Sapienza University of Rome

Ph.D. program in Computer Engineering

XXIV Cycle

Dealing with Inconsistencies and Updates in Description Logic Knowledge Bases

Domenico Fabio Savo

Sapienza University of Rome

Ph.D. program in Computer Engineering

XXIV Cycle

Domenico Fabio Savo

Dealing with Inconsistencies and Updates in Description Logic Knowledge Bases

Thesis Committee

Prof. Maurizio Lenzerini (Advisor)
Prof. Paolo Liberatore

Reviewers

Prof. Alexander Borgida
Prof. Marie-Christine Rousset

Author's address:
Domenico Fabio Savo
Dipartimento di Ingegneria Informatica Automatica e Gestionale
Antonio Ruberti
Sapienza Università di Roma
Via Ariosto 25, 00185 Roma, Italy
e-mail: savo@dis.uniroma1.it
www: http://www.dis.uniroma1.it/∼savo

To my family

Acknowledgements

I would like to express my sincere appreciation and gratitude to my advisor Prof. Maurizio Lenzerini, who has shared his board experience with me and has strongly supported and guided me throughout my Ph.D. course and the fulfillment of this work. I will always be indebted to him for teaching me how to become a researcher.

I also wish to thank Prof. Riccardo Rosati and Prof. Giuseppe De Giacomo for being always eager to advise me regarding my research.

A special thank goes to Domenico Lembo, who has spent several hours discussing with me on various issues ranging from my Ph.D. Thesis to my future as researcher. Working with him is a privilege.

I would like to express my gratitude to Antonella Poggi and Marco Ruzzi, which gave me a very important help. I am grateful to them for their advises and friendship.

I would like to thank all the people of the Dipartimento di Ingegneria Informatica Automatica e Gestionale of the University of Rome La Sapienza, Prof. Paolo Liberatore as internal referees, Prof. Alexander Borgida and Prof. Marie-Christine Rousset as external referees, the members of the Ph.D. Committee at Dipartimento di Ingegneria Informatica Automatica e Gestionale and all the people of which I enjoyed the company during my Ph.D..

Finally, I am very grateful to my parents, Antonio and Laura, my sister Valentina, my lovely girlfriend Francesca, and all my friends for their constant and important support during my studies.

Contents

Part I

Introduction and Background

Chapter 1

Introduction

While the amount of data stored in current information systems continuously grows, turning these data into information is still one of the most challenging tasks for Information Technology. The task is complicated by the proliferation of data sources both in single organizations, and in open environments. Specifically, the information systems of medium and large organizations are typically constituted by several, independent, and distributed data sources, and this poses great difficulties with respect to the goal of accessing data in a unified and coherent way. Such a unified access is crucial for getting useful information out of the system, as well as for taking decision based on them. This explains why organizations spend a great deal of time and money for the understanding, the governance, and the integration of data stored in different sources [54].

The following are some of the reasons why a unified and transparent access to the data sources of an organization is in general problematic.

- Despite the fact that the initial design of a collection of data sources (e.g., a database) is adequate, corrective maintenance actions tend to reshape the data sources into a form that often diverges from the original conceptual structure.
- It is common practice to change a data repository so as to adapt it both to specific application-dependent needs, and to new requirements. The result is that data sources often become data structures coupled to a specific application (or, a class of applications), rather than application-independent databases.
- The data stored in different sources tend to be redundant, and mutually inconsistent, mainly because of the lack of central, coherent and unified data management tasks.

Note that the above problems are still relevant even when the data of the organization is stored in a single data source.

In principle, there are two alternative solutions to the above problems. One solution is the re-engineering of the information system, i.e., the design of a new, coherent, and unified data repository serving all the applications of the organization [57], and replacing the original data sources. This approach is unfeasible in many situations, due to cost and organization problems. The other solution is to create a new stratum of the information system, co-existing with the data sources, according to the "data integration" paradigm [11]. Such new stratum is constituted by (*i*) a global (also called "mediated") schema, representing the unified structure presented to the clients, and (*ii*) the mapping relating the source data with the elements in the global schema. In turn, there are two methods for realizing such stratum, called materialized and virtual. In the materialized approach, the global schema is populated with concrete data deriving from the sources in accordance with the mapping. In the virtual approach, data are not moved, and queries posed to the system are answered by suitably accessing the sources [71].

Currently, in those information systems that embrace the latter approach, the global schema is expressed in terms of a logical database model, e.g. the relational data model [11]. It is well-known that the abstractions and the constructs provided by this kind of data models are influenced by implementation issues. It follows that the global schema represents a sort of unified data structure accommodating the various data at the sources, and the client, although freed from physical aspects of the source data (where they are, and how they can be accessed), is still exposed to issues concerning how data are packed into specific structures. Moreover, the need of better capturing the complex interrelationships in the domain of interest leads to consider also constraints expressed over the global schema as keys, foreign keys, and complex forms of assertions expressible in semantic data models, such as the Entity-Relationship model or UML class diagrams [24]. However, these constraints over the global schema have a deep impact on how answers are computed, and hence they must be fully taken into account during query answering [18, 19].

To overcome these problems, a new type of data integration systems is emerging, known as *Ontology-based Information Systems* (OIS). The basic idea behind these systems is to express the global schema as an ontology, i.e., a conceptual specification of the application domain. With this idea, the integrated view that the system provides to information consumers is not merely a data structure accommodating the various data at the sources, but a semantically rich description of the relevant concepts and relationships in the domain of interest, with the mapping acting as the reconciling mechanism between the conceptual level and the data sources.

Ontologies are introduced as an "explicit specification of a conceptualization" [53], and are able to capture domain knowledge in a generic way and to provide a commonly agreed understanding of a domain, which may be reused

and shared across applications and groups [35]. For this reason they are widely used as a means for conceptually structuring domains of interest.

There are a number of formalisms to represent ontologies, but Description Logics (DLs) have been recognized as the best means for the formal specification of ontologies, for their ability of combining modeling power and decidability of reasoning [8]. For these characteristics, DLs constitute the logical underpinning of prominent ontology languages, such as OWL 2, the W3C standard language for ontology specification[1].

There are different ways of employing the ontology in an information system. In particular, we are interested in *Stand-alone Ontology-based Information Systems* (SOISs) and *Ontology-based Data Integration Systems* (ODISs).

In both SOISs and ODISs, all the knowledge assets administered by the system is represented by a DL knowledge base (KB).

In the former a materializing approach is adopted. More specifically, in a SOIS the KB is constituted by a *TBox*, which is a set of assertions representing intensional knowledge, and an *ABox*, representing extensional knowledge. One possible and often used technical solution for storing the assertions in the ABox is adopt a relational database [41, 81, 93]. This solution allows for considering the ABox as an accessible and modifiable set of extensional assertions.

The latter offers a more complex scenario. As for SOISs, a formal conceptualization of the domain is given by means of a TBox expressed in a suitable DL, while the extensional level of the system is represented by two components: the set of independent and heterogenous data sources to be accessed, which are commonly handled by Relational Database Management Systems, and the mapping, which is a set of *mapping* assertions that specify how data at the source correspond to instance of the elements in the TBox [95]. Hence, in realizing ODISs, the virtual approach is adopted.

By referring to the so-called *functional view of knowledge representation* [75], OISs should be able to perform two kinds of operations, called ASK, which are operations used to extract information from the knowledge base, and TELL, which are operations aiming at changing the knowledge base according to new knowledge acquired over the domain.

For providing ASK operations, in case of SOISs, the systems have to provide well known reasoning services, as instance checking and query answering [30, 31, 60, 87, 88]. In case of ODISs, these services became more complex, since one has to consider also the mappings during the process. Concerning this point, the notion of *Ontology-Based Data Access* (OBDA) [95, 101] has been recently proposed. In this setting, the query answering service amounts to process a query expressed in terms of the TBox alphabet, and to compute its answer on the basis of the knowledge specified by the TBox, the mapping, and

[1] http://www.w3.org/TR/owl2-overview/

the data which are stored in autonomous databases. Therefore, we use just "query answering" for referring to ASK operations in case of SOISs, while we use "OBDA" for indicating ASK operations in case of ODISs.

As said above, TELL operations are related to the need of changing the KB in order to reflect a change in the domain of interest the KB is supposed to represent. In other words, TELL operations should be able to cope with the *update of the KB*. While ASK operations over OISs have been investigated in detail by the scientific community, existing otology-based systems do not provide explicit services for KB evolution. Nevertheless, many recent papers demonstrate that the interest towards a well-defined approach to KB evolution is growing significantly [33, 40, 45, 78, 109]. Generally speaking, updating a KB means adding or deleting assertions from the KB with the aim of sanctioning that a set of properties are respectively true or non-true in the state resulting from the change.

In dealing with evolution in SOISs, one of the major challenges is related to inconsistency management, that is how to react to the case where the set of properties specified in an update is inconsistent with the current knowledge. In addition, in defining a change operator the need of keeping the distance between the original KB and the KB resulting from the application of the evolution operator minimal is commonly perceived. When we are dealing with ODISs, the update problem is augmented by the problem of how to modify the data source in order to reflect changes.

Commonly, the TBox used in an OIS is usually a high-quality representation of the domain, designed in such a way to avoid inconsistencies in the modeling of concepts and relationships, i.e., it is usually a consistent theory. On the contrary, the assertions in the ABox in general derive from various autonomous sources which are independent on the conceptualization represented by the TBox, and therefore may contain data which are not coherent with it. Also, as said earlier, an update of the KB reflects new information on individuals in the domain, and this new information may contradict what was known about the individuals in the current state. From the above considerations, we conclude that, in the described scenario, one have to deal with both ASK operations and TELL operations in case the KB handled by the system is inconsistent,

In conclusion, for providing a complete analysis of OISs one needs to face with the following issues:

	without inconsistency	**with inconsistency**
SOIS		
ASK:	*answering queries posed over a consistent KB*	*answering queries posed over an inconsistent KB*
TELL:	*updating a consistent KB*	*updating an inconsistent KB*

ODIS

ASK:	*OBDA over a consistent ODIS*	*OBDA over an inconsistent ODIS*
TELL:	*updating a consistent ODIS*	*updating an inconsistent ODIS*

In this thesis we address all the issues related to SOISs, and leave the case of ODISs to future investigations. Moreover, motivated by the fact that in real worlds domain the size of extensional information is generally very large, in this thesis we focus on the logics of the *DL-Lite* family [31, 95]. These logics present the distinguishing characteristic of enabling first-order (FOL) rewritability of query answering of unions of conjunctive queries (UCQs). This means that to answer a UCQ q in *DL-Lite* it is possible to first rewrite q into a first-order query q_r, only on the basis of the knowledge specified in the TBox, and then evaluate q_r over the ABox, which can be seen as a plain database. FOL-rewritability of UCQs is a notable property, since many practical applications require the expressivity of UCQs for query answering, and their FOL-rewritability allows for delegating the management of the ABox to a relational DBMS, because the FOL queries produced by the rewriting process are directly translatable into SQL. In other words, in this way one can reduce a form of reasoning with incomplete information, i.e., query answering over a KB, to classical evaluation of an SQL query. Notably, the ABox does not need to be touched during the rewriting phase, and no data preprocessing is needed (as for example required in [41, 64]). This turns out to be crucial, for instance, in all those applications in which KBs, and in particular their intensional component, are used to access data stored in external repositories, such as in OBDA.

1.1 Contributions of the thesis

In this thesis, we study the topic of the update of DL KB at the instance level and we address the topic of management of inconsistency in that cases where the TBox, alone, is consistent, while the ABox may contain information which contradicts assertions entailed by the TBox. In what follows, we briefly summarize the main contributions we provide.

(i) We present *DL-Lite*$_{A,id,den}$ a new logic of the *DL-Lite*$_A$ family [31, 95], and we show that reasoning tasks such as KB satisfiability and conjunctive query answering can be managed efficiently with respect to the size of the ABox. We support this result by providing algorithms for both checking satisfiability of KBs expressed in *DL-Lite*$_{A,id,den}$ and answering union of conjunctive queries posed over a *DL-Lite*$_{A,id,den}$-KB that run in AC^0 with respect to the size of the ABox. Moreover, we provide an algorithm

for detecting those assertions in the ABox which contradict assertions belonging to the TBox.

(ii) We study the problem of updating consistent DL knowledge bases at the instance level, i.e., we focus on the cases where the initial KB is consistent, and we enforce the condition that the KB resulting from the update of a KB has the same TBox as the original KB. We analyze both the cases in which the change operation aims at inserting new knowledge, and in which the change operation aims at deleting parts of knowledge. We present an excursus on the update approaches reported in literatures and we propose update semantics for both insertion and deletion. We show that the proposed semantics exhibit several desirable properties. Then, we analyze the problem in the context of *DL-Lite*$_{A,id,den}$ and we provide algorithms for both insertion and deletion that compute the result of the update in polynomial time with respect to the size of the ABox.

(iii) We discuss how to deal with inconsistencies in querying DL KBs, in particular by addressing the problem of *consistent query answering* [36]. We present four inconsistency-tolerant semantics and we analyze these by highlighting strengths and weaknesses of each semantics. Moreover, we show that two of these semantics are not tractable in *DL-Lite*$_{A,id,den}$, while in the other two query answering is FOL-rewritable. Hence, we provide sound and complete query rewriting techniques under such semantics.

(iv) Finally, we study the problem of updating inconsistent KBs. We propose a semantics suitable for this purpose, and we analyze its properties. Moreover, we provide algorithms for updating possibly inconsistent KBs expressed in *DL-Lite*$_{A,id,den}$ with both insertion and deletion, and we prove that they compute the result of the update in polynomial time with respect to the size of the ABox.

1.2 Organization of the thesis

The thesis is organized as follows. In Chapter 2 we present some theoretical background. In particular, we present Description Logics and some of the issues related to knowledge bases expressed in such logics. Moreover, we illustrate the query languages which we deal with in the thesis, and present some basic notions needed for our dissertation. In Chapter 3 we conclude the preliminary part of the thesis by presenting the *DL-Lite* family of Description Logics [31], and the approaches based on such DLs to querying consistent KBs.

In Part II we formally present a new logic of the *DL-Lite*-family called *DL-Lite*$_{A,id,den}$. In Chapter 4 present syntax and semantics of such logic. In

Chapter 5, we provide algorithms for KB satisfiability and query answering over KB expressed in this language. We show that these algorithms run in AC^0 with respect to the size of the ABox.

In Part III, we study the problem of updating DL KBs at the instance level. In Chapter 6, we propose a semantics from updating consistent KB and we survey the most important update and revision approaches proposed in literature. In Chapter 7, we provide algorithms for computing the result of updating KBs expressed in $DL\text{-}Lite_{A,id,den}$ coherently with our update semantics, and we show that such algorithms run in polynomial time with respect to the size of the ABox. Part of the work presented in these chapters is published in [73].

In Part IV, we address the problem of consistent query answering. In Chapter 8 we provide four inconsistency-tolerant semantics, and we study the tractability of this semantics and the properties which characterize them. In Chapter 9 we give a complete complexity characterization of consistent query answering in $DL\text{-}Lite_{A,id,den}$ under the semantics presented in Chapter 9. In this chapter we also provide effective algorithms for query answering by rewriting under two of our inconsistency-tolerant semantics in $DL\text{-}Lite_{A,id,den}$. A preliminary version of this material appeared in [66, 68, 69].

In Part V, we study the problem of updating inconsistent DL KBs. In Chapter 10 we propose our inconsistency-tolerant update semantics, and we discuss some properties of such a semantics. In Chapter 11 we present algorithms for computing the result of updating possibly inconsistent $DL\text{-}Lite_{A,id,den}$-KBs under our semantics. We show that these algorithms are correct and that they run in polynomial time with respect to the size of the ABox. The work presented in the above chapters is based on [74].

Chapter 2

Formal framework for Description Logic Knowledge Bases

In this chapter, we introduce a formal framework description logic knowledge bases (KBs) underlying our investigation on evolution and inconsistency management.

We start by giving some preliminary notions of Computational Complexity which are useful for the rest of this work. Then, we present Description Logics and some of the issues related to knowledge bases expressed in these logics.

2.1 Preliminaries on computational complexity

We assume that the reader is familiar with basic notions about computational complexity, as defined in standard textbooks [50, 65, 89]. In this work, we will refer to the following complexity classes:

$$\text{AC}^0 \subsetneq \textsc{LogSpace} \subseteq \textsc{NLogSpace} \subseteq \textsc{PTime} \subseteq \text{NP} \subseteq \textsc{ExpTime}.$$

We have depicted the known relationships between these complexity classes. In particular, it is known that AC^0 is strictly contained in LogSpace, while it is open whether any of the other depicted inclusions is strict. However, it is known that PTime $\subsetneq$ ExpTime. Also, we will refer to the complexity class coNP, which is the class of problems that are the complement of a problem in NP.

In what follows, we only comment briefly on the complexity classes AC^0, LogSpace, and NLogSpace.

A decision problem belongs to LogSpace if it can be decided by a two-tape deterministic Turing machine that receives its input on the read-only

input tape and uses a number of cells of the read/write work tape that is at most logarithmic in the length of the input. The complexity class NLogSpace is defined analogously, except that a non-deterministic Turing machine is used instead of a deterministic one. A typical problem that is in LogSpace (but not in AC^0) is undirected graph reachability [100]. A typical problem that is in NLogSpace is directed graph reachability.

For the complexity class AC^0, we provide here only the basic intuitions, and refer to [108] for the formal definition, which is based on the circuit model. Intuitively, a problem belongs to AC^0 if it can be decided in constant time using a number of processors that is polynomial in the size of the input. A typical example of a problem that belongs to AC^0 is the evaluation of First-Order Logic (i.e., SQL) queries over relational databases, where only the database is considered to be the input, and the query is considered to be fixed [1]. This fact is of importance in the context of what discussed in this work, since the low complexity in the size of the data of the query evaluation problem provides an intuitive justification for the ability of relational database engines to deal efficiently with very large amounts of data. Also, whenever a problem is shown to be hard for a complexity class that *strictly* contains AC^0 (such as LogSpace and all classes above it), then it cannot be reduced to the evaluation of First-Order Logic queries.

2.2 Description Logic KBs

Description Logics (DLs) [7] are a family of knowledge representation languages that can be used to represent the knowledge of a domain of interest in a structured and formally well-understood way.

Description Logics represent the domain of interest in terms of *objects*, i.e., individuals, *concepts*, which are abstractions for sets of objects, and *roles*, which denote binary relations between objects. In addition, some DLs distinguish concepts from *value-domains*, which denote sets of values, and role from *attribute*, which denote binary relations between object and values.

In this section, we only illustrate some typical constructors commonly allowed by DLs.

2.2.1 DL Expressions

We consider an alphabet comprising symbols for atomic concept, value-domains, atomic roles, attributes, and constants. We denote with Γ the alphabet of constants, which we assume partitioned into Γ_P, containing symbols for predicates, i.e., atomic concepts, atomic roles, attributes and value-domains, and Γ_C, containing symbols for individual (object and value) constants. Since the

DLs that we consider in this work distinguish between object and value constants, we partition the set Γ_C into two disjoint sets Γ_O, which is the set of constants denoting objects, and Γ_V, which is the set of constants denoting values. The value types that we consider in this work are those corresponding to the data types adopted by the Resource Description Framework (RDF)[1], such as `xsd:string`, `xsd:integer`, etc. Thus value type represents sets of values that are pairwise disjoint. In the rest of this work, we denote such predefined valued-domains by $T_1, \ldots, T_n$.

Complex concept expressions are constructed starting from atomic concepts by applying suitable operators. Analogously, for complex role and complex attribute expressions. Different DLs allow for different operators in the constructs.

We introduce now syntax and semantics of concept, role, and attribute expressions. The syntax of DL expressions is defined by the the following rules:

$$
\begin{array}{lcl}
B & \longrightarrow & A \mid \exists Q \mid \delta(U) \\
C & \longrightarrow & \top_C \mid B \mid \neg C \mid C \sqcap \cdots \sqcap C \mid C \sqcup \cdots \sqcup C \mid \\
 & & \exists Q.C \mid \forall Q.C \mid \delta_F(U) \mid \{o\} \\
E & \longrightarrow & \rho(U) \\
F & \longrightarrow & \top_D \mid T_1 \mid \cdots \mid T_n \\
Q & \longrightarrow & P \mid P^- \\
R & \longrightarrow & Q \mid \neg Q \\
V & \longrightarrow & U \mid \neg U
\end{array}
$$

where A denotes an atomic concept, P an atomic role, U an atomic attribute (or simply attribute), and $\top_C$ the *universal concept*. An atomic concept, (resp., atomic role, or an atomic attribute) is a concept (resp., role or attribute) denoted by a name.

B and Q denote respectively basic concepts and basic roles. C denotes an arbitrary (i.e., either atomic, basic or complex) concept, R an arbitrary role, and V an arbitrary attribute. E denotes a *value-domain*, i.e., the range of an attribute, F a *value-domain expression*, $T_1, \ldots, T_n$ are unbounded pairwise disjoint predefined value-domains, and $\top_D$ the *universal value-domain*. $\neg C$ denotes the *negation* of an arbitrary concept C. The concept $\exists Q$, also called *unqualified existential restriction*, denotes the *domain* of a role Q, i.e., the set of objects that Q relates to some object. Similarly, $\delta(U)$ denotes the *domain* of an attribute U, i.e., the set of objects that U relates to some value. The concept $\exists Q.C$, also called *qualified existential restriction*, denotes the *qualified domain* of Q w.r.t. C, i.e., the set of objects that Q relates to some instance of C. Similarly, $\delta_F(U)$ denotes the *qualified domain* of U w.r.t. a value-domain F,

[1] http://www.w3.org/RDF/

$$
\begin{array}{rcl}
A^I & \subseteq & \Delta_O^I \\
(\exists Q)^I & = & \{\ o \mid \exists o'.\,(o,o') \in Q^I\ \} \\
(\delta(U))^I & = & \{\ o \mid \exists v.\,(o,v) \in U^I\ \} \\
(\exists Q.C)^I & = & \{\ o \mid \exists o'.\,(o,o') \in Q^I \wedge o' \in C^I\ \} \\
(\delta_F(U))^I & = & \{\ o \mid \exists v.\,(o,v) \in U^I \wedge v \in F^I\ \} \\
(\forall Q.C)^I & = & \{\ o \mid \forall o'.\,(o,o') \in Q^I \rightarrow o' \in C^I\} \\
(\rho(U))^I & = & \{\ v \mid \exists o.\,(o,v) \in U^I\ \} \\
P^I & \subseteq & \Delta_O^I \times \Delta_O^I \\
(P^-)^I & = & \{\ (o,o') \mid (o',o) \in P^I\ \} \\
\{o\} & = & \{o\}^{\mathcal{I}} \subseteq \Delta_O^I,\ \sharp\{o\}^{\mathcal{I}} = 1
\end{array}
\qquad
\begin{array}{rcl}
\top_C^I & = & \Delta_O^I \\
(\neg B)^I & = & \Delta_O^I \setminus B^I \\
\top_D^I & = & \Delta_V^I \\
T_i^I & = & val(T_i) \\
(\neg Q)^I & = & (\Delta_O^I \times \Delta_O^I) \setminus Q^I \\
U^I & \subseteq & \Delta_O^I \times \Delta_V^I \\
(\neg U)^I & = & (\Delta_O^I \times \Delta_V^I) \setminus U^I \\
(C_1 \sqcap \cdots \sqcap C_n)^I & = & C_1^I \cap \cdots \cap C_n^I \\
(C_1 \sqcup \cdots \sqcup C_n)^I & = & C_1^I \cup \cdots \cap C_n^I
\end{array}
$$

Figure 2.1: Semantics of the DL constructs considered in this work

i.e., the set of objects that U relates to some value in F. $\{o\}$ denotes *nominals*, i.e., concepts $\{o\}$ representing a singleton set consisting of the individual o. $\rho(U)$ denotes the *range* of an attribute U, i.e., the set of values to which U relates some object. Note that the range $\rho(U)$ of U is a value-domain, whereas the domain $\delta(U)$ of U is a concept. P^- denotes the *inverse* of an atomic role, and $\neg Q$ denotes the *negation* of a basic role. In the following, when Q is a basic role, the expression Q^- stands for P^- when $Q = P$, and for P when $Q = P^-$. Finally, $\neg U$ denotes the *negation* of an atomic attribute. All these symbols will be used with subscripts, when needed.

The semantics of a DL KB is given in terms of First-Order Logic (FOL) interpretations. All such interpretations agree on the semantics assigned to each predefined value-domain T_i and to each constant in Γ_V. In particular, each predefined valued-domain T_i is interpreted as the set $val(T_i)$ of values of the corresponding RDF data type, and each constant $c_i \in \Gamma_V$ is interpreted as one specific value, denoted $val(c_i)$, in $val(T_i)$. Note that, the data types $T_1, \ldots, T_n$ are pairwise disjoint, i.e., $val(T_i) \cap val(T_j) = \emptyset$, for $i \neq j$.

Based on the above observations, we can now define the notion of interpretation. An *interpretation* $\mathcal{I} = (\Delta^I, \cdot^I)$ consists of a nonempty interpretation domain Δ^I, which is the disjoint union of two non-empty sets: Δ_O^I, called the *domain of objects*, and Δ_V^I, called the *domain of values*, and an *interpretation function*, i.e., a function that assigns an element of Δ^I to each constant in Γ, a subset of Δ^I to each concept and value-domain, and a subset of $\Delta^I \times \Delta^I$ to each role and attribute, in such a way that the following holds:

- for each $v \in \Gamma_V$, $v^I = val(v)$;
- for each $o \in \Gamma_O$, $o^I \in \Delta_O^I$.

All the DLs discussed in this work follow the *unique name assumption* (UNA), and, therefore, we assume that, for each $a_1, a_2 \in \Gamma$, $a_1 \neq a_2$ implies $a_1^I \neq a_2^I$.

The semantics of all the constructs that are relevant for this work is shown in Figure 2.1, illustrating the rules for the interpretation function $\cdot^{\mathcal{I}}$.

2.2.2 DL Knowledge Bases

Given a DL language $\mathcal{L}$, an $\mathcal{L}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ over the alphabet Γ is a pair formed by a set $\mathcal{T}$ of intensional assertions, i.e., axioms specifying general properties of concepts, roles, and relations, expressed in $\mathcal{L}$, called *TBox*, and a set $\mathcal{A}$ of extensional assertions, i.e., axioms about individual objects, over Γ expressed in $\mathcal{L}$, called *ABox*. In this work we consider only the case where the ABox is constituted by extensional assertions built over atomic predicates, and therefore we omit to refer to $\mathcal{L}$ when we talk about ABox assertions.

Definition 1. A DL *knowledge base* (KB) over the alphabet Γ is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

- $\mathcal{T}$, called the TBox of $\mathcal{K}$, is a finite set of intensional assertions over Γ;
- $\mathcal{A}$, called the ABox of $\mathcal{K}$, is a finite set of extensional assertions over Γ of the form:

  $$\begin{array}{ll} A(a_1) & (\textit{concept membership assertion}); \\ P(a_1, a_2) & (\textit{role membership assertion}); \\ U(a_1, v) & (\textit{attribute membership assertion}); \end{array}$$

  where A, P, and U are symbols in Γ_P denoting respectively an atomic concept symbol, an atomic role symbol, and an atomic attribute symbol, a_1, a_2 are constants in Γ_C denoting objects, and v is a constant in Γ_V denoting a value.

Informally, a concept membership assertion specifies that an object is an instance of an atomic concept. Analogously, the other types of membership assertions specify instances of atomic roles and attributes.

We now specify formally the form of a TBox. We consider the following four kinds of TBox assertions:

- *Inclusion assertions between concepts*, stating that all instances of one concept are also instances of another concept. Analogous assertions specify inclusions between roles, and inclusions between attributes.
- *Inclusion assertions between value domains*, stating that all the values in one value-domain are also in another value-domain.
- *Functional assertions*, stating that a role or an attribute is functional.
- *Identification assertions*, stating that a set of properties identifies instances of concepts.

An *inclusion assertion* has one the forms

$$\begin{array}{ll} Cl \sqsubseteq Cr & (\textit{concept inclusion}); \\ Ql \sqsubseteq Qr & (\textit{role inclusion}); \\ Ul \sqsubseteq Ur & (\textit{attribute inclusion}); \\ E \sqsubseteq F & (\textit{value-domain inclusion}). \end{array}$$

Note that, in the concept inclusion assertions, Cl (resp., Cr) denotes a concept used in the left-hand side (resp., right-hand side) of the inclusion. The distinction between Cl and Cr is motivated by the fact that the constraints that the various DLs put on the form of concept expressions appearing in one side of the inclusion are often different with respect to those in the other side. Analogous observation holds for both role and relation inclusions assertions. Intuitively, an inclusion assertion states that, in every model of $\mathcal{T}$, each instance of the left-hand side expression is also an instance of the right-hand side expression.

A *functionality assertion* has one of the forms

$$\begin{array}{ll} (\mathsf{funct}\ Q) & (\textit{role functionality}); \\ (\mathsf{funct}\ U) & (\textit{attribute functionality}). \end{array}$$

Intuitively, a functionality assertion states that the binary relation represented by a role (respectively, an attribute) is a function.

An *identification assertion* makes use of the notion of path. A *path* is an expression built according to the following syntax,

$$\pi \ \longrightarrow \ S \ \mid \ D? \ \mid \ \pi \circ \pi,$$

where S denotes a basic role (i.e., an atomic role or the inverse of an atomic role), an atomic attribute, or the inverse of an atomic attribute, and $\pi_1 \circ \pi_2$ denotes the composition of the paths π_1 and π_2. Finally, D denotes an arbitrary concept or a value domain, and the expression $D?$ is called a *test relation*, which represents the identity relation on instances of D. Test relations are used in all those cases in which one wants to impose that a path involves instances of a certain concept.

A path π denotes a complex property for the instances of concepts: given an object o, every object that is reachable from o by means of π is called a *π-filler* for o. Note that for a certain o there may be several distinct π-fillers, or no π-fillers at all.

If π is a path, the length of π, denoted $\mathit{length}(\pi)$, is 0 if π has the form $D?$, is 1 if π has the form S, and is $\mathit{length}(\pi_1) + \mathit{length}(\pi_2)$ if π has the form $\pi_1 \circ \pi_2$. With the notion of path in place, we are ready for the definition of identification assertion, which is an assertion of the form

$$(\mathit{id}\ C\ \pi_1, \ldots, \pi_n),$$

where C is an arbitrary concept, $n \geq 1$, and $\pi_1, \ldots, \pi_n$ (called the *components* of the identifier) are paths such that $\mathit{length}(\pi_i) \geq 1$ for all $i \in \{1, \ldots, n\}$. Intuitively, such a constraint asserts that for any two different instances o, o' of C, there is at least one π_i such that o and o' differ in the set of their π_i-fillers. The identification assertion is called *local* if $\mathit{length}(\pi_i) = 1$ for at least one $i \in \{1, \ldots, n\}$. The term "local" emphasizes that at least one of the paths has length 1 and thus refers to a local property of B.

We specify the semantics of various TBox assertions mentioned above by defining when an interpretation $\mathcal{I}$ *satisfies* a TBox assertion α, denoted $\mathcal{I} \models \alpha$..

- An interpretation $\mathcal{I}$ satisfies an inclusion assertion

$$\begin{array}{rcll} Cl & \sqsubseteq & Cr, & \text{if} \quad Cl^I \subseteq Cr^I; \\ Ql & \sqsubseteq & Qr, & \text{if} \quad Ql^I \subseteq Qr^I; \\ Ul & \sqsubseteq & Ur, & \text{if} \quad Ul^I \subseteq Ur^I; \\ E & \sqsubseteq & F, & \text{if} \quad E^I \subseteq F^I; \end{array}$$

- An interpretation $\mathcal{I}$ satisfies a role functionality assertion (funct Q), if for each $o_1, o_2, o_3 \in \Delta_O^I$

$$(o_1, o_2) \in Q^I \text{ and } (o_1, o_3) \in Q^I \quad \text{implies} \quad o_2 = o_3.$$

- An interpretation $\mathcal{I}$ satisfies an attribute functionality assertion (funct U), if for each $o \in \Delta_O^I$ and $v_1, v_2 \in \Delta_V^I$

$$(o, v_1) \in U^I \text{ and } (o, v_2) \in U^I \quad \text{implies} \quad v_1 = v_2.$$

- In order to define the semantics of identification assertions, we first define the semantics of paths. The extension π^I of a path π in an interpretation $\mathcal{I}$ is defined as follows:
  - if $\pi = S$, then $\pi^I = S^I$,
  - if $\pi = D?$, then $\pi^I = \{(o, o) \mid o \in D^I\}$,
  - if $\pi = \pi_1 \circ \pi_2$, then $\pi^I = \pi_1^I \circ \pi_2^I$, where $\circ$ denotes the composition operator on relations.

  As a notation, we write $\pi^I(o)$ to denote the set of π-fillers for o in $\mathcal{I}$, i.e., $\pi^I(o) = \{o' \mid (o, o') \in \pi^I\}$. Then, an interpretation $\mathcal{I}$ satisfies an identification assertion $(id\ C\ \pi_1, \ldots, \pi_n)$ if for all $o, o' \in C^I$, $\pi_1^I(o) \cap \pi_1^I(o') \neq \emptyset \wedge \cdots \wedge \pi_n^I(o) \cap \pi_n^I(o') \neq \emptyset$ implies $o = o'$.

We are now ready to complete the definition of the semantics of DL KBs. For this purpose, the basic definitions are as follows:

- An interpretation $\mathcal{I}$ is a model of a TBox assertion α if $\mathcal{I}$ satisfies α, according to what is reported above.
- An interpretation $\mathcal{I}$ is a model of (or equivalently *satisfies*) a membership assertion

$$\begin{array}{lll} A(a), & \text{if} & a^I \in A^I; \\ P(a_1, a_2), & \text{if} & (a_1^I, a_2^I) \in P^I; \\ U(a, c), & \text{if} & (a^I, c^I) \in U^I. \end{array}$$

Let $\mathcal{I}$ be an interpretation. We use $\mathcal{I} \models \alpha$ to denote that $\mathcal{I}$ is a model for the assertion α. A *model of a KB* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is an interpretation $\mathcal{I}$ that is a model of all assertions in $\mathcal{T}$ and $\mathcal{A}$. A KB is *satisfiable* if it has at least one model, *unsatisfiable* otherwise. For a KB, we also use the term *consistent* (resp. *inconsistent*) to mean satisfiable (resp. unsatisfiable). We use $Mod(\mathcal{K})$ to denote the set of all models of $\mathcal{K}$. A KB $\mathcal{K}$ *logically implies* (an assertion) α, written $\mathcal{K} \models \alpha$, if all models of $\mathcal{K}$ are also models of α. Let $\mathcal{A}'$ be a set of ABox assertions, we say that $\mathcal{A}'$ is $\mathcal{T}$-consistent if the KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ is consistent, $\mathcal{T}$-inconsistent otherwise.

The *deductive closure of a TBox* $\mathcal{T}$ in a language $\mathcal{L}$, denoted $\mathsf{cl}(\mathcal{T})$, is the set of all TBox assertions α in $\mathcal{L}$, such that $\mathcal{T} \models \alpha$. For a consistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, the closure of $\mathcal{A}$ with respect to $\mathcal{T}$, denoted $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$, is the set of all ABox assertions β that are formed with individuals in $\mathcal{A}$, and are logically implied by $\langle \mathcal{T}, \mathcal{A} \rangle$. We say that a TBox $\mathcal{T}$ is *closed* if $\mathcal{T} = \mathsf{cl}(\mathcal{T})$, analogously, we say that an ABox $\mathcal{A}$ is *closed with respect to a TBox* $\mathcal{T}$ if $\mathcal{A} = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$.

Given an alphabet Γ, we denote with *HB*(Γ) the *Herbrand Base of* Γ, i.e., the set of atomic ABox assertions (ground atoms) that can be built over the signature Γ.

Before ending this section, we illustrate some notable properties of the closure of an ABox with respect to an $\mathcal{L}$-TBox. Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{L}$-KB, $\mathcal{A}'$ be an ABox, and β be an ABox assertion. Then:

- $\mathcal{A} \subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$;
- if $\mathcal{A} \subseteq \mathcal{A}'$ then $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$;
- $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) = \mathsf{cl}_{\mathcal{T}}(\mathsf{cl}_{\mathcal{T}}(\mathcal{A}))$;
- if $\langle \mathcal{T}, \mathcal{A} \rangle \models \beta$ then $\beta \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$;
- $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle$ are two logically equivalent $\mathcal{L}$-KBs, i.e., $Mod(\langle \mathcal{T}, \mathcal{A} \rangle) = Mod(\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle)$.

All the above properties are easy to prove.

Finally, we end the section by highlighting an important property of the DLs considered in this work. We recall that a logic is called monotonic if the

addition of new axioms to a theory based on such logic never leads to the loss of any theorem proved in this theory. By looking at the definition of the DL introduced in this section, and by observing that they are all subset of the First Order Logic with equalities, we conclude that they are all monotonic. This property will be used extensively in the subsequent chapters of this thesis.

2.3 Queries over DL KBs

We are interested in queries over DL KBs. Similarly to the case of relational databases, the basic query class that we consider is the class of unions of conjunctive queries (UCQ), which is a subclass of the class of FOL-queries.

2.3.1 Syntax of queries

A FOL-*query* q over a DL KB $\mathcal{K}$ (resp., TBox $\mathcal{T}$) is a, possibly open, FOL formula $\varphi(\vec{x})$ whose predicate symbols are atomic concepts, roles, or attributes of $\mathcal{K}$ (resp., $\mathcal{T}$). The free variables of $\varphi(\vec{x})$ are those appearing in $\vec{x}$, which is a tuple of (pairwise distinct) variables. In other words, the atoms of $\varphi(\vec{x})$ have the form $A(x)$, $P(x, y)$, $U(x, y)$, or $x = y$, where A is an atomic concept, P is an atomic role, and U is an attribute in $\mathcal{K}$, and x, y are either variables in $\vec{x}$ or constants in Γ. The *arity* of q is the arity of $\vec{x}$. A query of arity 0 is called a *boolean query*. When we want to make the arity of a query q explicit, we denote the query as $q(\vec{x})$.

A *conjunctive query* (CQ) $q(\vec{x})$ over a DL KB is a FOL query of the form

$$\exists \vec{y}.\ conj(\vec{x}, \vec{y}),$$

where $\vec{y}$ is a tuple of pairwise distinct variables not occurring among the free variables $\vec{x}$, and where $conj(\vec{x}, \vec{y})$ is a *conjunction* of atoms. The variables $\vec{x}$ are also called *distinguished* and the (existentially quantified) variables $\vec{y}$ are called *non-distinguished*.

A *union of conjunctive queries* (UCQ) is a FOL query that is the disjunction of a set of CQs of the same arity, i.e., it is a FOL formula of the form:

$$\exists \vec{y}_1.\ conj_1(\vec{x}, \vec{y}_1) \vee \cdots \vee \exists \vec{y}_n.\ conj_n(\vec{x}, \vec{y}_n).$$

We will often refer to a UCQs as a set of CQs.

2.3.2 Semantics of queries in one interpretation

Given an interpretation $\mathcal{I} = (\Delta^I, \cdot^I)$, the FOL query $q = \varphi(\vec{x})$ is interpreted in $\mathcal{I}$ as the set q^I of tuples $\vec{o} \in \Delta^I \times \cdots \times \Delta^I$ such that the formula φ evaluates

to *true* in $\mathcal{I}$ under the assignment that assigns each object in $\vec{o}$ to the corresponding variable in $\vec{x}$ [1]. We call q^I the *answer* to q over $\mathcal{I}$. Notice that the answer to a boolean query is either the empty tuple, "()", considered as *true*, or the empty set, considered as *false*. We will use to denote that a boolean query evaluates to *true* over an interpretation $\mathcal{I}$ with $\mathcal{I} \models q$.

We remark that a relational database (over the atomic concepts, roles, and attributes) corresponds to a finite interpretation. Hence the notion of answer to a query introduced here is the standard notion of answer to a query evaluated over a relational database.

Since in general a DL KB has many models, and we cannot single out a unique interpretation (or database) over which to answer the query, the notion of answer to a query introduced above is not sufficient to capture the situation where a query is posed over a KB. Instead, the KB determines a set of interpretations, i.e., the set of its models, which intuitively can be considered as the set of databases that are "compatible" with the information specified in the KB. Given a query, we are interested in those answers to this query that depend only on the information in the KB, i.e., that are obtained by evaluating the query over a database compatible with the KB, but independently of which is the actually chosen database. In other words, we are interested in those answers to the query that are obtained for *all* possible databases (including infinite ones) that are models of the KB. This corresponds to the fact that the KB conveys only incomplete information about the domain of interest, and we want to guarantee that the answers to a query that we obtain are *certain*, independently of how we complete this incomplete information. This leads us to the following definition of *certain answers* to a query over a KB.

Definition 2. Let $\mathcal{K}$ be a DL KB and q a UCQ over $\mathcal{K}$. A tuple $\vec{c}$ of constants appearing in $\mathcal{K}$ is a *certain answer* to q over $\mathcal{K}$, written $\vec{c} \in \mathit{cert}(q, \mathcal{K})$, if for every model $\mathcal{I}$ of $\mathcal{K}$, we have that $\vec{c}^{\,I} \in q^I$.

In the case when q is a boolean query, then we say that q evaluates to *true* over $\mathcal{K}$, written $\mathcal{K} \models q$, if q evaluates to *true* over every model of $\mathcal{K}$.

Notice that, in the case where $\mathcal{K}$ is an inconsistent KB, the set of certain answers to a (U)CQ q is the finite set of all possible tuples of constants whose arity is the one of q. We denote such a set by $\mathit{AllTup}(q, \mathcal{K})$.

2.4 Reasoning over KBs

In studying DL KBs, we are interested in several reasoning services, including the traditional DL reasoning services. Specifically, we consider the following problems for DL KBs:

- *KB satisfiability*, i.e., given a KB $\mathcal{L}$, verify whether $\mathcal{K}$ admits at least one model.
- *Query answering*, i.e., given a KB $\mathcal{K}$ and a query q (either a CQ or a UCQ) over $\mathcal{K}$, compute the set $cert(q, \mathcal{K})$.

The following decision problem, called *recognition problem*, is associated to the query answering problem: given a KB $\mathcal{K}$, a query q (either a CQ or a UCQ), and a tuple of constants $\vec{a}$ of $\mathcal{K}$, check whether $\vec{a} \in cert(q, \mathcal{K})$. When we talk about the computational complexity of query answering, in fact we implicitly refer to the associated recognition problem.

In analyzing the computational complexity of a reasoning problem over a DL KB, we distinguish between data complexity and combined complexity [107]: *data complexity* is the complexity measured with respect to the size of the ABox only, while *combined complexity* is the complexity measured with respect to the size of all inputs to the problem, i.e., the TBox, the ABox, and the query. The data complexity measure is of interest in all those cases where the size of the intensional level of the KB (i.e., the TBox) is negligible with respect to the size of the data (i.e., the ABox), as in ontology-based data access systems [27, 95].

2.5 The interpretation $DB(\mathcal{A})$

Given a set of ABox assertions $\mathcal{A}$, we denote with $DB(\mathcal{A})$, a specific interpretation, or relational structure, associated to $\mathcal{A}$. More precisely,

Definition 3. Let $\mathcal{A}$ be a set of ABox assertions. The interpretation $DB(\mathcal{A}) = \langle \Delta^{DB(\mathcal{A})}, \cdot^{DB(\mathcal{A})} \rangle$ is the interpretation defined as follows:

- $\Delta^{DB(\mathcal{A})}$ is the non-empty set consisting of the union of the set of all object constants occurring in $\mathcal{A}$ and the set $\{val(c) \mid c$ is a value constant that occurs in $\mathcal{A}\}$,
- $a^{DB(\mathcal{A})} = a$, for each object constant a,
- $A^{DB(\mathcal{A})} = \{a \mid A(a) \in \mathcal{A}\}$, for each atomic concept A,
- $P^{DB(\mathcal{A})} = \{(a_1, a_2) \mid P(a_1, a_2) \in \mathcal{A}\}$, for each atomic role P, and
- $U^{DB(\mathcal{A})} = \{(a, val(c)) \mid U(a, c) \in \mathcal{A}\}$, for each atomic attribute U.

We now introduce the notion of *witness* for a tuple of constants with respect to a conjunctive query. For a query $q'(\vec{x}) = \exists \vec{y}.\, conj(\vec{x}, \vec{y})$, we denote with $conj\text{-}set(\vec{x}, \vec{y})$ the set of atoms corresponding to the conjunction $conj(\vec{x}, \vec{y})$.

Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a query $q'(\vec{x}) = \exists \vec{y}.\, conj(\vec{x}, \vec{y})$ over $\mathcal{K}$, and a tuple $\vec{t}$ of constants occurring in $\mathcal{A}$, a set of membership assertions $\mathcal{G}$ is a *witness for* $\vec{t}$ *w.r.t.* q' if there exists a substitution σ from the variables $\vec{y}$ in $conj\text{-}set(\vec{t}, \vec{y})$ to constants in $\mathcal{G}$ such that the set of assertions $\sigma(conj\text{-}set(\vec{t}, \vec{y}))$ is equal to $\mathcal{G}$. In particular, we are interested in witnesses for a tuple $\vec{t}$ w.r.t. a CQ q' that are contained in $\mathcal{A}$. Intuitively, each such witness corresponds to a subset of $\mathcal{A}$ that is sufficient in order to conclude that the formula $\exists \vec{y}.\, conj(\vec{t}, \vec{y})$ evaluates to *true* in the interpretation $DB(\mathcal{A})$, and therefore the tuple $\vec{t} = \vec{t}^{DB(\mathcal{A})}$ belongs to $q'^{DB(\mathcal{A})}$. More precisely, we have that $\vec{t} \in q'^{DB(\mathcal{A})}$ if and only if there exists a witness $\mathcal{G}$ of $\vec{t}$ w.r.t. q' such that $\mathcal{G} \subseteq \mathcal{A}$. The cardinality of a witness $\mathcal{G}$, denoted by $|\mathcal{G}|$, is the number of membership assertions in $\mathcal{G}$.

With this notion in place we give the following definition of *image of a query over an ABox* $\mathcal{A}$.

Definition 4. Let $\mathcal{A}$ be an ABox and let q be a conjunctive queries. We say that a set of ABox assertion $\mathcal{G} \subseteq \mathcal{A}$ is an *image of* $q(\vec{x})$ *in* $\mathcal{A}$, if there exists a tuple $\vec{t} \in q^{DB(\mathcal{A})}$ such that $\mathcal{G}$ is a witness of $\vec{t}$ with respect to q.

Given an ABox $\mathcal{A}$ and a query q, $\mathsf{images}(q, \mathcal{A})$, denotes the set of images of q in $\mathcal{A}$.

If q is a boolean query, we have that a set $\mathcal{G} \subseteq \mathcal{A}$ is an image of q in $\mathcal{A}$, if there exists a substitution σ from the variables in q to constants in $\mathcal{G}$ such that the set of atoms in $\sigma(q)$ is equal to $\mathcal{G}$ and the formula $\sigma(q)$ evaluates to *true* in the interpretation $DB(\mathcal{A})$. In case $\mathsf{images}(q, \mathcal{A}) \neq \emptyset$, we say that the boolean query q is satisfied by $\mathcal{A}$ (denoted by $\langle \emptyset, \mathcal{A} \rangle \models q$).

Example 1. Let $\mathcal{A}$ be the ABox constituted by the following ABox assertions:

$$\{\ P(a,a), \quad P(a,b), \quad S(a,c), \quad C(b)\ \}.$$

Now, let $q()$ be the following boolean query:

$$q() = \exists x, y, z.P(x,y) \wedge P(y,z).$$

The set $\mathsf{images}(q, \mathcal{A})$ containing the images of $q()$ in $\mathcal{A}$ is constituted by the following sets of ABox assertions:

$$\begin{array}{l} \mathcal{G}_1 = \{\ P(a,a), \quad P(a,b)\ \}; \\ \mathcal{G}_2 = \{\ P(a,a)\ \}; \end{array}$$

□

2.6 The Notion of FOL-rewritability

We now introduce the notion of FOL-rewritability for both satisfiability and query answering, which will be used in the sequel.

Intuitively, FOL-rewritability of satisfiability (resp., query answering) captures the property that we can reduce satisfiability checking (resp., query answering) to evaluating a FOL query over the ABox $\mathcal{A}$ considered as a relational database, i.e., over $DB(\mathcal{A})$. The definitions follow.

Definition 5. Satisfiability in a DL $\mathcal{L}$ is *FOL-rewritable*, if for every TBox $\mathcal{T}$ expressed in $\mathcal{L}$, one can effectively compute a boolean FOL query q_s, over the alphabet of $\mathcal{T}$, such that for every non-empty ABox $\mathcal{A}$, the KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable if and only if q_s evaluates to *false* in $DB(\mathcal{A})$.

Definition 6. Query answering in a DL $\mathcal{L}$ is *FOL-rewritable* if for every UCQ q and every TBox $\mathcal{T}$ expressed over $\mathcal{L}$, one can effectively compute a FOL query q_r over the alphabet of $\mathcal{T}$ such that for every non-empty ABox $\mathcal{A}$ and every tuple of constants $\vec{a}$ occurring in $\mathcal{A}$, we have that $\vec{a} \in \mathit{cert}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ if and only if $\vec{a}^{DB(\mathcal{A})} \in q_r^{DB(\mathcal{A})}$.

We remark that FOL-rewritability of a reasoning problem that involves the ABox of a KB (such as satisfiability or query answering) is tightly related to low data complexity of the problem. Indeed, since the FOL query considered in the above definitions depends only on the TBox (and the query), but not on the ABox, and since the evaluation of a First-Order Logic query (i.e., an SQL query without aggregation) over an ABox is in AC^0 in data complexity [1], FOL-rewritability of a problem has as an immediate consequence that the problem is in AC^0 in data complexity. Hence, one way of showing that for a certain DL $\mathcal{L}$ a problem is *not* FOL-rewritable, is to show that the data complexity of the problem for the DL $\mathcal{L}$ is above AC^0, e.g., LogSpace-hard, NLogSpace-hard, PTime-hard, or even coNP-hard.

Chapter 3

DL-Lite Knowledge Bases

In several areas, such as Enterprise Application Integration, Data Integration [71], and the Semantic Web [56], clients need to access a shared conceptualization of the intensional level of the application domain in terms to specify the access to services exported by the system. Ontologies are nowadays considered as the ideal formal tool to provide such a shared conceptualization. Indeed, one of their most interesting usages is *ontology-based data access*, where, on top of the usual data layer of an information system, a conceptual layer is superimposed, allowing the client to abstract away from how the information is actually maintained in the data layer.

While ontologies are the best candidates for realizing the conceptual layer, relational DBMSs are natural candidates for the management of the data layer. The combination of these two mechanisms for representing and maintaining information new paradigm of data management [72]. Indeed, it requires, on the one hand, dealing with the characteristics of both formalisms, and on the other hand, addressing properly their interaction in a combined system.

In this chapter we present a family of DLs, called *DL-Lite* family[1], that has been proposed recently [25, 28, 29, 31, 95] with the aim of addressing issues related to the ontology-based data management.

One distinguishing feature of the logics of the *DL-Lite* family is that they are tightly related to conceptual modeling formalisms and are actually able to capture their most important features [10].

A further distinguishing feature of such DLs is that query answering over a KB can be performed as a two step process: in the first step, a query posed over the KB is reformulated, taking into account the intensional component (the TBox) only, obtaining a union of conjunctive queries; in the second step such a union is directly evaluated over the extensional component of the KB (the ABox). Under the assumption that the ABox is maintained by an RDBMS

[1] Not to be confused with the set of DLs studied in [6], which form the *DL-Lite*$_{bool}$ family.

in secondary storage, the evaluation can be carried out by an SQL engine, taking advantage of well established query optimization strategies. Since the first step does not depend on the data, and the second step is the evaluation of a relational query over a databases, the whole query answering process is in AC^0 in the size of the data [1], i.e., it has the same complexity as the plain evaluation of a conjunctive query over a relational database.

The results reported in this chapter are presented in [25, 28, 31, 95].

3.1 The DL-Lite Family

In this section, we introduce the principal DLs belonging to the *DL-Lite* family. We start presenting *DL-Lite*$_{core}$. All the other members of the *DL-Lite* family extend *DL-Lite*$_{core}$ with some constructs. In particular, we are interested in *DL-Lite*$_A$ [95] which extends *DL-Lite*$_{core}$ with attributes, and allows for specifying functional restrictions on role and attributes and role and attribute inclusion assertions. Finally, we present *DL-Lite*$_{A,id}$ [25], a DL of the *DL-Lite* family, that is also equipped with identification constraints [32]. Other DLs that are member of the *DL-Lite* family are *DL-Lite*$_F$ and *DL-Lite*$_R$ [31]. We will show in what follows that in these DLs the trade-off between expressive power and computational complexity of reasoning is optimized towards the needs that arise in ontology-based data access.

Like in any other logic, *DL-Lite*$_A$ expressions are built over an alphabet. Here, we refer to an alphabet Γ as described in Section 2.2.

Before presenting *DL-Lite*-KBs formally, we need to introduce some preliminary notions. An atomic role P (resp., an atomic attribute U) is called an *identifying property in a DL TBox* $\mathcal{T}$, if

- $\mathcal{T}$ contains a functionality assertion (funct P) or (funct P^-) (resp., (funct U)), or
- P (resp., U) appears (in either direct or inverse direction) in some path of an identification assertion in $\mathcal{T}$.

We say that an atomic role P (resp., an atomic attribute U) *appears positively* in the right-hand side of an inclusion assertion α if α has the form $Q \sqsubseteq P$ or $Q \sqsubseteq P^-$, for some basic role Q (resp., $U' \sqsubseteq U$, for some atomic attribute U'). An atomic role P (resp., an atomic attribute U) is called *primitive in a TBox* $\mathcal{T}$, if it does not appear positively in the right-hand side of an inclusion assertion of $\mathcal{T}$.

As shown below, every DL of the *DL-Lite* family allows negative concepts, negative roles, and negative attribute to occur only on the right-hand side of inclusion assertions. This syntactic characteristic allows us to distinguish inclusion assertions in a *DL-Lite* TBox as follows. An inclusion assertion that does not contain the symbol '$\neg$' in its right-hand side is called a *positive in-*

clusion (PI), while an inclusion assertion that contains the symbol '$\neg$' in its right-hand side is called a *negative inclusion (NI)*.

3.1.1 DL-Lite$_{core}$

DL-Lite$_{core}$ is the core language for the whole family. In *DL-Lite*$_{core}$ the alphabet Γ comprises symbols only for atomic concepts, atomic roles, and constants denoting objects. In other terms, *DL-Lite*$_{core}$ does not distinguish objects from values, and, therefore, does not distinguish roles from attributes. In the follows we implicity refer to an alphabet Γ.

Definition 7. A *DL-Lite*$_{core}$-KB is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

- the TBox $\mathcal{T}$ is a finite set of intensional assertions over Γ_P partitioned into two sets $\mathcal{T}_{inc}$ and $\mathcal{T}_{disj}$, where:
  - $\mathcal{T}_{inc}$ is a finite set of positive inclusion assertions of the form

$$\begin{array}{ll} A \sqsubseteq A', & A \sqsubseteq \exists Q, \\ \exists Q \sqsubseteq A', & \exists Q \sqsubseteq \exists Q'; \end{array}$$

  - $\mathcal{T}_{disj}$ is a finite set of negative inclusion assertions of the form

$$\begin{array}{ll} A \sqsubseteq \neg A', & A \sqsubseteq \neg \exists Q, \\ \exists Q \sqsubseteq \neg A', & \exists Q \sqsubseteq \neg \exists Q'; \end{array}$$

  where A and A' denote *atomic concepts* and Q and Q' denote *basic roles*, i.e., roles that are either an atomic role or the inverse of an atomic role.

- the ABox $\mathcal{A}$ is a finite set of extensional assertions of the form:

$$A(a) \qquad P(a, b)$$

  were A and P denoting respectively an atomic concept symbol, and an atomic role symbol, and a and b are constants.

3.1.2 DL-Lite$_A$

We now present the DL of the *DL-Lite* family, called *DL-Lite*$_A$. Such a DL is novel with respect to *DL-Lite*$_{core}$, in that it takes the distinction between objects and values into account, and therefore distinguishes:

- *concepts from value-domains*: while a concept is abstraction for a set of objects, a value-domain, also known as concrete domain [80], denotes a set of (data) values,

- *attributes from roles*: while a role denotes a binary relation between objects, a (concept) attribute denotes a binary relation between objects and values.

Moreover, *DL-Lite*$_A$ extends *DL-Lite*$_{core}$ with the ability of specifying inclusion assertions between roles and attributes, and with the ability of specifying functionality on roles and attributes. As we will see below, in *DL-Lite*$_A$ suitable limitations in the combinations of the TBox assertions are imposed in order to keep the complexity of reasoning low.

Definition 8. A *DL-Lite*$_A$-KB over an alphabet Γ is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

- the TBox $\mathcal{T}$ is a finite set of intensional assertions over Γ_P partitioned into four sets $\mathcal{T}_{inc}$, $\mathcal{T}_{type}$, $\mathcal{T}_{disj}$, and $\mathcal{T}_{funct}$ where:
  - $\mathcal{T}_{inc}$ is a finite set of positive inclusion assertions of the form:

    $$\begin{array}{ll} B \sqsubseteq B' & (\textit{inclusions between concepts}), \\ Q \sqsubseteq Q' & (\textit{inclusions between roles}), \\ U \sqsubseteq U' & (\textit{inclusions between attributes}); \end{array}$$

  - $\mathcal{T}_{type}$ is a finite set of positive inclusion assertions of the form:

    $$E \sqsubseteq F \qquad (\textit{inclusions between value-domains});$$

  - $\mathcal{T}_{disj}$ is a finite set of negative inclusion assertions of the form:

    $$\begin{array}{ll} B \sqsubseteq \neg B' & (\textit{disjunctions between concepts}), \\ Q \sqsubseteq \neg Q' & (\textit{disjunctions between roles}), \\ U \sqsubseteq \neg U' & (\textit{disjunctions between attributes}); \end{array}$$

  - $\mathcal{T}_{funct}$ is a finite set of functionality assertions of the form:

    $$\begin{array}{ll} (\mathsf{funct}\ Q) & (\textit{role functionality}), \\ (\mathsf{funct}\ U) & (\textit{attribute functionality}); \end{array}$$

  where B and B' denote *basic concepts*, E denotes a basic value-domain, F a *value-domain expression*, Q and Q' *basic roles*, U and U' denote *attributes*;

- the ABox $\mathcal{A}$ is a finite set of extensional assertions over Γ of the form:

  $$A(a) \qquad P(a, b) \qquad U(a, v)$$

  were A, P, and U in Γ_P denoting respectively an atomic concept symbol, an atomic role symbol, and an attribute symbol, and a and b are constants in Γ_O and v is a constant in Γ_V;

- such that each identifying property in $\mathcal{T}$ is primitive in $\mathcal{T}$.

Intuitively, the condition imposed over the TBox assertions states that, in *DL-Lite*$_A$ TBoxes, roles and attributes occurring in functionality assertions cannot be specialized. This limitation ensures the tractability of reasoning in this logic [95].

3.1.3 DL-Lite$_{A,id}$

We now present *DL-Lite*$_{A,id}$ a specific DL of the *DL-Lite* family, which extend *DL-Lite*$_A$ with identification assertions [31].

Definition 9. A *DL-Lite*$_{A,id}$-KB over an alphabet Γ is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

- the TBox $\mathcal{T}$ is a finite set of intensional assertions over Γ_P partitioned into five sets $\mathcal{T}_{inc}$, $\mathcal{T}_{type}$, $\mathcal{T}_{disj}$, $\mathcal{T}_{funct}$, and $\mathcal{T}_{id}$ where:
  - $\mathcal{T}_{inc}$, $\mathcal{T}_{type}$, $\mathcal{T}_{disj}$, and $\mathcal{T}_{funct}$ are as in Definition 8;
  - $\mathcal{T}_{id}$ is a finite set of identification assertions of the form:

    $$(id\ B\ \pi_1, \ldots, \pi_n);$$

    where B denotes a *basic concept*;
- the ABox $\mathcal{A}$ is a finite set of extensional assertions over Γ of the form:

  $$A(a) \qquad P(a, b) \qquad U(a, v)$$

  were A, P, and U in Γ_P denoting respectively an atomic concept symbol, an atomic role symbol, and an attribute symbol, and a and b are constants in Γ_O and v is a constant in Γ_V;
- such that the following conditions are satisfied:
  (1) Each identifying property in $\mathcal{T}$ is primitive in $\mathcal{T}$.
  (2) Each concept appearing in an identification assertion of $\mathcal{T}$ (either as the identified concept, or in some test relation of some path) is a basic concept, i.e., a concept of the form A, $\exists Q$, or $\delta(U)$.
  (3) Each identification assertion in $\mathcal{T}$ is *local*, i.e., it has at least one path π_i such that $length(\pi_i) = 1$.

Here, the condition stated at point (1) says that, in *DL-Lite*$_{A,id}$ TBoxes, roles and attributes occurring in functionality assertions or in paths of identification constraints cannot be specialized.

3.2 Satisfiability and query answering over *DL-Lite* KBs

In this section, we present the main results on satisfiability and query answering in the DLs belonging to the *DL-Lite* family. In particular, we show that both problems are FOL-rewritable, and hence in AC^0 with respect to data complexity. Moreover, we show that KB satisfiability is in PTIME with respect to the size of the whole KB, and that query answering is NP-complete in combined complexity.

The results presented in [25, 31, 95] show that the complexity of the reasoning services that we are considering is the same for every DLs in the *DL-Lite* family. For this reason, in what follows, we use to refer to a generic *DL-Lite*-KB to indicate a KB expressed in *DL-Lite*$_{core}$, *DL-Lite*$_A$, or *DL-Lite*$_{A,id}$.

As for some other reasoning services like logical implication, it is shown in [31] that in *DL-Lite* they can be reduced to KB satisfiability.

In order to show that KB satisfiability in *DL-Lite* is FOL-rewritable, we need to resort to a main construction, namely the *canonical interpretation*. The canonical interpretation of a *DL-Lite* KB is an interpretation constructed according to the notion of *chase* [1]. In particular, we adapt here the notion of *restricted chase* adopted by Johnson and Klug in [61].

We start by defining the notion of applicable TBox assertions, and then we exploit applicable TBox assertions to construct the chase for a *DL-Lite*-KB. For easiness of exposition, we make use of the following notation for a basic role Q and two constants a_1 and a_2:

$$Q(a_1, a_2) \text{ denotes } \begin{cases} P(a_1, a_2), & \text{if } Q = P, \\ P(a_2, a_1), & \text{if } Q = P^-. \end{cases}$$

Definition 10. Let $\mathcal{S}$ be a set of ABox assertions. Then, a TBox assertion α is *applicable* in $\mathcal{S}$ to an ABox assertion $\beta \in \mathcal{S}$ if

- $\alpha = A_1 \sqsubseteq A_2$, $\beta = A_1(a)$, and $A_2(a) \notin \mathcal{S}$;
- $\alpha = A \sqsubseteq \exists Q$, $\beta = A(a)$, and there does not exist any constant a' such that $Q(a, a') \in \mathcal{S}$;
- $\alpha = A \sqsubseteq \delta(U)$, $\beta = A(a)$, and there does not exist any constant a' such that $U(a, a') \in \mathcal{S}$;
- $\alpha = \exists Q \sqsubseteq A$, $\beta = Q(a, a')$, and $A(a) \notin \mathcal{S}$;
- $\alpha = \exists Q_1 \sqsubseteq \exists Q_2$, $\beta = Q_1(a_1, a_2)$, and there does not exist any constant a_2' such that $Q_2(a_1, a_2') \in \mathcal{S}$;

- $\alpha = \exists Q_1 \sqsubseteq \delta(U)$, $\beta = Q_1(a_1, a_2)$, and there does not exist any constant a_2' such that $U(a_1, a_2') \in \mathcal{S}$;
- $\alpha = \delta(U) \sqsubseteq A$, $\beta = U(a, a')$, and $A(a) \notin \mathcal{S}$;
- $\alpha = \delta(U) \sqsubseteq \exists Q$, $\beta = U(a_1, a_2)$, and there does not exist any constant a_2' such that $Q(a_1, a_2') \in \mathcal{S}$;
- $\alpha = \delta(U_1) \sqsubseteq \delta(U_2)$, $\beta = U_1(a_1, a_2)$, and there does not exist any constant a_2' such that $U_2(a_1, a_2') \in \mathcal{S}$;
- $\alpha = Q_1 \sqsubseteq Q_2$, $\beta = Q_1(a_1, a_2)$, and $Q_2(a_1, a_2) \notin \mathcal{S}$.
- $\alpha = U_1 \sqsubseteq U_2$, $\beta = U_1(a_1, a_2)$, and $U_2(a_1, a_2) \notin \mathcal{S}$.

Let $\mathcal{K} = \langle T, \mathcal{A} \rangle$ be a consistent *DL-Lite*-KB. Note that the only assertions defined as *applicable* are those belonging to $\mathcal{T}_{inc}$. Now we show how applicable TBox assertions can be used to construct the chase of a *DL-Lite*-KB. Intuitively, the chase of $\mathcal{K}$ is a (possibly infinite) set of ABox assertions, constructed step-by-step starting from the ABox $\mathcal{A}$. The set $\mathcal{S}$ is initially set to to $\mathcal{A}$. At each step of the construction, an assertion $\alpha \in \mathcal{T}_{inc}$ is applied to an ABox assertion $\beta \in \mathcal{S}$. Applying an assertion α to β means adding a new suitable ABox assertions to $\mathcal{S}$, thus obtaining a new set $\mathcal{S}'$ in which α is not applicable to β anymore. For example, if $\alpha = Man \sqsubseteq Person$ is applicable in $\mathcal{S}$ to $\beta = Man(a)$, the ABox assertion to be added to $\mathcal{S}$ is $Person(a)$, i.e., $\mathcal{S}' = \mathcal{S} \cup \{Person(a)\}$.

Notice that such a construction process strongly depends on the order in which we select both the assertion in $\mathcal{T}_{inc}$ to be applied at each step and the ABox assertion to which such a TBox assertion is applied, as well as on which constants we introduce at each step. Therefore, a number of syntactically distinct sets of ABox assertions might result from different executions of this process. However, in [31] is shown that the result is unique up to renaming of constants occurring in each such a set.

In what follows, we assume, as in [61], to have a fixed infinite set of constants, whose symbols are ordered in lexicographic way, and we select TBox assertions, ABox assertions and constant symbols in lexicographic order. More formally, given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we denote with Γ_A the set of all constant symbols occurring in $\mathcal{A}$. Also, we assume to have an infinite set Γ_N of constant symbols not occurring in $\mathcal{A}$, such that the set $\Gamma_C = \Gamma_A \cup \Gamma_N$ is totally ordered in lexicographic way.

Definition 11. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*-KB, let n be the number of ABox assertions in $\mathcal{A}$, and let Γ_N be the set of constants defined above. Assume that the ABox assertions in $\mathcal{A}$ are numbered from 1 to n following their lexicographic order, and consider the following definition of sets $\mathcal{S}_j$ of ABox assertions:

- $\mathcal{S}_0 = \mathcal{A}$
- $\mathcal{S}_{j+1} = \mathcal{S}_j \cup \{\beta_{new}\}$, where β_{new} is an ABox assertion numbered with $n + j + 1$ in $\mathcal{S}_{j+1}$ and obtained as follows:

  let β be the first ABox assertion in $\mathcal{S}_j$ such that there exists an assertion $\alpha \in \mathcal{T}_{inc}$ applicable in $\mathcal{S}_j$ to β
  let α be the lexicographically first assertion applicable in $\mathcal{S}_j$ to β
  let a_{new} be the first constant of Γ_N that follows lexicographically all constants in $\mathcal{S}_j$
  case α, β **of**

(**cr1**)	$\alpha = A_1 \sqsubseteq A_2$	and	$\beta = A_1(a)$
	then $\beta_{new} = A_2(a)$		
(**cr2**)	$\alpha = A \sqsubseteq \exists Q$	and	$\beta = A(a)$
	then $\beta_{new} = Q(a, a_{new})$		
(**cr3**)	$\alpha = A \sqsubseteq \delta(U)$	and	$\beta = A(a)$
	then $\beta_{new} = U(a, a_{new})$		
(**cr4**)	$\alpha = \exists Q \sqsubseteq A$	and	$\beta = Q(a, a')$
	then $\beta_{new} = A(a)$		
(**cr5**)	$\alpha = \exists Q_1 \sqsubseteq \exists Q_2$	and	$\beta = Q_1(a, a')$
	then $\beta_{new} = Q_2(a, a_{new})$		
(**cr6**)	$\alpha = \exists Q \sqsubseteq \delta(U)$	and	$\beta = Q(a, a')$
	then $\beta_{new} = U(a, a_{new})$		
(**cr7**)	$\alpha = \delta(U) \sqsubseteq A$	and	$\beta = U(a, a')$
	then $\beta_{new} = A_1(a)$		
(**cr8**)	$\alpha = \delta(U) \sqsubseteq \exists Q$	and	$\beta = U(a, a')$
	then $\beta_{new} = Q(a, a_{new})$		
(**cr9**)	$\alpha = \delta(U_1) \sqsubseteq \delta(U_2)$	and	$\beta = U_1(a, a')$
	then $\beta_{new} = U_2(a, a_{new})$		
(**cr10**)	$\alpha = Q_1 \sqsubseteq Q_2$	and	$\beta = Q_1(a, a')$
	then $\beta_{new} = Q_2(a, a')$		
(**cr11**)	$\alpha = U_1 \sqsubseteq U_2$	and	$\beta = U_1(a, a')$
	then $\beta_{new} = U(a, a')$.		

Then, we call *chase of* $\mathcal{K}$, denoted $chase(\mathcal{K})$, the set of ABox assertions obtained as the infinite union of all $\mathcal{S}_j$, i.e.,

$$chase(\mathcal{K}) = \bigcup_{j \in \mathbb{N}} \mathcal{S}_j.$$

Note that only the TBox assertions in $\mathcal{T}_{inc}$ play a role in constructing $chase(\mathcal{K})$. Indeed $chase(\mathcal{K})$ depends only on the ABox $\mathcal{A}$ and the positive TBox assertion in $\mathcal{T}_{inc}$. In other words, we have that $chase(\langle \mathcal{T}, \mathcal{A} \rangle) = chase(\langle \mathcal{T}_{inc}, \mathcal{A} \rangle)$.

With the notion of chase in place, we give the definition of canonical interpretation.

Definition 12. The *canonical interpretation* $can(\mathcal{K}) = \langle \Delta^{can(\mathcal{K})}, \cdot^{can(\mathcal{K})} \rangle$ of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is the interpretation defined as follows:

- $\Delta^{can(\mathcal{K})} = \Gamma_C$,
- $a^{can(\mathcal{K})} = a$, for each constant a occurring in *chase*,
- $A^{can(\mathcal{K})} = \{a \mid A(a) \in chase(\mathcal{K})\}$, for each atomic concept A, and
- $P^{can(\mathcal{K})} = \{(a_1, a_2) \mid P(a_1, a_2) \in chase(\mathcal{K})\}$, for each atomic role P.

According to the above definition, it is easy to see that $can(\mathcal{K})$ is unique.

Now, we are ready to give a notable property that holds for $can(\mathcal{K})$.

Lemma 1. [25, 31] *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite-KB. Then, $can(\mathcal{K})$ is a model of $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$.*

As a consequence of Lemma 1, every *DL-Lite*-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ such that $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id} = \emptyset$ is always satisfiable, since we can always construct $can(\mathcal{K})$, which is a model for $\mathcal{K}$.

Before showing how $can(\mathcal{K})$ can be exploited for checking the satisfiability of generic *DL-Lite*-KB, we need to deal with query answering over *DL-Lite*-KB. To this end, we first present some preliminary properties of *DL-Lite*, and then we present the algorithm PerfectRef for the reformulation of conjunctive queries. Finally we will use this algorithm in order to describe a technique for answering union of conjunctive queries in *DL-Lite*.

First, we recall that, in the case where $\mathcal{K}$ is an inconsistent KB, the answer to a UCQ q is the finite set of tuples $AllTup(q, \mathcal{K})$. Therefore, we focus for the moment on the case where $\mathcal{K}$ is consistent.

We start by showing a central property of the canonical interpretation $can(\mathcal{K})$. In particular, the following lemma shows that, for every model $\mathcal{M}$ of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, there is a homomorphism from $can(\mathcal{K})$ to $\mathcal{M}$ that maps the objects in the extension of concepts and roles in $can(\mathcal{K})$ to objects in the extension of concepts and roles in $\mathcal{M}$.

Lemma 2. [31] *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent DL-Lite-KB, and let $\mathcal{M} = (\Delta^{\mathcal{M}}, \cdot^{\mathcal{M}})$ be a model of $\mathcal{K}$. Then, there is an homomorphism from $can(\mathcal{K})$ to $\mathcal{M}$.*

Based on the above property, we give the following theorem that states that the canonical interpretation $can(\mathcal{K})$ of a consistent *DL-Lite*-KB $\mathcal{K}$ is able to represent all models of $\mathcal{K}$ with respect to UCQs.

Theorem 1. [25, 31] *Let $\mathcal{K}$ be a consistent DL-Lite-KB, and let q be a UCQ over $\mathcal{K}$. Then, $cert(q, \mathcal{K}) = q^{can(\mathcal{K})}$.*

The above property shows that the canonical interpretation $can(\mathcal{K})$ is a correct representative of all the models of a consistent *DL-Lite*-KB with respect to the problem of answering UCQs. In other words, for every UCQ q, the answers to q over $\mathcal{K}$ correspond to the evaluation of q in $can(\mathcal{K})$.

Theorem 1, together with the fact that $can(\mathcal{K})$ depends only on the TBox assertions in $\mathcal{T}_{inc}$ has an interesting consequence for *consistent* KB, namely that the certain answers to a UCQ depend only on the set of TBox assertions in $\mathcal{T}_{inc}$ and the ABox, but are not affected by the TBox assertion in $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id}$.

Corollary 1. [25, 31] *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent DL-Lite-KB, and let q be a UCQ over $\mathcal{K}$. Then, $cert(q, \mathcal{K}) = cert(q, \langle \mathcal{T}_{inc}, \mathcal{A} \rangle)$.*

The following result is a direct consequence of Theorem 1 and Corollary 1.

Corollary 2. [31] *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent DL-Lite-KB, and let q be a UCQ over $\mathcal{K}$. Then $cert(q, \mathcal{K}) = q^{can\langle \mathcal{T}_{inc}, \mathcal{A} \rangle}$.*

In case of boolean conjunctive queries, the corollary above states that a boolean UCQ q is entailed by a *DL-Lite*-KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if and only if q evaluates to *true* in $can\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$.

In *DL-Lite* the canonical interpretation may be infinite, for this reason, it cannot be effectively computed in order to solve the query answering problem in *DL-Lite*. In order to avoid the construction of $can(\mathcal{K})$, in [31], authors provide the algorithm $\mathsf{PerfectRef}(q, \mathcal{T})$, which reformulates a UCQ q (considered as a set of CQs) by taking into account only the TBox assertions in $\mathcal{T}_{inc}$. In other words, instead of computing $can(\mathcal{K})$ in order to solve the query answering problem, they compile the TBox into the query, thus simulating the evaluation of the query over $can(\mathcal{K})$ by evaluating a finite reformulation of the query over the ABox considered as a database.

Informally, the algorithm $\mathsf{PerfectRef}(q, \mathcal{T})$ first reformulates the atoms of each CQ $q' \in q$, and produces a new query for each atom reformulation. During the process, TBox assertion in $\mathcal{T}_{inc}$ are used as rewriting rules, applied from right to left, which allow one to compile away in the reformulation the intensional knowledge (represented by $\mathcal{T}$) that is relevant for answering q.

In order to compute the certain answers to a UCQ q over a consistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we need to evaluate the set q_{ref} of CQs produced by $\mathsf{PerfectRef}(q, \mathcal{T})$ over the ABox $\mathcal{A}$ considered as a relational database. The following lemma prove correctness of the query answering technique described above.

Lemma 3. [25, 31] *Let $\mathcal{T}$ be a DL-Lite TBox, q a UCQ over $\mathcal{T}$, and q_{ref} the UCQ returned by* PerfectRef$(q, \mathcal{T})$. *For every ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent, $cert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = q_{ref}^{DB(\mathcal{A})}$.*

As we say earlier, the reformulation of a UCQ q with respect to a TBox $\mathcal{T}$ computed by PerfectRef depends only on the set of TBox assertion $\mathcal{T}_{inc}$, and then the TBox assertion in $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id}$ do not play any role in such a process. Hence we have that PerfectRef$(q, \mathcal{T})$ = PerfectRef$(q, \mathcal{T}_{inc})$.

We now establish the complexity of the algorithm PerfectRef.

Lemma 4. [31] *Let $\mathcal{T}$ be a DL-Lite TBox, and q a UCQ over $\mathcal{T}$. The algorithm* PerfectRef$(q, \mathcal{T})$ *runs in time polynomial in the size of $\mathcal{T}$.*

We now turn our attention to the satisfiability problem, presenting the following result, which asserts that to establish satisfiability of a KB, we can resort to constructing the canonical interpretation.

Lemma 5. [25, 31] *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite-KB. Then, $can(\mathcal{K})$ is a model of $\mathcal{K}$ if and only if $\mathcal{K}$ is consistent.*

Consider a *DL-Lite*-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. Lemma 5, together with the fact that $can(\mathcal{K})$ is a model for $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$, implies that $\mathcal{K}$ is inconsistent if and only if there exists an assertion α belonging to $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id}$, such that $can(\mathcal{K}) \models \neg\alpha$. Now, consider the consistent KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$, from the observation above, it follows that $\mathcal{K}$ is inconsistent if and only if $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle \models \neg\alpha$. From this notable property of *DL-Lite* arises the idea presented in [31, 94] to reduce the problem of checking the satisfiability of $\mathcal{K}$ to the task of evaluating a suitable query $\mathcal{Q}_{violated(\mathcal{T})}$ over the consistent KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$. Intuitively, such a query, is a union of first-order queries that represent the negation of an assertion in $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle \models \neg\alpha$. By applying a slight variation to the algorithm PerfectRef, it is then possible compute the reformulation of $\mathcal{Q}_{violated(\mathcal{T})}$ with the aim to evaluate this reformulation over the ABox considered as a database. More formally.

Theorem 2. [25, 31] *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite-KB, and $\mathcal{Q}_{violated(\mathcal{T})}$ be the query which encodes the violation of the assertions in $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id}$. $\mathcal{K}$ is inconsistent if an only if* $(\text{PerfectRef}(\mathcal{Q}_{violated(\mathcal{T})}, \mathcal{T}_{inc}))^{DB(\mathcal{A})} = \emptyset$.

We will go into details of this technique for consistency checking in the next chapter.

We present the algorithm Satisfiable$(\mathcal{K})$, that takes as input a *DL-Lite*-KB and returns *true* if the KB is consistent, and *false* otherwise.

Correctness of the algorithm Satisfiable, together with the fact that the perfect reformulation is independent of the ABox, and, according to Lemma 4, can be computed in PTime in the size of the TBox, allows us to give the following theorem.

Input: a *DL-Lite*-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: *true* or *false*
begin
  if $(\mathsf{PerfectRef}(\mathcal{Q}_{violated(\mathcal{T})}, \mathcal{T}_{inc}))^{DB(\mathcal{A})} = \emptyset$
  then return *true*;
  else return *false*;
end

Algorithm 1: The algorithm Satisfiable that checks satisfiability of a *DL-Lite*-KB.

Theorem 3. [25] *In DL-Lite, KB satisfiability is FOL-rewritable, and hence in* AC^0 *in the size of the ABox (data complexity).*

We observe that the algorithm Satisfiable allows to to give a coNP upper bound of KB satisfiability in *DL-Lite* with respect to the size of the TBox.

With the algorithm Satisfiable in place we can present the algorithm Answer that, given a *DL-Lite*$_A$-KB $\mathcal{K}$ and a UCQ q, computes $cert(q, \mathcal{K})$. We remind the reader that in the case where $\mathcal{K}$ is an inconsistent KB, the answer to a UCQ q is the finite set of tuples $AllTup(q, \mathcal{K})$.

Input: a UCQ q and a *DL-Lite*-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: a set of tuples
begin
  if not $\mathsf{Satisfiable}(\mathcal{K})$
  then return $AllTup(q, \mathcal{K})$;
  else return $(\mathsf{PerfectRef}(q, \mathcal{T}))^{DB(\mathcal{A})}$;
end

Algorithm 2: The algorithm Answer that computes the certain answers to a UCQ over a *DL-Lite*-KB.

By exploiting the results given before in this section, and from the correctness of the algorithm Answer for computing the certain answers to a UCQ over a *DL-Lite*-KB, we are able to characterize the complexity of answering UCQs in *DL-Lite*.

Theorem 4. [25] *Answering UCQs in DL-Lite is in* AC^0 *in the size of the ABox (data complexity), and NP-complete in combined complexity.*

Also in this case we can give a coNP upper bound of query answering in *DL-Lite* with respect to the size of the TBox.

We conclude this section by reporting some interesting property of *DL-Lite*-KBs given in [40].

As a consequence of the fact that $chase(\mathcal{K})$ identifies a canonical model for conjunctive queries over a consistent *DL-Lite*-KB $\mathcal{K}$ we have the following lemma.

Lemma 6. [40] *For every consistent DL-Lite-KB $\mathcal{K}$, and for every ABox assertion α, $\mathcal{K} \models \alpha$ if and only if $\alpha \in chase(\mathcal{K})$.*

By exploiting this result it is possible to give the following theorem.

Theorem 5. [40] *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent DL-Lite-KB, and let α be an ABox assertion. If $\mathcal{K} \models \alpha$, then there exists in $\mathcal{A}$ an ABox assertion α' such that $\langle \mathcal{T}, \{\alpha'\} \rangle \models \alpha$.*

We recall that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ denote the closure of an ABox $\mathcal{A}$ with respect to a TBox $\mathcal{T}$, i.e., the set of all ABox assertions α that are formed with individuals in $\mathcal{A}$, and are logically implied by $\langle \mathcal{T}, \mathcal{A} \rangle$. In the rest of this work, for an ABox assertion α_1, we denote by $\mathsf{Subsumee}_{\langle \mathcal{T}, \mathcal{A} \rangle}(\alpha_1)$ the set of atoms $\alpha_2 \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \{\alpha_2\} \rangle \models \alpha_1$. Moreover, by observing the rules given in Definition 11 for computing $chase(\mathcal{K})$, it is immediate to come up with an algorithm for computing $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ in quadratic time with respect to the size of $\mathcal{T}$ and $\mathcal{A}$.

An interesting consequence of Theorem 5 is that, given a *DL-Lite*-TBox and two $\mathcal{T}$-consistent set of ABox assertions $\mathcal{A}_1$ and $\mathcal{A}_2$, we have

$$\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1 \cup \mathcal{A}_2) = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) \cup \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2).$$

Part II

The Description Logic *DL-Lite*$_{A,id,den}$

Chapter 4

The Language of *DL-Lite*$_{A,id,den}$

In this chapter, we present a new logic of the *DL-Lite* family [30], called *DL-Lite*$_{A,id,den}$, which extends *DL-Lite*$_{A,id}$ with denial assertions. Denial Assertions (DAs) are assertions that allow for imposing that the answer to a certain boolean conjunctive query over the KB is *false*, analogous to negative constraints in [20]. A DA is particularly useful for specifying general forms of disjointness, which are not supported in traditional DLs.

4.1 Denial assertions

Essentially, a *DL-Lite*$_{A,id,den}$-KB is a *DL-Lite*$_{A,id}$-KB that is also equipped with denial assertions. Therefore a *DL-Lite*$_{A,id,den}$ TBox may contains four types of intensional assertions, namely inclusion assertions, functionality assertions, identification assertions and denial assertions. Inclusion assertions, functionality assertions, identification assertions in *DL-Lite*$_{A,id,den}$ are as in *DL-Lite*$_{A,id}$.

