell(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_i)$ and

$$\forall j = 1, \ldots, i-1, \mathcal{S}_j''' \subseteq c\ell(\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_j).$$

□

The following two examples show that the ℓnd-repair remains incomparable with linear-based repair and $c\ell nd$-based repair.

Example 14 Let us consider the following knowledge base: $\mathcal{T} = \{A \sqsubseteq \neg B, B \sqsubseteq \neg C, A \sqsubseteq \neg D, D \sqsubseteq E\}$ $\mathcal{A} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ where:

$\mathcal{S}_1 = \{A(a)\}$,
$\mathcal{S}_2 = \{B(a), D(a)\}$, and
$\mathcal{S}_3 = \{C(a)\}$.

The ℓnd-repair $\ell nd(\mathcal{A}) = \mathcal{S}_1' \cup \mathcal{S}_2' \cup \mathcal{S}_3'$ is such that:

$\mathcal{S}_1' = free(\mathcal{S}_1) = \{A(a)\}$
$\mathcal{S}_2' = free(\mathcal{S}_1' \cup \mathcal{S}_2) = \emptyset$
$\mathcal{S}_3' = free(\mathcal{S}_3 \cup \mathcal{S}_1' \cup \mathcal{S}_2') = \{A(a), C(a)\}$.

Now, the $c\ell nd$-repair $c\ell nd(\mathcal{A}) = \mathcal{S}_1'' \cup \mathcal{S}_2'' \cup \mathcal{S}_3''$ is such that:

$\mathcal{S}_1'' = free(c\ell(\mathcal{S}_1)) = \{A(a)\}$.
$\mathcal{S}_2'' = free(c\ell(\mathcal{S}_1 \cup \mathcal{S}_2))$
$= free(\{A(a), B(a), D(a), E(a)\}) = \{E(a)\}$
$\mathcal{S}_3'' = free(c\ell(\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3)) = \{E(a)\}$

Clearly, we have:

$C(a) \in \ell nd(\mathcal{A})$ while $E(a) \notin \ell nd(\mathcal{A})$.

Similarly, $E(a) \in c\ell nd(\mathcal{A})$ but $C(a) \notin c\ell nd(\mathcal{A})$.

Hence, ℓ nd-repair is incomparable with $c\ell nd$-repair. ■

Example 15 Let us consider the following knowledge base:

$\mathcal{T} = \{A \sqsubseteq \neg B, D \sqsubseteq \neg C\}$, and
$\mathcal{A} = (\mathcal{S}_1, \mathcal{S}_2)$,

where:

- $\mathcal{S}_1 = \{A(a), B(a), E(a), D(a)\}$, and
- $\mathcal{S}_2 = \{C(a)\}$.

We have $\ell(\mathcal{A}) = \{C(a)\}$.

Besides, we have:

- $free(\mathcal{S}_1) = \{E(a), D(a)\}$ and
- $free(\mathcal{S}_2 \cup free(\mathcal{S}_1)) = \{E(a)\}$.

Hence $\ell nd(\mathcal{A}) = \{E(a), D(a)\}$.

Therefore, $\ell(\mathcal{A})$ and $\ell nd(\mathcal{A})$ are incomparable. ■

The following figure summarizes different set inclusion relations between different approaches presented in this paper:

In Fig. 1, The arc $X \rightarrow Y$ means that the repair X is included in the repair Y. The proof of the inclusion relations 1 and 2 (resp. 3 and 4) is given in Propositon 2 (resp. Propositions 5 and 4). A counterexample for the inverse of relations 1 and 2 (resp. 3 and 4) is given in Example 8 (resp. Example 12). Lastly, the incomparability relation between $\ell nd(\mathcal{A})$ and $c\ell nd(\mathcal{A})$ (resp. $\ell(\mathcal{A})$ vs $\ell nd(\mathcal{A})$) is illustrated by Example 14 (resp. Example 15) of the proof of Proposition 5

As it was stated previously, $c\ell nd(\mathcal{A})$ is not a repair of $\langle \mathcal{T}, \mathcal{A} \rangle$ but a repair of $\langle \mathcal{T}, c\ell(\mathcal{A}) \rangle$. Hence, the productivity of $c\ell nd(\mathcal{A})$ is not very surprising since it considers largest ABox than the

other repairs.