Syntax. In *DL-Lite*$_{A,id,den}$ a denial assertion is an assertion of the form

$$\forall \vec{y}.(conj(\vec{t}) \rightarrow \bot),$$

where $\vec{y}$ is a set of variables and $\vec{t}$ is a set of terms (i.e., constants or variables) such that each variable in $\vec{t}$ is also in $\vec{y}$, and $conj(\vec{t})$, as for CQs, is a conjunction of atoms of the form $A(z)$, $P(z, z')$ $U(z, z')$ where A is an atomic concept, P is an atomic role and U is an attribute, and z, z' are terms.

Intuitively, a denial assertion states that the answer to the boolean conjunctive query $\exists \vec{y}.conj(\vec{t})$ is *false*. This kind of assertion can be used to specify general form of disjointness assertions which cannot be expressed in other DLs of the *DL-Lite* family. Moreover, by means of a denial assertion we are able to specify irreflexivity of a role. For example the denial assertion

$\forall x.(hasFather(x,x) \rightarrow \bot)$ implies that the role $hasFather$ is irreflexive, i.e., that a person cannot be father of himself.

Semantics. We now specify the semantics of a denial assertion, by defining when an interpretation $\mathcal{I}$ satisfies a DA α. Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and a denial assertion $\alpha = \forall \vec{y}.(conj(\vec{t}) \rightarrow \bot)$, we say that $\mathcal{I}$ satisfies α if the FOL-formula $\exists \vec{y}.conj(\vec{t})$ evaluates to *false* in $\mathcal{I}$.

4.2 *DL-Lite*$_{A,id,den}$ KBs

With the notion of denial assertions given in the previous section in place, we can give in this section the formal definition of a *DL-Lite*$_{A,id,den}$-KB.

As for *DL-Lite*$_{A,id}$, a *DL-Lite*$_{A,id,den}$-TBox $\mathcal{T}$ must satisfy the condition stating that each role Q (resp. attribute U) appearing in $\mathcal{T}$ in a functionality assertion or in an identification assertion, cannot appear in the right-hand side of positive role (resp. attribute) inclusion assertions, i.e., in assertions of the form $Q' \sqsubseteq Q$ (resp. $U' \sqsubseteq U$).

The following definition describes formally a KB expressed in *DL-Lite*$_{A,id,den}$.

Definition 13. A *DL-Lite*$_{A,id,den}$-KB is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

- the TBox $\mathcal{T}$ is a finite set of intensional assertions partitioned into six sets $\mathcal{T}_{inc}$, $\mathcal{T}_{type}$, $\mathcal{T}_{disj}$, $\mathcal{T}_{funct}$, $\mathcal{T}_{id}$, and $\mathcal{T}_{den}$ where:
  - $\mathcal{T}_{inc}$ is a finite set of positive inclusion assertions of the form:

$$\begin{array}{ll} B \sqsubseteq B' & (\textit{inclusions between concepts}), \\ Q \sqsubseteq Q' & (\textit{inclusions between roles}), \\ U \sqsubseteq U' & (\textit{inclusions between attributes}); \end{array}$$

  - $\mathcal{T}_{type}$ is a finite set of positive inclusion assertions of the form:

$$E \sqsubseteq F \qquad (\textit{inclusions between value-domains});$$

  - $\mathcal{T}_{disj}$ is a finite set of negative inclusion assertions of the form:

$$\begin{array}{ll} B \sqsubseteq \neg B' & (\textit{disjunctions between concepts}), \\ Q \sqsubseteq \neg Q' & (\textit{disjunctions between roles}), \\ U \sqsubseteq \neg U' & (\textit{disjunctions between attributes}); \end{array}$$

  - $\mathcal{T}_{funct}$ is a finite set of functionality assertions of the form:

$$\begin{array}{ll} (\mathsf{funct}\ Q) & (\textit{role functionality}), \\ (\mathsf{funct}\ U) & (\textit{attribute functionality}); \end{array}$$

– $\mathcal{T}_{id}$ is a finite set of identification assertions of the form:

$$(id\ B\ \pi_1, \ldots, \pi_n);$$

– $\mathcal{T}_{den}$ is a finite set of denial assertions of the form:

$$\forall \vec{y}.(conj(\vec{t}) \rightarrow \bot);$$

where B and B' denote *basic concepts*, E denotes a basic value-domain, F a *value-domain expression*, Q and Q' *basic roles*, U and U' denote *attributes*, $\vec{y}$ is a set of variables and $\vec{t}$ is a set variables and constants in Γ_C, and $conj(\vec{t})$, is a conjunction of atoms in Γ_P;

- the ABox $\mathcal{A}$ is a finite set of extensional assertions over Γ of the form:

$$A(a) \qquad P(a,b) \qquad U(a,v)$$

were A, P, and U in Γ_P denoting respectively an atomic concept symbol, an atomic role symbol, and an attribute symbol, and a and b are constants in Γ_O and v is a constant in Γ_V;

- the following conditions are satisfied:

(1) Each identifying property in $\mathcal{T}$ is primitive in $\mathcal{T}$.

(2) Each concept appearing in an identification assertion of $\mathcal{T}$ (either as the identified concept, or in some test relation of some path) is a basic concept, i.e., a concept of the form A, $\exists Q$, or $\delta(U)$.

(3) Each identification assertion in $\mathcal{T}$ is *local*, i.e., it has at least one path π_i such that $length(\pi_i) = 1$.

As usual, an interpretation $\mathcal{I}$ is a model for a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K}$ (resp., TBox $\mathcal{T}$, ABox $\mathcal{A}$), written $\mathcal{I} \models \mathcal{K}$ (resp., $\mathcal{I} \models \mathcal{T}, \mathcal{I} \models \mathcal{A}$), if and only if $\mathcal{I}$ satisfies all assertions in $\mathcal{K}$ (resp., $\mathcal{T}, \mathcal{A}$).

From the definition above, we have that a *DL-Lite*$_{A,id,den}$ TBox is a set $\mathcal{T}$ partitioned into six pairwise disjoint sets $\mathcal{T} = \mathcal{T}_{inc} \cup \mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id} \cup \mathcal{T}_{den}$, where:
– $\mathcal{T}_{inc}$ contains only TBox assertions of the form $B_1 \sqsubseteq B_2$, $Q_1 \sqsubseteq Q_2$, $U_1 \sqsubseteq U_2$;
– $\mathcal{T}_{type}$ contains only TBox assertions of the form $E \sqsubseteq F$;
– $\mathcal{T}_{disj}$ contains only assertions of the form $B_1 \sqsubseteq \neg B_2$, $Q_1 \sqsubseteq \neg Q_2$, $U_1 \sqsubseteq \neg U_2$;
– $\mathcal{T}_{funct}$ contains only functionality assertions over basic roles and attribute;
– $\mathcal{T}_{id}$ contains only identification assertions;
– $\mathcal{T}_{den}$ contains only denial assertions.

and where roles and attributes occurring in functionality assertions in $\mathcal{T}_{den}$ or in paths of identification constraints in $\mathcal{T}_{id}$ cannot be specialized, i.e., they cannot appear on the right-hand side of assertions in $\mathcal{T}_{inc}$.

Clearly, a *DL-Lite*$_{A,id,den}$-KB where $\mathcal{T}_{den} = \emptyset$ is a *DL-Lite*$_{A,id}$-KB, and a *DL-Lite*$_{A,id,den}$-KB where $\mathcal{T}_{den} \cup \mathcal{T}_{id} = \emptyset$ is a *DL-Lite*$_A$-KB.

Example 2. In this example we present a KB modeling a very small portion of the telephone access network of a telecommunication industry. The telephone access network is that portion of the whole telephone network which connects customer homes to the nearest commutation node belonging to the network. In particular our KB focuses on how a device (*Device*) in the telephone access network may be connected to another device. We can connect two devices by connecting ports (*Port*) of (*of*) such devices. Each port is associated to exactly one device. There exists different kinds of port in a device which can be used for different purposes. Among the other, there are incoming ports (*PortIn*) and outgoing ports (*PortOut*). An incoming port cannot be also an outgoing port. A number (*number*) is associated to each port. There cannot exist two ports of the same device having the same number. A port can be connected to (*connectedTo*) another port. Connection between ports of devices belonging to a telephone access network has to obey the following rules:

- every connection between ports involves exactly two ports;
- two ports of the same device cannot be connected to each other;
- there cannot exist an incoming port and an outgoing port of the same device that are connected to ports of the same device.

Figure 4.1 attempts to depicts the domain described above.
The following *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$ captures our domain, where:

- the set $\mathcal{T}_{inc}$ is constituted by the following assertions:
  - $PortIn \sqsubseteq Port$,
  - $PortOut \sqsubseteq Port$,
  - $\exists connectedTo \sqsubseteq Port$,
  - $\exists of \sqsubseteq Port$,
  - $\exists connectedTo^- \sqsubseteq Port$,
  - $\exists of^- \sqsubseteq Device$,
  - $Port \sqsubseteq \delta(number)$,
  - $Device \sqsubseteq \exists of^-$,
  - $\delta(number) \sqsubseteq Port$
  - $Port \sqsubseteq \exists of$;
- the set $\mathcal{T}_{type}$ is constituted by the following assertion:
  - $\rho(number) \sqsubseteq$ `xsd:integer`;
- the set $\mathcal{T}_{disj}$ is constituted by the following assertions:
  - $PortIn \sqsubseteq \neg PortOut$,
  - $Port \sqsubseteq \neg Device$;

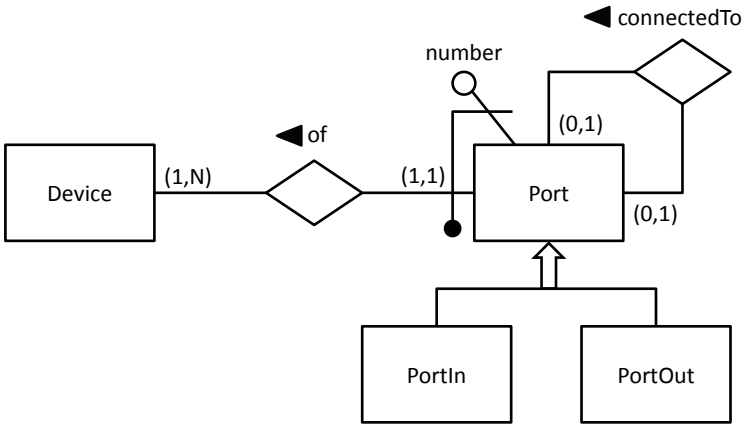

Figure 4.1: Graphical representation of a telephone access network

- the set $\mathcal{T}_{funct}$ is constituted by the following assertions:

  – $(funct\ connectedTo)$, – $(funct\ of\ )$,
  – $(funct\ connectedTo^{-}\ )$, – $(funct\ number)$;

- the set $\mathcal{T}_{id}$ is constituted by the following assertion:

  – $(id\ Port\ number,\ of\ )$;

- the set $\mathcal{T}_{den}$ is constituted by the following assertions:

  – $\forall x, y, z.(Port(x) \wedge Port(y) \wedge of(x, z) \wedge of(y, z)$
  $\wedge connectedTo(x, y) \rightarrow \bot)$,
  – $\forall x, y, z, k, w, v.(PortOut(x) \wedge of(x, y) \wedge connectedTo(x, z) \wedge of(z, k)$
  $\wedge\ PortIn(w) \wedge of(w, y) \wedge connectedTo(w, v)$
  $\wedge\ of(v, k) \rightarrow \bot)$.

In the TBox, the identification assertion model the fact that there does not exist in a device two ports having the same number. Moreover, the denial assertions model the following aspects:

- two ports of the same device cannot be connected to each other;

- there cannot exist an incoming port and an outgoing port of the same device that are connected to ports of the same device.

□

Chapter 5

Reasoning over *DL-Lite*$_{A,id,den}$ Knowledge Bases

In this chapter we study reasoning in *DL-Lite*$_{A,id,den}$. In particular, we focus on query answering and KB satisfiability. We show that in *DL-Lite*$_{A,id,den}$, as for the other DLs the *DL-Lite* family, query answering can be managed efficiently with respect to the size of the ABox. Moreover, we show that the introduction of denial assertions in the TBox leads to increase the complexity of both KB satisfiability and query answering problems with respect to the size of the whole KB.

5.1 Satisfiability in *DL-Lite*$_{A,id,den}$

In this section we present a technique for checking satisfiability in *DL-Lite*$_{A,id,den}$.

Let $\mathcal{T}$ be a satisfiable TBox, and let V be a set of ABox assertions. We say that V is a $\mathcal{T}$-inconsistent set if the KB $\mathcal{K} = \langle \mathcal{T}, V \rangle$ is inconsistent. Since, due to the characteristics of *DL-Lite*$_{A,id,den}$, we have that a *DL-Lite*$_{A,id,den}$ TBox is always satisfiable, i.e., it admits always a model under standard FOL-semantics, given an inconsistent KB $\langle \mathcal{T}, \mathcal{A} \rangle$, there exists always al least one subset V of $\mathcal{A}$, not necessary different from $\mathcal{A}$, that is a $\mathcal{T}$-inconsistent set.

In the previous chapter, Theorem 3 establishes a notable property of the DLs of the *DL-Lite* family, which is that, by virtue of the careful definition of the expressive power of that logic, satisfiability is FOL-rewritable [31]. This means that for every TBox $\mathcal{T}$ expressed in *DL-Lite*, it is possible to build a boolean FOL-query $\mathcal{Q}_{\mathcal{T}}^{unsat}$ over the alphabet of $\mathcal{T}$, such that for every non-empty ABox $\mathcal{A}$, the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent if and only if $\mathcal{Q}_{\mathcal{T}}^{unsat}$ evaluate to *false* in $DB(\mathcal{A})$, where $DB(\mathcal{A})$ is as defined in Section 2.5.

To be more precise, $\mathcal{Q}_{\mathcal{T}}^{unsat}$ is a union of boolean FOL-queries represented

here as a set of boolean queries corresponding to FOL-sentences of the form:

$$\exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} A_i(t_i^1) \wedge \bigwedge_{i=1}^{m} T_i(t_i^2) \wedge \bigwedge_{i=1}^{\ell} S_i(t_i^3, t_i^4) \wedge \bigwedge_{i=1}^{h} t_i^5 \neq t_i^6 \tag{5.1}$$

where every A_i is an atomic concept, every T_i is a value-domain, every S_i is a binary predicate, which is either an atomic role or an attribute, every t_i^j is a term (i.e., either a constant or a variable), and $z_1, \ldots, z_k$ are all the variables of the query. Notice that each sentence of the form (5.1) is a boolean conjunctive query enriched with limited forms of inequalities.

In what follows, we describe in detail this technique for checking satisfiability, and we show that the same property holds also for *DL-Lite*$_{A,id,den}$.

Firstly, we give the following result, which is the analogue of Lemma 5, and that allows us to establish a fundamental "separation" property for DAs, similar to the one for negative, functionality, and identification assertions given in [31, Lemma 14] and in [32, Theorem 6].

Lemma 7. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB. $\mathcal{K}$ is consistent if and only if $can(\mathcal{K})$ is a model of $\mathcal{K}$.*

Proof.
($\Rightarrow$) Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent *DL-Lite*$_{A,id,den}$. We have to prove that $can(\mathcal{K})$ is a model of $\mathcal{K}$. Suppose, by way of contradiction, that $can(\mathcal{K})$ is not a model of $\mathcal{K}$. Lemma 1 guarantees that $can(\mathcal{K})$ is a model for $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$. We also know from Lemma 5 that $can(\mathcal{K})$ satisfies the assertions in $\mathcal{T}_{inc} \cup \mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id}$. It follows that there is a denial assertion $\forall \vec{x}.(conj(\vec{x}) \rightarrow \bot)$ in $\mathcal{T}_{den}$ such that $can(\mathcal{K}) \models \neg(\forall \vec{x}.conj(\vec{t}) \rightarrow \bot)$, i.e., $can(\mathcal{K}) \models q()$, where $q()$ is the conjunctive query $\exists \vec{x}.conj(\vec{t})$. From Corollary 2 it follows that $\langle \mathcal{T} \setminus \mathcal{T}_{den}, \mathcal{A} \rangle \models q()$. Since *DL-Lite*$_{A,id,den}$ is monotonic, then $\langle \mathcal{T}, \mathcal{A} \rangle \models q()$. This means that $\mathcal{K} \models \neg(\forall \vec{x}.(conj(\vec{t}) \rightarrow \bot))$. Since the denial assertion $\forall \vec{x}.(conj(\vec{t}) \rightarrow \bot)$ belongs to $\mathcal{T}$, we have that $\mathcal{K} \models \forall \vec{x}.(conj(\vec{t}) \rightarrow \bot)$, and that $\mathcal{K} \models \neg(\forall \vec{x}.(conj(\vec{t}) \rightarrow \bot))$ at the same time. Hence, K is inconsistent, which is a contradiction.
($\Leftarrow$) If $can(\mathcal{K})$ is a model of $\mathcal{K}$, then $\mathcal{K}$ is clearly satisfiable. ∎

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_{A,id,den}$-KB. Intuitively, the lemma above states that we can refer to the canonical interpretation $can(\mathcal{K})$ for verifying satisfiability of $\mathcal{K}$. More specifically, since Lemma 1 states that $can(\mathcal{K})$ is a model for $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$, then the KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$ is always consistent. Therefore, $\mathcal{K}$ may be inconsistent only if there is an assertion α in $\mathcal{T} \setminus \mathcal{T}_{inc}$ such that $can(\mathcal{K}) \models \neg\alpha$.

Lemma 8. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB. $\mathcal{K}$ is inconsistent if and only if there exists an assertion $\alpha \in \mathcal{T} \setminus \mathcal{T}_{inc}$ such that $can(\mathcal{K}) \models \neg\alpha$.*

Proof.
($\Rightarrow$) We show that if $\mathcal{K}$ is inconsistent, then there is an assertion $\alpha \in \mathcal{T} \setminus \mathcal{T}_{inc}$ such that $can(\mathcal{K}) \models \neg\alpha$. Suppose, by way of contradiction, that for each assertion $\alpha \in \mathcal{T} \setminus \mathcal{T}_{inc}$, $can(\mathcal{K}) \models \alpha$. Since Lemma 1 guarantees that $can(\mathcal{K})$ is a model for $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$, then $can(\mathcal{K})$ is a model for $\mathcal{K}$. Hence, $\mathcal{K}$ is consistent, which is a contradiction.
($\Leftarrow$) Let α be an assertions in $\mathcal{T} \setminus \mathcal{T}_{inc}$. We have to prove that if $can(\mathcal{K}) \models \neg\alpha$, then $\mathcal{K}$ is inconsistent. Toward a contradiction, suppose that $\mathcal{K}$ is consistent. From Lemma 7, it follows that $can(\mathcal{K})$ is a model for $\mathcal{K}$, and then, since $can(\mathcal{K})(\mathcal{K}) \models \neg\alpha$, we have that $\mathcal{K} \models \neg\alpha$. Hence, since $\alpha \in \mathcal{T}$, we have that $\mathcal{K} \models \alpha$ and that $\mathcal{K} \models \neg\alpha$, which means that $\mathcal{K}$ is inconsistent. Hence, we have a contradiction. ∎

A notable consequence of Lemma 8 is that one can verify if a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent by checking if $can(\mathcal{K}) \models \neg\alpha$, for each $\alpha \in \mathcal{T} \setminus \mathcal{T}_{inc}$. In particular, if $\alpha \in \mathcal{T}_{den}$, then $\neg\alpha$ is a boolean conjunctive queries. Hence, by exploiting Corollary 2, we can check if every denial assertion $\alpha \in \mathcal{T}_{den}$ if satisfied, by evaluating the boolean conjunctive query $\neg\alpha$ over the consistent KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$. If $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle \models \neg\alpha$, then $\mathcal{K}$ is inconsistent. As shown in [31, 32] we can operate in a similar way for the other assertions in $\mathcal{T} \setminus (\mathcal{T}_{inc} \cup \mathcal{T}_{den})$, i.e., we can check if $\mathcal{K}$ is consistent by evaluating suitable boolean queries over the consistent KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$. In what follows, we show how it is possible to compute the before mentioned set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ by means of such queries.

Before presenting the method for computing the query $\mathcal{Q}_{\mathcal{T}}^{unsat}$, we need to introduce some definitions.

By following the same approach of [31] we next introduce the notion of *NI-closure* of a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$.

Definition 14. Let $\mathcal{T} = \mathcal{T}_{inc} \cup \mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id} \cup \mathcal{T}_{den}$ be a *DL-Lite*$_{A,id,den}$ TBox. We call *NI-closure* of $\mathcal{T}$, denoted by $cln(\mathcal{T})$, the TBox defined inductively as follows:

1. each functionality assertion in $\mathcal{T}_{funct}$ is in $cln(\mathcal{T})$;
2. each identification assertion in $\mathcal{T}_{id}$ is in $cln(\mathcal{T})$;
3. each denial assertion in $\mathcal{T}_{den}$ is in $cln(\mathcal{T})$;
4. each negative assertion in $\mathcal{T}_{disj}$ is in $cln(\mathcal{T})$;
5. if $B_1 \sqsubseteq B_2$ is in $\mathcal{T}$ and $B_2 \sqsubseteq \neg B_3$ or $B_3 \sqsubseteq \neg B_2$ is in $cln(\mathcal{T})$, then also $B_1 \sqsubseteq \neg B_3$ is in $cln(\mathcal{T})$;
6. if $Q_1 \sqsubseteq Q_2$ is in $\mathcal{T}$ and $Q_2 \sqsubseteq \neg Q_3$ or $Q_3 \sqsubseteq \neg Q_2$ is in $cln(\mathcal{T})$, then also $Q_1 \sqsubseteq \neg Q_3$ is in $cln(\mathcal{T})$.
7. if $U_1 \sqsubseteq U_2$ is in $\mathcal{T}$ and $U_2 \sqsubseteq \neg U_3$ or $U_3 \sqsubseteq \neg U_2$ are in $cln(\mathcal{T})$, then $U_1 \sqsubseteq \neg U_3$ is also in $cln(\mathcal{T})$;
8. if $B_1 \sqsubseteq \neg B_2$ is in $cln(\mathcal{T})$, then $B_2 \sqsubseteq \neg B_1$ is also in $cln(\mathcal{T})$;

9. if $Q_1 \sqsubseteq \neg Q_2$ is in $cln(\mathcal{T})$, then $Q_2 \sqsubseteq \neg Q_1$ is also in $cln(\mathcal{T})$;
10. if $Q_1 \sqsubseteq \neg Q_2$ is in $cln(\mathcal{T})$, then $Q_1^- \sqsubseteq \neg Q_2^-$ is also in $cln(\mathcal{T})$;
11. if $U_1 \sqsubseteq \neg U_2$ is in $cln(\mathcal{T})$, then $U_2 \sqsubseteq \neg U_1$ is also in $cln(\mathcal{T})$;
12. if $Q_1 \sqsubseteq Q_2$ is in $\mathcal{T}$ and $\exists Q_2 \sqsubseteq \neg B$ or $B \sqsubseteq \neg \exists Q_2$ is in $cln(\mathcal{T})$, then also $\exists Q_1 \sqsubseteq \neg B$ is in $cln(\mathcal{T})$;
13. if $Q_1 \sqsubseteq Q_2$ is in $\mathcal{T}$ and $\exists Q_2^- \sqsubseteq \neg B$ or $B \sqsubseteq \neg \exists Q_2^-$ is in $cln(\mathcal{T})$, then also $\exists Q_1^- \sqsubseteq \neg B$ is in $cln(\mathcal{T})$;
14. if $U_1 \sqsubseteq U_2$ is in $\mathcal{T}$ and $\delta(Q_2) \sqsubseteq \neg B$ or $B \sqsubseteq \neg \delta(U_2)$ is in $cln(\mathcal{T})$, then also $\delta(Q_1) \sqsubseteq \neg B$ is in $cln(\mathcal{T})$;
15. if $U_1 \sqsubseteq U_2$ is in $\mathcal{T}$ and $\rho(U_2) \sqsubseteq \neg T_1$ is in $cln(\mathcal{T})$, then $\rho(U_1) \sqsubseteq \neg T_1$ is in $cln(\mathcal{T})$;
16. if $\exists Q_1 \sqsubseteq \neg \exists Q_2$ or $\exists Q_1^- \sqsubseteq \neg \exists Q_2^-$ are in $cln(\mathcal{T})$, then $Q_1 \sqsubseteq \neg Q_2$ is also in $cln(\mathcal{T})$;
17. if $\delta(U_1) \sqsubseteq \neg \delta(U_2)$ is in $cln(\mathcal{T})$, then $U_1 \sqsubseteq \neg U_2$ is also in $cln(\mathcal{T})$;
18. if $\rho(U_1) \sqsubseteq T_1$ and $\rho(U_2) \sqsubseteq T_2$ are in $cln(\mathcal{T})$, with $T_1 \neq T_2$, then $U_1 \sqsubseteq \neg U_2$ is also in $cln(\mathcal{T})$;
19. if $\rho(U) \sqsubseteq T_i$ is in $\mathcal{T}$, then $\rho(U) \sqsubseteq \neg T_j$, for each value-domain $T_j \neq T_i$ in $\{T_1, \ldots \mathcal{T}_n\}$, is in $cln(\mathcal{T})$;
20. if at least one of the four assertions $\exists Q \sqsubseteq \neg \exists Q$, $\exists Q^- \sqsubseteq \neg \exists Q^-$, $Q \sqsubseteq \neg Q$, and $Q^- \sqsubseteq \neg Q^-$ are in $cln(\mathcal{T})$, then all four assertions are in $cln(\mathcal{T})$;
21. if at least one of the two assertions $\delta(U) \sqsubseteq \neg \delta(U)$ and $U \sqsubseteq \neg U$ are in $cln(\mathcal{T})$, then both assertions are in $cln(\mathcal{T})$;

The above definition is a straightforward adaptation to *DL-Lite*$_{A,id,den}$ of the notion of *NI-closure* given in [31]. We observe that rule 18 and rule 19 are used to make explicit the implicit disjointness rising from the fact that the TBox semantics imposes that for each pair T_i and T_j of different value-domains, T_i and T_j are disjoint.

Example 3. Consider the *DL-Lite*$_{A,id,den}$-TBox $\mathcal{T}$ presented in Example 2. According with Definition 14, the set $cln(\mathcal{T})$ contains the following TBox assertions:

(1.) $(funct\ connectedTo)$, (2.) $(funct\ of)$,
(3.) $(funct\ connectedTo^-)$, (4.) $(funct\ number)$,
(5.) $\rho(number) \sqsubseteq \neg$`xsd:string`, (6.) $\rho(number) \sqsubseteq \neg$`xsd:dateTime`,
(7.) $PortIn \sqsubseteq \neg PortOut$, (8.) $PortOut \sqsubseteq \neg PortIn$,
(9.) $Port \sqsubseteq \neg Device$, (10.) $Device \sqsubseteq \neg Port$,
(11.) $PortIn \sqsubseteq \neg Device$, (12.) $Device \sqsubseteq \neg PortIn$,
(13.) $PortOut \sqsubseteq \neg Device$, (14.) $Device \sqsubseteq \neg PortOut$,
(15.) $\exists connectedTo \sqsubseteq \neg Device$, (16.) $Device \sqsubseteq \neg \exists connectedTo$,
(17.) $\exists connectedTo^- \sqsubseteq \neg Device$, (18.) $Device \sqsubseteq \neg \exists connectedTo^-$,
(19.) $\exists of \sqsubseteq \neg \exists Device$, (20.) $Device \sqsubseteq \neg \exists of$,

(21.) $\delta(number) \sqsubseteq \neg\exists Device$,
(22.) $Device \sqsubseteq \neg\delta(number)$,
(23.) $Port \sqsubseteq \neg\exists of^-$,
(24.) $\exists of^- \sqsubseteq \neg Port$,
(25.) $PortIn \sqsubseteq \neg\exists of^-$,
(26.) $\exists of^- \sqsubseteq \neg PortIn$,
(27.) $PortOut \sqsubseteq \neg\exists of^-$,
(28.) $\exists of^- \sqsubseteq \neg PortOut$,
(29.) $\exists connectedTo \sqsubseteq \neg\exists of^-$,
(30.) $\exists of^- \sqsubseteq \neg\exists connectedTo$,
(31.) $\exists connectedTo^- \sqsubseteq \neg\exists of^-$,
(32.) $\exists of^- \sqsubseteq \neg\exists connectedTo^-$,
(33.) $\exists of \sqsubseteq \neg\exists of^-$,
(34.) $\exists of^- \sqsubseteq \neg\exists of$,
(35.) $\delta(number) \sqsubseteq \neg\exists of^-$,
(36.) $\exists of^- \sqsubseteq \neg\delta(number)$,
(37.) $of \sqsubseteq \neg of^-$,
(38.) $of^- \sqsubseteq \neg of$,
(39.) $of \sqsubseteq \neg connectedTo$,
(40.) $connectedTo \sqsubseteq \neg of$,
(41.) $of^- \sqsubseteq \neg connectedTo^-$,
(42.) $connectedTo^- \sqsubseteq \neg of^-$,
(43.) $of^- \sqsubseteq \neg connectedTo$,
(44.) $connectedTo \sqsubseteq \neg of^-$,
(45.) $of \sqsubseteq \neg connectedTo^-$,
(46.) $connectedTo^- \sqsubseteq \neg of$,
(47.) $(id\ Port\ number,\ of\ )$,
(48.) $\forall x, y, z.(Port(x) \wedge Port(y) \wedge of(x, z) \wedge of(y, z) \wedge connectedTo(x, y) \rightarrow \bot)$,
(49.) $\forall x, y, z, k, w, v.(PortOut(x) \wedge of(x, y) \wedge connectedTo(x, z) \wedge of(z, k) \wedge PortIn(w) \wedge of(w, y) \wedge connectedTo(w, v) \wedge of(v, k) \rightarrow \bot)$.

Note that, for the sake of brevity, we assume, in this example, that the admitted data types are only `xsd:integer`, `xsd:string`, and `xsd:dateTime`. □

By considering Lemma 16, we observe that in a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ an inconsistency may arise only for one of the following reasons:

1. there exist an atomic concept A in the TBox alphabet $\Gamma_{\mathcal{O}}$, a constant d in the alphabet of constants Γ_C, a TBox assertion $A \sqsubseteq \neg A$ in $cln(\mathcal{T})$, and an ABox assertion α in $\mathcal{A}$ such that $\langle \mathcal{T}_{inc}, \{\alpha\} \rangle \models A(d)$;

2. there exist an atomic role P in the TBox alphabet $\Gamma_{\mathcal{O}}$, a pair of constants d_1 and d_2 in the alphabet of constants Γ_C, a TBox assertion $P \sqsubseteq \neg P$ in $cln(\mathcal{T})$, and an ABox assertion α in $\mathcal{A}$ such that $\langle \mathcal{T}_{inc}, \{\alpha\} \rangle \models P(d_1, d_2)$;

3. there exist an attribute U in the TBox alphabet $\Gamma_{\mathcal{O}}$, a pair of constants d_1 and d_2 in the alphabet of constants Γ_C, a TBox assertion $U \sqsubseteq \neg U$ in $cln(\mathcal{T})$, and an ABox assertion α in $\mathcal{A}$ such that $\langle \mathcal{T}_{inc}, \{\alpha\} \rangle \models U(d_1, d_2)$;

4. there exist an atomic role P in the TBox alphabet $\Gamma_{\mathcal{O}}$, a constant d in the alphabet of constants Γ_C, a TBox assertion $P \sqsubseteq \neg P^-$ in $cln(\mathcal{T})$, and an ABox assertion α in $\mathcal{A}$ such that $\langle \mathcal{T}_{inc}, \{\alpha\} \rangle \models P(d, d)$;

5. there exist an atomic role P in the TBox alphabet $\Gamma_{\mathcal{O}}$, a constant d in the alphabet of constants Γ_C, a TBox assertion $\exists P \sqsubseteq \neg\exists P^-$ in $cln(\mathcal{T})$, and an ABox assertion α in $\mathcal{A}$ such that $\langle \mathcal{T}_{inc}, \{\alpha\} \rangle \models P(d, d)$;

6. there exist an attribute U in the TBox alphabet $\Gamma_{\mathcal{O}}$, a constant d in the alphabet of constants Γ_C, a constant v in the alphabet of values Γ_V, a TBox assertion $\rho(U) \sqsubseteq \neg T_i$ in $cln(\mathcal{T})$, and an ABox assertion α in $\mathcal{A}$ such that $\langle \mathcal{T}_{inc}, \{\alpha\}\rangle \models U(d, v)$ and the interpretation of v belongs to the interpretation set of T_i;

7. there exist an atomic role P in the TBox alphabet $\Gamma_{\mathcal{O}}$, constants d, d_1, d_2, with $d_1 \neq d_2$, in the alphabet of constants Γ_C, a TBox assertion (funct P) (resp. (funct P^-)) in $cln(\mathcal{T})$, and the ABox assertions $P(d, d_1)$ and $P(d, d_2)$ (resp. $P(d_1, d)$ and $P(d_2, d)$) belong to $\mathcal{A}$;

8. there exist an attribute U in the TBox alphabet $\Gamma_{\mathcal{O}}$, constants d, v_1, v_2, with $v_1 \neq v_2$, in the alphabet of constants Γ_C, a TBox assertion (funct U) in $cln(\mathcal{T})$, and the ABox assertions $U(d, v_1)$ and $U(d, v_2)$ belongs to $\mathcal{A}$;

9. there exist a pair of basic concepts B_1 and B_2, a constant d in the alphabet of constants Γ_C, a TBox assertion $B_1 \sqsubseteq \neg B_2$ in $cln(\mathcal{T})$, and a pair of ABox assertions α and β in $\mathcal{A}$ such that $\langle \mathcal{T}_{inc}, \{\alpha\}\rangle \models B_1(d)$ and $\langle \mathcal{T}_{inc}, \{\beta\}\rangle \models B_2(d)$ (for example, $A \sqsubseteq \neg\exists P \in cln(\mathcal{T})$ and $\{A(d), P(d, c)\}$ is a subset of $\mathcal{A}$);

10. there exist a pair of basic roles Q_1 and Q_2, a pair of constants d_1 and d_2 in the alphabet of constants Γ_C, a TBox assertion $Q_1 \sqsubseteq \neg Q_2$ in $cln(\mathcal{T})$, and a pair of ABox assertions α and β in $\mathcal{A}$ such that $\langle \mathcal{T}_{inc}, \{\alpha\}\rangle \models Q_1(d_1, d_2)$ and $\langle \mathcal{T}_{inc}, \{\beta\}\rangle \models Q_2(d_1, d_2)$ (for example, $P \sqsubseteq \neg S^- \in cln(\mathcal{T})$ and $\{P(a, b), S(b, a)\} \subseteq \mathcal{A}$);

11. there exist a pair of attributes U_1 and U_2 in the TBox alphabet $\Gamma_{\mathcal{O}}$, a pair of constants d_1 and d_2 in the alphabet of constants Γ_C, a TBox assertion $U_1 \sqsubseteq \neg U_2$ in $cln(\mathcal{T})$, and a pair of ABox assertions α and β in $\mathcal{A}$ such that $\langle \mathcal{T}_{inc}, \{\alpha\}\rangle \models U_1(d_1, d_2)$ and $\langle \mathcal{T}_{inc}, \{\beta\}\rangle \models U_2(d_1, d_2)$;

12. there exist an identification assertion $\alpha \in cln(\mathcal{T})$ and a set of ABox assertions $V \subseteq \mathcal{A}$ such that $\langle \mathcal{T}_{inc}, V\rangle \models \neg\alpha$, i.e. $\langle \mathcal{T}_{inc}, V\rangle$ implies the negation of α (for example, $\mathcal{T} \models (id\ A\ P \circ S, U)$ and $\{A(o),\ P(o, c),\ S(c, f),\ U(o, d),\ A(o'),\ P(o', e),\ S(e, f),\ U(o', d)\} \subseteq \mathcal{A}$);

13. there exist a denial assertion $\alpha \in cln(\mathcal{T})$ and a set of ABox assertions $V \subseteq \mathcal{A}$ such that $\langle \mathcal{T}_{inc}, V\rangle \models \neg\alpha$, i.e. $\langle \mathcal{T}_{inc}, V\rangle$ implies the negation of α (for example, $\mathcal{T} \models \forall x, y.(A(x), P(x, y) \rightarrow \bot)$ and $\{A(o), P(o, o')\} \subseteq \mathcal{A}$).

Note that, each violation of a TBox axiom listed in cases 1∼6 require the existence of a $\mathcal{T}$-inconsistent set formed by only one ABox assertion. The kinds of violation listed in cases 7∼11 require the existence of a $\mathcal{T}$-inconsistent set formed by only two ABox axioms. The violation of an identification or denial

assertion requires the existence of a $\mathcal{T}$-inconsistent set containing a number of ABox assertions depending on the form of the assertion. Anyway, for violating an identification assertion we need of a $\mathcal{T}$-inconsistent set containing at least two ABox assertions, while a denial assertion can be violated also by only one ABox assertion.

The idea for computing $\mathcal{Q}_{\mathcal{T}}^{unsat}$ consists in identifying a boolean query $q()$ of the form (5.1) for each assertion α in $cln(\mathcal{T})$ such that there exists in the ABox $\mathcal{A}$ a $\mathcal{T}$-inconsistent set V that violates α in $\mathcal{T}$ if and only if the consistent KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle \models q()$. To illustrate the technique, we first define a translation function φ from assertions in $cln(\mathcal{T})$ to boolean queries of the form (5.1).

- $\varphi((\mathsf{funct}\ P))$: $\exists x, x_1, x_2.P(x, x_1) \wedge P(x, x_2) \wedge x_1 \neq x_2$
- $\varphi((\mathsf{funct}\ P^-))$: $\exists x, x_1, x_2.P(x_1, x) \wedge P(x_2, x) \wedge x_1 \neq x_2$
- $\varphi((\mathsf{funct}\ U))$: $\exists x, x_1, x_2.U(x, x_1) \wedge U(x, x_2) \wedge x_1 \neq x_2$
- $\varphi(B_1 \sqsubseteq \neg B_2)$: $\exists x.\gamma(B_1, x) \wedge \gamma(B_2, x)$
- $\varphi(Q_1 \sqsubseteq \neg Q_2)$: $\exists x_1, x_2.\eta(Q_1, x_1, x_2) \wedge \eta(Q_2, x_1, x_2)$
- $\varphi(U_1 \sqsubseteq \neg U_2)$: $\exists x_1, x_2.U_1(x_1, x_2) \wedge U_2(x_1, x_2)$
- $\varphi(\rho(U) \sqsubseteq \neg T_i)$: $\exists x_1, x_2.U(x_1, x_2) \wedge T_i(x_2)$
- $\varphi(\forall \vec{x}.(conj(\vec{t}) \rightarrow \bot))$: $\exists \vec{x}.conj(\vec{t})$
- $\varphi((id\ B\ \pi_1, \ldots, \pi_n))$: $\exists \vec{x}.\gamma(B, x) \wedge \gamma(B, x') \wedge x \neq x' \wedge \bigwedge_{1 \leq i \leq n}(\rho(\pi_i(x, x_i)) \wedge \rho(\pi_i(x', x_i)))$

In the above definition, B is a basic concept, Q is a basic role, U is an attribute, T is a value-domain, t is a term, i.e., either a variable or a constant, and x, x_1, x_2 are variable. Furthermore, let y_{new} be a fresh variable symbols, i.e., a variable symbol not occurring elsewhere in the query, we pose:

$$\gamma(B, x) = \begin{cases} A(x) & \text{if } B = A, \\ P(x, y_{new}) & \text{if } B = \exists P, \\ P(y_{new}, x) & \text{if } B = \exists P^-, \\ U(x, y_{new}) & \text{if } B = \delta(U) \end{cases}$$

and

$$\eta(Q, x_1, x_2) = \begin{cases} P(x_1, x_2) & \text{if } Q = P, \\ P(x_2, x_1) & \text{if } Q = P^-. \end{cases}$$

Moreover, $\rho(\pi(x_1, x_2))$ is inductively defined on the structure of path π as follows:

if $\pi = D_1? \circ \cdots \circ D_h? \circ S \circ D'_1? \circ \cdots \circ D'_k?$ (with $h \geq 0$, $k \geq 0$), then
$\rho(\pi(x_1, x_2)) = \gamma(D_1, x_1) \wedge \cdots \wedge \gamma(D_h, x_1) \wedge S(x_1, x_2) \wedge \gamma(D'_1, x_2) \wedge \cdots \wedge \gamma(D'_k, x_2)$;
if $\pi = D_1? \circ \cdots \circ D_h? \circ U$ (with $h \geq 0$), then

$\rho(\pi(x_1, x_2)) = \gamma(D_1, x_1) \wedge \cdots \wedge \gamma(D_h, x_1) \wedge U(x_1, x_2)$;
if $\pi = \pi_1 \circ \pi_2$, where $\mathit{length}(\pi_1) = 1$ and $\mathit{length}(\pi_2) \geq 1$, then
$\rho(\pi(x_1, x_2)) = \rho(\pi_1(x_1, y_{new})) \wedge \rho(\pi_2(y_{new}, x_2))$

where S denotes an atomic role or the inverse of an atomic role, U denotes an attribute, and D denotes a basic concept.

Intuitively, given a TBox assertion $\alpha \in \mathit{cln}(\mathcal{T})$, the query $\varphi(\alpha)$ encodes the negation of α.

Example 4. Consider the following assertions belonging to the set $\mathit{cln}(\mathcal{T})$, where $\mathit{cln}(\mathcal{T})$ is the set of TBox assertions of Example 3.

(1.) $(funct\ connectedTo)$,
(5.) $(id\ Port\ number,\ of\ )$,
(6.) $\rho(number) \sqsubseteq \neg\texttt{xsd:string}$,
(15.) $\exists connectedTo \sqsubseteq \neg Device$,
(39.) $of \sqsubseteq \neg connectedTo$,
(47.) $\forall x, y, z.(Port(x) \wedge Port(y) \wedge of(x, z) \wedge of(y, z) \wedge connectedTo(x, y) \rightarrow \bot)$,

The queries corresponding to the negation of the above TBox assertions are:

$\varphi(1.) = \exists x, y, z.connectedTo(x, y) \wedge connectedTo(x, z) \wedge y \neq z$;
$\varphi(5.) = \exists x, y, z, k.Port(x) \wedge number(x, y) \wedge of(x, z) \wedge$
$Port(k) \wedge number(k, y) \wedge of(k, z) \wedge x \neq k$;
$\varphi(6.) = \exists x, y.number(x, y) \wedge \texttt{xsd:string}(y)$;
$\varphi(15.) = \exists x, y, z.connectedTo(x, y) \wedge Device(x)$;
$\varphi(39.) = \exists x, y.of(x, y) \wedge connectedTo(x, y)$;
$\varphi(47.) = \exists x, y, z.Port(x) \wedge Port(y) \wedge of(x, z) \wedge of(y, z) \wedge connectedTo(x, y)$.

□

We are now ready to present the algorithm $\mathsf{unsatQueries}(\mathcal{T})$ for computing $\mathcal{Q}_{\mathcal{T}}^{unsat}$.

The goal of $\mathsf{unsatQueries}(\mathcal{T})$ is to compute the set of queries corresponding to the negation of the assertion in $\mathit{cln}(\mathcal{T})$ by means of the translation function φ. In case where the assertion α in $\mathit{cln}(\mathcal{T})$ is an identification assertion or a denial assertion, $\mathsf{unsatQueries}(\mathcal{T})$ firstly computes the query $\varphi(\alpha)$ and then computes the *perfect reformulations* of such query [31]. To this aim, $\mathsf{unsatQueries}(\mathcal{T})$ makes use of the algorithm $\mathsf{PerfectRef}_{\neq}(\varphi(\alpha), \mathcal{T}_{inc})$, which is a slight variation of the algorithm PerfectRef given in [31], taking into account only the inclusion assertions in $\mathcal{T}_{inc}$. In this modified version, inequality is considered as a primitive role thus inequality are never rewritten by the algorithm, and the algorithm does not unify query atoms if this causes a violation of an inequality. Note that the result of $\mathsf{PerfectRef}_{\neq}$ is a union of boolean

Input: a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$
Output: a set of boolean queries $\mathcal{Q}_{\mathcal{T}}^{unsat}$
begin
  $\mathcal{Q}_{\mathcal{T}}^{unsat} \leftarrow \emptyset$;
  foreach $\alpha \in cln(\mathcal{T})$ **do**
    if $\alpha = (\mathsf{funct}\ Q))$ **or**
      $\alpha = (\mathsf{funct}\ U)$ **or**
      $\alpha = B_1 \sqsubseteq \neg B_2$ **or**
      $\alpha = Q_1 \sqsubseteq \neg Q_2$ **or**
      $\alpha = U_1 \sqsubseteq \neg U_2$ **or**
      $\alpha = \rho(U) \sqsubseteq \neg T_i$
    then $\mathcal{Q}_{\mathcal{T}}^{unsat} \leftarrow \mathcal{Q}_{\mathcal{T}}^{unsat} \cup \{\varphi(\alpha)\}$;
    if $\alpha = (id\ B\ \pi_1, \ldots, \pi_n)$ **or**
      $\alpha = (\forall \vec{x}.(conj(\vec{t})))$
    then $\mathcal{Q}_{\mathcal{T}}^{unsat} \leftarrow \mathcal{Q}_{\mathcal{T}}^{unsat} \cup \mathsf{PerfectRef}_{\neq}(\varphi(\alpha), \mathcal{T}_{inc})$;
  return $\mathcal{Q}_{\mathcal{T}}^{unsat}$
end

Algorithm 3: unsatQueries($\mathcal{T}$)

queries of the form (5.1), represented as a set of queries, as usual. We point out that if $\mathcal{T}_{id} \cup \mathcal{T}_{den} = \emptyset$, as for *DL-Lite*$_A$-KBs, the reformulation step in the algorithm is unnecessary [31]. Indeed, every negative inclusion assertion and inclusion between value-domains entailed by the TBox $\mathcal{T}$ belongs to $cln(\mathcal{T})$. Hence, in these cases the call to the PerfectRef$_{\neq}$ algorithm in unsatQueries is not necessary. On the other hand, by handling adequately the implicit constraints, following from the semantics of $\mathcal{T}$, namely that, for every T_i and T_j of value-domains, T_i and T_j are disjoint, and by exploiting the crucial task of perfect reformulation that is carried out by PerfectRef$_{\neq}$, it is possible to compute $\mathcal{Q}_{\mathcal{T}}^{unsat}$ without first computing $cln(\mathcal{T})$. For more details we refer the reader to [94, 95].

The following lemma shows that the algorithm unsatQueries terminates, when applied to a *DL-Lite*$_{A,id,den}$ TBox.

Lemma 9. *Let $\mathcal{T}$ be a DL-Lite$_{A,id,den}$ TBox. Then, unsatQueries($\mathcal{T}$) terminates.*

Proof. Termination of unsatQueries($\mathcal{T}$) directly follows from the finiteness of the set $\mathcal{T}$ and from the termination of the algorithm PerfectRef$_{\neq}$ [31]. ∎

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_{A,id,den}$-KB. Since $\mathcal{K}$ is consistent if and only if there are no $\mathcal{T}$-inconsistent sets in $\mathcal{A}$, the following theorem states that the query produced by the algorithm unsatQueries($\mathcal{T}$) can be used to check satisfiability of $\mathcal{K}$.

Theorem 6. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB. A $\mathcal{T}$-inconsistent set $V \subseteq \mathcal{A}$ exists if and only if $\langle \emptyset, \mathcal{A} \rangle \models \mathsf{unsatQueries}(\mathcal{T})$.*

Proof. A $\mathcal{T}$-inconsistent set $V \subseteq \mathcal{A}$ exists if and only if $\mathcal{K}$ is inconsistent. Lemma 8 states that $\mathcal{K}$ is inconsistent if and only if there exists an assertion $\alpha \in \mathcal{T} \setminus \mathcal{T}_{inc}$ such that $can(\mathcal{K}) \models \neg\alpha$. Hence, to prove the theorem we have to show that there exists an assertion $\alpha \in \mathcal{T} \setminus \mathcal{T}_{inc}$ such that $can(\mathcal{K}) \models \neg\alpha$ if and only if there exists a query q in $\mathsf{unsatQueries}(\mathcal{T})$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$. If $\alpha \in \mathcal{T} \setminus (\mathcal{T}_{inc} \cup \mathcal{T}_{den})$, the proof directly follows from the results given in [31, 32]. Let $\alpha \in \mathcal{T}_{den}$. In this case, the algorithm $\mathsf{unsatQueries}$ adds to $\mathcal{Q}_{\mathcal{T}}^{unsat}$ the queries $\mathsf{PerfectRef}_{\neq}(\varphi(\alpha), \mathcal{T}_{inc})$. Since $can(\mathcal{K})$ depends only on the ABox $\mathcal{A}$ and the positive TBox assertion in $\mathcal{T}_{inc}$, we have that $can(\mathcal{K}) = can(\langle \mathcal{T}_{inc}, \mathcal{A} \rangle)$. From Corollary 2 and Lemma 3, it follows that $can(\mathcal{K}) \models \neg\alpha$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models \mathsf{PerfectRef}_{\neq}(\varphi(\alpha), \mathcal{T}_{inc})$. Hence, we can conclude that there exists a $\mathcal{T}$-inconsistent set $V \subseteq \mathcal{A}$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models \mathsf{unsatQueries}(\mathcal{T})$. ∎

In other terms, it is possible to check the satisfiability of $\mathcal{K}$ by evaluating the union of queries $\mathsf{unsatQueries}(\mathcal{T})$ over the ABox $\mathcal{A}$ considered as a relational database.

Next, we present the algorithm $\mathsf{Satisfiable}_{DA}$ for checking satisfiability of a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

Input: a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: *true* or *false*
begin
  $\mathcal{Q}_{\mathcal{T}}^{unsat} \leftarrow \mathsf{unsatQueries}(\mathcal{T})$;
  foreach $q \in \mathcal{Q}_{\mathcal{T}}^{unsat}$ **do**
    if $q^{DB(\mathcal{A})} \neq \emptyset$
    then return *false*;
  return *true*
end

Algorithm 4: $\mathsf{Satisfiable}_{DA}(\mathcal{K})$

Firstly, the algorithm uses the algorithm $\mathsf{unsatQueries}(\mathcal{T})$ for computing the set of queries $\mathcal{Q}_{\mathcal{T}}^{unsat}$. Then, it evaluates every query $q \in \mathcal{Q}_{\mathcal{T}}^{unsat}$ over $\langle \emptyset, \mathcal{A} \rangle$. If every query in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ evaluates to *false* over the interpretation $DB(\mathcal{A})$, then the algorithm return *true*, i.e., the KB $\mathcal{K}$ is consistent, since the ABox does not contain any $\mathcal{T}$-inconsistent set that violates an assertion implied by $\mathcal{T}$. Otherwise, the algorithm returns *false*, i.e., the KB is inconsistent.

The following lemmas establish termination and correctness of Algorithm 4.

Lemma 10. *Let $\mathcal{K}$ be a DL-Lite$_{A,id,den}$-KB. Then $\mathsf{Satisfiable}_{DA}(\mathcal{K})$ terminates.*

Proof. Termination of $\mathsf{Satisfiable}_{DA}(\mathcal{K})$ follows directly from the termination of $\mathsf{unsatQueries}(\mathcal{T})$ and of evaluation of FOL-query over databases. ∎

Lemma 11. *Let $\mathcal{K}$ be a DL-Lite$_{A,id,den}$-KB. $\mathcal{K}$ is consistent if and only if $\mathsf{Satisfiable}_{DA}(\mathcal{K})$ returns* true*.*

Proof. The correctness of $\mathsf{Satisfiable}_{DA}(\mathcal{K})$ is a direct consequence of Theorem 6. ∎

The following theorem gives the computational characterization for the KB satisfiability problem in *DL-Lite*$_{A,id,den}$. We recall that answering UCQ, in *DL-Lite* is NP-complete with respect to the size of the query (cf. Theorem 4).

Theorem 7. *In DL-Lite$_{A,id,den}$ KB satisfiability is in NP in the size of the whole KB, and in AC^0 in the size if the ABox.*

Proof. By exploiting the results given in Theorem 6 and in Theorem 11, we have that KB satisfiability in *DL-Lite*$_{A,id,den}$ can be checked by performing the following steps. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_{A,id,den}$-KB. We can check if $\mathcal{K}$ is consistent by:

1. computing the set of TBox assertions $cln(\mathcal{T})$;
2. computing the set of queries $\mathcal{Q}_{\mathcal{T}}^{unsat}$ by adding to $\mathcal{Q}_{\mathcal{T}}^{unsat}$,
   (a) for each assertion α in $cln(\mathcal{T})$ that is neither an identification assertion nor a denial assertion, the query $\varphi(\alpha)$, and
   (b) for each assertion α $cln(\mathcal{T})$ that is an identification assertion or a denial assertion, the queries obtained by computing the perfect reformulation of $\varphi(\alpha)$ by means of the algorithm $\mathsf{PerfectRef}_{\neq}$;
3. evaluating each query $q \in \mathcal{Q}_{\mathcal{T}}^{unsat}$ over the KB $\langle \emptyset, \mathcal{A} \rangle$.

We first prove that, in *DL-Lite*$_{A,id,den}$, KB satisfiability is in AC^0 in the size of the ABox (data complexity). Since prefect reformulation is independent of the ABox [31], by observing the four steps above, it is clear that the size of the ABox comes into play only in the last step, i,e., in the evaluation of the queries in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ over $\langle \emptyset, \mathcal{A} \rangle$. This task can be performed in AC^0 with respect to the size of $\mathcal{A}$ [1, 107]. Hence, we can conclude that KB satisfiability is in AC^0 in the size of the ABox.

We now prove that KB satisfiability in *DL-Lite*$_{A,id,den}$ is in NP with respect to the size of the TBox. Let δ be a denial assertion in $\mathcal{T}_{den}$, clearly it belong to $cln(\mathcal{T})$. Since no restrictions are imposed on the definition of a denial assertion α, we have that $\varphi(\alpha)$ is a generic conjunctive query. Step 2 and 3 above require to evaluate $\varphi(\delta)$ over the KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$. Hence, KB satisfiability

in *DL-Lite*$_{A,id,den}$ has the same complexity of answering CQs over the *DL-Lite*-KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$ with respect to combined complexity (i.e., with respect to the size of the whole KB and the size of the query). Finally, from Theorem 4 we have that answering CQs over $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$ is in NP. ∎

We conclude this section with an example illustrating the whole procedure for checking satisfiability of a *DL-Lite*$_{A,id,den}$-KB.

Example 5. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_{A,id,den}$-KB, where $\mathcal{T}$ is the TBox of Example 2, and $\mathcal{A}$ is the ABox formed by the following assertions:

$$PortIn(p_1), \quad PortOut(p_1), \quad connectedTo(p_1, p_2).$$

In words, $\mathcal{A}$ states that p_1 is both an incoming port and an outgoing port, and that p_1 is connected to a port p_2.

It is immediate to verify that $\mathcal{K}$ is inconsistent since the assertions $PortIn(p_1)$ and $PortOut(p_2)$ violates the assertion in $\mathcal{T}$ stating that a port cannot be an incoming port and an outgoing port at the same time.

The first step of algorithm $\mathsf{Satisfiable}_{DA}(\mathcal{K})$ is to compute the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ by means of $\mathsf{unsatQueries}(\mathcal{T})$. To this aim, $\mathsf{unsatQueries}(\mathcal{T})$ first computes the set $cln(\mathcal{T})$. The result is shown in Example 3.

For this example, let us focus only on the following two assertions in $\mathcal{T}$.

(7.) $PortIn \sqsubseteq \neg PortOut$;
(48.) $\forall x, y, z, k, w, v.(PortOut(x) \land of(x, y) \land connectedTo(x, z) \land of(z, k) \land PortIn(w) \land of(w, y) \land connectedTo(w, v) \land of(v, k) \rightarrow \bot)$.

For each identification assertion or denial assertion α in $cln(\mathcal{T})$, the algorithm $\mathsf{unsatQueries}(\mathcal{T})$ computes the perfect reformulation with respect to $\mathcal{T}_{inc}$ of the query $\varphi(\alpha)$, which encodes the violation of α, and adds the resulting queries to $\mathcal{Q}_{\mathcal{T}}^{unsat}$. If the assertion is not an identification assertion or an denial assertions, the algorithm just adds the query $\varphi(\alpha)$ to $\mathcal{Q}_{\mathcal{T}}^{unsat}$. For the assertions (7.) and (48.) the queries corresponding to their negation are:

$\varphi(7.) = \exists x.PortIn(x) \land PortOut(x)$;
$\varphi(48.) = \exists x, y, z, k, w, v.PortOut(x) \land of(x, y) \land connectedTo(x, z) \land of(z, k) \land PortIn(w) \land of(w, y) \land connectedTo(w, v) \land of(v, k)$.

The algorithm $\mathsf{PerfectRef}_{\neq}$, when applied to the query $\varphi(48.)$ returns:

$q_1() = \exists x, y, z, k, w, v.PortOut(x) \land of(x, y) \land connectedTo(x, z) \land of(z, k) \land PortIn(w) \land of(w, y) \land connectedTo(w, v) \land of(v, k)$;
$q_2() = \exists x, y, z, k.PortOut(x) \land of(x, y) \land connectedTo(x, k) \land PortIn(z) \land of(z, y) \land connectedTo(z, k)$;
$q_3() = \exists x, y.PortIn(x) \land PortOut(x) \land connectedTo(x, y)$.

Hence, $\mathcal{Q}_{\mathcal{T}}^{unsat}$ contains the following queries:

$q_1() = \exists x, y, z, k, w, v.PortOut(x) \wedge of(x, y) \wedge connectedTo(x, z) \wedge$
$of(z, k) \wedge PortIn(w) \wedge of(w, y) \wedge connectedTo(w, v) \wedge of(v, k);$
$q_2() = \exists x, y, z, k.PortOut(x) \wedge of(x, y) \wedge connectedTo(x, k) \wedge$
$PortIn(z) \wedge of(z, y) \wedge connectedTo(z, k);$
$q_3() = \exists x, y.PortIn(x) \wedge PortOut(x) \wedge connectedTo(x, y);$
$q_4() = \exists x.PortIn(x) \wedge PortOut(x).$

Finally, for each query q in $\mathcal{Q}_{\mathcal{T}}^{unsat}$, algorithm $\mathsf{Satisfiable}_{DA}(\mathcal{K})$ checks if $\langle \emptyset, \mathcal{A} \rangle \models q$. It is easy to verify that, for the queries considered in this example, we have:

$$\langle \emptyset, \mathcal{A} \rangle \not\models q_1; \qquad \langle \emptyset, \mathcal{A} \rangle \not\models q_2;$$
$$\langle \emptyset, \mathcal{A} \rangle \models q_3; \qquad \langle \emptyset, \mathcal{A} \rangle \models q_4.$$

Hence, it is possible to conclude that the KB $\mathcal{K}$ is inconsistent since the assertions in $\mathcal{A}$ violate both the disjunction $PortIn \sqsubseteq \neg PortOut$ and the denial assertion $\forall x, y, z, k, w, v.(PortOut(x) \wedge of(x, y) \wedge connectedTo(x, z) \wedge of(z, k) \wedge PortIn(w) \wedge of(w, y) \wedge connectedTo(w, v) \wedge of(v, k) \rightarrow \bot)$ in $\mathcal{T}$. □

5.2 Query answering in *DL-Lite*$_{A,id,den}$

In this section we study query answering in *DL-Lite*$_{A,id,den}$. To this aim we first establish a fundamental "separation" property for denial assertions in *DL-Lite*$_{A,id,den}$. In other words, we show that denial assertion does not affect query answering, and thus it is possible to ignore denial assertions in query answering process.

Theorem 8. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent DL-Lite$_{A,id,den}$-KB and q be a UCQ. Then $\langle \mathcal{T}, \mathcal{A} \rangle \models q$ if and only if $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle \models q$.*

Proof.
($\Rightarrow$) We have to prove that if $\langle \mathcal{T}, \mathcal{A} \rangle \models q$, then $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle \models q$. Suppose that q is formed by only one conjunctive queries. Since $\langle \mathcal{T}, \mathcal{A} \rangle \models q$, then the KB obtained from $\mathcal{K}$ by adding to $\mathcal{T}$ the denial assertion $\neg q$ is inconsistent, i.e., $\langle \mathcal{T} \cup \{\neg q\}, \mathcal{A} \rangle$ is inconsistent. Lemma 7 states that $can(\langle \mathcal{T} \cup \{\neg q\}, \mathcal{A} \rangle)$ is models of $\langle \mathcal{T} \cup \{\neg q\}, \mathcal{A} \rangle$ if and only if $\langle \mathcal{T} \cup \{\neg q\}, \mathcal{A} \rangle$ is consistent. Thus, $can(\langle \mathcal{T} \cup \{\neg q\}, \mathcal{A} \rangle)$ is not a model for $\langle \mathcal{T} \cup \{\neg q\}, \mathcal{A} \rangle$. From Lemma 8, it follows that there is an assertion $\alpha \in \mathcal{T} \setminus \mathcal{T}_{inc}$ such that $can(\langle \mathcal{T} \cup \{\neg q\}, \mathcal{A} \rangle) \models \neg\alpha$. Since the assertions in $\mathcal{T} \setminus \mathcal{T}_{inc}$ have no role in constructing $chase(\langle \mathcal{T} \cup \cup \{\neg q\}, \mathcal{A} \rangle)$, we have that $chase(\langle \mathcal{T} \cup \{\neg q\}, \mathcal{A} \rangle) = chase(\langle \mathcal{T}_{inc}, \mathcal{A} \rangle)$, and then, $can(\langle \mathcal{T} \cup \{\neg q\}, \mathcal{A} \rangle) = can(\langle \mathcal{T}_{inc}, \mathcal{A} \rangle)$. This means that there is an assertion $\alpha \in \mathcal{T} \setminus \mathcal{T}_{inc}$ such that $can(\langle \mathcal{T}_{inc}, \mathcal{A} \rangle) \models \neg\alpha$. Since $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent, by

exploiting Lemma 8, we can conclude that $\alpha = \neg q$. Hence, $can(\langle \mathcal{T}_{inc}, \mathcal{A} \rangle) \models q$. Finally, from Corollary 2, it follows that $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle \models q$.
($\Leftarrow$) Trivially, if $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle \models q$, then $\langle \mathcal{T}, \mathcal{A} \rangle \models q$, hence the claim follows. ∎

From Theorem 8, we can conclude that for answering union of conjunctive queries over a *DL-Lite*$_{A,id,den}$-KB, we can exploits algorithm Answer presented in Section 3.2 for answering union of conjunctive queries over *DL-Lite*-KBs, with the only difference that now satisfiability check is done by taking in account also denial assertions.

We are now ready to present the algorithm Answer$_{DA}$ for answering union of conjunctive queries over *DL-Lite*$_{A,id,den}$-KBs.

Input: a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and an UCQ q
Output: $cert(q, \mathcal{K})$
begin
  if not Satisfiable$_{DA}(\mathcal{K})$
  then return $AllTup(q, \mathcal{K})$;
  else return Answer$(q, \mathcal{K})$;
end

Algorithm 5: Answer$_{DA}(q, \mathcal{K})$

Algorithm Answer$_{DA}(q, \mathcal{K})$ takes in input a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K}$ and a union of conjunctive query q and, if $\mathcal{K}$ is consistent, it returns the certain answer to q over $\mathcal{K}$. If $\mathcal{K}$ is not consistent, Answer$_{DA}(q, \mathcal{K})$ returns $AllTup(q, \mathcal{K})$ that is the finite set of all possible tuples of constants whose arity is the one of q. With the aim of checking satisfiability of $\mathcal{K}$, Answer$_{DA}(q, \mathcal{K})$ makes use of the algorithm Satisfiable$_{DA}(\mathcal{K})$.

Next, we deal with termination and correctness of Algorithm 5.

Lemma 12. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB, and let q be a union of conjunctive queries. Then Answer$_{DA}(q, \mathcal{K})$ terminates.*

Proof. Termination of Answer$_{DA}(q, \mathcal{K})$ follows directly from the termination of Satisfiable$_{DA}(\mathcal{K})$ and Answer$(q, \mathcal{K})$ [25], and from the fact that both $\mathcal{T}$ and $\mathcal{A}$ are finite sets of assertions. ∎

The following lemma states that we can use Answer$_{DA}(q, \mathcal{K})$ for computing the certain answers to a union conjunctive queries posed over a *DL-Lite*$_{A,id,den}$-KB.

Lemma 13. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB, and let q be a union of conjunctive queries. Then $cert(q, \mathcal{K}) =$ Answer$_{DA}(q, \mathcal{K})$.*

Proof. The proof follows directly from Theorem 8 and from the correctness of the algorithms Satisfiable$_{DA}(\mathcal{K})$ and Answer$(q, \mathcal{K})$. ∎

A notable consequence of the lemmas above, is that, as for the other DLs of the *DL-Lite* family, the problem of answering union of conjunctive queries over consistent *DL-Lite*$_{A,id,den}$-KBs is FOL-rewritable. We are also able to establish the complexity of answering UCQs in *DL-Lite*$_{A,id,den}$ in combined complexity.

Theorem 9. *Answering UCQs in DL-Lite*$_{A,id,den}$ *is in* AC^0 *in the size of the ABox (data complexity), in NP in the size of the KB, and in NP in combined complexity.*

Proof. The proof follows from the following results:

- for computing the certain answers of a union of conjunctive queries over a consistent *DL-Lite*$_{A,id,den}$-KB $\langle \mathcal{T}, \mathcal{A} \rangle$ we can refer to the *DL-Lite*-KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$.
- Theorem 7 states that KB satisfiability in *DL-Lite*$_{A,id,den}$ is in AC^0 in data complexity.

The above observations prove that the problem is in AC^0 in data complexity. Moreover, we have:

- Theorem 7 states that KB satisfiability in *DL-Lite*$_{A,id,den}$ is in NP with respect to the size of $\mathcal{K}$.

Hence, the problem is in NP with respect to the size of the KB. Finally, we have that answering CQs over a databases is NP-complete in the size of the query [1]. Therefore, we can conclude that the problem is in NP also in combined complexity. ∎

We conclude this section by observing that adding denial assertions to *DL-Lite*$_A$-KBs does not increase the data complexity for both the satisfiability problem and query answering. Unfortunately, our results show that the presence of denial assertions leads the complexity of both problems to became intractable with respect to the size of the KB, in particular with respect to the size of $\mathcal{T}_{den}$.

5.3 Properties of *DL-Lite*$_{A,id,den}$

In this section, we present some results and algorithms for *DL-Lite*$_{A,id,den}$-KBs, which are useful for the rest of this work.

In what follows, we show how, by exploiting the results given in the previous sections, it is possible to single out those assertions in the ABox of an inconsistent *DL-Lite*$_{A,id,den}$-KB that lead to violate assertions in the TBox.

As mentioned in Section 2.5, a boolean query q is satisfied by an ABox $\mathcal{A}$ (denoted by $\langle \emptyset, \mathcal{A} \rangle \models q$) if there exists a substitution σ from the variables in q to

constants in $\mathcal{A}$ such that the formula $\sigma(q)$ is satisfied in the FOL-interpretation $DB(\mathcal{A})$ (cf. Definition 3). Moreover, when the query q is satisfied by $\mathcal{A}$, we denote by $\mathsf{images}(q, \mathcal{A})$ the set constituted by all the images of q in $\mathcal{A}$.

Now, let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an inconsistent *DL-Lite*$_{A,id,den}$-KB. Theorem 6 guarantees that $\langle \emptyset, \mathcal{A} \rangle \models \mathsf{unsatQueries}(\mathcal{T})$, and then there exists a query $q \in \mathsf{unsatQueries}(\mathcal{T})$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$. It follows that the set $\mathsf{images}(q, \mathcal{A})$ is non-empty. Intuitively, a set $V \in \mathsf{images}(q, \mathcal{A})$ contains those ABox assertions belonging to $\mathcal{A}$ which lead to violate a TBox assertion in $cln(\mathcal{T})$. We name the set V a $\mathcal{K}$-*clash*. More formally:

Definition 15. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be *DL-Lite*$_{A,id,den}$-KB such that $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{\mathcal{T}}^{unsat}$. We say that a non-empty set of ABox assertions V is a $\mathcal{K}$-*clash* if there exists in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ a query q such that $V \in \mathsf{images}(q, \mathcal{A})$.

Intuitively, given an inconsistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, the set of $\mathcal{K}$-*clashes* may represent the base from which one can build every $\mathcal{T}$-inconsistence set in $\mathcal{A}$. Indeed, we have the following.

Lemma 14. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB such that $Mod(\langle \mathcal{T}, \mathcal{A} \rangle) = \emptyset$. A set $V \subseteq \mathcal{A}$ is a $\mathcal{T}$-inconsistent set if and only if there exists a $\mathcal{K}$-clash V' such that $V' \subseteq V$.*

Proof.
($\Rightarrow$) Let V be a $\mathcal{T}$-inconsistent set. There exists a $\mathcal{K}$-*clash* V' such that $V' \subseteq V$. Since $\langle \mathcal{T}, V \rangle$ is inconsistent, it follows from Theorem 6 that there exists in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ a query q such that $\langle \emptyset, V \rangle \models q$. This means that $\mathsf{images}(q, V)$ is non-empty. Since for every set $V' \in \mathsf{images}(q, V)$ we have that $V' \subset V$, it follows that there exists a $\mathcal{K}$-*clash* $\subseteq V$.
($\Leftarrow$) Let V' be a $\mathcal{K}$-*clash*, then there exists a $\mathcal{T}$-inconsistent set V such that $V' \subseteq V$. Since V' is a $\mathcal{K}$-*clash*, then there exists a query $q \in \mathcal{Q}_{\mathcal{T}}^{unsat}$ such that $\langle \emptyset, V' \rangle \models q$. It follows from Theorem 6 that V' is a $\mathcal{T}$-inconsistent set, and then the claim follows. ∎

Example 6. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_{A,id,den}$-KB where:

$$\mathcal{T} = \{C_1 \sqsubseteq \neg\exists P_1, \quad C_2 \sqsubseteq C_1\}$$
$$\mathcal{A} = \{C_2(a), \quad C_1(b), \quad P_1(a, c), \quad P_1(d, e)\}$$

In this example, we have that:

- $cln(\mathcal{T})$ is as follows:

$$cln(\mathcal{T}) = \{\ C_1 \sqsubseteq \neg\exists P_1; \quad \exists P_1 \sqsubseteq \neg C_1$$
$$C_2 \sqsubseteq \neg\exists P_1; \quad \exists P_1 \sqsubseteq \neg C_2\ \};$$

- $\mathcal{Q}_{\mathcal{T}}^{unsat}$ contains the following queries:

$$q_1 = \exists x, y.C_1(x) \wedge P_1(x, y);$$
$$q_2 = \exists x, y.C_2(x) \wedge P_1(x, y).$$

It is easy to verify that the only $\mathcal{K}$-*clash* is the set of assertions:

$$V_1 = \{C_2(a), \quad P_1(a, c)\}.$$

Moreover, it easy to verify that the only $\mathcal{T}$-inconsistent sets in $\mathcal{A}$ are:

$$V_1 = \{C_2(a), \quad P_1(a, c)\};$$
$$V_2 = \{C_2(a), \quad P_1(a, c), \quad C_1(b)\};$$
$$V_3 = \{C_2(a), \quad P_1(a, c), \quad P_1(d, e)\};$$
$$V_4 = \{C_2(a), \quad P_1(a, c), \quad C_1(b), \quad P_1(d, e)\}.$$

Note that, for every $\mathcal{T}$-inconsistent set V in $\mathcal{A}$,we have that $V_1 \subseteq V$. Moreover, we note that $V_4 = \mathcal{A}$, which means that the whole ABox $\mathcal{A}$ is a $\mathcal{T}$-inconsistent set. □

Given an inconsistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, computing the set containing every $\mathcal{T}$-inconsistent set in $\mathcal{A}$ may be computationally costly. For this reason, we now presents the algorithm InconsistentSets that takes as input a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ expressed in *DL-Lite*$_{A,id,den}$, and computes a set containing all the $\mathcal{K}$-*clashes*. In fact, as Lemma 14 shows, for every $\mathcal{T}$-inconsistent set V', which is also a $\mathcal{K}$-*clash*, there exists a $\mathcal{T}$-inconsistent set V such that $V' \subseteq V$. Therefore, in that cases in which one needs to refer to the set constituted by the $\mathcal{T}$-inconsistent sets in $\mathcal{A}$, the set containing the $\mathcal{K}$-*clash* constitutes an excellent candidate for replacing it. The algorithm InconsistentSets will be used intensively in the rest of this work.

Before presenting the algorithm, we need to introduce some preliminary notions.

Let $q()$ be a boolean query of the form (5.1), and let $\vec{x}$ be all the existential variable occurring in $q()$. We denote with $q(\vec{x})$ the query obtained by transforming all existential variables in $q()$ into distinguished variables. For example, consider the following boolean query:

$$q() = \exists x, y, z.A(x) \wedge R(x, y) \wedge B(y) \wedge A(z) \wedge R(z, y) \wedge B(y).$$

With $q(x, y, z)$ we refer to the query:

$$q(x, y, z) = A(x) \wedge R(x, y) \wedge B(y) \wedge A(z) \wedge R(z, y) \wedge B(y).$$

Moreover, let $q()$ be a boolean query of the form (5.1) over a TBox $\mathcal{T}$, and let $q(\vec{x})$ denote the non-boolean version of $q()$. Let $\vec{t}$ be a tuple of constants

with the same arity of $\vec{x}$. We denote by $\mathsf{facts}(q(\vec{x}), \vec{t})$ the set of ABox assertions built over the concept, role, and attribute names occurring in q by replacing the variables $\vec{x}$ in the body of q with the tuple $\vec{t}$. For instance, consider the following query:

$$q() = \exists x, y, z.pers(x) \wedge name(x, y) \wedge surname(x, z) \wedge z \neq \text{`white'}.$$

and the tuple of constants $\langle$'obj-p', 'bob', 'white'$\rangle$. We have that:

$$\mathsf{facts}(q(\vec{x}), \langle\text{'obj-p'}, \text{'bob'}, \text{'white'}\rangle) = \{pers(obj\text{-}p), name(obj\text{-}p, bob), surname(obj\text{-}p, white\}.$$

We are now ready to present the algorithm $\mathsf{InconsistentSets}(\langle\mathcal{T}, \mathcal{A}\rangle)$. We first recall that, given a query $q(\vec{x})$, $q^{DB(\mathcal{A})}$ denote the answer to $q(\vec{x})$ over the interpretation $DB(\mathcal{A})$, and that every tuple $\vec{t} \in q^{DB(\mathcal{A})}$ has the same cardinality of $\vec{x}$ and is formed by constants occurring in $\mathcal{A}$.

Input: *DL-Lite*$_{A,id,den}$-KB $\langle\mathcal{T}, \mathcal{A}\rangle$
Output: a set containing sets of ABox assertions
begin
  $W \leftarrow \emptyset$;
  $\mathcal{Q}_{\mathcal{T}}^{unsat} \leftarrow \mathsf{unsatQueries}(\mathcal{T})$;
  foreach $q() \in \mathcal{Q}_{\mathcal{T}}^{unsat}$ **do**
    foreach $\vec{t} \in q^{DB(\mathcal{A})}$ **do**
      $W \leftarrow W \cup \{\mathsf{facts}(q(\vec{x}), \vec{t})\}$;
  return W;
end

Algorithm 6: $\mathsf{InconsistentSets}(\langle\mathcal{T}, \mathcal{A}\rangle)$

Informally, $\mathsf{InconsistentSets}(\langle\mathcal{T}, \mathcal{A}\rangle)$ first computes the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ by means of the algorithm $\mathsf{unsatQueries}(\mathcal{T})$, then, for each boolean query q in $\mathcal{Q}_{\mathcal{T}}^{unsat}$, it computes the answers to $q(\vec{x})$ over the KB $\langle\emptyset, \mathcal{A}\rangle$ by evaluating q over the ABox $\mathcal{A}$ considered as a relational databases, i.e., over $DB(\mathcal{A})$. Finally, by using the answers to $q(\vec{x})$, the algorithm builds those sets of assertions in $\mathcal{A}$ that constitute the images of the query q in $\mathcal{A}$.

Lemma 15. *Let $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ be a possibly inconsistent DL-Lite$_{A,id}$-KB. Then $\mathsf{InconsistentSets}(\mathcal{K})$ terminates.*

Proof. Termination follows directly from the termination of the algorithm $\mathsf{unsatQueries}(\mathcal{T})$ and of evaluation of FOL-query over databases. ∎

The following theorem states that every set V in $\mathsf{InconsistentSets}(\mathcal{K})$ is actually a $\mathcal{K}$-*clash*, and that no set of ABox assertions, other than those contained in $\mathsf{InconsistentSets}(\mathcal{K})$, is a $\mathcal{K}$-*clash*.

Theorem 10. *Let $\mathcal{K}$ be a possibly inconsistent DL-Lite$_{A,id}$-KB. A set of ABox assertions V is a $\mathcal{K}$-clash if and only if $V \in$ InconsistentSets$(\mathcal{K})$.*

Proof. The proof follows directly from Theorem 6. ∎

The following example aims to clarify how InconsistentSets$(\mathcal{K})$ computes the *$\mathcal{K}$-clashes* relative to an inconsistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K}$.

Example 7. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be the inconsistent *DL-Lite*$_{A,id,den}$-KB of Example 5. The first step InconsistentSets$(\mathcal{K})$ is computing the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ by means of unsatQueries$(\mathcal{T})$. As shown in Example 5, the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ contains, among the others, also the following queries:

$$
\begin{aligned}
q_1() &= \exists x, y, z, k, w, v.PortOut(x) \wedge of(x, y) \wedge connectedTo(x, z) \wedge \\
&\quad of(z, k) \wedge PortIn(w) \wedge of(w, y) \wedge connectedTo(w, v) \wedge of(v, k); \\
q_2() &= \exists x, y, z, k.PortOut(x) \wedge of(x, y) \wedge connectedTo(x, k) \wedge \\
&\quad PortIn(z) \wedge of(z, y) \wedge connectedTo(z, k); \\
q_3() &= \exists x, y.PortIn(x) \wedge PortOut(x) \wedge connectedTo(x, y); \\
q_4() &= \exists x.PortIn(x) \wedge PortOut(x);.
\end{aligned}
$$

We recall that the query $q_4()$ is built starting from the disjointness $PortIn \sqsubseteq \neg PortOut$, and that the remaining queries are built starting from the denial assertion $\forall x, y, z, k, w, v.(PortOut(x) \wedge of(x, y) \wedge connectedTo(x, z) \wedge of(z, k) \wedge PortIn(w) \wedge of(w, y) \wedge connectedTo(w, v) \wedge of(v, k) \rightarrow \bot)$.

The second step of InconsistentSets$(\mathcal{K})$ is evaluating $q(\vec{x})$ for each query $q()$ in $\mathcal{Q}_{\mathcal{T}}^{unsat}$. Since the interpretation $DB(\mathcal{A})$ is

$$
\begin{aligned}
PortIn^{DB(\mathcal{A})} &= \{p_1^{DB(\mathcal{A})}\}, \\
PortOut^{DB(\mathcal{A})} &= \{p_1^{DB(\mathcal{A})}\}, \\
connectedTo^{DB(\mathcal{A})} &= \{(p_1^{DB(\mathcal{A})}, p_2^{DB(\mathcal{A})})\},
\end{aligned}
$$

for the queries considered in this example we have:

$$
\begin{aligned}
q_1(x, y, z, k, w, v)^{DB(\mathcal{A})} &= \emptyset \\
q_2(x, y, z, k)^{DB(\mathcal{A})} &= \emptyset \\
q_3(x, y)^{DB(\mathcal{A})} &= \{\langle p_1, p_2 \rangle\} \\
q_4(x)^{DB(\mathcal{A})} &= \{\langle p_1 \rangle\}
\end{aligned}
$$

Finally, by using the so obtained tuples, the algorithm builds the following *$\mathcal{K}$-clashes*.

$$
\begin{aligned}
\mathsf{facts}(q_1(x), \langle p_1 \rangle) &= \{PortIn(p_1), PortOut(p_1)\}; \\
\mathsf{facts}(q_4(x, y), \langle p_1, p_2 \rangle) &= \{PortIn(p_1), PortOut(p_1), connectedTo(p_1, p_2)\}.
\end{aligned}
$$

□

We conclude this section by giving a notable properties of *DL-Lite*$_{A,id,den}$. From Lemma 7, it it follows that both Lemma 6 and Theorem 5 also hold for *DL-Lite*$_{A,id,den}$. Thus, we have the following lemma.

Lemma 16. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent DL-Lite$_{A,id,den}$-KB, and let α be an ABox assertion. If $\mathcal{K} \models \alpha$, then there exists in $\mathcal{A}$ an ABox assertion α' such that $\langle \mathcal{T}, \{\alpha'\} \rangle \models \alpha$.*

Proof. The proof is similar to the proof of Theorem 5 given in in [40], and follows from Lemma 7. ∎

Part III

Updating Consistent Knowledge Bases

Chapter 6

Updating consistent Description Logic KBs

There are two types of change operators for a knowledge base, corresponding to inserting, and deleting pieces of knowledge, respectively. In the case of insertion into a consistent KB, the aim is to incorporate new knowledge into the KB, and the corresponding operator should be defined in such a way to compute a consistent KB that supports the new knowledge. In the case of deletion from a consistent KB, the aim is to come up with a consistent KB where the retracted knowledge is not valid. In both cases, the crucial aspect is that evolving a consistent KB should not introduce inconsistencies.

Moreover, different change operators are appropriate depending on whether the KB is undergoing a *revision*, i.e., a correction of the state of belief, or an *update*, reflecting change of the world [62].

Existing DL systems do not provide explicit services for KB change. Nevertheless, many recent papers demonstrate that the interest towards a well-defined approach to knowledge base (KB) change is growing significantly [33, 40, 45, 78, 109].

Following the tradition of the work on knowledge revision and update, all the above papers advocate some minimality criterion in the changes of the KB that must be undertaken to realize the change operations. In other words, the need is commonly perceived of keeping the distance between the original KB and the KB resulting from the application of a change operator minimal. There are two main approaches to define such a distance, called *model-based* and *formula-based*, respectively.

The intuition behind model-based approaches is the following. There is an actual *state-of-affairs* of the world of which, however, we have only an incomplete specification. Such specification identifies a (typically infinite) set of models, each corresponding to a *state-of-affairs* that we consider possible.

Among them there is one model corresponding to the actual *state-of-affairs*, but we don't know which one is. Now when we perform an update we aim at changing the actual *state-of-affairs*. However, since we don't really know which of our models correspond to the actual *state-of-affairs*, we apply the change on every possible model. One basic problem with this approach is to characterize the language needed to express the KB that exactly captures the resulting set of models.

Conversely, in the formula-based approaches, the objects of change are sets of formulae. That is, the result of the change is explicitly defined in terms of a formula, by resorting to some minimality criterion with respect to the formula expressing the original KB. Here, the basic problem is that the formula constituting the result of an evolution operation is not unique in general.

In this chapter, we study the problem of updating a KB. In particular, we concentrate on the case where the KB is consistent, and we focus our attention to scenarios characterized by the following assumptions:

1. We consider the case where the evolution affects only the instance level of the KB, i.e., the ABox. In other words, we enforce the condition that the KB resulting from the application of the evolution operators has the same TBox as the original KB (similarly to [40, 78]). This assumptions are motivated by the fact that in information and knowledge management systems, extensional level tends to change frequently, while the domain representation provided by the intensional level typically remains unchanged for longer periods of time. In addition, changes in the intensional level are usually the result of an accurate manual design process.

2. We aim at a situation where the KB resulting from the evolution can be expressed in the same DL as the original KB. This is coherent with our goal of providing the foundations for the problem of equipping DL systems with evolution operators: if a DL system S is able to manage KBs expressed in a DL $\mathcal{L}$, the result of evolving such KBs should be expressible in $\mathcal{L}$.

3. The KBs resulting from the application of an evolution operator on two logically equivalent KBs should be mutually equivalent. In other words, we want the result to be independent from the syntactic form of the original KB.

Assumption (1), although limiting the generality of our approach, captures several interesting scenarios, including *ontology-based data management*, where the DL ontology is used as a logic-based interface to existing data sources.