9 Experimental Studies

For experimental evaluation, we implemented in Java algorithms for computing a single preferred repair.

These algorithms are built upon two main components: a *DL-Lite* ontology parser (that parses *DL-Lite* knowledge

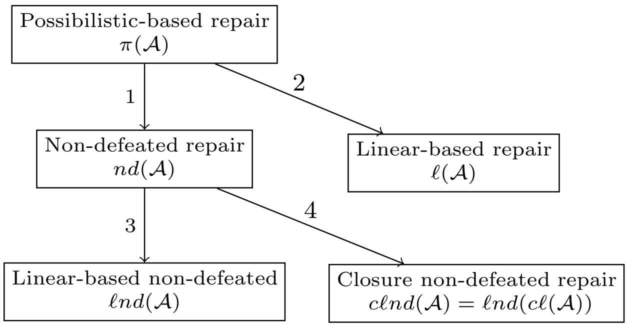

Fig. 1 A comparative study between $\pi(\mathcal{A})$, $\ell(\mathcal{A})$, $nd(\mathcal{A})$, $c\ell nd(\mathcal{A})$, and $\ell nd(\mathcal{A})$

bases expressed in *OWL2-QL* function syntax) **and a SQLite database engine**. As one can observe in the proposed algorithms, they are mainly based on some of the following operations: consistency checking, computing conflict sets and deductive closure. To efficiently compute conflicting elements (and inconsistency checking), we evaluate over the ABox (stored as a relational DB) queries expressed from the negated closure of the TBox [17] to exhibit whether the ABox contains conflicting elements. The negative closure of an ABox is made of the list of all negative axioms (of the form $A \sqsubseteq \neg B$) that can be derived from $\mathcal{T}$ by applying positive rules on negative ones (for more details see [17]). Note that the negated closure can be pre-computed and kept in memory during all experimentations. When all these ingredients are available, we proceed to computing the single preferred repair using different algorithms proposed in this paper.

9.1 Experimental Setting

We first present the setting used for our experimental studies. As benchmark[2] we considered the LUBM$^{\exists}$20 ontology (i.e. TBox), which corresponds to the *DL-Lite*$_R$ version of the original LUBM ontology [29], and we used the Extended University Data Generator (EUDG) in order to generate the ABox assertions. The LUBM$^{\exists}$20 ontology contains only 208 positive inclusion axioms. To this end, we used the set of negative inclusion axioms from [14] in order to allow for inconsistency. Computing the negated closure of this ontology is done in 876.906 s. Note that the time needed to compute the negated closure will not be taken into account as parameter in our experimental studies since it is computed only once and then stored in memory. Once the negated closure is computed, we use an SQL engine to compute conflict sets (and inconsistency check). This allows an efficient handling of inconsistency. For example, checking consistency for an ABox containing 2432 facts through query evaluation (after storing the ABox into a database) takes 0.952 s while without using query evaluation it takes 812.4 s. Similarly, we use an SQL engine to compute conflict sets. It is important to mention here that the time needed to compute conflict sets is basically the same as the one needed to check consistency. Finally, we consider three different cases:

Case 1: We generated (using EUDG) four ABoxes and we split them respectively into 3 strata, 5 strata and then 7 strata. These ABoxes contain respectively 2532, 2832, 3432, and 4432 assertions with respectively 50, 200, 500, and 1000 conflict sets. The results of this case are shown in Table 1.

Case 2: We randomly generated an ABox and we varied the number of strata from 1 to 15. For each number of strata, we partitioned the ABox uniformly and we ran the proposed algorithms for computing repairs. The results of this case are presented in Table 2.

Case 3: We considered the ABox used in Case 2 and we fixed the number of strata (7 strata). We then varied the percentage of conflicting elements from 0 to 100% of the size of the ABox. In each scenario we ran all the algorithms. The results of this case are given in Table 3.

In all cases, we partitioned the Box uniformly and we ran the proposed algorithms for computing repairs.

9.2 Experimental Results

We analyse in this section our experimental results. We first analyse results of Case 1, given in Table 1. From a productivity point of view, the possibilistic-based repair is very cautious comparing to the other strategies. Unsurprisingly, this consequently ensures low productivity in terms of the size of the selected repair. Namely, for a given ABox and a given number of strata, possibilistic-based repair has the largest number of dropped elements. For instance, when a conflict is detected in the first stratum, the whole ABox is discarded (hence, 100% of the assertions will be dropped as shown in the first row of Table 1). This similarly holds for the linear-based repair when there exists at least a conflict in each strata. As shown on the results of Table 1, computing the π-repair requires on most our experiments less time than for computing the other repairs. Regarding the computation of the non-defeated repair (resp. closed non-defeated repair), it depends on the number of conflicts in the ABox. More precisely, the time needed for computing the non-defeated repair (resp. closed non-defeated repair) increases with the size of conflicts in the ABox.

[2] Available at: https://code.google.com/p/combo-obda/.

Table 1 Percentage of deleted facts and time (in seconds) taken to compute conflicts, π, ℓ, nd, ℓnd, and $c\ell nd$ while varying the number of conflicts and number of strata for a fixed ABox

# Conflict	# Strata	Conflicts	π		ℓ		nd		ℓnd		$c\ell nd$	
			$\mathcal{O}(log_2(nbs)) * Conf$		$Conf * nbs$		$\mathcal{O}(nbs) + Conf$		$\mathcal{O}(nbs) + Conf$		$\mathcal{O}(nbs) + Conf$	
			Time	%	Time	%	Time	%	Time	%	Time	%
50	3	1451	21.47	100	22.19	100	94.91	2.98	159.47	2.98	458.73	45.04
	5	1601	21.92	79.44	65.31	78.63	156.35	0.48	257.61	0.44	907.65	36.31
	7	1311	32.72	85.06	65.84	67.82	188.52	0.48	340.98	2.17	1284.90	36.63
200	3	2965	24.04	100	56.35	100	91.81	14.07	185.15	15.13	478.24	44.74
	5	2885	24.94	81.39	66.19	81.39	134.86	14.05	272.25	14.05	956.20	36.19
	7	3258	33.07	59.40	77.07	51.04	193.99	15.04	414.73	11.43	1202.55	36.68
500	3	6288	27.72	86.94	28.83	86.94	101.37	28.40	293.60	28.40	497.08	52.65
	5	5591	36.24	86.74	59.97	86.74	186.72	26.29	488.98	26.29	1116.06	57.05
	7	6894	39.78	88.07	86.83	76.75	198.86	726.87	669.14	26.66	1424.63	56.95
1000	3	10500	57.26	100	57.66	87.15	126.60	45.53	722.23	45.53	1222.88	25.98
	5	11339	59.10	91.58	84.43	82.87	159.41	45.65	1246.23	45.65	1511.58	31.22
	7	11046	112.11	96.36	171.47	96.36	221.15	40.20	1897.57	40.23	1881.15	43.86

Table 2 The impact of the number of strata on the productivity and the computation time of the repairs

# strata	π		ℓ		nd		ℓnd		$c\ell nd$	
	Time	%	Time	%	Time	%	Time	%	Time	%
1	28.67	100	28.77	100	48.20	72.29	98.55	39.74	128.65	35.12
3	22.74	100	24.69	100	96.27	39.65	161.52	19.67	399.98	35.04
5	24.01	88.66	30.35	77.01	147.36	37.35	243.82	26.66	915.95	27.85
7	23.11	89.80	51.26	80.12	189.95	37.29	314.54	27.02	1287.85	27.80
9	33.72	92.01	58.68	91.84	255.03	37.20	555.74	28.50	2328.47	27.68
11	21.67	100	61.07	79.57	271.81	36.45	1335.72	28.42	2448.05	27.68
13	23.90	100	61.85	78.72	314.18	36.42	1608.26	28.42	2979.21	27.61
15	30.89	100	64.18	77.85	355.38	35.89	1842.48	28.40	3846.21	20.36