As for item (2), we note that virtually all model-based approaches suffer from the expressibility problem. This has been reported in many recent papers, including [33, 40, 78], for various DLs. For this reason, we adopt a formula-based approach, inspired in particular by the work developed in [44] for updating logical theories. As in [44], we consider both insertions and deletions. However, we differ from [44] for an important aspect. We already noted that the formula constituting the result of an evolution operation is not unique in general. While [44] essentially proposes to keep the whole set of such formulas, we take a radical approach, and consider their intersection as the result of the evolution. In other words, we follow the *When In Doubt Throw It Out* (WIDTIO) [110] principle.

Finally, with regard to item (3), we sanction that the notion of distance between KBs refers to the closure of the ABox of a KB, rather than to the ABox itself. By basing the definition of distance on the closure of ABoxes and by preventing changes to the TBox, we achieve the goal of making the result of our operators independent from the form of the original KB.

The following example, which we use as running example in this part of the thesis, gives an intuition of the problem of updating a KB.

Example 8. We consider a portion of the Formula One domain. We know that official drivers (OD) and test drivers (TD) are both team members (TM), and official drivers are not test drivers. Every team member is a member of (mf) exactly one team (FT), and every team has at most one official driver. Finally, no race director (RD) is a member of a team. We also know that *john* is the official driver of team t_1, that *bob* is a test driver, and that *tom* is a team member. The corresponding *DL-Lite*$_{A,id}$-KB $\mathcal{K}$ is:

$$\begin{array}{lll} \mathcal{T} = \{ & OD \sqsubseteq TM, \quad TD \sqsubseteq TM, \quad OD \sqsubseteq \neg TD, \quad RD \sqsubseteq \neg TM, \\ & TM \sqsubseteq \exists mf, \quad TM \sqsubseteq \neg FT, \quad \exists mf \sqsubseteq TM, \quad \exists mf^- \sqsubseteq FT, \\ & FT \sqsubseteq \exists mf^-, \quad RD \sqsubseteq \neg FT, \quad (id\, OD\, mf), \quad (\mathsf{funct}\ mf) \qquad \} \\ \mathcal{A} = \{ & OD(john), \quad mf(john, t_1), \quad TD(bob), \quad TM(tom) \quad \} \end{array}$$

What happens if the domain changes in such a way that *tom* becomes the new official driver of team t_1? Which is the KB resulting from the fact that we retract that *bob* is a test driver? □

In this chapter, we first present some desirable properties that, in our opinion, characterize a reasonable update operator in our setting. Then, through a comparison among the various formula-based approaches proposed in literature, we illustrate our propose of update operators for both inserting and deleting facts from a consistent KB, and we show that they posses all the features that a reasonable update operator for KBs should have.

In all the chapters 6 and 7, we assume that the KB, that is subjected to update, is consistent.

6.1 Desirable properties for KB update

The following list collects a set of desirable properties that, in our opinion, an update should capture.

1. (*Success of the update*): in the case where the update operation is an insertion of a set of facts, the KB resulting from the update operation should entail these facts. Conversely, if the update operation is a deletion of a set of facts, then the resulting KB should not entail these facts.

2. (*Uniqueness of the result*): it is reasonable to assume that ontology-based information management systems cannot handle more than one knowledge base at a time. For this reason, we argue that the result of the update operation should be unique. Alternative, the update operator can return a set of KB leaving to the user the decision of which one should be chosen as result.

3. (*Consistency preservation*): when the original KB is consistent, the update operator should guarantee that the resulting KB is also consistent.

4. (*Expressibility*): the KB resulting from the update operation should be expressed in the same DL as the original KB. As we said earlier, this is coherent with our goal of providing the foundations for equipping DL ontology-based information management systems with update operators.

5. (*Syntax independence*): the KBs resulting from the application of the update operator on two logically equivalent KBs should be mutually equivalent. In other words, we want the result to be independent from the syntactic form of the original KB.

6. (*Compliance with the minimal-change principle*): the KB resulting from the update operation should be as close as possible to the original KB. In other words, we follow the idea of retaining as much knowledge as possible from the original KB.

Clearly, the *Uniqueness of the result* property is related to the *expressibility* property. The above properties are not new in the study of knowledge change. In fact, these kinds of property are often needed to meet rationality postulates, similar to those in the AGM theory [3] in belief revision (see. [46, 99] for a dissertation of the AGM theory in Description Logic), or to the postulates for belief update proposed by Katsuno and Mendelzon [62]. Moreover, similar properties have been proposed in literature as comparison criteria among change operators (e.g., in [97]).

6.2 Accomplishing an update

In this section we provide a formal definition of when an ABox "realizes" the update of a KB; where an update can be of two types: "update by insertion" and "update by deletion". In this chapter, if $\mathcal{K}$ is the DL KB that we want update with a finite set of atomic ABox assertions F, then we assume that $\mathcal{K}$ is consistent. In other words, we do not consider the update of inconsistent KBs (see Chapter 10 for a treatment of the update of inconsistent KBs).

According to the *success of the update* property, if $\mathcal{K}'$ is the result of updating the KB $\mathcal{K}$ with the insertion of F, we expect that $\mathcal{K}'$ entails F. Also, according to the *consistency preservation* property, we expect that $\mathcal{K}'$ is consistent. Similarly, if $\mathcal{K}'$ is the KB resulting from the deletion of F from $\mathcal{K}$, then $\mathcal{K}'$ should be consistent and should not entail F. This may require changing the ABox of $\mathcal{K}$ in a manner that guarantees the expected result.

Following those principles, and inspired by the work in [44], we adopt the following definition, which specifies when a set of ABox assertions "realizes" the insertion of a set of ABox assertions into a KB $\langle \mathcal{T}, \mathcal{A} \rangle$.

Definition 16. Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB. An ABox $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{A}'$ is $\mathcal{T}$-consistent, and $\langle \mathcal{T}, \mathcal{A}' \rangle \models F$.

Example 9. Consider the KB $\mathcal{K}$ of Example 8, and suppose that *bob* becomes a race director, and *tom* becomes the new official driver of team t_1. In order to reflect this new information, we change $\mathcal{K}$ with the insertion of $F = \{RD(bob), OD(tom), mf(tom, t_1)\}$. Since the TBox implies that a race director cannot be a team member, $RD(bob)$ contradicts $TD(bob)$. Also, since every team has at most one official driver, $\{OD(tom), mf(tom, t_1)\}$ contradicts the fact that *john* is the official driver of team t_1. According to Definition 16, the following ABoxes accomplish the update of $\mathcal{K}$ with the insertion of F.

$$
\begin{array}{lllll}
\mathcal{A}_1^+ = \{ & RD(bob), & OD(tom), & mf(tom, t_1) \} & \\
\mathcal{A}_2^+ = \{ & RD(bob), & OD(tom), & mf(tom, t_1), & TM(john) \} \\
\mathcal{A}_3^+ = \{ & RD(bob), & OD(tom), & mf(tom, t_1), & mf(john, t_1) \} \\
\mathcal{A}_4^+ = \{ & RD(bob), & OD(tom), & mf(tom, t_1), & OD(john) \} \\
\mathcal{A}_5^+ = \{ & RD(bob), & OD(tom), & mf(tom, t_1), & OD(john) \quad mf(john, t_2) \}
\end{array}
$$

□

Similarly, we define when a set of ABox assertions accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$.

Definition 17. Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB. An ABox $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ if $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models F$.

Example 10. Consider again the KB $\mathcal{K}$ presented in Example 8. Now, suppose that we do not know anymore whether it is true that *john* is a member of the formula one team t_1, or, even more, whether it is true that *john* is a team member at all. Moreover, we do not know anymore if it is true that *bob* is a test driver. Then, we change $\mathcal{K}$ with the deletion of $F_2 = \{mf(john, t_1), TM(john), TD(bob)\}$. According to Definition 17, it is easy to verify that the following ABoxes accomplish the update of $\mathcal{K}$ with the deletion of F_2.

$\mathcal{A}_1^- = \{\ \}$
$\mathcal{A}_2^- = \{\ TM(tom)\ \}$
$\mathcal{A}_3^- = \{\ TD(bob),\quad TM(tom),\quad FT(t_1)\ \}$
$\mathcal{A}_4^- = \{\ OD(john),\quad mf(john, t_1),\quad TM(tom)\ \}$
$\mathcal{A}_5^- = \{\ OD(john),\quad mf(john, t_1),\quad TM(tom),\quad TM(bob)\ \}$
$\mathcal{A}_6^- = \{\ OD(john),\quad TD(bob),\quad TM(tom)\ \}$
$\mathcal{A}_7^- = \{\ OD(john),\quad TD(bob),\quad TM(tom),\quad FT(t_1)\ \}$
$\mathcal{A}_8^- = \{\ OD(john),\quad TD(bob),\quad TM(tom),\quad FT(t_1),\quad OD(tom)\ \}$

□

Note that, according to the definitions above, if $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of the insertion of the set F, then we have that $F \subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$, and if $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of the deletion of F, then $F \not\subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$.

In general, if $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a consistent KB and F is a set of ABox assertions, the existence of at least one ABox accomplishing the update is not guaranteed. The following propositions specify when such an ABox exists in the case insertion and deletion respectively.

Proposition 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{L}$-KB and let F be a set of ABox assertions. An ABox $\mathcal{A}'$ accomplishing the insertion of F into $\mathcal{K}$ exists if and only if $Mod(\langle \mathcal{T}, F \rangle) \neq \emptyset$.*

Proof.
($\Rightarrow$) Let $\mathcal{A}'$ be an ABox accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$. Since $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, by Definition 16, $\langle \mathcal{T}, \mathcal{A}' \rangle$ is $\mathcal{T}$-consistent and $\langle \mathcal{T}, \mathcal{A}' \rangle \models F$. This means that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$, and $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \subseteq Mod(\langle \mathcal{T}, F \rangle)$. Therefore $Mod(\langle \mathcal{T}, F \rangle) \neq \emptyset$.

($\Leftarrow$) Let F be a set of ABox assertions such that $Mod(\langle \mathcal{T}, F \rangle) \neq \emptyset$. We have that F is $\mathcal{T}$-consistent and, obviously, $\langle \mathcal{T}, F \rangle \models F$. Hence, F accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$. ∎

Proposition 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent $\mathcal{L}$-KB and let F be a set of ABox assertions. An ABox $\mathcal{A}'$ accomplishing the deletion of F from $\mathcal{K}$ exists if and only if $\langle \mathcal{T}, \emptyset \rangle \not\models F$.*

Proof.
($\Rightarrow$) Let $\mathcal{A}'$ be an ABox accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. By Definition 17, we have that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \not\subseteq Mod(\langle \mathcal{T}, F \rangle)$, i.e., there exists a model $m \in Mod(\langle \mathcal{T}, \mathcal{A} \rangle)$ such that $m \notin Mod(\langle \mathcal{T}, F \rangle)$. Since $\mathcal{L}$ is monotonic, we have $Mod(\langle \mathcal{T}, \mathcal{A} \rangle) \subseteq Mod(\langle \mathcal{T}, \emptyset \rangle)$. Hence, there exists a model $m \in Mod(\langle \mathcal{T}, \emptyset \rangle)$ such that $m \notin Mod(\langle \mathcal{T}, F \rangle)$, i.e., $\langle \mathcal{T}, \emptyset \rangle \not\models F$.

($\Leftarrow$) Clearly, if $\langle \mathcal{T}, \emptyset \rangle \not\models F$, then the empty ABox $\emptyset$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. ∎

Note that the conditions that guarantee the existence of ABoxes accomplishing the update depend only on the TBox of the original KB.

We conclude this section with an example showing a case where no ABoxes accomplishing the deletion of a set of facts from a KB exists.

Example 11. Consider the TBox $\mathcal{T}$ constituted by the following assertions:

$$\begin{array}{lcl@{\qquad}lcl} \{red\} & \sqsubseteq & \mathsf{WarmColor}, & \{brown\} & \sqsubseteq & \mathsf{WarmColor}, \\ \{green\} & \sqsubseteq & \mathsf{CoolColor}, & \{blue\} & \sqsubseteq & \mathsf{CoolColor}, \\ \mathsf{CoolColor} & \sqsubseteq & \mathsf{Color}, & \mathsf{WarmColor} & \sqsubseteq & \mathsf{Color}; \end{array}$$

and consider the set of ABox assertions $F = \{\mathsf{Color}(red), \mathsf{Color}(blue)\}$. It is easy to see that every interpretation $\mathcal{I}$ has to satisfy the conditions below in order to be a model of $\mathcal{T}$.

$$\begin{array}{lcl@{\qquad}lcl} (red)^{\mathcal{I}} & \in & \mathsf{WarmColor}^{\mathcal{I}} & (red)^{\mathcal{I}} & \in & \mathsf{Color}^{\mathcal{I}} \\ (brown)^{\mathcal{I}} & \in & \mathsf{WarmColor}^{\mathcal{I}} & (brown)^{\mathcal{I}} & \in & \mathsf{Color}^{\mathcal{I}} \\ (green)^{\mathcal{I}} & \in & \mathsf{CoolColor}^{\mathcal{I}} & (green)^{\mathcal{I}} & \in & \mathsf{Color}^{\mathcal{I}} \\ (blue)^{\mathcal{I}} & \in & \mathsf{CoolColor}^{\mathcal{I}} & (blue)^{\mathcal{I}} & \in & \mathsf{Color}^{\mathcal{I}} \end{array}$$

Therefore, for every ABox $\mathcal{A}$, there does not exist an ABox $\mathcal{A}'$ which accomplishes the deletion of F from the KB $\langle \mathcal{T}, \mathcal{A} \rangle$. □

6.3 Accomplishing an update minimally

After adding new facts into an existing KB, one may find that the revised KB becomes inconsistent. A strategy to overcome such a situation can be to remove a part of the original ABox with the aim of preventing inconsistency. Similarly, if the goal is to update a KB with the deletion of a set of facts, we could be forced to retract facts from the original KB.

Intuitively, Definitions 16 and 17 allow only for obtaining new KBs that obey the *success of the update* property. Another principle which is undoubtedly considered as one of the most important in knowledge base evolution, is the *minimal change* property. Such principle states that the KB resulting

from the update should be as "close" as possible to the original one, and that, during the process, the "loss of information" should be minimized. The terms "closeness" and "loss of information" have no single interpretation in the literature. For example, in model-based approach, closeness and loss of information are measured in model-theoretic terms, by the definition of distance metric between the models satisfying the modified knowledge base and the original one.

In the formula-based approach, the goal becomes the preservation of the facts contained in the original ABox or, alternatively, the ones entailed by the original KB. As the reader will see, we follow the latter approach.

In order to give a characterization of the notion of "closeness", we refer to the relation of *fewer changes* between two sets of ABox assertions with respect to another one given by Fagin, Ullman and Vardi in [44].

Let $\mathcal{A}$, $\mathcal{A}_1$, and $\mathcal{A}_2$ be three finite sets of ABox assertions. We say that $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_2$ with respect to $\mathcal{A}$ if $\mathcal{A} \setminus \mathcal{A}_1 \subset \mathcal{A} \setminus \mathcal{A}_2$. Also, we say that $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same deletions with respect to $\mathcal{A}$ if $\mathcal{A} \setminus \mathcal{A}_1 = \mathcal{A} \setminus \mathcal{A}_2$. Finally, we say that $\mathcal{A}_1$ has fewer insertions than $\mathcal{A}_2$ with respect to $\mathcal{A}$ if $\mathcal{A}_1 \setminus \mathcal{A} \subset \mathcal{A}_2 \setminus \mathcal{A}$.

Definition 18. Let $\mathcal{A}$, $\mathcal{A}_1$, and $\mathcal{A}_2$ be three finite sets of ABox assertions. We say that $\mathcal{A}_1$ has *fewer changes* than $\mathcal{A}_2$ with respect to $\mathcal{A}$ if

(*i*) $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_2$ with respect to $\mathcal{A}$, or

(*ii*) $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same deletions with respect to $\mathcal{A}$, and $\mathcal{A}_1$ has fewer insertions than $\mathcal{A}_2$ with respect to $\mathcal{A}$.

Note that, the notion of *fewer changes* gives preference to such ABoxes which have fewer deletions over the ABoxes which have fewer insertions. Such behavior is justified by the idea of retaining as much knowledge as possible from the original KB.

The notion of fewer changes specify a relation between two ABoxes with respect to a third ABox. In what follows, we give a notable property of such relation. Firstly, we recall that given an alphabet Γ, we denote with $HB(\Gamma)$ the *Herbrand Base of* Γ, i.e., the set of atomic ABox assertions that can be built over the alphabet Γ. Moreover, we denote with $\wp(HB(\Gamma))$ the powerset of $HB(\Gamma)$.

Lemma 17. *Let Γ be an alphabet, and let $\mathcal{P} \subseteq \wp(HB(\Gamma))$. The relation "has fewer changes" is a strict partial order on $\mathcal{P}$.*

Proof. Let $\mathcal{A}$, $\mathcal{A}_1$, and $\mathcal{A}_2$ be three ABoxes in $\mathcal{P}$. To prove the claim we have to show that:

(*i*) the statement $\mathcal{A}_1$ has fewer changes than $\mathcal{A}_1$ with respect to $\mathcal{A}$ does not hold (anti-reflexivity);

(*ii*) if $\mathcal{A}_1$ has fewer changes than $\mathcal{A}_2$ with respect to $\mathcal{A}$, and $\mathcal{A}_2$ has fewer changes than $\mathcal{A}_3$ with respect to $\mathcal{A}$, then $\mathcal{A}_1$ has fewer changes than $\mathcal{A}_3$ with respect to $\mathcal{A}$. (transitivity).

The first item directly follows from Definition 18. Hence, it remains to prove that if $\mathcal{A}_1$ has fewer changes than $\mathcal{A}_2$ with respect to $\mathcal{A}$, and $\mathcal{A}_2$ has fewer changes than $\mathcal{A}_3$ with respect to $\mathcal{A}$, then $\mathcal{A}_1$ has fewer changes than $\mathcal{A}_3$ with respect to $\mathcal{A}$. According to Definition 18, the following cases are conceivable:

1. $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_2$ w.r.t. $\mathcal{A}$, and $\mathcal{A}_2$ has fewer deletions than $\mathcal{A}_3$ w.r.t. $\mathcal{A}$. This means that $\mathcal{A} \setminus \mathcal{A}_1 \subset \mathcal{A} \setminus \mathcal{A}_2$ and $\mathcal{A} \setminus \mathcal{A}_2 \subset \mathcal{A} \setminus \mathcal{A}_3$. It follows that $\mathcal{A} \setminus \mathcal{A}_1 \subset \mathcal{A} \setminus \mathcal{A}_3$. Hence $\mathcal{A}_1$ has fewer changes than $\mathcal{A}_3$ w.r.t. $\mathcal{A}$.

2. $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_2$ w.r.t. $\mathcal{A}$, and $\mathcal{A}_2$ and $\mathcal{A}_3$ have the same deletions w.r.t $\mathcal{A}$. This means that $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_3$ w.r.t. $\mathcal{A}$. Hence, $\mathcal{A}_1$ has fewer changes than $\mathcal{A}_3$ w.r.t. $\mathcal{A}$.

3. $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same deletions w.r.t. $\mathcal{A}$ and $\mathcal{A}_1$ has fewer insertions than $\mathcal{A}_2$ w.r.t. $\mathcal{A}$. Moreover, $\mathcal{A}_2$ has fewer deletions than $\mathcal{A}_3$ w.r.t. $\mathcal{A}$. It follows that $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_3$ w.r.t. $\mathcal{A}$. Hence, $\mathcal{A}_1$ has fewer changes than $\mathcal{A}_3$ w.r.t. $\mathcal{A}$.

4. $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ have the same deletions w.r.t. $\mathcal{A}$, and: (1) $\mathcal{A}_1$ has fewer insertions than $\mathcal{A}_2$ w.r.t. $\mathcal{A}$; (2) $\mathcal{A}_2$ has fewer insertions than $\mathcal{A}_3$ w.r.t. $\mathcal{A}$. This means that $\mathcal{A}_1 \setminus \mathcal{A} \subset \mathcal{A}_2 \setminus \mathcal{A}$, and $\mathcal{A}_2 \setminus \mathcal{A} \subset \mathcal{A}_3 \setminus \mathcal{A}$. It follows that $\mathcal{A}_1 \setminus \mathcal{A} \subset \mathcal{A}_3 \setminus \mathcal{A}$. Hence, $\mathcal{A}_1$ has fewer changes than $\mathcal{A}_3$ w.r.t. $\mathcal{A}$.

In conclusion, we have shown that the relation "has fewer changes" is a strict partial order on $\mathcal{P}$. ∎

Following the notion of *fewer changes* we define when an ABox accomplishes the insertion of a set of facts into a KB minimally.

Definition 19. Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB. An ABox $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally if:

1. $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, and

2. there is no $\mathcal{A}''$ that accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ has fewer changes than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$.

Note that, differently from the definition given in [44], we refer to the closure of the ABox of a KB, rather than to the ABox itself. This choice is motivated by the argument that, in our opinion, the knowledge, that the KB provides about the world, is constituted by the all the facts that we can derive from the whole KB, i.e. from both the TBox and the ABox, and not only from the facts in the ABox.

Example 12. We refer to Example 9. According to Definition 19, we have that only the ABoxes $\mathcal{A}_3^+$ and $\mathcal{A}_4^+$ accomplish the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Indeed, it is easy to see that both $\mathsf{cl}_\mathcal{T}(\mathcal{A}_3^+)$ and $\mathsf{cl}_\mathcal{T}(\mathcal{A}_4^+)$ have fewer deletions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}_1^+)$ and $\mathsf{cl}_\mathcal{T}(\mathcal{A}_2^+)$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$, and that $\mathsf{cl}_\mathcal{T}(\mathcal{A}_4^+)$ has fewer insertions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}_5^+)$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. □

Similarly, we define the notion of an ABox which accomplishes the deletion of a set of facts from a KB minimally.

Definition 20. Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB. An ABox $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally if:

1. $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and
2. there is no $\mathcal{A}''$ that accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ has fewer changes than $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$.

Example 13. Consider the ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ of Example 10. The reader can verify that only the ABoxes $\mathcal{A}_5^-$ and $\mathcal{A}_7^-$ accomplish the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Indeed, we have that $\mathsf{cl}_\mathcal{T}(\mathcal{A}_5^-)$ has fewer deletions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}_1^-)$, $\mathsf{cl}_\mathcal{T}(\mathcal{A}_2^-)$, and $\mathsf{cl}_\mathcal{T}(\mathcal{A}_4^-)$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. Whereas, $\mathsf{cl}_\mathcal{T}(\mathcal{A}_7^-)$ has fewer deletions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}_1^-)$, $\mathsf{cl}_\mathcal{T}(\mathcal{A}_2^-)$, $\mathsf{cl}_\mathcal{T}(\mathcal{A}_3^-)$, and $\mathsf{cl}_\mathcal{T}(\mathcal{A}_4^-)$, and that it has fewer insertions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}_8^-)$. □

Definition 19 and Definition 20 are *non-constructive* in the sense that they do not give any indications of how to build those ABoxes that accomplish an update minimally. The following theorems, which are an adaptation to our setting of two results reported in [44], give constructive equivalent conditions.

Theorem 11. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent DL KB, and let $\mathcal{A}'$ and F be two sets of ABox assertions. $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally if and only if $\mathsf{cl}_\mathcal{T}(\mathcal{A}') = \mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$, for some maximal subset $\mathcal{A}''$ of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent.*

Proof.
($\Rightarrow$) Let $\mathcal{A}'$ be an ABox accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. We prove that there exists a set $\mathcal{A}''$, such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}') = \mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$ and $\mathcal{A}''$

is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. Since $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, from Proposition 1 we know that $\langle \mathcal{T}, F \rangle$ is consistent.

We start proving that $\mathcal{A}''$ is a subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. Obviously, $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent since $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$. Now, suppose that $\mathcal{A}'' \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A})$, which means that there exists an assertion $\alpha \in \mathcal{A}''$ such that $\alpha \notin \mathsf{cl}_\mathcal{T}(\mathcal{A})$.

Consider the set of ABox assertions $\mathcal{B} = (\mathcal{A}'' \setminus \{\alpha\}) \cup F$. Clearly, we have that $\langle \mathcal{T}, \mathcal{B} \rangle$ is consistent, and that $\langle \mathcal{T}, \mathcal{B} \rangle \models F$. The following cases are conceivable:

- $\mathsf{cl}_\mathcal{T}(\mathcal{B})$ has the same deletions of $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$, and $\mathsf{cl}_\mathcal{T}(\mathcal{B})$ has fewer insertions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. This means, that $\mathcal{A}'$ does not accomplish the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, which is a contradiction.

- $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ has fewer deletions of $\mathsf{cl}_\mathcal{T}(\mathcal{B})$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. This means that there is a subset $\mathcal{A}'''$ of $\mathcal{A}''$ such that $\alpha \in \mathcal{A}'''$ and $\langle \mathcal{A}''' \cup F \rangle \models \beta$, where $\beta \in \mathsf{cl}_\mathcal{T}(\mathcal{A})$, but $\alpha \notin \mathsf{cl}_\mathcal{T}(\mathcal{B})$. Hence, we can build the $\mathcal{T}$-consistent set $\mathcal{B}' = \mathcal{B} \cup \{\beta\}$ which accomplish the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, and such that $\mathsf{cl}_\mathcal{T}(\mathcal{B}')$ has fewer changes than $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. This implies, that $\mathcal{A}'$ does not accomplish the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, which is a contradiction.

Now, we prove that $\mathcal{A}''$ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. Assume, by contradiction, that $\mathcal{A}''$ is not a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. This means that there is an assertion $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\alpha \notin \mathcal{A}''$, and $\mathcal{A}'' \cup \{\alpha\} \cup F$ is $\mathcal{T}$-consistent. Moreover, this means that $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus (\mathcal{A}'' \cup \{\alpha\}) \subset \mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathcal{A}''$. Now, consider the ABox $\mathcal{A}''' = \mathcal{A}'' \cup \mathsf{cl}_\mathcal{T}(F) \cup \{\alpha\}$. Since $\mathcal{A}'' \cup \{\alpha\} \cup F$ is $\mathcal{T}$-consistent, and since $\mathsf{cl}_\mathcal{T}(F) \subseteq \mathcal{A}'''$, then we have that $\mathcal{A}'''$ is $\mathcal{T}$-consistent and that $\langle \mathcal{T}, \mathcal{A}''' \rangle \models F$. According to Definition 16, this means that $\mathcal{A}'''$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$. Moreover, we have that $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(\mathcal{A}''') \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. Hence, $\mathsf{cl}_\mathcal{T}(\mathcal{A}''')$ has fewer deletions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. Thus, we have a contradiction.

($\Leftarrow$) Let $\mathcal{A}'$ be an ABox such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}') = \mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. We prove that $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Since $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent, then by Proposition 1 we know that the set $\mathcal{U}$ of the ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ is not empty.

Obviously, $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, then we have to prove that $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Toward a

contradiction, consider a $\mathcal{T}$-consistent ABox $\mathcal{A}'''$ such that $F \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}''')$. This means that $\mathcal{A}'''$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$. Now, suppose that $\mathsf{cl}_\mathcal{T}(\mathcal{A}''')$ has fewer changes than $\mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. There are two possible cases:

1. $\mathsf{cl}_\mathcal{T}(\mathcal{A}''')$ has fewer deletions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. Hence, there is an assertion $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A}''')$, $\alpha \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$, and $\mathsf{cl}_\mathcal{T}(\mathcal{A} \cup F) \cup \{\alpha\}$ is $\mathcal{T}$-consistent. Thus, contradicting the hypothesis that $\mathcal{A}'$ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}' \cup F$ is $\mathcal{T}$-consistent.

2. $\mathsf{cl}_\mathcal{T}(\mathcal{A}''')$ and $\mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$ have the same deletions with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$, and $\mathsf{cl}_\mathcal{T}(\mathcal{A}''')$ has fewer insertions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. This means that there is an assertion α in $\mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$ which is not in $\mathsf{cl}_\mathcal{T}(\mathcal{A}''')$. But this is impossible , since $\mathcal{A}'' \subseteq \mathcal{A}$ and since $\mathsf{cl}_\mathcal{T}(\mathcal{A}''')$ and $\mathsf{cl}_\mathcal{T}(\mathcal{A}'' \cup F)$ have the same deletions with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. Then, we have a contradiction.

■

We now focus on the deletion, and we provide the analogous of Theorem 11 for deletion operations.

Theorem 12. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent $\mathcal{L}$-KB and let $\mathcal{A}'$ and F be two sets of ABox assertions. $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally if and only if $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$.*

Proof.
($\Rightarrow$) Let $\mathcal{A}'$ be an ABox accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. We prove that $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. Proposition 2 ensures that $\langle \mathcal{T}, \emptyset \rangle \not\models F$. From Definition 17, we have that $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. Toward a contradiction, assume that $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ is not a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. This means that there is an ABox $\mathcal{A}''$ such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}') \subset \mathsf{cl}_\mathcal{T}(\mathcal{A}'') \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$, and $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}'')$. Obviously, $\mathcal{A}''$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and, since $\mathsf{cl}_\mathcal{T}(\mathcal{A}') \subset \mathsf{cl}_\mathcal{T}(\mathcal{A}'') \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$, we have that $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(\mathcal{A}'') \subset \mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(\mathcal{A}')$, then $\mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ has fewer changes than $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. Therefore, $\mathcal{A}'$ does not accomplish the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, which is a contradiction.

($\Leftarrow$) Let $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ be a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. We prove that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Since $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$ then Proposition 2 ensures that at least one ABox accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ exists.

Toward a contradiction, assume that $\mathcal{A}'$ does not accomplish the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Thus, there is an ABox $\mathcal{A}''$ such that, $\mathcal{A}''$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and $\mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ has fewer changes than

$\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. Since $\mathcal{A}''$ accomplishes the deletion of F, we have that $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}'')$. Now, since $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ is a subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$, we have that $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ has no insertions with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$, thus, $\mathsf{cl}_\mathcal{T}(\mathcal{A})''$ has no insertions too. This means that $\mathsf{cl}_\mathcal{T}(\mathcal{A})''$ has fewer changes than $\mathsf{cl}_\mathcal{T}(\mathcal{A})'$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ since $\mathsf{cl}_\mathcal{T}(\mathcal{A})''$ has fewer deletions. But this implies that there is an assertion $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that: (i) $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A})''$; (ii) $\alpha \notin \mathsf{cl}_\mathcal{T}(\mathcal{A})'$; (iii) $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}' \cup \{\alpha\})$. It follows that $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ is not a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$, which is a contradiction. ∎

We end this subsection with a property that will be used in the following.

Lemma 18. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a KB and let F be a set of ABox assertions. Let $\mathcal{A}'$ be a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}' \cup F$ is $\mathcal{T}$-consistent. Then $\mathcal{A}' = \mathsf{cl}_\mathcal{T}(\mathcal{A}')$.*

Proof. Since $\mathcal{A}'$ is maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}' \cup F$ is $\mathcal{T}$-consistent then:

1. $\mathcal{A}' \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A})$;
2. $\langle \mathcal{T}, \mathcal{A} \cup F \rangle$ is consistent;
3. there is no set $\mathcal{A}''$ such that $\mathcal{A} \subset \mathcal{A}'' \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A})$ and $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle$ is consistent.

We have to prove that if $\mathcal{A}'$ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}' \cup F$ is $\mathcal{T}$-consistent, then $\mathcal{A}'$ is closed with respect to $\mathcal{T}$. Obviously, if $\mathcal{A}' \cup F$ is $\mathcal{T}$-consistent, then also $\mathsf{cl}_\mathcal{T}(\mathcal{A}') \cup F$ is $\mathcal{T}$-consistent. The proof proceeds by contradiction as follows. Suppose that $\mathcal{A}' \neq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. The following two cases are conceivable:

- There exists an assertion α such that $\alpha \in \mathcal{A}'$ and $\alpha \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. But, since $\mathcal{A}' \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}')$ then $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A}')$, which contradicts $\alpha \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}')$.

- There exists an assertion α such that $\alpha \notin \mathcal{A}'$ and $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. That is, $\mathcal{A}' \subset \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. If $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A}')$, then there exists a subset σ of $\mathcal{A}'$ such that $\langle \mathcal{T}, \sigma \rangle \models \alpha$. Since $\mathcal{A}' \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A})$ then, $\sigma \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A})$ and $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A})$. Hence, $\mathcal{A}' \subset \mathsf{cl}_\mathcal{T}(\mathcal{A}') \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A})$, which contradict the fact that $\mathcal{A}'$ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}' \cup F$ is $\mathcal{T}$-consistent. Hence, $\mathcal{A}' = \mathsf{cl}_\mathcal{T}(\mathcal{A}')$.

Therefore we have shown that $\mathcal{A}' = \mathsf{cl}_\mathcal{T}(\mathcal{A}')$. ∎

6.4 Update operators based on the WIDTIO principle

Consider the KB $\mathcal{K}$ formed by a TBox $\mathcal{T}$ constituted by the axioms $C_1 \sqcap C_2 \sqsubseteq C_3$ and $C_3 \sqsubseteq \neg C_4$, and by the ABox $\mathcal{A}$ containing the assertion $C_3(o)$. It is easy to verify that there is only one ABox accomplishing the insertion of the new fact $C_4(o)$ into $\mathcal{K}$, that is $\mathcal{A}^u = \{C_4(o)\}$. The example singles out a case where there is a unique ABox accomplishing the insertion minimally, and therefore, it is obvious to take such an ABox as result of the change. Now, consider for example, the KB $\mathcal{K}'$ formed by the TBox $\mathcal{T}$ described above, and by the ABox $\mathcal{A}' = \{C_1(o), C_2(o)\}$. It is easy to verify that there are two ABoxes that accomplish the insertion of $\{C_4(o)\}$ into $\mathcal{K}'$ minimally: $\mathcal{A}_1^u = \{C_1(o), C_4(o)\}$ and $\mathcal{A}_2^u = \{C_2(o), C_4(o)\}$. What should be done if several ABoxes accomplish the insertion of new facts into a KB minimally?

Different approaches are conceivable. The most straightforward approach to cope with the *multiple results* problem, is the one inspired by the *Set-Of-Theories* (SOT) approach proposed by Ginsberg [52] and, independently, by Fagin, Ullman, and Vardi in [44]. In such an approach the idea is considering as result of the update the set of KBs built by using the ABoxes which accomplish the update minimally. We refer to this approach as *SOT approach.*

Definition 21. [SOT approach] Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent $\mathcal{L}$-KB, F a set of ABox assertions, and $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be the set of all ABoxes accomplishing the insertion of F into $\mathcal{K}$ minimally. Then the result of updating $\mathcal{K}$ with the insertion of F is the set of KBs $\mathcal{K} \oplus_{SOT} F$ defined as follows.

$$\mathcal{K} \oplus_{SOT} F = \{\langle \mathcal{T}, \mathcal{A}_i \rangle \mid \mathcal{A}_i \in \mathcal{U}\}$$

Example 14. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and F be respectively the KB and the set of ABox assertions presented in Example 9. Then

$\mathcal{K} \oplus_{SOT} F = \{$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), mf(john, t_1)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), mf(john, t_1), TM(tom)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), mf(john, t_1), FT(t_1)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), mf(john, t_1), TM(john)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), mf(john, t_1), TM(tom), FT(t_1)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), mf(john, t_1), TM(tom), TM(john)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), mf(john, t_1), FT(t_1), TM(john)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), mf(john, t_1), TM(tom), FT(t_1), TM(john)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), OD(john)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), OD(john), TM(tom)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), OD(john), FT(t_1)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), OD(john), TM(john)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), OD(john), TM(tom), FT(t_1)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), OD(john), TM(tom), TM(john)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), OD(john), FT(t_1), TM(john)\}\rangle,$
$\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), OD(john), TM(tom), FT(t_1), TM(john)\}\rangle\}$

□

Therefore, $\mathcal{K} \oplus_{SOT} F$ is no more a unique KB, but it is actually a family of KBs, more precisely the family of KBs $\langle \mathcal{K}, \mathcal{A}_i \rangle$, with $\mathcal{A}_i$ belonging to the set of ABoxes accomplishing the insertion of F into $\mathcal{K}$ minimally. Similarly, it is possible to define the operator $\mathcal{K} \ominus_{\mathcal{G}} F$ for deletion.

Under the *SOT approach*, the logical consequences of the update are defined as the logical consequences of all the KBs in the set $\mathcal{K} \oplus_{SOT} F$. More formally, let ϕ be a first-order sentence. $\mathcal{K} \oplus_{SOT} F \models \phi$ if and only for all $\mathcal{K}' \in \mathcal{K} \oplus_{SOT} F$, $\mathcal{K}' \models \phi$. In other words all the ABoxes accomplishing the update are considered equally plausible, and inference is defined *skeptically* [76].

It is obvious that the *SOT approach* does not satisfy the property of *uniqueness of the result*. This means that a further step that involves a choice strategy is needed. One possibility is to let the user choose which ABox should be considered the result of the update, as proposed in [105]. Calvanese et al., in [33, 34], propose a pragmatic alternative that consists in choosing nondeterministically the ABox of the KB resulting from the update among the ABoxes accomplishing the update minimally. Obviously, such an approach brings the result of their update operator to be not uniquely defined.

In order to obtain a single KB from the application of the update operator, in [44] authors propose the *Cross-Product approach*. Roughly speaking, by adopting the *Cross-Product approach* the ABox resulting from the update is formed by the disjunction of all the ABoxes $\mathcal{A}_i$ accomplishing the insertion of F into $\mathcal{K}$ minimally, viewing each $\mathcal{A}_i$ as the conjunction of its membership assertions. As in the *SOT approach*, adopting the *Cross-Product approach* we have no loss of information. The price to pay is that the resulting KB can be exponentially larger than the original one [17]. Moreover, such approach requires that the DL used to express the KB allows for the disjunction operator. Consequently, in contrast with the *expressibility* property, it is not ensured that the KB resulting from the update can be expressed in the same DL as the original KB.

A radical approach to cope with the *multiple results* problem, and that allows for dealing with capturing *expressibility* property, is the one suggested by the *WIDTIO* (*When In Doubt Throw It Out* [110]) *principle* [51, 52]. The idea at the base of the *WIDTIO principle* consists in combining the KBs resulting from the *SOT approach* into a single one, by considering their intersection. Following the *WIDTIO principle*, in our update operators we define as result of the update the KB formed by the TBox of the original KB, and by the ABox computed as the intersection of the deductive closure of all ABoxes accomplishing the update. The following definitions provide a formal description of our update operators for both insertion and deletion.

Definition 22. Let $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be the set of all ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, and let $\mathcal{A}'$ be an ABox. $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F if

1. $\mathcal{U}$ is empty, and $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle$, or

2. $\mathcal{U}$ is nonempty, and $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \bigcap_{1 \leq i \leq n} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle$.

Notice that, by definition of our operators, in the case where no ABoxes accomplishing the insertion exist, the result of the update is logically equivalent to $\mathcal{K}$.

Example 15. Refer to the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and to the set of ABox assertions $F = \{RD(bob), OD(tom), mf(tom, t_1)\}$ of Example 9. By exploiting the results of Example 14 we have that:

$$\begin{aligned}
&\langle \mathcal{T},\ \{RD(bob), OD(tom), mf(tom, t_1), mf(john, t_1), TM(tom), FT(t_1),\\
&\qquad TM(john)\} \bigcap \{RD(bob), OD(tom), mf(tom, t_1), OD(john),\\
&\qquad TM(tom), FT(t_1), TM(john)\}\rangle\ =\\
&\langle \mathcal{T}, \{RD(bob), OD(tom), mf(tom, t_1), TM(tom), FT(t_1), TM(john)\}\rangle
\end{aligned}$$

□

Definition 23. Let $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be the set of all ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, and let $\mathcal{A}'$ be an ABox. $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F if

1. $\mathcal{U}$ is empty, and $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle$, or

2. $\mathcal{U}$ is nonempty, and $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \bigcap_{1 \leq i \leq n} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle$.

Note that, also in case of update with deletion, when no ABoxes accomplishing the update exist, the result is logically equivalent to $\mathcal{K}$.

Example 16. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and F be respectively the KB and the set of ABox assertions of Example 10. According to Definition 23, the result of updating $\mathcal{K}$ with the deletion of F is:

$$\begin{aligned}
&\langle \mathcal{T},\ \{OD(john), TM(john), TM(tom), mf(john, t_1), TM(bob), FT(t_1)\} \bigcap\\
&\qquad \{OD(john), TM(john), TM(tom), TD(bob), TM(bob), FT(t_1)\}\rangle =\\
&\langle \mathcal{T},\ \{OD(john), TM(john), TM(tom), TM(bob), FT(t_1)\}\rangle
\end{aligned}$$

□

6.5 Properties of our update operators

In this section, we first discuss how our update operators relates to the properties of update presented in Section 6.1, and then provide a discussion on how our update approach relates to the classical update postulates [62].

In what follows we assume that $\langle \mathcal{T}, \mathcal{A} \rangle$ is a consistent DL KB, and that F is a finite set of ABox assertions.

First, we show that, under the assumption that at least one ABox accomplishing the update of $\langle \mathcal{T}, \mathcal{A} \rangle$ with F exists, our update operators capture the *success of the update* property. This is stated in the next two propositions.

Proposition 3. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent $\mathcal{L}$-KB, and let F be a $\mathcal{T}$-consistent set of ABox assertions. Let $\langle \mathcal{T}, \mathcal{A}' \rangle$ be a KB resulting from updating $\mathcal{K}$ with the insertion of F according to the semantics in Definition 22. Then $\langle \mathcal{T}, \mathcal{A}' \rangle \models F$.*

Proof. We proceed by contradiction. Suppose that F is $\mathcal{T}$-consistent, and $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models F$, i.e. there exists an assertion $\alpha \in F$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models \alpha$. Since $\langle \mathcal{T}, F \rangle$ is consistent, then by Proposition 1 we have that the set $\mathcal{U} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ of ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally is not empty. Consequently, by Definition 22 we have that $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \bigcap_{A_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle$. Since $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models F$, we have that $F \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$, and then $F \notin \bigcap_{A_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i)$. This means that there exists an ABox $\mathcal{A}_j \in \mathcal{U}$ such that $F \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_j)$. Since a set of ABox assertions belongs to $\mathcal{U}$ if and only if such a set accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, it follows that $\langle \mathcal{T}, \mathcal{A}_j \rangle \models F$, i.e., $F \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_j)$, which is a contradiction. ∎

Proposition 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent $\mathcal{L}$-KB, and let F be a $\mathcal{T}$-consistent set of ABox assertions. Let $\langle \mathcal{T}, \mathcal{A}' \rangle$ be a KB resulting from updating $\mathcal{K}$ with the deletion of F according to the semantics in Definition 23. Then $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models F$.*

Proof. We proceed by contradiction. Since $\langle \mathcal{T}, \emptyset \rangle \not\models F$, by Proposition 2 we have that the set $\mathcal{U} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ of ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally is not empty. Consequently, by Definition 23 we have that $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \bigcap_{A_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle$. Suppose that $\langle \mathcal{T}, \mathcal{A}' \rangle \models F$. This means that for all $\mathcal{A}_i \in \mathcal{U}$ we have that $F \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i)$, hence, $\langle \mathcal{T}, \mathcal{A}_i \rangle \models F$. Since a set of ABox assertions belongs to $\mathcal{U}$ if and only if such a set accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, it follows that no ABox $\mathcal{A}_j$ such that $\langle \mathcal{T}, \mathcal{A}_j \rangle \models F$ belongs to $\mathcal{U}$, which is a contradiction. ∎

As we said earlier, we argue that the result of the update operation should be unique. This is also one of the reasons why we adopt the WIDTIO principle.

Closely related to this point is the *uniqueness of the result* property. In this regard we have the following proposition showing that, under the assumption that $\mathcal{A}$ is $\mathcal{T}$-consistent, both our update operators are actually functional.

Proposition 5. *If $\langle \mathcal{T}, \mathcal{A} \rangle$ is a consistent $\mathcal{L}$-KB, then, up to logical equivalence, there is exactly one result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F, and exactly one result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F.*

Proof. We start proving that if $\langle \mathcal{T}, \mathcal{A} \rangle$ is a consistent KB, then up to logical equivalence, there is exactly one result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F. The following two cases are conceivable.

1. if $\langle \mathcal{T}, F \rangle$ is not consistent, then, according to Definition 22 the result of the update is $\langle \mathcal{T}, \mathcal{A} \rangle$ itself. Hence, the claim is proved.

2. if $\langle \mathcal{T}, F \rangle$ is consistent, then, according to Definition 22, a KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathcal{A}_i$, where $\mathcal{U}$ is the set containing all the ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Toward a contradiction, suppose that both $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ are results of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F, and that $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ are not logically equivalent. This means that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) \neq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$, which is a contradiction.

We now prove the claim in case of deletion. Again, two cases are conceivable.

1. if $\langle \mathcal{T}, \emptyset \rangle \models F$, then, according to Definition 23 the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F is $\langle \mathcal{T}, \mathcal{A} \rangle$ itself. Hence, the claim is proved.

2. if $\langle \mathcal{T}, \emptyset \rangle \not\models F$. then, according to Definition 23, a KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F, if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathcal{A}_i$, where $\mathcal{U}$ is the set containing all the ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Again, if both the two KBs $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ are results of the update, then $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2) = \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathcal{A}_i$. This means that $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ are logically equivalent.

■

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB and let F be a set of ABox assertions. In the rest of this work, the result of updating $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F according to our semantics will be denoted by $\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F$, and the result of updating $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F will be denoted by $\mathcal{K} \ominus^{\mathcal{T}}_{\cap} F$.

Next, we show that our update operators obey the *consistency preservation* property.

Proposition 6. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent $\mathcal{L}$-KB, and let F be a set of ABox assertions. We have that $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F$ is consistent.*

Proof. Let $\langle \mathcal{T}, \mathcal{A}' \rangle$ be the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F. If $\langle \mathcal{T}, F \rangle$ is not consistent, then by Definition 22 we have that $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle$ is equal to $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle$, which is consistent. On the other hand, if $\langle \mathcal{T}, F \rangle$ is consistent then $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle = \langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle$, where $\mathcal{U}$ is the set of ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Definition 16 guarantees that every $\mathcal{A}_i \in \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ is $\mathcal{T}$-consistent. Hence, if follows that $\mathcal{A}'$ is $\mathcal{T}$-consistent too, which proves the claim. ∎

Proposition 7. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent DL KB, and let F be a set of ABox assertions. We have that $\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F$ is consistent.*

Proof. Let $\langle \mathcal{T}, \mathcal{A}' \rangle$ be the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F. In the case where $\langle \mathcal{T}, \emptyset \rangle \models F$, by Definition 23, we have that $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle$ is equal to $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle$, which is consistent. We now focus on the case where $\langle \mathcal{T}, \emptyset \rangle \not\models F$. Toward a contradiction, suppose that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is not consistent, i.e., $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) = \emptyset$. Proposition 2 guarantees that the set $\mathcal{U}$ of ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ is non-empty. Let $\mathcal{A}''$ be an ABox in $\mathcal{U}$. Since by Definition 23 we have that $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle = \bigcap_{\mathcal{A}^1_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}^1_i)$, then $Mod(\langle \mathcal{T}, \mathcal{A}'' \rangle) \subseteq Mod(\langle \mathcal{T}, \mathcal{A}' \rangle)$ which is a contradiction. ∎

Proposition 5 guarantees that our update operators respect the uniqueness of the result property. A notable consequence of Proposition 5 is that, regardless of the DL $\mathcal{L}$ used to express the original KB, the KB resulting from the application of our update operators can be expressed in $\mathcal{L}$. This means that our update operators also obey the *expressibility* property, as the following propositions states.

Proposition 8. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB in $\mathcal{L}$, and let F be a set of ABox assertions. Then both $\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F$ and $\mathcal{K} \ominus^{\mathcal{T}}_{\cap} F$ are KBs in $\mathcal{L}$.*

Proof. The proof follows from Proposition 5 and from Definition 22 and Definition 23. Indeed, if $\mathcal{U}$ is a set of ABoxes, and $\mathcal{T}$ is a TBox in $\mathcal{L}$, then $\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathcal{A}_i \rangle$ is a KB in $\mathcal{L}$. ∎

In the belief revision community the formula-based approach is often understood as an approach that does not satisfy the *syntax independence* property, namely, the update of logically equivalent KBs should give the same result. In our setting, where we keep fixed the TBox, the *syntax independence* property is rephrased as follows. Let $\mathcal{K}_1 = \langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\mathcal{K}_2 = \langle \mathcal{T}, \mathcal{A}_2 \rangle$ be two (not necessarily different) $\mathcal{L}$-KBs. Let $\mathcal{K}'_1 = \langle \mathcal{T}, \mathcal{A}'_1 \rangle$ be the result of updating $\mathcal{K}_1$ with F, and let $\mathcal{K}'_2 = \langle \mathcal{T}, \mathcal{A}'_2 \rangle$ be the result of updating $\mathcal{K}_2$ with F. If $\mathcal{K}_1$ is logically equivalent to $\mathcal{K}_2$, then $\mathcal{K}'_1$ is logically equivalent to $\mathcal{K}'_2$.

Next propositions show that our update approach captures the *syntax independence* property.

Proposition 9. *Let $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ be two consistent, logically equivalent $\mathcal{L}$-KBs. Let F be a set of ABox assertions. Then, we have that $\langle \mathcal{T}, \mathcal{A}_1 \rangle \oplus^{\mathcal{T}}_{\cap} F$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle \oplus^{\mathcal{T}}_{\cap} F$ are logically equivalent.*

Proof. Let $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ be two logically equivalent $\mathcal{L}$-KBs, and let F be a set of atomic ABox assertions. We prove that $\langle \mathcal{T}, \mathcal{A}_1 \rangle \oplus^{\mathcal{T}}_{\cap} F$ is logically equivalent to $\langle \mathcal{T}, \mathcal{A}_2 \rangle \oplus^{\mathcal{T}}_{\cap} F$.

Let $\langle \mathcal{T}, \mathcal{A}'_1 \rangle$ be the result of updating $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ with the insertion of F, and let $\langle \mathcal{T}, \mathcal{A}'_2 \rangle$ be the result of updating $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ with the insertion of F. The proof proceeds by contradiction. Suppose that $\langle \mathcal{T}, \mathcal{A}'_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}'_2 \rangle$ are not logically equivalent. This means that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'_1) \neq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'_2)$, and then $\bigcap_{\mathcal{A}^1_i \in \mathcal{U}_1} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}^1_i) \neq \bigcap_{\mathcal{A}^2_i \in \mathcal{U}_2} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}^2_i)$, where $\mathcal{U}_1$ and $\mathcal{U}_2$ are respectively the set of ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ minimally, and the set of ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ minimally. It follows that there exists an ABox $\mathcal{A}^1_j \in \mathcal{U}_1$ such that $\mathcal{A}^1_j \notin \mathcal{U}_2$. We know from Theorem 11, that $\mathcal{A}^1_j$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ minimally if and only if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}^1_j) = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1)$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. But since $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$, then $\mathcal{A}''$ is also a maximal subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$. Hence, $\mathcal{A}^1_j$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ minimally, and then $\mathcal{A}^1_j \in \mathcal{U}_2$, which is a contradiction. ∎

Proposition 10. *Let $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ be two consistent, logically equivalent $\mathcal{L}$-KBs. Let F be a set of ABox assertions. Then, we have that $\langle \mathcal{T}, \mathcal{A}_1 \rangle \ominus^{\mathcal{T}}_{\cap} F$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle \ominus^{\mathcal{T}}_{\cap} F$ are logically equivalent.*

Proof. Let $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ be two logically equivalent $\mathcal{L}$-KBs, and let F be a set of atomic ABox assertions. We prove that $\langle \mathcal{T}, \mathcal{A}_1 \rangle \ominus^{\mathcal{T}}_{\cap} F$ is logically equivalent to $\langle \mathcal{T}, \mathcal{A}_2 \rangle \ominus^{\mathcal{T}}_{\cap} F$.

Let $\langle \mathcal{T}, \mathcal{A}'_1 \rangle$ be the result of updating $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ with the deletion of F, and let $\langle \mathcal{T}, \mathcal{A}'_2 \rangle$ be the result of updating $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ with the deletion of F. The proof proceeds by contradiction. Suppose that $\langle \mathcal{T}, \mathcal{A}'_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}'_2 \rangle$ are not logically equivalent. This means that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'_1) \neq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'_2)$, and then $\bigcap_{\mathcal{A}^1_i \in \mathcal{U}_1} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}^1_i) \neq \bigcap_{\mathcal{A}^2_i \in \mathcal{U}_2} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}^2_i)$, where $\mathcal{U}_1$ and $\mathcal{U}_2$ are respectively the set of ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ minimally, and the set of ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ minimally. It follows that there exists an ABox $\mathcal{A}^1_j \in \mathcal{U}_1$ such that $\mathcal{A}^1_j \notin \mathcal{U}_2$. We know from Theorem 12, that $\mathcal{A}^1_j$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ minimally if and only if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}^1_j)$, is a maximal subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1)$ such that $F \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}^1_j)$. But since $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$, then $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}^1_j)$ is also a maximal subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$. Hence, $\mathcal{A}^1_j$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ minimally, which contradicts $\mathcal{A}^1_j \notin \mathcal{U}_2$. ∎

Finally, we discuss how our update approach relates to the *compliance with the minimal-change principle* property. Definition 19 and Definition 20 provide the notion of ABox accomplishing the update minimally. In order to cope with the *multiple results* problem, and to guarantee properties as the *expressibility* property, which are essential for our purposes, we chose to adopt the WIDTIO principle. Following the *WIDTIO principle* our operators combine all the ABoxes accomplishing the update minimally into a single ABox, by considering their intersection. Clearly, in the general case, the adoption of the WIDTIO principle leads to increase the distance between the result of the update and the original KB. It is reasonable to argue that other approaches, as the one coming from the adoption of the *Set-Of-Theories* approach, better capture the *compliance with the minimal-change principle* property, than our approach.

Indeed, the cost to pay in the *WIDTIO approach* for the *expressibility* property is expressed in term of loss of information: in this approach, only the ground facts that are true in all the ABoxes accomplishing the update are retained.

The following theorem determines the relation between our update operators and the ones given by the *SOT approach*.

Theorem 13. *Let $\mathcal{K} = \langle \mathcal{T}, F \rangle$ be a consistent DL KB and let F be a set of ABox assertions. Then*

1. *if $Mod(\langle \mathcal{T}, F \rangle) \neq \emptyset$, then $Mod(\mathcal{K} \oplus_{SOT} F) \subseteq Mod(\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F)$;*
2. *if $\langle \mathcal{T}, \emptyset \rangle \not\models F$, then $Mod(\mathcal{K} \ominus_{SOT} F) \subseteq Mod(\mathcal{K} \ominus^{\mathcal{T}}_{\cap} F)$.*

Proof. We remind that the change impact only on the ABox and therefore the TBox is invariant. We start proving that if $Mod(\langle \mathcal{T}, F \rangle) \neq \emptyset$ then $Mod(\mathcal{K} \oplus_{SOT} F) \subseteq Mod(\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F)$. Let $\mathcal{K} = \langle \mathcal{T}, F \rangle$ be a DL KB and let F be a set of ABox assertions. By Proposition 1 we have that if $Mod(\langle \mathcal{T}, F \rangle) \neq \emptyset$ then the set $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ of the ABoxes accomplishing the insertion of F into $\mathcal{K}$ is not empty. By Definition 22, we have that $\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F = \langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle$. Let $\mathcal{A}_{\cap}$ be an ABox such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_{\cap}) = \mathsf{cl}_{\mathcal{T}}(\bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i))$. Obviously, we have that $Mod(\langle \mathcal{T}, \mathcal{A}_{\cap} \rangle) = Mod(\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F)$. Let α be an ABox assertion. $\alpha \in \mathcal{A}_{\cap}$ if and only if $\alpha \in \mathcal{A}_i$ for $1 \leq i \leq n$. Hence, $\mathcal{A}_{\cap} \subseteq \mathcal{A}_i$. It follows that $Mod(\langle \mathcal{T}, \mathcal{A}_i \rangle) \subseteq Mod(\langle \mathcal{T}, \mathcal{A}_{\cap} \rangle)$ for all $\mathcal{A}_i \in \mathcal{U}$.

An interpretation $\mathcal{I} \in Mod(\mathcal{K} \oplus_{SOT} F)$ if and only if it is a model for every KB $\langle \mathcal{T}, \mathcal{A}_i \rangle \in \mathcal{K} \oplus_{SOT} F$, which means that $Mod(\mathcal{K} \oplus_{SOT} F) \subseteq Mod(\langle \mathcal{T}, \mathcal{A}_i \rangle)$ for all $\mathcal{A}_i \in \mathcal{U}$. Hence, $Mod(\mathcal{K} \oplus_{SOT} F) \subseteq Mod(\langle \mathcal{T}, \mathcal{A}_{\cap} \rangle)$.

The proof that if $\langle \mathcal{T}, \emptyset \rangle \not\models F$, then $Mod(\mathcal{K} \ominus_{\mathcal{G}} F) \subseteq Mod(\mathcal{K} \ominus^{\mathcal{T}}_{\cap} F)$ follows directly from the observations above and from Proposition 2. ∎

In other words, Theorem 13 states that our operator is a sound approximation of the *SOT approach*, in the sense that every fact logically implied by

the result of updating a KB following our approach, is also logically implied by the result of updating the same KB following the *SOT approach*. Clearly, in the general case, the converse does not hold, as the next example shows.

Example 17. Consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where:

$$\begin{array}{ll} \mathcal{T} = \{ & C_1 \sqsubseteq \exists S \quad (id\ C_1\ R) \quad \} \\ \mathcal{A} = \{ & C_1(o), \quad C_2(o') \quad \} \end{array}$$

It is easy to verify that updating $\mathcal{K}$ with the insertion of $F = \{R(o,p), R(o',p)\}$ we obtain:

$\mathcal{K} \oplus_{SOT} F = \{\ \langle \mathcal{T}, \{R(o,p), R(o',p), C_1(o)\}\rangle,\ \langle \mathcal{T}, \{R(o,p), R(o',p), C_1(o')\}\rangle\ \}$
$\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F = \{\ \langle \mathcal{T}, \{R(o,p), R(o',p)\}\rangle\ \}$

Now, consider the FOL sentence $\exists x, y.S(x,y)$. Clearly, we have:

- $\mathcal{K} \oplus_{SOT} F \models \exists x, y.S(x,y)$;
- $\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F \not\models \exists x, y.S(x,y)$.

□

In [62], Katsuno and Mendelzon present some postulates, which they argue must be satisfied by any reasonable operator which permits the update of logic theories.

In what follows, similarly to the works in [46, 99] on the AGM postulates [3] for revision operators, we first present the postulates as given in [62] for proposition logic theory, and then we formulate them for DL KB in our setting where: (*i*) the change affects only the ABox, (*ii*) it is enforced the condition that the KB resulting from the update has the same TBox as the original KB, and (*iii*) only ABoxes formed by atomic membership assertions are considered.

In what follows we use $T \circ p$ to denote the result of updating the KB T with the insertion of the sentence p. The postulates for insertion operator proposed by Katsuno and Mendelzon are:

U1: $T \circ p$ implies p;
U2: if T implies p then $T \circ p$ is equivalent to T;
U3: if both T and p are consistent, then $T \circ p$ is also consistent;
U4: if $T_1 \equiv T_2$ and $p_1 \equiv p_2$, then $T_1 \circ p_1 \equiv T_2 \circ p_2$;
U5: $(T \circ p_1) \wedge p_2$ implies $T \circ (p_1 \wedge p_2)$;
U6: if $T \circ p_1$ implies p_2 and $T \circ p_2$ implies p_1, then $T \circ p_1 \equiv T \circ p_2$;
U7: if T is complete, then $(T \circ p_1) \wedge (T \circ p_2)$ implies $T \circ (p_1 \vee p_2)$;
U8: $(T_1 \vee T_2) \circ p \equiv (T_1 \circ p) \vee (T_2 \circ p)$.

In words, **U1** assures that the theory resulting from the update implies p. Intuitively, the first postulate coincides with the *success of the update* property. Postulate **U2** states that if the sentence p is derivable from $\mathcal{T}$, then the resulting theory is logically equivalent to T. Such postulate agrees with the *compliance with the minimal-change principle* property, since it specifies that no changes have to be made on the original theory if they are not necessary. Postulate **U3** is a condition preventing an update from introducing inconsistency. Postulate **U4** says that update operators should be independent from the syntactical forms of knowledge bases. Clearly, **U4** coincides with our *syntax independence* property. Postulates from **U5** to **U8** point that the change has to be minimal. To be more precise, **U5** says that the result of updating T with the insertion of $p_1 \wedge p_2$, is always implied by the conjunction of p_2 and the update of T with the insertion of p_1. **U6** states that if the update of T with p_1 entails p_2, and the update of T with p_2 entails p_1, then the two updates have the same effect. **U7** is applicable only to *complete* KBs. We remind that a KB T is called *complete* if for every sentence φ, $T \models \varphi$ or $T \models \neg\varphi$. By introducing Postulates **U7**, Katsuno and Mendelzon argue that, in cases where there is no uncertainty over what are the possible worlds, then if a possible world results from both updating T with p_1 and with p_2, then it also should result from updating T with the insertion of $p_1 \vee p_2$. Finally, **U8** states that every possible world described by T has to be changed separately.

Firstly, note that both **U7** and **U8** are not applicable to our setting, since **U7** involves update with a disjunction of atoms, and since **U8** refers to disjunction of KBs, which are generally not allowed in DL languages. Postulates **U1**∼**U6** are rephrased as follows.

U1*: $\langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} F \models F$;
U2*: if $\langle \mathcal{T}, \mathcal{A} \rangle \models F$, then $\langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} F \equiv \langle \mathcal{T}, \mathcal{A} \rangle$;
U3*: if both $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}, F \rangle$ are consistent, then $\langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} F$ is also consistent;
U4*: if $\langle \mathcal{T}, \mathcal{A}_1 \rangle \equiv \langle \mathcal{T}, \mathcal{A}_2 \rangle$, then $\langle \mathcal{T}, \mathcal{A}_1 \rangle \oplus_{\cap}^{\mathcal{T}} F \equiv \langle \mathcal{T}, \mathcal{A}_2 \rangle \oplus_{\cap}^{\mathcal{T}} F$;
U5*: let $\langle \mathcal{T}, \mathcal{A}' \rangle \equiv \langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} F_1$, then $\langle \mathcal{T}, \mathcal{A}' \cup F_2 \rangle \models \langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} (F_1 \cup F_2)$;
U6*: if $\langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} F_1 \models F_2$ and $\langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} F_2 \models F_1$, then $\langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} F_1 \equiv \langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} F_2$;

We have the following proposition.

Proposition 11. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent $\mathcal{L}$-KB, and let F be a set of ABox assertions. Then $\langle \mathcal{T}, \mathcal{A} \rangle \oplus_{\cap}^{\mathcal{T}} F$ satisfies postulates **U1***, **U2***, **U3***, **U4***, and **U6***,*

Proof.

U1* The proof follows directly from Proposition 3.

U2* We prove that if $\langle \mathcal{T}, \mathcal{A} \rangle \models F$, then the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F is logically equivalent to $\langle \mathcal{T}, \mathcal{A} \rangle$. Let $\langle \mathcal{T}, \mathcal{A}' \rangle$ be a consistent $\mathcal{L}$-KB such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Since $\langle \mathcal{T}, \mathcal{A} \rangle \models F$, we have that $F \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Hence, $F \in \langle \mathcal{T}, \mathcal{A}' \rangle$, that is $\langle \mathcal{T}, \mathcal{A}' \rangle \models F$. In addition, we have that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$, which means that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ has no changes with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. From the considerations above it follows that $\mathcal{A}' \in \mathcal{U}$, where $\mathcal{U}$ is the set of ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Toward a contradiction, suppose that $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F \not\equiv \langle \mathcal{T}, \mathcal{A} \rangle$. Only the following two cases are conceivable:

1. there exists an ABox assertion α such that $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F \models \alpha$ and $\langle \mathcal{T}, \mathcal{A} \rangle \not\models F$. Since $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ then $\langle \mathcal{A}' \rangle \not\models \alpha$, and since $\mathcal{A}' \in \mathcal{U}$, then also $\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle$ does not implies α. Hence, we have a contradiction.
2. there exists an ABox assertion α such that $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F \not\models \alpha$ and $\langle \mathcal{T}, \mathcal{A} \rangle \models F$. This means that there exists in $\mathcal{U}$ an ABox $\mathcal{A}''$ such that $\alpha \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$. But, since $\mathcal{A}' \in \mathcal{U}$, then every other ABox appearing in $\mathcal{U}$ must have no changes with respect to $\mathcal{A}$. Therefore, $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ and then $\alpha \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$, which is a contradiction.

U3* The proof follows directly from Proposition 6

U4* The proof follows directly from Proposition 9

U6* We have to prove that if $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F_1 \models F_2$ and $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F_2 \models F_1$, then $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F_1$ is logically equivalent to $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F_2$. Let $\langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle$ respectively be the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F_1, and the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F_2. Let $\mathcal{U}_1$ and $\mathcal{U}_2$ be respectively the set of ABoxes accomplishing the insertion of F_1 into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, and the set of ABoxes accomplishing the insertion of F_2 into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. We prove the claim showing that $\mathcal{U}_1 = \mathcal{U}_2$. First of all, note that: (1) since $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F_1 \models F_2$ then each ABox in $\mathcal{U}_1$ accomplishes the insertion of F_2 into $\langle \mathcal{T}, \mathcal{A} \rangle$; (*ii*) since $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F_2 \models F_1$ then each ABox in $\mathcal{U}_2$ accomplishes the insertion of F_1 into $\langle \mathcal{T}, \mathcal{A} \rangle$. We start proving that $\mathcal{U}_1 \subseteq \mathcal{U}_2$. Toward a contradiction, assume that $\mathcal{A}_1$ is an ABox in $\mathcal{U}_1$, but $\mathcal{A}_1 \notin \mathcal{U}_2$. Since we assume that $\mathcal{A}_1 \notin \mathcal{U}_2$, for each $\mathcal{A}_2 \in \mathcal{U}_2$ we have that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$ has fewer changes than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1)$ with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Hence, $\mathcal{A}_1$ does not accomplish the insertion of F_1 minimally, which contradict that $\mathcal{A}_1 \in \mathcal{U}_1$. Therefore, $\mathcal{A}_1 \in \mathcal{U}_2$. Similarly, we can prove that $\mathcal{U}_2 \subseteq \mathcal{U}_1$.

■

Unfortunately, $\oplus^{\mathcal{T}}_{\cap}$ does not satisfy **U5***, as the following example shows.

Example 18. Let $F_1 = \{C_1(o')\}$ and $F_2 = \{C_2(o)\}$ be two set of ABox assertions, and let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_{A,id}$-KB, where:

$$\begin{array}{lll} \mathcal{T} & = \{ & C_1 \sqsubseteq \neg C_2, \quad (id\ C_1\ R) \quad \} \\ \mathcal{A} & = \{ & C_1(o), \quad R(o,p), \quad R(o',p) \quad \} \end{array}$$

The insertion of $C_1(o')$ violates the identification assertion $(id\ C_1\ R)$, then the ABoxes accomplishing the insertion of F_1 into $\mathcal{K}$ minimally are:

$$\begin{array}{l} \mathcal{A}_1^+ = \{\ C_1(o'), \quad C_1(o), \quad R(o,p)\ \} \\ \mathcal{A}_2^+ = \{\ C_1(o'), \quad C_1(o), \quad R(o',p)\ \} \\ \mathcal{A}_3^+ = \{\ C_1(o'), \quad R(o,p), \quad R(o',p)\ \} \end{array}$$

Hence, the KB $\mathcal{K}'$ resulting from the update is:

$$\begin{array}{lll} \mathcal{T} & = \{ & C_1 \sqsubseteq \neg C_2, \quad (id\ C_1\ R) \quad \} \\ \mathcal{A}' & = \{ & C_1(o') \quad \} \end{array}$$

Adding F_2 to the ABox $\mathcal{A}'$, we do not contradict any assertion in $\mathcal{K}'$ and we obtain the following consistent KB $\mathcal{K}''$.

$$\begin{array}{lll} \mathcal{T} & = \{ & C_1 \sqsubseteq \neg C_2, \quad (id\ C_1\ R) \quad \} \\ \mathcal{A}'' & = \{ & C_1(o'), \quad C_2(o) \quad \} \end{array}$$

We show that $\mathcal{K}'' \not\models \mathcal{K} \oplus_{\cap}^{\mathcal{T}} (F_1 \cup F_2)$. Indeed, only the following ABox accomplishing the insertion of $F_1 \cup F_2$ into $\mathcal{K}$ minimally exists.

$$\mathcal{A}^+ = \{\ C_1(o'), \quad C_2(o), \quad R(o,p), \quad R(o',p)\ \}$$

Then, the KB resulting from the insertion of $F_1 \cup F_2$ into $\mathcal{K}$ is:

$$\begin{array}{lll} \mathcal{T} & = \{ & C_1 \sqsubseteq \neg C_2, \quad (id\ C_1\ R) \quad \} \\ \mathcal{A}''' & = \{ & C_1(o'), \quad C_2(o), \quad R(o,p), \quad R(o',p) \quad \} \end{array}$$

which is not implied by $\mathcal{K}''$. □

An interesting consequence of Example 18 is that, in general, updating a KB $\mathcal{K}$ with the insertion of a set F is different from updating $\mathcal{K}$ with the insertion of all atoms in F in a specific order. It can be shown that the same behavior characterizes the deletion operator.

For what concerns erasure of a sentence from a KB, Katsuno and Mendelzon propose in [62] the following six postulates for propositional logic. $T \bullet p$ denotes the theory resulting from the update of T with the deletion of the sentence p.

E1: T implies $T \bullet p$;
E2: if T implies $\neg p$, then $T \bullet p \equiv T$;
E3: if T is satisfiable and p is not a tautology then $T \bullet p$ does not imply p;
E4: if $T_1 \equiv T_2$ and $p_1 \equiv p_2$, then $T_1 \bullet p_1 \equiv T_2 \bullet p_2$;
E5: if $(T \bullet p) \wedge p$ implies T;
E8: $(T_1 \vee T_2) \bullet p \equiv (T_1 \bullet p) \vee (T_2 \bullet p)$.

Postulate **E1** guarantees that during the deletion process no information which is not implied by the original KB is added to the resulting theory. **E2** assures that the deletion of a sentence p does not influence a KB T if T entails then negation of p. **E3** assures that the result of the update with the deletion of a sentence p does not implies p. Postulate **E4** is the analog of **U4**. It says that delete operator has to be independent from the syntactical forms of the original knowledge base. **E5** is called *recovery postulate* and it says that by adding to $T \bullet p$ the sentence p we obtain a new KB which implies the original KB T. Finally, in the same way of **U8**, **E8** states that every possible world described by T has to be changed separately.

Now we present a reformulation of the above postulates so that they can be applied to our setting. Clearly, as for insertion postulates, we are compelled to renounce to **U8**, since it refers to disjunction of KBs. The reformulated postulates can be found in the following list, where the numbering of each postulate corresponds to the original postulate presented in [62].

E1*: $\langle \mathcal{T}, \mathcal{A} \rangle \models \langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F$;
E2*: if $\langle \mathcal{T}, \mathcal{A} \rangle$ implies $\neg F$, then $\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F \equiv \langle \mathcal{T}, \mathcal{A} \rangle$;
E3*: if $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent and $\langle \mathcal{T}, \emptyset \rangle \not\models F$ then $\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F \not\models F$;
E4*: if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$, then $\langle \mathcal{T}, \mathcal{A}_1 \rangle \ominus^{\mathcal{T}}_{\cap} F \equiv \langle \mathcal{T}, \mathcal{A}_2 \rangle \ominus^{\mathcal{T}}_{\cap} F$;
E5*: let $\langle \mathcal{T}, \mathcal{A}' \rangle \equiv \langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F$, then $\langle \mathcal{T}, \mathcal{A}' \cup F \rangle \models \langle \mathcal{T}, \mathcal{A} \rangle$.

Note that **E3** is clearly linked to the *success of the update* property. In order to rephrase it in our setting, where the change affects only the ABox, we have substituted the condition stating that F is not a tautology, with the condition stating that F is not implied by the sole TBox.

The following proposition states that our approach for updating a KB with the deletion of a set of ABox assertions satisfies only four of the postulates above.

Proposition 12. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent $\mathcal{L}$-KB, and let F be a set of ABox assertions. Then $\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F$ satisfies postulates **E1***, **E2***, **E3***, and **E4***.*

Proof.

E1* If $\langle \mathcal{T}, \emptyset \rangle \models F$, then by Definition 23 the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F is $\langle \mathcal{T}, \mathcal{A} \rangle$ itself. Hence, the claim is proved. If $\langle \mathcal{T}, \emptyset \rangle \not\models$

F, then, by Proposition 2, the set $\mathcal{U}$ of all ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally is not empty. In this case the proof is a direct consequence of Theorem 12 stating that if $\mathcal{A}_i$ is an ABox in $\mathcal{U}$ then $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i)$ is a subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Hence, $\bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Thus, $\langle \mathcal{T}, \mathcal{A} \rangle \models \langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F$.

E2* We show that, if $\langle \mathcal{T}, \mathcal{A} \rangle \models \neg F$, then the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F is logically equivalent to $\langle \mathcal{T}, \mathcal{A} \rangle$. Let $\mathcal{U}$ be the set of all ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Trivially, since $\langle \mathcal{T}, \mathcal{A} \rangle \models \neg F$, then $\mathcal{A} \in \mathcal{U}$. Toward a contradiction, suppose that there exists another ABox $\mathcal{A}' \in \mathcal{U}$, such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \neq \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. But, since $\mathcal{A} \in \mathcal{U}$, then the closure with respect to $\mathcal{T}$ of every other ABox in $\mathcal{U}$ must have no changes with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$, and this contradicts that $\mathcal{A}' \in \mathcal{U}$. Thus, $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$.

E3* The proof follows directly from Proposition 4.

E4* The proof follows directly from Proposition 10.

■

We have shown that our deletion operator obeys Postulate **E2***, which guarantees that if the original KB $\mathcal{K}$ implies the negation of F, then the deletion operator has no effects on $\mathcal{K}$. The next lemma shows that the same holds if the original KB does not imply F.

Lemma 19. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a KB and let F be a set of ABox assertions such that $\langle \mathcal{T}, \mathcal{A} \rangle \not\models F$. Then $\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F \equiv \langle \mathcal{T}, \mathcal{A} \rangle$.*

Proof. The proof proceeds as for **E2***. Since $\langle \mathcal{T}, \mathcal{A} \rangle \models \neg F$, then $\mathcal{A} \in \mathcal{U}$, where $\mathcal{U}$ is the set of ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Suppose that there exists ABox $\mathcal{A}' \in \mathcal{U}$, such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \neq \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Since $\mathcal{A} \in \mathcal{U}$, then the closure with respect to $\mathcal{T}$ of every other ABox in $\mathcal{U}$ must have no changes with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$, and this contradicts $\mathcal{A}' \in \mathcal{U}$. Thus, $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. ■

Under our update semantics, the operation of updating a KB with the deletion of a set of ABox assertions, may cause an irreversible loss of information. This gives an explanation to why our deletion operator does not satisfy **E5***. The next example illustrates the matter.

Example 19. Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a KB, where:

$$\begin{array}{lll} \mathcal{T} & = \{ & C_1 \sqsubseteq C_2 \quad \} \\ \mathcal{A} & = \{ & C_1(o) \quad \} \end{array}$$

Suppose to update $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of $F = \{C_2(o)\}$. For this purpose, since $\langle \mathcal{T}, \{C_1(o)\} \rangle \models C_2(o)$, we need to remove $C_1(o)$ from $\mathcal{A}$. Then the KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ resulting from the change is:

$$\begin{array}{lll} \mathcal{T} & = \{ & C_1 \sqsubseteq C_2 \quad \} \\ \mathcal{A} & = & \emptyset \end{array}$$

Clearly, $\langle \mathcal{T}, \mathcal{A}' \cup F \rangle \not\models \langle \mathcal{T}, \mathcal{A} \rangle$ since neither $\langle \mathcal{T}, \mathcal{A}' \rangle$ nor F "remember" $C_1(o)$. □

6.6 Comparison with related work

We mentioned in the introduction several model-based approaches to update DL KBs, and noticed that they all suffer from the expressibility problem. This problem is also shared by [109], that uses *features* instead of models, and proposes the notion of approximation to cope with the expressibility problem, similarly to [40].

In what follows, we briefly recall those model-based approaches that are proposed as suitable for update of knowledge bases [62], and we attempt to rephrase then into the context of updating DL KB at the instance level. More extensive discussions about model-based approaches can be found in [33, 42, 90].

Within the framework of model-based update approaches, the objects of change are models. Consider a DL KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and a set of ABox assertions F. We denote with $\mathcal{K} \circ F$ the update of $\mathcal{K}$ with the insertion on F. The basic idea of model-based approaches is that, for each model $\mathcal{I}_\mathcal{K}$ of $\mathcal{K}$, the update operator must select those models of F which are *closest* to $\mathcal{I}_\mathcal{K}$. The models of $\mathcal{K} \circ F$ are the union of these selected models.

Intuitively the result of updating a KB $\mathcal{K}$ with a finite set of extensional assertions F, should be that KB which logically implies all assertions in F, and whose set of models minimally differs from the set of models of $\mathcal{K}$.

In order to specify the notion of *distance between models*, we need to introduce the notion of *difference between interpretations* [40]. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two interpretation over the same signature $\mathcal{S}$, and let S indicate an atomic concept, an atomic role, or an atomic concept attribute in $\mathcal{S}$. We define the *difference between* $\mathcal{I}_1$ and $\mathcal{I}_2$, written $\mathcal{I}_1 \triangle \mathcal{I}_2$, as the interpretation $(\Delta, \cdot^{\mathcal{I}_1 \triangle \mathcal{I}_2})$ such that $S^{\mathcal{I}_1 \triangle \mathcal{I}_2} = (S^{\mathcal{I}_1} \cup S^{\mathcal{I}_2}) \setminus (S^{\mathcal{I}_1} \cap S^{\mathcal{I}_2})$, for every $S \in \mathcal{S}$.

In [15], Borgida defines his update operator using a distance metric based on the notion of *containment between interpretations*. We say that $\mathcal{I}_1$ is *contained* in $\mathcal{I}_2$, written $\mathcal{I}_1 \subseteq \mathcal{I}_2$, if $S^{\mathcal{I}_1} \subseteq S^{\mathcal{I}_2}$ for all $S \in \mathcal{S}$. Moreover, we say that $\mathcal{I}_1$ *is properly contained* $\mathcal{I}_2$, written $\mathcal{I}_1 \subset \mathcal{I}_2$, if $\mathcal{I}_1 \subseteq \mathcal{I}_2$ and $\mathcal{I}_2 \not\subseteq \mathcal{I}_1$.

Let $\mathcal{T}$ be a TBox in a DL $\mathcal{L}$, $\mathcal{I}$ a model of $\mathcal{T}$, and F a finite set of ABox assertions. The update of $\mathcal{I}$ with the insertion of F with respect to $\mathcal{T}$ under containment, written $U^{\mathcal{T}}_{\subseteq}(\mathcal{I}, F)$, is the set of models defined as follows:

$$U^{\mathcal{T}}_{\subseteq}(\mathcal{I}, F) = \{\mathcal{I}' \mid \mathcal{I}' \in Mod(\langle \mathcal{T}, F\rangle) \text{ and there exists no } \mathcal{I}'' \in Mod(\langle \mathcal{T}, F\rangle) \text{ s.t. } \mathcal{I}\triangle\mathcal{I}'' \subset \mathcal{I}\triangle\mathcal{I}'\}$$

The KB $\langle \mathcal{T}, \mathcal{A}\rangle \circ_B F$ resulting from the update of $\langle \mathcal{T}, \mathcal{A}\rangle$ with the insertion of F according to Borgida is:

if $Mod(\langle \mathcal{T}, \mathcal{A} \cup F\rangle) \neq \emptyset$
then $\langle \mathcal{T}, \mathcal{A}\rangle \circ_B F = \langle \mathcal{T}, \mathcal{A} \cup F\rangle$;
else $Mod(\langle \mathcal{T}, \mathcal{A}\rangle \circ_B F) = \bigcup_{\mathcal{I} \in Mod(\langle \mathcal{T}, \mathcal{A}\rangle)} U^{\mathcal{T}}_{\subseteq}(\mathcal{I}, F)$.

Winslett [110], proposed the so called *possible worlds approach* (PMA), defined within the context of reasoning about actions. In the PMA approach is adopted the same distance metric of Borgida. But their approach differs in case new facts are consistent with the original KB. More formally, according to the PMA approach, the KB $\langle \mathcal{T}, \mathcal{A}\rangle \circ_{PMA} F$ resulting from updating $\langle \mathcal{T}, \mathcal{A}\rangle$ with the insertion of F is:

$$Mod(\langle \mathcal{T}, \mathcal{A}\rangle \circ_{PMA} F) = \bigcup_{\mathcal{I} \in Mod(\langle \mathcal{T}, \mathcal{A}\rangle)} U^{\mathcal{T}}_{\subseteq}(\mathcal{I}, F).$$

Both Borgida and Winslett adopt a distance metric based on the *containment between interpretations*. Differently, the approach presented by Forbus in [47], takes into account cardinality.

Let $\mathcal{T}$ be a TBox in a DL $\mathcal{L}$, $\mathcal{I}$ a model of $\mathcal{T}$, and F a finite set of ABox assertions. The update of $\mathcal{I}$ with the insertion of F with respect to $\mathcal{T}$ under cardinality, written $U^{\mathcal{T}}_{\sharp}(\mathcal{I}, F)$, is the set of models defined as follows:

$$U^{\mathcal{T}}_{\sharp}(\mathcal{I}, F) = \{\mathcal{I}' \mid \mathcal{I}' \in Mod(\langle \mathcal{T}, F\rangle) \text{ and there exists no } \mathcal{I}'' \in Mod(\langle \mathcal{T}, F\rangle) \text{ s.t. } |\mathcal{I}\triangle\mathcal{I}''| < |\mathcal{I}\triangle\mathcal{I}'|\}$$

The KB $\langle \mathcal{T}, \mathcal{A}\rangle \circ_B F$ resulting from the update of $\langle \mathcal{T}, \mathcal{A}\rangle$ with the insertion of F according to the Forbus approach is:

$$Mod(\langle \mathcal{T}, \mathcal{A}\rangle \circ_F F) = \bigcup_{\mathcal{I} \in Mod(\langle \mathcal{T}, \mathcal{A}\rangle)} U^{\mathcal{T}}_{\sharp}(\mathcal{I}, F).$$

Observe that, by means of cardinality, Forbus can compare (and discard) models which are incomparable in both Borgida and Winslett approaches.

By following the model-based approaches presented above, the result of the update in defined in term of models. The principal issue in adapting such approaches to DL KBs is how construct the KB having as set of models the one resulting from the update. Several works [33, 40, 63, 78] point out that

various DLs are not closed with respect to the update operation under model-based approaches. In other words, it happens that, given a KB $\mathcal{K}$ in a DL $\mathcal{L}$, a finite set of ABox assertions F, and the set of models $\mathbb{M}$ resulting from updating $\mathcal{K}$ with the insertion of F, it is not ensure that a KB $\mathcal{K}'$ in $\mathcal{L}$ exists such that $Mod(\mathcal{K}') = \mathbb{M}$. Hence model-based approaches do not capture the *expressibility* property.

In the following example we show that the DL *DL-Lite*$_{A,id}$ is not closed with respect to the PMA update semantics.