Finally, for the linear non-defeated repair, two parameters may influence the time taken to compute it, namely the number of layers and the number of conflicts. The proportion of dropped assertions regarding the different repairs confirms the relationships between π, ℓ, nd, ℓnd, and $c\ell nd$ as stated in the previous sections. Table 1 also summarizes time complexity. *Cons* denotes the time complexity of consistency check in standard *DL-Lite*. *Conf* is the time complexity of computing conflicts. *nbs* is the number of strata in standard *DL-Lite*.

From Table 1, it is obvious that the number of strata and the number of conflicting elements in the assertional bases are the two main parameters that influence the productivity of the repairs and the time needed to compute them. Tables 2 and 3 analyse separately these two parameters.

Table 2 shows the impact of the number of strata on the productivity (in terms of the percentage of the deleted assertions when computing the repairs) and the computation of the repairs. For each ABox (used in Case 1), and by only varying the number of layers, one can observe that the productivity of $c\ell nd(\mathcal{A})$, $\ell nd(\mathcal{A})$, and $nd(\mathcal{A})$ increases proportionally with the number of strata. As a negative effect, the running time also increases proportionally with respect to the number of strata. For $\ell(\mathcal{A})$, the running time is better than the ones of $c\ell nd(\mathcal{A})$, $\ell nd(\mathcal{A})$, and $nd(\mathcal{A})$, and also increases proportionally with respect to the number of strata.

Contrarily to Tables 2, 3 focuses on the impact of the number of conflicts on the time taken to compute the repairs and on the percentage of deleted facts. For a fixed number of stratification (7 strata), we vary the number of conflicts between 0 and 100% of the size of ABox. One can observe that for all strategies, the running time and the set of deleted facts increases with the percentage of conflicting

Table 3 Impact of the number of conflicting elements on productivity and computation time of the repairs

% Conflicts	π		ℓ		nd		ℓnd		$c\ell nd$	
	Time	%	Time	%	Time	%	Time	%	Time	%
0	86.92	0	111.08	0	131.72	0	370.25	0	597.12	0
10	18.96	69.27	27.86	7.63	108.90	3.63	422.68	3.63	682.24	8.46
20	20.19	71.83	31.56	9.27	116.84	7.00	540.77	7.00	817.59	6.75
30	21.87	74.00	35.23	57.30	157.66	12.84	653.57	12.84	820.25	16.30
40	21.14	75.85	51.53	50.64	142.94	15.07	836.99	15.07	1318.68	28.77
50	30.97	100	45.49	53.53	199.97	19.40	840.44	19.40	1495.83	31.19
60	31.59	100	48.97	56.5	178.30	23.18	1070.42	23.18	1453.53	32.65
70	32.32	100	54.08	54.05	262.86	24.17	1071.20	24.17	1341.92	41.32
80	34.48	100	58.77	90.77	249.04	26.77	1379.35	26.77	1501.09	42.11
90	41.41	100	60.89	97.31	281.91	28.31	1513.00	25.68	1604.95	46.52
100	45.46	100	64.14	100	342.45	29.10	1754.43	25.95	2156.46	46.91

elements in the ABox. Table 3 confirms that $c\ell nd(\mathcal{A})$, $\ell nd(\mathcal{A})$ and $nd(\mathcal{A})$ are more productive than $\ell(\mathcal{A})$ and $\pi(\mathcal{A})$ since they remove less facts than $\ell(\mathcal{A})$ and $\pi(\mathcal{A})$.

Lastly, it is important to note that computing assertional conflicts can be done in an incremental way (it is enough to only compute the new conflicts raised by the arrival of a new assertion or a new axiom). Computing conflicts in an incremental way has an impact on our algorithms in order to efficiently update the set of free assertions or for updating the inconsistency degree (which can only decrease when new assertions are added). Besides, most of the repairs presented in the paper are defined incrementally, starting from the layer 1 until layer n. Hence, if some new assertions are added to a layer i, then all computations already done on layers $1, \ldots i-1$ remain valid.