Example 20. Consider the following *DL-Lite*$_{A,id}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. Note that $\mathcal{T}$ is a portion of the TBox of the KB presented in Example 8.

$$\begin{array}{lll} \mathcal{T} = \{ & RD \sqsubseteq \neg TM, \quad TM \sqsubseteq \neg FT, \quad TM \sqsubseteq \exists mf, \quad RD \sqsubseteq \neg FT & \\ & FT \sqsubseteq \exists mf^-, \quad \exists mf \sqsubseteq TM, \quad \exists mf^- \sqsubseteq FT & \} \\ \mathcal{A} = \{ & RD(sam), \quad RD(bob), \quad TM(tim), \quad FT(t_1) \quad mf(tim, t_1) & \} \end{array}$$

In words, the ABox $\mathcal{A}$ states that both *sam* and *bob* are racer directors, *tim* is a team member, and that t_1 is a formula one team. Moreover, *tim* is member of t_1. Assume that *tim* lives t_1 to become a racer director, which means update $\mathcal{K}$ with the insertion of $\{RD(tim)\}$. Note that, in each model of $\mathcal{K}$ both *sam* and *bob* must be racer directors. The following interpretation $\mathcal{I}$ is a model of $\mathcal{K}$.

$$\mathcal{I} : RD^{\mathcal{I}} = \{sam, bob\}, \quad TM^{\mathcal{I}} = \{tim\}, \quad FT^{\mathcal{I}} = \{t_1\} \quad mf^{\mathcal{I}} = \{(tim, t_1)\}$$

Consider the following models of $\mathcal{T}$.

$$\mathcal{I}'_1 : RD^{\mathcal{I}'_1} = \{tim, sam\},\ TM^{\mathcal{I}'_1} = \{bob\},\ FT^{\mathcal{I}'_1} = \{t_1\},\ mf^{\mathcal{I}'_1} = \{(bob, t_1)\}$$
$$\mathcal{I}'_2 : RD^{\mathcal{I}'_1} = \{tim, bob\},\ TM^{\mathcal{I}'_1} = \{sam\},\ FT^{\mathcal{I}'_1} = \{t_1\},\ mf^{\mathcal{I}'_1} = \{(sam, t_1)\}$$

Both $\mathcal{I}'_1$ and $\mathcal{I}'_2$ are model of $\langle \mathcal{T}, \{RD(tim)\} \rangle$ that differs minimally from $\mathcal{I}$, which means that both $\mathcal{I}'_1$ and $\mathcal{I}'_2$ are in $U^{\mathcal{T}}_{\subseteq}(\mathcal{I}, \{RD(tim)\})$. Note that $\mathcal{I}'_1$ does not satisfy $RD(bob)$ and $\mathcal{I}'_2$ does not satisfy $RD(sam)$. Since each interpretation $\mathcal{I}'$ in $U^{\mathcal{T}}_{\subseteq}(\mathcal{I}, \{RD(tim)\})$ must differ minimally from $\mathcal{I}$, then the following three cases are possible:

- $\mathcal{I}'$ supports $RD(bob)$ but not $RD(sam)$;
- $\mathcal{I}'$ supports $RD(sam)$ but not $RD(bob)$;
- $\mathcal{I}'$ supports both $RD(bob)$ and $RD(sam)$.

It follows that each KB representing the update of $\mathcal{K}$ with the insertion of $\{RD(tim)\}$ should imply $RD(sam) \vee RD(bob)$ and neither the single assertion $RD(sam)$ nor $RD(bob)$. But this is impossible for a *DL-Lite*$_{A,id}$-KB. □

Related to our proposal are several formula-based approaches proposed in the literature. We already pointed out that our definition of *fewer changes* inspired by the analysis of update semantics for logical databases presented by Fagin, Ullman, and Vardi in [44]. Their update approach, which gives origin to the *Set-Of-Theories* approach, can be formalized as follows. Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{L}$-KB, and let F be a set of ABox assertions. We denote with $\sigma_{\mathcal{T}}(\mathcal{A}, F)$ the set of all maximal subsets $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{A}' \cup F$ is $\mathcal{T}$-consistent. The result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F according to [44], denoted by $\langle \mathcal{T}, \mathcal{A} \rangle \circ_{FUV} F$, is:

$$\langle \mathcal{T}, \mathcal{A} \rangle \circ_{FUV} F \quad = \quad \{\langle \mathcal{T}, \mathcal{A}_i \cup F \rangle \mid \mathcal{A}_i \in \sigma_{\mathcal{T}}(\mathcal{A}, F)\}.$$

Independently by [44], Ginsberg [52] proposes the same change semantics. Note that, differently from our approach, they refer to the ABox of the original KB, instead of its deductive closure. In what follows we briefly discuss such a choice. When a KB is changed by an update operation, some facts could be retracted. In such a case, they claim that also the consequences of the retracted facts should be removed if they are not supported otherwise, i.e., they perform *reason maintenance* [86]. For instance, consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ presented in Example 8. Since the TBox $\mathcal{T}$ specifies that test drivers (TD) are team members (TM), and since the ABox asserts that *bob* is a test driver. We have that both the facts $TD(bob)$ and $TM(bob)$ are entailed by $\mathcal{K}$. Suppose that we do not know anymore if it is true that *bob* is a test driver. Then, we update $\mathcal{K}$ with the deletion of $F = \{TD(bob)\}$, which means that we are forced to retract the fact $TD(bob)$. Since no other fact in $\mathcal{A}$ entails with $\mathcal{T}$ the fact $TM(bob)$, then, by enforcing the *reason maintenance*, the KB resulting from the update should not entail $TM(bob)$. On the contrary, by adopting our approach, where we aim to preserve as much as possible the facts entailed by the original KB, we attempt to keep the fact $TM(bob)$ if it does not act against the update. We are not convinced that *reason maintenance* has to be considered as one of the fundamental properties that an update operator for KBs should possess. As a matter of fact, we belief that *reason maintenance* can result from an adequate choice of the facts adopted to update the original KB. Coming back to the example above. We act a deletion driven by the fact that "*we do not know anymore if it is true that bob is a test driver*", which is different to assert that "*we do not know anymore if it is true that bob is a team member*". By promoting the last assertion, we need to update $\mathcal{K}$ with the deletion of $\{TM(bob)\}$. Hence, we are forced to retract both $TM(bob)$ and $TD(bob)$. In [86], Nebel provide a complete discussion on the *reason maintenance* and its relation with *belief revision*. Moreover, it studies the employment of formula-based approaches on deductively closed theories, and shows that formula-based change operators acting on deductively closed theories satisfy AGM postulates [3].

Perhaps, in reference to the fact that we refer to the closure of the ABox of a KB, rather than to the ABox itself, the closest approach to the one proposed in this work is that reported in [33], where formula-based evolution (actually, insertion) of *DL-Lite*-KBs is studied. The main difference with our work is that we cope with the *multiple results* problem adopting the WIDTIO principles, and therefore we take, as ABox of the KB resulting from the evolution, the intersection of all ABoxes accomplishing the change minimally. Conversely, in the *bold* semantics discussed in [33], the ABox of the result is chosen non-deterministically among the ABoxes accomplishing the change minimally. Another difference is that while [33] addresses the issue of evolution of both the TBox and the ABox, we only deal with the case of fixed TBox (in the terminology of [33], this corresponds to keep the TBox *protected*). It is interesting to observe that the specific DL considered in [33] is $\mathit{DL\text{-}Lite}_{F\mathcal{R}}$, and for this logic, exactly one KB accomplishes the insertion of a set of ABox assertions minimally exists. It follows that for instance-level insertion, their bold semantics coincides with ours. On the other hand, remaining within the *DL-Lite* family, the presence of identification assertions in $\mathit{DL\text{-}Lite}_{A,id}$ and, even more, the presence of denial assertions in $\mathit{DL\text{-}Lite}_{A,id,den}$, changes the picture considerably, since with such assertions in the TBox, many ABoxes may exist accomplishing the insertion (or deletion) minimally. In this case, the two approaches are very different. Finally, [33] proposes a variant of the bold semantics, called *careful semantics*, for instance-level insertion in $\mathit{DL\text{-}Lite}_{F\mathcal{R}}$. Intuitively, such semantics aims to realize a weak form *reason maintenance* by disregarding knowledge that is entailed neither by the original KB, nor by the set of newly asserted facts, as shown in the following example.

Example 21. Consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ of Example 8, and suppose that *john* becomes the official driver of the team t_2, which means changing $\mathcal{K}$ with the insertion of $\{OD(john), mf(john, t_2)\}$. Notice that, since the TBox implies that every team member is a member of exactly one team, $mf(john, t_2)$ contradicts $mf(john, t_1)$. Therefore, in our approach, the result of the insertion is the KB $\langle \mathcal{T}, \mathcal{A}' \rangle$, where:

$$\mathcal{A}' = \{ \quad OD(john), \quad mf(john, t_2), \quad TD(bob), \quad TM(tom), \quad FT(t_1) \quad \}$$

Conversely, one can verify that the result under the careful semantics is the KB $\langle \mathcal{T}, \mathcal{A}'' \rangle$, where:

$$\mathcal{A}'' = \{ \quad OD(john), \quad mf(john, t_2), \quad TD(bob), \quad TM(tom) \quad \}$$

which loses the information that t_1 is a formula one team. □

Chapter 7

Updating consistent *DL-Lite*$_{A,id,den}$ KBs

In this chapter we study the problem of updating consistent KBs expressed in *DL-Lite*$_{A,id,den}$ according to the semantics proposed in previous chapter. As shown in Chapter 4, this logic is very attractive, because in *DL-Lite*$_{A,id,den}$, as for the other logics of the *DL-Lite* family [31], the trade-off between expressive power and computational complexity of reasoning is optimized towards the needs that arise in *ontology-based data access*. In fact, if on one hand this logic is able to capture the most important features of the conceptual modeling formalism, allowing also for specifying particularly useful general forms of disjointness, on the other hand, in this logic query answering can be managed efficiently with respect to the size of the ABox.

Finally, we provide two algorithms for computing respectively the KB resulting from the update by insertion and from the update by deletion of a consistent *DL-Lite*$_{A,id,den}$-KB. We prove that these algorithms are correct, and we show that they run in polynomial time with respect to the size of the original ABox.

7.1 Update by insertion in *DL-Lite*$_{A,id,den}$

In this section we study insertion under the assumption that the DL language $\mathcal{L}$ is *DL-Lite*$_{A,id,den}$. Thus, in what follows, we implicitly refer to a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and we address the problem of updating $\mathcal{K}$ with the insertion of a finite set F of ABox assertions. We also assume that $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent.

By assuming that $\langle \mathcal{T}, F \rangle$ is consistent, Theorem 11 tells us that, in principle, we can compute the KB resulting from the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ by building all maximal subsets $\mathcal{A}''$ of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent,

and then computing their intersection. The main problem to be faced with this method is that, depending on the DL used, there can be an exponential number of maximal subsets $\mathcal{A}''$ of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. Note that this cannot happen in those DLs of the *DL-Lite* family which do not admit the use of identification assertions and denial assertions, such as the DL studied in [33]. In particular, in *DL-Lite*$_{A,id,den}$, building all maximal subsets $\mathcal{A}''$ of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent, and then computing their intersection is computationally costly. Fortunately, we show in what follows that $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F$ can be computed without computing all maximal subsets $\mathcal{A}''$ of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent.

We recall that, if $\mathcal{T}$ is a consistent TBox, then a set V of ABox assertions is called a *$\mathcal{T}$-inconsistent set* if V is $\mathcal{T}$-inconsistent. Moreover, $cln(\mathcal{T})$ denotes the NI-closure of $\mathcal{T}$, which is the set of TBox assertion built from the assertions in $\mathcal{T}$ as shown in Definition 14.

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent *DL-Lite*$_{A,id,den}$-KB, and let F be a set of ABox assertions. As shown in the previous chapter, it may happen that the KB $\langle \mathcal{T}, \mathcal{A} \cup F \rangle$ in inconsistent. We observe that in this case, an inconsistency may arise in $\langle \mathcal{T}, \mathcal{A} \cup F \rangle$ only for one of the following reasons (the enumeration of the items in the list below follows the enumeration of the list given in Section 5.1):

7. there exist an atomic role P in the TBox alphabet $\Gamma_{\mathcal{O}}$, constants d, d_1, d_2 in the alphabet of constants Γ_C, a TBox assertion (funct P) (resp. (funct P^-)) in $cln(\mathcal{T})$, and the ABox assertions $P(d, d_1)$ and $P(d, d_2)$ (resp. $P(d_1, d)$ and $P(d_2, d)$) belong to $\mathcal{A} \cup F$;

8. there exist an attribute U in the TBox alphabet $\Gamma_{\mathcal{O}}$, constants d, v_1, v_2 in the alphabet of constants Γ_C, a TBox assertion (funct U) in $cln(\mathcal{T})$, and the ABox assertions $U(d, v_1)$ and $U(d, v_2)$ belongs to $\mathcal{A} \cup F$;

9. there exist a pair of basic concepts B_1 and B_2, a constant d in the alphabet of constants Γ_C, a TBox assertion $B_1 \sqsubseteq \neg B_2$ in $cln(\mathcal{T})$, and a pair of ABox assertions α and β in $\mathcal{A} \cup F$ such that $\langle \mathcal{T}_{inc}, \{\alpha\} \rangle \models B_1(d)$ and $\langle \mathcal{T}_{inc}, \{\beta\} \rangle \models B_2(d)$;

10. there exist a pair of basic roles Q_1 and Q_2, a pair of constants d_1 and d_2 in the alphabet of constants Γ_C, a TBox assertion $Q_1 \sqsubseteq \neg Q_2$ in $cln(\mathcal{T})$, and a pair of ABox assertions α and β in $\mathcal{A} \cup F$ such that $\langle \mathcal{T}_{inc}, \{\alpha\} \rangle \models Q_1(d_1, d_2)$ and $\langle \mathcal{T}_{inc}, \{\beta\} \rangle \models Q_2(d_1, d_2)$;

11. there exist a pair of attributes U_1 and U_2 in the TBox alphabet $\Gamma_{\mathcal{O}}$, a pair of constants d_1 and d_2 in the alphabet of constants Γ_C, a TBox assertion $U_1 \sqsubseteq \neg U_2$ in $cln(\mathcal{T})$, and a pair of ABox assertions α and β in $\mathcal{A} \cup F$ such that $\langle \mathcal{T}_{inc}, \{\alpha\} \rangle \models U_1(d_1, d_2)$ and $\langle \mathcal{T}_{inc}, \{\beta\} \rangle \models U_2(d_1, d_2)$;

12. there exist an identification assertion $\alpha \in cln(\mathcal{T})$ and a set of ABox assertions $V \subseteq \mathcal{A} \cup F$ such that $\langle \mathcal{T}_{inc}, V \rangle \models \neg\alpha$, i.e. $\langle \mathcal{T}_{inc}, V \rangle$ implies the negation of α;

13. there exist a denial assertion $\alpha \in cln(\mathcal{T})$ and a set of ABox assertions $V \subseteq \mathcal{A} \cup F$ such that $\langle \mathcal{T}_{inc}, V \rangle \models \neg\alpha$, i.e. $\langle \mathcal{T}_{inc}, V \rangle$ implies the negation of α;

Note that, each violation requires the existence of a $\mathcal{T}$-inconsistent set formed by at least two ABox assertions. Obviously, at least one ABox assertions, among the ones that play a role in the violation of an assertion in $cln(\mathcal{T})$, belongs to F. A consequence of the observations above, is that the assertion in $\mathcal{T}_{type}$ cannot be violated by an insertion.

In order to compute the KB resulting from the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, we need to compute the intersection of the closure with respect to $\mathcal{T}$ of all ABoxes accomplishing the insertion minimally. To this purpose, we exploit the following property of *DL-Lite*$_{A,id,den}$ which follows directly from Lemma 16. Let $\mathcal{T}$ be a *DL-Lite*$_{A,id,den}$ TBox and let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two ABoxes such that $\mathcal{A}_1 \cup \mathcal{A}_2$ is $\mathcal{T}$-consistent. Then,

$$\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1 \cup \mathcal{A}_2) = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) \cup \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2).$$

We know from Theorem 11 that the ABox $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally if and only if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent.

Let $\mathcal{U} = \{\mathcal{A}'_1, \ldots, \mathcal{A}'_n\}$ be the set of all ABoxes accomplishing the insertion of F into $\mathcal{K}$ minimally, and let $\mathcal{M} = \{\mathcal{A}''_1, \ldots, \mathcal{A}''_n\}$ be the set of all maximal subsets of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that, for all $\mathcal{A}'' \in \mathcal{M}$, $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. We have that:

$$\bigcap_{\mathcal{A}'_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'_i) = \bigcap_{\mathcal{A}''_j \in \mathcal{M}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}''_j \cup F) = \mathsf{cl}_{\mathcal{T}}(F) \cup \bigcap_{\mathcal{A}''_j \in \mathcal{M}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}''_j)$$

By exploiting Lemma 18 we have:

$$\bigcap_{\mathcal{A}'_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'_i) = \mathsf{cl}_{\mathcal{T}}(F) \cup \bigcap_{\mathcal{A}''_j \in \mathcal{M}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}''_j) = \mathsf{cl}_{\mathcal{T}}(F) \cup \bigcap_{\mathcal{A}''_j \in \mathcal{M}} \mathcal{A}''_j$$

Then, in order to compute $\bigcap_{\mathcal{A}'_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'_i)$ it is sufficient to compute all maximal subsets $\mathcal{A}''_j$ of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}''_j \cup F$ is $\mathcal{T}$-consistent, and then compute their intersection. In other words, we have to individuate such assertions in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ that are not in $\bigcap_{\mathcal{A}''_j \in \mathcal{M}} \mathcal{A}''_j$, and remove them from $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$. Since for every ABox $\mathcal{A}''$ in $\mathcal{M}$ we have that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent, then all the assertions in $\mathsf{cl}_{\mathcal{T}}(F) \cap \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ are in $\bigcap_{\mathcal{A}''_j \in \mathcal{M}} \mathcal{A}''_j$.

As for the ABox assertions in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(F)$, it is easy to see that one such assertion α is not in $\bigcap_{\mathcal{A}''_j \in \mathcal{M}} \mathcal{A}''_j$ if and only if there exists a maximal subset Σ of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\Sigma \cup F$ is $\mathcal{T}$-consistent, and Σ does not contain α.

Taking into account the above observation, the next theorem is the key to our solution.

Theorem 14. *Let α be in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(F)$. There exists a maximal subset Σ of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\Sigma \cup F$ is $\mathcal{T}$-consistent, and Σ does not contain α if and only if there is a $\mathcal{T}$-inconsistent set V in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ such that $\alpha \in V$ and $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent.*

Proof.
($\Rightarrow$) Let α be an assertion in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(F)$, and let Σ be a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that: (*i*) $\Sigma \cup F$ is $\mathcal{T}$-consistent; (*ii*) $\alpha \notin \Sigma$. We prove that exists a $\mathcal{T}$-inconsistent set V in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ such that: (*i*) $\alpha \in V$; (*ii*) $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent. Toward a contradiction, suppose that there is no $\mathcal{T}$-inconsistent set V in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ such that: (*i*) $\alpha \in V$; (*ii*) $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent. Firstly we inspect the case where no $\mathcal{T}$-inconsistent set V in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ containing α exists. This means that for each $\mathcal{T}$-consistent subset $\mathcal{A}'$ of $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ we have that $\langle \mathcal{T}, \mathcal{A}' \cup \{\alpha\} \rangle$ is consistent. Hence, every maximal subset Σ' of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$, such that $\Sigma' \cup F$ is $\mathcal{T}$-consistent, contains α. This contradicts the fact that $\alpha \notin \Sigma$.

We turn to the case where for each $\mathcal{T}$-inconsistent set V in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ such that $\alpha \in V$, we have that $F \cup (V \setminus \{\alpha\})$ is not $\mathcal{T}$-consistent. It directly follows that for each subset Σ' of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\Sigma' \cup F$ is $\mathcal{T}$-consistent, $\Sigma' \cup F \cup \{\alpha\}$ is $\mathcal{T}$-consistent. Hence, $\Sigma \cup F \cup \{\alpha\}$ is $\mathcal{T}$-consistent, but this contradicts the fact that Σ is maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\Sigma \cup F$ is $\mathcal{T}$-consistent.
($\Leftarrow$) Suppose that there is a $\mathcal{T}$-inconsistent set in V in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ such that $\alpha \in V$ and $\langle \mathcal{T}, F \cup (V \setminus \{\alpha\}) \rangle$ is consistent. Since $\langle \mathcal{T}, F \cup (V \setminus \{\alpha\}) \rangle$ is consistent, the set of maximal subsets Σ of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \Sigma \cup F \cup (V \setminus \{\alpha\}) \rangle$ is consistent is non-empty. Consider any Σ in such a set, i.e., assume that Σ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \Sigma \cup F \cup (V \setminus \{\alpha\}) \rangle$ is consistent. It can be shown that (1) Σ does not contain α, and (2) Σ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \Sigma \cup F \rangle$ is consistent.

(1) Since Σ is a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \Sigma \cup F \cup (V \setminus \{\alpha\}) \rangle$ is consistent, it follows that Σ does not contain α, otherwise $\Sigma \cup F \cup (V \setminus \{\alpha\})$ would contain V, and therefore $\langle \mathcal{T}, \Sigma \cup F \cup (V \setminus \{\alpha\}) \rangle$ would be inconsistent.

(2) We have to prove that, for every $\beta \in \mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \Sigma$, $\langle \mathcal{T}, F \cup \Sigma \cup \{\beta\} \rangle$ is inconsistent. First of all, notice that, since Σ is a maximal subset

of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \Sigma \cup F \cup (V \setminus \{\alpha\})\rangle$ is consistent, we have that $(V \setminus \{\alpha\}) \cap \mathsf{cl}_\mathcal{T}(\mathcal{A}) \subseteq \Sigma$. This implies that, adding α to Σ yields the $\mathcal{T}$-inconsistent set V, and this means that $\langle \mathcal{T}, \Sigma \cup F \cup \{\alpha\}\rangle$ is inconsistent. Let $\beta \in \mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \Sigma$ be different from α. Consider $\Sigma \cup \{\beta\}$. Since $(V \setminus \{\alpha\}) \cap \mathsf{cl}_\mathcal{T}(\mathcal{A}) \subseteq \Sigma$, the fact that $\langle \mathcal{T}, \Sigma \cup F \cup \{\beta\}\rangle$ is consistent implies that $\langle \mathcal{T}, \Sigma \cup F \cup (V \setminus \{\alpha\}) \cup \{\beta\}\rangle$ is consistent, i.e., that Σ is not a maximal subset of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \Sigma \cup F \cup (V \setminus \{\alpha\})\rangle$ is consistent, which is a contradiction. Therefore, we conclude that $\langle \mathcal{T}, \Sigma \cup F \cup \{\beta\}\rangle$ is inconsistent.

In conclusion, we have proved that, if there is a $\mathcal{T}$-inconsistent set V in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ such that $\alpha \in V$, and $\langle \mathcal{T}, F \cup (V \setminus \{\alpha\})\rangle$ is consistent, then there exists a maximal subset Σ of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \Sigma \cup F\rangle$ is consistent and Σ does not contain α. ∎

Example 22. Consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ of Example 8. Suppose that *john* becomes a test driver, and that *tom* becomes the new official driver of the team t_1, which means update $\mathcal{K}$ with the insertion of the set of ABox assertions $F = \{TD(john), OD(tom), mf(tom, t_1)\}$. Since the TBox implies that a test driver cannot be an official driver, $TD(john)$ contradicts $OD(john)$. Also, since every team has at most one official driver, $\{OD(tom), mf(tom, t_1)$ contradicts the fact that *john* is the official driver of the team t_1. This means that the set $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(\mathcal{A})$ is not consistent with $\mathcal{T}$. Indeed, $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(\mathcal{A})$ contains the following $\mathcal{T}$-inconsistent sets:

$V_1 = \{\ TD(john), \quad OD(john)\ \};$
$V_2 = \{\ OD(tom), \quad mf(tom, t_1), \quad OD(john), \quad mf(john, t_1)\ \}.$

Where V_1 violates the negative inclusion assertion $OD \sqsubseteq \neg TD$ in $\mathcal{T}$, and V_2 violates the identification assertion $(id\ OD\ mf)$. There are two assertions in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(F)$ that appear in at least one $\mathcal{T}$-inconsistent set: $OD(john)$ and $mf(john, t_1)$. Observe, that $(V_1 \cup F) \setminus \{OD(john)\}$ is $\mathcal{T}$-consistent. On the other hand, $(V_2 \cup F) \setminus \{mf(john, t_1)\}$ is not $\mathcal{T}$-consistent, since it contains both $TD(john)$ and $OD(john)$. Exploiting Theorem 14 the KB resulting from the insertion of F into $\mathcal{K}$ is:

$$\langle \mathcal{T}, \{TD(john), mf(john, t_1), OD(tom), mf(tom, t_1), TD(bob)\}\rangle$$

□

Theorem 14 suggests immediately the algorithm ComputeInsertion for computing $\mathcal{K} \oplus^\mathcal{T}_\cap F$.

Algorithm ComputeInsertion takes in input a consistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ and a finite set of ABox assertions F, and returns a consistent

Input: a consistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a finite set of ABox assertions F such that $\langle \mathcal{T}, F \rangle$ is consistent
Output: a *DL-Lite*$_{A,id,den}$-KB
begin
  if $Mod(\langle \mathcal{T}, F \rangle) = \emptyset$
  then return $\langle \mathcal{T}, \mathcal{A} \rangle$;
  $W \leftarrow \mathsf{InconsistentSets}(\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F) \rangle)$;
  $D \leftarrow \emptyset$;
  foreach $\alpha \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$ **do**
    if $\exists w \in W$ s.t. $\alpha \in w$ and $\langle \mathcal{T}, F \cup w \setminus \{\alpha\} \rangle$ is consistent
    then $D \leftarrow D \cup \{\alpha\}$;
  return $\langle \mathcal{T}, F \cup \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus D \rangle$;
end

Algorithm 7: *ComputeInsertion*$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$

KB in *DL-Lite*$_{A,id,den}$. According to Definition 22, if $\langle \mathcal{T}, F \rangle$ is not consistent, then the algorithm returns the original KB $\mathcal{K}$. If $\langle \mathcal{T}, F \rangle$ is consistent, then ComputeInsertion essentially computes the set D of ABox assertions in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$ which do not belong to at least one ABox accomplishing the insertion of F into $\mathcal{K}$ minimally. It proceeds as follows. In order to exploit Theorem 14 the algorithm needs to compute all $\mathcal{T}$-inconsistent sets in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$.

To this end, ComputeInsertion uses the algorithm InconsistentSets presented in Section 5.3. Here, $\mathsf{InconsistentSets}(\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F) \rangle)$ computes the set W of all those $\mathcal{T}$-inconsistent sets in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ that are also $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F) \rangle$-*clash*. We note that by Lemma 14, we have every $\mathcal{T}$-inconsistent set V' in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ there exists in W a set w such that $w \subseteq V'$. Afterwards it adds to the set D each assertion $\alpha \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$ that is contained in at least one $w \in W$ and such that $F \cup w \setminus \{\alpha\}$ is $\mathcal{T}$-consistent. We recall that, by Theorem 14, every assertion in D cannot appear in $\bigcap_{\mathcal{A}'_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'_i)$, where $\mathcal{U}$ is the set containing all the ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Taking into account such observation, the algorithm returns the KB $\langle \mathcal{T}, F \cup \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus D \rangle$.

In what follows, we discuss formal and computational properties of the algorithm ComputeInsertion.

We start by dealing with termination of Algorithm 7.

Lemma 20. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB in DL-Lite*$_{A,id,den}$*, and let F be a finite set of ABox assertions. Then* ComputeInsertion$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$ *terminates.*

Proof. The termination follows directly from the termination of InconsistentSets and from the finiteness of sets $\mathcal{T}$, $\mathcal{A}$ and F. ∎

The following lemma states that the result of $\mathsf{ComputeInsertion}(\langle\mathcal{T},\mathcal{A}\rangle, F)$ is a consistent *DL-Lite*$_{A,id,den}$-KB.

Lemma 21. *Let $\langle\mathcal{T},\mathcal{A}\rangle$ be a consistent KB in DL-Lite$_{A,id,den}$, and let F be a finite set of ABox assertions, and $\langle\mathcal{T},\mathcal{A}'\rangle = \mathsf{ComputeInsertion}(\langle\mathcal{T},\mathcal{A}\rangle, F)$. We have that $\langle\mathcal{T},\mathcal{A}'\rangle$ is a consistent DL-Lite$_{A,id,den}$-KB.*

Proof. Let $\langle\mathcal{T},\mathcal{A}'\rangle = \mathsf{ComputeInsertion}(\langle\mathcal{T},\mathcal{A}\rangle, F)$. Firstly, we show that $\langle\mathcal{T},\mathcal{A}'\rangle$ is a consistent KB. Trivially, if $\langle\mathcal{T}, F\rangle$ is not consistent, ComputeInsertion returns $\langle\mathcal{T},\mathcal{A}\rangle$ which is consistent. We now consider the case where $\langle\mathcal{T}, F\rangle$ is consistent. By construction of Algorithm 7, $\langle\mathcal{T},\mathcal{A}'\rangle$ is obtained from $\langle\mathcal{T},\mathcal{A}\rangle$ by:

- inserting into $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ a finite set F of atomic ABox assertions such that $\langle\mathcal{T}, F\rangle$ is consistent;
- removing from $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ the set of assertions $D = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(F)$ such that, for $1 \leq i \leq n$, α_i belongs to at least one subset V of $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ which is a $\mathcal{T}$-inconsistent set and such that $F \cup (V \setminus \{\alpha\}$ is $\mathcal{T}$-consistent. Hence, there exists no $\mathcal{A}'' \subseteq \mathsf{cl}_\mathcal{T}(F) \cup (\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus D)$ that is $\mathcal{T}$-inconsistent.

Therefore, we have that $\mathcal{A}' = F \cup (\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus D)$, which is consistent. As shown above, ComputeInsertion computes $\langle\mathcal{T},\mathcal{A}'\rangle$ by adding and deleting atomic ABox assertions from the ABox $\mathcal{A}$. Since $\mathcal{T}$ is a *DL-Lite*$_{A,id,den}$ TBox by hypothesis, then it is clear that $\langle\mathcal{T},\mathcal{A}'\rangle$ is a *DL-Lite*$_{A,id,den}$-KB. ∎

The theorem below states that the algorithm $\mathsf{ComputeInsertion}(\langle\mathcal{T},\mathcal{A}\rangle, F)$ can be used for computing $\langle\mathcal{T},\mathcal{A}\rangle \oplus^\mathcal{T}_\cap F$.

Theorem 15. *Let $\langle\mathcal{T},\mathcal{A}\rangle$ be a consistent KB in DL-Lite$_{A,id,den}$, and let F be a finite set of ABox assertions. Then $\langle\mathcal{T},\mathcal{A}\rangle \oplus^\mathcal{T}_\cap F$ is logically equivalent to $\mathsf{ComputeInsertion}(\langle\mathcal{T},\mathcal{A}\rangle, F)$.*

Proof. By analyzing Algorithm 7. According to Definition 22, if $\langle\mathcal{T}, F\rangle$ is not consistent, then ComputeInsertion returns the original KB $\mathcal{K}$. We turn to the case where $\langle\mathcal{T}, F\rangle$ is consistent. Let $\mathcal{M}$ be the set of all maximal subsets $\mathcal{A}''_j$ of $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ such that $\mathcal{A}''_j \cup F$ is $\mathcal{T}$-consistent. As shown above, by using Theorem 11 and by exploiting Lemma 16, we know that for a *DL-Lite*$_{A,id,den}$-KB $\langle\mathcal{T},\mathcal{A}\rangle$

$$\langle\mathcal{T},\mathcal{A}\rangle \oplus^\mathcal{T}_\cap F \equiv \langle\mathcal{T}, \bigcap_{\mathcal{A}'_i \in \mathcal{U}} \mathsf{cl}_\mathcal{T}(\mathcal{A}'_i)\rangle = \langle\mathcal{T}, \mathsf{cl}_\mathcal{T}(F) \cup \bigcap_{\mathcal{A}''_j \in \mathcal{M}} \mathcal{A}''_j\rangle$$

By exploiting Theorem 14, ComputeInsertion computes $\bigcap_{\mathcal{A}''_j \in \mathcal{M}} \mathcal{A}''_j$ by deleting from $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ every assertion α such that there exists a $\mathcal{T}$-inconsistent set V in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ containing α and such that $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent. ∎

Finally, we turn to the complexity of computing the result of updating a consistent *DL-Lite*$_{A,id,den}$-KB with the insertion of a finite set of ABox assertions. By analyzing Algorithm 7 we get:

Theorem 16. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB in DL-Lite$_{A,id,den}$, and F be a finite set of ABox assertions. Then* ComputeInsertion*$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$ computes $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{\cap} F$ in polynomial time with respect to $|\mathcal{A}|$, and $|F|$, and in exponential time with respect to $|\mathcal{T}|$.*

Proof. The proof of this theorem is an immediate consequence of the following observations:

- $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ can be computed in quadratic time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$; moreover $\mathsf{cl}_{\mathcal{T}}(F)$ can be computed in quadratic time with respect to $|\mathcal{T}|$ and $|F|$;
- checking satisfiability of $\langle \mathcal{T}, F \rangle$ can be done in *DL-Lite*$_{A,id,den}$ in AC^0 with respect to $|F|$ and in exponential time with respect to $|\mathcal{T}|$ (cf. Theorem 7);
- there is one call to InconsistentSets, which computes the set W of all $\mathcal{T}$-inconsistent sets in polynomial time with respect to $|\mathcal{A}|$, and $|F|$, and in exponential time with respect to $|\mathcal{T}|$; moreover the size of W is polynomial with respect to $|\mathcal{T} \setminus \mathcal{T}_{id} \cup \mathcal{T}_{den}|$, $|\mathcal{A}|$ and $|F|$, and exponential with respect to $|\mathcal{T}_{id} \cup \mathcal{T}_{den}|$; finally, the size of each set $w \in W$ is polynomial in the size of $\mathcal{T}$.
- for each $\alpha \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$ and for each $w \in W$ the check of satisfiability of $\langle \mathcal{T}, F \cup (w \setminus \{\alpha\}) \rangle$ can be done in polynomial time with respect to $|F \cup (w \setminus \{\alpha\})|$ and in exponential time with respect to $|\mathcal{T}|$;
- the size of D is polynomial with respect to the size of $\mathcal{A}$;
- for each $\alpha \in D$ the cost of eliminating α from $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ is clearly polynomial in the size of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$.

∎

7.2 Update by deletion in *DL-Lite*$_{A,id,den}$

In this section, we refer to a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and address the problem of updating $\mathcal{K}$ with the deletion of a finite set F of ABox assertions. As in the previous section, we assume that $\mathcal{K}$ is consistent.

By exploiting Lemma 16 and by analyzing the form of *DL-Lite*$_{A,id,den}$ TBoxes, we have that an ABox assertion α can be entailed by a consistent *DL-Lite*$_{A,id,den}$-KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if and only if one of the following cases holds:

(i) $\alpha \in \mathcal{A}$;
(ii) exists in $\mathcal{A}$ an ABox assertion β such that $\langle \mathcal{T}, \{\beta\} \rangle \models \alpha$.

In other worlds, *DL-Lite*$_{A,id,den}$ does not allow that the TBox alone entails an ABox assertion, i.e., for each TBox $\mathcal{T}$ in *DL-Lite*$_{A,id,den}$ and for each ABox assertion α, we have that $\langle \mathcal{T}, \emptyset \rangle \not\models \alpha$. From the observation above, and by Proposition 2, it directly follows the next lemma.

Lemma 22. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a KB in DL-Lite$_{A,id,den}$, and let F be a finite set of ABox assertions. Then there always exists an ABox $\mathcal{A}'$ accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$.*

Proof. The proof directly follows from Lemma 16 and from Proposition 2. ∎

Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_{A,id,den}$-KB, and let $\mathcal{U}$ be the set of ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Lemma 22 guarantees that $\mathcal{U}$ is non-empty. Then, from Definition 23, we have that

$$\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F = \langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle.$$

It directly follows that, if α is an assertion in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$, then $\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F \not\models \alpha$ if at least one ABox $\mathcal{A}_i$ exists in $\mathcal{U}$, such that $\alpha \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$. Consequently, we can compute $\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F$ by building all ABoxes $\mathcal{A}_i$ in $\mathcal{U}$ and then, for each $\mathcal{A}_i \in \mathcal{U}$, removing from $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ all the assertions in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i)$.

Let us start by considering the case where the set F is constituted by just one assertion f. By exploiting Theorem 12, it is easy to conclude that there is exactly one ABox accomplishing the deletion of $\{f\}$ from a given *DL-Lite*$_{A,id,den}$-KB.

Lemma 23. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB and let f be an ABox assertion. Up to logical equivalence, there is exactly one ABox $\mathcal{A}'$ that accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally.*

Proof. We prove that, up to logical equivalence, there is exactly one ABox $\mathcal{A}'$ that accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. If $\langle \mathcal{T}, \mathcal{A} \rangle \not\models F$, then the proof directly follows from Lemma 19 which assures that in such a case the KB resulting from updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F is logically equivalent to $\langle \mathcal{T}, \mathcal{A} \rangle$.

Let $\langle \mathcal{T}, \mathcal{A} \rangle \models f$. Toward a contradiction, suppose that both $\mathcal{A}_1$ and $\mathcal{A}_2$ accomplish the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, and suppose that

Input: a consistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, an ABox assertion f.
Output: a set of ABox assertions
begin
  $\mathcal{A}' \leftarrow \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$;
  foreach $\alpha \in \mathcal{A}'$ **do**
    if $\langle \mathcal{T}, \{\alpha\} \rangle \models f$
    then $\mathcal{A}' \leftarrow \mathcal{A}' \setminus \{\alpha\}$;
  return $\mathcal{A}'$;
end

Algorithm 8: *Delete*$(\langle \mathcal{T}, \mathcal{A} \rangle, f)$

$\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) \neq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$. Clearly, $f \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1)$ and $f \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$. Since $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) \neq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$, there exists an assertion $\alpha \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1)$ such that $\alpha \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$. From Theorem 12 it follows that $\alpha \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Moreover, since we suppose that $f \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1)$ then $\langle \mathcal{T}, \{\alpha\} \rangle \not\models f$. We recall that an assertion γ is implied by a *DL-Lite*$_{A,id,den}$-KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if and only if $\gamma \in \mathcal{A}$ or there exists an assertion $\beta \in \mathcal{A}$ such that $\langle \mathcal{T}, \{\beta\} \rangle \models \gamma$. Hence, $\langle \mathcal{T}, \mathcal{A}_2 \cup \{\alpha\} \rangle \not\models f$. This means that $\mathcal{A}_2 \cup \{\alpha\}$ accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2 \cup \{\alpha\})$ has fewer changes than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$ with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Since $\mathcal{A}_2$ accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, we have a contradiction. ∎

Let $\mathcal{A}'$ be an ABox accomplishing the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. With the aim to prove that $\mathcal{A}'$ can be computed in polynomial time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$, we provide Algorithm 8.

Lemma 24. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB, f be an ABox assertion. Then* Delete$(\langle \mathcal{T}, \mathcal{A} \rangle, f)$ *terminates returning a set of ABox assertions in polynomial time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$.*

Proof. The terminations of $\mathsf{Delete}(\langle \mathcal{T}, \mathcal{A} \rangle, f)$ follows directly from the finiteness of the set $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$.

Now, we prove that $\mathcal{A}' = \mathsf{Delete}(\langle \mathcal{T}, \mathcal{A} \rangle, f)$ can be computed in polynomial time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$ by means of the following observations. Algorithm 8 proceeds as follows. Firstly, it computes the set $\mathcal{A}' = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ which can be done in polynomial time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$. Moreover, the size of $\mathcal{A}'$ is polynomial in $|\mathcal{A}|$ and $|\mathcal{T}|$. Next, for each assertion α in $\mathcal{A}'$, Algorithm 8 removes α from $\mathcal{A}'$ if and only if $\langle \mathcal{T}, \{\alpha\} \rangle \models f$. The check $\langle \mathcal{T}, \{\alpha\} \rangle \models f$ can be done in polynomial time with respect to $|\mathcal{T}|$. Finally, the cost of eliminating α from $\mathcal{A}'$ is polynomial in the size of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. ∎

Lemma 25. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB, f be an ABox assertion, and*

$\mathcal{A}' = \mathsf{Delete}(\langle \mathcal{T}, \mathcal{A}\rangle, f)$. *Then* $\mathcal{A}'$ *accomplishes the deletion of* $\{f\}$ *from* $\langle \mathcal{T}, \mathcal{A}\rangle$ *minimally.*

Proof. Essentially, Algorithm 8 proceeds as follows.

step 1: it computes the set $\mathcal{A}' = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$;
step 2: for each assertion $\alpha \in \mathcal{A}'$, if $\langle \mathcal{T}, \{\alpha\}\rangle \models f$, it removes α from $\mathcal{A}'$.

We have to prove that $\mathcal{A}'$ accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally. Lemma 23 tells us that every other ABox accomplishing the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally is logically equivalent to $\mathcal{A}'$. We start proving that $\mathcal{A}'$ accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$, then we prove that it does this minimally.

It easy to see that if $\langle \mathcal{T}, \mathcal{A}\rangle \not\models f$, then there are no assertions $\alpha \in \mathcal{A}'$ such that $\langle \mathcal{T}, \{\alpha\}\rangle \models f$ and then, according to Lemma 19, Algorithm 8 returns $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. In what follows, we assume that $\langle \mathcal{T}, \mathcal{A}\rangle \models F$. Assume by way of contradiction that $\mathcal{A}'$ does not accomplish the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$, this means that $\langle \mathcal{T}, \mathcal{A}'\rangle \models \{f\}$. Hence, there is an assertion $\alpha \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ (not necessary different from f) such that $\langle \mathcal{T}, \{\alpha\}\rangle \models \alpha$. But this is prevented by **step 2**. It follows that $\alpha \notin \mathcal{A}'$ and then that $\langle \mathcal{T}, \mathcal{A}'\rangle \not\models f$, which is a contradiction.

Suppose that $\mathcal{A}'$ does not accomplish the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally. This means that exists an ABox $\mathcal{A}''$ which accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$ and such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$ has fewer changes than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Since Algorithm 8 only deletes atoms from $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$, the observation above implies that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$ has fewer deletions than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. That is, there exists an assertion $\beta \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\beta \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$ and $\beta \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$. But this means that, during the process, β has been removed from $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. The check in **step 2** allows us saying that $\langle \mathcal{T}, \{\beta\}\rangle \models f$, so $\langle \mathcal{T}, \mathcal{A}''\rangle \models f$, which is a contradiction. ∎

Let us now consider the case of arbitrary F, i.e., the case where $F = \{f_1, \ldots, f_m\}$, for $m \geq 0$.

Let $\mathcal{U}^* = \{\mathcal{A}_1 \ldots \mathcal{A}_m\}$ be a set of ABoxes $\mathcal{A}_i$, such that, for every $1 \leq i \leq m$, $\mathcal{A}_i$ accomplishes the deletion of $\{f_i\} \subseteq F$ from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally. Let $\mathcal{U}$ be the set of ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally.

Lemma 19 states that if $\langle \mathcal{T}, \mathcal{A}\rangle \not\models F$, then the KB resulting from updating $\langle \mathcal{T}, \mathcal{A}\rangle$ with the deletion of F is logically equivalent to $\langle \mathcal{T}, \mathcal{A}\rangle$. This means that if there exists in F an assertion f_i such that $\langle \mathcal{T}, \mathcal{A}\rangle \not\models f_i$, then the deletion does not affect the original KB. In what follows we focus on the case where $\langle \mathcal{T}, \mathcal{A}\rangle \models F$.

Assume that $\langle \mathcal{T}, \mathcal{A}\rangle \models F$. One might wonder whether the set $\mathcal{U}$ coincides (modulo logical equivalence) with $\mathcal{U}^*$. The next lemma tells us that one di-

rection is indeed valid: for each $\mathcal{A}' \in \mathcal{U}$ there exists an $\mathcal{A}'' \in \mathcal{U}^*$ such that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) = Mod(\langle \mathcal{T}, \mathcal{A}'' \rangle)$.

Lemma 26. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB, and let F be a set of ABox assertions such that $\langle \mathcal{T}, \mathcal{A} \rangle \models F$. If $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, then there exists an assertion $f' \in F$ such that $\mathcal{A}'$ accomplishes the deletion of $\{f'\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally.*

Proof. Suppose that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Obviously, there exists at least one assertion f_j in F such that $\mathcal{A}'$ accomplishes the deletion of $\{f_j\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ otherwise, for each $f_i \in F$, $\langle \mathcal{T}, \mathcal{A}' \rangle \models f_i$, which means $\langle \mathcal{T}, \mathcal{A}' \rangle \models F$. But this contradicts that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. Let $\mathcal{J} \subseteq F$ be the set of all assertions f_j such that $\mathcal{A}'$ accomplishes the deletion of $\{f_j\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$. We prove that there exists at least one $f' \in \mathcal{J}$ such that $\mathcal{A}'$ accomplishes the deletion of $\{f'\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Assume by way of contradiction that for all $f_j \in \mathcal{J}$ we have that $\mathcal{A}'$ does not accomplish the deletion of $\{f_j\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. It follows that for every f_j there exists an ABox $\mathcal{A}_j$ such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}') \subset \mathsf{cl}_\mathcal{T}(\mathcal{A}_j) \subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A})$, and $\langle \mathcal{T}, \mathcal{A}_j \rangle \not\models f_j$. This means that $\mathcal{A}_j$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and that $\mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$ has fewer deletions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. Hence, $\mathcal{A}'$ does not accomplish the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, which is a contradiction. ∎

However, the next example shows that the other direction does not hold: there may exist $\mathcal{A}' \in \mathcal{U}^*$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is not logically equivalent to any $\langle \mathcal{T}, \mathcal{A}'' \rangle$ such that $\mathcal{A}'' \in \mathcal{U}$.

Example 23. Let $\mathcal{T} = \{B \sqsubseteq C,\ C \sqsubseteq D,\ E \sqsubseteq D\}$, $\mathcal{A} = \{B(a),\ E(a)\}$, and $F = \{C(a),\ D(a)\}$. It is easy to see that the deletion of $D(a)$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ is accomplished minimally by $\emptyset$, while the deletion of $C(a)$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ is accomplished minimally by $\{E(a)\}$. Hence, in this case, we have $\mathcal{U}^* = \{\emptyset, \{E(a)\}\}$. Also, one can verify that $\{E(a)\}$ is the only (up to logical equivalence) ABox accomplishing the deletion of F minimally, i.e., $\mathcal{U} = \{\{E(a)\}\}$. Thus, there is an ABox in $\mathcal{U}^*$, namely $\mathcal{A}' = \emptyset$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is not logically equivalent to any $\langle \mathcal{T}, \mathcal{A}'' \rangle$ such that $\mathcal{A}'' \in \mathcal{U}$. □

The next theorem characterizes when a given $\mathcal{A}_i \in \mathcal{U}^*$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally.

Theorem 17. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB, and let F be a set of ABox assertions such that $\langle \mathcal{T}, \mathcal{A} \rangle \models F$. Let f_j be an assertion in F, and let $\mathcal{A}_j$ be the ABox accomplishing the deletion of $\{f_j\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. $\mathcal{A}_j$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally if and only if there is in F no assertion $f_h \neq f_j$ such that $\langle \mathcal{T}, \{f_h\} \rangle \models f_j$ and $\langle \mathcal{T}, \{f_j\} \rangle \not\models f_h$.*

Proof. Let $f_j \in F$. Lemma 23 assures that there is exactly one ABox $\mathcal{A}_j$ that accomplishes the deletion of $\{f_j\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally.
($\Rightarrow$) Assume that $\mathcal{A}_j$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Then we prove that there is no assertion $f_h \in F$ such that $f_h \neq f_j$, $\langle \mathcal{T}, \{f_h\} \rangle \models f_j$, and $\langle \mathcal{T}, \{f_j\} \rangle \not\models f_h$. Toward a contradiction, assume that F contains an assertion $f_h \neq f_j$ such that $\langle \mathcal{T}, \{f_h\} \rangle \models f_j$ and such that $\langle \mathcal{T}, \{f_j\} \rangle \not\models f_h$. Let $\mathcal{A}_h$ be the ABox accomplishing the deletion of $\{f_h\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Clearly, $\mathcal{A}_h$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$.

Firstly we show that $f_j \in \mathsf{cl}_\mathcal{T}(\mathcal{A}_h)$. Suppose, by way of contradiction, that $f_j \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}_h)$. Since $\langle \mathcal{T}, \{f_j\} \rangle \not\models f_h$ then the ABox $\mathsf{cl}_\mathcal{T}(\mathcal{A}_h \cup \{f_j\})$ accomplishes the deletion of $\{f_h\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ and has fewer changes than $\mathsf{cl}_\mathcal{T}(\mathcal{A}_h)$ with respect to $\mathsf{cl}_\mathcal{T}(\mathcal{A})$. This contradicts that $\mathcal{A}_h$ accomplishes the deletion of $\{f_h\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Hence, $f_j \in \mathcal{A}_h$. In the same manner, we can show that every assertion $\alpha \in \mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$ such that $\langle \mathcal{T}, \{\alpha\} \rangle \not\models f_h$ is in $\mathsf{cl}_\mathcal{T}(\mathcal{A}_h)$. Now, we show that $\mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$ does not contain f_h. Toward a contradiction, assume that $f_h \in \mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$. Trivially, since $\langle \mathcal{T}, \{f_h\} \rangle \models f_j$ then $f_j \in \mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$. Which contradicts that $\mathcal{A}_j$ accomplishes the deletion of $\{f_j\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Similarly, we can show that every assertion in $\mathsf{cl}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(\mathcal{A}_h)$ is not in $\mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$. It directly follows that $\mathsf{cl}_\mathcal{T}(\mathcal{A}_j) \subset \mathsf{cl}_\mathcal{T}(\mathcal{A}_h)\mathsf{cl}_\mathcal{T}(\mathcal{A})$. Hence, $\mathcal{A}_j$ does not accomplish the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, which is a contradiction.

($\Leftarrow$) Let f_j be an assertion in F such that there exists no assertion $f_h \neq fj$ in F such that $\langle \mathcal{T}, \{f_h\} \rangle \models f_j$, and $\langle \mathcal{T}, \{f_j\} \rangle \not\models f_h$. Let $\mathcal{A}_j$ be the ABox accomplishing the deletion of $\{f_j\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. We prove that $\mathcal{A}_j$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Since $f_j \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$ then $F \not\subseteq \mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$, which means that $\mathcal{A}_j$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. Assume by contradiction that $\mathcal{A}_j$ does not accomplish the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. This means that there is an ABox $\mathcal{A}_i$ which accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}_j) \subset \mathsf{cl}_\mathcal{T}(\mathcal{A}_i) \subset \mathsf{cl}_\mathcal{T}(\mathcal{A})$. It follows that there exists an assertion $f_i \in \mathsf{cl}_\mathcal{T}(\mathcal{A}_i)$ such that $f_i \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$. Since $\mathcal{A}_j$ accomplishes the deletion of $\{f_j\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, then $\langle \mathcal{T}, \{f_i\} \rangle \models f_j$. This means that $f_j \in \mathsf{cl}_\mathcal{T}(\mathcal{A}_i)$. Since $\mathcal{A}_i$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and $f_j \in \mathsf{cl}_\mathcal{T}(\mathcal{A}_i)$, then there exists in F an assertion $f_h \neq f_j$ such that $f_h \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}_i)$. But since $f_h \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}_i)$ then $f_h \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}_j)$. Clearly, since $f_j \in \mathsf{cl}_\mathcal{T}(\mathcal{A}_i)$ then $\langle \mathcal{T}, \{f_j\} \rangle \not\models f_h$. Moreover, since $\mathcal{A}_j$ accomplishes the deletion of $\{f_j\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, then $\langle \mathcal{T}, \{f_h\} \rangle \models f_j$. This means that there exists in F and assertion $f_h \neq f_j$ such that $\langle \mathcal{T}, \{f_h\} \rangle \models f_j$ and $\langle \mathcal{T}, \{f_j\} \rangle \not\models f_h$, which is a contradiction. ∎

Lemma 19 and Theorem 17 suggest immediately the algorithm ComputeDeletion to compute $\mathcal{K} \ominus^\mathcal{T}_\cap F$. Algorithm ComputeDeletion takes in input a consistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ in *DL-Lite*$_{A,id,den}$ and a finite set of ABox assertions F, and returns a consistent KB in *DL-Lite*$_{A,id,den}$.

Input: a consistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a finite set of ABox assertions F.
Output: a *DL-Lite*$_{A,id,den}$-KB
begin
  if $\langle \mathcal{T}, \mathcal{A} \rangle \not\models F$
  then return $\langle \mathcal{T}, \mathcal{A} \rangle$;
  $F' \leftarrow F$;
  foreach $f_i \in F'$ and $f_j \in F$ such that $f_i \neq f_j$ **do**
    if $\langle \mathcal{T}, \{f_j\} \rangle \models f_i$ and $\langle \mathcal{T}, \{f_i\} \rangle \not\models f_j$
    then $F' \leftarrow F' \setminus \{f_i\}$;
  $\mathcal{A}' \leftarrow \mathcal{A}$;
  foreach $f \in F'$ **do**
    $\mathcal{A}' \leftarrow \mathsf{Delete}(\langle \mathcal{T}, \mathcal{A}' \rangle, f)$;
  return $\langle \mathcal{T}, \mathcal{A}' \rangle$;
end

Algorithm 9: *ComputeDeletion*$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$

It proceeds as follows. According to Lemma 19, if $\langle \mathcal{T}, \mathcal{A} \rangle$ does not entail F then ComputeDeletion$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$ returns $\langle \mathcal{T}, \mathcal{A} \rangle$. If $\langle \mathcal{T}, \mathcal{A} \rangle \models F$ then, by exploiting Theorem 17, it marks those assertions f_i in F for which there exists in F an assertion $f_i \neq f_j$ such that $\langle \mathcal{T}, \{f_j\} \rangle \models f_i$ and $\langle \mathcal{T}, \{f_i\} \rangle \not\models f_j$. Next, it builds the set $\mathcal{A}' = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. Finally, by means of Algorithm Delete, Algorithm 9 iteratively modifies $\mathcal{A}'$ minimally in such a way that, for each assertion f in F that are not marked, then $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models f$.

Lemma 27. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB in DL-Lite$_{A,id,den}$, and let F be a finite set of ABox assertions, and $\langle \mathcal{T}, \mathcal{A}' \rangle$ = ComputeDeletion$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$. We have that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is a consistent KB in DL-Lite$_{A,id,den}$.*

Proof. Let $\langle \mathcal{T}, \mathcal{A}' \rangle$ = ComputeDeletion$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$. The proof follows directly from the facts that ComputeDeletion does not modify the TBox of the original KB, and that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. ∎

Next, we deal with termination and correctness of ComputeDeletion.

Lemma 28. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB in DL-Lite$_{A,id,den}$, and let F be a finite set of ABox assertions. Then ComputeDeletion$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$ terminates.*

Proof. The termination of ComputeDeletion$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$ follows directly from the termination of the algorithm Delete. ∎

The next theorem sanctions the correctness of Algorithm 9.

Theorem 18. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB in DL-Lite$_{A,id,den}$, and F be a finite set of ABox assertions. Then $\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F$ is logically equivalent to* ComputeDeletion$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$.

Proof. Let $F = \{f_1, \ldots, f_m\}$ be a set of ABox assertions.

Firstly, if $\langle \mathcal{T}, \mathcal{A} \rangle \not\models F$, then, according to Lemma 19, Algorithm 9 returns $\langle \mathcal{T}, \mathcal{A} \rangle$. Secondly, it computes the set $F' = \{f_1, \ldots, f_n\}$, with $n \leq m$ by removing from F every assertion α such that there exists in F an assertion $\beta \neq \alpha$, $\langle \mathcal{T}, \{\beta\} \rangle \models \alpha$, and $\langle \mathcal{T}, \{\alpha\} \rangle \models \beta$.

Let $\mathcal{U}^* = \{\mathcal{A}_1 \ldots \mathcal{A}_n\}$ be the set of ABoxes $\mathcal{A}_i$, such that, for every $i \in \{1, \ldots, n\}$, $\mathcal{A}_i$ accomplishes the deletion of $\{f_i\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Let $\mathcal{U}$ be the set of ABoxes accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Theorem 17 and Lemma 26 assure that (modulo logical equivalence) $\mathcal{U} = \mathcal{U}^*$.

Finally, by means of Algorithm Delete, Algorithm 9 computes the ABox $\mathcal{A}' = \bigcap_{\mathcal{A}_j \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_j)$ by removing from $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ every assertion which is not in at least one ABox in $\mathcal{U}^*$. Then it returns $\langle \mathcal{T}, \mathcal{A}' \rangle$. ∎

Finally, we turn to the complexity of updating a *DL-Lite*$_{A,id,den}$-KB with the deletion of a set of ABox assertions F. By analyzing Algorithm 9, and by exploiting Lemma 23 we can provide the following theorem.

Theorem 19. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a consistent KB in DL-Lite$_{A,id,den}$, and let F be a finite set of ABox assertions. Then* ComputeDeletion$(\langle \mathcal{T}, \mathcal{A} \rangle, F)$ *computes $\langle \mathcal{T}, \mathcal{A} \rangle \ominus^{\mathcal{T}}_{\cap} F$ in polynomial time with respect to $|\mathcal{T}|$, $|\mathcal{A}|$ and $|F|$.*

Proof. The proof is an immediate consequence of the following observations.

- Given a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$ and two ABoxes $\mathcal{A}$ and F, deciding if $\langle \mathcal{T}, \mathcal{A} \rangle \models F$ can be done in polynomial time with respect to $|\mathcal{T}|$, $|\mathcal{A}|$ and $|F|$.
- The algorithm checks if, for all $f_i \in F$, there exists an $f_j \in F$, with $i \neq j$, such that $\langle \mathcal{T}, \{f_j\} \rangle \models f_i$ and $\langle \mathcal{T}, \{f_i\} \rangle \models f_j$. This can be done in polynomial time in $|\mathcal{T}|$ and $|F|$.
- In the worst case there is one call Delete$(\langle \mathcal{T}, \mathcal{A}' \rangle, f)$ for each atom f in F.
- From Lemma 24 we know that Delete$(\langle \mathcal{T}, \mathcal{A}' \rangle, f)$ runs in polynomial time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$.

∎

Part IV

Inconsistency-tolerant Query Answering

Chapter 8

Query answering over inconsistent Description Logic KBs

It is well-known that inconsistency causes severe problems in logic-based Knowledge Representation. In particular, since an inconsistent logical theory has no model, it logically implies every formula (*ex falso quodlibet*), and, therefore, traditionally query answering over an inconsistent knowledge base becomes meaningless.

There are many approaches for devising inconsistency-tolerant inference systems [12], originated in different areas, including Logic, Artificial Intelligence, and Databases.

Let $\mathcal{K}$ be an inconsistent knowledge base that contains both α and $\neg\alpha$. How to answer the query *"is α true?"*. The most direct approach is to answer the query only after we have *cleaned* the knowledge base from all inconsistencies [84, 96], i.e. we have restored consistency. Another strategy would be to leave unchanged the KB, but trying to obtain only consistent information during query answering. This approach was born in the Database community and is commonly known as *consistent query answering* [36]. Another approach is to consider inconsistencies as an natural phenomenon in realistic data which are to be handled by a logic which tolerates it [82, 83, 92, 106]. Such logics are called paraconsistent. Roughly speaking, a paraconsistent logic is a logic rejecting *ex falso quodlibet* principle, i.e., the principle stating that from a contradiction it is possible to derive everything.

In this chapter, we address the problem of dealing with inconsistencies in Description Logic knowledge bases. Our work is especially inspired by the approaches to *consistent query answering* in databases [4].

The main tool used to obtain consistent information from an inconsistent

database is the notion of database *repair*: a *repair* of a database contradicting a set of integrity constraints is a database obtained by applying a "minimal" set of changes which restore consistency. In general, there are many possible *repairs* for a database $\mathcal{DB}$, and, therefore, the approach sanctions that what is consistently true in $\mathcal{DB}$ is simply what is true in all possible *repair*s of $\mathcal{DB}$. Thus, inconsistency-tolerant query answering amounts to compute the tuples that are answers to the query in all possible *repairs*. By adopting different notions of "minimality", it is possible to give rise to different inconsistency-tolerant semantics.

Similarly to the work in [70], adopting the notion of *repair*, we study DL semantical frameworks which are inconsistency-tolerant, and we devise techniques for answering unions of conjunctive queries posed to DL KBs under such inconsistency-tolerant semantics.

Depending on the expressive power of the underlying language, the TBox alone might be inconsistent, or the TBox might be consistent, but the axioms in the ABox might contradict the axioms in the TBox.

In several scenarios as Ontology-based Data Access [26, 95], the TBox is usually a high quality representation of the domain, designed in such a way to avoid inconsistencies in the modeling of concepts and relationships. On the contrary, the ABox derives from data sources which are independent on the conceptualization represented by the TBox, and therefore may contain data which are not coherent with it.

The ability of dealing with such a form of inconsistency is of critical importance, in particular to obtain meaningful answers to queries posed over inconsistent KBs. Therefore, our study is carried out under the assumption that the TBox is consistent, and inconsistency may arise between the ABox and the TBox (inconsistencies in the TBox are considered, e.g., in [55, 59, 83, 91, 98, 109]). This is particularly suited in those settings in which the intensional knowledge, i.e., the TBox, is the faithful representation of the domain, whereas data may be not compliant to such representation, and therefore violate some TBox assertions, for several reasons, e.g., because they come from different autonomous sources, as in data integration.

In Section 8.1, by adopting different criteria of minimal changes, we define different notions of repair. On the basis of such notions we propose in Section 8.2 different consistency-tolerant semantics, named *AR*-semantics, *CAR*-semantics, *IAR*-semantics, and *ICAR*-semantics. In Section 8.3 we provide a discussion on the properties of such semantics. Finally, in Section 8.4 we study query answering under such semantics.

8.1 The notion of repair

The inconsistency-tolerant semantics we propose in this work are based on the notion of *repair*. Intuitively, given a possibly inconsistent DL KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a repair $\mathcal{A}_r$ for $\mathcal{K}$ is an ABox such that the KB $\langle \mathcal{T}, \mathcal{A}_r \rangle$ is consistent under the first-order semantics, and $\mathcal{A}_r$ "minimally" differs from $\mathcal{A}$. Notice that in general not a single, but several repairs may exist, depending on the particular minimality criterion adopted. We consider here different notions of "minimality", which give rise to different inconsistency-tolerant semantics. In all cases, such semantics coincide with the classical first-order semantics when inconsistency does not come into play, i.e., when the KB is consistent under standard first-order semantics.

In order to give a characterization of the notion of "minimality", we refer to the relation of *fewer changes* between two sets of ABox assertions with respect to another one, given by Fagin, Ullman and Vardi in [44]. We have already introduced such notion in Section 6.3. Next we just briefly recall it. Let $\mathcal{A}$, $\mathcal{A}_1$, and $\mathcal{A}_2$ be three finite sets of ABox assertions. We say that $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_2$ with respect to $\mathcal{A}$ if $\mathcal{A} \setminus \mathcal{A}_1 \subset \mathcal{A} \setminus \mathcal{A}_2$. Also, we say that $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same deletions with respect to $\mathcal{A}$ if $\mathcal{A} \setminus \mathcal{A}_1 = \mathcal{A} \setminus \mathcal{A}_2$. Moreover, we say that $\mathcal{A}_1$ has fewer insertions than $\mathcal{A}_2$ with respect to $\mathcal{A}$ if $\mathcal{A}_1 \setminus \mathcal{A} \subset \mathcal{A}_2 \setminus \mathcal{A}$. Finally, we say that $\mathcal{A}_1$ has *fewer changes* than $\mathcal{A}_2$ with respect to $\mathcal{A}$ if (i) $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_2$ with respect to $\mathcal{A}$, or (ii) $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same deletions with respect to $\mathcal{A}$, and $\mathcal{A}_1$ has fewer insertions than $\mathcal{A}_2$ with respect to $\mathcal{A}$.

With the notion of *fewer changes* in place, we can illustrate the first notion of repair that we consider. A repair $\mathcal{A}_r$ of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is an ABox that is $\mathcal{T}$-consistent and such that there does not exist another ABox $\mathcal{A}'_r$ that is $\mathcal{T}$-consistent and that has fewer changes than $\mathcal{A}_r$ with respect to $\mathcal{A}$. Intuitively, a repair $\mathcal{A}_r$ is obtained by changing minimally the initial ABox $\mathcal{A}$ in order to make it $\mathcal{T}$-consistent. The formal definition is given below.

Definition 24. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an DL KB in $\mathcal{L}$. An *ABox repair (A-repair)* of $\mathcal{K}$ is a set $\mathcal{A}'$ of ABox assertions such that:

1. $\mathcal{A}'$ is $\mathcal{T}$-consistent, i.e., $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$, and
2. there is no set $\mathcal{A}''$ of ABox assertions that is $\mathcal{T}$-consistent, and has fewer changes than $\mathcal{A}'$ with respect to $\mathcal{A}$.

It is easy to see that, in general, more than one *A*-repair of a KB $\mathcal{K}$ exist. In what follows we denote by *AR-Set*($\mathcal{K}$) the set of *A*-repairs of $\mathcal{K}$.

We now present an example illustrating the notion of *A*-repair.

Example 24. Consider the following KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, which is loosely inspired by the OWL ontology *people*[1]. The TBox and the ABox contain respectively the following assertions:

$$\begin{aligned}
\mathcal{T} = \{ \quad & \mathsf{Man} \sqsubseteq \mathsf{Adult} \sqcap \mathsf{Male} \sqcap \mathsf{Person}, && \mathsf{Kid} \sqsubseteq \mathsf{Person} \sqcap \mathsf{Young}, \\
& \mathsf{Adult} \sqcap \mathsf{Male} \sqcap \mathsf{Person} \sqsubseteq \mathsf{Man}, && \mathsf{Person} \sqcap \mathsf{Young} \sqsubseteq \mathsf{Kid}, \\
& \mathsf{Woman} \sqsubseteq \mathsf{Adult} \sqcap \mathsf{Female} \sqcap \mathsf{Person}, && \mathsf{Adult} \sqsubseteq \neg \mathsf{Young}, \\
& \mathsf{Adult} \sqcap \mathsf{Female} \sqcap \mathsf{Person} \sqsubseteq \mathsf{Woman}, && \mathsf{Male} \sqsubseteq \neg \mathsf{Female}, \\
& \exists \mathsf{hasFather}^{-} \sqsubseteq \mathsf{Man}, && (\mathit{funct}\ \mathsf{hasFather}) \qquad \}
\end{aligned}$$

$$\begin{aligned}
\mathcal{A} = \{ \quad & \mathsf{hasFather}(\mathit{tom}, \mathit{taylor}), && \mathsf{Kid}(\mathit{tom}), && \mathsf{Man}(\mathit{tom}), \\
& \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), && \mathsf{Woman}(\mathit{taylor}) && \mathsf{Woman}(\mathit{sam}) \quad \}
\end{aligned}$$

This ABox states that *tom* is both a kid and a man, that both *taylor* and *sam* are women, and that they are both fathers of *tom*. Notice that this implies that both *taylor* and *sam* are men. It is easy to see that $\mathcal{K}$ is unsatisfiable, since the two assertions $\mathsf{Kid}(\mathit{tom})$ and $\mathsf{Man}(\mathit{tom})$ violate the disjointness between kid and man. Moreover, the two assertions $\mathsf{hasFather}(\mathit{tom}, \mathit{taylor})$ and $\mathsf{hasFather}(\mathit{tom}, \mathit{sam})$ violate the functionality of the role hasFather, and both *taylor* and *sam* violate the disjointness between woman and man.

The set *AR-Set*($\mathcal{K}$) is constituted by the following $\mathcal{T}$-consistent sets of ABox assertions:

$$\begin{aligned}
&\mathit{A\text{-}rep}_1 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{taylor}), \mathsf{Kid}(\mathit{tom}), \mathsf{Woman}(\mathit{sam})\ \} \\
&\mathit{A\text{-}rep}_2 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{taylor}), \mathsf{Man}(\mathit{tom}), \mathsf{Woman}(\mathit{sam})\ \} \\
&\mathit{A\text{-}rep}_3 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Kid}(\mathit{tom}), \mathsf{Woman}(\mathit{taylor})\ \} \\
&\mathit{A\text{-}rep}_4 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Man}(\mathit{tom}), \mathsf{Woman}(\mathit{taylor})\ \}
\end{aligned}$$

□

The following theorem provides a characterization of the notion of repair.

Theorem 20. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB, and let $\mathcal{A}'$ be a set of ABox assertions. $\mathcal{A}' \in$ AR-Set($\mathcal{K}$) if and only if $\mathcal{A}'$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$.*

Proof.
($\Rightarrow$) Suppose that $\mathcal{A}' \in$ *AR-Set*($\mathcal{K}$), we prove that $\mathcal{A}'$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$. Firstly, we show that $\mathcal{A}'$ is a subset of $\mathcal{A}$. We proceed by contradiction. Suppose that there is an assertion $\alpha \in \mathcal{A}'$ such that $\alpha \notin \mathcal{A}$. Since $\mathcal{A}'$ is $\mathcal{T}$-consistent, then $\mathcal{A}' \setminus \{\alpha\}$ is $\mathcal{T}$-consistent too. Hence, $\mathcal{A}' \setminus \{\alpha\}$ is a $\mathcal{T}$-consistent set of ABox assertions that has the same deletions of $\mathcal{A}'$ with

[1] http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/people+pets.owl.rdf

respect to $\mathcal{A}$, and that has fewer insertions than $\mathcal{A}'$ with respect to $\mathcal{A}$, which contradicts that $\mathcal{A}' \in \mathit{AR\text{-}Set}(\mathcal{K})$. Now we show that $\mathcal{A}'$ is a subset of $\mathcal{A}$ that is maximal. Toward a contradiction, suppose that $\mathcal{A}'$ is not a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$. This means that there exists an assertion $\alpha \in \mathcal{A}$ such that $\mathcal{A}' \cup \{\alpha\}$ is $\mathcal{T}$-consistent. Hence, $\mathcal{A}' \cup \{\alpha\}$ is a $\mathcal{T}$-consistent set of ABox assertions that has fewer deletions than $\mathcal{A}'$ with respect to $\mathcal{A}$, which contradicts that $\mathcal{A}' \in \mathit{AR\text{-}Set}(\mathcal{K})$.

($\Leftarrow$) Suppose that $\mathcal{A}'$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$. We prove that $\mathcal{A}' \in \mathit{AR\text{-}Set}(\mathcal{K})$. First we observe that $\mathcal{A}'$ is $\mathcal{T}$-consistent and then it is a candidate for being an A-repair. Suppose, by contradiction, that $\mathcal{A}' \notin \mathit{AR\text{-}Set}(\mathcal{K})$. The following two cases are possible.

1. There exists in $\mathit{AR\text{-}Set}(\mathcal{K})$ a set of ABox assertions $\mathcal{A}''$ such that $\mathcal{A}''$ has fewer deletions than $\mathcal{A}'$ with respect to $\mathcal{A}$. This means that there exists an assertion $\beta \in \mathcal{A}$ such that $\beta \in \mathcal{A}''$ and $\beta \notin \mathcal{A}'$ and that $\mathcal{A}' \cup \{\beta\}$ is $\mathcal{T}$-consistent. Hence, $\mathcal{A}'$ is not a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$, which is a contradiction.

2. There exists in $\mathit{AR\text{-}Set}(\mathcal{K})$ a set of ABox assertions $\mathcal{A}''$ such that $\mathcal{A}''$ has the same deletions of $\mathcal{A}'$ with respect to $\mathcal{A}$, and $\mathcal{A}''$ has fewer insertions than $\mathcal{A}'$ with respect to $\mathcal{A}$. Hence, there exists an assertion $\gamma \in \mathcal{A}'$ such that neither $\gamma \in \mathcal{A}''$ nor $\gamma \in \mathcal{A}$. Hence, $\mathcal{A}' \not\subseteq \mathcal{A}$, which is a contradiction.

■

Note that the above theorem guarantees that any A-repair of a KB $\mathcal{K}$ is finite, and therefore is actually an ABox, and that the set of A-repairs of $\mathcal{K}$ is finite. This property of the set $\mathit{AR\text{-}Set}(\mathcal{K})$, together with the fact that Lemma 17 shows that the relation "has fewer changes" is a strict partial order on a set of ABoxes, allows us to provide the following result.

Proposition 13. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. $\mathit{AR\text{-}Set}(\mathcal{K}) \neq \emptyset$ if and only if $\langle \mathcal{T}, \emptyset \rangle$ is consistent.*

Proof.
($\Rightarrow$) Suppose that $\mathit{AR\text{-}Set}(\mathcal{K}) \neq \emptyset$. We show that $\langle \mathcal{T}, \emptyset \rangle$ is consistent. Toward a contradiction, suppose that $\mathit{Mod}(\langle \mathcal{T}, \emptyset \rangle) = \emptyset$. Since $\mathit{AR\text{-}Set}(\mathcal{K}) \neq \emptyset$, there exists an A-repair $\mathcal{A}'$ of $\mathcal{K}$. From Definition 24 we have that $\mathit{Mod}(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$, but $\mathit{Mod}(\langle \mathcal{T}, \mathcal{A}' \rangle) \subseteq \mathit{Mod}(\langle \mathcal{T}, \emptyset \rangle) = \emptyset$, which is a contradiction.

($\Leftarrow$) We have to prove that if $\mathit{Mod}(\langle \mathcal{T}, \emptyset \rangle) \neq \emptyset$, then $\mathit{AR\text{-}Set}(\mathcal{K}) \neq \emptyset$. Let $\mathcal{U}$ be the set containing all the $\mathcal{T}$-consistent subsets of $\mathcal{A}$. Since $\mathit{Mod}(\langle \mathcal{T}, \emptyset \rangle) \neq \emptyset$, then $\mathcal{U}$ is non-empty, moreover, from Theorem 20, we have that $\mathit{AR\text{-}Set}(\mathcal{K}) \subseteq \mathcal{U}$. By using Lemma 17 it follows that the relation "has fewer changes" is a

strict partial order on $\mathcal{U}$, and therefore we can conclude that the set $\mathcal{U}$ is a finite partially ordered set. Hence, there is in $\mathcal{U}$ at least one ABox $\mathcal{A}'$ for which there does not exist any ABox $\mathcal{A}'' \in \mathcal{U}$ such that $\mathcal{A}''$ has fewer changes than $\mathcal{A}'$ with respect to $\mathcal{A}$. According to Definition 24, this means that $\mathcal{A}' \in AR\text{-}Set(\mathcal{K})$, and therefore $AR\text{-}Set(\mathcal{K})$ is non-empty. ∎

In the rest of this work we always assume that if $\langle \mathcal{T}, \mathcal{A} \rangle$ is a KB, then $Mod(\langle \mathcal{T}, \emptyset \rangle) \neq \emptyset$.

Although Definition 24 can be considered to some extent the natural choice for the setting we are considering, since each A-repairs stays as close as possible to the original ABox, it has the characteristic to be dependent from the form of the original ABox. Suppose that $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ differs from the inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, simply because $\mathcal{A}'$ includes assertions that logically follow, using $\mathcal{T}$, from a consistent subset of $\mathcal{A}$ (implying that $\mathcal{K}'$ is also inconsistent). One could argue that the repairs of $\mathcal{K}'$ and the repairs of $\mathcal{K}$ should coincide. Conversely, the next example shows that the two sets of A-repairs are generally different.

Example 25. Consider the KB $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{T}$ is the same TBox of the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ of Example 24, and the ABox $\mathcal{A}'$ is as follows:

$$\begin{array}{lllll} \mathcal{A}' = \{ & \mathsf{hasFather}(tom, taylor), & \mathsf{Kid}(tom), & \mathsf{Man}(tom), & \\ & \mathsf{hasFather}(tom, sam), & \mathsf{Woman}(taylor) & \mathsf{Woman}(sam) & \\ & \mathsf{Male}(tom) & & & \} \end{array}$$

Notice that $\mathcal{A}'$ can be obtained by adding $\mathsf{Male}(tom)$ to $\mathcal{A}$. Since $\mathsf{Male}(tom)$ is entailed by the KB $\langle \mathcal{T}, \{\mathsf{Man}(tom)\} \rangle$, i.e., a KB constituted by the TBox $\mathcal{T}$ of $\mathcal{K}$ and a subset of $\mathcal{A}$ that is consistent with $\mathcal{T}$, one intuitively would expect that $\mathcal{K}$ and $\mathcal{K}'$ have the same repairs under the A-semantics. This is however not the case, since we have that $AR\text{-}Set(\mathcal{K}')$ is formed by:

$$\begin{array}{l} A\text{-}rep'_1 = \{ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Woman}(sam), \mathsf{Male}(tom) \} \\ A\text{-}rep'_2 = \{ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Woman}(sam), \mathsf{Male}(tom) \} \\ A\text{-}rep'_3 = \{ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Woman}(taylor), \mathsf{Male}(tom) \} \\ A\text{-}rep'_4 = \{ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Woman}(taylor), \mathsf{Male}(tom) \} \end{array}$$

Let us finally consider the ground sentence $\mathsf{Male}(tom)$. It is easy to see that $\mathsf{Male}(tom)$ is entailed by every KB $\langle \mathcal{T}, A\text{-}rep'_i \rangle$, where $A\text{-}rep'_i \in AR\text{-}Set(\mathcal{K}')$, but it is not entailed by every KB $\langle \mathcal{T}, A\text{-}rep_i \rangle$, where $A\text{-}rep_i \in AR\text{-}Set(\mathcal{K})$. □

Depending on the particular scenario, and the specific application at hand, the above behavior might be considered incorrect. This motivates the presentation of a new definition of repair that does not present such a characteristic.

According to this new definition, that we call *Closed ABox* repair (*CA*-repair), a repair takes into account not only the assertions explicitly included in the original ABox, but also those that are implied, through the TBox, by at least one subset of the ABox that is consistent with the TBox.

To formalize the above idea, we need to introduce some preliminary definitions. Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ let $\Gamma_{\mathcal{K}}$ denote the alphabet of $\mathcal{K}$, i.e., the set of concepts, role, attributes, and individual names occurring in $\mathcal{K}$. We recall that, given an alphabet Γ, we denote with $HB(\Gamma)$ the *Herbrand Base of* Γ, i.e., the set of atomic ABox assertions that can be built over the alphabet Γ. Then, given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we define the *consistent logical consequences* of $\mathcal{A}$ with respect to $\mathcal{T}$ as the following set.

$$\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) = \{\alpha \mid \alpha \in HB(\Gamma_{\mathcal{K}}), \text{ and there exists } \mathcal{A}' \subseteq \mathcal{A} \text{ such that } \mathcal{A}' \text{ is } \mathcal{T}\text{-consistent, and } \langle \mathcal{T}, \mathcal{A}' \rangle \models \alpha\}$$

Finally, we say that two KBs $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}, \mathcal{A}' \rangle$ are *consistently equivalent* (*C-equivalent*) if $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$.

It is easy to see that if $\langle \mathcal{T}, \mathcal{A} \rangle$ is a consistent KB, then $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$. We argue that the consistent logical consequences $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ of an ABox $\mathcal{A}$ captures a reasonable notion of closure of $\mathcal{A}$, both in the case where $\mathcal{A}$ is $\mathcal{T}$-consistent and in the case where $\mathcal{A}$ is $\mathcal{T}$-inconsistent.

Based on *C-equivalence*, we propose the following notion of *CA*-repair.

Definition 25. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB in $\mathcal{L}$. A *Closed ABox repair (CA-repair)* $\mathcal{A}'$ of $\mathcal{K}$ is a set of ABox assertions such that:

1. $\mathcal{A}'$ is $\mathcal{T}$-consistent, i.e., $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$, and
2. there is no set $\mathcal{A}''$ of ABox assertions that is $\mathcal{T}$-consistent, and such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$ has fewer changes than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

Intuitively, a *CA*-repair is a $\mathcal{T}$-consistent ABox whose closure with respect to $\mathcal{T}$ "minimally" differs from $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. The set of all *CA*-repairs of a KB $\mathcal{K}$ is denoted by *CAR-Set*($\mathcal{K}$).

The following example illustrates the notion of *CA*-repair.

Example 26. Consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ presented in Example 24, it is easy to verify that the set $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ containing the consistent logical consequences of

$\mathcal{A}$ with respect to $\mathcal{T}$ is:

$$\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) = \{$$

$\mathsf{Man}(tom)$,	$\mathsf{Man}(taylor)$,	$\mathsf{Man}(sam)$,
$\mathsf{Person}(tom)$,	$\mathsf{Person}(taylor)$,	$\mathsf{Person}(sam)$,
$\mathsf{Male}(tom)$,	$\mathsf{Male}(taylor)$,	$\mathsf{Male}(sam)$,
$\mathsf{Adult}(tom)$,	$\mathsf{Adult}(taylor)$,	$\mathsf{Adult}(sam)$,
$\mathsf{hasFather}(tom, taylor)$,	$\mathsf{Kid}(tom)$,	$\mathsf{Young}(tom)$,
$\mathsf{hasFather}(tom, sam)$,	$\mathsf{Woman}(taylor)$,	$\mathsf{Woman}(sam)$,
$\mathsf{Female}(taylor)$,	$\mathsf{Female}(sam)$,	$\}$

The set *CAR-Set*($\mathcal{K}$) is constituted, up to logical equivalence with respect to the TBox $\mathcal{T}$, by the following $\mathcal{T}$-consistent ABoxes:

$CA\text{-}rep_1 = \{ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Man}(taylor) \}$
$CA\text{-}rep_2 = \{ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Woman}(taylor) \}$
$CA\text{-}rep_3 = \{ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Man}(sam) \}$
$CA\text{-}rep_4 = \{ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Woman}(sam) \}$
$CA\text{-}rep_5 = \{ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Man}(taylor), \mathsf{Male}(tom) \}$
$CA\text{-}rep_6 = \{ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Woman}(taylor), \mathsf{Male}(tom) \}$
$CA\text{-}rep_7 = \{ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Man}(sam), \mathsf{Male}(tom) \}$
$CA\text{-}rep_8 = \{ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Woman}(sam), \mathsf{Male}(tom) \}$

Let $\mathcal{K}'$ be the KB presented in Example 25. Clearly, $\mathcal{K}$ and $\mathcal{K}'$ are C-equivalent. Indeed, it is easy to verify that *CAR-Set*($\mathcal{K}$) = *CAR-Set*($\mathcal{K}'$). □

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB. In what follows, we show that both the analogous of Theorem 20 for *CA*-repair and the analogous of Proposition 13 for *CA*-repair hold.

Theorem 21. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB in $\mathcal{L}$, and $\mathcal{A}'$ be a set of ABox assertions. $\mathcal{A}' \in$ CAR-Set($\mathcal{K}$) if and only if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ is a maximal $\mathcal{T}$-consistent subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.*

Proof.
($\Rightarrow$) Suppose that $\mathcal{A}' \in$ *CAR-Set*($\mathcal{K}$). We prove that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ is a maximal $\mathcal{T}$-consistent subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

First we prove that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ is a subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Suppose, by contradiction, that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ is not a subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. This implies that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \neq \emptyset$. Let $\mathcal{A}'' = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \cap \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. We observe that, since $\mathcal{A}'$ is $\mathcal{T}$-consistent, then every subset of $\mathcal{A}'$ is $\mathcal{T}$-consistent, and then $\mathcal{A}''$ is $\mathcal{T}$-consistent. We also observe that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'') = \mathcal{A}''$. Hence, $\mathcal{A}''$ is a $\mathcal{T}$-consistent set of ABox assertions such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$ has the same deletions than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ with respect

to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$, and such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ has fewer insertions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$, which contradicts that $\mathcal{A}' \in \mathit{CAR\text{-}Set}(\mathcal{K})$.

Now, we show that $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ is a subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ that is maximal. Toward a contradiction, suppose that $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ is not a maximal $\mathcal{T}$-consistent subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. This means that there exists an assertion $\alpha \in \mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that the set $\mathcal{A}'' = \mathsf{cl}_\mathcal{T}(\mathcal{A}') \cup \{\alpha\}$ is $\mathcal{T}$-consistent. Note that $\mathsf{cl}_\mathcal{T}(\mathcal{A}'') \subseteq \mathsf{clc}_\mathcal{T}(\mathcal{A})$. Hence, $\mathcal{A}''$ is a $\mathcal{T}$-consistent set of ABox assertions such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ has fewer deletions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$, which contradicts that $\mathcal{A}' \in \mathit{CAR\text{-}Set}(\mathcal{K})$.

($\Leftarrow$) Suppose that $\mathcal{A}'$ is a set of ABox assertions such that $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ is a maximal $\mathcal{T}$-consistent subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. We prove that $\mathcal{A}' \in \mathit{CAR\text{-}Set}(\mathcal{K})$. First, we note that since $\mathcal{A}'$ is $\mathcal{T}$-consistent, then $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ is consistent too. Also we note that $\mathcal{A}'$ is a candidate for being an *CA*-repair of $\mathcal{K}$. Suppose, by contradiction, that $\mathcal{A}' \notin \mathit{CAR\text{-}Set}(\mathcal{K})$. The following two cases are therefore possible.

1. There exists a set of ABox assertions $\mathcal{A}''$ in $\mathit{CAR\text{-}Set}(\mathcal{K})$ such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ has fewer deletions than $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. This means that there exists an assertion $\beta \in \mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that $\beta \in \mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ and $\beta \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}')$ and that $\mathsf{cl}_\mathcal{T}(\mathcal{A}') \cup \{\beta\}$ is $\mathcal{T}$-consistent. Hence, $\mathsf{cl}_\mathcal{T}(\mathcal{A})$ is not a maximal $\mathcal{T}$-consistent subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$, which is a contradiction.

2. There exists in $\mathit{CAR\text{-}Set}(\mathcal{K})$ a set of ABox assertions $\mathcal{A}''$ such that $\mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ has the same deletions of $\mathsf{cl}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$, and $\mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ has fewer insertions than $\mathsf{cl}_{|\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. Hence, there exists an assertion $\gamma \in \mathsf{cl}_\mathcal{T}(\mathcal{A}')$ such that neither $\gamma \in \mathsf{cl}_\mathcal{T}(\mathcal{A}'')$ nor $\gamma \in \mathsf{clc}_\mathcal{T}(\mathcal{A})$. Hence, $\mathsf{cl}_\mathcal{T}(\mathcal{A}') \not\subseteq \mathsf{clc}_\mathcal{T}(\mathcal{A})$, which is a contradiction.

∎

From Theorem 21 it follows that a *CA*-repair is a $\mathcal{T}$-consistent subset of $\mathcal{A}$ whose closure with respect to $\mathcal{T}$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. Moreover, we have that any *CA*-repair of a KB $\mathcal{K}$ is finite, and therefore is actually an ABox, and that the set $\mathit{CAR\text{-}Set}(\mathcal{K})$ is finite.

Proposition 14. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. $\mathit{CAR\text{-}Set}(\mathcal{K}) \neq \emptyset$ if and only if $\langle \mathcal{T}, \emptyset \rangle$ is consistent.*

Proof. The proof is a straightforward adaptation of the one proposed for Proposition 13. ∎

8.2 Inconsistency-tolerant semantics

In this section we present our inconsistency-tolerant semantics based on the notions of repair given in the previous section.

We assume that for a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\mathcal{T}$ is consistent, whereas $\mathcal{A}$ may be $\mathcal{T}$-inconsistent, i.e., the set of models of $\mathcal{K}$ may be empty. The challenge is to provide semantic characterizations for $\mathcal{K}$, which are *inconsistency-tolerant*, i.e., they allow $\mathcal{K}$ to be interpreted with a non-empty set of models even in the case where it is inconsistent under the classical first-order semantics.

The first inconsistency-tolerant semantics we present, named *ABox Repair* semantics (*AR*-semantics), is based on the notion of *A*-repair.

Definition 26. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. According to the *ABox Repair* (*AR*) semantics, the set of models of $\mathcal{K}$, denoted $Mod_{AR}(\mathcal{K})$, is defined as follows:

$$Mod_{AR}(\mathcal{K}) = \{\mathcal{I} \mid \mathcal{I} \in Mod(\langle \mathcal{T}, \mathcal{A}_i \rangle), \text{ for some } \mathcal{A}_i \in \textit{AR-Set}(\mathcal{K})\}$$

In other words, an interpretation $\mathcal{I}$ is an *ABox Repair Model*, or simply an *AR-model*, of $\mathcal{K}$ if there exists $\mathcal{A}' \in \textit{AR-Set}(\mathcal{K})$ such that $\mathcal{I} \models \langle \mathcal{T}, \mathcal{A}' \rangle$.

The following notion of consistent entailment is the natural generalization of classical entailment to the *AR*-semantics.

Definition 27. Let $\mathcal{K}$ be a possibly inconsistent KB, and let ϕ be a first-order sentence. We say that ϕ is *AR-consistently entailed*, or simply *AR-entailed*, by $\mathcal{K}$, written $\mathcal{K} \models_{AR} \phi$, if $\mathcal{I} \models \phi$ for every $\mathcal{I} \in Mod_{AR}(\mathcal{K})$.

Example 27. Consider the DL KB $K = \langle \mathcal{T}, \mathcal{A} \rangle$ presented in Example 24 and the following FOL-sentences:

$\phi_1 : \exists x, y.\mathsf{Person}(x) \wedge \mathsf{hasFather}(x, y)$;
$\phi_2 : \exists x.\mathsf{Male}(x) \wedge \mathsf{Person}(x) \wedge \mathsf{hasFather}(x, y)$;
$\phi_3 : \exists x.\mathsf{Woman}(x)$.

ϕ_1 asks for the existence of a person who has a father, ϕ_2 asks for the existence of a parson who is a male and has a father, and ϕ_3 asks for the existence of a woman.

In accordance with Definition 27, given a FOL-sentence ϕ, we have that $\mathcal{K} \models_{AR} \phi$, if $\mathcal{I} \models \phi$ for every $\mathcal{I} \in Mod_{AR}(\mathcal{K})$. This means that $\mathcal{K}$ models ϕ under *AR*-semantics if for every *A*-repair $\textit{A-rep} \in \textit{AR-Set}(\mathcal{K})$, the consistent KB $\langle \mathcal{T}, \textit{A-rep} \rangle$ models ϕ.

Referring to the set $\textit{AR-Set}(\mathcal{K})$ presented in Example 24, we have:

$\langle \mathcal{T}, \textit{A-rep}_1 \rangle \models \phi_1$;	$\langle \mathcal{T}, \textit{A-rep}_1 \rangle \not\models \phi_2$;	$\langle \mathcal{T}, \textit{A-rep}_1 \rangle \models \phi_3$;
$\langle \mathcal{T}, \textit{A-rep}_2 \rangle \models \phi_1$;	$\langle \mathcal{T}, \textit{A-rep}_2 \rangle \models \phi_2$;	$\langle \mathcal{T}, \textit{A-rep}_2 \rangle \models \phi_3$;
$\langle \mathcal{T}, \textit{A-rep}_3 \rangle \models \phi_1$;	$\langle \mathcal{T}, \textit{A-rep}_3 \rangle \not\models \phi_2$;	$\langle \mathcal{T}, \textit{A-rep}_3 \rangle \models \phi_3$;
$\langle \mathcal{T}, \textit{A-rep}_4 \rangle \models \phi_1$;	$\langle \mathcal{T}, \textit{A-rep}_4 \rangle \models \phi_2$;	$\langle \mathcal{T}, \textit{A-rep}_4 \rangle \models \phi_3$.

Hence, $\mathcal{K} \models_{AR} \phi_1$ and $\mathcal{K} \models_{AR} \phi_3$, but $\mathcal{K} \not\models_{AR} \phi_2$. □

It is possible to show that the *AR*-semantics given above in fact coincides with the inconsistency-tolerant semantics for DL KBs presented in [70], and with the loosely-sound semantics studied in [22] in the context of inconsistent databases.

Analogously, we give the definition of *Closed ABox Repair* semantics that is based on the notion of *CA*-repair.

Definition 28. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. According to the *Closed ABox Repair* (*CAR*) semantics, the set of models of $\mathcal{K}$, denoted $Mod_{CAR}(\mathcal{K})$, is defined as follows:

$$Mod_{CAR}(\mathcal{K}) = \{\mathcal{I} \mid \mathcal{I} \in Mod(\langle \mathcal{T}, \mathcal{A}_i \rangle), \text{ for some } \mathcal{A}_i \in \mathit{CAR\text{-}Set}(\mathcal{K})\}$$

That is, an interpretation $\mathcal{I}$ is a *Closed ABox Repair model*, or simply a *CAR-model*, of $\mathcal{K}$ if there exists $\mathcal{A}' \in \mathit{CAR\text{-}Set}(\mathcal{K})$ such that $\mathcal{I} \models \langle \mathcal{T}, \mathcal{A}' \rangle$.

Next, we give the notion of *CAR*-entailment.

Definition 29. Let $\mathcal{K}$ be a possibly inconsistent KB, and let ϕ be a first-order sentence. We say that ϕ is *CAR-consistently entailed*, or simply *CAR-entailed*, by $\mathcal{K}$, written $\mathcal{K} \models_{CAR} \phi$, if $\mathcal{I} \models \phi$ for every $\mathcal{I} \in Mod_{CAR}(\mathcal{K})$.

Example 28. In this example we give an intuition of consistent entailment under *CAR*-semantics. Consider again the KB $\mathcal{K}$ of Example 24. We recall that, as shown in Example 26, the set $\mathit{CAR\text{-}Set}(\mathcal{K})$ is constituted, up to logical equivalence with respect to $\mathcal{T}$, by the following $\mathcal{T}$-consistent ABoxes:

$\mathit{CA\text{-}rep}_1 = \{\ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Man}(taylor)\ \}$
$\mathit{CA\text{-}rep}_2 = \{\ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Woman}(taylor)\ \}$
$\mathit{CA\text{-}rep}_3 = \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Man}(sam)\ \}$
$\mathit{CA\text{-}rep}_4 = \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Woman}(sam)\ \}$
$\mathit{CA\text{-}rep}_5 = \{\ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Man}(taylor), \mathsf{Male}(tom)\ \}$
$\mathit{CA\text{-}rep}_6 = \{\ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Woman}(taylor), \mathsf{Male}(tom)\ \}$
$\mathit{CA\text{-}rep}_7 = \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Man}(sam), \mathsf{Male}(tom)\ \}$
$\mathit{CA\text{-}rep}_8 = \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Woman}(sam), \mathsf{Male}(tom)\ \}$

Let ϕ_1, ϕ_2, and ϕ_3 be the FOL-sentences presented in Example 27. We have:

$\langle \mathcal{T}, \mathit{A\text{-}rep}_1 \rangle \models \phi_1$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_1 \rangle \models \phi_2$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_1 \rangle \not\models \phi_3$;
$\langle \mathcal{T}, \mathit{A\text{-}rep}_2 \rangle \models \phi_1$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_2 \rangle \models \phi_2$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_2 \rangle \models \phi_3$;
$\langle \mathcal{T}, \mathit{A\text{-}rep}_3 \rangle \models \phi_1$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_3 \rangle \models \phi_2$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_3 \rangle \not\models \phi_3$;
$\langle \mathcal{T}, \mathit{A\text{-}rep}_4 \rangle \models \phi_1$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_4 \rangle \models \phi_2$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_4 \rangle \models \phi_3$;
$\langle \mathcal{T}, \mathit{A\text{-}rep}_5 \rangle \models \phi_1$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_5 \rangle \models \phi_2$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_5 \rangle \not\models \phi_3$;
$\langle \mathcal{T}, \mathit{A\text{-}rep}_6 \rangle \models \phi_1$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_6 \rangle \models \phi_2$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_6 \rangle \models \phi_3$;
$\langle \mathcal{T}, \mathit{A\text{-}rep}_7 \rangle \models \phi_1$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_7 \rangle \models \phi_2$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_7 \rangle \not\models \phi_3$;
$\langle \mathcal{T}, \mathit{A\text{-}rep}_8 \rangle \models \phi_1$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_8 \rangle \models \phi_2$;	$\langle \mathcal{T}, \mathit{A\text{-}rep}_8 \rangle \models \phi_3$.

It follows that $\mathcal{K} \models_{AR} \phi_1$ and $\mathcal{K} \models_{AR} \phi_2$, but $\mathcal{K} \not\models_{AR} \phi_3$. □

Interestingly, the above examples show that there are sentences entailed by a KB under the *CAR*-semantics that are not entailed under the *AR*-semantics, and there are sentences entailed by a KB under the *AR*-semantics that are not entailed under the *CAR*-semantics.

As we shall see later, entailment of a union of conjunctive queries from a KB $\mathcal{K}$ is polynomially intractable with respect to ABox complexity both under the *AR*-semantics and the *CAR*-semantics, even for DLs with very limited expressive power, e.g., for *DL-Lite*$_{core}$ [31] that is the less expressive DL of the *DL-Lite* family. Since this can be an obstacle in the practical use of such semantics, we introduce here two new semantics named *Intersection ABox Repair* (*IAR*) semantics and *Intersection Closed ABox Repair* (*ICAR*) semantics, which are respectively approximations of the *AR*-semantics and of the *CAR*-semantics. We will show that under such semantics entailment of unions of conjunctive queries is tractable.

There is another reason that leads us to introduce the *IAR*-semantics and the *ICAR*-semantics. There may be scenarios where modifying the ABox for *cleaning* the KB from possible inconsistencies is needed [84]. Given a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ in a DL $\mathcal{L}$ and an inconsistency-tolerant semantics α, cleaning amounts to identify a $\mathcal{T}$-consistent ABox $\mathcal{A}'$ such that the KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ is expressible in $\mathcal{L}$ and the set of models of $\langle \mathcal{T}, \mathcal{A}' \rangle$ coincides with the set of model of $\langle \mathcal{T}, \mathcal{A} \rangle$ under the α-semantics.

Unfortunately, both the *AR*-semantics and the *CAR*-semantics are not applicable for this purpose. It is clear the similarity between the *AR*-semantics and the update semantics proposed by Fagin, Ullman, and Vardi (cf. Section 6.6), and between the *CAR*-semantics and the SOT approach for updating consistent KBs (cf. Section 6.4). Indeed, given a possibly inconsistent KB $\langle \mathcal{T}, \mathcal{A} \rangle$, in general, there may exist more than one *A*-repair and more than one *CA*-repair. For this reason, as for the aforesaid update semantics, also *AR*-semantics and *CAR*-semantics suffer from a form of inexpressibility problem. In other words, it is not ensured that there may exist a $\mathcal{T}$-consistent ABox $\mathcal{A}'$ such that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) = Mod_{AR}(\langle \mathcal{T}, \mathcal{A} \rangle)$ or $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) = Mod_{CAR}(\langle \mathcal{T}, \mathcal{A} \rangle)$.

Also in this case different solutions are conceivable. For example, one possibility is to let the user choose which repair should be consider the cleaned ABox; another one is to choose non-deterministically the cleaned ABox among the repairs of the original KB.

IAR-semantics and *ICAR*-semantics represent our solution to the inexpressibility problem of the *AR*-semantics and the *CAR*-semantics. Indeed, both *IAR*-semantics and *ICAR*-semantics are inconsistency-tolerant semantics that allow for expressing ABox repairs in terms of a single ABox.

In both cases, the approximation consists in taking as unique repair the

intersection of all the A-repairs and of all the CA-repairs, respectively. This actually corresponds to follow the WIDTIO (*When In Doubt Throw It Out*) principle, proposed in the area of belief revision and update [42, 110].

Definition 30. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. According to the *Intersection ABox Repair* (*IAR*) semantics, the set of models of $\mathcal{K}$, denoted $Mod_{IAR}(\mathcal{K})$, is defined as follows:

$$Mod_{IAR}(\mathcal{K}) = \{\mathcal{I} \mid \mathcal{I} \in Mod(\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i \rangle)\}$$

Next, we give the notions of IAR-entailment.

Definition 31. Let $\mathcal{K}$ be a possibly inconsistent KB, and let ϕ be a first-order sentence. We say that ϕ is *IAR-consistently entailed*, or simply *IAR-entailed*, by $\mathcal{K}$, written $\mathcal{K} \models_{IAR} \phi$, if $\mathcal{I} \models \phi$ for every $\mathcal{I} \in Mod_{IAR}(\mathcal{K})$.

Example 29. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be the KB presented in Example 24. We recall that the set $AR\text{-}Set(\mathcal{K})$ is constituted by the following $\mathcal{T}$-consistent ABoxes:

$$\begin{aligned}
&A\text{-}rep_1 = \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Woman}(sam)\ \}\\
&A\text{-}rep_2 = \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Woman}(sam)\ \}\\
&A\text{-}rep_3 = \{\ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Woman}(taylor)\ \}\\
&A\text{-}rep_4 = \{\ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Woman}(taylor)\ \}
\end{aligned}$$

The set of models of $\mathcal{K}$ under IAR-semantics coincides with the set of models of the KB constituted by the TBox $\mathcal{T}$ and by the ABox $\mathcal{A}^{AR}_{\cap}$ obtained by computing the intersection of all the A-repairs of $\mathcal{K}$. Therefore, we have:

$$\begin{aligned}
\mathcal{A}^{AR}_{\cap} = \\
&\{\ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Woman}(sam)\ \} \quad \cap\\
&\{\ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Woman}(sam)\ \} \quad \cap\\
&\{\ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Woman}(taylor)\ \} \quad \cap\\
&\{\ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Woman}(taylor)\ \} = \quad \emptyset
\end{aligned}$$

and then:

$$Mod_{IAR}(\mathcal{K}) = Mod(\langle \mathcal{T}, \mathcal{A}^{AR}_{\cap} \rangle) = Mod(\langle \mathcal{T}, \emptyset \rangle)$$

Now, consider the KB $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ presented in Example 25 where the TBox $\mathcal{T}$ is the same of $\mathcal{K}$ and the ABox $\mathcal{A}'$ is obtained from the ABox $\mathcal{A}$ of $\mathcal{K}$ by adding to $\mathcal{A}$ the assertion $\mathsf{Male}(tom)$. The set $AR\text{-}Set(\mathcal{K}')$ contains the following A-repairs:

$$A\text{-}rep'_1 = \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Woman}(sam), \mathsf{Male}(tom)\ \}$$
$$A\text{-}rep'_2 = \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Woman}(sam), \mathsf{Male}(tom)\ \}$$
$$A\text{-}rep'_3 = \{\ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Woman}(taylor), \mathsf{Male}(tom)\ \}$$
$$A\text{-}rep'_4 = \{\ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Woman}(taylor), \mathsf{Male}(tom)\ \}$$

In this case, we have:

$$\begin{aligned}\mathcal{A}'^{IAR}_{\cap} = \\ &\{\ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Man}(taylor)\ \} &\cap\\ &\{\ \mathsf{hasFather}(tom, sam), \mathsf{Man}(tom), \mathsf{Woman}(taylor)\ \} &\cap\\ &\{\ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Man}(sam)\ \} &\cap\\ &\{\ \mathsf{hasFather}(tom, taylor), \mathsf{Man}(tom), \mathsf{Woman}(sam)\ \} &\cap\\ &\{\ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Man}(taylor), \mathsf{Male}(tom)\ \} &\cap\\ &\{\ \mathsf{hasFather}(tom, sam), \mathsf{Kid}(tom), \mathsf{Woman}(taylor), \mathsf{Male}(tom)\ \} &\cap\\ &\{\ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Man}(sam), \mathsf{Male}(tom)\ \} &\cap\\ &\{\ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Woman}(sam), \mathsf{Male}(tom)\ \}\\ &= \{\ \mathsf{Male}(tom)\ \}\end{aligned}$$

and therefore:

$$Mod_{IAR}(\mathcal{K}) = Mod(\langle \mathcal{T}, \mathcal{A}'^{IAR}_{\cap} \rangle) = Mod(\langle \mathcal{T}, \{\ \mathsf{Male}(tom)\ \} \rangle)$$

We note that, differently to the case of the KB $\mathcal{K}$, in this case the ABox $\mathcal{A}'^{IAR}_{\cap}$ is not empty, but contains the assertion $\mathsf{Male}(tom)$. This is due to the fact that such an assertion does not take part in the violation of any assertion in the TBox $\mathcal{T}$. □

Analogously, we give below the definition of *Intersection Closed ABox Repair* (*ICAR*) semantics, and we give the notions of *ICAR*-entailment.

Definition 32. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. According to the *Intersection Closed ABox Repair* (*ICAR*) semantics, the set of models of $\mathcal{K}$, denoted $Mod_{ICAR}(\mathcal{K})$, is defined as follows:

$$Mod_{ICAR}(\mathcal{K}) = \{\mathcal{I} \mid \mathcal{I} \in Mod(\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in CAR\text{-}Set(\mathcal{K})} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle)\}$$

Note that, in the definition above, we refer to the intersection of the deductive closure with respect to $\mathcal{T}$ of all *CAR*-repairs of $\mathcal{K}$.

Definition 33. Let $\mathcal{K}$ be a possibly inconsistent KB, and let ϕ be a first-order sentence. We say that ϕ is *ICAR-consistently entailed*, or simply *ICAR-entailed*, by $\mathcal{K}$, written $\mathcal{K} \models_{ICAR} \phi$, if $\mathcal{I} \models \phi$ for every $\mathcal{I} \in Mod_{ICAR}(\mathcal{K})$.

Example 30. Consider again the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ of Example 24. As it shown in Example 26, the set $\mathit{CAR}\text{-}\mathit{Set}(\mathcal{K})$ is constituted, up to logical equivalence with respect to the TBox $\mathcal{T}$, by the following $\mathcal{T}$-consistent ABoxes:

$\mathit{CA}\text{-}\mathit{rep}_1 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Man}(\mathit{tom}), \mathsf{Man}(\mathit{taylor})\ \}$
$\mathit{CA}\text{-}\mathit{rep}_2 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Man}(\mathit{tom}), \mathsf{Woman}(\mathit{taylor})\ \}$
$\mathit{CA}\text{-}\mathit{rep}_3 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{taylor}), \mathsf{Man}(\mathit{tom}), \mathsf{Man}(\mathit{sam})\ \}$
$\mathit{CA}\text{-}\mathit{rep}_4 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{taylor}), \mathsf{Man}(\mathit{tom}), \mathsf{Woman}(\mathit{sam})\ \}$
$\mathit{CA}\text{-}\mathit{rep}_5 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Kid}(\mathit{tom}), \mathsf{Man}(\mathit{taylor}), \mathsf{Male}(\mathit{tom})\ \}$
$\mathit{CA}\text{-}\mathit{rep}_6 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Kid}(\mathit{tom}), \mathsf{Woman}(\mathit{taylor}), \mathsf{Male}(\mathit{tom})\ \}$
$\mathit{CA}\text{-}\mathit{rep}_7 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{taylor}), \mathsf{Kid}(\mathit{tom}), \mathsf{Man}(\mathit{sam}), \mathsf{Male}(\mathit{tom})\ \}$
$\mathit{CA}\text{-}\mathit{rep}_8 = \{\ \mathsf{hasFather}(\mathit{tom}, \mathit{taylor}), \mathsf{Kid}(\mathit{tom}), \mathsf{Woman}(\mathit{sam}), \mathsf{Male}(\mathit{tom})\ \}$

According to Definition 32, the set of models of the KB $\mathcal{K}$ under the ICAR-semantics coincides with the set of models of the KB constituted by the TBox $\mathcal{T}$ and by the ABox $\mathcal{A}_{\cap}^{\mathit{CAR}}$ obtained by computing the intersection of the deductive closure with respect to $\mathcal{T}$ of all the CA-repairs of $\mathcal{K}$.

$\mathcal{A}_{\cap}^{\mathit{CAR}} =$

$\{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Man}(\mathit{tom}), \mathsf{Man}(\mathit{taylor}), \mathsf{Man}(\mathit{sam}),$
$\mathsf{Person}(\mathit{tom}), \mathsf{Male}(\mathit{tom}), \mathsf{Adult}(\mathit{tom}),$
$\mathsf{Person}(\mathit{sam}), \mathsf{Male}(\mathit{sam}), \mathsf{Adult}(\mathit{sam}),$
$\mathsf{Person}(\mathit{taylor}), \mathsf{Male}(\mathit{taylor}), \mathsf{Adult}(\mathit{taylor})\ \}$ $\cap$
$\{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Man}(\mathit{tom}), \mathsf{Woman}(\mathit{taylor}), \mathsf{Man}(\mathit{sam}),$
$\mathsf{Person}(\mathit{tom}), \mathsf{Male}(\mathit{tom}), \mathsf{Adult}(\mathit{tom}),$
$\mathsf{Person}(\mathit{sam}), \mathsf{Male}(\mathit{sam}), \mathsf{Adult}(\mathit{sam}),$
$\mathsf{Person}(\mathit{taylor}), \mathsf{Female}(\mathit{taylor}), \mathsf{Adult}(\mathit{taylor})\ \}$ $\cap$
$\{\ \mathsf{hasFather}(\mathit{tom}, \mathit{taylor}), \mathsf{Man}(\mathit{tom}), \mathsf{Man}(\mathit{sam}), \mathsf{Man}(\mathit{taylor}),$
$\mathsf{Person}(\mathit{tom}), \mathsf{Male}(\mathit{tom}), \mathsf{Adult}(\mathit{tom}),$
$\mathsf{Person}(\mathit{sam}), \mathsf{Male}(\mathit{sam}), \mathsf{Adult}(\mathit{sam}),$
$\mathsf{Person}(\mathit{taylor}), \mathsf{Male}(\mathit{taylor}), \mathsf{Adult}(\mathit{taylor})\ \}$ $\cap$
$\{\ \mathsf{hasFather}(\mathit{tom}, \mathit{taylor}), \mathsf{Man}(\mathit{tom}), \mathsf{Woman}(\mathit{sam}), \mathsf{Man}(\mathit{taylor}),$
$\mathsf{Person}(\mathit{tom}), \mathsf{Male}(\mathit{tom}), \mathsf{Adult}(\mathit{tom}),$
$\mathsf{Person}(\mathit{sam}), \mathsf{Female}(\mathit{sam}), \mathsf{Adult}(\mathit{sam}),$
$\mathsf{Person}(\mathit{taylor}), \mathsf{Male}(\mathit{taylor}), \mathsf{Adult}(\mathit{taylor})\ \}$ $\cap$
$\{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Kid}(\mathit{tom}), \mathsf{Man}(\mathit{taylor}), \mathsf{Man}(\mathit{sam}),$
$\mathsf{Person}(\mathit{tom}), \mathsf{Male}(\mathit{tom}), \mathsf{Young}(\mathit{tom}),$
$\mathsf{Person}(\mathit{sam}), \mathsf{Male}(\mathit{sam}), \mathsf{Adult}(\mathit{sam}),$
$\mathsf{Person}(\mathit{taylor}), \mathsf{Male}(\mathit{taylor}), \mathsf{Adult}(\mathit{taylor})\ \}$ $\cap$
$\{\ \mathsf{hasFather}(\mathit{tom}, \mathit{sam}), \mathsf{Kid}(\mathit{tom}), \mathsf{Woman}(\mathit{taylor}), \mathsf{Man}(\mathit{sam}),$
$\mathsf{Person}(\mathit{tom}), \mathsf{Male}(\mathit{tom}), \mathsf{Young}(\mathit{tom}),$
$\mathsf{Person}(\mathit{sam}), \mathsf{Male}(\mathit{sam}), \mathsf{Adult}(\mathit{sam}),$

$$
\begin{aligned}
& \mathsf{Person}(taylor), \mathsf{Female}(taylor), \mathsf{Adult}(taylor) \ \} && \cap \\
& \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Man}(sam), \mathsf{Man}(taylor), \\
& \quad \mathsf{Person}(tom), \mathsf{Male}(tom), \mathsf{Young}(tom), \\
& \quad \mathsf{Person}(sam), \mathsf{Male}(sam), \mathsf{Adult}(sam), \\
& \quad \mathsf{Person}(taylor), \mathsf{Male}(taylor), \mathsf{Adult}(taylor) \ \} && \cap \\
& \{\ \mathsf{hasFather}(tom, taylor), \mathsf{Kid}(tom), \mathsf{Woman}(sam), \mathsf{Man}(taylor), \\
& \quad \mathsf{Person}(tom), \mathsf{Male}(tom), \mathsf{Young}(tom), \\
& \quad \mathsf{Person}(sam), \mathsf{Female}(sam), \mathsf{Adult}(sam), \\
& \quad \mathsf{Person}(taylor), \mathsf{Male}(taylor), \mathsf{Adult}(taylor) \ \} \\
= \quad & \{\ \mathsf{Person}(tom), \mathsf{Person}(taylor), \mathsf{Person}(sam), \mathsf{Male}(tom), \mathsf{Adult}(sam), \\
& \quad \mathsf{Adult}(taylor) \ \}
\end{aligned}
$$

The set of models of $\mathcal{K}$ under the $ICAR$-semantics is:

$$
\begin{aligned}
& Mod_{ICAR}(\mathcal{K}) = \\
& \quad Mod(\langle \mathcal{T}, \mathcal{A}_{\cap}^{CAR} \rangle) = \\
& \quad\quad Mod(\langle \mathcal{T}, \{\mathsf{Person}(tom), \mathsf{Person}(taylor), \mathsf{Person}(sam), \mathsf{Male}(tom), \\
& \quad\quad\quad\quad \mathsf{Adult}(sam), \mathsf{Adult}(taylor)\}\rangle)
\end{aligned}
$$

Intuitively, each assertion in $\mathcal{A}_{\cap}^{CAR}$ is an assertion belonging to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ that does not take part in the violation of any assertion in the TBox $\mathcal{T}$. □

8.3 Properties of our inconsistency-tolerant semantics

In this section we discuss some properties of the inconsistency-tolerant semantics presented in the previous section.

In what follows we assume that $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a possibly inconsistent DL KB.

First, we show that if $Mod(\mathcal{T}) \neq \emptyset$, then the set of models of $\mathcal{K}$ according with AR-, CAR- IAR-, and $ICAR$-semantics is non-empty.

Proposition 15. *Let $\mathcal{T}$ be a consistent TBox in $\mathcal{L}$, and let $\mathcal{A}$ be an ABox. For each $\alpha \in \{AR, CAR, IAR, ICAR\}$, we have that $Mod_{\alpha}(\langle \mathcal{T}, \mathcal{A} \rangle) \neq \emptyset$.*

Proof. Let $\mathcal{T}$ be a TBox such that $Mod(\mathcal{T}, \emptyset) \neq \emptyset$. At first, we prove that for every ABox $\mathcal{A}$ $Mod_{AR}(\langle \mathcal{T}, \mathcal{A} \rangle) \neq \emptyset$. Toward a contradiction, suppose that there exists an ABox $\mathcal{A}$ such that $Mod_{AR}(\langle \mathcal{T}, \mathcal{A} \rangle) = \emptyset$. This means that the set $AR\text{-}Set(\langle \mathcal{T}, \mathcal{A} \rangle) = \emptyset$. Hence, from Proposition 13 it follows that $Mod(\mathcal{T}, \emptyset) = \emptyset$, which is a contradiction. Similarly, we can prove that $Mod_{CAR}(\langle \mathcal{T}, \mathcal{A} \rangle) \neq \emptyset$. For the IAR-semantics and the $ICAR$-semantics the proof directly follows

from the fact that $Mod_{AR}(\langle \mathcal{T}, \mathcal{A} \rangle) \neq \emptyset$ and $Mod_{CAR}(\langle \mathcal{T}, \mathcal{A} \rangle) \neq \emptyset$, and from the fact that $\mathcal{L}$ is monotonic, then $Mod(\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i \rangle) \neq \emptyset$, and $Mod(\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in CAR\text{-}Set(\mathcal{K})} \mathcal{A}_i \rangle) \neq \emptyset$ ∎

The next proposition states that if $\langle \mathcal{T}, \mathcal{A} \rangle$ is a consistent KB, then AR-, CAR- IAR-, and $ICAR$-semantics coincide with the classical first-order semantics.

Proposition 16. *If $\langle \mathcal{T}, \mathcal{A} \rangle$ is a consistent KB, then*

$$Mod_{AR}(\langle \mathcal{T}, \mathcal{A} \rangle) = Mod_{CAR}(\langle \mathcal{T}, \mathcal{A} \rangle) = Mod_{IAR}(\langle \mathcal{T}, \mathcal{A} \rangle) = Mod_{ICAR}(\langle \mathcal{T}, \mathcal{A} \rangle) = Mod(\langle \mathcal{T}, \mathcal{A} \rangle).$$

Proof. At first, we prove that if $Mod(\langle \mathcal{T}, \mathcal{A} \rangle) \neq \emptyset$ then the following hold:

1. $AR\text{-}Set(\langle \mathcal{T}, \mathcal{A} \rangle) = \{\mathcal{A}\}$;
2. for each $\mathcal{A}' \in CAR\text{-}Set(\langle \mathcal{T}, \mathcal{A} \rangle)$, $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$.

With regard to the first statement. Since $Mod(\langle \mathcal{T}, \mathcal{A} \rangle) \neq \emptyset$ then, clearly, $\mathcal{A} \in AR\text{-}Set(\langle \mathcal{T}, \mathcal{A} \rangle)$ and, since $\mathcal{A} \in AR\text{-}Set(\langle \mathcal{T}, \mathcal{A} \rangle)$, every other ABox in the set $AR\text{-}Set(\langle \mathcal{T}, \mathcal{A} \rangle)$ must have no changes with respect to $\mathcal{A}$. Then we deduce that $\mathcal{A}$ is the only set in $AR\text{-}Set(\langle \mathcal{T}, \mathcal{A} \rangle)$. Similarly, as for the second statement, since $\mathcal{A} \in CAR\text{-}Set(\langle \mathcal{T}, \mathcal{A} \rangle)$, then for each other $\mathcal{T}$-consistent set $\mathcal{A}' \in CAR\text{-}Set(\langle \mathcal{T}, \mathcal{A} \rangle)$, $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ must have no changes with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$.

At this point, the claim follows directly from the definitions of AR-, CAR- IAR-, and $ICAR$-models. ∎

Now, we show that the IAR-semantics is a sound approximation of the AR-semantics, i.e., for any KB $\mathcal{K}$ $Mod_{AR}(\mathcal{K}) \subseteq Mod_{IAR}(\mathcal{K})$, implying that the logical consequences of $\mathcal{K}$ under the IAR-semantics are contained in the logical consequences of $\mathcal{K}$ under the AR-semantics, as it is stated by the following theorem.

Theorem 22. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB in $\mathcal{L}$, and ϕ a first-order sentence. If $\mathcal{T}$ is consistent, then, $\mathcal{K} \models_{IAR} \phi$ implies $\mathcal{K} \models_{AR} \phi$.*

Proof. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB in $\mathcal{L}$, and let $AR\text{-}Set(\mathcal{K})$ be the set of A-repairs of $\mathcal{K}$. First, we observe that, by assuming that $Mod(\langle T, \emptyset \rangle) \neq \emptyset$, then, from Proposition 13, it follows that $AR\text{-}Set(\mathcal{K}) \neq \emptyset$. From Definition 30 we have that $Mod_{IAR}(\mathcal{K}) = Mod(\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i \rangle)$. Let $\mathcal{A}' = \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i$. Clearly, for each $\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})$ we have that $\mathcal{A}' \subseteq \mathcal{A}_i$. Since $\mathcal{L}$ is monotonic, then for each $\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})$, $Mod(\langle \mathcal{T}, \mathcal{A}_i \rangle) \subseteq Mod(\langle \mathcal{T}, \mathcal{A}' \rangle)$. We conclude that $Mod_{AR}(\langle \mathcal{T}, \mathcal{A} \rangle) \subseteq Mod(\langle \mathcal{T}, \mathcal{A}' \rangle)$, which establishes the claim. ∎

Conversely, as Example 27 and Example 29 show, there are sentences entailed by a KB $\mathcal{K}$ under the *AR*-semantics that are not entailed by $\mathcal{K}$ under the *IAR*-semantics.

It is not difficult to prove that also in case of *CAR*- and *ICAR*-semantics, we have that the *ICAR*-semantics is a sound approximation of the *CAR*-semantics in the sense of consistent entailment of FOL-sentences.

Theorem 23. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB in $\mathcal{L}$, and ϕ a first-order sentence. If $\mathcal{T}$ is consistent, then, $\mathcal{K} \models_{ICAR} \phi$ implies $\mathcal{K} \models_{CAR} \phi$.*

Proof. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB in $\mathcal{L}$, and let *CAR-Set*($\mathcal{K}$) be the set of *CA*-repairs of $\mathcal{K}$. From Proposition 13 and from the assumption that $Mod(\langle T, \emptyset \rangle) \neq \emptyset$ we have that *CAR-Set*($\mathcal{K}$) is non-empty. From Definition 32 we have that $Mod_{ICAR}(\mathcal{K}) = Mod(\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in CAR\text{-}Set(\mathcal{K})} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle)$. Consider the set of ABox assertions $\mathcal{A}' = \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i)$. For each $\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})$ we have that $\mathcal{A}' \subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i)$. Since for each $\mathcal{T}$-consistent ABox $\mathcal{A}'''$ we have that $Mod(\langle \mathcal{T}, \mathcal{A}''' \rangle) = Mod(\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}''') \rangle)$, since $\mathcal{L}$ is monotonic, then for each $\mathcal{A}_i \in CAR\text{-}Set(\mathcal{K})$, $Mod(\langle \mathcal{T}, \mathcal{A}_i \rangle) \subseteq Mod(\langle \mathcal{T}, \mathcal{A}' \rangle)$. Hence $Mod_{CAR}(\langle \mathcal{T}, \mathcal{A} \rangle) \subseteq Mod(\langle \mathcal{T}, \mathcal{A}' \rangle)$, which establishes the claim. ∎

One can easily see that the converse is not true in general.

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ be two *C*-equivalent KBs, as shown in Example 25, it may happen that $Mod_{AR}(\mathcal{K}) \neq Mod_{AR}(\mathcal{K}')$. Clearly, it may also happens that $Mod_{IAR}(\mathcal{K}) \neq Mod_{IAR}(\mathcal{K}')$. As we said earlier, there may exist settings in which such a behavior might be considered incorrect. The next proposition shows that both *CAR*-semantics and *ICAR*-semantics do not present such a characteristic.

Proposition 17. *Let $\mathcal{K}_1 = \langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\mathcal{K}_2 = \langle \mathcal{T}, \mathcal{A}_2 \rangle$ be two possibly inconsistent KBs, such that $\mathcal{K}_1$ and $\mathcal{K}_2$ are C-equivalent. Then for every $\alpha \in \{CAR, ICAR\}$ we have that $Mod_\alpha(\mathcal{K}_1) = Mod_\alpha(\mathcal{K}_2)$.*

Proof. Suppose that $\mathcal{K}_1 = \langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\mathcal{K}_2 = \langle \mathcal{T}, \mathcal{A}_2 \rangle$ are C-equivalent. We prove that $Mod_{CAR}(\mathcal{K}_1) = Mod_{CAR}(\mathcal{K}_2)$ by shoving that every *CA*-repair of $\mathcal{K}_1$ is also a *CA*-repair of $\mathcal{K}2$. Toward a contradiction, suppose that there exists a *CA*-repairs $\mathcal{A}'$ in *CAR-Set*($\mathcal{K}_1$) such that $\mathcal{A}' \notin CAR\text{-}Set(\mathcal{K}_2)$. Since $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_1) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_2)$, then there is no set $\mathcal{A}''$ of ABox assertions that is $\mathcal{T}$-consistent, and such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$ has fewer changes than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_2)$, which means that $\mathcal{A}'$ is a *CA*-repair of $\mathcal{K}_2$. Hence we have a contradiction.

The proof of $Mod_{ICAR}(\mathcal{K}_1) = Mod_{ICAR}(\mathcal{K}_2)$ follows directly from the fact that *CAR-Set*($\mathcal{K}_1$) = *CAR-Set*($\mathcal{K}_2$). ∎

In other words, Proposition 17 states that both *CAR*-semantics and *ICAR*-semantics are closed with respect to the *C-equivalence*.

Next, we consider the instance checking problem under *CAR*-semantics, and show that instance checking under *CAR*-semantics coincides with instance checking under the *ICAR*-semantics.

Lemma 29. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in $\mathcal{L}$, and let α be an ABox assertion. If $\mathcal{T}$ is consistent, then $\mathcal{K} \models_{CAR} \alpha$ if and only if $\mathcal{K} \models_{ICAR} \alpha$.*

Proof.
($\Rightarrow$) Let α be an ABox assertion such that $\mathcal{K} \models_{CAR} \alpha$. We prove that $\mathcal{K} \models_{ICAR} \alpha$. The proof proceeds by contradiction as follows. Suppose that $\mathcal{K} \not\models_{ICAR} \alpha$. This means that there is a *CA*-repair $\mathcal{A}'$ of $\mathcal{K}$ such that $\alpha \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ and then $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models \alpha$. From Definition 29 it follows that $\mathcal{K} \not\models_{CAR} \alpha$, which is a contradiction.
($\Leftarrow$) Let α be an ABox assertion such that $\mathcal{K} \models_{ICAR} \alpha$. We prove that $\mathcal{K} \models_{CAR} \alpha$. The proof follows directly from Theorem 23 that states that the *ICAR*-semantics is a sound approximation of the *CAR*-semantics. ∎

We remark that the analogous of Lemma 29 does *not* hold for *AR*-semantics and *IAR*-semantics, because *IAR*-semantics is not defined on the intersection of the deductive closure of the *IA*-repairs.

Next, we focus on of the *IAR*-semantics and the *ICAR*-semantics and we provide some their notable properties. First we need to introduce the notion of minimal inconsistent set. We recall that, if $\mathcal{T}$ is a set of TBox assertions, then a set V of ABox assertions is called a *$\mathcal{T}$-inconsistent set* if V is $\mathcal{T}$-inconsistent.

Definition 34. Let $\mathcal{T}$ be a set of TBox assertions and let V be a set of ABox assertions. We say that V is a minimal $\mathcal{T}$-inconsistent set if V is $\mathcal{T}$-inconsistent and there is no proper subset V' of V that is a $\mathcal{T}$-inconsistent set, i.e. for each assertions $\alpha \in V'$, the KB $\langle \mathcal{T}, V' \setminus \{\alpha\} \rangle$ is consistent.

With the notion of minimal $\mathcal{T}$-inconsistent set in place, we give the following theorem.

Theorem 24. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an inconsistent DL KB, and let α be an ABox assertion in $\mathcal{A}$. If $\mathcal{T}$ is consistent, then an A-repair $\mathcal{A}'$ of $\mathcal{K}$ such that $\alpha \notin \mathcal{A}'$ exists if and only if there exists a minimal $\mathcal{T}$-inconsistent set V in $\mathcal{A}$ such that $\alpha \in V$.*

Proof.
($\Rightarrow$) Suppose that $\mathcal{A}'$ is an *A*-repair of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ that does not contain α. We prove that there exists a subset V of $\mathcal{A}$ containing α that is a minimal $\mathcal{T}$-inconsistent set. By exploiting Theorem 20 we have that $\mathcal{A}'$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$. It follows that $\mathcal{A}' \cup \{\alpha\}$ is $\mathcal{T}$-inconsistent, and then $\mathcal{A}' \cup \{\alpha\}$ is a $\mathcal{T}$-inconsistent set. There are two possible cases:

1. $\mathcal{A}' \cup \{\alpha\}$ is a minimal $\mathcal{T}$-inconsistent set;
2. there exists a proper subset V' of $\mathcal{A}' \cup \{\alpha\}$ which is a minimal $\mathcal{T}$-inconsistent set, and, since $\mathcal{A}' \setminus \{\alpha\}$ does not contain any $\mathcal{T}$-inconsistent set, it follows that V' contains α.

In both cases, there is a minimal $\mathcal{T}$-inconsistent set V contained in $\mathcal{A}$ such that $\alpha \in V$.
($\Leftarrow$) Suppose that V is a subset of $\mathcal{A}$ containing α that is a minimal $\mathcal{T}$-inconsistent set. We prove that there exists an A-repair $\mathcal{A}'$ of $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ that does not contains α. Toward a contradiction, suppose that every A-repair of $\mathcal{K}$ contains α, i.e., every maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$ contains α. Since V is a minimal $\mathcal{T}$-inconsistent set, we have that $V \setminus \{\alpha\}$ is $\mathcal{T}$-consistent. We have two possible cases: (i) $V \setminus \{\alpha\}$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$, hence we contradict the fact that every maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$ contains α, or, there exists a maximal $\mathcal{T}$-consistent subset $\mathcal{A}''$ of $\mathcal{A}$ such that $V \setminus \{\alpha\} \subset \mathcal{A}''$, but this is impossible since every maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$ contains α. ∎

The following theorem is the analogous of Theorem 24 for the *ICAR*-semantics.

Theorem 25. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an inconsistent DL KB, and let α be an ABox assertion in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. If $\mathcal{T}$ is consistent, then a CA-repair $\mathcal{A}'$ of $\mathcal{K}$ such that $\alpha \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ exists if and only if there exists a minimal $\mathcal{T}$-inconsistent set V in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\alpha \in V$.*

Proof. The proof is a straightforward adaptation of one proposed for Theorem 24. ∎

Let now $AR\text{-}Set(\mathcal{K}) = \{\mathcal{A}_1, \ldots \mathcal{A}_n\}$ be the set of A-repairs of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and let ϕ be a first-order sentence. From Definition 26, we derive that $\mathcal{K} \models_{IAR} \phi$ if and only if there exists a subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{A}' \subseteq \mathcal{A}_i$ for every $1 \leq i \leq n$ and $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$. From the observation above, we derive the following corollary of Theorem 24, which characterizes the notion of *IAR*-entailment.

Corollary 3. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an inconsistent DL KB, and let ϕ be a first-order sentence. If $\mathcal{T}$ is consistent, then $\mathcal{K} \models_{IAR} \phi$ if and only if there exists $\mathcal{A}' \subseteq \mathcal{A}$ such that:*

(i) $\mathcal{A}'$ is $\mathcal{T}$-consistent;

(ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$;

(iii) there is no minimal $\mathcal{T}$-inconsistent set V in $\mathcal{A}$ such that $\mathcal{A}' \cap V \neq \emptyset$.

Proof.
($\Rightarrow$) Suppose that $\mathcal{K} \models_{IAR} \phi$. We prove that there exists a $\mathcal{T}$-consistent set of ABox assertions $\mathcal{A}' \subseteq \mathcal{A}$ such that: $\mathcal{A}'$ is $\mathcal{T}$-consistent, $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$, and $\mathcal{A}' \cap V = \emptyset$ for every minimal $\mathcal{T}$-inconsistent set $V \subseteq \mathcal{A}$. The proof proceeds by contradiction. From Definition 31 we have that $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *IAR*-entails ϕ if and only if $\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i \rangle \models \phi$. Let $\mathcal{U}$ be the set of all $\mathcal{T}$-consistent subset of $\mathcal{A}$. Clearly, $AR\text{-}Set(\mathcal{K}) \subseteq \mathcal{U}$. Suppose that for every $\mathcal{A}'' \in \mathcal{U}$ we have that $\langle \mathcal{T}, \mathcal{A}'' \rangle \not\models \phi$. Hence, for each A-repair $\mathcal{A}_r \in AR\text{-}Set(\mathcal{K})$, $\langle \mathcal{T}, \mathcal{A}_r \rangle \not\models \phi$, then $\mathcal{K} \not\models_{AR} \phi$. Since $\mathcal{T}$ is consistent, then from Theorem 22 we have that $\mathcal{K} \not\models_{AR} \phi$ implies $\mathcal{K} \not\models_{IAR} \phi$, which is a contradiction.

Now, suppose that there exists a $\mathcal{T}$-consistent subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$ and such that for every assertion $\gamma \in \mathcal{A}'$, $\mathcal{A}' \setminus \{\gamma\} \not\models \phi$. Moreover, suppose that there exists a minimal $\mathcal{T}$-inconsistent set in $\mathcal{A}$ such that $\mathcal{A}' \cap V \neq \emptyset$. Let β be an assertion in $\mathcal{A}' \cap V$. From Theorem 24 it follows that there exists an A-repair $\mathcal{A}_r$ of $\mathcal{K}$ such that $\beta \notin \mathcal{A}_r$, and then $\beta \notin \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i$. Hence, $\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i \rangle \not\models \phi$, which contradicts that $\mathcal{K} \models_{IAR} \phi$.
($\Leftarrow$) Suppose that $\mathcal{A}'$ is a $\mathcal{T}$-consistent subset of $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$, and such that for every minimal $\mathcal{T}$-inconsistent set $V \subseteq \mathcal{A}$ we have that $\mathcal{A}' \cap V = \emptyset$. We prove that $\mathcal{K} \models_{IAR} \phi$. Toward a contradiction, suppose that $\mathcal{K} \not\models_{IAR} \phi$. This means that $\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i \rangle \not\models \phi$, hence $\mathcal{A}' \not\subseteq \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i$. It follows that there exists an assertion $\beta \in \mathcal{A}'$ such that $\beta \notin \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i$. Therefore, there exists an A-repair $\mathcal{A}_i$ in $AR\text{-}Set(\mathcal{K})$ such that $\beta \notin \mathcal{A}_i$. From Theorem 24 it follows that there exists a minimal $\mathcal{T}$-inconsistent set $V \subseteq \mathcal{A}$ such that $\beta \in V$, but this contradict that for every minimal $\mathcal{T}$-inconsistent set $V \subseteq \mathcal{A}$, $\mathcal{A}' \cap V = \emptyset$. ∎

Similarly, we give the following corollary of Theorem 25, which characterizes the notion of *ICAR*-entailment.

Corollary 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an inconsistent KB, and let ϕ be a first-order sentence. If $\mathcal{T}$ is consistent, then $\mathcal{K} \models_{ICAR} \phi$ if and only if there exists $\mathcal{A}' \subseteq \mathsf{clc}_{\mathcal{T}}(A)$ such that:*

(i) $\mathcal{A}'$ is $\mathcal{T}$-consistent;

(ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$;

(iii) there is no minimal $\mathcal{T}$-inconsistent set V in $\mathsf{clc}_{\mathcal{T}}(A)$ such that $\mathcal{A}' \cap V \neq \emptyset$.

Proof. The proof is a straightforward adaptation of one proposed for Corollary 3. ∎

By exploiting Corollary 3 and Corollary 4 it easy to see that the following lemma holds.

Lemma 30. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be an inconsistent KB, and let ϕ be a first-order sentence. If $\mathcal{T}$ is consistent, then $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} \phi$ if and only if $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \rangle \models_{IAR} \phi$.*

Proof.
($\Rightarrow$) We have to prove that if $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} \phi$, then $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \rangle \models_{IAR} \phi$. From Corollary 3 it follows that there is $\mathcal{A}' \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that: (i) $\mathcal{A}'$ is $\mathcal{T}$-consistent; (ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$; (iii) there is no minimal $\mathcal{T}$-inconsistent set V in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}' \cap V \neq \emptyset$. From Corollary 4 it directly follows that $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \rangle \models_{IAR} \phi$.
($\Leftarrow$) We have to prove that if $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \rangle \models_{IAR} \phi$, then $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} \phi$. From Corollary 4 it follows that there is $\mathcal{A}' \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that: (i) $\mathcal{A}'$ is $\mathcal{T}$-consistent; (ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$; (iii) there is no minimal $\mathcal{T}$-inconsistent set V in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}' \cap V \neq \emptyset$. Hence, from Corollary 3, $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} \phi$. ∎

We conclude this section by showing that it is possible to adopt both *IAR*-semantics and *ICAR*-semantics in order to repair a KB affected by inconsistencies regardless of the DL used to express it.

Proposition 18. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB expressed in $\mathcal{L}$. If $\mathcal{T}$ is consistent, then the following statements hold.*

1. *there exists a consistent KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ expressed in $\mathcal{L}$ such that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) = Mod_{IAR}(\langle \mathcal{T}, \mathcal{A} \rangle)$;*
2. *there exists a consistent KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ expressed in $\mathcal{L}$ such that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) = Mod_{ICAR}(\langle \mathcal{T}, \mathcal{A} \rangle)$.*

Proof. The proof directly follows from Proposition 15, from Definition 30 and Definition 32, and from the following observations.

- If $\langle \mathcal{T}, \mathcal{A} \rangle$ is a KB in $\mathcal{L}$, then $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \rangle$ is a KB in $\mathcal{L}$.
- If $\mathcal{U}$ is a set of ABoxes, and $\mathcal{T}$ is a TBox in $\mathcal{L}$, then $\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathcal{A}_i \rangle$ is a KB in $\mathcal{L}$.

∎

8.4 On the decidability of consistent query answering

In this section we study the problem of answering boolean union of conjunctive queries under the inconsistency-tolerant semantics given in Section 8.2. More precisely, in what follows, without referring to a particular DL $\mathcal{L}$, we identify the conditions under which the problem of answering a query posed over a

possibly inconsistent KB under such semantics is decidable. In the rest of this section we consider only consistent TBoxes.

We start by focusing on the AR-semantics.

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB expressed in a DL $\mathcal{L}$. Theorem 20 states that an ABox $\mathcal{A}'$ is an AR-repair of $\mathcal{K}$ if $\mathcal{A}'$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$. This means that, in order to guarantee that $\mathcal{A}'$ is an AR-repair of $\mathcal{K}$, the following conditions must be satisfied:

1. $\mathcal{A}' \subseteq \mathcal{A}$, i.e., $\mathcal{A}'$ is a subset of $\mathcal{A}$;
2. $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$, i.e., $\mathcal{A}'$ is $\mathcal{T}$-consistent;
3. for every other subset $\mathcal{A}''$ of $\mathcal{A}$ such that $\mathcal{A}' \subset \mathcal{A}'' \subseteq \mathcal{A}$ we have that $Mod(\langle \mathcal{T}, \mathcal{A}'' \rangle) = \emptyset$.

Condition 1 requires that the ABox $\mathcal{A}'$ must be a subset of the original ABox $\mathcal{A}$. It follows that every AR-repair of $\mathcal{K}$ is a finite set of ABox assertions, and that the set $AR\text{-}Set(\mathcal{K})$ of all the AR-repairs of $\mathcal{K}$ is finite.

Condition 2 requires that the ABox $\mathcal{A}'$ must be $\mathcal{T}$-consistent. It follows that, in order to compute the set $AR\text{-}Set(\mathcal{K})$, the problem of checking the satisfiability of a KB has to be decidable. This poses a first restriction on the applicability of the AR-semantics.

Finally, Condition 3 requires to check that every possible superset of $\mathcal{A}'$, built using the assertions in $\mathcal{A}$, is $\mathcal{T}$-inconsistent. Intuitively, since $\mathcal{A}$ is a finite set of ABox assertions, also in this case the decidability of the problem of checking the satisfiability of a KB is needed.

The considerations above suggest a direct strategy for computing the set of all AR-repairs of a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, illustrated by the algorithm $\mathsf{ComputeARSet}(\mathcal{K})$ (Algorithm 10).

To be precise, $\mathsf{ComputeARSet}(\mathcal{K})$ is an algorithm only under the assumption that for the DL used to express $\mathcal{K}$ we have an algorithm for checking if $\mathcal{K}$ is consistent. Algorithm 10 essentially computes the set $\mathcal{R}$ containing all possible subset of $\mathcal{A}$ that are $\mathcal{T}$-consistent. Then, it removes from $\mathcal{R}$ those $\mathcal{T}$-consistent subset of $\mathcal{A}$ that are not maximal.

In what follows, we deal with termination, soundness and completeness of $\mathsf{ComputeARSet}(\mathcal{K})$.

Lemma 31. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$. If an algorithm for checking $\mathcal{K}$ for satisfiability exists, then $\mathsf{ComputeARSet}(\mathcal{K})$ terminates.*

Proof. The termination of $\mathsf{ComputeARSet}(\mathcal{K})$ directly follows from the termination of the algorithm used for checking consistency, and from the finiteness of $\mathcal{A}$. ∎

Next, we show that Algorithm 10 is sound and complete.

Input: a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: a set of $\mathcal{T}$-consistent ABoxes
begin
  $\mathcal{R} \leftarrow \emptyset$;
  foreach $\mathcal{A}' \subseteq \mathcal{A}$ **do**
    if $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$ **then**
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{A}'\}$;
  foreach $\mathcal{A}'$ and $\mathcal{A}''$ in $\mathcal{R}$ **do**
    if $\mathcal{A}' \subset \mathcal{A}''$ **then**
      $\mathcal{R} \leftarrow \mathcal{R} \setminus \{\mathcal{A}'\}$;
  return $\mathcal{R}$;
end

Algorithm 10: *ComputeARSet*($\mathcal{K}$)

Lemma 32. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$. If an algorithm for checking $\mathcal{K}$ for satisfiability exists, then*

$$\mathsf{ComputeARSet}(\mathcal{K}) \subseteq \mathit{AR\text{-}Set}(\mathcal{K}).$$

Proof. We observe that an ABox $\mathcal{A}'$ does not belong to $\mathit{AR\text{-}Set}(\mathcal{K})$ if at least one of the following statements holds:

1. $\mathcal{A}' \not\subseteq \mathcal{A}$;
2. $Mod(\mathcal{T}, \mathcal{A}') = \emptyset$;
3. there exists an ABox $\mathcal{A}''$ such that:
   (a) $\mathcal{A}'' \subseteq \mathcal{A}$;
   (b) $Mod(\mathcal{T}, \mathcal{A}'') \neq \emptyset$;
   (c) $\mathcal{A}' \subset \mathcal{A}''$.

It easy to see that for every ABox in $\mathsf{ComputeARSet}(\mathcal{K})$ no one of such statements holds. Indeed, the set $\mathcal{R}$ in ComputeARSet is initially built by adding to it only subsets of the original ABox $\mathcal{A}$ which are $\mathcal{T}$-consistent, this guarantees that statements 1 and 2 are not satisfied. Then, every ABox belonging to $\mathcal{R}$ for which there exists a superset in $\mathcal{R}$ is removed from $\mathcal{R}$. This guarantees that statement 3 does not hold. Hence, we have that $\mathsf{ComputeARSet}(\mathcal{K}) \subseteq \mathit{AR\text{-}Set}(\mathcal{K})$. ∎

Lemma 33. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$. If an algorithm for checking $\mathcal{K}$ for satisfiability exists, then*

$$\mathit{AR\text{-}Set}(\mathcal{K}) \subseteq \mathsf{ComputeARSet}(\mathcal{K}).$$

Proof. Let $\mathcal{A}'$ be an ABox belonging to $AR\text{-}Set(\mathcal{K})$. We prove that $\mathcal{A}'$ belongs to $\mathsf{ComputeARSet}(\mathcal{K})$. Toward a contradiction, suppose that $\mathcal{A}' \notin \mathsf{ComputeARSet}(\mathcal{K})$. Hence, by observing Algorithm 10, only the following cases are conceivable:

1. $\mathcal{A}' \not\subseteq \mathcal{A}$, but this contradicts the condition imposing to an A-repair to be a subset of the original ABox $\mathcal{A}'$.
2. $Mod(\mathcal{T}, \mathcal{A}') = \emptyset$, but this contradicts the condition imposing to an A-repair to be $\mathcal{T}$-consistent.
3. there exists an ABox $\mathcal{A}''$ in $\mathsf{ComputeARSet}(\mathcal{K})$ such that $\mathcal{A}' \subset \mathcal{A}''$, but, since $\mathcal{A}''$ is a $\mathcal{T}$-consistent subset of $\mathcal{A}$, this contradicts the condition imposing to an A-repair to be a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$.

This means that, if $\mathcal{A}'$ is an A-repair, then $\mathcal{A}' \in \mathsf{ComputeARSet}(\mathcal{K})$. ∎

From lemmas above, we get the following theorem that sanctions the correctness of $\mathsf{ComputeARSet}(\mathcal{K})$.

Theorem 26. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$. If an algorithm for checking $\mathcal{K}$ for satisfiability exists, then*

$$AR\text{-}Set(\mathcal{K}) = \mathsf{ComputeARSet}(\mathcal{K}).$$

Proof. The proof directly follows from Lemma 32 and Lemma 33. ∎

From Theorem 26 it follows that, under the assumption that for the DL used to express $\mathcal{K}$ we have an algorithm for checking KB satisfiability, there exists an algorithm for computing the set of all AR-repairs of $\mathcal{K}$.

Now we turn our attention to the problem of answering boolean union of conjunctive queries posed over a possibly inconsistent KB under AR-semantics.

We recall that, given a consistent KB $\mathcal{K}$, an interpretation $\mathcal{I}$ of $\mathcal{K}$ satisfies a boolean conjunctive query q, if and only if the interpretation function can be extended to the variables in q in such a way that satisfies every term in q. A query q evaluates to *true* over $\mathcal{K}$, written $\mathcal{K} \models q$, if and only if every interpretation that is a models for $\mathcal{K}$ satisfies q. A boolean UCQ $Q = \bigvee_{i=1}^{n} q_i$ evaluates to *true* over $\mathcal{K}$, written $\mathcal{K} \models Q$, if there exists $i \in \{1, \ldots n\}$ such that q_i evaluates to *true* over $\mathcal{K}$. By exploiting Definition 27, we extend the above notion of boolean UCQ entailment to the AR-semantics. A query q evaluates to *true* over $\mathcal{K}$ under AR-semantics, written $\mathcal{K} \models_{AR} q$, if and only if every interpretation that is an AR-models for $\mathcal{K}$ satisfies q. That is, $\mathcal{K} \models_{AR} q$ if and only if for every $\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})$, $\langle \mathcal{T} \mathcal{A}_i \rangle \models q$. In this case we say that $\mathcal{K}$ AR-entails q. A boolean UCQ $Q = \bigvee_{i=1}^{n} q_i$ evaluates to *true* over a possibly inconsistent KB $\mathcal{K}$ under AR-semantics, written $\mathcal{K} \models_{AR} Q$, if there exists $i \in \{1, \ldots n\}$ such that q_i is AR-entailed by $\mathcal{K}$.

Under the assumption that for the DL used to express $\mathcal{K}$ we have an algorithm both for checking KB satisfiability, which allows us to compute the set $AR\text{-}Set(\mathcal{K})$, and for answering queries posed to the KB, given a boolean UCQ Q, and a possibly inconsistent KB $\mathcal{K}$, the most direct way to compute $\mathcal{K} \models_{AR} Q$ would be the one sketched in algorithm $\mathsf{NaiveComputeAnswers}_{AR}$, that is also the method implicitly used in Example 27.

Input: a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and an UCQ $\mathcal{Q}$
Output: `true` or `false`
begin
  $\mathcal{R} \leftarrow \mathsf{ComputeARSet}(\mathcal{K})$;
  foreach $\mathcal{A}' \in \mathcal{R}$ **do**
    if $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models Q$ **then**
      return `false`;
  return `true`;
end

Algorithm 11: $\mathsf{NaiveComputeAnswers}_{AR}(\mathcal{Q}, \mathcal{K})$

Under the assumption that, for the DL used to express $\mathcal{K}$ we have an algorithm both for checking KB satisfiability, and for answering boolean UCQ posed to the KB, $\mathsf{NaiveComputeAnswers}_{AR}$ shows that the problem of answering union of conjunctive queries under the AR-semantics is decidable.

The following lemma shows that $\mathsf{NaiveComputeAnswers}_{AR}(Q, \mathcal{K})$ terminates.

Lemma 34. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$, and let Q be a boolean union of conjunctive queries. If both an algorithm for checking $\mathcal{K}$ for satisfiability and an algorithm for answering boolean UCQ posed to $\mathcal{K}$ exist, then $\mathsf{NaiveComputeAnswers}_{AR}(Q, \mathcal{K})$ terminates.*

Proof. the claim directly follows from the termination of Algorithm 10 and by the assumption that there exists an algorithm for answering boolean UCQ posed to $\mathcal{K}$. ∎

Given the previous lemma, by construction of Algorithm 11, we can immediately claim the correctness of $\mathsf{NaiveComputeAnswers}_{AR}(Q, \mathcal{K})$.

Theorem 27. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$, and let Q be a boolean UCQ. If both an algorithm for checking $\mathcal{K}$ for satisfiability and an algorithm for answering boolean UCQ posed to $\mathcal{K}$ exist, then $\mathcal{K} \models_{AR} Q$ if and only if $\mathsf{NaiveComputeAnswers}_{AR}(Q, \mathcal{K})$ returns true.*

Proof. The algorithm first computes the set containing the A-repairs of $\mathcal{K}$ by using the algorithm $\mathsf{ComputeARSet}(\mathcal{K})$, and then, in accordance with Definition 27, it checks that, for every A-repair $\mathcal{A}_{rep} \in \mathsf{ComputeARSet}(\mathcal{K})$, $\mathcal{Q}$ is

entailed by the consistent KB $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle$. In case at least one of these KBs does not entail $\mathcal{Q}$, $\mathsf{NaiveComputeAnswers}_{AR}(Q, \mathcal{K})$ returns *false*, otherwise it returns *true*. Thanks to observations above and to Theorem 26, we can conclude that $\mathcal{K} \models_{AR} Q$ if and only if $\mathsf{NaiveComputeAnswers}_{AR}(Q, \mathcal{K})$ returns `true`. ∎

Now, we turn to focus on the *CAR*-semantics.

The main feature that makes the *CAR*-semantics different from the *AR*-semantics consists in referring in the minimality condition which characterizes the notion of *CAR*-repair to the *consistent closure* of the original ABox instead of the original ABox alone.

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB expressed in a DL $\mathcal{L}$. By exploiting Theorem 21 we have the following characterization of a *CAR*-repair $\mathcal{A}'$ of $\mathcal{K}$.

1. $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, i.e., $\mathcal{A}'$ is a subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$;
2. $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$, i.e., $\mathcal{A}'$ is $\mathcal{T}$-consistent;
3. for every other subset $\mathcal{A}''$ of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \subset \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'') \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ we have that $Mod(\langle \mathcal{T}, \mathcal{A}'' \rangle) = \emptyset$.

We note that, since the signature $\mathcal{S}$ of $\mathcal{K}$ is finite, then also $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ is finite. Therefore, from Conditions 1 above, we have that every *CAR*-repair of $\mathcal{K}$ is a finite set of ABox assertions, and that the set *CAR-Set*($\mathcal{K}$) containing all *CAR*-repairs of $\mathcal{K}$ is finite.

Also in this case we needed that an algorithm for checking $\mathcal{K}$ for satisfiability exist for the DL used to express $\mathcal{K}$,. With such an assumption in place, we are also able to sanction the existence of an algorithm for computing $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ [8], and we can show that the computation of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ it is possible. We recall that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ contains an assertion α if and only if α is built over the signature $\mathcal{S}$ of $\mathcal{K}$, i.e., the set of concepts, role, attributes, and individual names occurring in $\mathcal{K}$, and such that there exists a $\mathcal{T}$-consistent subset S of $\mathcal{A}$ such that $\langle \mathcal{T}, S \rangle \models \alpha$. We can prove that the following algorithm can be used to compute $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

Lemma 35. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$. If an algorithm for checking $\mathcal{K}$ for satisfiability exists, then $\mathsf{ComputeCLC}(\mathcal{K})$ terminates.*

Proof. The termination follows directly from the finiteness of the ABox $\mathcal{A}$. ∎

Lemma 35 sanctions the termination of Algorithm 12. Next, we show that if $\langle \mathcal{T}, \mathcal{A} \rangle$ is the input of Algorithm 12, then Algorithm 12 returns the consistent closure of $\mathcal{A}$.

Input: a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: a set of ABox assertions
begin
  $\mathcal{C} \leftarrow \emptyset$;
  foreach $\mathcal{A}' \subseteq \mathcal{A}$ **do**
    if $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$;
  return $\mathcal{C}$;
end

Algorithm 12: ComputeCLC($\mathcal{K}$)

Lemma 36. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$. If an algorithm for checking $\mathcal{K}$ for satisfiability exists, then*

$$\mathit{ComputeCLC}(\mathcal{K}) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}).$$

Proof. We start by proving that $\mathsf{ComputeCLC}(\mathcal{K}) \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. We observe that the algorithm $\mathsf{ComputeCLC}(\mathcal{K})$ obtains the set $\mathcal{C}$ by adding to it the closure with respect to $\mathcal{T}$ of every $\mathcal{T}$-consistent subset of $\mathcal{A}$. Toward a contradiction, suppose that $\mathsf{ComputeCLC}(\mathcal{K}) \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. It follows that there exists an assertion $\alpha \in \mathsf{ComputeCLC}(\mathcal{K})$ such that $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{K})$, and then there does not exist any $\mathcal{T}$-consistent subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle \models \alpha$. This means that, for every $\mathcal{T}$-consistent subset $\mathcal{A}'$ of $\mathcal{A}$, we have that $\alpha \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$, and then we have that $\alpha \in \mathsf{ComputeCLC}(\mathcal{K})$, which is a contradiction.

We now prove that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \subseteq \mathsf{ComputeCLC}(\mathcal{K})$. Suppose, by contradiction, that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \not\subseteq \mathsf{ComputeCLC}(\mathcal{K})$. It follows that there exists an assertion $\beta \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ that does not belong to $\mathsf{ComputeCLC}(\mathcal{K})$. Since $\beta \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then there exists a subset $\mathcal{A}'$ of $\mathcal{A}$ that is $\mathcal{T}$-consistent and such that $\langle \mathcal{T}, \mathcal{A}' \rangle \models \beta$. Now, since $\beta \notin \mathsf{ComputeCLC}(\mathcal{K})$, we have that $\beta \notin \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$, which is a contradiction. ∎

By assuming that for the DL used to express $\mathcal{K}$ an algorithm for checking $\mathcal{K}$ for satisfiability exists, we can get the following algorithm for computing the set *CAR-Set*($\mathcal{K}$).

The first step of the algorithm is to compute $\mathcal{A}_{clc}$ as the consistent closure of the original ABox $\mathcal{A}$. Afterward, it computes the set of all *CAR*-repairs of $\langle \mathcal{T}, \mathcal{A} \rangle$ by means of the algorithm ComputeARSet.

Next, we prove the termination and the correctness of $\mathsf{ComputeCARSet}(\mathcal{K})$.

Lemma 37. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$. If an algorithm for checking $\mathcal{K}$ for satisfiability exists, then $\mathit{ComputeCARSet}(\mathcal{K})$ terminates.*

Input: a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: a set of $\mathcal{T}$-consistent ABoxes
begin
  $\mathcal{A}_{clc} \leftarrow \mathsf{ComputeCLC}(\langle \mathcal{T}, \mathcal{A} \rangle)$;
  return $\mathsf{ComputeARSet}(\langle \mathcal{T}, \mathcal{A}_{clc} \rangle)$;
end

Algorithm 13: *ComputeCARSet*($\mathcal{K}$)

Proof. The termination follows directly from the termination of Algorithm 12 and Algorithm 10. ∎

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. To be precise, if $\mathcal{A}_i$ is an ABox in *CAR-Set*($\mathcal{K}$), then may be that $\mathcal{A}_i \notin \mathsf{ComputeCARSet}(\mathcal{K})$. What we show below, is that if an ABox $\mathcal{A}_i$ appears in *CAR-Set*($\mathcal{K}$) then $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \in \mathsf{ComputeCARSet}(\mathcal{K})$, i.e., that, up to logical equivalence, *CAR-Set*($\mathcal{K}$) and $\mathsf{ComputeCARSet}(\mathcal{K})$ coincide.

Theorem 28. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$. If an algorithm for checking $\mathcal{K}$ for satisfiability exists, then, up to logical equivalence, we have that*

$$\textit{CAR-Set}(\mathcal{K}) = \textit{ComputeCARSet}(\mathcal{K}).$$

Proof. In order to prove the claim, we first show that if $\langle \mathcal{T}, \mathcal{A} \rangle$ is a possibly inconsistent KB such that $\mathcal{A} = \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then, up to logical equivalence, we have that *AR-Set*($\langle \mathcal{T}, \mathcal{A} \rangle$) = *CAR-Set*($\langle \mathcal{T}, \mathcal{A} \rangle$).

Let $\mathcal{A}_{AR}$ be an ABox belonging to *AR-Set*($\langle \mathcal{T}, \mathcal{A} \rangle$), we have to prove that $\mathcal{A}_{AR} \in$ *CAR-Set*($\langle \mathcal{T}, \mathcal{A} \rangle$). Since $\mathcal{A}_{AR}$ is an *A*-repair, then, from Theorem 20, we have that $\mathcal{A}_{AR}$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$. Since $\mathcal{A} = \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then $\mathcal{A}_{AR}$ is a maximal $\mathcal{T}$-consistent subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, and then, from Theorem 21, $\mathcal{A}_{AR} \in$ *CAR-Set*($\langle \mathcal{T}, \mathcal{A} \rangle$). Similarly, we can also prove that if $\mathcal{A}_{CAR}$ is an ABox in *CAR-Set*($\langle \mathcal{T}, \mathcal{A} \rangle$), then $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_{CAR}) \in$ *AR-Set*($\langle \mathcal{T}, \mathcal{A} \rangle$).

Finally, the claim follows directly from Theorem 26 ∎

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent $\mathcal{L}$-KB, and let Q be a boolean UCQ. From Definition 29, we have that a boolean CQ q evaluates to *true* over $\mathcal{K}$ under *CAR*-semantics, written $\mathcal{K} \models_{CAR} q$, if and only if every interpretation that is a *CAR*-models for $\mathcal{K}$ satisfies q. If $\mathcal{K} \models_{CAR} q$, we say that $\mathcal{K}$ *CAR*-entails q. Note that $\mathcal{K} \models_{CAR} q$ if and only if for every $\mathcal{A}_i \in$ *CAR-Set*($\mathcal{K}$), $\langle \mathcal{T} \mathcal{A}_i \rangle \models q$. A boolean UCQ $Q = \bigvee_{i=1}^{n} q_i$ evaluates to *true* over a possibly inconsistent KB $\mathcal{K}$ under *CAR*-semantics, written $\mathcal{K} \models_{CAR} Q$, if there exists $i \in \{1, \ldots n\}$ such that q_i is *CAR*-entailed by $\mathcal{K}$.

Under the same assumptions discussed for $\mathsf{NaiveComputeAnswers}_{AR}$, i.e., when for the DL $\mathcal{L}$ used to express $\mathcal{K}$, we have an algorithm both for checking

KB satisfiability, and for answering queries posed to the KB, now that we have an algorithm for computing $CAR\text{-}Set(\mathcal{K})$, it easy to came up with the algorithm $\mathsf{NaiveComputeAnswers}_{CAR}$ for deciding if $\mathcal{K} \models_{CAR} Q$.

Input: a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and an UCQ $\mathcal{Q}$
Output: `true` or `false`
begin
  $\mathcal{R} \leftarrow (\mathcal{K})$;
  foreach $\mathcal{A}' \in \mathsf{ComputeCARSet}\mathcal{R}$ **do**
    if $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models Q$ **then**
      return `false`;
  return `true`;
end

Algorithm 14: $\mathsf{NaiveComputeAnswers}_{CAR}(\mathcal{Q}, \mathcal{K})$

The reader can verify that $\mathsf{NaiveComputeAnswers}_{CAR}$ is a light modification of $\mathsf{NaiveComputeAnswers}_{AR}$.

We have the following lemma.

Lemma 38. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$, and let Q be a boolean union of conjunctive queries. If both an algorithm for checking $\mathcal{K}$ for satisfiability and an algorithm for answering boolean UCQ posed to $\mathcal{K}$ exist, then $\mathsf{NaiveComputeAnswers}_{CAR}(Q, \mathcal{K})$ terminates.*

Proof. Termination follows directly from Lemma 37 and from the hypothesis of existence of an algorithm for answering boolean UCQ posed to $\mathcal{K}$. ∎

The following theorem shows that $\mathsf{NaiveComputeAnswers}_{CAR}(Q, \mathcal{K})$ computes $\mathcal{K} \models_{CAR} Q$.

Theorem 29. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$, and let Q be a boolean UCQ. If both an algorithm for checking $\mathcal{K}$ for satisfiability and an algorithm for answering boolean UCQ posed to $\mathcal{K}$ exist, then $\mathcal{K} \models_{CAR} Q$ if and only if $\mathsf{NaiveComputeAnswers}_{CAR}(Q, \mathcal{K})$ returns true.*

Proof. The first step of the algorithm is the computation of the set $\mathcal{R}$ containing the CA-repairs of $\mathcal{K}$. To do this, it makes use of $\mathsf{ComputeCARSet}(\mathcal{K})$. Finally, in accordance with Definition 27, it checks that, for every CA-repair $\mathcal{A}_{rep} \in \mathsf{ComputeCARSet}(\mathcal{K})$, $\mathcal{Q}$ is entailed by the consistent KB $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle$. In case at least one of these KBs does not entail $\mathcal{Q}$, $\mathsf{NaiveComputeAnswers}_{AR}(Q, \mathcal{K})$ returns *false*, otherwise it returns *true*. Since Theorem 28 ensures that every CA-repairs of $\mathcal{K}$ belong to $\mathcal{R}$, then we can conclude that $\mathcal{K} \models_{CAR} Q$ if and only if $\mathsf{NaiveComputeAnswers}_{CAR}(Q, \mathcal{K})$ returns `true`. ∎

We now discuss consistent query answering under both *IAR*-semantics and *ICAR*-semantics.

As we said in Section 8.2, *IAR*-semantics and *ICAR*-semantics are sound approximations respectively of the *AR*-semantics and of the *CAR*-semantics.

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. From Definition 31 it follows that a boolean CQ q evaluates to *true* over $\mathcal{K}$ under *IAR*-semantics, written $\mathcal{K} \models_{IAR} q$, if and only if every interpretation that is an *IAR*-models for $\mathcal{K}$ satisfies q. In Definition 30 the set $Mod_{IAR}(\mathcal{K})$ of *IAR*-models of $\mathcal{K}$ is defined as the set of models of the consistent KB $\mathcal{K}_\cap = \langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_i \rangle$. Hence, $\mathcal{K} \models_{IAR} q$ if and only if $\mathcal{K}_\cap \models q$. It follows that a boolean UCQ $Q = \bigvee_{i=1}^{n} q_i$ evaluates to *true* over a possibly inconsistent KB $\mathcal{K}$ under *IAR*-semantics, written $\mathcal{K} \models_{IAR} Q$, if there exists $i \in \{1, \ldots n\}$ such that $\mathcal{K}_\cap \models q_i$.

On the basis of the observations above we can provide the following algorithm for checking if $\mathcal{K} \models_{IAR} Q$.

Input: a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and an UCQ $\mathcal{Q}$
Output: `true` or `false`
begin
  $\mathcal{R} \leftarrow$ ComputeARSet($\mathcal{K}$);
  $\mathcal{A}_\cap = \bigcap_{\mathcal{A}_i \in \mathcal{R}} \mathcal{A}_i$;
  return $\langle \mathcal{T}, \mathcal{A}_\cap \rangle \models Q$;
end

Algorithm 15: NaiveComputeAnswers$_{IAR}(\mathcal{Q}, \mathcal{K})$

Clearly, Algorithm 15 is an algorithm only under the assumption that, for the DL used to express $\mathcal{K}$ we have an algorithm both for checking KB satisfiability, and for answering boolean UCQ posed to the KB.

We have the following results.

Lemma 39. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$, and let Q be a boolean union of conjunctive queries. If both an algorithm for checking $\mathcal{K}$ for satisfiability and an algorithm for answering boolean UCQ posed to $\mathcal{K}$ exist, then* NaiveComputeAnswers$_{IAR}(Q, \mathcal{K})$ *terminates.*

Proof. Termination follows directly from Lemma 31 and from the hypothesis of existence of an algorithm for answering boolean UCQ posed to $\mathcal{K}$. ∎

Theorem 30. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$, and let Q be a boolean UCQ. If both an algorithm for checking $\mathcal{K}$ for satisfiability and an algorithm for answering boolean UCQ posed to $\mathcal{K}$ exist, then $\mathcal{K} \models_{IAR} Q$ if and only if* NaiveComputeAnswers$_{IAR}(Q, \mathcal{K})$ *returns true.*

Proof. The proof is trivial. Indeed, it is easy to see that Algorithm 15 faithfully applies Definition 30. Hence, the claim directly follows from Theorem 26. ∎

We now turn our attention on the *ICAR*-semantics.

In Definition 30 the set $Mod_{ICAR}(\mathcal{K})$ of *ICAR*-models of $\mathcal{K}$ is defined as the set of models of the consistent KB $\mathcal{K}^*_\cap = \langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in CAR\text{-}Set(\mathcal{K})} \mathsf{cl}_\mathcal{T}(\mathcal{A}_i) \rangle$. Note that $\mathcal{K}^*_\cap$ is the KB built on the TBox $\mathcal{T}$ and on the intersection of the deductive closure with respect to $\mathcal{T}$ of all the *CAR*-repairs of $\mathcal{K}$. From Definition 33 we have that a boolean CQ q evaluates to *true* over $\mathcal{K}$ under *ICAR*-semantics, written $\mathcal{K} \models_{ICAR} q$, if and only if every interpretation in $Mod_{ICAR}(\mathcal{K})$ satisfies q. Hence, $\mathcal{K} \models_{ICAR} q$ if and only if $\mathcal{K}^*_\cap \models q$. It follows that a boolean UCQ $Q = \bigvee_{i=1}^{n} q_i$ evaluates to *true* over a possibly inconsistent KB $\mathcal{K}$ under *ICAR*-semantics, written $\mathcal{K} \models_{ICAR} Q$, if there exists $i \in \{1, \ldots n\}$ such that $\mathcal{K}^*_\cap \models q_i$.

As for the other inconsistency-tolerant semantics presented here, under the assumption that for the DL used to express $\mathcal{K}$ we have an algorithm both for checking KB satisfiability, and for answering boolean UCQ posed to the KB, we can provide the algorithm $\mathsf{NaiveComputeAnswers}_{ICAR}$ for deciding if $\mathcal{K} \models_{IAR} Q$.

Input: a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and an UCQ $\mathcal{Q}$
Output: `true` or `false`
begin
  $\mathcal{R} \leftarrow \mathsf{ComputeCARSet}(\mathcal{K})$;
  $\mathcal{A}^*_\cap = \bigcap_{\mathcal{A}_i \in \mathcal{R}} \mathsf{cl}_\mathcal{T}(\mathcal{A}_i)$;
  return $\langle \mathcal{T}, \mathcal{A}^*_\cap \rangle \models Q$;
end

Algorithm 16: $\mathsf{NaiveComputeAnswers}_{ICAR}(\mathcal{Q}, \mathcal{K})$

We conclude this section by dealing with termination and correctness of Algorithm 16.

Lemma 40. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$, and let Q be a boolean union of conjunctive queries. If both an algorithm for checking $\mathcal{K}$ for satisfiability and an algorithm for answering boolean UCQ posed to $\mathcal{K}$ exist, then $\mathsf{NaiveComputeAnswers}_{ICAR}(Q, \mathcal{K})$ terminates.*

Proof. Termination follows directly from Lemma 37 and from the hypothesis of existence of an algorithm for answering boolean UCQ posed to $\mathcal{K}$. ∎

Theorem 31. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in a DL $\mathcal{L}$, and let Q be a boolean UCQ. If both an algorithm for checking $\mathcal{K}$ for satisfiability and an algorithm for answering boolean UCQ posed to $\mathcal{K}$ exist, then $\mathcal{K} \models_{ICAR} Q$ if and only if $\mathsf{NaiveComputeAnswers}_{ICAR}(Q, \mathcal{K})$ returns true.*

Proof. As for Theorem 30, the proof is trivial. Indeed, the claim directly follows from Theorem 28. ∎

8.5 Related work

Inconsistency-tolerance has been studied in various forms in several areas, including Logic, Artificial Intelligence, and Databases.

In Logic, several semantics have been proposed to the aim of providing more meaningful notions of logical entailment for classically inconsistent theories. Some notable examples come from the field of paraconsistent logics [82, 83, 92, 106], i.e., logics that allow for inference from classically inconsistent theories in a non-trivial way. Each of the proposals has advantages and disadvantages, and the choice of the paraconsistent logic for an application depends on the requirements of the application. In what follows, we concentrate on revising the various approaches that are most related to our work, in particular, those dealing with inconsistency handling in ontologies, belief revision, databases, and specific instance-level inconsistency-tolerant semantics for DLs.

Inconsistency handling in ontologies. Several works of the Semantic Web community focus on the issue of locating inconsistencies in knowledge base. In [59], the authors present a framework for reasoning with inconsistent KBs. At the basis of such framework it is the notion of selection function, which allows for choosing some consistent sub-theory from an inconsistent KB. Standard reasoning is then applied to the selected sub-theory. An instantiation of the framework, based on a syntactic relevance-based selection function is also shortly described. In [55], a more extended framework that generalizes four approaches to inconsistency handling is presented. In particular, consistency KB evolution, repairing inconsistency, reasoning with inconsistent KB, and KB versioning are considered.