10 Conclusion

This paper focused on how to select a single preferred repair from a prioritized inconsistent *DL-Lite* knowledge base. Selecting only one repair is important since it allows efficient query answering once the preferred repair is computed. We first reviewed some well-known approaches that select one repair such as possibilistic repair or linear-based repair. Then, we presented different strategies for selecting one preferred repair based on the non-defeated repair by adding deductive closure and consistency criterion. In the context of inconsistent lightweight ontologies, the approaches based on non-defeated repairs (nd, ℓnd, $c\ell nd$) have not been proposed before. In particular, the concept of 'prioritized closure' used in this paper, for defining new inconsistency tolerant approaches, has not been considered before. The common feature of all these repairs is that they produce as many safe conclusions as possible and, as shown in our experimental studies, they allow an efficient handling of inconsistency. Such facts make all the repairs suitable for *DL-Lite*.

Acknowledgements The authors would like to thank referees for their useful remarks and comments. This work has received support from the European project H2020 Marie Sklodowska-Curie Actions (MSCA) research and Innovation Staff Exchange (RISE): *AniAge* (High Dimensional Heterogeneous Data based Animation Techniques for Southeast Asian Intangible Cultural Heritage Digital Content), Project Number 691215.

References

1. Arenas M, Bertossi EL, Chomicki J (1999) Consistent query answers in inconsistent databases. In: Proceedings of the eighteenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, Philadelphia, Pennsylvania, USA, pp 68–79, 1999
2. Artale A, Calvanese D, Kontchakov R, Zakharyaschev M (2009) The DL-Lite family and relations. J Artif Intell Res (JAIR) 36:1–69
3. Baral C, Kraus S, Minker J, Subrahmanian VS (1992) Combining knowledge bases consisting of first-order analysis. Comput Intell 8:45–71
4. Benferhat S, Bouraoui Z (2015) Min-based possibilistic DL-Lite. J Logic Comput. doi:10.1093/logcom/exv014
5. Benferhat S, Bouraoui Z, Papini O, Würbel E (2014) A prioritized assertional-based revision for DL-Lite knowledge bases. In: European conference on logics in artificial intelligence, volume 8761 of LNCS, pp 442–456. Springer, 2014
6. Benferhat S, Bouraoui Z, Tabia K (2015) How to select one preferred assertional-based repair from inconsistent and prioritized DL-Lite knowledge bases? In: Yang Q, Wooldridge M (eds) Proceedings of the twenty-fourth international joint conference on artificial intelligence IJCAI 2015, Buenos Aires, Argentina, July 25–31, 2015, pp 1450–1456. AAAI Press, 2015
7. Benferhat S, Cayrol C, Dubois D, Lang J, Prade H (1993) Inconsistency management and prioritized syntax-based entailment. In: International joint conference on artificial intelligence, pp 640–647. Morgan Kaufmann, 1993