Many works in this field focus on the issue of locating inconsistencies in ontologies. In [91], the authors present a framework for detecting and diagnosing errors in OWL ontologies. In [104], the authors discuss a number of alternative methods to explain incoherence of TBoxes, unsatisfiability of concepts and concept subsumption, in order to provide support to knowledge engineers who are building terminologies using Description Logic reasoners. In [9], a visual tool is presented that allows the user to check consistency of formal KBs. In [58], the authors present algorithms and optimizations for checking OWL ontology consistency during their construction process. In [26], a tool is presented that allows to verify whether a *DL-Lite* KB is consistent. The check is reduced to evaluate first-order queries over the ABox. However, all these works provide no

support (or limited) to inconsistency management. Furthermore, with the exception of [2], they are mainly focused on inconsistencies at the terminological level.

Belief revision. The form of inconsistency-tolerance considered in this work is deeply connected to the study of update in database and belief revision/update.

Consider a consistent knowledge base $\mathcal{K}$ and a new information $\mathcal{N}$. Suppose that our intention is to change $\mathcal{K}$ with the insertion of $\mathcal{N}$. If $\mathcal{K} \cup \mathcal{N}$ is inconsistent, then update and revision semantics assume that the original knowledge base $\mathcal{K}$ has to be modified so that the result of the change is consistent. Hence, if $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A} \rangle$ is a possibly inconsistent knowledge base, with respect to the knowledge base revision/update framework, we can consider the ABox $\mathcal{A}$ as the initial knowledge, whereas the TBox $\mathcal{T}$ represents the incoming knowledge. Following the change semantics the result of the insertion should be a new consistent knowledge base. Based on such a correspondence, the studies in belief revision appear very relevant for reasoning over inconsistent KBs. In this respect, we notice also that the *AR*-semantics is strictly related to the work presented in [44] where the authors propose a framework for updating theories and logical databases. The *CAR*-semantics, instead, is comparable with the SOT-update approach presented in Chapter 6. Moreover *IAR* and the *ICAR*-semantics both conform to the WIDTIO principle of belief revision [42]. It is right to point out that, since the update semantics proposed in this work allow only for insertion of ABox assertions, they cannot be used for consistency handling purpose.

Differently, in [96], authors propose an algorithm for handling inconsistency based on a revision operator. Their approach allows to resolve conflicts changing the original knowledge base by weakening both ABox and TBox assertions. Similarly, in [85] inconsistency is resolved by transforming every concept inclusion assertions in the TBox into a cardinality restriction. Then, if a cardinality restriction is involved in a conflict, one weakens it by relaxing the restrictions of the number of elements it may have.

To the best of our knowledge, the approach studied in this work, based on the instance-level repair only, is novel for Description Logics, and it is actually inspired by the work on inconsistency management in Database.

Inconsistency tolerance in databases. In [38] the authors argue that in database context inconsistency may happen because data rise from different and independent data sources, as in data integration [71], or because data are handled by complex and long-running activities. We are convinced that the same causes can lead to inconsistency in ontology-based information systems.

Traditionally, in database, consistency is preserved by aborting update op-

eration. Clearly, such an approach is not applicable in data integration context or in scenarios where we aim to merge different data source as in [77]. In [77], authors deal with the problem of integrating information from conflicting data sources modeling integration by the operation of *merging databases* under constraints represented by first-order theories. Their approach aims to obtain a maximal amount of information from each database by means of a majority criterion used in case of conflict in which a distance criterion based on the cardinality of the symmetric difference between models is adopted. In this work they consider a large set of constraints, namely the constraints that can be expressed as first-order formulae. Unfortunately, neither computational results about their merging procedure are provided, nor an algorithm to compute consistent query answers is presented. Furthermore, the problem of dealing with incomplete databases is not taken into account.

In [43], authors maintain that inconsistencies have to be handled when they are detected. The same approach is also advocated in [38] where authors study the process for restoring database integrity with the aim of changing the original database as little as possible. This scenario is called *minimal-change integrity maintenance*. As for this work, they base their restoration process on the notion of *repair* [4]. A repair for a possibly inconsistent database is defined as a database instance that satisfies integrity constraints and minimally differs from the original database. Authors show how the notion of minimality can be interpreted in different ways, depending from the kinds of constraints are considered. Indeed for a large class of constraints, e.g. *denial constraints* which are a generalization of key dependencies and functional dependencies, violations lead to obtain a database instance where information is *complete* but not *correct*. In this case the only way to restore the integrity of a database is retract part of it. Clearly, is such a scenario in the notion of minimal changes it is sufficient consider only deletions. On the other hands, if the information is both incorrect and incomplete, as in that case where also inclusion dependencies are considered, both insertion and deletions should be considered. Alternatively, some data integration approaches give the completeness assumption up [67, 71]. In [37] authors consider inconsistency in presence of both denial constraints and inclusion dependencies, and present a semantics in which only tuple elimination is allowed.

In the works above, it is proposed to deal with inconsistency by modifying original database in order to restore integrity. A different approach to deal with inconsistency, which is the one adopted in this thesis, is *consistent query answering* [36]. In such an approach, it is embraced the idea of living with inconsistencies. In other words, following this approach, one does not attempt to modify the database in order to resolve possible inconsistency, but one tries to obtain only consistent information during query answering.

In the database area, the notion of consistent query answers was originally

given in [16]. However, the approach in [16] is completely proof-theoretic, and no computational technique for obtaining consistent answers from inconsistent database is provided.

In [4], authors define an algorithm for consistent query answers in inconsistent databases based on the notion of *residues*, originally defined in the context of semantic query optimization. The method is proved to be sound and complete only for the class of universally quantified binary constraints, i.e., constraints that involve two database relations. In [5] the same authors propose a new method that can handle arbitrary universally quantified constraints by specifying the database repairs into *logic rules with exceptions*. The problem of consistent query answering is studied in several works within the database context [22, 23, 39, 48]. In particular, the loosely-sound semantics studied in [22] in the context of inconsistent databases coincides in fact with the AR-semantics for DL KBs presented in this work.

For a survey of consistent query answers in inconsistent databases, the reader is referred to [36].

Instance-level inconsistency-tolerant semantics for DLs. Certainly, the closest approach to the one proposed in this thesis is the one reported in [70], where an inconsistency tolerant semantics for DLs is provided, and a study on the computational complexity of consistent query answering over KBs specified in *DL-Lite* is presented. The semantics proposed in [70] is based on a notion of repair that differs from the notion of A-repair given in Section 8.1 for the adopted distance criteria.

Indeed, while we say that an ABox $\mathcal{A}'$ is a repair of a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{A}'$ is $\mathcal{T}$-consistent and no ABox $\mathcal{A}''$ exists that is $\mathcal{T}$-consistent and has *fewer changes* that $\mathcal{A}'$ with respect to $\mathcal{A}$, in [70] $\mathcal{A}'$ is a repair of $\mathcal{K}$ if and only if $\mathcal{A}'$ is $\mathcal{T}$-consistent, $\mathcal{A}' \subseteq \mathcal{A}$, and no ABox $\mathcal{A}''$ exists that is $\mathcal{T}$-consistent and such that $\mathcal{A}' \subset \mathcal{A}'' \subseteq \mathcal{A}$. In other words, while for defining a repair we adopt a distance criteria based on the notion of *fewer changes* given by Fagin, Ullman and Vardi [44] that aims to give preference to such $\mathcal{T}$-consistent ABoxes which have fewer deletions over the ABoxes which have fewer insertions with respect to the original ABox, in [70] a repair is forced to be a subset of the initial ABox.

Actually, we showed in Section 8.1 (Theorem 20) that due to the monotonicity of the languages considered in this work, the notion of repair given in [70] coincides with our notion of A-repair, and then, their consistency-tolerant semantics coincides with our AR-semantics.

Recently, in [13] the author carries out an investigation for *DL-Lite*-KB, with the aim of better understanding the cases in which consistent query answering under AR-semantics is feasible, and in particular, can be done using

query rewriting. Specifically, the author formulates some general conditions which can be used to prove that a consistent query rewriting does or does not exist for a given *DL-Lite*$_{core}$ TBox and instance query. Afterwards, in [14], the same author conducts a complexity analysis of the AR-semantics with the aim to characterize the complexity of consistent query answering based on the properties of the KB and the conjunctive query at hand. By focusing on very simple language, which is a fragment of *DL-Lite*$_{core}$, the author succeeds to identify the number of quantified variables in the query as an important factor in determining the complexity of consistent query answering. To be more precise, in [14] it is shown that consistent query answering is:

- always first-order expressible for conjunctive queries with at most one quantified variable;
- the problem has polynomial data complexity when there are two quantified variables;
- the problem may become coNP-hard starting from three quantified variables.

In the same paper, the author proposes a novel inconsistency-tolerant semantics which is a sound approximation of the AR-semantics. This semantics, named *intersection of closed repairs* (ICR), corresponds to closing AR-repairs with respect to the TBox before intersecting them. More formally:

Definition 35. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. A boolean query q is said to be entailed from $\mathcal{K}$ under ICR-semantics (*intersection of closed repairs*), written $\mathcal{K} \models_{ICR} q$, if

$$\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle$$

It easy to see that the ICR-semantics approximates the AR-semantics better than the IAR-semantics as the following example shows.

Example 31. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a *DL-Lite*$_A$-KB, where:

$\mathcal{T} = \{ C_1 \sqsubseteq \exists R, \exists R^- \sqsubseteq C_2, (funct\ R) \}$
$\mathcal{A} = \{ R(o, o'), R(o, o'') \}$

The KB $\mathcal{K}$ is clearly inconsistent since the assertions in the TBox violate the functionality of the role R. The set AR-*Set*($\mathcal{K}$) is formed by the following ABoxes:

$\mathcal{A}_{rep1} = \{ R(o, o') \}$
$\mathcal{A}_{rep2} = \{ R(o, o'') \}$

Consider the following queries:

$q_1() : \exists x.C_1(x)$
$q_2() : \exists x.C_2(x)$.

It is easy to verify that:

$$\mathcal{K} \models_{AR} q_1(); \qquad \mathcal{K} \not\models_{IAR} q_1(); \qquad \mathcal{K} \models_{ICR} q_1().$$
$$\mathcal{K} \models_{AR} q_2(); \qquad \mathcal{K} \not\models_{IAR} q_2(); \qquad \mathcal{K} \not\models_{ICR} q_2().$$

□

In [14], it is shown that first-order expressibility of consistent query answering under ICR-semantics is guaranteed only for *DL-Lite*$_{core}$ ontologies without inverse roles. For *DL-Lite*$_{core}$ the problem of deciding if a possibly inconsistent *DL-Lite*$_{core}$-KB entails a boolean conjunctive query under ICR-semantics is instead coNP-hard [14].

We conclude this section by reporting some results presented in [102]. In this work, the author studies instance checking and conjunctive query entailment under AR and CAR-semantics for a wide spectrum of DLs, ranging from tractable ones ($\mathcal{EL}$) to very expressive ones ($\mathcal{SHIQ}$), showing that reasoning under the above semantics is inherently intractable, even for very simple DLs. Moreover, the author studies the sound approximations of the above semantics, namely IAR and $ICAR$-semantics, showing that reasoning under such approximated semantics is intractable even for tractable DLs. A summary of the complexity results obtained in [102] is reported in Table 8.1.

Finally, [79] presents an approach that is also very close to the present work. In fact, it proposes an application of the IAR semantics to the framework of Datalog+/- [21]. In particular, a fragment of Datalog+/- is considered that comprises linear tuple-generating dependencies (TGDs), negative constraints (NCs) and a restricted form of equality-generating dependencies (EGDs). Then, a technique for query answering under such dependencies is presented, which is technically very similar to the one that we present in Chapter 9. In fact, the dependencies considered in [79] have a tight connection with the TBox assertions of *DL-Lite*$_{A,id,den}$, in particular: linear TGDs correspond to a generalized form of positive inclusion assertions, NCs correspond to denial assertions, and EGDs are related to identification assertions. However, there are two main differences: first, the framework of [79] does not consider value-domains, hence many of the issues dealt with by our technique are not present in the above Datalog+/- framework; moreover, although identification constraints correspond to EGDs, they are not captured by the restricted form of EGDs (called non-conflicting EGDs) considered in [79].

	data complexity		**combined complexity**	
	IC	**UCQ**	**IC**	**UCQ**
$\mathcal{EL}_\bot$				
AR	coNP	coNP	coNP	Π_2^p
CAR	DP	$\Delta_2^p[\mathcal{O}(log\ n)]$	DP	Π_2^p
IAR	coNP	coNP	coNP	$\Delta_2^p[\mathcal{O}(log\ n)]$
ICAR	DP	$\Delta_2^p[\mathcal{O}(log\ n)]$	DP	$\Delta_2^p[\mathcal{O}(log\ n)]$
$\mathcal{ALC}$				
AR	Π_2^p	Π_2^p	ExpTime	ExpTime
CAR	$\mathrm{BH}_2(\Sigma_2^p)$	$\Delta_3^p[\mathcal{O}(log\ n)]$	ExpTime	ExpTime
IAR	Π_2^p	Π_2^p	ExpTime	ExpTime
ICAR	$\mathrm{BH}_2(\Sigma_2^p)$	$\Delta_3^p[\mathcal{O}(log\ n)]$	ExpTime	ExpTime
$\mathcal{SHIQ}$				
AR	Π_2^p	Π_2^p	ExpTime	2-ExpTime
CAR	$\mathrm{BH}_2(\Sigma_2^p)$	$\Delta_3^p[\mathcal{O}(log\ n)]$	ExpTime	2-ExpTime
IAR	Π_2^p	Π_2^p	ExpTime	2-ExpTime
ICAR	$\mathrm{BH}_2(\Sigma_2^p)$	$\Delta_3^p[\mathcal{O}(log\ n)]$	ExpTime	2-ExpTime

Table 8.1: Complexity of UCQ entailment over DL KBs under inconsistency-tolerant semantics.

Chapter 9

Query answering over inconsistent *DL-Lite*$_{A,id,den}$ KBs

Motivated by the fact that the size of real world KBs is scaling up and that the ability of dealing with KBs with very large ABoxes has become a crucial requirement for many modern applications, various DLs have been recently proposed that allow for tractable reasoning, still guaranteeing good expressive power. Among such DLs, the logics of the *DL-Lite* family [31, 95], and in particular *DL-Lite*$_{A,id,den}$, which is the most expressive logic in the family, present the distinguishing characteristic of enabling first-order (FOL) rewritability of query answering of unions of conjunctive queries (UCQs). This means that to answer a UCQ q in *DL-Lite* it is possible to first rewrite q into a first-order query q_r, only on the basis of the knowledge specified in the TBox, and then evaluate q_r over the ABox, which can be seen as a plain database. Notably, the ABox does not need to be touched during the rewriting phase, and no data preprocessing is needed (as for example required in [41, 64]). This turns out to be crucial, for instance, in all those applications in which KBs, and in particular their intensional component, are used to access data stored in external repositories, such as in ontology-based data integration [26, 95].

In these applications, however, even though the TBox of the ontology is usually a consistent theory, its axioms may often be contradicted by assertions of the ABox, in general collected from various autonomous sources. This actually implies that the resulting KB is inconsistent, and that reasoning over it is trivialized. The ability of dealing with such a form of inconsistency is of critical importance, in particular to obtain meaningful answers to queries posed over inconsistent KB.

For this reasons, in this chapter we study the problem of answering boolean UCQs posed to inconsistent KBs expressed in *DL-Lite*$_{A,id,den}$ under the inconsistency-tolerant semantics presented in Section 8.2.

We first focus on *AR*-semantics and we show that for the DLs of the *DL-Lite* family, inconsistency-tolerant query answering under such a semantics is coNP-complete even for ground atomic queries, thus showing that inconsistency-tolerant instance checking is already intractable.

Then we study *CAR*-semantics showing that while inconsistency-tolerant instance checking is tractable under this semantics, query answering is coNP-complete for unions of conjunctive queries.

Finally, we consider the *IAR*-semantics and the *ICAR*-semantics and we provide algorithms for the FOL-rewritability of UCQs under such semantics.

9.1 Consistent query answering in *DL-Lite*$_{A,id,den}$ under *AR*-semantics

In this section we deal with the problem of answering boolean union of conjunctive queries posed to inconsistent *DL-Lite*$_{A,id,den}$-KBs under the *AR*-semantics.

Firstly, we observe that a TBox expressed in *DL-Lite*$_{A,id,den}$ cannot be inconsistent, i.e., given a TBox $\mathcal{T}$ in *DL-Lite*$_{A,id,den}$, it is always true that $Mod(\langle \mathcal{T}, \emptyset \rangle) \neq \emptyset$. This directly follows from Lemma 1 stating that the KB $\langle \mathcal{T}_{inc}, \mathcal{A} \rangle$ is always consistent, together with the fact that from Theorem 6 follows that the violation of an assertion in $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id} \cup \mathcal{T}_{den}$ arise only if at least one of the queries in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ evaluates true over the non-empty ABox $\mathcal{A}$.

Moreover, all the properties presented in Section 8.3 hold. In particular, Proposition 15 and Propositions 16, which guarantee that the set of models of a possibly inconsistent KB according with our inconsistency-tolerant semantics is non-empty, and that in case of consistent KBs, the reasoning over inconsistency-tolerant semantics coincides with reasoning over classical FOL-semantics.

In addition, we note that since both consistency checking and query answering are decidable in *DL-Lite*$_{A,id,den}$ (cf. Section 5.1 and Section 5.2), in principle, we could use the algorithms presented in Section 8.4 for query answering under our inconsistency-tolerant semantics.

As we said in Section 8.5 the *AR*-semantics coincides with the inconsistency-tolerant semantics for DL KBs presented in [70].

From the result presented by the authors in [70, Theorem 1 and Theorem 2], we known that UCQ entailment is intractable under *AR*-semantics already for *DL-Lite*$_R$ and *DL-Lite*$_F$ [31]. Hence, we are able to give the following theorem (for the proof of the theorem refer to [70]).

Theorem 32. [70] *Let* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *be a KB in DL-Lite*$_{A,id,den}$ *and* Q *be a*

boolean union of conjunctive queries. Deciding whether $\mathcal{K} \models_{AR} Q$ *is coNP-complete with respect to data complexity.*

Here, we strengthen this result, and show that instance checking under *AR*-semantics is already coNP-hard in data complexity even if the KB is expressed in *DL-Lite*$_{core}$. We recall that *DL-Lite*$_{core}$ is the least expressive logic in the *DL-Lite* family (for more details see Section 3.1).

Theorem 33. *Let* $\mathcal{K}$ *be a DL-Lite*$_{core}$*-KB and let* α *be an ABox assertion. Deciding whether* $\mathcal{K} \models_{AR} \alpha$ *is coNP-complete with respect to data complexity.*

Proof. Membership in coNP follows from coNP-completeness of UCQ entailment under *AR*-semantics [70, Theorem 1].

We prove the coNP-hardness by reducing unsatisfiability of a 3-CNF to query entailment of a CQ in *DL-Lite*$_{core}$.

Let ϕ be a 3-CNF of the form $c_1 \wedge \ldots \wedge c_n$ with $c_i = l_i^1 \vee l_i^2 \vee l_i^3$, where every l_i^j is a literal from a set of propositional variables $\{x_1, \ldots, x_m\}$. Given a literal ℓ, let $s(\ell)$ denote the sign of ℓ, i.e., $s(\ell) = t$ if ℓ is a positive literal, and $s(\ell) = f$ otherwise; moreover, let $v(\ell)$ denote the propositional variable occurring in the literal ℓ.

We define the following *DL-Lite*$_{core}$ TBox $\mathcal{T}$:

$$\mathcal{T} = \{\exists R \sqsubseteq \mathit{Unsat}, \quad \exists R^- \sqsubseteq \neg\exists L_t^-, \quad \exists R^- \sqsubseteq \neg\exists L_f^-, \quad \exists L_t \sqsubseteq \neg\exists L_f\}$$

and the following ABox $\mathcal{A}$:

$$\mathcal{A} = \{R(a, c_i) \mid 1 \leq i \leq n\} \cup \{L_{s(l_i^j)}(v(l_i^j), c_i) \mid 1 \leq i \leq n,\ 1 \leq j \leq 3\}$$

We now prove that $\langle \mathcal{T}, \mathcal{A} \rangle \models_{AR} \mathit{Unsat}(a)$ if and only if ϕ is unsatisfiable.

First, if ϕ is satisfiable, then there exists an interpretation I for $\{x_1, \ldots, x_m\}$ such that I is a model for ϕ. Now consider the ABox

$$\begin{aligned} \mathcal{A}' \quad = \quad & \{L_t(x_j, c_i) \mid L_t(x_j, c_i) \in \mathcal{A} \text{ and } I(x_j) = \mathit{true}\} \cup \\ & \{L_f(x_j, c_i) \mid L_f(x_j, c_i) \in \mathcal{A} \text{ and } I(x_j) = \mathit{false}\} \end{aligned}$$

It is immediate to see that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is satisfiable. Moreover, since I is a model for ϕ, for every conjunct c_i of ϕ, there exists a propositional variable x_j such that either the literal x_j occurs positively in c_i and $I(x_j) = \mathit{true}$, or x_j occurs negatively in c_i and $I(x_j) = \mathit{false}$. This implies that, for every i such that $1 \leq i \leq n$, $\mathcal{A}' \cup \{R(a, c_i)\}$ is $\mathcal{T}$-inconsistent. Then, due to the presence of $\exists L_t \sqsubseteq \neg\exists L_f$ in $\mathcal{T}$, it immediately follows that, for every assertion α of the form $L_t(x_j, c_i)$ or $L_t(x_j, c_i)$ such that $\alpha \in \mathcal{A} \setminus \mathcal{A}'$, $\mathcal{A}' \cup \{\alpha\}$ is $\mathcal{T}$-inconsistent. Therefore, $\mathcal{A}'$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$. And since $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models \mathit{Unsat}(a)$, it follows that $\langle \mathcal{T}, \mathcal{A} \rangle \not\models_{AR} \mathit{Unsat}(a)$.

Then, suppose $\langle \mathcal{T}, \mathcal{A} \rangle \not\models_{AR} \mathit{Unsat}(a)$. Hence, there exists $\mathcal{A}' \subseteq \mathcal{A}$ such that $\mathcal{A}'$ is a $\mathcal{T}$-maximal consistent subset of $\mathcal{A}$ and $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models \mathit{Unsat}(a)$. Now let I be the interpretation of $\{x_1, \ldots, x_n\}$ defined as follows: $I(x_j) = \mathit{true}$ if there exists $L_t(x_j, c_i) \in \mathcal{A}'$ for some i, and $I(x_j) = \mathit{false}$ if there exists $L_f(x_j, c_i) \in \mathcal{A}'$ for some i. Now, since $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models \mathit{Unsat}(a)$, it follows that no assertion of the form $R(x_j, c_i)$ is in $\mathcal{A}'$, and since $\mathcal{A}'$ is $\mathcal{T}$-maximal, it follows that, for every i such that $1 \leq i \leq n$, there exists an assertion of the form $L_t(x_j, c_i)$ or $L_f(x_j, c_i)$ in $\mathcal{A}'$ for some j. In turn, this immediately implies that the conjunct c_i of ϕ is satisfied in I, therefore I is a model of ϕ, which proves the thesis. ∎

Note that Theorem 33 corrects a wrong result presented in [70, Theorem 6], which asserts tractability of *AR*-entailment of ABox assertions from KBs specified in *DL-Lite*$_F$, which is a superset of *DL-Lite*$_{core}$ [31]. It turns out that, while the algorithm presented in [70] (on which the above cited Theorem 6 was based) is actually unable to deal with general TBoxes, such a technique can be adapted to prove that *AR*-entailment of ABox assertions is tractable for *DL-Lite*$_A$-KBs without inclusion assertions with negation in the right-hand side, i.e. in *DL-Lite*$_A$-KBs where $\mathcal{T}_{disj} = \emptyset$.

9.2 Consistent query answering in *DL-Lite*$_{A,id,den}$ under *CAR*-semantics

In this section, we focus on the *CA*-semantics, and show that, as for *AR*-semantics, CQ entailment under this semantics is coNP-hard even if the TBox language is restricted to *DL-Lite*$_{core}$. To this aim, we give the following theorem.

Theorem 34. *Let $\mathcal{K}$ be a DL-Lite$_{core}$-KB and let q be a CQ. Deciding whether $\mathcal{K} \models_{CAR} q$ is coNP-hard with respect to data complexity.*

Proof. We prove the claim by reducing unsatisfiability of a 3-CNF to query entailment of a CQ in *DL-Lite*$_{core}$ (actually, in the DL allowing only atomic concept disjunctions in the TBox).

Let ϕ be a 3-CNF of the form $c_1 \wedge \ldots \wedge c_n$ with $c_i = l_i^1 \vee l_i^2 \vee l_i^3$, where every l_i^j is a literal from a set of propositional variables $\{x_1, \ldots, x_m\}$. Given a literal ℓ, let $s(\ell)$ denote the sign of ℓ, i.e., $s(\ell) = t$ if ℓ is a positive literal, and $s(\ell) = f$ otherwise. Moreover, let $\overline{s}$ denote the complement of s, i.e., $s(\ell) = t$ if ℓ is a negative literal, and $s(\ell) = f$ otherwise. Finally, let $v(\ell)$ denote the propositional variable occurring in the literal ℓ.

We define the following *DL-Lite*$_{core}$ TBox $\mathcal{T}$:

$$\mathcal{T} = \{ \mathit{True} \sqsubseteq \neg \mathit{False} \}$$

and the following ABox $\mathcal{A}$:

$$\begin{aligned}\mathcal{A} \;=\; & \{L^j_{s(l_i^j)}(v(l_i^j), c_i) \mid 1 \le i \le n \text{ and } 1 \le j \le 3\} \cup \\ & \{L^j_{\overline{s}(l_i^j)}(s(l_i^j), c_i) \mid 1 \le i \le n \text{ and } 1 \le j \le 3\} \cup \\ & \{\mathit{True}(x_i), \mathit{False}(x_i) \mid 1 \le i \le m\} \cup \\ & \{\mathit{True}(t), \mathit{False}(f)\}\end{aligned}$$

Now let q be the following conjunctive query:

$$\begin{aligned}q \;\leftarrow\; & L^1_f(x_1, z), \mathit{True}(x_1), L^1_t(y_1, z), \mathit{False}(y_1), \\ & L^2_f(x_2, z), \mathit{True}(x_2), L^2_t(y_2, z), \mathit{False}(y_2), \\ & L^3_f(x_3, z), \mathit{True}(x_3), L^3_t(y_3, z), \mathit{False}(y_3)\end{aligned}$$

We prove that $\langle \mathcal{T}, \mathcal{A}\rangle \models_{CA} q$ if and only if ϕ is unsatisfiable.

First, if ϕ is satisfiable, then there exists an interpretation I for $\{x_1, \ldots, x_m\}$ such that I is a model for ϕ.
Now consider the ABox

$$\begin{aligned}\mathcal{A}' \;=\; & \{L^j_t(x_k, c_i) \mid L^j_t(x_k, c_i) \in \mathcal{A}\} \cup \\ & \{L^j_f(x_k, c_i) \mid L^j_f(x_k, c_i) \in \mathcal{A}\} \cup \\ & \{\mathit{True}(x_k) \mid 1 \le k \le m \text{ and } I(x_k) = \mathit{true}\} \cup \\ & \{\mathit{False}(x_k) \mid 1 \le k \le m \text{ and } I(x_k) = \mathit{false}\} \cup \\ & \{\mathit{True}(t), \mathit{False}(f)\}\end{aligned}$$

It is immediate to see that $\langle \mathcal{T}, \mathcal{A}'\rangle$ is satisfiable and that $\mathcal{A}'$ is a maximal $\mathcal{T}$-consistent subset of $\mathcal{A}$. Moreover, since I is a model for ϕ, for every conjunct c_i of ϕ, there exists a propositional variable x_k such that:

- (i) either the literal x_k occurs positively in c_i and $I(x_k) = \mathit{true}$, and in this case there exists j such that both $\mathit{True}(x_k)$ and $L^j_t(x_k, c_i)$ belong to $\mathcal{A}'$, and therefore $\langle \mathcal{T}, \mathcal{A}'\rangle \not\models q$; or
- (ii) x_k occurs negatively in c_i and $I(x_k) = \mathit{false}$, and in this case there exists j such that both $\mathit{False}(x_k)$ and $L^j_f(x_k, c_i)$ belong to $\mathcal{A}'$, and therefore $\langle \mathcal{T}, \mathcal{A}'\rangle \not\models q$. Consequently, $\langle \mathcal{T}, \mathcal{A}'\rangle \not\models q$, which immediately implies that $\langle \mathcal{T}, \mathcal{A}\rangle \not\models_{CA} q$.

Then, suppose $\langle \mathcal{T}, \mathcal{A}\rangle \not\models_{CA} q$. Hence, there exists $\mathcal{A}' \subseteq \mathcal{A}$ such that $\mathcal{A}'$ is a $\mathcal{T}$-maximal consistent subset of $\mathcal{A}$ and $\langle \mathcal{T}, \mathcal{A}'\rangle \not\models q$. Due to the form of $\mathcal{T}$, $\mathcal{A}'$ contains the set of assertions $\{L^j_t(x_k, c_i) \mid L^j_t(x_k, c_i) \in \mathcal{A}\} \cup \{L^j_f(x_k, c_i) \mid L^j_f(x_k, c_i) \in \mathcal{A}\}$; moreover, for every k such that $1 \le k \le m$, $\mathcal{A}'$ contains exactly one of the assertions $\mathit{True}(x_k)$, $\mathit{False}(x_k)$.

Now let I be the interpretation of $\{x_1, \ldots, x_n\}$ defined as follows: for every k such that $1 \le k \le m$, $I(x_k) = \mathit{true}$ if $\mathit{True}(x_k) \in \mathcal{A}'$, and $I(x_k) = \mathit{false}$ if $\mathit{False}(x_k) \in \mathcal{A}'$. Now, since $\langle \mathcal{T}, \mathcal{A}'\rangle \not\models q$, it follows that, for every conjunct

c_i of ϕ (observe that $c_1, \ldots, c_n$ are the possible bindings of the query variable z), there exists $j \in \{1, 2, 3\}$ such that for every k such that $1 \leq k \leq m$, if $L^j_t(x_k, c_i) \in \mathcal{A}'$ then $\mathit{False}(x_k) \notin \mathcal{A}'$ (and therefore $\mathit{True}(x_k) \in \mathcal{A}'$), and if $L^j_f(x_k, c_i) \in \mathcal{A}'$ then $\mathit{True}(x_k) \notin \mathcal{A}'$ (and therefore $\mathit{False}(x_k) \in \mathcal{A}'$). This immediately implies that the conjunct c_i of ϕ is satisfied by I. Therefore, I is a model for ϕ, thus the thesis follows. ∎

Notice that, differently from the *AR*-semantics, the above intractability result for the *CAR*-semantics does not hold already for the instance checking problem. Indeed, we will show later in this chapter that instance checking is actually tractable under the *CAR*-semantics.

9.3 Consistent query answering in *DL-Lite*$_{A,id,den}$ under *IAR*-semantics

In this section, we focus on *DL-Lite*$_{A,id,den}$-KBs, and provide our technique for computing FOL-rewritings of boolean UCQs (BUCQs) under the *IAR*-semantics.

We remind the reader that BUCQs in *DL-Lite*$_{A,id,den}$ are FOL-rewritable (cf. Section 5.2), i.e., for every boolean union of conjunctive queries q and every *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$, there exists a FOL query q_r, over the alphabet of $\mathcal{T}$, such that for every non-empty ABox $\mathcal{A}$ it holds that $\langle \mathcal{T}, \mathcal{A} \rangle \models q$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models q_r$. The query q_r is called a *perfect FOL reformulation* of q w.r.t. $\mathcal{T}$. The algorithm for computing such a reformulation, called PerfectRef, is provided in [31]. In a nutshell, PerfectRef takes as input a BUCQ q and a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$ and compiles in q the knowledge of $\mathcal{T}$ useful for answering q, returning another UCQs over $\mathcal{T}$ which is a perfect FOL reformulation of q w.r.t. $\mathcal{T}$.

The notion of FOL-rewritability under *IAR*-semantics is essentially the same used under FOL-semantics. More precisely, we say that BUCQs in *DL-Lite*$_{A,id,den}$ are FOL-rewritable under the *IAR*-semantics if, for each BUCQs q and each *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$, there exists a FOL-query q_r such that, for any ABox $\mathcal{A}$, $\langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} q$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models q_r$. We call such q_r the *IAR-perfect reformulation* of q w.r.t. $\mathcal{T}$.

First, we recall that, given a DL TBox $\mathcal{T}$ and a set of V of ABox assertions, we say that V is a minimal $\mathcal{T}$-inconsistent set if the KB $\langle \mathcal{T}, V \rangle$ is inconsistent, and if for each proper subset V' of V, the KB $\langle \mathcal{T}, V' \rangle$ is consistent (cf. Definition 34). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL KB. We denote with $\mathit{minIncSets}(\mathcal{K})$ the set of minimal $\mathcal{T}$-inconsistent sets contained in $\mathcal{A}$. Notice that $\mathcal{K}$ is consistent if and only if $\mathit{minIncSets}(\mathcal{K}) = \emptyset$.

Since a TBox expressed in *DL-Lite*$_{A,id,den}$ cannot be inconsistent, Theorem 24 and Corollary 3 hold for every KB expressed in *DL-Lite*$_{A,id,den}$. It

follows that, given an inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ expressed in *DL-Lite*$_{A,id,den}$ and an ABox assertion α belonging to $\mathcal{A}$, there exists an A-repair of $\mathcal{K}$ that does not contains α if and only if there exists a $V \in minIncSets(\mathcal{K})$ containing α. Moreover, given a boolean conjunctive query q, $\mathcal{K} \models_{IAR} q$ if and only if there is a subset $\mathcal{A}'$ of $\mathcal{A}$ such that:

(i) $\mathcal{A}'$ is $\mathcal{T}$-consistent;

(ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models q$;

(iii) $\mathcal{A}' \cap V = \emptyset$ for every $V \in minIncSets(\mathcal{K})$.

To come up with our reformulation method, we exploit Corollary 3. Roughly speaking, in the reformulation of a BUCQ q over a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$, we encode into a FOL-formula all violations that can involve assertions belonging to images of q in any ABox $\mathcal{A}$. Indeed, this can be done by reasoning only on the TBox, and considering each query atom separately. Intuitively, we deal with inconsistency by rewriting each atom α of q into a FOL-formula α_r in such a way that $\langle \emptyset, \mathcal{A} \rangle \models \alpha_r$ only if $\langle \emptyset, \mathcal{A} \rangle \models \alpha$ and any images of α_r belongs to minimal $\mathcal{T}$-inconsistent set.

This inconsistency-driven rewriting is then suitably cast into the final reformulation, which takes into account also positive knowledge of the TBox, i.e., the inclusion assertions in $\mathcal{T}_{inc}$. We will show later in this section, that this can be done by means of the slight variation of the algorithm PerfectRef introduced in Section 5.1.

In Section 5.1 we showed that in a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ inconsistency may arise only if there exists in $\mathcal{A}$ a set of ABox assertions V such that the consistent KB $\langle \mathcal{T}_{inc}, V \rangle$ entails the negation of at least one assertions in $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id} \cup \mathcal{T}_{den}$. Moreover, we showed that such a check can be reduced to checking if at least one of the queries in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ is entailed by the KB $\langle \emptyset, \mathcal{A} \rangle$, where $\mathcal{Q}_{\mathcal{T}}^{unsat}$ is a set of suitable boolean queries built from the assertions in $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id} \cup \mathcal{T}_{den}$ and by reasoning on the assertions in $\mathcal{T}_{inc}$. Every query q in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ corresponds to FOL-sentences of the form (5.1), that is

$$\exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} A_i(t_i^1) \wedge \bigwedge_{i=1}^{m} T_i(t_i^2) \wedge \bigwedge_{i=1}^{\ell} P_i(t_i^3, t_i^4) \wedge \bigwedge_{i=1}^{j} U_i(t_i^4, t_i^5) \wedge \bigwedge_{i=1}^{h} t_i^6 \neq t_i^7$$

In order to compute the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$, we provided the algorithm unsatQueries($\mathcal{T}$). Then, in Section 5.3, we showed that, by evaluating the "non-boolean" version $q(\vec{x})$ of $q()$, for each boolean query $q() \in \mathcal{Q}_{\mathcal{T}}^{unsat}$, over the ABox $\mathcal{A}$ considered as a relational database, we can obtain a set of tuples that we can use to build some $\mathcal{T}$-inconsistent sets in $\mathcal{A}$ that we call $\mathcal{K}$-*clashes*. Moreover, we

showed that from such $\mathcal{T}$-inconsistent sets it is possible to build every other $\mathcal{T}$-inconsistent set in $\mathcal{A}$ (cf. Lemma 14).

The next example aims to clarify the relation between $\langle\mathcal{T},\mathcal{A}\rangle$-*clashes* and minimal $\mathcal{T}$-inconsistent sets in $\mathcal{A}$.

Example 32. Let $\mathcal{K} = \langle\mathcal{T},\mathcal{A}\rangle$ be the inconsistent *DL-Lite*$_{A,id,den}$-KB of Example 5. In Example 7, it is shown that the following $\mathcal{K}$-*clashes* appear in $\mathcal{A}$.

$$\begin{array}{rcl} \mathcal{K}\text{-}clash_1 & = & \{PortIn(p_1), PortOut(p_1)\}; \\ \mathcal{K}\text{-}clash_2 & = & \{PortIn(p_1), PortOut(p_1), connectedTo(p_1,p_2)\}. \end{array}$$

Observe that, for each ABox assertions $\alpha \in \mathcal{K}\text{-}clash_1$, we have that:

$$Mod(\langle\mathcal{T}, \mathcal{K}\text{-}clash_1 \setminus \{\alpha\}\rangle) \neq \emptyset$$

therefore, $\mathcal{K}\text{-}clash_1$ is a minimal $\mathcal{T}$-inconsistent set. Conversely, it is immediate to see that $\mathcal{K}\text{-}clash_2$ is not a minimal $\mathcal{T}$-inconsistent set. Indeed,

$$Mod(\langle\mathcal{T}, \mathcal{K}\text{-}clash_1 \setminus \{connectedTo(p_1,p_2)\}\rangle) = \emptyset$$

. □

In order to exploit Corollary 3 towards the definition of a FOL-rewriting procedure, we need however to identify those assertions in $\mathcal{A}$ that participate to a *minimal* $\mathcal{T}$-inconsistent set. Example 32 shows that we cannot use the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ to this end.

The theorem below follows from the definition of minimal $\mathcal{T}$-inconsistent set and from Theorem 6.

Theorem 35. *Let $\mathcal{K} = \langle\mathcal{T},\mathcal{A}\rangle$ be an inconsistent DL-Lite$_{A,id,den}$-KB, and let V be a $\mathcal{K}$-clash. V is a minimal $\mathcal{T}$-inconsistent set in $\mathcal{A}$ if and only if for every proper subset V' of V, and every query $q \in \mathcal{Q}_{\mathcal{T}}^{unsat}$, we have that $\langle\emptyset, V'\rangle \not\models q$.*

Proof.
($\Rightarrow$) Let V be a minimal $\mathcal{T}$-inconsistent set in $\mathcal{A}$. We prove that for every $V' \subset V$ and for every $q \in \mathcal{Q}_{\mathcal{T}}^{unsat}$, $\langle\emptyset, V'\rangle \not\models q$. Suppose, by way of contradiction, that there exists a query $q \in \mathcal{Q}_{\mathcal{T}}^{unsat}$ and a proper subset V' of V, such that $\langle\emptyset, V'\rangle \models q$. From Theorem 6 it follows that V' is a $\mathcal{T}$-inconsistent set, but this contradicts that V is a minimal $\mathcal{T}$-inconsistent set.
($\Leftarrow$) Let V be a $\mathcal{K}$-*clash* such that for every $V' \subset V$ and for every $q \in \mathcal{Q}_{\mathcal{T}}^{unsat}$, $\langle\emptyset, V'\rangle \not\models q$. We show that V is a minimal $\mathcal{T}$-inconsistent set in $\mathcal{A}$. Toward a contradiction, suppose that V is not a minimal $\mathcal{T}$-inconsistent set. This means that there is a proper subset V'' of V that is a $\mathcal{T}$-inconsistent set. From Theorem 6 it follows that there exists in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ a query q such that $\langle\emptyset, V''\rangle \models q$. Hence, we have a contradiction. ∎

On the base of above results, we present the algorithm minUnsatQueries($\mathcal{T}$) which, starting from the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ computed by unsatQueries($\mathcal{T}$), computes a new set $\mathcal{Q}_{\mathcal{T}}^{min}$ of boolean queries, which enjoys the following properties:

(i) For each boolean query $q \in \mathcal{Q}_{\mathcal{T}}^{min}$, $\langle \emptyset, \mathcal{A} \rangle \models q$ if and only if there exists in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ a query q' such that $\langle \emptyset, \mathcal{A} \rangle \models q'$. This guarantees that Theorem 6 also holds with minUnsatQueries($\mathcal{T}$) in place of unsatQueries($\mathcal{T}$).

(ii) For each boolean query $q \in \mathcal{Q}_{\mathcal{T}}^{min}$, if $\langle \emptyset, \mathcal{A} \rangle \models q$, then for every set of ABox assertions $V \in \mathsf{images}(q, \mathcal{A})$, $\langle \emptyset, V' \rangle \not\models q'$, where $V' \subset V$ and $q' \in \mathcal{Q}_{min}$. This guarantees that if a query $q \in \mathcal{Q}_{\mathcal{T}}^{min}$ is such that $\langle \emptyset, \mathcal{A} \rangle \models q$, then every image of q in $\mathcal{A}$ is a minimal $\mathcal{T}$-inconsistent set.

Before presenting the algorithm minUnsatQueries, we need to introduce some preliminary notions. Given a boolean query q, we say that a term t occurs in an *object position* of q if q contains an atom of the form $A(t)$, $P(t, t')$, $P(t', t)$, $U(t, t')$, whereas we say that t occurs in a *value position* of q if q contains an atom of the form $T(t)$, $U(t', t)$, where A, P, U, and T have the usual meaning. Given two terms t_1 and t_2 occurring in a query q, we say that t_1 and t_2 are *compatible* in q if either both t_1 and t_2 appear only in object positions of q or both t_1 and t_2 appear only in value positions of q.

Now, let q be a boolean query, and σ be a substitution function from the variables in q to constants of Σ_C. Moreover, let t_1 and t_2 be two different and compatible terms in q. Two possible cases are conceivable:

(i) σ substitutes t_1 and t_2 respectively with the two different constants c_1 and c_2 of Σ_C, or

(ii) σ substitutes both t_1 and t_2 with the same constant c of Σ_C.

It follows that, for every ABox $\mathcal{A}$, no answer is lost by evaluating over the interpretation $DB(\mathcal{A})$ the union of queries $(q \wedge t_1 = t_2) \vee (q \wedge t_1 \neq t_2)$, instead of q. The same holds if we first force to be equal, and then to be difference, more than two compatible terms in q. We call *inequalities saturation* of a query q the set containing the queries obtained from q by applying such a transformation in every possible way.

Example 33. Consider the following boolean conjunctive query

$$q \quad = \quad \exists x, y, z.C(x) \wedge P(x, y) \wedge P(y, z) \wedge C(z).$$

The *inequalities saturation* of q is the set $\mathcal{Q}_{str}$ containing the following queries.

$$\begin{array}{lcl} q_1 & = & \exists x, y, z.C(x) \wedge P(x, y) \wedge P(y, z) \wedge C(z) \wedge x \neq y \wedge x \neq z \wedge y \neq z; \\ q_2 & = & \exists x, y.C(x) \wedge P(x, y) \wedge P(y, x) \wedge C(x) \wedge x \neq y; \\ q_3 & = & \exists x, z.C(x) \wedge P(x, x) \wedge P(x, z) \wedge C(z) \wedge x \neq z; \\ q_4 & = & \exists x, y.C(x) \wedge P(x, y) \wedge P(y, y) \wedge C(y) \wedge x \neq y; \\ q_5 & = & \exists x.C(x) \wedge P(x, x). \end{array}$$

Let $\mathcal{A}$ be the following set of ABox assertions:

$$\{ \quad C(a), \quad C(b) \quad P(a,a), \quad B(a) \quad \}.$$

We note, that $\langle \emptyset, \mathcal{A} \rangle \models q$ and $\langle \emptyset, \mathcal{A} \rangle \models q_5$. □

We present the algorithm Saturate($\mathcal{Q}$) (Algorithm 17) that takes in input a set of queries $\mathcal{Q}$ and returns the set of queries $\mathcal{Q}''$ obtained by computing, for every query $q \in \mathcal{Q}$, the *inequality saturation* of q. Starting from each query $q \in \mathcal{Q}$, Saturate($\mathcal{Q}$) first computes the set $\mathcal{Q}'$ by unifying compatible terms in q in all possible ways; then, for any query q' in $\mathcal{Q}'$ and for each pair of terms t_1 and t_2 occurring in q' that are syntactically different, it adds the inequality atom $t_1 \neq t_2$ to q' and add such a new query to $\mathcal{Q}''$. In the algorithm $q[t_1/t_2]$ denotes the query obtained by replacing in q every occurrence of the term t_1 with the term t_2.

Input: a set of queries $\mathcal{Q}$
Output: a set of queries $\mathcal{Q}''$
begin
  $\mathcal{Q}' \leftarrow \emptyset$;
  while $\mathcal{Q} \neq \mathcal{Q}'$ **do**
    $\mathcal{Q}' \leftarrow \mathcal{Q}$;
    foreach $q \in \mathcal{Q}$ **do**
      foreach pair of different terms t_1 and t_2 in q **do**
        if $t_1 \neq t_2$ does not occur in q **and**
          t_1 and t_2 are compatible in q
        then $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{q[t_1/t_2]\}$;
  $\mathcal{Q}'' \leftarrow \emptyset$;
  foreach $q \in \mathcal{Q}'$ **do**
    foreach pair of different terms t_1 and t_2 in q **do**
      $q \leftarrow q \wedge (t_1 \neq t_2)$;
    $\mathcal{Q}'' \leftarrow \mathcal{Q}'' \cup \{q\}$;
  return $\mathcal{Q}''$;
end

Algorithm 17: Saturate($\mathcal{Q}$)

The following lemma shows that the algorithm Saturate terminates, when applied to a finite set of queries.

Lemma 41. *Let $\mathcal{Q}$ be a finite set of queries of form (5.1). Then, Saturate($\mathcal{Q}$) terminates and runs in exponential time with respect to the size of $\mathcal{Q}$.*

Proof. The termination of Saturate, for each finite set of queries $\mathcal{Q}$ given as input, immediately follows from the following facts:

1. The set $\mathcal{Q}$ contains a finite number of queries q. Let $|\mathcal{Q}'| = n$.
2. For each query $q \in \mathcal{Q}$, the number of atoms and terms in q is finite, and the algorithm does not generate new terms. Suppose that every query in $\mathcal{Q}$ has a number of different terms that is less than or equal to m.
3. At the beginning of the first iteration of the **while loop** the set $\mathcal{Q}'$ contains $|\mathcal{Q}| = n$ queries. At the end of the first iteration it contains a number of queries that is less than or equal to
$$|\mathcal{Q}'|_1 = \left(\frac{m(m-1)}{2}\right) n + n.$$
At the end of the second iteration such a number rises to
$$|\mathcal{Q}'|_2 = \left(\frac{(m-1)(m-2)}{2}\right)\left(\frac{m(m-1)}{2}\right) n + \left(\frac{m(m-1)}{2}\right) n + n$$
and it keep arising in this manner for the further iterations. Therefore, the number of iteration of the **while loop** is less than or equal to m. At the end of the **while loop** the set $\mathcal{Q}'$ contains a number of queries that is not greater than $\left(\frac{m^{2m}}{2^m}\right) n$.
4. Finally, for each query $q \in \mathcal{Q}'$, the algorithm adds the inequality atom $t_1 \neq t_2$ to q, where t_1 and t_2 are two syntactically different terms occurring in q and adds the so generated query to the set $\mathcal{Q}''$. The number of queries in $\mathcal{Q}''$ is equal to $|\mathcal{Q}'|$. Since both $|\mathcal{Q}'|$ and the number of terms in each query in $\mathcal{Q}'$ are finite, the number of iterations of this step is finite.

From the observation above, we can conclude that Saturate($\mathcal{Q}$) terminates and computes the set $\mathcal{Q}''$ with a number of executions that is exponential with respect to the maximum number of different terms occurring in a query belonging to $\mathcal{Q}$. ∎

The next theorem shows that the process applied by Saturate($\mathcal{Q}$) to the union of queries $\mathcal{Q}$ does not affect the result of the evaluation of $\mathcal{Q}$.

Theorem 36. *Let $\mathcal{A}$ be an ABox and let $\mathcal{Q}$ be a set of boolean queries. Then, $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models$ Saturate($\mathcal{Q}$).*

Proof.
($\Rightarrow$) Suppose that there is a query $q = \psi(\vec{x}) \in \mathcal{Q}$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$. We show that there exists a query q' in Saturate($\mathcal{Q}$) such that $\langle \emptyset, \mathcal{A} \rangle \models q'$.

Since $\langle \emptyset, \mathcal{A} \rangle \models q$, there is a substitution σ from the variables in q to constants in $\mathcal{A}$ such that the formula $\sigma(q)$ evaluates to *true* in the interpretation $DB(\mathcal{A})$.

Let x and y be two compatible variables in q and let c_1 and c_2 be two different constants in $\mathcal{A}$. Suppose that x and y are the only variables in q and that c_1 and c_2 are the only constants in $\mathcal{A}$. The following cases are conceivable:

1. the function σ substitutes the variable x (resp. y) with the constant c_1 and the variable y (resp. x) with the constant c_2;

2. the function σ substitutes both the variables x and y with either c_1 or c_2.

Intuitively, we have that in the first case σ *imposes* to x and y to be different, instead in the second case it *imposes* to x and y to be equal. Since $DB(\mathcal{A}) \models q$ then the formula obtained by either applying σ in the first case or applying σ in the second case evaluates to *true* over $DB(\mathcal{A})$. But this means that one of the following queries is entailed by $\langle \emptyset, \mathcal{A} \rangle$:

$$q_1 = \psi(x, y) \wedge x \neq y;$$
$$q_2 = \psi(x, y) \wedge x = y.$$

Indeed, we have that for the query q_1 a substitution function can only substitutes the variable x (resp. y) with the constant c_1 and the variable y (resp. x) with the constant y, while for the query q_2 a substitution function can only substitutes both the variables x and y with either c_1 or c_2.

As it is possible to see by observing Algorithm 17, if we apply Saturate to the query $q = \psi(x, y)$, it first computes the set $\mathcal{Q}'$ by:

- adding to $\mathcal{Q}'$ the query $q = \psi(x, y)$, and by
- adding to $\mathcal{Q}'$ the query $q_u = \psi(x)$ obtained by unifying the variables x and y in q;

then for each query q' in $\mathcal{Q}'$ and for each pair of terms t_1 and t_2 occurring in q' that are syntactically different, it adds the inequality atom $t_1 \neq t_2$ to q'. Hence it computes the following queries

$$q^{\neq} = \psi(x, y) \wedge x \neq y;$$
$$q_u = \psi(x);$$

which are logically equivalent to the queries q_1 and q_2 above. Since this technique can be generalized to deal with arbitrary boolean queries and arbitrary ABoxes, we have proved the claim.

($\Leftarrow$) Let q' be a query in $\mathsf{Saturate}(\mathcal{Q})$ such that $\langle \emptyset, \mathcal{A} \rangle \models q'$. We show now that there is a query q in $\mathcal{Q}$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$. Suppose, by way of contradiction, that for every query $q \in \mathcal{Q}$ we have that $\langle \emptyset, \mathcal{A} \rangle \not\models q$. This means that there does

not exist a substitution σ from the variables in q to constants in $\mathcal{A}$ such that the formula $\sigma(q)$ evaluates to *true* in $DB(\mathcal{A})$. Hence, for each query q_u obtained from q by unifying different terms q we have that $\langle \emptyset, \mathcal{A} \rangle \not\models q_u$, and the same holds if we impose to different terms in such queries to be different. But this contradicts that there exists in $\mathsf{Saturate}(\mathcal{Q})$ a query q' such that $\langle \emptyset, \mathcal{A} \rangle \models q'$. ∎

A notable consequence of Theorem 36 is that, by applying the algorithm Saturate to the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ computed by the algorithm $\mathsf{unsatQueries}(\mathcal{T})$, all the properties of $\mathcal{Q}_{\mathcal{T}}^{unsat}$ still hold. In particular, we have the following.

Lemma 42. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB. A $\mathcal{T}$-inconsistent set $V \subseteq \mathcal{A}$ exists if and only if $\langle \emptyset, \mathcal{A} \rangle \models \mathsf{Saturate}(\mathsf{unsatQueries}(\mathcal{T}))$.*

Proof. The proof directly follows from Theorem 6 and Theorem 36. ∎

In what follows, we clarify the role that the algorithm Saturate plays in computing the set of queries $\mathcal{Q}_{\mathcal{T}}^{min}$, and why it is necessary for our purposes. For reasons of simplicity, we ignore, for the moment, inconsistencies caused by erroneous assignments of values to attributes. Therefore, we start considering only KBs without inclusions between value-domains, and then with $\mathcal{T}_{type} = \emptyset$. Firstly, we need to introduce the notion of *proper syntactical subset* of a query. Let q and q' be two boolean queries. We say that q is a *proper syntactical subset* of q', written $q \prec_{Rn} q'$ if there exists a renaming function $Rn(q, q')$ of the variables in q to the variables in q', such that every atom $S(\vec{t})$ occurring in $Rn(q, q')$ occurs also in q' and an analogous renaming from q' to q does not exist. The next example briefly illustrates this syntactical form of containment between two queries.

Example 34. Consider the following set of queries which is the set corresponding to the inequalities saturation of the query $q = \exists x, y, z.C(x) \wedge P(x, y) \wedge P(y, z) \wedge C(z)$ in Examples 33.

$$
\begin{array}{lcl}
q_1 & = & \exists x, y, z.C(x) \wedge P(x, y) \wedge P(y, z) \wedge C(z) \wedge x \neq y \wedge x \neq z \wedge y \neq z; \\
q_2 & = & \exists x, y.C(x) \wedge P(x, y) \wedge P(y, x) \wedge C(x) \wedge x \neq y; \\
q_3 & = & \exists x, z.C(x) \wedge P(x, x) \wedge P(x, z) \wedge C(z) \wedge x \neq z; \\
q_4 & = & \exists x, y.C(x) \wedge P(x, y) \wedge P(y, y) \wedge C(y) \wedge x \neq y; \\
q_5 & = & \exists x.C(x) \wedge P(x, x).
\end{array}
$$

Given the query q and q', we have that $q \prec_{Rn} q'$ if, by applying a suitable renaming function $Rn(q, q')$ of the variable in q to the variable in q', we have that every atom $S(\vec{t})$ occurring in $Rn(q, q')$ occurs also in q' and there does not exist a renaming function $Rn'(q', q)$ such that every atom $S'(\vec{t})$ occurring in $Rn'(q, q')$ occurs also in q. By applying this definition to the set of queries above, we have:

- $q_5 \prec_{Rn} q_3$;
- $q_5 \prec_{Rn} q_4$.

Which means that the query q_5 is a *proper syntactical subset* of both the query q_3 and the query q_4. □

We aim to compute the set of queries $\mathcal{Q}_{\mathcal{T}}^{min}$ that can be used to check satisfiability of a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and that is characterized by the fact that for each query $q \in \mathcal{Q}_{\mathcal{T}}^{min}$, and for each image V of q, V is a minimal $\mathcal{T}$-inconsistent set. Consider the following example.

Example 35. Let $\mathcal{T}$ be the *DL-Lite*$_{A,id,den}$ TBox presented in Example 2. We focus on the following denial assertion belonging to $\mathcal{T}$.

$$\forall x, y, z.(Port(x) \wedge Port(y) \wedge of(x, z) \wedge of(y, z) \wedge connectedTo(x, y) \rightarrow \bot).$$

Suppose to have the following ABox:

$$\begin{array}{rcll} \mathcal{A} & = & \{ & Port(p_1),\ Device(d_1),\ of(p_1, d_1),\ connectedTo(p_1, p_1), \\ & & & Port(p_2),\ of(p_2, d_1),\ Port(p_3),\ of(p_3, d_1),\ connectedTo(p_2, p_3) \quad \} \end{array}$$

It is not difficult to verify that the *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is not consistent. Indeed, the set of queries $\mathcal{Q}_{\mathcal{T}}^{unsat}$ contains, among the others, the following boolean queries:

$$\begin{array}{rcl} q_1 & = & \exists x, y, z.Port(x) \wedge Port(y) \wedge of(x, z) \wedge of(y, z) \wedge connectedTo(x, y); \\ q_2 & = & \exists x, y, z.of(x, z) \wedge of(y, z) \wedge connectedTo(x, y); \\ q_3 & = & \exists x.connectedTo(x, x). \end{array}$$

By considering the interpretation $DB(\mathcal{A})$, and the non-boolean version of the queries above, we obtain the following $\mathcal{K}$-*clashes*:

$$\begin{array}{rcll} V_1 & = & \{ & Port(p_2),\ of(p_2, d_1),\ Port(p_3),\ of(p_3, d_1),\ connectedTo(p_2, p_3) \quad \}; \\ V_2 & = & \{ & Port(p_1),\ of(p_1, d_1),\ Port(p_1),\ of(p_1, d_1),\ connectedTo(p_1, p_1) \quad \}; \\ V_3 & = & \{ & of(p_2, d_1),\ of(p_3, d_1),\ connectedTo(p_2, p_3) \quad \}; \\ V_4 & = & \{ & of(p_1, d_1),\ of(p_1, d_1),\ connectedTo(p_1, p_1) \quad \}; \\ V_5 & = & \{ & connectedTo(p_1, p_1) \quad \}. \end{array}$$

It is easy to see that only the sets V_3 and V_5 are minimal $\mathcal{T}$-inconsistent sets. Indeed, we have the following AR-repairs of $\mathcal{K}$.

$$\begin{array}{rcll} \mathcal{A}_{rep1} & = & \{ & Port(p_1),\ Port(p_2),\ Port(p_3),\ Device(d_1),\ of(p_1, d_1), \\ & & & of(p_3, d_1),\ connectedTo(p_2, p_3) \quad \}; \\ \mathcal{A}_{rep2} & = & \{ & Port(p_1),\ Port(p_2),\ Port(p_3),\ Device(d_1),\ of(p_1, d_1), \\ & & & of(p_2, d_1),\ connectedTo(p_2, p_3) \quad \}; \\ \mathcal{A}_{rep3} & = & \{ & Port(p_1), Port(p_2),\ Port(p_3),\ Device(d_1),\ of(p_1, d_1), \\ & & & of(p_2, d_1),\ of(p_3, d_1), \quad \}. \end{array}$$

Now, let us consider the following queries:

$$q_a = Port(p_1) \wedge of(p_1, d_1);$$
$$q_b = \exists x.of(p_2, x) \wedge of(p_3, x).$$

According to Definition 30 we have that $\mathcal{K} \models_{IAR} q_a$ and $\mathcal{K} \not\models_{IAR} q_b$. □

How can we modify the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ of Example 35 in order to compute the set $\mathcal{Q}_{\mathcal{T}}^{min}$?

Since the TBox $\mathcal{T}$ contains the assertion $\exists of \sqsubseteq Port$, one can immediately observe that, by removing the query q_1 from $\mathcal{Q}_{\mathcal{T}}^{unsat}$, we obtain a new set of queries $\mathcal{Q}_{\mathcal{T}}^{2-unsat}$ that (i) can be used in place of $\mathcal{Q}_{\mathcal{T}}^{unsat}$ for checking satisfiability of $\mathcal{K}$ (cf. [31, 103]); and (ii) enjoys the appreciable property that both V_1 and V_2 do not coincide with the image in $\mathcal{A}$ of any query in $q \in \mathcal{Q}_{\mathcal{T}}^{2-unsat}$.

We observe that we cannot remove any other query from $\mathcal{Q}_{\mathcal{T}}^{2-unsat}$ without nullifying its capacity of replacing $\mathcal{Q}_{\mathcal{T}}^{unsat}$ for checking satisfiability of a KB $\langle \mathcal{T}, \mathcal{A}' \rangle$, for every ABox $\mathcal{A}'$.

So, how can we do better?

The following example illustrates our solution based on the notions of *inequalities saturation* and *proper syntactical subset* of a query q

Example 36. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be the *DL-Lite*$_{A,id,den}$-KB of Example 35. As shown, the following queries belong to $\mathcal{Q}_{\mathcal{T}}^{unsat}$.

$$q_1 = \exists x, y, z.Port(x) \wedge Port(y) \wedge of(x, z) \wedge of(y, z) \wedge connectedTo(x, y);$$
$$q_2 = \exists x, y, z.of(x, z) \wedge of(y, z) \wedge connectedTo(x, y);$$
$$q_3 = \exists x.connectedTo(x, x).$$

We compute the set $\mathcal{Q}_{str}$ containing all the queries obtained by computing the inequality saturation of each query $q \in \mathcal{Q}_{\mathcal{T}}^{unsat}$. As we show earlier, we can do this by means of the algorithm Saturate. By focusing only on the three queries above, we obtain that $\mathcal{Q}_{str}$ contains the following boolean queries.

$$q_1^1 = \exists x, y, z.Port(x) \wedge Port(y) \wedge of(x, z) \wedge of(y, z) \wedge connectedTo(x, y) \wedge x \neq y \wedge x \neq z \wedge y \neq z;$$
$$q_1^2 = \exists x, y.Port(x) \wedge Port(y) \wedge of(x, y) \wedge of(y, y) \wedge connectedTo(x, y) \wedge x \neq y;$$
$$q_1^3 = \exists x, y.Port(x) \wedge Port(y) \wedge of(x, x) \wedge of(y, x) \wedge connectedTo(x, y) \wedge x \neq y;$$
$$q_1^4 = \exists x, z.Port(x) \wedge of(x, z) \wedge connectedTo(x, x) \wedge x \neq z;$$
$$q_1^5 = \exists x.Port(x) \wedge of(x, x) \wedge connectedTo(x, x);$$
$$q_2^1 = \exists x, y, z.of(x, z) \wedge of(y, z) \wedge connectedTo(x, y) \wedge x \neq y \wedge x \neq z \wedge$$

$$
\begin{array}{rcl}
 & & y \neq z;\\
q_2^2 & = & \exists x, y.of(x,y) \wedge of(y,y) \wedge connectedTo(x,y) \wedge x \neq y;\\
q_2^3 & = & \exists x, y.of(x,y) \wedge of(y,x) \wedge connectedTo(x,y) \wedge x \neq y;\\
q_2^4 & = & \exists x, z.of(x,z) \wedge connectedTo(x,x) \wedge x \neq z;\\
q_2^5 & = & \exists x.of(x,x) \wedge connectedTo(x,x)\\
q_3^1 & = & \exists x.connectedTo(x,x).
\end{array}
$$

Now, we apply the notion *proper syntactical subset* to the queries in $\mathcal{Q}_{str}$. It is easy to verify that the following hold:

$$
\begin{array}{lll}
q_2^1 \prec_{Rn} q_1^1; & q_2^2 \prec_{Rn} q_1^2; & q_2^3 \prec_{Rn} q_1^3;\\
q_2^4 \prec_{Rn} q_1^4; & q_2^5 \prec_{Rn} q_1^5; & q_3^1 \prec_{Rn} q_1^4;\\
q_3^1 \prec_{Rn} q_1^5; & q_3^1 \prec_{Rn} q_2^4; & q_3^1 \prec_{Rn} q_2^5;
\end{array}
$$

Let $\mathcal{Q}_{str}^{\prec_{Rn}}$ be the set computed by removing from $\mathcal{Q}_{str}$ every query q such that there exists in $\mathcal{Q}_{str}$ a query q' such that $q' \prec_{Rn} q$. Hence, we have:

$$
\begin{array}{rcl}
q_2^1 & = & \exists x, y, z.of(x,z) \wedge of(y,z) \wedge connectedTo(x,y) \wedge x \neq y \wedge x \neq z \wedge\\
 & & y \neq z;\\
q_2^2 & = & \exists x, y.of(x,y) \wedge of(y,y) \wedge connectedTo(x,y) \wedge x \neq y;\\
q_2^3 & = & \exists x, y.of(x,x) \wedge of(y,x) \wedge connectedTo(x,y) \wedge x \neq y;\\
q_3^1 & = & \exists x.connectedTo(x,x).
\end{array}
$$

The only queries in $\mathcal{Q}_{str}^{\prec_{Rn}}$ that evaluate to *true* over $DB(\mathcal{A})$ are q_2^1 and q_3^1. Moreover, theirs images in $\mathcal{A}$ are:

$$
\begin{array}{rcl}
V_2^1 & = & \{\ of(p_2,d_1),\ of(p_3,d_1),\ connectedTo(p_2,p_3)\ \};\\
V_3^1 & = & \{\ connectedTo(p_1,p_1)\ \};
\end{array}
$$

that coincide respectively with the two minimal $\mathcal{T}$-inconsistent sets V_3 and V_5 of Example 35. □

Given a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$, the procedure used in the example above can be summarized as follows:

1) compute the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ by means of the algorithm unsatQueries($\mathcal{T}$);
2) compute the set $\mathcal{Q}_{str}$ by means of the algorithm Saturate($\mathcal{Q}_{\mathcal{T}}^{unsat}$);
3) compute the set $\mathcal{Q}_{str}^{\prec_{Rn}}$ as follows:
   3.1) $\mathcal{Q}_{str}^{\prec_{Rn}} \leftarrow \mathcal{Q}_{str}$;
   3.2) for each pair of query q and q' in $\mathcal{Q}_{str}$, if $q \prec_{Rn} q'$, then remove q' from $\mathcal{Q}_{str}^{\prec_{Rn}}$.

We provide the following lemma which shows that, if $\mathcal{T}$ is a TBox expressed in *DL-Lite*$_{A,id,den}$, then, for every ABox $\mathcal{A}$, we can use the set $\mathcal{Q}_{str}^{\prec_{Rn}}$, computed as shown above, for checking the satisfiability of the KB $\langle \mathcal{T}, \mathcal{A} \rangle$.

Lemma 43. *Let $\mathcal{T}$ be a DL-Lite$_{A,id,den}$ TBox, and let $\mathcal{A}$ be an ABox. Then, $Mod(\langle \mathcal{T}, \mathcal{A} \rangle) = \emptyset$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{str}^{\prec_{Rn}}$.*

Proof.
($\Rightarrow$) We show that, if $\mathcal{T}$ is a *DL-Lite*$_{A,id,den}$ TBox and $\mathcal{A}$ is an ABox such that $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{\mathcal{T}}^{unsat}$, then $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{str}^{\prec_{Rn}}$. We proceed by contradiction. Suppose that $\langle \emptyset, \mathcal{A} \rangle \not\models \mathcal{Q}_{str}^{\prec_{Rn}}$. This means that for every query $q^{\prec}$ in $\mathcal{Q}_{str}^{\prec_{Rn}}$, $\langle \emptyset, \mathcal{A} \rangle \not\models q^{\prec}$. Since $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{\mathcal{T}}^{unsat}$, then Lemma 42 states that $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{str}$, where $\mathcal{Q}_{str} = \mathsf{Saturate}(\mathcal{Q}_{\mathcal{T}}^{unsat})$. Hence there is a query $q \in \mathcal{Q}_{str}$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$. Since $\langle \emptyset, \mathcal{A} \rangle \not\models \mathcal{Q}_{str}^{\prec_{Rn}}$, then $q \notin \mathcal{Q}_{str}^{\prec_{Rn}}$. This means that there exists in $\mathcal{Q}_{str}^{\prec_{Rn}}$ a query q' such that $q' \prec_{Rn} q$. Hence, there exists a renaming function $Rn(q', q)$ of the variables in q' to the variables in q, such that every atom $S(\vec{t})$ occurring in $Rn(q', q)$ occurs also in q and an analogous renaming from q to q' does not exist. Since, $\langle \emptyset, \mathcal{A} \rangle \models q$ then there exists a substitution σ from the variables in q to constants in $\mathcal{A}$ such that the formula $\sigma(q)$ evaluates to *true* in the interpretation $DB(\mathcal{A})$. But this means that also $\sigma(Rn(q', q))$ evaluates to *true* in $DB(\mathcal{A})$, then we have a contradiction.
($\Leftarrow$) We show that if $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{str}^{\prec_{Rn}}$, then $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{\mathcal{T}}^{unsat}$. If $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{str}^{\prec_{Rn}}$, then there is a query $q \in \mathcal{Q}_{str}^{\prec_{Rn}}$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$. Since $\mathcal{Q}_{str}^{\prec_{Rn}} \subseteq \mathcal{Q}_{str}$, then $q \in \mathcal{Q}_{str}$, and then $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{str}$. Finally, from Lemma 42, it follows that $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{\mathcal{T}}^{unsat}$. ∎

The next lemma guarantees that, given a *DL-Lite*$_{A,id}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with $\mathcal{T}_{type} = \emptyset$ and a query q in $\mathcal{Q}_{str}^{\prec_{Rn}}$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$, every image V of q in $\mathcal{A}$ is a minimal $\mathcal{T}$-inconsistent set.

Lemma 44. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{A,id,den}$-KB with $\mathcal{T}_{type} = \emptyset$, and let q be a query in $\mathcal{Q}_{str}^{\prec_{Rn}}$. Then, for every $V' \subset V$, where $V \in \mathsf{images}(q, \mathcal{A})$, and for every $q' \in \mathcal{Q}_{str}^{\prec_{Rn}}$, $\langle \emptyset, V' \rangle \not\models q'$.*

Proof. Since $\mathcal{T}_{type} = \emptyset$, then every query in q belonging to $\mathcal{Q}_{\mathcal{T}}^{unsat}$ is of the form $\exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} A_i(t_i^1) \wedge \bigwedge_{i=1}^{m} P_i(t_i^2, t_i^3) \wedge \bigwedge_{i=1}^{\ell} U_i(t_i^4, t_i^5) \wedge \bigwedge_{i=1}^{h} t_i^6 \neq t_i^7$, where every A_i, P_i, and U_i are respectively an atomic concept name, an atomic role name, and an attribute name appearing in $\mathcal{T}$, every t_i^e is a term (i.e., either a constant or a variable), and $z_1, \ldots, z_k$ are all the variables appearing in q. In what follows, given a query q, we denote with $atoms(q)$, the set of atoms occurring in q.

If $\mathcal{K}$ is consistent, then Lemma 43 guarantees that for every query $q \in \mathcal{Q}_{str}^{\prec_{Rn}}$, $\langle \emptyset, \mathcal{A} \rangle \not\models q$. Hence there is no minimal $\mathcal{T}$-inconsistent set in $\mathcal{A}$. Let $\mathcal{K}$ be inconsistent. The proof proceeds by contradiction as follows. Let q be a

query in $\mathcal{Q}_{str}^{\prec_{Rn}}$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$, and let $V \in \mathsf{images}(q, \mathcal{A})$. Hence, there is a substitution σ from the variables in q to constants in $\mathcal{A}$ such that the formula $\sigma(q)$ evaluates to *true* in the interpretation $DB(V)$. Obviously, every constant occurring in q occurs also in V. Since $\mathcal{Q}_{str}^{\prec_{Rn}} \subseteq \mathcal{Q}_{str}$, then we have constrained $t_1 \neq t_2$ for each pair of terms t_1 and t_2 in q. Hence, for each pair of different variables x and y in q, σ substitutes the variable x with a constant c_1 in V and the variable y with the constant c_2 in V with $c_1 \neq c_2$; Let σ^{-1} be the inverse function of the function σ, i.e. we have that σ^{-1} is the function that, for each variable x in q and for each constant c in V, substitutes x to c iff σ substitutes x with c.

Now, suppose that there is a query $q' \in \mathcal{Q}_{str}^{\prec_{Rn}}$ such that $\langle \emptyset, V' \rangle \models q'$, where V' is a proper subset of V. Clearly, since σ^{-1} perform a renaming of the constant in V without unifying any constant, we have that $\langle \emptyset, \sigma^{-1}(V') \rangle \models q'$. But this means that there is a substitution σ' from the variables in q' to the terms in q such that each atom in $\sigma(q')$ is in $atom(q)$. Moreover, since $q' \in \mathcal{Q}_{str}^{\prec_{Rn}}$, from the observations above, we have that for each pair of variables x' and y' in q', σ' substitutes the variable x' with the term t_1 in q and the variable y' with the term t_2 in q with $t_1 \neq t_2$; and since $V' \subset V$, then $\sigma'(q') \subset atoms(q)$. Hence, σ' constitutes a renaming function of the variables in q' to the variables in q, such that every atom occurring in $\sigma(q')$ occurs also in q' and an analogous renaming from q' to q does not exist. This means that $q' \prec_{Rn} q$, which contradicts that $q \in \mathcal{Q}_{str}^{\prec_{Rn}}$. ∎

Up to here, we have shown how computing the set $\mathcal{Q}_{\mathcal{T}}^{min}$ in that cases in which the TBox $\mathcal{T}$ does not contains value-domain inclusion assertions, i.e., where $\mathcal{T}_{type} = \emptyset$. Now, we turn to the case where $\mathcal{T}_{type}$ may be non-empty.

In Section 5.1, it is shown that a *DL-Lite*$_{A,id,den}$-KB can be affected by inconsistencies caused by erroneous assignments of values to attributes, i.e., inconsistencies arising when a value of type T_i is assigned to an attribute of value-type T_j. This kind of inconsistency is properly detected by the algorithm $\mathsf{Satisfiable}_{DA}(\mathcal{K})$ by means of queries in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ of the form $\exists x, y.U(x, y) \wedge T_j(y)$, where U is an attribute name, and T_j is a value-domain. More precisely, if a TBox $\mathcal{T}$ entails that the range of an attribute U is of type T_i , i.e., $\mathcal{T} \models \rho(U) \sqsubseteq T_i$, in the set $cln(\mathcal{T})$ are added the queries $\exists x, y.U(x, y) \wedge T_j(y)$, for each value-domain T_j in $T_1, \ldots, T_n$ different to T_i. We recall that for every value v occurring in an ABox, we assume that there is a predefined value-domain T_i, such that $val(v) \in val(T_i)$.

The following example points out the issue related to this kind of inconsistency in computing $\mathcal{Q}_{\mathcal{T}}^{min}$.

Example 37. Let $\mathcal{T}$ be the TBox presented in Example 2. Specifically, here,

we are interested in the following TBox assertions:

$$\begin{array}{lcll} \mathcal{T}_{type} & = & \{ & \rho(number) \sqsubseteq \texttt{xsd:integer} \quad \} \\ \mathcal{T}_{id} & = & \{ & (id\ Port\ number,\ of\ ) \quad \} \end{array}$$

In words, the assertion in $\mathcal{T}_{type}$ states that the range of the attribute *number* is restricted to be an integer, while the identification assertion imposes that there cannot exist two different ports of the same device having the same number. The queries in $\mathcal{Q}_{\mathcal{T}}^{unsat}$ originated from the assertions above are given below. For sake of brevity, we assume that each constant v in Γ_V is interpreted as a specific value $val(v)$ belonging to $val(\texttt{xsd:integer}) \cup val(\texttt{xsd:string}) \cup val(\texttt{xsd:dateTime})$. In other words, the value type that we are considering here are only those corresponding respectively to the data types `xsd:integer`, `xsd:string`, and `xsd:dateTime`.

$$\begin{array}{lcl} q_1 & = & \exists x, y, d, n.Port(x) \wedge of(x, d) \wedge number(x, n) \wedge \\ & & \quad Port(y) \wedge of(y, d) \wedge number(y, n) \wedge x \neq y; \\ q_2 & = & \exists x, y, d, n.of(x, d) \wedge number(x, n) \wedge \\ & & \quad of(y, d) \wedge number(y, n) \wedge x \neq y; \\ q_3 & = & \exists x, y.number(x, y) \wedge \texttt{xsd:string}(y); \\ q_4 & = & \exists x, y.number(x, y) \wedge \texttt{xsd:dateTime}(y). \end{array}$$

Consider the following ABox:

$$\begin{array}{lcll} \mathcal{A} & = & \{ & Port(p_1), of(p_1, d_1), number(p_1, \text{'9XK11'}), \\ & & & Port(p_2), of(p_2, d_1), number(p_2, \text{'9XK11'}) \quad \} \end{array}$$

where '9XK11' is a string. Clearly, the *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent. Indeed, the following $\mathcal{K}$-*clashes* are pinpointed by $\mathsf{InconsistentSets}(\mathcal{K})$.

$$\begin{array}{lcll} V_1 & = & \{ & Port(p_1), of(p_1, d_1), number(p_1, \text{'9XK11'}), \\ & & & Port(p_2), of(p_2, d_1), number(p_2, \text{'9XK11'}) \quad \} \\ V_2 & = & \{ & of(p_1, d_1), number(p_1, \text{'9XK11'}), \\ & & & of(p_2, d_1), number(p_2, \text{'9XK11'}) \quad \} \\ V_3 & = & \{ & number(p_1, \text{'9XK11'}) \quad \} \\ V_4 & = & \{ & number(p_2, \text{'9XK11'}) \quad \} \end{array}$$

It is easy to see that only V_3 and V_4 are minimal $\mathcal{T}$-inconsistent sets. Indeed, both the assertions $number(p_1, \text{'9XK11'})$ and $number(p_2, \text{'9XK11'})$ violate the assertion $\rho(number) \sqsubseteq \neg\texttt{xsd:string} \in cln(\mathcal{T})$ originated by the assertion $\rho(number) \sqsubseteq \texttt{xsd:integer}$ in $\mathcal{T}$ and by expliciting the disjointness between the various predefined value-domains.

Now, we try to apply the technique adopted in Example 36, i.e. we first compute the set $\mathsf{Saturate}(\mathcal{Q}_{\mathcal{T}}^{unsat})$, and then we remove from the resulting set

every query q for which there exists in the set another query q' such that $q' \prec_{Rn} q$. We denote the resulting set with $\mathcal{Q}_{str}^{\prec_{Rn}}$. It is given below.

$$\begin{aligned}
q_2^1 &= \exists x, y, d, n.of(x, d) \wedge number(x, n) \wedge of(y, d) \wedge number(y, n) \wedge \\
&\quad x \neq y \wedge x \neq d \wedge x \neq n \wedge y \neq d \wedge y \neq n \wedge d \neq n; \\
q_2^2 &= \exists x, y, n.of(x, y) \wedge number(x, n) \wedge of(y, y) \wedge number(y, n) \wedge \\
&\quad x \neq y \wedge x \neq n \wedge y \neq n; \\
q_2^3 &= \exists x, y, n.of(x, x) \wedge number(x, n) \wedge of(y, x) \wedge number(y, n) \wedge \\
&\quad x \neq y \wedge x \neq n \wedge y \neq n; \\
q_3^1 &= \exists x, y.number(x, y) \wedge \texttt{xsd:string}(y); \\
q_4^1 &= \exists x, y.number(x, y) \wedge \texttt{xsd:dateTime}(y).
\end{aligned}$$

Observe that both q_2^1 and q_3^1 evaluate to *true* over the interpretation $DB(\mathcal{A})$, but only for each $V \in \mathsf{images}(q_3^1, \mathcal{A})$ we have that V is a minimal $\mathcal{T}$-inconsistent set. □

The example above shows that, if we are considering *DL-Lite*$_{A,id,den}$ TBoxes with $\mathcal{T}_{type} \neq \emptyset$, the procedure adopted in Example 36 may miss computing a set with the same feature of $\mathcal{Q}_{\mathcal{T}}^{min}$. Indeed, by referring to Example 37, the q_2^1 query must evaluate to *true* over $DB(\mathcal{A})$ only in that cases where the variable n in the query is substituted with a values v belonging to $val(\texttt{xsd:integer})$. A direct method to achieve this target is forcing in the query the variable n to be an integer. That is, modifying q_2^1 in the following manner:

$$\begin{aligned}
q_2^1 &= \exists x, y, d, n.of(x, d) \wedge number(x, n) \wedge of(y, d) \wedge number(y, n) \wedge \\
&\quad x \neq y \wedge x \neq d \wedge x \neq n \wedge y \neq d \wedge y \neq n \wedge d \neq n \wedge \\
&\quad \texttt{xsd:integer}(n);
\end{aligned}$$

Clearly, the above describe method, adopted for avoiding the error pointed out by Example 37, has to be extended to the general case. That is, given a *DL-Lite*$_{A,id,den}$ TBox, for each query q in $\mathsf{Saturate}(\mathcal{Q}_{\mathcal{T}}^{unsat})$, and for each atom $U(t_1, t_2)$ in q, where U is an attribute name, if $\mathcal{T} \models \rho(U) \sqsubseteq T_i$ and there exist no atoms $T_j(t_2)$ in q, then we have to substitute the atom $U(x, y)$ in q with the conjunction of atoms $U(t_1, t_2) \wedge T_i(y)$.

By exploiting the results given above, it easy to come up with a method for computing the set $\mathcal{Q}_{\mathcal{T}}^{min}$ from a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$. Thus, we are now ready to present the algorithm $\mathsf{minUnsatQueries}(\mathcal{T})$ for computing $\mathcal{Q}_{\mathcal{T}}^{min}$. The algorithm proceeds as follows.

Step 1 The algorithm initializes $\mathcal{Q}_{\mathcal{T}}^{unsat}$ to the set $\mathsf{unsatQueries}(\mathcal{T})$.

Step 2 The algorithm computes the set $\mathcal{Q}_{str}$ through the algorithm Saturate. Starting from each query $q \in \mathcal{Q}_{str}$, such an algorithm first unifies pairs

Input: a $DL\text{-}Lite_{A,id,den}$ TBox $\mathcal{T}$
Output: a set of queries
begin
  $\mathcal{Q}_{\mathcal{T}}^{unsat} \leftarrow \mathsf{unsatQueries}(\mathcal{T})$; /* step 1 */
  $\mathcal{Q}_{str} \leftarrow \mathsf{Saturate}(\mathcal{Q}_{\mathcal{T}}^{unsat})$; /* step 2 */
  $\mathcal{Q}'_{str} \leftarrow \mathcal{Q}_{str}$;
  foreach $q \in \mathcal{Q}'_{str}$ **do** /* step 3 */
    foreach atom $U(t, t')$ in q **do**
      if there exists no atom $T(t')$ in q **then**
        foreach value-domain T_i in $\{T_1, \ldots T_n\}$ **do**
          if $\mathcal{T} \models \rho(U) \sqsubseteq T_i$ **then**
            $\mathcal{Q}'_{str} \leftarrow \mathcal{Q}'_{str} \setminus \{q\}$;
            $\mathcal{Q}'_{str} \leftarrow \mathcal{Q}'_{str} \cup \{q \wedge T_i(t')\}$;
  foreach $q \in \mathcal{Q}'_{str}$ **do** /* step 4 */
    foreach term t occurring in q **do**
      if both $T_i(t)$ and $T_j(t)$ occur in q, with $i \neq j$ **then**
        $\mathcal{Q}'_{str} \leftarrow \mathcal{Q}'_{str} \setminus \{q\}$;
  foreach q and q' in $\mathcal{Q}'_{str}$ **do** /* step 5 */
    if $q \prec_{Rn} q'$ **then**
      $\mathcal{Q}'_{str} \leftarrow \mathcal{Q}'_{str} \setminus \{q'\}$;
  return $\mathcal{Q}'_{str}$; /* step 6 */
end

Algorithm 18: $\mathsf{minUnsatQueries}(\mathcal{T})$

of compatible terms in q in all possible ways; then, for any query q' computed in this way, for each pair of terms t_1 and t_2 occurring in q' that are syntactically different, it adds the inequality atom $t_1 \neq t_2$ to q'.

Step 3 Let $\{T_1 \ldots T_n\}$ be the set of value-domains. The algorithm computes the set $\mathcal{Q}'_{str}$ by producing from each query $q \in \mathcal{Q}_{str}$ the following query: for each atom $U(x, y)$, where U is an attribute name, if no atoms $T(y)$ appears in q, where T is a value-domain, then for each $T_i \in \{T_1 \ldots T_n\}$, if $\mathcal{T} \models \rho(U) \sqsubseteq T_i$, then the algorithm builds a new queries by substituting the atom $U(x, y)$ with the conjunction of atoms $U(x, y) \wedge T_i(y)$.

Step 4 The algorithm removes from $\mathcal{Q}'_{str}$ every query q in which a term t occurs in two atoms of the form $T_i(t)$ and $T_j(t)$ with $i \neq j$. This is an optimizing step, indeed, since T_i and T_j are disjoint then for each constant c in Σ_V, $T_1(c) \wedge T_2(c)$ is a contradiction.

Step 5 The algorithm removes from the set $\mathcal{Q}'_{str}$ each query q' such that there is in $\mathcal{Q}'_{str}$ a different query q whose atoms form, up to renaming

of the variables in q, a proper subset of the atoms appearing in q'. This simplified form of query containment guarantees that for every ABox $\mathcal{A}$ and every query q in $\mathcal{Q}'_{str}$, there does not exist a query q' in $\mathcal{Q}'_{str}$, such that for every image $V \in \mathsf{images}(q, \mathcal{A})$, $\langle \emptyset, V' \rangle \models q'$ where $V' \subset V$.

Step 6 The algorithm terminates by returning the set $\mathcal{Q}'_{str}$.

Example 38. Let us focus on the same portion of the TBox $\mathcal{T}$ of Example 2 considered in Example 37. The set $\mathsf{minUnsatQueries}(\mathcal{T})$ contains, among the others, the following queries:

$$
\begin{aligned}
q_2^1 &= \exists x, y, d, n.\mathit{of}(x, d) \wedge \mathit{number}(x, n) \wedge \mathit{of}(y, d) \wedge \mathit{number}(y, n) \wedge \\
&\quad \texttt{xsd:integer}(n) \wedge x \neq y \wedge x \neq d \wedge x \neq n \wedge y \neq d \wedge y \neq n \wedge d \neq n; \\
q_2^2 &= \exists x, y, n.\mathit{of}(x, y) \wedge \mathit{number}(x, n) \wedge \mathit{of}(y, y) \wedge \mathit{number}(y, n) \wedge \\
&\quad \texttt{xsd:integer}(n) \wedge x \neq y \wedge x \neq n \wedge y \neq n; \\
q_2^3 &= \exists x, y, n.\mathit{of}(x, x) \wedge \mathit{number}(x, n) \wedge \mathit{of}(y, x) \wedge \mathit{number}(y, n) \wedge \\
&\quad \texttt{xsd:integer}(n) \wedge x \neq y \wedge x \neq n \wedge y \neq n; \\
q_3^1 &= \exists x, y.\mathit{number}(x, y) \wedge \texttt{xsd:string}(y); \\
q_4^1 &= \exists x, y.\mathit{number}(x, y) \wedge \texttt{xsd:dateTime}(y).
\end{aligned}
$$

□

We now prove termination and complexity of the algorithm minUnsatQueries.

Lemma 45. *Let $\mathcal{T}$ be a DL-Lite$_{A,id,den}$ TBox. Then, $\mathsf{minUnsatQueries}(\mathcal{T})$ terminates and runs in exponential time with respect to the size of $\mathcal{T}$.*

Proof. The proof follows immediately from considering each steps of the algorithm.

1. In order to compute the set $\mathcal{Q}_{\mathcal{T}}^{unsat}$ the algorithm makes use of the algorithm $\mathsf{unsatQueries}(\mathcal{T})$. Lemma 9 guarantees that $\mathsf{unsatQueries}(\mathcal{T})$ terminates, and from Lemma 4 it follows that it runs in polynomial time in the size of $\mathcal{T}$.

2. The algorithm computes $\mathcal{Q}_{str}$ by means of the algorithm $\mathsf{Saturate}(\mathcal{Q}_{\mathcal{T}}^{unsat})$. Lemma 41 shows that $\mathsf{Saturate}(\mathcal{Q}_{\mathcal{T}}^{unsat})$ terminates and that runs in exponential time with respect to the size of $\mathcal{Q}_{\mathcal{T}}^{unsat}$.

3. The algorithm compute the set $\mathcal{Q}'_{str}$ as follows: (i) for each query q in $\mathcal{Q}_{str}$, if an atom $U(x, y)$ occurs in q and no atom $T(y)$ occurs in q, then for each $T_i \in \{T_1, \ldots T_n\}$ it checks if $\mathcal{T} \models \rho(U) \sqsubseteq T_i$ (this step can be done in polynomial time with respect to the size of $\mathcal{T}$); (ii) if

$\mathcal{T} \models \rho(U) \sqsubseteq T_i$, then the algorithm substitutes in q the atom $U(x, y)$ with the conjunction of atoms $U(x, y) \wedge T_i(y)$. Hence, the procedure can be done in polynomial time with respect to $\mathcal{Q}_{str}$ and $\{T_1, \ldots T_n\}$. No queries are added or removed. Hence, $|\mathcal{Q}'_{str}| = |\mathcal{Q}_{str}|$.

4. For each query q in $\mathcal{Q}'_{str}$ and for each term t in q, the algorithm checks if t occurs in two atoms of the form $T_i(t)$ and $T_j(t)$. In this case, it removes q from $\mathcal{Q}'_{str}$. This step can be clearly done in polynomial time with respect to $|\mathcal{Q}'_{str}|$. No query are added to $\mathcal{Q}'_{str}$ during the process, hence the resulting set has cardinality less than or equal to $\mathcal{Q}'_{str}$.

5. Finally, the algorithm removes from the set obtained by the previous step each query q such that there is another query q' such that $q' \prec_{Rn} q$. Since $\mathcal{Q}'_{str}$ is finite, then also this step terminates in polynomial time with respect to $|\mathcal{Q}'_{str}|$.

■

The following lemmas show that, given a *DL-Lite*$_{A,id,den}$ TBox, we can use the algorithm minUnsatQueries($\mathcal{T}$) for computing the set $\mathcal{Q}^{min}_{\mathcal{T}}$. The proofs are omitted since they can be straightforwardly adapted respectively from the proof of Lemma 43 and from the proof of Lemma 44, by observing that for each constant c in Σ_V, the sentence $T_1(c) \vee, \ldots, \vee T_n(c)$ evaluates to *true* over every ABox, and that $T_1(c) \wedge, \ldots, \wedge T_n(c)$ is a contradiction.

The next lemma states that the algorithm minUnsatQueries($\mathcal{T}$) can be used for checking if a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent.

Lemma 46. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB. Then, $\langle \emptyset, \mathcal{A} \rangle \models$ minUnsatQueries($\mathcal{T}$) if and only if $\langle \emptyset, \mathcal{A} \rangle \models$ unsatQueries($\mathcal{T}$).*

Note that, from the lemma above, it directly follows that Theorem 6 also holds with minUnsatQueries($\mathcal{T}$) in place of unsatQueries($\mathcal{T}$).

The following crucial lemma guarantees that one can use the queries produced by minUnsatQueries($\mathcal{T}$) in order to compute every minimal $\mathcal{T}$-inconsistent set in $\mathcal{A}$.

Lemma 47. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and let q be a query in minUnsatQueries($\mathcal{T}$). If $\langle \emptyset, \mathcal{A} \rangle \models q$, then every image of q in $\mathcal{A}$ is a minimal $\mathcal{T}$-inconsistent set.*

With the algorithm minUnsatQueries($\mathcal{T}$) in place we can provide our rewriting technique for query answering under *IAR*-semantics.

Let α, β be two atoms. We say that β is compatible with α if there exists a mapping μ of the variables occurring in β to the terms occurring in α such that $\mu(\beta) = \alpha$ (and in this case we denote the above mapping μ with the symbol

$\mu_{\alpha/\beta}$). Given an atom α and a query q, we denote by $CompSet(\alpha, q)$ the set of atoms of q which are compatible with α. Then, let $\mathcal{T}$ be a *DL-Lite*$_{A,id,den}$ TBox and let α be an atom, we define $MinIncSet^{\mathcal{T}}(\alpha)$ as follows.

$$MinIncSet^{\mathcal{T}}(\alpha) = \bigvee_{q \in \mathsf{minUnsatQueries}(\mathcal{T}) \wedge CompSet(\alpha,q) \neq \emptyset} \left(\bigvee_{\beta \in CompSet(\alpha,q)} \mu_{\alpha/\beta}(q) \right)$$

The following key property holds.

Theorem 37. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and let α be an ABox assertion. There exists a minimal $\mathcal{T}$-inconsistent set V in $\mathcal{A}$ such that $\alpha \in V$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models MinIncSet^{\mathcal{T}}(\alpha)$.*

Proof.
($\Rightarrow$) Let $\alpha(\vec{v})$ be an ABox assertion. We prove that if there is a minimal $\mathcal{T}$-inconsistent set V in $\mathcal{A}$ that contains $\alpha(\vec{v})$, then $\langle \emptyset, \mathcal{A} \rangle \models MinIncSet^{\mathcal{T}}(\alpha(\vec{v}))$. Suppose, by way of contradiction, that $\langle \emptyset, \mathcal{A} \rangle \not\models MinIncSet^{\mathcal{T}}(\alpha(\vec{v}))$. Since V is a minimal $\mathcal{T}$-inconsistent set in $\mathcal{A}$, then Lemma 47 guarantees that there is a query q in $\mathsf{minUnsatQueries}(\mathcal{T})$ such that $V \in \mathsf{images}(q, \mathcal{A})$. This means that there is a substitution σ from the variables in q to the constants in V such that the formula $\sigma(q)$ evaluates to *true* over $DB(V)$. Since $\alpha(\vec{v}) \in V$, then there is in q an atom $\alpha(\vec{t})$ such that $\sigma(\alpha(\vec{t})) = \alpha(\vec{v})$. But this means that the query $\mu_{\alpha(\vec{v})/\alpha(\vec{t})}(q)$ evaluates to *true* in $DB(V)$. Since $\alpha(\vec{t})$ is clearly compatible with $\alpha(\vec{v})$ then, by considering the union of queries $MinIncSet^{\mathcal{T}}(\alpha(\vec{v}))$ as set of queries, we have that $\mu_{\alpha(\vec{v})/\alpha(\vec{t})}(q) \in MinIncSet^{\mathcal{T}}(\alpha(\vec{v}))$. Thus, $\langle \emptyset, V \rangle \models MinIncSet^{\mathcal{T}}(\alpha(\vec{v}))$, and then $\langle \emptyset, \mathcal{A} \rangle \models MinIncSet^{\mathcal{T}}(\alpha(\vec{v}))$, which is a contradiction.
($\Leftarrow$) We show that if $\langle \emptyset, \mathcal{A} \rangle \models MinIncSet^{\mathcal{T}}(\alpha)$ then there is a minimal $\mathcal{T}$-inconsistent set V in $\mathcal{A}$ such that $\alpha \in V$. Suppose, toward a contradiction, that there is no minimal $\mathcal{T}$-inconsistent set V in $\mathcal{A}$ such that $\alpha \in V$. Let us consider $MinIncSet^{\mathcal{T}}(\alpha)$ as a set of queries. Hence, there is in $MinIncSet^{\mathcal{T}}(\alpha)$ a query q such that $\langle \emptyset, \mathcal{A} \rangle \models q$. Moreover, for every V' in $\mathsf{images}(q, \mathcal{A})$, we have that $\alpha \in V$'. Clearly, for each $V' \in \mathsf{images}(q, \mathcal{A})$, we have that $\langle \emptyset, V' \rangle \models q$. Since $q \in MinIncSet^{\mathcal{T}}(\alpha)$, then there is in $\mathsf{minUnsatQueries}(\mathcal{T})$ a query q' and an atom β in q' such that $q = \mu_{\alpha/\beta}(q')$. Since q' is more general than q, then $\mathsf{images}(q, \mathcal{A}) \subseteq \mathsf{images}(q', \mathcal{A})$. Hence, for each $V' \in \mathsf{images}(q, \mathcal{A})$, we have that $V' \in \mathsf{images}(q', \mathcal{A})$. Finally, by exploiting Lemma 47, we conclude that every $V' \in \mathsf{images}(q, \mathcal{A})$ is a minimal $\mathcal{T}$-inconsistent set. Thus, there is a minimal $\mathcal{T}$-inconsistent set V such that $\alpha \in V$, which is a contradiction. ∎

Let $\mathcal{T}$ be a *DL-Lite*$_{A,id,den}$ TBox, and let q be a BCQ with inequalities of the form

$$\exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} A_i(t_i^1) \wedge \bigwedge_{i=1}^{m} P_i(t_i^2, t_i^3) \wedge \bigwedge_{i=1}^{\ell} U_i(t_i^4, t_i^5) \wedge \bigwedge_{i=1}^{h} t_i^6 \neq t_i^7 \tag{9.1}$$

where every A_i is an atomic concept, every P_i is an atomic role, every U_i is an attribute, every t_i^e is a term (i.e., either a constant or a variable), and $z_1, \ldots, z_k$ are all the variables appearing in q. We denote by $IncRewr_{IAR}(q, \mathcal{T})$ the following FOL-sentence:

$$\begin{array}{lllll} \exists z_1, \ldots, z_k. & \bigwedge_{i=1}^{n} A_i(t_i^1) & \wedge & \neg MinIncSet^{\mathcal{T}}(A_i(t_i^1)) & \wedge \\ & \bigwedge_{i=1}^{m} P_i(t_i^2, t_i^3) & \wedge & \neg MinIncSet^{\mathcal{T}}(P_i(t_i^2, t_i^3)) & \wedge \\ & \bigwedge_{i=1}^{\ell} U_i(t_i^4, t_i^5) & \wedge & \neg MinIncSet^{\mathcal{T}}(U_i(t_i^4, t_i^5)) & \wedge \\ & \bigwedge_{i=1}^{h} t_i^6 \neq t_i^7 & & & \end{array}$$

Let $\mathcal{Q}$ be a set of queries of the form (9.1), we define

$$IncRewrUCQ_{IAR}(\mathcal{Q}, \mathcal{T}) = \bigvee_{q_i \in \mathcal{Q}} IncRewr_{IAR}(q_i, \mathcal{T}).$$

We are then able to give our final results on reformulation of UCQs under the *IAR*-semantics. In the theorem below, PerfectRef coincides with the algorithm of [95]. We recall that this algorithm takes as input the set of positive inclusions $\mathcal{T}_{inc}$ and the query $\mathcal{Q}$, and computes the perfect reformulation under FOL-semantics of $\mathcal{Q}$ with respect to $\mathcal{T}_{inc}$. PerfectRef$(\mathcal{Q}, \mathcal{T}_{inc})$ returns a set of CQs specified over $\mathcal{T}$. Through this reformulation we first preprocess each query according to "positive" knowledge of the TBox, and then manage it for dealing with possible inconsistencies. Then, the algorithm Saturate previously described is applied to the queries thus obtained producing a set of boolean CQ with inequalities of the form (9.1). Theorem 36 guarantees that the application of the algorithm Saturate does not affect the result of the evaluation of a query. This step is necessary for technical reasons. Roughly speaking, for each query atom α, this step is indispensable for exactly identifying the queries in minUnsatQueries$(\mathcal{T})$, whose images correspond to inconsistent sets to which an image of α might belong. According to the definition of $IncRewrUCQ_{IAR}$, this query is finally reformulated.

Theorem 38. *Let $\mathcal{T}$ be a DL-Lite$_{A,id,den}$ TBox and let $\mathcal{Q}$ be a boolean UCQ. For every ABox $\mathcal{A}$, we have that $\langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} \mathcal{Q}$ if and only if*

$$\langle \emptyset, \mathcal{A} \rangle \models IncRewrUCQ_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T}).$$

Proof.
($\Rightarrow$) Let $\mathcal{Q}$ be a boolean union of conjunctive queries represented as a set of queries. We show that if $\langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} \mathcal{Q}$ then

$$\langle \emptyset, \mathcal{A} \rangle \models \mathit{IncRewrUCQ}_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T}).$$

Since $\langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} \mathcal{Q}$ then, from Corollary 3, if follows that there are a query $q \in \mathcal{Q}$ and a set of ABox assertions $\mathcal{A}' \subseteq \mathcal{A}$, such that: (i) $\mathcal{A}'$ is $\mathcal{T}$-consistent; (ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models q$; (iii) $\mathcal{A}' \cap V = \emptyset$ for every $V \in \mathit{minIncSets}(\mathcal{K})$. Since there is a $\mathcal{T}$-consistent set of ABox assertions $\mathcal{A}' \subseteq \mathcal{A}$, such that $\langle \mathcal{T}, \mathcal{A}' \rangle \models q$, then from Lemma 13 we have that $\langle \emptyset, \mathcal{A}' \rangle \models \mathsf{PerfectRef}(q, \mathcal{T}_{inc})$, and therefore, $\langle \emptyset, \mathcal{A} \rangle \models \mathsf{PerfectRef}(q, \mathcal{T}_{inc})$. Theorem 36 guarantees that, if $\langle \emptyset, \mathcal{A} \rangle \models \mathsf{PerfectRef}(q, \mathcal{T}_{inc})$, then $\langle \emptyset, \mathcal{A} \rangle \models \mathsf{Saturate}(\mathsf{PerfectRef}(q, \mathcal{T}_{inc}))$. This means that there is in $\mathsf{Saturate}(\mathsf{PerfectRef}(q, \mathcal{T}_{inc}))$ a query q' such that $\langle \emptyset, \mathcal{A}' \rangle \models q'$. Let $q' = \exists \vec{z}. \bigwedge_{i=1}^{n} S_i(\vec{t_i}) \wedge \bigwedge_{i=1}^{h} t'_i \neq t''_i$. Since $\mathcal{A}' \cap V = \emptyset$ for every $V \in \mathit{minIncSets}(\mathcal{K})$, then Theorem 37 guarantees that for every assertions $\alpha \in \mathcal{A}'$, $\langle \emptyset, \mathcal{A}' \rangle \not\models \mathit{MinIncSet}^{\mathcal{T}}(\alpha)$. Hence, for every $\alpha \in \mathcal{A}'$ the sentence $\neg(\mathit{MinIncSet}^{\mathcal{T}}(\alpha))$ evaluates to *true* in $DB(\mathcal{A}')$. Hence the query

$$q'_{iar} = \exists \vec{z}. \bigwedge_{i=1}^{n} S_i(\vec{t_i}) \wedge \neg \mathit{MinIncSet}^{\mathcal{T}}(S_i(\vec{t_i})) \wedge \bigwedge_{i=1}^{h} t'_i \neq t''_i$$

evaluates to *true* in $DB(\mathcal{A}')$. Since $q'_{iar} = \mathit{IncRewr}_{IAR}(q', \mathcal{T})$, then $q'_{iar} \in \mathit{IncRewr}_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(q, \mathcal{T}_{inc})), \mathcal{T})$. Moreover, since $q \in \mathcal{Q}$, then $q'_{iar} \in \mathit{IncRewrUCQ}_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T})$. This means that

$$\langle \emptyset, \mathcal{A}' \rangle \models \mathit{IncRewrUCQ}_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T}).$$

From the fact that $\mathcal{A}' \subseteq \mathcal{A}$, we can conclude that

$$\langle \emptyset, \mathcal{A} \rangle \models \mathit{IncRewrUCQ}_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T}).$$

Hence, the thesis is true.

($\Leftarrow$) Let $\mathcal{Q}^{IAR}_{rew} = \mathit{IncRewrUCQ}_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T})$. Suppose that $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}^{IAR}_{rew}$. We show that $\langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} \mathcal{Q}$. Let q be a query in $\mathcal{Q}^{IAR}_{rew}$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$. Suppose that q is as follows

$$\exists \vec{z}. \bigwedge_{i=1}^{n} S_i(\vec{t_i}) \wedge \neg \mathit{MinIncSet}^{\mathcal{T}}(S_i(\vec{t_i})) \wedge \bigwedge_{i=1}^{h} t'_i \neq t''_i$$

Since $\langle \emptyset, \mathcal{A} \rangle \models q$, then, for each $S_i(\vec{t_i})$ with $1 \leq i \leq n$, $\mathit{MinIncSet}^{\mathcal{T}}(S_i(\vec{t_i}))$ is false in $DB(\mathcal{A})$. Let q' be the query in $\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc}))$ such that $q = \mathit{IncRewr}_{IAR}(q', \mathcal{T})$. Since $\langle \emptyset, \mathcal{A} \rangle \models q$, then we have that $\langle \emptyset, \mathcal{A} \rangle \models q'$ and,

from Theorem 37, that for every assertion $\alpha \in \mathcal{A}'$, with $\mathcal{A}' \in \mathsf{images}(q', \mathcal{A})$, we have that $\alpha \notin V$, for each $V \in minIncSets(\mathcal{K})$. Thus, for each $V \in minIncSets(\mathcal{K})$, $\mathcal{A}' \cap V = \emptyset$, and therefore $\mathcal{A}'$ is a $\mathcal{T}$-consistent subset of $\mathcal{A}$. From Lemma 13 and Theorem 36 we have that there is in $\mathcal{Q}$ a query q'' such that $q' \in \mathsf{Saturate}(\mathsf{PerfectRef}(q'', \mathcal{T}_{inc}))$, and such that $\langle \mathcal{T}, \mathcal{A}' \rangle \models q''$. Hence, (i) $\mathcal{A}'$ is $\mathcal{T}$-consistent; (ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models q''$; (iii) $\mathcal{A}' \cap V = \emptyset$ for every $V \in minIncSets(\mathcal{K})$. Since $q'' \in \mathcal{Q}$, then from Corollary 3, it follows that $\langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} \mathcal{Q}$. Hence, we have the claim. ∎

The following complexity result is a consequence of Theorem 38, since establishing whether $\langle \emptyset, \mathcal{A} \rangle \models IncRewrUCQ_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T})$ simply amounts to evaluating a FOL-query over the ABox $\mathcal{A}$, which is in AC^0 in data complexity, that is, the complexity computed with respect to the size of the ABox only.

Corollary 5. *Let $\mathcal{K}$ be a $DL\text{-}Lite_{A,id,den}$-KB and let $\mathcal{Q}$ be a UCQ. Deciding whether $\mathcal{K} \models_{IAR} \mathcal{Q}$ is in AC^0 in data complexity.*

Proof. The proof follows from Theorem 38 and from the complexity of evaluation of FOL-queries over relational databases in data complexity [1]. ∎

Corollary 5 and Lemma 45, together with the complexity results given in Section 5.1 and in Section 5.2 allow us to give the following theorem.

Theorem 39. *Let $\mathcal{K}$ be a $DL\text{-}Lite_{A,id,den}$-KB and let $\mathcal{Q}$ be a UCQ. Deciding whether $\mathcal{K} \models_{IAR} \mathcal{Q}$ can be done in AC^0 with respect to $|\mathcal{A}|$, and in exponential time with respect to $|\mathcal{T}|$ and $|\mathcal{Q}|$.*

Proof. The proof follows from the following observations:

1. Corollary 5 guarantees that $\mathcal{K} \models_{IAR} \mathcal{Q}$ can be decided in AC^0 with respect to $|\mathcal{A}|$.
2. From Lemma 4 we have that the perfect reformulation of a query $q \in \mathcal{Q}$ can be computed in polynomial time in the size of $\mathcal{T}$ and in exponential time with respect to $|q|$. Lemma 41 shows that we can compute $\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc}))$ in exponential time with respect to $|\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})|$.
3. Lemma 45 states that the set $\mathcal{Q}_{\mathcal{T}}^{min}$ can be computed in exponential time with respect to $\mathcal{T}$. Hence, given a query q, $IncRewr_{IAR}(q, \mathcal{T})$ can be computed in polynomial time with respect $|q|$ and in exponential time with respect to $|\mathcal{T}|$. Thus, since q is a query in $\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc}))$, we obtain the claim.

∎

With the aim to clarify our technique for computing FOL-rewritings of BUCQs under the *IAR*-semantics, we conclude this section by presenting an example illustrating the whole procedure.

Example 39. In this example we consider a simple *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ describing a portion of the domain of Airports and Flights.

The TBox $\mathcal{T}$ is constituted by the following assertions.

- the set $\mathcal{T}_{inc}$ contains the following assertions:
  - $\mathsf{Flight} \sqsubseteq \exists \mathsf{arrival}$,
  - $\mathsf{Flight} \sqsubseteq \exists \mathsf{departure}$,
  - $\exists \mathsf{arrival} \sqsubseteq \mathsf{Flight}$,
  - $\exists \mathsf{departure} \sqsubseteq \mathsf{Flight}$,
  - $\exists \mathsf{arrival}^- \sqsubseteq \mathsf{Airport}$,
  - $\exists \mathsf{departure}^- \sqsubseteq \mathsf{Airport}$,
  - $\mathsf{Airport} \sqsubseteq \delta(name)$,
  - $\exists \mathsf{locatedIn} \sqsubseteq \mathsf{Airport}$,
  - $\mathsf{Airport} \sqsubseteq \exists \mathsf{locatedIn}$,
  - $\exists \mathsf{locatedIn}^- \sqsubseteq \mathsf{City}$,
  - $\mathsf{City} \sqsubseteq \delta(name)$;
- the set $\mathcal{T}_{type}$ contains the following assertion:
  - $\rho(name) \sqsubseteq \texttt{xsd:string}$;
- the set $\mathcal{T}_{disj}$ contains the following assertions:
  - $\mathsf{Flight} \sqsubseteq \neg \mathsf{Airport}$,
  - $\mathsf{Airport} \sqsubseteq \neg \mathsf{City}$,
  - $\mathsf{Flight} \sqsubseteq \neg \mathsf{City}$;
- the set $\mathcal{T}_{funct}$ contains the following functionality assertions:
  - $(\mathsf{funct\ locatedIn})$,
  - $(\mathsf{funct}\ name)$,
  - $(\mathsf{funct\ departure})$,
  - $(\mathsf{funct\ arrival})$;
- the set $\mathcal{T}_{id}$ contains the following identification assertion:
  - $(\mathsf{id\ City}\ name)$,
  - $(\mathsf{id\ Airport}\ name, \mathsf{locatedIn})$;
- the set $\mathcal{T}_{den}$ contains the following assertions:
  - $\forall x, y.(\mathsf{arrival}(x, y) \wedge \mathsf{departure}(x, y) \rightarrow \bot)$
  - $\forall x, y, w, c.(\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, c) \rightarrow \bot)$

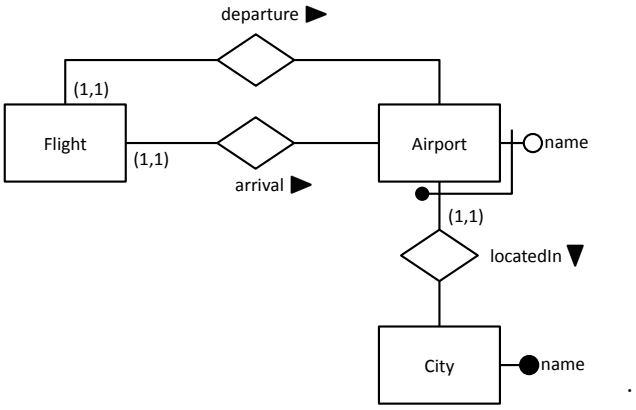

Figure 9.1: Airports and Flights scenario

In words, $\mathcal{T}$ specifies that every Airport is located in (role locatedIn) exactly one City, and that every Airport has exactly one *name* that is a string. By means of the identification assertion (id Airport *name*, locatedIn) we impose that two Airports with the same *name* cannot be located in the same City. The role locatedIn has Airport as domain and City as range. Every City has exactly one *name*. The identification assertion (id City *name*) imposes that there cannot be two City instances with the same *name*. Moreover, every Flight takes off from exactly one Airport (role departure) and lands in exactly one Airport (role arrival). Both roles departure and arrival have Flight as domain and Airport as range. Finally, a Flight is neither an Airport nor a City, and a City is not an Airport.

In order to better express the domain of interest, we include in the TBox the denial assertions in $\mathcal{T}_{den}$. The first denial assertion imposes that a Flight cannot depart from the same Airport in which it arrives, and the second one imposes that a Flight cannot link two airports located in the same City. The TBox $\mathcal{T}$ can be depicted by means of the diagram presented in Figure 9.1.

For this example we consider the ABox $\mathcal{A}$ as formed by the following ABox assertions.

$$\begin{array}{lll} \mathsf{Airport}(LHR), & \mathsf{locatedIn}(LHR, london), & \mathsf{Flight}(f1), \\ \mathsf{arrival}(f1, LHR), & \mathsf{departure}(f1, LHR). & \end{array}$$

It is easy to see that the KB $\mathcal{K}$ is inconsistent because both the denial assertions in $\mathcal{T}$ are violated by the following set of ABox assertions.

$$\begin{array}{rcl} \mathcal{A}_R & = & \{\mathsf{departure}(f1, LHR), \mathsf{arrival}(f1, LHR), \mathsf{locatedIn}(LHR, london)\} \\ \mathcal{A}'_R & = & \{\mathsf{departure}(f1, LHR), \mathsf{arrival}(f1, LHR))\} \end{array}$$

Note that the set $\mathcal{A}_R$ is not a minimal $\mathcal{T}$-inconsistent set, indeed the set $\mathcal{A}'_R$ violates the first denial assertion in the TBox and is a proper subset of $\mathcal{A}_R$. The

set *AR-Set*($\mathcal{K}$) containing the *A*-repairs of $\mathcal{K}$ is constituted by the following sets of ABox assertions.

$$\begin{array}{rcl} \mathcal{A}_{rep1} & = & \{\mathsf{Airport}(LHR), \mathsf{Flight}(f1), \mathsf{arrival}(f1, LHR), \mathsf{locatedIn}(LHR, london)\} \\ \mathcal{A}_{rep2} & = & \{\mathsf{Airport}(LHR), \mathsf{Flight}(f1), \mathsf{departure}(f1, LHR), \mathsf{locatedIn}(LHR, london)\} \end{array}$$

Suppose we are interested in asking if an airport located in *london* exists. We can do this by evaluating the following boolean query q.

$$q \ = \ \exists x.\mathsf{locatedIn}(x, london)$$

One can immediately verify that $\mathcal{K} \models_{IAR} q$. Indeed we have that

$$\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in AR\text{-}Set(\mathcal{K})} \mathcal{A}_1 \rangle = \langle \mathcal{T}, \{\mathsf{Airport}(LHR), \mathsf{Flight}(f1), \mathsf{locatedIn}(LHR, london)\} \rangle \models q$$

In what follows, we show how our technique computes the *IAR-perfect rewriting* of q. For reasons related to the size of the result produced by the procedure, we suppose here that the only assertions in the TBox that can be violated by facts in the ABox are the two denial assertions in $\mathcal{T}$. For the same reasons, we do not illustrate the steps of the algorithm $\mathsf{minUnsatQueries}(\mathcal{T})$ that concern violations caused by erroneous assignments of values to attributes. In other words, we consider the TBox $\mathcal{T}$ constituted as follows:

- the set $\mathcal{T}_{inc}$ contains the following positive assertions:
  - $\mathsf{Flight} \sqsubseteq \exists \mathsf{arrival}$,
  - $\exists \mathsf{arrival} \sqsubseteq \mathsf{Flight}$,
  - $\exists \mathsf{arrival}^- \sqsubseteq \mathsf{Airport}$,
  - $\exists \mathsf{locatedIn} \sqsubseteq \mathsf{Airport}$,
  - $\mathsf{Airport} \sqsubseteq \exists \mathsf{locatedIn}$;
  - $\mathsf{Flight} \sqsubseteq \exists \mathsf{departure}$,
  - $\exists \mathsf{departure} \sqsubseteq \mathsf{Flight}$,
  - $\exists \mathsf{departure}^- \sqsubseteq \mathsf{Airport}$,
  - $\exists \mathsf{locatedIn}^- \sqsubseteq \mathsf{City}$,
- the set $\mathcal{T}_{den}$ contains the following denial assertions:
  - $\forall x, y.(\mathsf{arrival}(x, y) \wedge \mathsf{departure}(x, y) \ \rightarrow \ \bot)$
  - $\forall x, y, w, c.(\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, c) \ \rightarrow \ \bot)$;
- moreover, we consider $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id} = \emptyset$.

We start illustrating how the algorithm $\mathsf{minUnsatQueries}(\mathcal{T})$ proceeds.

Since $\mathcal{T}_{type} \cup \mathcal{T}_{disj} \cup \mathcal{T}_{funct} \cup \mathcal{T}_{id} = \emptyset$, the set $cln(\mathcal{T})$ coincides with $\mathcal{T}_{den}$. The first step of $\mathsf{minUnsatQueries}(\mathcal{T})$ consists in computing the set $\mathsf{unsatQueries}(\mathcal{T})$ as follows: it first computes for each assertion $\tau \in cln(\mathcal{T})$ the boolean query corresponding to the negation of τ by means of the translation function φ presented in Section 5.1, and then computes the perfect reformulation under FOL-semantics of such query by means of the algorithm $\mathsf{PerfectRef}_{\neq}$.

The set $\mathsf{unsatQueries}(\mathcal{T})$ contains the following queries.

$q' = \exists x, y.\mathsf{arrival}(x, y) \wedge \mathsf{departure}(x, y)$
$q'' = \exists x, y, w, c.\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, c)$

The second step consists in computing the set $\mathsf{Saturate}(\mathsf{unsatQueries}(\mathcal{T}))$. The resulting set is constituted by the following queries.

$q_1 = \exists x, y.\mathsf{arrival}(x, y) \wedge \mathsf{departure}(x, y) \wedge x \neq y$
$q_2 = \exists x.\mathsf{arrival}(x, x) \wedge \mathsf{departure}(x, x)$
$q_3 = \exists x, y, w, c.\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, c) \wedge$
$x \neq y \wedge x \neq w \wedge x \neq c \wedge y \neq w \wedge y \neq c \wedge w \neq c$
$q_4 = \exists x, y, w.\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(x, c) \wedge \mathsf{locatedIn}(c, c) \wedge$
$x \neq y \wedge x \neq c \wedge y \neq c$
$q_5 = \exists x, w, c.\mathsf{arrival}(x, c) \wedge \mathsf{locatedIn}(c, c) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, c) \wedge$
$x \neq w \wedge x \neq c \wedge w \neq c$
$q_6 = \exists y, w, c.\mathsf{arrival}(c, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(c, w) \wedge \mathsf{locatedIn}(w, c) \wedge$
$y \neq w \wedge y \neq c \wedge w \neq c$
$q_7 = \exists x, w, c.\mathsf{arrival}(x, w) \wedge \mathsf{locatedIn}(w, c) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, c) \wedge$
$x \neq w \wedge x \neq c \wedge w \neq c$
$q_8 = \exists y, w, c.\mathsf{arrival}(w, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(w, w) \wedge \mathsf{locatedIn}(w, c) \wedge$
$y \neq w \wedge y \neq c \wedge w \neq c$
$q_9 = \exists y, w, c.\mathsf{arrival}(y, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(y, w) \wedge \mathsf{locatedIn}(w, c) \wedge$
$y \neq w \wedge y \neq c \wedge w \neq c$
$q_{10} = \exists x, y.\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, y) \wedge \mathsf{departure}(x, y) \wedge \mathsf{locatedIn}(y, y) \wedge$
$x \neq y$
$q_{11} = \exists x, y.\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, x) \wedge \mathsf{departure}(x, x) \wedge \mathsf{locatedIn}(x, x) \wedge$
$x \neq y$
$q_{12} = \exists x, w.\mathsf{arrival}(x, w) \wedge \mathsf{locatedIn}(w, w) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, w) \wedge$
$x \neq w$
$q_{13} = \exists x, w.\mathsf{arrival}(x, x) \wedge \mathsf{locatedIn}(x, x) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, x) \wedge$
$x \neq w$
$q_{14} = \exists x, c.\mathsf{arrival}(x, c) \wedge \mathsf{locatedIn}(c, c) \wedge \mathsf{departure}(x, c) \wedge \mathsf{locatedIn}(c, c) \wedge$
$x \neq c$
$q_{15} = \exists x, c.\mathsf{arrival}(x, x) \wedge \mathsf{locatedIn}(x, c) \wedge \mathsf{departure}(x, x) \wedge \mathsf{locatedIn}(x, c) \wedge$
$x \neq c$
$q_{16} = \exists y, w.\mathsf{arrival}(w, y) \wedge \mathsf{locatedIn}(y, w) \wedge \mathsf{departure}(w, w) \wedge \mathsf{locatedIn}(w, w) \wedge$
$y \neq w$
$q_{17} = \exists y, w.\mathsf{arrival}(y, y) \wedge \mathsf{locatedIn}(y, y) \wedge \mathsf{departure}(y, w) \wedge \mathsf{locatedIn}(w, y) \wedge$
$y \neq w$
$q_{18} = \exists y, c.\mathsf{arrival}(c, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(c, c) \wedge \mathsf{locatedIn}(c, c) \wedge$
$y \neq c$
$q_{19} = \exists y, c.\mathsf{arrival}(y, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(y, y) \wedge \mathsf{locatedIn}(y, c) \wedge$
$y \neq c$
$q_{20} = \exists w, c.\mathsf{arrival}(c, c) \wedge \mathsf{locatedIn}(c, c) \wedge \mathsf{departure}(c, w) \wedge \mathsf{locatedIn}(c, c) \wedge$
$w \neq c$
$q_{21} = \exists w, c.\mathsf{arrival}(w, w) \wedge \mathsf{locatedIn}(w, c) \wedge \mathsf{departure}(w, w) \wedge \mathsf{locatedIn}(w, c) \wedge$
$w \neq c$
$q_{22} = \exists x.\mathsf{arrival}(x, x) \wedge \mathsf{departure}(x, x) \wedge \mathsf{locatedIn}(x, x)$

As we said above, we are not considering those parts of the rewriting process concerning violations caused by erroneous assignments of values to attributes. For this reason, by skipping steps 3 and 4, we go directly to analyze step 5 of $\mathsf{minUnsatQueries}(\mathcal{T})$. This step requires to remove, from the above set of queries, each query q' such that there exists a different query q in the set whose atoms form, up to renaming of the variables in q, a proper subset of the atoms appearing in q'. For instance, the set of atoms in the query

$$q_1 = \exists x, y.\mathsf{arrival}(x, y) \wedge \mathsf{departure}(x, y) \wedge x \neq y$$

forms, up to renaming of variables, a subset of the set of atoms occurring in the query

$$q_7 = \exists x, w, c.\mathsf{arrival}(x, w) \wedge \mathsf{locatedIn}(w, c) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, c) \wedge x \neq w \wedge x \neq c \wedge w \neq c$$

The set of queries resulting from step 5 contains the following queries, which constitutes the set returned by $\mathsf{minUnsatQueries}(\mathcal{T})$:

$$\begin{aligned}
q_1 &= \exists x, y.\mathsf{arrival}(x, y) \wedge \mathsf{departure}(x, y) \wedge x \neq y \\
q_2 &= \exists x.\mathsf{arrival}(x, x) \wedge \mathsf{departure}(x, x) \\
q_3 &= \exists x, y, w, c.\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, c) \wedge \\
&\quad x \neq y \wedge x \neq w \wedge x \neq c \wedge y \neq w \wedge y \neq c \wedge w \neq c \\
q_4 &= \exists x, y, c.\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(x, c) \wedge \mathsf{locatedIn}(c, c) \wedge \\
&\quad x \neq y \wedge x \neq c \wedge y \neq c \\
q_5 &= \exists x, w, c.\mathsf{arrival}(x, c) \wedge \mathsf{locatedIn}(c, c) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, c) \wedge \\
&\quad x \neq w \wedge x \neq c \wedge w \neq c \\
q_6 &= \exists y, w, c.\mathsf{arrival}(c, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(c, w) \wedge \mathsf{locatedIn}(w, c) \wedge \\
&\quad y \neq w \wedge y \neq c \wedge w \neq c \\
q_8 &= \exists y, w, c.\mathsf{arrival}(w, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(w, w) \wedge \mathsf{locatedIn}(w, c) \wedge \\
&\quad y \neq w \wedge y \neq c \wedge w \neq c \\
q_9 &= \exists y, w, c.\mathsf{arrival}(y, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(y, w) \wedge \mathsf{locatedIn}(w, c) \wedge \\
&\quad y \neq w \wedge y \neq c \wedge w \neq c \\
q_{11} &= \exists x, y.\mathsf{arrival}(x, y) \wedge \mathsf{locatedIn}(y, x) \wedge \mathsf{departure}(x, x) \wedge \mathsf{locatedIn}(x, x) \wedge \\
&\quad x \neq y \\
q_{13} &= \exists x, w.\mathsf{arrival}(x, x) \wedge \mathsf{locatedIn}(x, x) \wedge \mathsf{departure}(x, w) \wedge \mathsf{locatedIn}(w, x) \wedge \\
&\quad x \neq w \\
q_{16} &= \exists y, w.\mathsf{arrival}(w, y) \wedge \mathsf{locatedIn}(y, w) \wedge \mathsf{departure}(w, w) \wedge \mathsf{locatedIn}(w, w) \wedge \\
&\quad y \neq w \\
q_{17} &= \exists y, w.\mathsf{arrival}(y, y) \wedge \mathsf{locatedIn}(y, y) \wedge \mathsf{departure}(y, w) \wedge \mathsf{locatedIn}(w, y) \wedge \\
&\quad y \neq w \\
q_{18} &= \exists y, c.\mathsf{arrival}(c, y) \wedge \mathsf{locatedIn}(y, c) \wedge \mathsf{departure}(c, c) \wedge \mathsf{locatedIn}(c, c) \wedge \\
&\quad y \neq c \\
q_{20} &= \exists w, c.\mathsf{arrival}(c, c) \wedge \mathsf{locatedIn}(c, c) \wedge \mathsf{departure}(c, w) \wedge \mathsf{locatedIn}(c, c) \wedge \\
&\quad w \neq c
\end{aligned}$$

With the result of the algorithm $\mathsf{minUnsatQueries}(\mathcal{T})$ in place, we can compute the set of FOL-sentences

$$IncRewr_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(q, \mathcal{T}_{inc})), \mathcal{T})$$

that corresponds to the AR-perfect rewriting of the query q. It is easy to see that the set $\mathsf{Saturate}(\mathsf{PerfectRef}(q, \mathcal{T}_{inc}))$ is constituted by the following queries.

$q_{u1} = \exists x.\mathsf{locatedIn}(x, london)$
$q_{u2} = \mathsf{locatedIn}(london, london)$

Finally, for each query q_u in $\mathsf{Saturate}(\mathsf{PerfectRef}(q, \mathcal{T}_{inc}))$, we compute the FOL-sentence $IncRewr_{IAR}(q_u, \mathcal{T})$. Below we give an hint of the result.

$IncRewr_{IAR}(\exists x.\mathsf{locatedIn}(x, london), \mathcal{T}) =$
$\exists x.\mathsf{locatedIn}(x, london) \wedge$
$\neg(\exists x', w.\mathsf{arrival}(x', x) \wedge \mathsf{locatedIn}(x, london) \wedge \mathsf{departure}(x', w) \wedge \mathsf{locatedIn}(w, london) \wedge$
$x' \neq x \wedge x' \neq w \wedge x' \neq london \wedge x \neq w \wedge x \neq london \wedge w \neq london$
$\vee\ \exists x', y.\mathsf{arrival}(x', y) \wedge \mathsf{locatedIn}(y, london) \wedge \mathsf{departure}(x', x) \wedge \mathsf{locatedIn}(x, london) \wedge$
$x' \neq y \wedge x' \neq x \wedge x' \neq c \wedge y \neq x \wedge y \neq london \wedge x \neq c$
$\vee\ \exists x'.\mathsf{arrival}(x', x) \wedge \mathsf{locatedIn}(x, london) \wedge \mathsf{departure}(x', london) \wedge$
$\mathsf{locatedIn}(london, london) \wedge x' \neq y \wedge x' \neq london \wedge x \neq london$
$\vdots$
$\vdots$
$\vee\ \mathsf{arrival}(london, x) \wedge \mathsf{locatedIn}(x, london) \wedge \mathsf{departure}(london, london) \wedge$
$\mathsf{locatedIn}(london, london) \wedge x \neq london\ )$

$IncRewr_{IAR}(\exists x.\mathsf{locatedIn}(london, london), \mathcal{T}) =$
$\mathsf{locatedIn}(london, london) \wedge$
$\neg(\exists x', w.\mathsf{arrival}(x', london) \wedge \mathsf{locatedIn}(london, london) \wedge \mathsf{departure}(x', w) \wedge$
$\mathsf{locatedIn}(w, london) \wedge x' \neq london \wedge x' \neq w \wedge x' \neq london \wedge x \neq w \wedge$
$london \neq london \wedge w \neq london$
$\vee\ \exists x', y.\mathsf{arrival}(x', y) \wedge \mathsf{locatedIn}(y, london) \wedge \mathsf{departure}(x', london) \wedge$
$\mathsf{locatedIn}(london, london) \wedge x' \neq y \wedge x' \neq london \wedge x' \neq c \wedge y \neq london \wedge$
$y \neq london \wedge x \neq c$
$\vee\ \exists x'.\mathsf{arrival}(x', london) \wedge \mathsf{locatedIn}(london, london) \wedge \mathsf{departure}(x', london) \wedge$
$\mathsf{locatedIn}(london, london) \wedge x' \neq y \wedge x' \neq london \wedge london \neq london$
$\vdots$
$\vdots$
$\vee\ \mathsf{arrival}(london, london) \wedge \mathsf{locatedIn}(london, london) \wedge \mathsf{departure}(london, london) \wedge$
$\mathsf{locatedIn}(london, london) \wedge london \neq london\ )$

It can be shown that $\langle \emptyset, \mathcal{A} \rangle \models IncRewr_{IAR}(q, \mathcal{T})$. In particular we have that $\langle \emptyset, \mathcal{A} \rangle \models IncRewr_{IAR}(\exists x.\mathsf{locatedIn}(x, london), \mathcal{T})$. □

9.4 Consistent query answering in *DL-Lite*$_{A,id,den}$ under *ICAR*-semantics

In this section we analyze the problem of answering BUCQs posed over a KB expressed in *DL-Lite*$_{A,id,den}$ under *ICAR*-semantics. In details, we present a technique for computing FOL-rewritings of boolean UCQs under the *ICAR*-semantics. Thus, we show that the problem of deciding if a BUCQ is *ICAR*-entailed by a *DL-Lite*$_{A,id,den}$-KB is in AC^0 with respect to data complexity.

Similarly to *IAR*-semantics, we say that BUCQs in *DL-Lite*$_{A,id,den}$ are FOL-rewritable under the *ICAR*-semantics if, for each BUCQs q and each *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$, there exists a FOL-query q_r such that, for any ABox $\mathcal{A}$. $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} q$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models q_r$. We call such q_r the *ICAR-perfect reformulation* of q with respect to $\mathcal{T}$.

The same reason that guarantees that Corollary 3 holds for *DL-Lite*$_{A,id,den}$-KBs, i.e., that a *DL-Lite*$_{A,id,den}$ TBox is always consistent, also guarantees that Corollary 4 holds for *DL-Lite*$_{A,id,den}$-KBs. Thus, given a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a boolean conjunctive query q, we have that $\mathcal{K} \models_{ICAR} q$ if and only if there exists $\mathcal{A}' \subseteq \mathsf{clc}_{\mathcal{T}}(A)$ such that:

(i) $\mathcal{A}'$ is $\mathcal{T}$-consistent;

(ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models q$;

(iii) there is no minimal $\mathcal{T}$-inconsistent set in $\mathsf{clc}_{\mathcal{T}}(A)$ such that $\mathcal{A}' \cap V \neq \emptyset$.

Hence, a query q is *ICAR*-entailed from a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if there is an image of q in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ that does not overlap with any minimal $\mathcal{T}$-inconsistent set in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Therefore, in the same spirit of our technique for rewriting query under *IAR*-semantics, we can base our reformulation method for *ICAR*-semantics on the idea of rewriting each atom α of q into a FOL-formula α_r in such a way that $\langle \emptyset, \mathcal{A} \rangle \models \alpha_r$ only if $\langle \emptyset, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \rangle \models \alpha$ and there is no image of α_r in $\mathcal{A}$ that overlap with a minimal $\mathcal{T}$-inconsistent set.

The first step for reaching our purpose is, given a *DL-Lite*$_{A,id,den}$-KB $\langle \mathcal{T}, \mathcal{A} \rangle$, studying the conditions under which an atom α belong to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. We recall that an atom $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ if there is in $\mathcal{A}$ a $\mathcal{T}$-consistent set $\mathcal{A}'$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle \models \alpha$. From Lemma 16, we have that in *DL-Lite*$_{A,id,den}$ an assertion is entailed by a consistent KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if there exists in $\mathcal{A}$ an assertion β such that $\langle \mathcal{T}, \{\beta\} \rangle \models \alpha$. Thus, given a *DL-Lite*$_{A,id,den}$-KB $\langle \mathcal{T}, \mathcal{A} \rangle$, an assertions α, belongs to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ if:

(i) $\alpha \in \mathcal{A}$ and $\{\alpha\}$ is not $\mathcal{T}$-inconsistent, or

(ii) $\alpha \notin \mathcal{A}$, but there is an assertion $\beta \in \mathcal{A}$ such that $\{\beta\}$ is not $\mathcal{T}$-inconsistent and $\langle \mathcal{T}, \{\beta\} \rangle \models \alpha$

Example 40. Consider the *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$ constituted as follows:

$$\begin{array}{lcll}
\mathcal{T}_{inc} & = & \{ & \mathsf{Man} \sqsubseteq \mathsf{Person}, \quad \mathsf{Woman} \sqsubseteq \mathsf{Person}, \\
 & & & \exists \mathsf{hasFather}^{-} \sqsubseteq \mathsf{Man} \quad \}; \\
\mathcal{T}_{disj} & = & \{ & \mathsf{Man} \sqsubseteq \neg \mathsf{Woman} \quad \}; \\
\mathcal{T}_{funct} & = & \{ & (funct\ \mathsf{hasFather}) \quad \}; \\
\mathcal{T}_{den} & = & \{ & \forall x.(\mathsf{hasFather}(x,x) \rightarrow \bot) \quad \}.
\end{array}$$

In words, $\mathcal{T}$ states that both men and women are persons, that a man cannot be a woman, and that a person that is father of somebody is a man. Moreover, the functionality assertion specifies that one can have at most one father, while the denial assertion guarantees that one cannot be father of oneself.

Let $\mathcal{A}$ be the ABox containing the following assertions:

$$\begin{array}{lcll}
\mathcal{A} & = & \{ & \mathsf{hasFather}(sam, bill), \quad \mathsf{Man}(sam), \quad \mathsf{Woman}(sam), \\
 & & & \mathsf{hasFather}(tom, tom) \quad \}.
\end{array}$$

It is easy to verify that the KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent. Indeed, the two assertions $\mathsf{Man}(sam)$ and $\mathsf{Woman}(sam)$ in $\mathcal{A}$ violate the disjointness $\mathsf{Man} \sqsubseteq \neg \mathsf{Woman}$ belonging to $\mathcal{T}$. Moreover, the assertion $\mathsf{hasFather}(tom, tom)$ violates the denial assertion in $\mathcal{T}$.

The set $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ contains the following ABox assertions:

$$\begin{array}{lcll}
\mathsf{clc}_{\mathcal{T}}(A) & = & \{ & \mathsf{hasFather}(sam, bill), \quad \mathsf{Man}(sam), \quad \mathsf{Woman}(sam), \\
 & & & \mathsf{Person}(sam), \quad \mathsf{Man}(bill), \quad \mathsf{Person}(bill) \quad \}.
\end{array}$$

Note that the assertions $\mathsf{hasFather}(tom, tom)$ does not belong to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Moreover, in accordance with Definition 25, the set *CAR-Set*$(\langle \mathcal{T}, \mathcal{A} \rangle)$ is constituted by the following $\mathcal{T}$-consistent sets of ABox assertions:

$$\begin{array}{l}
\textit{CA-rep}_1 = \{\ \mathsf{Man}(sam), \mathsf{Person}(sam), \mathsf{hasParent}(sam, bill), \\
\qquad\qquad \mathsf{Man}(bill), \mathsf{Person}(bill)\ \}; \\
\textit{CA-rep}_2 = \{\ \mathsf{Woman}(sam), \mathsf{Person}(sam), \mathsf{hasParent}(sam, bill), \\
\qquad\qquad \mathsf{Man}(bill), \mathsf{Person}(bill)\ \}.
\end{array}$$

□

Since for every *DL-Lite*$_{A,id,den}$ TBox, we have that $Mod(\langle \mathcal{T}, \emptyset \rangle) \neq \emptyset$, clearly, we have that every singleton $\{\alpha\}$ that is $\mathcal{T}$-inconsistent is a minimal $\mathcal{T}$-inconsistent set. Thus, from Lemma 47, it follows that there exists in $\mathsf{minUnsatQueries}(\mathcal{T})$ a boolean query q_{us} such that $\langle \emptyset, \{\alpha\} \rangle \models q_{us}$. Since every query $q' \in \mathsf{minUnsatQueries}(\mathcal{T})$ is "*saturated with respect to inequalities*", i.e., $\mathsf{Saturate}(q') = q'$, then q_{us} has the form of one of the following boolean queries:

1. $\exists x.A(x)$, where A is an atomic concepts and x is a variable;
2. $A(c)$, where A is an atomic concepts and c is a constant;
3. $\exists x, y.P(x, y) \wedge x \neq y$, where P is an atomic role and x and y are variables;
4. $\exists x.P(x, x)$, where P is an atomic role and x is a variable;
5. $\exists x.P(x, c) \wedge x \neq c$ (resp. $P(c, x) \wedge x \neq c$), where P is an atomic role, x is a variable, and c is a constant;
6. $P(c_1, c_2) \wedge c_1 \neq c_2$, where P is an atomic role and c_1 and c_2 are constants;
7. $P(c, c)$, where P is an atomic role and c is a constant;
8. $\exists x, y.U(x, y) \wedge x \neq y$, where U is an attribute and x and y are variables;
9. $\exists x.U(x, c) \wedge x \neq c$ (resp. $U(c, x) \wedge x \neq c$), where U is an attribute, x is a variable, and c is a constant;
10. $U(c_1, c_2) \wedge c_1 \neq c_2$, where U is an attribute and c_1 and c_2 are constants.
11. $\exists x, y.U(x, y) \wedge T_i(y) \wedge x \neq y$, where U is an attribute, T_i is a value-domain, and x and y are variables;
12. $\exists x.U(x, c) \wedge T_i(c) \wedge x \neq c$ (resp. $U(c, x) \wedge T_i(x) \wedge x \neq c$), where U is an attribute, T_i is a value-domain, x is a variable, and c is a constant;
13. $U(c_1, c_2) \wedge T_i(c_2) \wedge c_1 \neq c_2$, where U is an attribute, T_i is a value-domain, and c_1 and c_2 are constants.

In other words, q_{us} is a boolean conjunctive query, possibly with inequalities, in $\mathsf{minUnsatQueries}(\mathcal{T})$ in which there occurs at most one atom of the form $A(t)$, $P(t_1, t_2)$, or $U(t_1, t_2)$. Let $\mathcal{T}$ be a *DL-Lite*$_{A,id,den}$ TBox. We denote with $\mathcal{Q}_{\mathcal{T}}^{singleton}$ the subset of $\mathsf{minUnsatQueries}(\mathcal{T})$ constituted by all the queries in $\mathsf{minUnsatQueries}(\mathcal{T})$ that have the same form of the queries above.

Example 41. Let $\mathcal{T}$ be the TBox presented in Example 40. The set $cln(\mathcal{T})$ contains the following TBox assertions.

(1.) $(funct\ \mathsf{hasFather})$, (2.) $\forall x.(\mathsf{hasFather}(x, x) \rightarrow \bot)$,
(3.) $\mathsf{Man} \sqsubseteq \neg\mathsf{Woman}$, (4.) $\exists\mathsf{hasFather}^- \sqsubseteq \neg\mathsf{Woman}$,
(5.) $\mathsf{Woman} \sqsubseteq \neg\mathsf{Man}$, (6.) $\mathsf{Woman} \sqsubseteq \neg\exists\mathsf{hasFather}^-$.

By observing the assertions in $cln(\mathcal{T})$, it is easy to verify that the set $\mathcal{Q}_{\mathcal{T}}^{min}$, computed by $\mathsf{minUnsatQueries}(\mathcal{T})$, is constituted by the following queries:

$q_1 = \exists x, y, z.\mathsf{hasFather}(x, y) \wedge \mathsf{hasFather}(x, z) \wedge y \neq z \wedge x \neq y \wedge x \neq z;$

$q_2 = \exists x.\mathsf{hasFather}(x, x);$

$q_3 = \exists x.\mathsf{Man}(x) \wedge \mathsf{Woman}(x);$

$q_4 = \exists x, y.\mathsf{hasFather}(x, y) \wedge \mathsf{Woman}(y) \wedge x \neq y;$

Therefore, the set $\mathcal{Q}_{\mathcal{T}}^{singleton}$ contains only the query q_2, i.e.,

$$\mathcal{Q}_{\mathcal{T}}^{singleton} = \{ \exists x.\mathsf{hasFather}(x, x) \}$$

□

We have the following lemma.

Lemma 48. *Let $\mathcal{T}$ be a DL-Lite$_{A,id,den}$ TBox, and let $\mathcal{A}$ be an ABox. There is a $\mathcal{T}$-inconsistent singleton $\mathcal{S} \subseteq \mathcal{A}$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models \mathcal{Q}_{\mathcal{T}}^{singleton}$.*

Proof.
($\Rightarrow$) Let $\{\alpha\} \subseteq \mathcal{A}$ such that the KB $\langle \mathcal{T}, \{\alpha\} \rangle$ is inconsistent. We show that there is in $\mathcal{Q}_{\mathcal{T}}^{singleton}$ a query q such that $\langle \emptyset, \mathcal{A} \rangle \models q$. Let $\mathcal{Q}_{\mathcal{T}}^{min} = \mathsf{minUnsatQueries}(\mathcal{T})$. Since $\{\alpha\}$ is $\mathcal{T}$-inconsistent then Lemma 46 guarantees that there is in $\mathcal{Q}_{\mathcal{T}}^{min}$ a query q such that $\langle \emptyset, \{\alpha\} \rangle \models q$. Moreover, since $\{\alpha\}$ is a minimal $\mathcal{T}$-inconsistent set, then, from Lemma 47 is follows that there is no query $q' \neq q$ in $\mathcal{Q}_{\mathcal{T}}^{min}$ such that $\langle \emptyset, \{\alpha\} \rangle \models q'$. Since $\mathsf{Saturate}(q) = q$, then for every pair of different terms t_1 and t_2 in q we have in q the inequalities $t_1 \neq t_2$. Now, since $\{\alpha\} \in \mathsf{images}(q, \mathcal{A})$, then there can be only one atom in q of from $A(t)$, $P(t, t')$, or $U(t, t')$, where A, P, and U have the usual meaning. Hence, $q \in \mathcal{Q}_{\mathcal{T}}^{singleton}$, from which the claim follows.

($\Leftarrow$) Now, we show that if there exists a query $q \in \mathcal{Q}_{\mathcal{T}}^{singleton}$ such that $\langle \emptyset, \mathcal{A} \rangle \models q$, then there is in $\mathcal{A}$ an assertion α such that $\{\alpha\}$ is $\mathcal{T}$-inconsistent. Since $\mathcal{Q}_{\mathcal{T}}^{singleton} \subseteq \mathcal{Q}_{\mathcal{T}}^{min}$, then from Lemma 46 and Lemma 47 it follows that there is in $\mathcal{A}$ a minimal $\mathcal{T}$-inconsistent set. Since in q there occurs at most one atom of the form $A(t)$, $P(t_1, t_2)$, or $U(t_1, t_2)$, the every image of q in $\mathcal{A}$ is a singleton. Thus, since for every $V \in \mathsf{images}(q, \mathcal{A})$ we have that $\mathcal{A}$ is a minimal $\mathcal{T}$ inconsistent set, we can conclude that there is an assertion $\alpha \in \mathcal{A}$ such that $\{\alpha\}$ is $\mathcal{T}$-inconsistent. ∎

Now, let $\mathcal{T}$ be a *DL-Lite*$_{A,id,den}$ TBox and let α be an atom of the form $A(t)$, $P(t, t')$, or $U(t, t')$, where A is an atomic concept name, P is an atomic role name, U is an attribute name, and t and t' are terms, i.e., constants or variables. We define $IncSng^{\mathcal{T}}(\alpha)$ as follows.

$$IncSng^{\mathcal{T}}(\alpha) = \bigvee_{q \in \mathcal{Q}_{\mathcal{T}}^{singleton} \wedge CompSet(\alpha, q) \neq \emptyset} \left(\bigvee_{\beta \in CompSet(\alpha, q)} \mu_{\alpha/\beta}(q) \right)$$

The following lemma holds.

Lemma 49. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and let α be an ABox assertion. There exists in $\mathcal{A}$ a $\mathcal{T}$-inconsistent singleton $\{\alpha\}$ if and only if $\langle \emptyset, \mathcal{A} \rangle \models IncSng^{\mathcal{T}}(\alpha)$.*

Proof. The proof is easily obtainable from the proof of Theorem 37 by replacing $MinIncSet^{\mathcal{T}}(\alpha)$ with $IncSng^{\mathcal{T}}(\alpha)$ and Lemma 47 with Lemma 48. ∎

The lemma above states that, given a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, we can use $IncSng^{\mathcal{T}}(\alpha)$ in order to check if there is in $\mathcal{A}$ an assertion α such that the singleton $\{\alpha\}$ is $\mathcal{T}$-inconsistent.

In what follows, we say that a boolean query q_s is a *singleton-query* if q_s has the form $\exists\vec{x}.S(\vec{t})$, where S is an concept name, a role name, or an attribute name, $\vec{x}$ is a tuple of variables, and $\vec{t}$ is tuple of terms such that each variable in $\vec{t}$ occurs also in $\vec{x}$. A *singleton-query with inequalities* is a query of the form $\exists\vec{x}.S(\vec{t}) \wedge t_1 \neq t_2$, where S, $\vec{x}$, and $\vec{t}$ are as above, and t_1 and t_2 are terms occurring in $\vec{t}$.

As we say earlier, in order to exploit Corollary 4, we need to identify those assertions which are in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. To this aim we provide a technique for rewriting a singleton-query $\exists\vec{x}.S(\vec{t})$ into a new query q^s_{rew} such that q^s_{rew} evaluates to *true* over $DB(\mathcal{A})$ if and only if $\exists\vec{x}.S(\vec{t})$ evaluates to true over $DB(\mathsf{clc}_{\mathcal{T}}(\mathcal{A}))$.

Let $q_s = \exists\vec{x}.S(\vec{t}) \wedge t_1 \neq t_2$ be a singleton-query with inequalities. We denote by $ConsAtom(q_s, \mathcal{T})$ the following FOL-sentence:

$$\exists\vec{x}.S(\vec{t}) \wedge \neg IncSng^{\mathcal{T}}(S(\vec{t})) \wedge t_1 \neq t_2.$$

Let $\mathcal{Q}$ be a set of singleton-queries with inequalities, we define

$$ConsAtomSet(\mathcal{Q}, \mathcal{T}) = \bigvee_{q_i \in \mathcal{Q}} ConsAtom(q_s, \mathcal{T}).$$

We are now able to provide our technique for deciding if an ABox assertion belongs to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ by evaluating a singleton-query over the KB $\langle\emptyset, \mathcal{A}\rangle$.

In the lemma below we use the algorithm PerfectRef [31] for rewriting the atom $S(\vec{t})$ in the query q_s, considering all variables occurring in $S(\vec{t})$ as *bound terms*. To highlight the fact that this algorithm is used here in this particular manner, we denote it by $\mathsf{PerfectRef}_B$. The use of the algorithm Saturate, as well as for Theorem 38, is necessary for technical reason. In fact, since the queries in $\mathcal{Q}^{singleton}_{\mathcal{T}}$ are saturated with respect to inequalities, it is necessary that also q_s is saturated with respect to inequalities so as to ensure the success of the search of the set of atoms that are compatible with the atoms in q_s. In this regard, we recall that Theorem 36 guarantees that the application of the algorithm Saturate does not affect the answer of a query.

Lemma 50. *Let $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and let q_s be a singleton-query. Then $\langle\emptyset, \mathsf{clc}_{\mathcal{T}}(\mathcal{A})\rangle \models q_s$ if and only if $\langle\emptyset, \mathcal{A}\rangle \models ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T})$.*

Proof.
($\Rightarrow$) Let $q_s = \exists\vec{x}.S_n(\vec{t})$. We have to prove that if $\langle\emptyset, \mathsf{clc}_{\mathcal{T}}(\mathcal{A})\rangle \models q_s$ then $\langle\emptyset, \mathcal{A}\rangle \models ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T})$. Since $\langle\emptyset, \mathsf{clc}_{\mathcal{T}}(\mathcal{A})\rangle \models$

q_s, then there exists an ABox assertion $S_n(\vec{c})$ in $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that $S_n(\vec{c}) \in \mathsf{images}(q_s, \mathsf{clc}_\mathcal{T}(\mathcal{A}))$. This means that there is an ABox assertion $S_0(\vec{c})$ in $\mathcal{A}$ and a chain of positive inclusion assertions $\{S_0 \sqsubseteq S_1, S_1 \sqsubseteq S_2, \ldots, S_{n-1} \sqsubseteq S_n\}$ in $\mathcal{T}_{inc}$ such that $Mod(\langle \mathcal{T}, \{S_i(\vec{c})\}\rangle) \neq \emptyset$, for each $i \in \{0, \ldots, n\}$. As shown in [31, Lemma 39], if $\{S_0 \sqsubseteq S_1, S_1 \sqsubseteq S_2, \ldots S_{n-1} \sqsubseteq S_n\} \subseteq \mathcal{T}_{inc}$, then $\mathsf{PerfectRef}_B(q_s, \mathcal{T})$ returns the a set of queries $\mathcal{Q}_s^{rew}$ such that $\exists \vec{x}.S_0(\vec{t}) \in \mathcal{Q}_s^{rew}$. Since $S_0(\vec{c}) \in \mathcal{A}$, then $\langle \emptyset, \mathcal{A}\rangle \models \exists \vec{x}.S_0(\vec{t})$. Clearly, $\langle \emptyset, \mathcal{A}\rangle \models \mathsf{PerfectRef}_B(q_s, \mathcal{T})$. By applying the algorithm Saturate to the query $\exists \vec{x}.S_0(\vec{t})$, we obtain a set of singleton-queries with inequalities $\mathcal{Q}_0 = \{q_0^1, \ldots q_0^m\}$. From Theorem 36, since $\langle \emptyset, \mathcal{A}\rangle \models \exists \vec{x}.S_0(\vec{t})$, then there is a query q_0^i in $\mathcal{Q}_0$ such that $\langle \emptyset, \mathcal{A}\rangle \models q_0^i$. Thus, $S_0(\vec{c}) \in \mathsf{images}(q_0^i, \mathcal{A})$. Since $q_0^i \in \mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T}))$, then there is a query $ConsAtom(q_0^i, \mathcal{T})$ in $ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T})$. From Lemma 49, it follows that since $Mod(\langle \mathcal{T}, \{S_0(\vec{c})\}\rangle) \neq \emptyset$, then $S_0(\vec{c}) \in \mathsf{images}(ConsAtom(q_0^i, \mathcal{T}), \mathcal{A})$, i.e., $\langle \emptyset, \mathcal{A}\rangle \models ConsAtom(q_0^i, \mathcal{T})$. Finally, since $ConsAtom(q_0^i, \mathcal{T}) \in ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T})$, we can conclude that $\langle \emptyset, \mathcal{A}\rangle \models ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T})$.

($\Leftarrow$) We show that, if $\langle \emptyset, \mathcal{A}\rangle \models ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T})$, then $\langle \emptyset, \mathsf{clc}_\mathcal{T}(\mathcal{A})\rangle \models q_s$. Let $q_s = \exists \vec{x}.S_n(\vec{t})$. From Lemma 49, since $\langle \emptyset, \mathcal{A}\rangle \models ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T})$, then there are a query q_0^i, belonging to $\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T}))$, and an ABox assertion $S_0(\vec{t}) \in \mathcal{A}$, such that $S_0(\vec{t}) \in \mathsf{images}(q_0^i, \mathcal{A})$, and $Mod(\langle \mathcal{T}, \{S_0(\vec{t})\}\rangle) \neq \emptyset$. From Theorem 36, there is a query q_0 in $\mathsf{PerfectRef}_B(q_s, \mathcal{T})$ such that $q_0^i \in \mathsf{Saturate}(q_0)$, and $S_0(\vec{t}) \in \mathsf{images}(q_0, \mathcal{A})$. Since $q_0 \in \mathsf{PerfectRef}_B(q_s, \mathcal{T})$, then, from the results given in [31, Lemma 39], there is in $\mathcal{T}_{inc}$ a chain of positive inclusion assertions $\{S_0 \sqsubseteq S_1, S_1 \sqsubseteq S_2, \ldots, S_{n-1} \sqsubseteq S_n\}$. Thus, $\mathcal{T} \models S_0 \sqsubseteq S_n$. Since $Mod(\langle \mathcal{T}, \{S_0(\vec{t})\}\rangle) \neq \emptyset$, and $\mathcal{T} \models S_0 \sqsubseteq S_n$, then $S_0(\vec{t}) \in \mathsf{clc}_\mathcal{T}(\mathcal{A})$. It directly follows, that $\langle \emptyset, \mathsf{clc}_\mathcal{T}(\mathcal{A})\rangle \models \exists \vec{x}.S_n(\vec{t})$. Hence, the claim is true. ∎

Example 42. Consider the *DL-Lite*$_{A,id,den}$ TBox T presented Example 40, and consider the following boolean conjunctive query asking for the existence of a person:

$$q = \exists x.\mathsf{Person}(x).$$

The set $\mathsf{PerfectRef}_B(q, \mathcal{T})$, which coincides with the perfect reformulation of q with respect to $\mathcal{T}$ by considering the x variable as a bound terms, contains the following set of queries:

$$\begin{aligned} q_{r1} &= \exists x.\mathsf{Person}(x) \\ q_{r2} &= \exists x.\mathsf{Man}(x) \\ q_{r3} &= \exists x.\mathsf{Woman}(x) \\ q_{r4} &= \exists x, y.\mathsf{hasFather}(x, y) \end{aligned}$$

By applying the algorithm Saturate we get:

$$\begin{aligned} q^1_{r1} &= \exists x.\mathsf{Person}(x) \\ q^1_{r2} &= \exists x.\mathsf{Man}(x) \\ q^1_{r3} &= \exists x.\mathsf{Woman}(x) \\ q^1_{r4} &= \exists x, y.\mathsf{hasFather}(x, y) \wedge x \neq y \\ q^2_{r4} &= \exists x.\mathsf{hasFather}(x, x) \end{aligned}$$

As shown in Example 41 we have that

$$\mathcal{Q}^{singleton}_{\mathcal{T}} = \{ \exists x.\mathsf{hasFather}(x, x) \}$$

Hence, we have that $ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q, \mathcal{T})), \mathcal{T})$ contains the following FOL-queries:

$$\begin{aligned} q^1_{con1} &= \exists x.\mathsf{Person}(x) \\ q^1_{con2} &= \exists x.\mathsf{Man}(x) \\ q^1_{con3} &= \exists x.\mathsf{Woman}(x) \\ q^1_{con4} &= \exists x, y.\mathsf{hasFather}(x, y) \wedge x \neq y \\ q^2_{con4} &= \exists x.\mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \end{aligned}$$

Now, let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the following ABoxes.

$$\begin{aligned} \mathcal{A}_1 &= \{ \mathsf{hasFather}(sam, bill) \} \\ \mathcal{A}_2 &= \{ \mathsf{hasFather}(tom, tom) \}. \end{aligned}$$

Respectively, we have:

$$\begin{aligned} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_1) &= \{ \mathsf{hasFather}(sam, bill), \mathsf{Man}(bill), \mathsf{Person}(bill) \} \\ \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_2) &= \emptyset. \end{aligned}$$

It is easy to verify that:

$$\langle \emptyset, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_1) \rangle \models q; \qquad \langle \emptyset, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_2) \rangle \not\models q;$$

and that

$$\begin{aligned} &\langle \emptyset, \mathcal{A}_1 \rangle \models ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T}); \\ &\langle \emptyset, \mathcal{A}_2 \rangle \not\models ConsAtomSet(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T}). \end{aligned}$$

□

Let $\mathcal{T}$ be a *DL-Lite*$_{A,id,den}$ TBox and let q be a BCQ with inequalities of the form

$$\exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} A_i(t_i^1) \wedge \bigwedge_{i=1}^{m} P_i(t_i^2, t_i^3) \wedge \bigwedge_{i=1}^{\ell} U_i(t_i^4, t_i^5) \wedge \bigwedge_{i=1}^{h} t_i^6 \neq t_i^7$$

where every A_i is an atomic concept, every P_i is an atomic role, every U_i is an attribute, every t_i^e is a term (i.e., either a constant or a variable), and $z_1, \ldots, z_k$ are all the variables appearing in q. We denote by $\mathit{IncRewr}_{ICAR}(q, \mathcal{T})$ the following FOL-sentence:

$$\begin{aligned}
\exists z_1, \ldots, z_k. \quad & \textstyle\bigwedge_{i=1}^{n} \; \mathit{ConsAtomSet}(\mathsf{Saturate}(\mathsf{PerfectRef}_B(A_i(t_i^1), \mathcal{T})), \mathcal{T}) \\
& \wedge \quad \neg \mathit{MinIncSet}^{\mathcal{T}}(A_i(t_i^1)) \qquad \wedge \\
& \textstyle\bigwedge_{i=1}^{m} \; \mathit{ConsAtomSet}(\mathsf{Saturate}(\mathsf{PerfectRef}_B(P_i(t_i^2, t_i^3), \mathcal{T})), \mathcal{T}) \\
& \wedge \quad \neg \mathit{MinIncSet}^{\mathcal{T}}(P_i(t_i^2, t_i^3)) \qquad \wedge \\
& \textstyle\bigwedge_{i=1}^{\ell} \; \mathit{ConsAtomSet}(\mathsf{Saturate}(\mathsf{PerfectRef}_B(U_i(t_i^4, t_i^5), \mathcal{T})), \mathcal{T}) \\
& \wedge \quad \neg \mathit{MinIncSet}^{\mathcal{T}}(U_i(t_i^4, t_i^5)) \qquad \wedge \\
& \textstyle\bigwedge_{i=1}^{h} \; t_i^6 \neq t_i^7.
\end{aligned}$$

Moreover, let $\mathcal{Q}$ be a set of BCQs with inequalities, we define

$$\mathit{IncRewrUCQ}_{ICAR}(\mathcal{Q}, \mathcal{T}) = \bigvee_{q_i \in \mathcal{Q}} \mathit{IncRewr}_{ICAR}(q_i, \mathcal{T}).$$

The following theorem constitutes the main result of this section, and is our final results on reformulation of boolean UCQs under the *ICAR*-semantics.

Theorem 40. *Let $\mathcal{T}$ be a DL-Lite$_{A,id,den}$ TBox and let $\mathcal{Q}$ be a UCQ. For every ABox $\mathcal{A}$, we have that $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} \mathcal{Q}$ if and only if*

$$\langle \emptyset, \mathcal{A} \rangle \models \mathit{IncRewrUCQ}_{ICAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T}).$$

Proof.
($\Rightarrow$) Let $\mathcal{T}$ be a *DL-Lite*$_{A,id,den}$ TBox, $\mathcal{Q}$ be a UCQ, and $\mathcal{A}$ be an ABox. We have to prove that if $\mathcal{Q}$ is *ICAR*-entailed by the KB $\langle \mathcal{T}, \mathcal{A} \rangle$, then $\langle \emptyset, \mathcal{A} \rangle \models \mathit{IncRewrUCQ}_{ICAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T})$. Since $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} \mathcal{Q}$, then there is a query $q \in \mathcal{Q}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} q$. From Corollary 4, there is a set of ABox assertions $\mathcal{A}' = \{S_1(\vec{c_1}), \ldots, S_n(\vec{c_n})\}$ such that (i) $\mathcal{A}' \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, (ii) $\mathcal{A}'$ is $\mathcal{T}$-consistent, (iii) $\langle \mathcal{T}, \mathcal{A}' \rangle \models q$, and (iv) for every minimal $\mathcal{T}$-inconsistent set $V \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, $\mathcal{A}' \cap V = \emptyset$. From Lemma 30, since $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} q$, then $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \rangle \models_{IAR} q$. From Theorem 38 it follows that $\langle \emptyset, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \rangle \models \mathit{IncRewrUCQ}_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(q, \mathcal{T}_{inc})), \mathcal{T})$. Therefore, there is a query

$q^{str}_{rew} = \exists\vec{z}.S_1(\vec{t_1}) \wedge \ldots \wedge S_n(\vec{t_n}) \wedge \bigwedge_{i=1}^{h} t'_i \neq t''_i$ in $\mathsf{Saturate}(\mathsf{PerfectRef}(q, \mathcal{T}_{inc}))$ such that the FOL-query

$$q_{clc} = \exists\vec{z}. \bigwedge_{i=1}^{n} S_i(\vec{t_i}) \wedge \neg MinIncSet^{\mathcal{T}}(S_i(\vec{t_i})) \wedge \bigwedge_{i=1}^{h} t'_i \neq t''_i$$

evaluates to *true* over $DB(\mathsf{clc}_{\mathcal{T}}(\mathcal{A}))$. More specifically, q_{clc} evaluates to *true* over $DB(\mathcal{A}')$. Since $\mathcal{A}' \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, there is a subset $\mathcal{A}'_0 = \{S^0_1(\vec{c_1}), \ldots, S^0_n(\vec{c_n})\}$ of $\mathcal{A}$, such that for every assertion $S^0_i(\vec{c_i}) \in \mathcal{A}'_0$, $Mod(\langle\mathcal{T}, \{S^0_i(\vec{c_i})\}\rangle) \neq \emptyset$, and $\mathcal{T}_{inc} \models S^0_i \sqsubseteq S_i$, with $1 \leq i \leq n$. From Lemma 50 it follows that $\langle\emptyset, \mathcal{A}\rangle \models ConsAtomSet(S_i(\vec{t_i}), \mathcal{T})$, with $1 \leq i \leq n$. Hence,

$$\langle\emptyset, \mathcal{A}\rangle \models \exists\vec{z}. \bigwedge_{i=1}^{n} ConsAtomSet(S_i(\vec{t_i}), \mathcal{T}) \wedge \neg MinIncSet^{\mathcal{T}}(S_i(\vec{t_i})) \wedge \bigwedge_{i=1}^{h} t'_i \neq t''_i.$$

Since the query above is in $IncRewrUCQ_{ICAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T})$, then $\langle\emptyset, \mathcal{A}\rangle \models IncRewrUCQ_{ICAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T})$.

($\Leftarrow$) Let $\mathcal{Q}^{ICAR}_{rew} = IncRewrUCQ_{ICAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T})$. We have to prove that, if $\langle\emptyset, \mathcal{A}\rangle \models \mathcal{Q}^{ICAR}_{rew}$, then $\langle\mathcal{T}, \mathcal{A}\rangle \models_{ICAR} \mathcal{Q}$. Since $\langle\emptyset, \mathcal{A}\rangle \models \mathcal{Q}^{ICAR}_{rew}$, then there are a query $q^{\mathcal{A}}$ in $\mathcal{Q}^{ICAR}_{rew}$ of the form

$$\exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} (S^{\mathcal{A}}_i(\vec{t_i}) \ \wedge \neg IncSng^{\mathcal{T}}(S^{\mathcal{A}}_i(\vec{t_i}))) \ \wedge \bigwedge_{i=1}^{h} t'_i \neq t''_i,$$

and a set of ABox assertions $\mathcal{A}' = \{S^{\mathcal{A}}_1(\vec{c_1}), \ldots, S^{\mathcal{A}}_n(\vec{c_n})\}$, such that $\mathcal{A}' \in \mathsf{images}(q^{\mathcal{A}}, \mathcal{A})$. Since each pair $(S^{\mathcal{A}}_i(\vec{t_i}) \wedge \neg IncSng^{\mathcal{T}}(S^{\mathcal{A}}_i(\vec{t_i})))$ occurring in $q^{\mathcal{A}}$, with $1 \leq i \leq n$, belongs to $ConsAtomSet(S^{clc}_i(t_i), \mathcal{T})$, from Lemma 50, it follows that $Mod(\langle\mathcal{T}, \{S^{\mathcal{A}}_i(\vec{c_i})\}\rangle) \neq \emptyset$, with $1 \leq i \leq n$, and that there is a set $\mathcal{A}^{clc} = \{S^{clc}_1(\vec{c_1}), \ldots, S^{clc}_n(\vec{c_n})\}$ such that $\mathcal{A}^{clc} \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, and $\mathcal{A}^{clc} \in \mathsf{images}(q^{clc}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}))$, where

$$q^{clc} = \exists z_1, \ldots, z_k. \bigwedge_{i=1}^{n} S^{clc}_i(t_i) \ \wedge \neg MinIncSet^{\mathcal{T}}(S^{clc}_i(t_i)) \ \wedge \bigwedge_{i=1}^{h} t'_i \neq t''_i.$$

Hence, there exists a query $q \in \mathcal{Q}$ such that

$$q^{clc} \in IncRewrUCQ_{IAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(q, \mathcal{T}_{inc})), \mathcal{T}).$$

Since $\langle\emptyset, \mathsf{clc}_{\mathcal{T}}(\mathcal{A})\rangle \models q$, then, from Theorem 38, $\langle\mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A})\rangle \models_{IAR} q$. From Lemma 30, it follows that $\langle\mathcal{T}, \mathcal{A}\rangle \models_{ICAR} q$. Finally, since $q \in \mathcal{Q}$, we can conclude that $\langle\mathcal{T}, \mathcal{A}\rangle \models_{ICAR} \mathcal{Q}$. ∎

Example 43. Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be the *DL-Lite*$_{A,id,den}$-KB of Example 40, and let q be the following boolean conjunctive query:

$$q = \exists x, y.\mathsf{Person}(x) \wedge \mathsf{hasFather}(x, y).$$

In words, the query q asks for the existence of a person with a father.

Since the set *CAR-Set*$(\langle \mathcal{T}, \mathcal{A} \rangle)$ is constituted by the following $\mathcal{T}$-consistent ABoxes:

$$\begin{aligned} \textit{CA-rep}_1 = \{\ & \mathsf{Man}(sam), \mathsf{Person}(sam), \mathsf{hasParent}(sam, bill), \\ & \mathsf{Man}(bill), \mathsf{Person}(bill)\ \}; \\ \textit{CA-rep}_2 = \{\ & \mathsf{Woman}(sam), \mathsf{Person}(sam), \mathsf{hasParent}(sam, bill), \\ & \mathsf{Man}(bill), \mathsf{Person}(bill)\ \}, \end{aligned}$$

we have that $\langle \mathcal{T}, \mathcal{A} \rangle \models_{ICAR} q$.

As shown in Example 41, $\mathcal{Q}_{\mathcal{T}}^{min}$ is constituted by the following queries:

$$\begin{aligned} q_1 &= \exists x, y, z.\mathsf{hasFather}(x, y) \wedge \mathsf{hasFather}(x, z) \wedge y \neq z \wedge x \neq y \wedge x \neq z; \\ q_2 &= \exists x.\mathsf{hasFather}(x, x); \\ q_3 &= \exists x.\mathsf{Man}(x) \wedge \mathsf{Woman}(x); \\ q_4 &= \exists x, y.\mathsf{hasFather}(x, y) \wedge \mathsf{Woman}(y) \wedge x \neq y; \end{aligned}$$

and $\mathcal{Q}_{\mathcal{T}}^{singleton} = \{\ \exists x.\mathsf{hasFather}(x, x)\ \}$.