8. Benferhat S, Didier D, Henri P (1997) Some syntactic approaches to the handling of inconsistent knowledge bases: a comparative study part 1: The flat case. Studia Logica 58(1):17–45
9. Benferhat S, Dubois D, Prade H (1992) Representing default rules in possibilistic logic. In: Knowledge representation and reasoning, pp 673–684. Morgan Kaufmann, 1992
10. Benferhat S, Dubois D, Prade H (1995) How to infer from inconsistent beliefs without revising? In: International joint conference on artificial intelligence, pp 1449–1457. Morgan Kaufmann, 1995
11. Benferhat S, Dubois D, Prade H (1998) Some syntactic approaches to the handling of inconsistent knowledge bases: a comparative study. Part 2: the prioritized case, volume 24, pp 473–511. Physica-Verlag, Heidelberg, 1998
12. Bertossi LE (2011) Database repairing and consistent query answering. Synthesis lectures on data management. Morgan & Claypool Publishers, San Rafael
13. Bienvenu M (2012) On the complexity of consistent query answering in the presence of simple ontologies. In: Proceedings of the twenty-sixth AAAI conference on artificial intelligence, 2012
14. Bienvenu M, Bourgaux C, Goasdoué F (2014) Querying inconsistent description logic knowledge bases under preferred repair semantics. In: AAAI, pp 996–1002, 2014
15. Bienvenu M, Rosati R (2013) Tractable approximations of consistent query answering for robust ontology-based data access. In: International joint conference on artificial intelligence. IJCAI/AAAI, 2013
16. Brewka G (1989) Preferred subtheories: an extended logical framework for default reasoning. In: Sridharan NS (ed) International joint conference on artificial intelligence, pp 1043–1048. Morgan Kaufmann, 1989
17. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2007) Tractable reasoning and efficient query answering in description logics: the DL-Lite family. J Autom Reason 39(3):385–429
18. Calvanese D, Kharlamov E, Nutt W, Zheleznyakov D (2010) Evolution of DL-Lite knowledge bases. Int Semant Web Conf 1:112–128
19. Chomicki J (2007) Consistent query answering: five easy pieces. In: Database theory—ICDT 2007, volume 4353 of lecture notes in computer science, pp 1–17. Springer, 2007
20. Didier D, Lang J, Henri P (1994) Possibilistic logic. In: Volume 3 of handbook on logic in artificial intelligence and logic programming, pp 439–513. Oxford University press, 1994
21. Du J, Qi G, Shen Y (2013) Weight-based consistent query answering over inconsistent SHIQ knowledge bases. Knowl Inform Syst 34(2):335–371
22. Dubois D, Prade H (1988) Possibility theory. Plenum Press, New York
23. Dubois D, Prade H (1991) Epistemic entrenchment and possibilistic logic. Artif Intell 50(2):223–239
24. Lembo D, Lenzerini M, Rosati R, Ruzzi M, Fabio Savo D (2010) Inconsistency-tolerant semantics for description logics. In: Hitzler P, Lukasiewicz T (eds) RR, volume 6333 of LNCS, pp 103–117. Springer, 2010
25. Lembo D, Lenzerini M, Rosati R, Ruzzi M, Fabio Savo D (2015) Inconsistency-tolerant query answering in ontology-based data access. J Web Semant 33:3–29
26. Lenzerini M (2012) Ontology-based data management. In: Proceedings of the 6th Alberto Mendelzon international workshop on foundations of data management 2012, pp 12–15, 2012
27. Lukasiewicz T, Vanina Martinez M, Pieris A, Simari GI (2015) From classical to consistent query answering under existential rules. In: Proceedings of the twenty-ninth AAAI conference on artificial intelligence 2015, pp 1546–1552, 2015
28. Lukasiewicz T, Vanina Martinez M, Simari GI (2012) Inconsistency handling in datalog+/− ontologies. In: 20th European conference on artificial intelligence ECAI, 2012, pp 558–563, 2012
29. Lutz C, Seylan I, Toman D, Wolter F (2013) The combined approach to OBDA: taming role hierarchies using filters. In: The semantic web—ISWC, volume 8218 of lecture notes in computer science, pp 314–330. Springer, 2013
30. Martinez MV, Parisi F, Pugliese A, Simari GI, Subrahmanian VS (2008)Inconsistency management policies. In: Knowledge representation and reasoning, pp 367–377. AAAI Press, 2008
31. Nebel B (1994) Base revision operations and schemes: semantics, representation and complexity. In: European conference on artificial intelligence, pp 341–345, 1994
32. Nicholas R, Ruth M (1970) On inference from inconsistent premisses. Theory Decis 1(2):179–217
33. Poggi A, Lembo D, Calvanese D, De Giacomo G, Lenzerini M, Rosati R (2008) Linking data to ontologies. J Data Semant 10:133–173
34. Qi G, Ji Q, Pan JZ, Du J (2011) Extending description logics with uncertainty reasoning in possibilistic logic. Int J Intell Syst 26(4):353–381
35. Staworko S, Chomicki J, Marcinkowski J (2012) Prioritized repairing and consistent query answering in relational databases. Ann Math Artif Intell 64(2–3):209–246