The *CAR*-perfect reformulation of the query q with respect to $\mathcal{T}$ is partially constituted as follows:

$$\begin{aligned} q_{r1} &= \exists x, y, k.\mathsf{Person}(x) \wedge \mathsf{hasFather}(x, y) \wedge \neg\mathsf{hasFather}(x, k) \\ &\quad \wedge\neg\mathsf{Woman}(y) \wedge x \neq y \wedge y \neq k; \\ q_{r2} &= \exists x, y, k.\mathsf{Man}(x) \wedge \mathsf{hasFather}(x, y) \wedge \neg\mathsf{hasFather}(x, k) \\ &\quad \wedge\neg\mathsf{Woman}(y) \wedge x \neq y \wedge y \neq k; \\ q_{r3} &= \exists x, y, k.\mathsf{Woman}(x) \wedge \mathsf{hasFather}(x, y) \wedge \neg\mathsf{hasFather}(x, k) \\ &\quad \wedge\neg\mathsf{Woman}(y) \wedge x \neq y \wedge y \neq k; \\ q_{r4} &= \exists x, y, z, k.\mathsf{hasFather}(z, x) \wedge \mathsf{hasFather}(x, y) \wedge \neg\mathsf{hasFather}(x, k) \\ &\quad \wedge\neg\mathsf{Woman}(y) \wedge x \neq y \wedge y \neq k \wedge x \neq z; \\ q_{r5} &= \exists x, y, k.\mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \wedge \mathsf{hasFather}(x, y) \\ &\quad \wedge\neg\mathsf{hasFather}(x, k) \wedge \neg\mathsf{Woman}(y) \wedge x \neq y \wedge y \neq k; \\ q_{r6} &= \exists x.\mathsf{Person}(x) \wedge \mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \wedge \neg\mathsf{Woman}(x); \\ q_{r7} &= \exists x.\mathsf{Man}(x) \wedge \mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \wedge \neg\mathsf{Woman}(x); \\ q_{r8} &= \exists x.\mathsf{Woman}(x) \wedge \mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \wedge \neg\mathsf{Woman}(x); \end{aligned}$$

$$
\begin{aligned}
q_{r9} &= \exists x, z.\mathsf{hasFather}(z, x) \wedge \mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \\
&\quad \wedge\neg\mathsf{Woman}(x) \wedge x \neq z; \\
q_{r10} &= \exists x.\mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \wedge \neg\mathsf{Woman}(x); \\
q_{r11} &= \exists x, y.\mathsf{Man}(x) \wedge \mathsf{hasFather}(x, y) \wedge x \neq y; \\
q_{r12} &= \exists x, y, k.\mathsf{Man}(x) \wedge \mathsf{hasFather}(x, y) \wedge \neg\mathsf{hasFather}(x, k) \\
&\quad \wedge\neg\mathsf{Woman}(y) \wedge \neg\mathsf{Woman}(x) \wedge x \neq y \wedge y \neq k; \\
q_{r13} &= \exists x, y, k, z.\mathsf{hasFather}(z, x) \wedge \mathsf{hasFather}(x, y) \wedge \neg\mathsf{hasFather}(x, k) \\
&\quad \wedge\neg\mathsf{Woman}(y) \wedge \neg\mathsf{Woman}(x) \wedge x \neq y \wedge y \neq k \wedge x \neq z; \\
q_{r14} &= \exists x, y, k.\mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \wedge \mathsf{hasFather}(x, y) \\
&\quad \wedge\neg\mathsf{hasFather}(x, k) \wedge \neg\mathsf{Woman}(y) \wedge \neg\mathsf{Woman}(x) \wedge x \neq y \wedge y \neq k; \\
q_{r15} &= \exists x.\mathsf{Man}(x) \wedge \mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \wedge \neg\mathsf{Woman}(x); \\
q_{r16} &= \exists x, z.\mathsf{hasFather}(z, x) \wedge \mathsf{hasFather}(x, x) \wedge \neg\mathsf{hasFather}(x, x) \\
&\quad \wedge\neg\mathsf{Woman}(x) \wedge x \neq z; \\
q_{r17} &= \exists x, y.\mathsf{Woman}(x) \wedge \mathsf{hasFather}(x, y) \wedge \neg\mathsf{Man}(x) \wedge \mathsf{hasFather}(x, k) \\
&\quad \wedge\neg\mathsf{Woman}(y) \wedge x \neq y \wedge y \neq k; \\
q_{r18} &= \exists x.\mathsf{Woman}(x) \wedge \mathsf{hasFather}(x, x) \wedge \neg\mathsf{Man}(x) \wedge \neg\mathsf{hasFather}(x, x) \\
&\quad \wedge\neg\mathsf{Woman}(x); \\
&\quad \vdots \\
&\quad \vdots \\
q_{rnn} &= \exists x, y, k, z.\mathsf{hasFather}(y, x) \wedge \mathsf{hasFather}(x, y) \wedge \neg\mathsf{hasFather}(y, z) \wedge y \neq z \\
&\quad \wedge\neg\mathsf{hasFather}(x, k) \wedge \neg\mathsf{Woman}(x) \wedge \neg\mathsf{Woman}(y) \wedge x \neq y \wedge x \neq k;
\end{aligned}
$$

As one can verify, both the queries q_{r2} and q_{r3} evaluates to *true* over $DB(\mathcal{A})$. It follows that $\langle\emptyset, \mathcal{A}\rangle \models \mathit{IncRewrUCQ}_{ICAR}(\mathsf{Saturate}(\mathsf{PerfectRef}(\mathcal{Q}, \mathcal{T}_{inc})), \mathcal{T})$ as expected. □

Theorem 40 allows us to give a complete complexity characterization of the problem of answering a boolean UCQ over a possibly inconsistent *DL-Lite*$_{A,id,den}$-KB under *ICAR*-semantics.

The following result is a direct consequence of Theorem 40.

Corollary 6. *Let $\mathcal{K}$ be a DL-Lite$_{A,id,den}$-KB and let $\mathcal{Q}$ be a UCQ. Deciding whether $\mathcal{K} \models_{ICAR} \mathcal{Q}$ is in AC^0 in data complexity.*

Proof. The proof directly follows from Theorem 40 and from the complexity of evaluation of FOL-queries over relational databases in the size of the data [1]. ■

Moreover, from the results given in the previous section and from Corollary 6, it follows the next theorem.

Theorem 41. *Let $\mathcal{K}$ be a DL-Lite$_{A,id,den}$-KB and let $\mathcal{Q}$ be a UCQ. Deciding whether $\mathcal{K} \models_{ICAR} \mathcal{Q}$ can be done in AC^0 with respect to $|\mathcal{A}|$, and in exponential time with respect to $|\mathcal{T}|$ and $|\mathcal{Q}|$.*

Proof. The proof can be easily adapted from the proof of Theorem 39 by observing that $\mathcal{Q}_{\mathcal{T}}^{singleton}$ can be computed in polynomial time with respect to $\mathcal{Q}_{\mathcal{T}}^{min}$, and that, from Lemma 4 and from the results given in the proof of Lemma 41, it follows that $ConsAtom(\mathsf{Saturate}(\mathsf{PerfectRef}_B(q_s, \mathcal{T})), \mathcal{T})$ can be computed in exponential time with respect to both $|\mathcal{T}|$ and $|q_s|$ for each query q_s. ∎

Since instance checking can be reduced to a particular form of query answering, the above results shows that instance checking under *ICAR*-semantics is in AC^0. This result, together with Lemma 29, allows us to give the following notable property of the *CAR*-semantics.

Theorem 42. *Let $\mathcal{K}$ be a DL-Lite$_{A,id,den}$-KB and let α be an ABox assertion. Deciding whether $\mathcal{K} \models_{CAR} \alpha$ is AC^0 with respect to data complexity.*

Proof. The proof follows directly from Corollary 6, and from Lemma 29, which states that instance checking under *CAR*-semantics coincides with instance checking under *ICAR*-semantics. ∎

Part V

Inconsistency-tolerant Update

Chapter 10

Updating inconsistent Description Logic KBs

Inconsistency is ubiquitous in many KBs used in real world applications. In particular, in the context of Ontology-based Data Access [26, 95], KBs affected by ABox inconsistency are very common. This kind of inconsistency may arise since facts in the ABox contradict assertions belonging to a consistent TBox.

As we said in the introduction of the thesis, we aim to equip an Ontology-based Information Systems with suitable mechanisms for inconsistency tolerance for addressing the following issues:

(*i*) How to answer queries that are posed to an inconsistent KB (inconsistency-tolerant query answering);

(*ii*) How to compute the KB resulting from updating a possibly inconsistent KB with both the insertion and deletion of assertions (inconsistent-tolerant update).

Motivated by the first mentioned requirement, we have proposed in Chapter 8 various *inconsistency-tolerant semantics*, inspired by the studies on inconsistency handling in belief revision [49] and by the work on consistent query answering in databases [36].

In this chapter, we focus on the second mentioned requirement, i.e., how to compute the KB resulting from updating a possibly inconsistent KB.

Inconsistency-tolerant update in the context of DL KBs is a new problem. Indeed, at the best of our knowledge, the approaches for updating DL KBs, proposed in literature, assume that the evolving KB is consistent, and the update itself preserves consistency. As explained in Chapter 6, updating a consistent KB means to modify it in order to adhere to a change in the domain of interest by preserving the consistency of the changed KB. The modification may concern either the insertion or the deletion of assertions.

In Section 6.4, we propose a semantics for updating consistent DL KBs in which the formula-based approach [42] is adopted. Essentially, in this update semantics, the result of the update is defined in term of formulae, by changing the original KB as little as possible. As we already noted in Section 6.3, the formula constituting the result of an evolution operation is not unique in general, and this may lead to expressibility problems. For addressing this problem, we propose, in Section 6.4, to follow the *When In Doubt Throw It Out* (WIDTIO) [110] principle.

On the base of the update semantics proposed in Section 6.4, and on the inconsistency-tolerant semantics presented in Section 8.2, specifically on the *CAR*-semantics, we present, in this chapter a semantics for updating inconsistent DL KBs with both the insertion and the deletion of a set of ABox assertions. Our update mechanism is based on the idea that realizing an insertion into (resp., deletion from) a KB $\mathcal{K}$ of a set F of ABox assertions means computing the (possibly inconsistent) KB $\mathcal{K}'$ that minimally differ from $\mathcal{K}$ and such that all the repairs of $\mathcal{K}'$ entail (resp., do not entail) F. Based on the WIDTIO principle, the result of the update is the intersection of all the KBs realizing the update.

10.1 Inconsistency-tolerant update semantics

In this section we present our semantics for updating possibly inconsistent DL KBs with both the insertion and the deletion of a finite set of ABox assertions. In what follows, $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a possibly inconsistent $\mathcal{L}$-KB, with $\mathcal{T}$ satisfiable. Moreover, F denotes a finite set of $\mathcal{T}$-consistent ABox assertions. In the rest of this work, we use the term "update" as a generalization of "update by insertion" and "update by deletion".

We follow the idea presented in Section 6.4 for updating consistent KBs, that we now briefly recall.

Firstly, we need to recall the notion of "few changes" introduced in [44]. Let $\mathcal{A}$, $\mathcal{A}'$, and $\mathcal{A}''$ be three finite set of ABox assertions. We say that $\mathcal{A}'$ has fewer deletions than $\mathcal{A}''$ with respect to $\mathcal{A}$ if $\mathcal{A} \setminus \mathcal{A}' \subset \mathcal{A} \setminus \mathcal{A}''$. Also, we say that $\mathcal{A}'$ and $\mathcal{A}''$ have the same deletions with respect to $\mathcal{A}$ if $\mathcal{A} \setminus \mathcal{A}' = \mathcal{A} \setminus \mathcal{A}''$, and that $\mathcal{A}'$ has fewer insertions than $\mathcal{A}''$ with respect to $\mathcal{A}$ if $\mathcal{A}' \setminus \mathcal{A} \subset \mathcal{A}'' \setminus \mathcal{A}$. Finally, we say $\mathcal{A}_1$ has *fewer changes* (cf. Definition 18) than $\mathcal{A}_2$ with respect to $\mathcal{A}$ if

(i) $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_2$ with respect to $\mathcal{A}$, or
(ii) $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same deletions with respect to $\mathcal{A}$, and $\mathcal{A}_1$ has fewer insertions than $\mathcal{A}_2$ with respect to $\mathcal{A}$.

Now, suppose that $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is consistent, and we want to update $\mathcal{K}$ with either the insertion or the deletion of F. Essentially, in the case of insertion, the result of the update is the KB formed by $\mathcal{T}$ and the intersection of all the ABoxes accomplishing the insertion of F into $\mathcal{K}$ minimally. Similarly, the result of updating $\mathcal{K}$ with the deletion of F is the KB formed by $\mathcal{T}$ and the intersection of all the ABoxes accomplishing the deletion of F from $\mathcal{K}$ minimally. An ABox $\mathcal{A}'$ accomplishes the insertion (resp., deletion) of F minimally if $\mathcal{A}'$ is $\mathcal{T}$-consistent, $\langle \mathcal{T}, \mathcal{A}' \rangle$ logically entails F (resp., does not logically entail F), and no $\mathcal{T}$-consistent ABox $\mathcal{A}''$ exists that logically entails F (resp., does not logically entail F) with $\mathcal{T}$, and such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$ has fewer changes than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$.

Note that the above described update semantics makes use of the notion of logical entailment. In case where $\mathcal{K}$ is not consistent, the reasoning becomes meaningless, since any conclusion can be inferred from an inconsistent $\mathcal{K}$. So, how can we apply this idea in the case where $\mathcal{K}$ is inconsistent?

Generally speaking, in our solution, we make use of the notion of *closed ABox repair* (*CA*-repair) of an inconsistent KB given in Section 8.1.

We recall that, given a possibly inconsistent $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ denotes the *consistent logical consequences of* $\mathcal{A}$ *with respect to* $\mathcal{T}$, that corresponds to the set $\{\alpha \mid \alpha \in \mathit{HB}(\mathcal{K})$ and there exists $S \subseteq \mathcal{A}$ such that $\mathit{Mod}(\langle \mathcal{T}, S \rangle) \neq$

$\emptyset$ and $\langle \mathcal{T}, S \rangle \models \alpha\}$, where $HB(\mathcal{K})$ denotes the *Herbrand Base of* $\mathcal{K}$. Moreover, we say that two KBs $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}, \mathcal{A}' \rangle$ are *consistently equivalent* (*C-equivalent*) if $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$.

A *CA*-repair for a possibly inconsistent $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a $\mathcal{T}$-consistent ABox $\mathcal{A}'$ such that there does not exist a $\mathcal{T}$-consistent ABox $\mathcal{A}''$ such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}'')$ has fewer changes than $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})'$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. We denote with *CAR-Set*($\mathcal{K}$) the set containing the closed ABox repairs for $\mathcal{K}$.

We now present a simple example illustrating the notion of *CA*-repair.

Example 44. The following TBox $\mathcal{T}$ is a portion of a knowledge base describing the domain of rowing competitions.

$$
\begin{array}{lll}
OA \sqsubseteq ATH, & CX \sqsubseteq ATH, & CX \sqsubseteq \neg OA, \\
ATH \sqsubseteq \exists mf, & CH \sqsubseteq \exists mf, & CH \sqsubseteq \neg ATH, \\
\exists mf^{-} \sqsubseteq RTM, & CR \sqsubseteq \exists fb, & \exists fb \sqsubseteq CR, \\
\exists fb^{-} \sqsubseteq ATH, & \exists mf \sqsubseteq ATH \sqcup CH, & (funct\ fb^{-}), \\
(funct\ mf), & (id\ CX\ fb^{-}). &
\end{array}
$$

The axioms state that oars (OA) and coxs (CX) are both athletes (ATH), and oars are not coxs. Every athlete is member of (mf) exactly one rowing team (RTM). A crew (CR) is formed by (fb) athletes, among which there is exactly one cox. Moreover, a coach (CH) is a member of exactly one rowing team, and is not an athlete. Finally, those who are members of a rowing team are either athletes or coaches.

Let us consider the ABox $\mathcal{A}$ containing the following assertions:

$$
\begin{array}{lll}
CX(c_1), & mf(c_1, t_1), & fb(w, c_1), \\
CX(c_2), & mf(c_2, t_1), & fb(w, c_2), \\
CH(h), & mf(h, t_2), & OA(h).
\end{array}
$$

In words, $\mathcal{A}$ specifies that both c_1 and c_2 are coaxes that are members of the rowing team t_1, and that the crew w is formed by c_1 and c_2. Moreover, $\mathcal{A}$ specifies that h is a member of the rowing team t_2, and that h is both a coach and an oar.

It is easy to see that the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent, since the crew w has two coxs, namely c_1 and c_2, and h is both a coach and an athlete.

Up to logical equivalence, the set *CAR-Set*($\mathcal{K}$) is constituted by the following $\mathcal{T}$-consistent ABoxes:

$$
\begin{array}{l}
\textit{CA-rep}_1 = \{\ CX(c_1), mf(c_1, t_1), CX(c_2), mf(c_2, t_1), fb(w, c_1), \\
\qquad\qquad mf(h, t_2), CH(h)\ \}; \\
\textit{CA-rep}_2 = \{\ CX(c_1), mf(c_1, t_1), CX(c_2), mf(c_2, t_1), fb(w, c_1), \\
\qquad\qquad mf(h, t_2), OA(h)\ \}; \\
\textit{CA-rep}_3 = \{\ CX(c_1), mf(c_1, t_1), CX(c_2), mf(c_2, t_1), fb(w, c_2),
\end{array}
$$

$$
\begin{aligned}
&\quad mf(h,t_2), CH(h)\ \};\\
\textit{CA-rep}_4 = \{\ &CX(c_1), mf(c_1,t_1), CX(c_2), mf(c_2,t_1), fb(w,c_2),\\
&mf(h,t_2), OA(h)\ \};\\
\textit{CA-rep}_5 = \{\ &ATH(c_1), mf(c_1,t_1), CX(c_2), mf(c_2,t_1), fb(w,c_1),\\
&fb(w,c_2), mf(h,t_2), CH(h)\ \};\\
\textit{CA-rep}_6 = \{\ &ATH(c_1), mf(c_1,t_1), CX(c_2), mf(c_2,t_1), fb(w,c_1),\\
&fb(w,c_2), mf(h,t_2), OA(h)\ \};\\
\textit{CA-rep}_7 = \{\ &CX(c_1), mf(c_1,t_1), ATH(c_2), mf(c_2,t_1), fb(w,c_1),\\
&fb(w,c_2), mf(h,t_2), CH(h)\ \};\\
\textit{CA-rep}_8 = \{\ &CX(c_1), mf(c_1,t_1), ATH(c_2), mf(c_2,t_1), fb(w,c_1),\\
&fb(w,c_2), mf(h,t_2), OA(h)\ \}.
\end{aligned}
$$

□

Our solution for updating inconsistent KBs is based on a simple modification of the notions of "accomplishing the insertion" and "accomplishing the deletion" given in Section 6.2 in case of consistent KBs.

We sanction that an ABox $\mathcal{A}'$ accomplishes the insertion of F into a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if, for all *CA*-repairs $\mathcal{A}''$ of $\langle \mathcal{T}, \mathcal{A}' \rangle$, we have that $\langle \mathcal{T}, \mathcal{A}'' \rangle$ logically entails F. Similarly, we say that an ABox $\mathcal{A}'$ accomplishes the deletion of F from a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if, for all *CA*-repairs $\mathcal{A}''$ of $\langle \mathcal{T}, \mathcal{A}' \rangle$, we have that $\langle \mathcal{T}, \mathcal{A}'' \rangle$ does not logically entail F. More formally.

Definition 36. An ABox $\mathcal{A}'$ *accomplishes the insertion* of F into $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if for all $\mathcal{A}'' \in \textit{CAR-Set}(\langle \mathcal{T}, \mathcal{A}' \rangle)$, we have that $\langle \mathcal{T}, \mathcal{A}'' \rangle \models F$.

Intuitively, the above definition states that an ABox $\mathcal{A}'$ accomplishes the insertion of F into a possibly inconsistent DL KB $\mathcal{K}$ if the KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ implies under *CAR*-semantics all the assertions in F.

In the following definition we formally describe when an ABox accomplishes the deletion of a set of ABox assertions from a possibly inconsistent KB.

Definition 37. An ABox $\mathcal{A}'$ *accomplishes the deletion* of F from $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if for all $\mathcal{A}'' \in \textit{CAR-Set}(\langle \mathcal{T}, \mathcal{A}' \rangle)$, we have that $\langle \mathcal{T}, \mathcal{A}'' \rangle \not\models F$.

Observe that, in the case of deletion we do not sanction that an ABox $\mathcal{A}'$ accomplishes the deletion of F from $\mathcal{K}$ if the KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ does not entail at least one of the assertions in F. Put in other terms, it is not sufficient that $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models_{CAR} F$. Indeed, we impose that every *CAR*-repair of the ABox $\mathcal{A}'$ does not entail at least one of the assertions in F. Clearly, the latter implies the former, i.e., if $\mathcal{A}'$ accomplishes the deletion of F from $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, then $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models_{CAR} F$.

The following proposition establishes the condition under which an ABox accomplishing the insertion of a finite set of ABox assertions F into a possibly inconsistent $\mathcal{L}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ always exists. We remind the reader that in our study we consider only monotonic languages and that we are assuming that the TBox $\mathcal{T}$ is consistent.

Proposition 19. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in $\mathcal{L}$, and let F be a set of ABox assertions. An ABox $\mathcal{A}'$ accomplishing the insertion of F into $\mathcal{K}$ exists if and only if $Mod(\langle \mathcal{T}, F \rangle) \neq \emptyset$.*

Proof.
($\Rightarrow$) Let $\mathcal{A}'$ be an ABox accomplishing the insertion of F into $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. This means that, for each *CAR*-repair $\mathcal{A}''$ of $\langle \mathcal{T}, \mathcal{A}' \rangle$, we have that $\langle \mathcal{T}, \mathcal{A}'' \rangle \models F$. Since $\mathcal{A}''$ is $\mathcal{T}$-consistent, then also F is $\mathcal{T}$-consistent.
($\Leftarrow$) Let F be a $\mathcal{T}$-consistent set of ABox assertions. This means that the KB $\langle \mathcal{T}, F \rangle$ is consistent and therefore F is a *CAR*-repair for $\langle \mathcal{T}, F \rangle$. Clearly, $\langle \mathcal{T}, F \rangle \models F$. Hence, we can conclude that F accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$. ∎

The analogous for deletion is the following.

Proposition 20. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in $\mathcal{L}$, and let F be a set of ABox assertions. An ABox $\mathcal{A}'$ accomplishing the deletion of F from $\mathcal{K}$ exists if and only if $\langle \mathcal{T}, \emptyset \rangle \not\models F$.*

Proof.
($\Rightarrow$) Let $\mathcal{A}'$ be an ABox accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, we prove that $\langle \mathcal{T}, \emptyset \rangle \not\models F$. Suppose, by way of contradiction, that $\langle \mathcal{T}, \emptyset \rangle \models F$. Since $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, we have that each *CAR*-repair $\mathcal{A}''$ of $\langle \mathcal{T}, \mathcal{A}' \rangle$ does not model F. Hence, also $\langle \mathcal{T}, \emptyset \rangle$ does not model F.
($\Leftarrow$) Suppose that $\langle \mathcal{T}, \emptyset \rangle \not\models F$. From the fact that $\emptyset$ is $\mathcal{T}$-consistent, we conclude that $\emptyset$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. ∎

Moreover, we have the following lemmas.

Lemma 51. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB in $\mathcal{L}$, and let F be a $\mathcal{T}$-inconsistent set of ABox assertions. Then, $\mathcal{A}$ accomplishes the deletion of F from $\mathcal{K}$.*

Proof. Since every set $\mathcal{A}_{rep} \in$ *CAR-Set*$\langle \mathcal{T}, \mathcal{A} \rangle$ is $\mathcal{T}$-consistent, then $F \not\subseteq \mathsf{Cl}_{\mathcal{T}}(\mathcal{A}_{rep})$. Hence, for every *CAR*-repair $\mathcal{A}_{rep}$ of $\mathcal{K}$ we have that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \not\models F$. That means that $\mathcal{A}$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. ∎

The following lemma shows that in case F is $\mathcal{T}$-consistent, then it does not belong to the consistent logical consequences with respect to $\mathcal{T}$ of any ABox accomplishing the deletion of F from a KB.

Lemma 52. *An ABox $\mathcal{A}'$ accomplishes the deletion of a $\mathcal{T}$-consistent set F of ABox assertions from a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if and only if $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$.*

Proof. ($\Rightarrow$) We show that, if $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, then $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Since $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, then Proposition 20 guarantees that $F \not\subseteq \mathsf{cl}_{\mathcal{T}}(\langle \mathcal{T}, \emptyset \rangle)$. Suppose, by way of contradiction, that $F \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Since F is $\mathcal{T}$-consistent, Theorem 21 guarantees that one can build a *CAR*-repair of $\langle \mathcal{T}, \mathcal{A}' \rangle$ that contains F. Hence, there exists at least one *CAR*-repair $\mathcal{A}_{rep}$ of $\langle \mathcal{T}, \mathcal{A}' \rangle$ such that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \models F$, but this contradicts that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$.
($\Leftarrow$) Now, we show that if $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, then $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. Since $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, then there does not exist any $\mathcal{T}$-consistent subset $\mathcal{A}''$ of $\mathcal{A}'$ such that $\langle \mathcal{T}, \mathcal{A}'' \rangle \models F$. Hence, for every $\mathcal{A}_{rep} \in$ *CAR-Set*$\langle \mathcal{T}, \mathcal{A} \rangle$, we have that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \not\models F$. Then, we can conclude that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. ∎

Now, we specify when a set of ABox assertions accomplishes the update of a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with F minimally.

Definition 38. Let $\mathcal{A}'$ be an ABox. $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ *minimally* if $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, and there is no ABox $\mathcal{A}''$ that accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, and such that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$ has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

Note that in the above definition we refer to the notion of consistent logical consequences of $\mathcal{A}$ with respect to $\mathcal{T}$.

Example 45. Consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ presented in Example 44. Suppose that we want to update $\mathcal{K}$ with the insertion of the set of ABox assertions $F^+ = \{CX(c1), fb(w, c1), mf(h, t_1)\}$. In words, F^+ states that c_1 is the cox of the crew w, and that h is a member of the rowing team t_1.

Consider the following ABoxes.

$$\mathcal{A}_1^+ = \{\ CX(c_1), mf(c_1, t_1), fb(w, c_1), mf(c_2, t_1), mf(h, t_1), OA(h), CH(h), \\ RTM(t_2), fb(w, c_2)\ \};$$
$$\mathcal{A}_2^+ = \{\ CX(c_1), mf(c_1, t_1), fb(w, c_1), mf(c_2, t_1), mf(h, t_1), OA(h), CH(h), \\ RTM(t_2), CX(c_2)\ \}.$$

It is easy to see that both $\mathcal{A}_1^+$ and $\mathcal{A}_2^+$ are $\mathcal{T}$-inconsistent, indeed the assertions $OA(h)$ and $CH(h)$ contradict the assertion in the TBox stating that a coach cannot be an athlete. We have:

- The set *CAR-Set*$\langle \mathcal{T}, \mathcal{A}_1^+ \rangle$ is constituted by the following *CAR*-repairs.

$$\begin{aligned}
\textit{CA-rep}_1 = \{\ & CX(c_1), mf(c_1,t_1), fb(w,c_1), mf(c_2,t_1), mf(h,t_1),\\
& OA(h), CH(h), RTM(t_2), fb(w,c_2)\ \} \setminus \{\ OA(h)\ \};\\
\textit{CA-rep}_2 = \{\ & CX(c_1), mf(c_1,t_1), fb(w,c_1), mf(c_2,t_1), mf(h,t_1),\\
& OA(h), CH(h), RTM(t_2), fb(w,c_2)\ \} \setminus \{\ CH(h)\ \}.
\end{aligned}$$

- The set $\textit{CAR-Set}\langle \mathcal{T}, \mathcal{A}_2^+ \rangle$ is constituted by the following *CAR*-repairs.

$$\begin{aligned}
\textit{CA-rep}_1 = \{\ & CX(c_1), mf(c_1,t_1), fb(w,c_1), mf(c_2,t_1), mf(h,t_1),\\
& OA(h), CH(h), RTM(t_2), CX(c_2)\ \} \setminus \{\ OA(h)\ \};\\
\textit{CA-rep}_2 = \{\ & CX(c_1), mf(c_1,t_1), fb(w,c_1), mf(c_2,t_1), mf(h,t_1),\\
& OA(h), CH(h), RTM(t_2), CX(c_2)\ \} \setminus \{\ CH(h)\ \}.
\end{aligned}$$

Since each ABox above entails together with $\mathcal{T}$ the set of facts belonging to F, we can conclude that both $\mathcal{A}_1^+$ and $\mathcal{A}_2^+$ accomplish the insertion of F^+ into $\mathcal{K}$. Moreover, it can be shown that they do it minimally. □

The following theorem provides a constructive characterization of the notion of an ABox accomplishing the insertion of a set of facts into a possibly inconsistent KB minimally.

Theorem 43. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent $\mathcal{L}$-KB and let $\mathcal{A}'$ and F be two sets of ABox assertions. $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally if and only if $\mathsf{clc}_\mathcal{T}(\mathcal{A}') = \mathsf{clc}_\mathcal{T}(\mathcal{A}'' \cup F)$, for some maximal subset $\mathcal{A}''$ of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle \models_{CAR} F$.*

Proof.
($\Rightarrow$) We show that if $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, then $\mathsf{clc}_\mathcal{T}(\mathcal{A}') = \mathsf{clc}_\mathcal{T}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle \models_{CAR} F$. Since $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$, from Proposition 19 we have that F is $\mathcal{T}$-consistent.

The proof proceeds by contradiction as follows. Suppose that there does not exist a set $\mathcal{A}''$ that is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle \models_{CAR} F$ and such that $\mathsf{clc}_\mathcal{T}(\mathcal{A}') = \mathsf{clc}_\mathcal{T}(\mathcal{A}'' \cup F)$. Let $\mathsf{clc}_\mathcal{T}(\mathcal{A}') = \mathsf{clc}_\mathcal{T}(\mathcal{A}_{tmp} \cup F)$. Firstly, we show that $\mathcal{A}_{tmp}$ is a subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{A}_{tmp} \cup F \rangle \models_{CAR} F$. Since $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, then $\langle \mathcal{T}, \mathcal{A}_{tmp} \cup F \rangle \models_{CAR} F$. Moreover, from Theorem 25, we have that in $\mathsf{clc}_\mathcal{T}(\mathcal{A}_{tmp} \cup F)$ there does not exist a minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$.

Suppose, by way of contradiction, that $\mathcal{A}_{tmp} \setminus \mathsf{clc}_\mathcal{T}(\mathcal{A}) = \alpha$. This means that there is an assertion α, in $\mathcal{A}_{tmp}$ that does not belong to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. Consider the set of ABox assertion $\mathcal{B} = \mathcal{A}_{tmp} \setminus \{\alpha\}$. Since the monotonicity of the language used to express $\mathcal{T}$ ensures that by removing an assertion from a $\mathcal{T}$-inconsistent ABox we cannot introduce a new inconsistency, we have that in $\mathsf{clc}_\mathcal{T}(\mathcal{B} \cup F)$ there does not exist a minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$.

Hence, $\langle \mathcal{T}, \mathcal{B} \cup F\rangle \models_{CAR} F$. Since $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then the following cases are conceivable:

- $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{B} \cup F) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, and $\mathsf{clc}_{\mathcal{T}}(\mathcal{B}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Hence, $\mathsf{clc}_{\mathcal{T}}(\mathcal{B})$ has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Which contradicts the fact that $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally.

- $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{B} \cup F)$. This can happens, for instance, if there is in $\mathsf{clc}_{\mathcal{T}}(\{\alpha\} \cup F)$ an assertion β, such that $\beta \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ and $\beta \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp} \setminus \{\alpha\} \cup F)$. But, since $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A}\rangle$, then the set $\{\beta\} \cup F$ is $\mathcal{T}$-consistent. Hence, we can build the set $\mathcal{B}' = \mathcal{B} \cup \{\beta\}$, which accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A}\rangle$ and has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Which contradicts the fact that $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally.

Hence, we conclude that $\mathcal{A}_{tmp}$ is a subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

Assume that $\mathcal{A}_{tmp}$ is not a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}_{tmp} \cup F\rangle \models_{CAR} F$. This means that there is an assertion α in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}_{tmp} \cup F \cup \{\alpha\}\rangle \models_{CAR} F$, and that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus (\mathcal{A}_{tmp} \cup \{\alpha\}) \subset \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathcal{A}_{tmp}$. Let $\mathcal{A}''' = \mathcal{A}_{tmp} \cup \mathsf{cl}_{\mathcal{T}}(F) \cup \{\alpha\}$. Clearly, $\mathcal{A}'''$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A}\rangle$. The following cases are conceivable:

(i) $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}''') = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, which means that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}''') = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ and then $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup \{\alpha\} \cup F)$.

(ii) $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}''') \subset \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Hence, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}''')$ has fewer deletions than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Which contradicts that $\mathcal{A}'$ accomplishes the insertion minimally.

Case (i) states that if $\mathcal{A}_{tmp}$ is not a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}_{tmp} \cup F\rangle \models_{CAR} F$, then there is an $\mathcal{A}''$ which is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F\rangle \models_{CAR} F$, and such that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp} \cup F) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$. Which means that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F\rangle \models_{CAR} F$.

($\Leftarrow$) Let $\mathcal{A}'$ be an ABox such that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F\rangle \models_{CAR} F$. We show that $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally. Since $\langle \mathcal{T}, \mathcal{A}'' \cup F\rangle \models_{CAR} F$, then for each $\mathcal{A}_{rep} \in \mathit{CAR\text{-}Set}(\langle \mathcal{T}, \mathcal{A}'' \cup F\rangle)$, we have that $\langle \mathcal{T}, \mathcal{A}_{rep}\rangle \models F$. That is $\langle \mathcal{T}, \mathcal{A}'' \cup F\rangle$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A}\rangle$. Proposition 19 guarantees that $\mathit{Mod}(\langle \mathcal{T}, F\rangle) \neq \emptyset$. We now show that $\langle \mathcal{T}, \mathcal{A}'' \cup F\rangle$ does it minimally. The proof proceeds by contradiction. Suppose that there is an

ABox $\mathcal{A}_{tmp}$ accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ such that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp})$ has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. One of the following cases occurs:

- $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp})$ has fewer deletions than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. This means that there is an assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp})$ such that $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ and $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$. Hence, $\langle \mathcal{T}, \mathcal{A}'' \cup \{\alpha\} \cup F \rangle \models_{CAR} F$, which contradicts that $\mathcal{A}''$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle \models_{CAR} F$.
- $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp})$ has the same deletions of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$ and $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp})$ has fewer insertions than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Which means that there is an assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$ that does not belong neither to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp})$ nor to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Since $\mathcal{A}'' \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, $\alpha \in \mathsf{cl}_{\mathcal{T}}(F)$ or α is a consequence of the assertions in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$ together with $\mathcal{T}$ and the assertions in $\mathsf{cl}_{\mathcal{T}}(F)$. Since, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp})$ and $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$ have the same deletions with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, and since $\mathsf{cl}_{\mathcal{T}}(F) \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp})$, we can conclude that $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_{tmp})$. Hence, we have a contradiction.

■

The analogous of Definition 38 for deletion is as follows.

Definition 39. Let $\mathcal{A}'$ be an ABox. $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ *minimally* if $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and there is no ABox $\mathcal{A}''$ that accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and such that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$ has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

Example 46. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be the KB presented in Example 44. Suppose that we want to update $\mathcal{K}'$ with the deletion of $F^- = \{mf(c_1, t_1), OA(h)\}$. It is easy to verify that both the following $\mathcal{T}$-inconsistent ABoxes accomplish the deletion of F^- from $\mathcal{K}'$ minimally.

$$\mathcal{A}_1^- = \{\ CX(c_1), fb(w, c_1), CX(c_2), mf(c_2, t_1), fb(w, c_2), CH(h), \\ mf(h, t_2), OA(h)\ \};$$
$$\mathcal{A}_2^- = \{\ CX(c_1), fb(w, c_1), CX(c_2), mf(c_2, t_1), fb(w, c_2), CH(h), mf(h, t_2), \\ ATH(h), mf(c_1, t_1)\ \}.$$

Indeed, we have that

$\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \not\models mf(c_1, t_1)$, for each $\mathcal{A}'' \in \mathit{CAR\text{-}Set}\langle \mathcal{A}_1^- \rangle$,
$\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \not\models OA(h)$, for each $\mathcal{A}'' \in \mathit{CAR\text{-}Set}\langle \mathcal{A}_2^- \rangle$,

and, for each ABox $\mathcal{A}'''$ such that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \neq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_1^-)$ (resp. $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \neq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_2^-)$) which accomplishes the deletion of F from $\mathcal{K}$, we have that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_1^-)$ (resp. $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_2^-)$) has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. □

The following lemma states that, if F is $\mathcal{T}$-inconsistent, then the ABox $\mathcal{A}$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally.

Lemma 53. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB and let F be a $\mathcal{T}$-inconsistent set of ABox assertions. Then, $\mathcal{A}$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally.*

Proof. The proof directly follows from Lemma 51 which guarantees that $\mathcal{A}$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. ∎

Similarly to Theorem 43, the following theorem provides a constructive characterization of the notion of an ABox accomplishing the deletion of F from an inconsistent KB in that cases where F is $\mathcal{T}$-consistent.

Theorem 44. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent $\mathcal{L}$-KB and let F be a $\mathcal{T}$-consistent set of ABox assertions. An ABox $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally if and only if $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$.*

Proof.
($\Rightarrow$) We show that if $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, then $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Since $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ then $F \not\subseteq \mathsf{cl}_{\mathcal{T}}(\langle \mathcal{T}, \emptyset \rangle)$. Moreover, Lemma 52 guarantees that $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. We proceed by contradiction. First, suppose that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Then, there is at least one assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ that does not belong to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Since $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, then, for each $\mathcal{A}_{rep} \in \mathit{CAR\text{-}Set}(\langle \mathcal{T}, \mathcal{A} \rangle)$, we have that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \not\models F$. Due to the monotonicity of $\mathcal{L}$, we have that also $\langle \mathcal{T}, \mathcal{A}_{rep} \setminus \{\alpha\} \rangle \not\models F$. Hence, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \setminus \{\alpha\}$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ and has fewer insertions than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, but this contradicts that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Now, suppose that there exists an assertion $\beta \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that the set $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \cup \{\beta\}$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and suppose that $\beta \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. This means that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is not a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. It follows that the set $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \cup \{\beta\}$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\mathsf{clc}_{\mathcal{T}}(\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \cup \{\beta\})$ has fewer deletions than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. This, clearly, contradicts that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally.

($\Leftarrow$) Now, we show that if $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, then $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Since $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, then $\langle \mathcal{T}, \emptyset \rangle \not\models F$ and, from Lemma 52, we have that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$.

Suppose, by way of contradiction, that $\mathcal{A}'$ does not accomplish the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. This means that there is an ABox $\mathcal{A}''$ that

accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$, and such that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$ has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. The following two cases are conceivable:

(i) $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$ has fewer insertions than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. This implies that there is an assertion α in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ that does not belong to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, which contradicts that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is a subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

(i) $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$ has fewer deletions than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. This implies that there exists an assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$ that belongs to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ and that does not belong to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Since $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \subset \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$. Hence, $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}' \cup \{\alpha\})$. Therefore, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is not a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, which is a contradiction.

■

As shown by Example 45 and Example 46, in general there may exist more than one ABox that accomplishes the update. As discussed in Section 6.4 there are different approaches to cope with the *multiple results* problem. Motivated by the same reasons that led us to adopt the *When In Doubt Throw It Out principle* [51, 52] (WIDTIO) for updating consistent KB (cf. Section 6.4), we base, also in this case, our semantics for update on the intersection of all the ABoxes accomplishing the update minimally.

Therefore, according with the WIDTIO principle, we give below the formal definition of the result of updating a possibly inconsistent KB with the insertion of a set of ABox assertions.

Definition 40. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB, and let F be a finite set of ABox assertions. Moreover, let $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be the set of all ABoxes accomplishing the insertion of F into $\mathcal{K}$ minimally, and let $\mathcal{A}'$ be an ABox. The KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of updating $\mathcal{K}$ with the insertion of F if

1. $\mathcal{U}$ is empty, and $\langle \mathcal{T}, \mathcal{A}' \rangle = \langle \mathcal{T}, \mathcal{A} \rangle$, or
2. $\mathcal{U}$ is nonempty, and $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \bigcap_{1 \leq i \leq n} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i) \rangle$.

Note that in that cases where an ABox accomplishing insertion of F into $\mathcal{K}$ does not exist, we establish that the result of updating $\mathcal{K}$ with the insertion of F is $\mathcal{K}$ itself.

Similarly, we define below when a KB is the result of updating a possibly inconsistent KB with the deletion of a set of facts.

Definition 41. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB, and let F be a finite set of ABox assertions. Moreover, let $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be the set of all ABoxes accomplishing the deletion of F from $\mathcal{K}$ minimally, and let $\mathcal{A}'$ be an ABox. The KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of updating $\mathcal{K}$ with the deletion of F if

1. $\mathcal{U}$ is empty, and $\langle \mathcal{T}, \mathcal{A}' \rangle = \langle \mathcal{T}, \mathcal{A} \rangle$, or

2. $\mathcal{U}$ is nonempty, and $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \bigcap_{1 \leq i \leq n} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i) \rangle$.

We observe that also in this case, when there is no ABox accomplishing the update, the update operation does not alter the original KB.

10.2 Properties of our inconsistency-tolerant update operators

In this section, we discuss how the inconsistency-tolerant update operator presented in the previous section, relates to the update properties presented in Section 6.1. In what follows, we use the term "update" as a generalization of "update by insertion" and "update by deletion". Moreover, we assume that $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a possibly inconsistent DL KB, that $\mathcal{T}$ is a consistent TBox, and that F is a finite set of ABox assertions.

Since, we are assuming that $\mathcal{K}$ may be inconsistent, we cannot refer to the classic notion of entailment. For this reason we adopt the notion of *CAR*-entailment given in Section 8.2. Preposition 16 guarantees that if $\mathcal{K}$ is a consistent KB, then the notion of *CAR*-entailment coincide with the notion FOL-entailment. Moreover, if $\mathcal{K}$ is consistent, then $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

We start by showing that, if at least one ABox accomplishing the update of a KB $\mathcal{K}$ with a set of ABox assertions F exists, then the inconsistency-tolerant update operators capture the *success of the update* property. In other words, we show that in case of insertion the result of the update implies, under *CAR*-semantics, the set F, conversely, in case of deletion, the result of the update does not entail F under *CAR*-semantics.

Proposition 21. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be possibly inconsistent $\mathcal{L}$-KB, and let F be a set of ABox assertions. Let $\langle \mathcal{T}, \mathcal{A}' \rangle$ be a KB resulting from updating $\mathcal{K}$ with the insertion (resp. deletion) of F. Then*

- *if F is $\mathcal{T}$-consistent, then $\langle \mathcal{T}, \mathcal{A}' \rangle \models_{CAR} F$;*
- *if $\langle \mathcal{T}, \emptyset \rangle \not\models F$, then $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models_{CAR} F$.*

Proof. We start showing that if $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of updating $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of a $\mathcal{T}$-consistent set F, then $\langle \mathcal{T}, \mathcal{A}' \rangle \models_{CAR} F$. Since F is $\mathcal{T}$-consistent, from Proposition 19 it directly follows that the set $\mathcal{U} = \{\mathcal{A}_1, \ldots \mathcal{A}_n\}$ containing the ABoxes accomplishing the insertion of F into $\mathcal{K}$ minimally is non-empty. Suppose, by way of contradiction, that $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models_{CAR} F$. The following cases are conceivable:

- $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. This means that there is at least one ABox $\mathcal{A}'' \in \mathcal{U}$ such that $F \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$. It follows that, for each *CAR*-repairs $\mathcal{A}_{rep}$ of $\langle \mathcal{T}, \mathcal{A}'' \rangle$, we have that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \not\models F$. Hence $\langle \mathcal{T}, \mathcal{A}'' \rangle$ cannot accomplish the insertion of F into $\mathcal{K}$, which is a contradiction.

- there exists a *CAR*-repair $\mathcal{A}_{rep}$ of F such that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \not\models_{CAR} F$. From Theorem 25, it follows that there exists in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ a minimal $\mathcal{T}$-inconsistent set V such that $V \cap F \neq \emptyset$. Since $V \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, then $V \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$ for each ABox $\mathcal{A}'' \in \mathcal{U}$. Hence, by exploiting again Theorem 25, we have that there exists at least one *CAR*-repair $\mathcal{A}''_{rep}$ of $\langle \mathcal{T}, \mathcal{A}'' \rangle$ such that $\langle \mathcal{T}, \mathcal{A}_r ep \rangle \not\models F$. Hence, $\langle \mathcal{T}, \mathcal{A}'' \rangle$ cannot accomplish the insertion of F into $\mathcal{K}$, which is a contradiction.

We now show that if $\langle \mathcal{T}, \emptyset \rangle \not\models F$, and $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of updating $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F, then $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models_{CAR} F$. Since $\langle \mathcal{T}, \emptyset \rangle \not\models F$, from Proposition 20, it directly follows that the set $\mathcal{U} = \{\mathcal{A}_1, \ldots \mathcal{A}_n\}$ containing the ABoxes accomplishing the deletion of F from $\mathcal{K}$ minimally is non-empty. Toward a contradiction. Suppose that $\langle \mathcal{T}, \mathcal{A}' \rangle \models_{CAR} F$. This means that for each repair $\mathcal{A}_{rep}$ of $\langle \mathcal{T}, \mathcal{A}' \rangle$, $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \models F$. Hence, F is $\mathcal{T}$-consistent. Clearly, we have also that, for every assertion α in F, the set $\{\alpha\}$ is $\mathcal{T}$-consistent. Moreover, from the fact that $\langle \mathcal{T}, \mathcal{A}' \rangle \models_{CAR} F$, we can conclude that $F \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, and then that $F \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$, for each $\mathcal{A}'' \in \mathcal{U}$. Since F is $\mathcal{T}$-consistent, there does not exist any minimal T-inconsistent set $V \subseteq F$. Hence, from Theorem 20, it follows that, for each $\mathcal{A}'' \in \mathcal{U}$, there exists at least one repair $\mathcal{A}''_{rep}$ in *CAR-Set*$\langle \mathcal{T}, \mathcal{A}'' \rangle$, such that $\langle \mathcal{T}, \mathcal{A}''_{rep} \rangle \models F$. Hence, $\mathcal{A}''$ cannot accomplish the deletion of F from $\mathcal{K}$, which is a contradiction. ∎

We now focus our attention on a property regarding the *uniqueness of the result*. By providing the following proposition, we state that if both the KBs $\mathcal{K}'$ and $\mathcal{K}''$ result from updating a possibly inconsistent KB with the same set of ABox assertions, then $\mathcal{K}'$ and $\mathcal{K}''$ are C-equivalent. Therefore, from Proposition 17, it follows that, with respect to the *CAR*-semantics, our inconsistency-tolerant update operators are functional.

Proposition 22. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent KB. Up to C-equivalence, there is exactly one result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F, and exactly one result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F.*

Proof. Firstly, we show that, given two KBs $\langle \mathcal{T}, \mathcal{A}^1 \rangle$ and $\langle \mathcal{T}, \mathcal{A}^2 \rangle$ such that both result from updating $\mathcal{K}$ with the insertion of F, we have that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}^1) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}^2)$. Note that if F is $\mathcal{T}$-inconsistent, then from Proposition 19, we have that the set $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ containing the ABoxes that accomplish the insertions of F into $\mathcal{K}$ minimally is empty. Then, by following Definition 40,

the KB resulting from the update is $\mathcal{K}$ itself. Hence, if F is $\mathcal{T}$-inconsistent, then the claim is trivially proved.

Now, suppose that F is $\mathcal{T}$-consistent. From Definition 40 we have that:

- $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}^1)\rangle = \langle \mathcal{T}, \bigcap_{1\leq i\leq n} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)\rangle$, with $\mathcal{A}_i \in \mathcal{U}$;
- $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}^2)\rangle = \langle \mathcal{T}, \bigcap_{1\leq i\leq n} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)\rangle$, with $\mathcal{A}_i \in \mathcal{U}$;

Hence, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}^1) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}^2)$.

We now prove the claim in case of deletion. Let us start considering the case in which the set $\mathcal{U}$ containing the ABoxes accomplishing the deletion of F from $\mathcal{K}$ minimally is empty. According with Definition 41, we have that the result of updating $\mathcal{K}$ with the deletion of F is $\mathcal{K}$ itself. Therefore the claim is trivially proved.

Suppose that the set $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_2\}$ containing the ABoxes accomplishing the deletion of F into $\mathcal{K}$ minimally is non-empty. According with Definition 41 we have:

- $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}^1)\rangle = \langle \mathcal{T}, \bigcap_{1\leq i\leq n} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)\rangle$, with $\mathcal{A}_i \in \mathcal{U}$;
- $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}^2)\rangle = \langle \mathcal{T}, \bigcap_{1\leq i\leq n} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)\rangle$, with $\mathcal{A}_i \in \mathcal{U}$;

Hence, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}^1) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}^2)$. ∎

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ be a possibly inconsistent KB and let F be a set of ABox assertions. In what follows, we denote the result of updating $\mathcal{K}$ with the insertion of F according to our inconsistency-tolerant semantics with $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F$, and the result of updating $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ with the deletion of F with $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F$.

The following proposition states that, in the case where the original KB is consistent, our inconsistency-tolerant update semantics coincide with the semantics for updating consistent KBs presented in Section 6.4.

Proposition 23. *Let $\langle \mathcal{T}, \mathcal{A}\rangle$ be a consistent KB, and let F be a set of ABox assertions. Up to logical equivalence, we have that:*

- $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F = \mathcal{K} \oplus^{\mathcal{T}}_{\cap} F$;
- $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F = \mathcal{K} \ominus^{\mathcal{T}}_{\cap} F$.

Proof.

We first show that if $\mathcal{K}$ is consistent, then $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F = \mathcal{K} \oplus^{\mathcal{T}}_{\cap} F$. If F is $\mathcal{T}$-inconsistent, from Definition 40 we have that $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F = \mathcal{K}$. Moreover, from Definition 22 we have that $\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F$ is logical equivalent to $\mathcal{K}$. Hence, $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F$ is logical equivalent to $\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F$.

Let F be $\mathcal{T}$-consistent. We prove the claim by showing that if an ABox $\mathcal{A}'$ accomplishes the insertion of F minimally according to Definition 38, then it

accomplishes the insertion of F minimally also according to Definition 19, and vice-versa.

To this aim we first prove that, if both $\mathcal{A}$ and F are $\mathcal{T}$-consistent then every ABox accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally according to Definition 38 is $\mathcal{T}$-consistent. Let $\mathcal{A}'$ be an ABox accomplishing the insertions of F into $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ minimally according to Definition 38. We have that for each $\mathcal{A}'' \in \mathit{CAR\text{-}Set}(\langle \mathcal{T}, \mathcal{A}' \rangle)$, $\langle \mathcal{T}, \mathcal{A}'' \rangle \models F$, which means that $\mathsf{cl}_{\mathcal{T}}(F) \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. The set $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ can be partitioned into two sets of ABox assertions, namely, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') = \mathcal{A}_A \cup \mathcal{A}_F$ where:

- $\mathcal{A}_A \cap \mathcal{A}_F = \emptyset$;
- $\mathcal{A}_F = \{\alpha \mid \exists \mathcal{A}''' \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')\ s.t.\ \alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}''' \cup F)\ and\ \alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}''')\}$.

Intuitively, $\mathcal{A}_F$ contains those assertions in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ that are consequences of assertions in F, i.e., $\mathsf{cl}_{\mathcal{T}}(F) \subseteq \mathcal{A}_F$, or that can only be derived by assertions in $\mathcal{A}_A$ in conjunction with at least one assertion in F.

Since $\langle \mathcal{T}, \mathcal{A}' \rangle \models_{CAR} F$. It follows, from Theorem 25, that for each assertion $\alpha \in \mathcal{A}_F$, there is no minimal $\mathcal{T}$-inconsistent set V such that $\alpha \in V$. Indeed, suppose, by way of contradiction, that there exists a minimal $\mathcal{T}$-inconsistent set V such that $\alpha \in V$. Since $\alpha \in V$, then there exists a possibly empty subset $\mathcal{A}'_A$ of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_A)$ and a non-empty set $F' \subseteq F$ such that $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'_A \cup F')$, and such that for each assertion $\beta \in (\mathcal{A}'_A \cup F')$, $\alpha \notin \mathsf{clc}_{\mathcal{T}}((\mathcal{A}'_A \cup F') \setminus \{\beta\})$. Which means that $(V \setminus \{\alpha\}) \cup (\mathcal{A}'_A \cup F')$ is a minimal $\mathcal{T}$-inconsistent set. Hence, there exists a minimal $\mathcal{T}$-inconsistent set that overlaps with F, which contradicts Theorem 25. From the fact that there does not exist any minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathcal{A}_F \neq \emptyset$, we can conclude that $\mathcal{A}_F$ is $\mathcal{T}$-consistent, and that, if $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is $\mathcal{T}$-inconsistent, then every hypothetical minimal $\mathcal{T}$-inconsistent set in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is a subset of $\mathcal{A}_A$.

Theorem 43 states that if $\mathcal{A}'$ accomplishes the insertion of F minimally, then there exists a maximal subset $\mathcal{A}''$ of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle \models_{CAR} F$, and such that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'' \cup F)$. Hence, since $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ is $\mathcal{T}$-consistent, and since $\mathcal{T}$ is expressed in a monotonic language, then also $\mathcal{A}''$ is $\mathcal{T}$-consistent. Therefore, we can conclude that $\mathcal{A}'$ is $\mathcal{T}$-consistent. By exploiting Proposition 16 and Theorem 43, we get that $\mathcal{A}'$ is a $\mathcal{T}$-consistent set such that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. Hence, from Theorem 11 we have that $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally in accordance with Definition 22.

So, we have shown that if $\langle \mathcal{T}, \mathcal{A} \rangle$ is $\mathcal{T}$-consistent, then every ABox accomplishing the insertions of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ according to Definition 38, also accomplishes the insertion of F minimally according to Definition 19.

On the other hand, let $\mathcal{A}'$ be an ABox accomplishing the insertion of the $\mathcal{T}$-consistent set F into the $\mathcal{T}$-consistent KB $\langle \mathcal{T}, \mathcal{A} \rangle$ according to Definition 19. From Theorem 11 we have that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal

subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. Again, since by removing assertions from a $\mathcal{T}$-consistent set of ABox assertions one cannot obtain a $\mathcal{T}$-inconsistent set of ABox assertions, then by applying Proposition 16, we have that $\mathcal{A}''$ is a maximal subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle \models_{CAR} F$. Hence, $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ also according to Definition 38. Finally, since the set of ABox assertions accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ according to Definition 38 coincides with the set of ABox assertions accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ according to Definition 19, we can conclude that, up to logical equivalence, $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F = \mathcal{K} \oplus^{\mathcal{T}}_{\cap} F$.

We now show that if $\mathcal{K}$ is a consistent KB, then $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F = \mathcal{K} \ominus^{\mathcal{T}}_{\cap} F$. Firstly, if $\langle \mathcal{T}, \emptyset \rangle \models F$, then, from Proposition 2 and Proposition 20, the set $\mathcal{U}$ of ABoxes accomplishing the deletion of F from the consistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is empty for both the update semantics. Hence, according to Definition 23, and to Definition 41, we have that $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F$ is logically equivalent to $\mathcal{K} \ominus^{\mathcal{T}}_{\cap} F$. Let $\langle \mathcal{T}, \emptyset \rangle \not\models F$, and let F be $\mathcal{T}$-inconsistent. It directly follows from Theorem 12 and from Lemma 53 that every ABox accomplishing the deletion of F according with both Definition 20 and Definition 39 is logically equivalent to $\mathcal{A}$. Hence, if F is $\mathcal{T}$-inconsistent, then, also in case, $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F$ is logically equivalent to $\mathcal{K} \ominus^{\mathcal{T}}_{\cap} F$.

Let F be a $\mathcal{T}$-consistent set of ABox assertions, and let $\langle \mathcal{T}, \emptyset \rangle \not\models F$. Let $\mathcal{A}'$ be an ABox accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally according to Definition 39. From Theorem 44 $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Since $\mathcal{A}$ is $\mathcal{T}$-consistent, then $\mathcal{A}'$ is $\mathcal{T}$-consistent. This means that $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$ is a maximal subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $F \not\subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$. It follows from Theorem 12 that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally according to Definition 20. Similarly. we can show that if $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally according to Definition 20, then it accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally according to Definition 39. It directly follows that $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F = \mathcal{K} \ominus^{\mathcal{T}}_{\cap} F$. ∎

We observe that the above proposition, together with Proposition 6 and Proposition 7, implies that both $\oplus^{\mathcal{T}}_{CAR}$ and $\ominus^{\mathcal{T}}_{CAR}$ obey the *consistency preservation* property.

The following Proposition guarantees that the result of updating a possibly inconsistent $\mathcal{L}$-KB can be expressed in $\mathcal{L}$. This means that both the proposed inconsistency-tolerant update operators obey the *expressibility* property. As discussed in Section 6.1, this property is essential for equipping DL KB management systems with evolution operators.

Proposition 24. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly KB in $\mathcal{L}$, and let F be a set of ABox assertions. Then both $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F$ and $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F$ are KBs in $\mathcal{L}$.*

Proof. The proof is an obvious adaptation of the one proposed for Proposi-

tion 8. Indeed, it follows form Proposition 5, which guarantees that the result of the update is unique, and from the following observations:

- if $\langle \mathcal{T}, \mathcal{A} \rangle$ is expressed in $\mathcal{L}$, then $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \rangle$ is a KB in $\mathcal{L}$;
- if $\mathcal{U}$ is a set of ABoxes, and $\mathcal{T}$ is a TBox in $\mathcal{L}$, then $\langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathcal{A}_i \rangle$ is a KB in $\mathcal{L}$;
- if $\mathcal{A}$ and $\mathcal{A}'$ are two ABoxes, and $\langle \mathcal{T}, \mathcal{A} \rangle$ is a KB in $\mathcal{L}$, then $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle$ is a KB in $\mathcal{L}$.

■

We conclude this section with a discussion on the *syntax independence* property. In Section 6.5, we showed that by updating two consistent and logically equivalent KBs with the same set of facts, we obtain two KBs which are logically equivalent. If we are dealing with inconsistent KBs, the classical notion of logically equivalence becomes unusable for our purposes.

A possible adjustment of the syntax independence property to inconsistency-tolerant update could refer to the following notion of equivalence. Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be two possibly inconsistent KBs. We say that $\mathcal{K}_1$ and $\mathcal{K}_2$ are equivalent with respect to CAR-entailment, written $\models_{CAR}$-equivalent, if for every sentence σ, we have that $\mathcal{K}_1 \models_{CAR} \sigma$ and $\mathcal{K}_2 \models_{CAR} \sigma$.

In this case, we say that an inconsistency-tolerant update operator obeys the syntax independence property if for every possibly inconsistent KBs $\mathcal{K}_1$ and $\mathcal{K}_2$ that are $\models_{CAR}$-equivalent, and for every set of ABox assertions F, the result of updating $\mathcal{K}_1$ with F is $\models_{CAR}$-equivalent to the result of updating $\mathcal{K}_2$ with F.

The following example shows that our inconsistency-tolerant update operators do not obey the above notion of the syntax independence property.

Example 47. Let $\mathcal{T}$ be the TBox containing the following assertions.

$$\mathsf{Son} \sqsubseteq \mathsf{Man}, \qquad \mathsf{Husband} \sqsubseteq \mathsf{Man},$$
$$\mathsf{Man} \sqsubseteq \neg\mathsf{Woman}.$$

Consider the two KBs $\mathcal{K}_1 = \langle \mathcal{T}, \mathcal{A}_1 \rangle$ and $\mathcal{K}_2 = \langle \mathcal{T}, \mathcal{A}_2 \rangle$ where

$$\begin{array}{lcl} \mathcal{A}_1 & = & \{\, \mathsf{Son}(andrea), \mathsf{Woman}(andrea) \,\}; \\ \mathcal{A}_2 & = & \emptyset. \end{array}$$

It is easy to see that $\mathcal{K}_1$ and $\mathcal{K}_2$ are $\models_{CAR}$-equivalent. In particular, if we consider the sentence $\mathsf{Son}(andrea)$, we have:

$$\mathcal{K}_1 \not\models_{CAR} \mathsf{Son}(andrea), \qquad \mathcal{K}_2 \not\models_{CAR} \mathsf{Son}(andrea).$$

Now, suppose we update both $\mathcal{K}_1$ and $\mathcal{K}_2$ with the insertion of the set of ABox assertions $F = \{\mathsf{Husband}(andrea)\}$. By applying the inconsistency-tolerant update semantics given in Section 10.1, we have that:

$$\begin{array}{lcl} \mathcal{K}_1 \oplus^{\mathcal{T}}_{CAR} F & = & \langle \mathcal{T}, \{\ \mathsf{Son}(andrea), \mathsf{Husband}(andrea)\ \}\rangle; \\ \mathcal{K}_2 \oplus^{\mathcal{T}}_{CAR} F & = & \langle \mathcal{T}, \{\ \mathsf{Husband}(andrea)\ \}\rangle. \end{array}$$

Considering again the sentence $\mathsf{Son}(andrea)$, we have:

$$\begin{array}{lll} \mathcal{K}_1 \oplus^{\mathcal{T}}_{CAR} F & \models_{CAR} & \mathsf{Son}(andrea); \\ \mathcal{K}_2 \oplus^{\mathcal{T}}_{CAR} F & \not\models_{CAR} & \mathsf{Son}(andrea). \end{array}$$

Therefore, $\mathcal{K}_1 \oplus^{\mathcal{T}}_{CAR} F$ and $\mathcal{K}_2 \oplus^{\mathcal{T}}_{CAR} F$ are not $\models_{CAR}$-equivalent. □

A different manner to compare two inconsistent KBs refers to the notion of C-equivalence given in Section 8.1. By adopting this notion of equivalence, we say that an inconsistency-tolerant update operator obeys the syntax independence property if for every possibly inconsistent KBs $\mathcal{K}_1$ and $\mathcal{K}_2$ that are C-equivalent, and for every set of ABox assertions F, the result of updating $\mathcal{K}_1$ with F is C-equivalent to the result of updating $\mathcal{K}_2$ with F.

The next proposition states that our inconsistency-tolerant update operators obey this notion of syntax independence property.

Proposition 25. *Let $\mathcal{K}_1 = \langle \mathcal{T}, \mathcal{A}_1\rangle$ and $\mathcal{K}_2 = \langle \mathcal{T}, \mathcal{A}_2\rangle$ be two possibly inconsistent KBs, and let F be a set of ABox assertions. If $\mathcal{K}_1$ and $\mathcal{K}_2$ are C-equivalent. Then,*

- *$\mathcal{K}_1 \oplus^{\mathcal{T}}_{CAR} F$ and $\mathcal{K}_2 \oplus^{\mathcal{T}}_{CAR} F$ are C-equivalent;*
- *$\mathcal{K}_1 \ominus^{\mathcal{T}}_{CAR} F$ and $\mathcal{K}_2 \ominus^{\mathcal{T}}_{CAR} F$ are C-equivalent.*

Proof. Let $\langle \mathcal{T}, \mathcal{A}_1\rangle$ and $\langle \mathcal{T}, \mathcal{A}_2\rangle$ be two C-equivalent KBs, and let F be a set of atomic ABox assertions. We start showing that $\mathcal{K}_1 \oplus^{\mathcal{T}}_{CAR} F$ and $\mathcal{K}_2 \oplus^{\mathcal{T}}_{CAR} F$ are C-equivalent. If F is $\mathcal{T}$-inconsistent, the proof follows directly from Proposition 19 and from Definition 40. Moreover, if $\mathcal{K}_1$ and $\mathcal{K}_2$ are two consistent KBs, the proof directly follows from Proposition 23 and from Proposition 9. Let $\langle \mathcal{T}, \mathcal{A}_1\rangle$ and $\langle \mathcal{T}, \mathcal{A}_2\rangle$ be two inconsistent C-equivalent KBs, and let F be $\mathcal{T}$-consistent. Let $\mathcal{U}^1 = \{\mathcal{A}^1_1, \ldots, \mathcal{A}^1_n\}$ be the set of ABox accomplishing the insertion of F into $\mathcal{K}_1$ minimally, and let $\mathcal{U}^2 = \{\mathcal{A}^2_1, \ldots, \mathcal{A}^2_n\}$ be the set of ABox accomplishing the insertion of F into $\mathcal{K}_2$ minimally. From Theorem 43 we have that:

- $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}^1_j) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'^1_j \cup F)$, where $\mathcal{A}'^1_j$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_1)$ such that $\langle \mathcal{T}, \mathcal{A}'^1_j \cup F\rangle \models_{CAR} F$, for each $\mathcal{A}^1_j \in \mathcal{U}^1$;

- $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j^2) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j'^2 \cup F)$, where $\mathcal{A}_j'^2$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_2)$ such that $\langle \mathcal{T}, \mathcal{A}_j'^2 \cup F \rangle \models_{CAR} F$, for each $\mathcal{A}_j^2 \in \mathcal{U}^2$.

Since $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_1) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_2)$, it follows that:

- $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j^1) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j'^1 \cup F)$, where $\mathcal{A}_j'^1$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_2)$ such that $\langle \mathcal{T}, \mathcal{A}_j'^1 \cup F \rangle \models_{CAR} F$, for each $\mathcal{A}_j^1 \in \mathcal{U}^1$;
- $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j^2) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j'^2 \cup F)$, where $\mathcal{A}_j'^2$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_1)$ such that $\langle \mathcal{T}, \mathcal{A}_j'^2 \cup F \rangle \models_{CAR} F$, for each $\mathcal{A}_j^2 \in \mathcal{U}^2$.

This means that $\mathcal{U}_1 = \mathcal{U}_2$ and therefore $\mathcal{K}_1 \oplus_{CAR}^{\mathcal{T}} F$ and $\mathcal{K}_2 \oplus_{CAR}^{\mathcal{T}} F$ are C-equivalent.

Similarly, one can easily prove that $\mathcal{K}_1 \ominus_{CAR}^{\mathcal{T}} F$ and $\mathcal{K}_2 \ominus_{CAR}^{\mathcal{T}} F$ are C-equivalent. ∎

Chapter 11

Updating inconsistent *DL-Lite*$_{A,id,den}$ KBs

In this chapter we turn our attention to *DL-Lite*$_{A,id,den}$-KBs and we study the problem of updating possibly inconsistent KBs expressed in this logic. To this aim we first provide some results which give a constructive characterization of the problem of updating a possibly inconsistent *DL-Lite*$_{A,id,den}$-KB by adopting the inconsistency-tolerant update semantics presented in the previous chapter. Then, on the base of such results, we design two algorithms, one for updating by insertion, and the other one for updating by deletion a possibly inconsistent KBs expressed in the DL *DL-Lite*$_{A,id,den}$. We recall that, as the other logics belonging to the *DL-Lite* family, *DL-Lite*$_{A,id,den}$ has been specifically designed to keep all reasoning tasks polynomially tractable in the size of the ABox. By characterizing the computational complexity of the insertion and the deletion algorithms for *DL-Lite*$_{A,id,den}$, we show that this property still holds for inconsistency-tolerant update.

11.1 Inconsistency-tolerant insertion in *DL-Lite*$_{A,id,den}$

In this section we address the problem of updating a possibly inconsistent *DL-Lite*$_{A,id}$-KB $\mathcal{K}$ with the insertion of a set of ABox assertions F, according to the semantics given in Section 10.1. Thus, our goal is computing the KB $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F$. To this aim we present the algorithm $\mathsf{ComputeInsertion}_{CAR}$, that takes in input a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K}$ and a finite set of facts F, and computes the result of updating $\mathcal{K}$ with the insertions of F.

In accordance with Definition 40, we have that, a *DL-Lite*$_{A,id}$-KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of updating the *DL-Lite*$_{A,id,den}$-KB $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of a $\mathcal{T}$-consistent set of ABox assertions F, if $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i) \rangle$, where $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ is the set of all ABoxes accomplishing the insertion

of F into $\mathcal{K}$ minimally. Thus, by exploiting Proposition 22, one can compute $\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{CAR} F$, by first computing the ABox $\mathcal{A}_\cap = \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{clc}_\mathcal{T}(\mathcal{A}_i)$. In what follows we show how one can obtain $\mathcal{A}_\cap$ without computing every ABox $\mathcal{A}_i \in \mathcal{U}$.

We recall that *DL-Lite*$_{A,id,den}$ enjoys the following property. Given a TBox in *DL-Lite*$_{A,id,den}$ and two $\mathcal{T}$-consistent ABoxes $\mathcal{A}_1$ and $\mathcal{A}_2$, we have that $\mathsf{cl}_\mathcal{T}(\mathcal{A}_1 \cup \mathcal{A}_2) = \mathsf{cl}_\mathcal{T}(\mathcal{A}_1) \cup \mathsf{cl}_\mathcal{T}(\mathcal{A}_2)$. Clearly, such a property holds also if we consider the set of consistent logical consequences of an ABox in place of the deductive closure. Therefore, given a *DL-Lite*$_{A,id,den}$ TBox and two ABoxes $\mathcal{A}_1$ and $\mathcal{A}_2$, we have that $\mathsf{clc}_\mathcal{T}(\mathcal{A}_1 \cup \mathcal{A}_2) = \mathsf{clc}_\mathcal{T}(\mathcal{A}_1) \cup \mathsf{clc}_\mathcal{T}(\mathcal{A}_2)$.

We also recall that Theorem 25 guarantees that, given a possibly inconsistent KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and an ABox assertion $\alpha \in \mathsf{clc}_\mathcal{T}(\mathcal{A})$, a *CA*-repair $\mathcal{A}'$ of $\mathcal{K}$ such that $\alpha \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}')$ exists if and only if there exists a minimal $\mathcal{T}$-inconsistent set V in $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that $\alpha \in V$.

On the base of the above results, it easy to come up with the following proposition, that, by exploiting Theorem 43, provides a characterization of the notion of accomplishing the insertion of F minimally into $\mathcal{K}$ in the context of *DL-Lite*$_{A,id,den}$.

Proposition 26. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and $\mathcal{A}'$ be an ABox. $\mathcal{A}'$ accomplishes the insertion of F into $\mathcal{K}$ minimally if and only if $\mathsf{clc}_\mathcal{T}(\mathcal{A}')$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ that contains $\mathsf{cl}_\mathcal{T}(F)$, and does not contain any minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$.*

Proof.
($\Rightarrow$) Let $\mathcal{A}'$ accomplish the insertion of F into $\mathcal{K}$ minimally. Since $\mathcal{A}'$ accomplishes the insertion of F into $\mathcal{K}$, then, from Proposition 19, F is $\mathcal{T}$-consistent, and, from Theorem 43, $\mathsf{clc}_\mathcal{T}(\mathcal{A}') = \mathsf{clc}_\mathcal{T}(\mathcal{A}'' \cup F)$, where $\mathcal{A}''$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle \models_{CAR} F$, which means that $\mathsf{clc}_\mathcal{T}(\mathcal{A}') = \mathsf{clc}_\mathcal{T}(\mathcal{A}'') \cup \mathsf{cl}_\mathcal{T}(F)$ where $\mathcal{A}''$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle \models_{CAR} F$. Clearly, $\mathsf{cl}_\mathcal{T}(F) \in \mathsf{clc}_\mathcal{T}(\mathcal{A}')$, and then $\mathsf{clc}_\mathcal{T}(\mathcal{A}')$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ such that $\langle \mathcal{T}, \mathcal{A}'' \cup F \rangle \models_{CAR} F$. Finally, from Theorem 25, we have that $\mathcal{A}'$ does not contain any minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F)$.

($\Leftarrow$) Let $\mathcal{A}'$ be a set of ABox assertions such that $\mathsf{clc}_\mathcal{T}(\mathcal{A}')$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ that contains $\mathsf{cl}_\mathcal{T}(F)$, and does not contain any minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$. We show that $\mathcal{A}'$ accomplishes the insertion of F into $\mathcal{K}$ minimally. We first show that $\mathcal{A}'$ accomplishes the insertion of F into $\mathcal{K}$. Indeed, suppose that there exists $\mathcal{A}'' \in \mathit{CAR\text{-}Set}(\langle \mathcal{T}, \mathcal{A}' \rangle)$ such that $\langle \mathcal{T}, \mathcal{A}'' \rangle \not\models F$. This means that there is an assertion α in F such that $\alpha \notin \mathsf{cl}_\mathcal{T}(\mathcal{A}'')$. From Theorem 25, it follows that there is a minimal $\mathcal{T}$-inconsistent set V in $\mathsf{clc}_\mathcal{T}(\mathcal{A}')$ such that $\alpha \in V$, which is contradiction.

We now show that $\mathcal{A}'$ accomplishes the insertion of F into $\mathcal{K}$ minimally. Indeed, suppose that there exists $\mathcal{A}''$ that accomplishes the insertion of F and such that $\mathsf{clc}_\mathcal{T}(\mathcal{A}'')$ has fewer changes than $\mathsf{clc}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. From Theorem 43, both $\mathsf{clc}_\mathcal{T}(\mathcal{A}')$ and $\mathsf{clc}_\mathcal{T}(\mathcal{A}'')$ are subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$. This means that $\mathsf{clc}_\mathcal{T}(\mathcal{A}'')$ has fewer deletions than $\mathsf{clc}_\mathcal{T}(\mathcal{A}')$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. That is, there is an assertion α in $\mathsf{clc}_\mathcal{T}(\mathcal{A}'')$ such that $\alpha \notin \mathsf{clc}_\mathcal{T}(\mathcal{A}')$. Since both $\mathcal{A}'$ and $\mathcal{A}''$ accomplish the insertion of F into $\mathcal{A}$, from the observations above, it follows that α is not in $\mathsf{cl}_\mathcal{T}(F)$, and then $\alpha \in \mathsf{clc}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(F)$. Theorem 25 guarantees that there is no minimal $\mathcal{T}$-inconsistent set in $\mathsf{clc}_\mathcal{T}(\mathcal{A}'')$ such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$. Hence, for each minimal $\mathcal{T}$-inconsistent set V in $\mathsf{clc}_\mathcal{T}(\mathcal{A}'')$ such that $\alpha \in V$, we have that $V \cap \mathsf{cl}_\mathcal{T}(F) = \emptyset$. Since $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{clc}_\mathcal{T}(\mathcal{A}'') = \mathsf{clc}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{clc}_\mathcal{T}(\mathcal{A}') \cup \mathsf{cl}_\mathcal{T}(\alpha)$, we have that $\mathsf{clc}_\mathcal{T}(\mathcal{A}') \cup \mathsf{cl}_\mathcal{T}(\alpha)$ does not contains any minimal $\mathcal{T}$-inconsistent set such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$. It follows that $\mathsf{clc}_\mathcal{T}(\mathcal{A}')$ is not a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ that contains $\mathsf{cl}_\mathcal{T}(F)$ and that contains no minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$, which is contradiction. ∎

The algorithm we present is based on the characterization in the context of *DL-Lite*$_{A,id,den}$ of when an atom in $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ does not belong to some ABox accomplishing the insertion of a set of atoms minimally. Indeed, according with the WIDTIO principle, an atom α will not be in the result of the update exactly when it does not appear in at least one ABox accomplishing the insertion of F into $\mathcal{K}$ minimally. By exploiting Proposition 26, we are able to provide the following result.

Proposition 27. *Let α be an atom in $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(F)$. There is $\mathcal{A}'$ accomplishing the insertion of F into $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ minimally with $\alpha \notin \mathsf{clc}_\mathcal{T}(\mathcal{A}')$ if and only if there is a $\mathcal{T}$-inconsistent set V in $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ such that:*

(*i*) $\alpha \in V$;

(*ii*) $F \cup (V \setminus \{\alpha\})$ *is $\mathcal{T}$-consistent;*

(*iii*) $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$.

Proof.
($\Rightarrow$) Let α be an atom in $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{cl}_\mathcal{T}(F)$, and $\mathcal{A}'$ be an ABox accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally such that $\alpha \notin \mathsf{clc}_\mathcal{T}(\mathcal{A}')$. We show that there exists a $\mathcal{T}$-inconsistent set V that contains α, and that $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent, and $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$.

Since $\mathcal{A}'$ accomplish the insertion of F into $\mathcal{K}$, then, (1) from Proposition 19, F is $\mathcal{T}$-consistent; (2) $\langle \mathcal{T}, \mathcal{A}' \rangle \models_{CAR} F$; (3) from Proposition 26, $\mathcal{A}'$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ that contains $\mathsf{cl}_\mathcal{T}(F)$, and does not contain any $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$. The proof proceeds by contradiction as follows.

Suppose that for every $\mathcal{T}$-inconsistent set V' in $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$, we have that $\alpha \notin V'$. Proposition 26 guarantees that $\mathcal{A}' \subseteq \mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$, this means that also in $\mathcal{A}' \cup \alpha$ there is no $\mathcal{T}$-inconsistent set containing α. Hence, $\mathcal{A}'$ is not a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ that contains $\mathsf{cl}_\mathcal{T}(F)$, and does not contain any minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$, that is a contradiction. Suppose now that there are some $\mathcal{T}$-inconsistent sets in $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ containing α, and suppose that for every such $\mathcal{T}$-inconsistent set V' in $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ we have $V' \setminus \{\alpha\}$ is $\mathcal{T}$-inconsistent. This means that V' is not a minimal $\mathcal{T}$-inconsistent set. It leads to contradicts the fact that $\mathcal{A}'$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ that contains $\mathsf{cl}_\mathcal{T}(F)$, and does not contain any minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$. Thus, let V' be a $\mathcal{T}$-inconsistent set in $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ such that $\alpha \in V'$, and $V' \setminus \{\alpha\}$ is $\mathcal{T}$-consistent. This means that V' is a minimal $\mathcal{T}$-inconsistent set. Suppose that $F \cup (V' \setminus \{\alpha\})$ is $\mathcal{T}$-inconsistent. Since $V' \setminus \{\alpha\}$ is $\mathcal{T}$-consistent, and F is $\mathcal{T}$-consistent, then there is a $\mathcal{T}$-consistent subset V'' of $V' \setminus \{\alpha\}$ and a $\mathcal{T}$-consistent subset F' of F, such that $F'' \cup V''$ is a minimal $\mathcal{T}$-inconsistent set. It follows from Proposition 26, that in every ABox $\mathcal{A}''$ that accomplishes the insertion of F into $\mathcal{K}$ minimally, we have that $V'' \not\subseteq \mathcal{A}''$. But this means that also $V' \not\subseteq \mathcal{A}''$, hence, there is no minimal $\mathcal{T}$-inconsistent set V' in $\mathcal{A}''$ that contains α. Similarly to the previous case, this leads to contradicts the fact that $\mathcal{A}'$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ that contains $\mathsf{cl}_\mathcal{T}(F)$, and does not contain any minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$. Finally, suppose that there are some $\mathcal{T}$-inconsistent sets that contain α. Let Λ be the set containing such $\mathcal{T}$-inconsistent sets. Suppose that for each $V' \in \Lambda$ we have that $F \cup (V' \setminus \{\alpha\})$ is $\mathcal{T}$-consistent. This means that V' is a minimal $\mathcal{T}$-inconsistent set, and that for every subset V'' of $V' \setminus \{\alpha\}$, $V'' \cup F$ is not a minimal $\mathcal{T}$-inconsistent set. Moreover, suppose that for every $V \in \Lambda$, we have that $V' \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$. It follows that, given the ABox $\mathcal{A}'$, the KB $\langle \mathcal{T}, \mathcal{A}' \cup \{\alpha\} \rangle$, does not contains any minimal $\mathcal{T}$-inconsistent set V such that $V \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$. But, again, this contradicts that $\mathcal{A}'$ is a maximal subset of $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ that contains $\mathsf{cl}_\mathcal{T}(F)$, and does not contain any minimal $\mathcal{T}$-inconsistent set that overlaps with $\mathsf{cl}_\mathcal{T}(F)$.

($\Leftarrow$) Let V be a $\mathcal{T}$-inconsistent set in $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \cup \mathsf{cl}_\mathcal{T}(F)$ that overlaps with $\mathsf{cl}_\mathcal{T}(F)$ and that contains α. We show that if $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent, then there is an ABox $\mathcal{A}'$ accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally such that $\alpha \notin \mathsf{clc}_\mathcal{T}(\mathcal{A}')$. Suppose, by way of contradiction, that for every ABox $\mathcal{A}'$ accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ we have that $\alpha \in \mathsf{clc}_\mathcal{T}(\mathcal{A}')$. Since $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent, then also $\mathsf{cl}_\mathcal{T}(F) \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent. Hence, one can build a set $\mathcal{A}''$ by adding to $\mathsf{cl}_\mathcal{T}(F) \cup (V \setminus \{\alpha\})$ every assertions in $\mathsf{clc}_\mathcal{T}(\mathcal{A})$ if this addition does not lead to obtain a minimal $\mathcal{T}$-inconsistent set V'' such that $V'' \cap \mathsf{cl}_\mathcal{T}(F) \neq \emptyset$. That is, one can build a maximal subset

of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ that contains $\mathsf{cl}_{\mathcal{T}}(F)$, and does not contain any minimal $\mathcal{T}$-inconsistent set that overlaps with $\mathsf{cl}_{\mathcal{T}}(F)$, and that does not contains α. Thus, from Proposition 26, it follows that there is ABox $\mathcal{A}'$ accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ such that $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, which is a contradiction. ∎

Example 48. Consider the *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$ presented in Example 2, and consider the following ABox.

$$\mathcal{A} \;=\; \{\;\; \mathit{PortIn}(p_1),\ \mathit{PortOut}(p_1),\ \mathit{of}(p_1, d_1),\ \mathit{connectedTo}(p_1, p_2) \;\;\}$$

In words, $\mathcal{A}$ states that the port p_1 is both an incoming port and an outgoing port, that it is a port of the device d_1, and that it is connected to the port p_2.

Immediately one can see that the *DL-Lite*$_{A,id,den}$-KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent, since the two assertions $\mathit{PortIn}(p_1)$ and $\mathit{PortOut}(p_1)$ in $\mathcal{A}$ violate the disjointness $\mathit{PortIn} \sqsubseteq \neg \mathit{PortOut}$ belonging to $\mathcal{T}$.

Suppose to change the KB $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of the following set of ABox assertions.

$$F \;=\; \{\;\; \mathit{of}(p_2, d_1),\ \mathit{connectedTo}(p_1, p_3) \;\;\}$$

The ABox $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ is constituted by the following ABox assertions:

$$\begin{array}{lllll} \mathit{PortIn}(p_1), & \mathit{Port}(p_1), & \mathit{Port}(p_2), & \mathit{of}(p_1, d_1), & \mathit{connectedTo}(p_1, p_2), \\ \mathit{PortOut}(p_1), & \mathit{Port}(p_3), & \mathit{Device}(d_1), & \mathit{of}(p_2, d_1), & \mathit{connectedTo}(p_1, p_3) \end{array}$$

In $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ the algorithm $\mathsf{InconsistentSets}(\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F) \rangle)$ identifies the following $\mathcal{T}$-inconsistent sets:

$$\begin{array}{l} V_1 = \{\ \mathit{PortIn}(p_1),\ \mathit{PortOut}(p_1)\ \}; \\ V_2 = \{\ \mathit{connectedTo}(p_1, p_2),\ \mathit{connectedTo}(p_1, p_3)\ \}; \\ V_3 = \{\ \mathit{Port}(p_1),\ \mathit{of}(p_1, d_1),\ \mathit{Port}(p_2),\ \mathit{of}(p_2, d_1),\ \mathit{connectedTo}(p_1, p_2)\ \}; \\ V_4 = \{\ \mathit{PortIn}(p_1),\ \mathit{of}(p_1, d_1),\ \mathit{Port}(p_2),\ \mathit{of}(p_2, d_1),\ \mathit{connectedTo}(p_1, p_2)\ \}; \\ V_5 = \{\ \mathit{PortOut}(p_1),\ \mathit{of}(p_1, d_1),\ \mathit{Port}(p_2),\ \mathit{of}(p_2, d_1),\ \mathit{connectedTo}(p_1, p_2)\ \}; \\ V_6 = \{\ \mathit{of}(p_1, d_1),\ \mathit{of}(p_2, d_1),\ \mathit{connectedTo}(p_1, p_2)\ \}; \end{array}$$

where V_1 violates the disjointness $\mathit{PortIn} \sqsubseteq \neg \mathit{PortOut}$, V_2 violates the functionality assertion $(\mathit{funct}\ \mathit{connectedTo})$, and V_3, V_4, V_5, and V_6 violates the DA $\forall x, y, z.(\mathit{Port}(x) \wedge \mathit{Port}(y) \wedge \mathit{of}(x, z) \wedge \mathit{of}(y, z) \wedge \mathit{connectedTo}(x, y) \rightarrow \bot)$. Note that only the sets V_1, V_2, and V_6 are minimal $\mathcal{T}$-inconsistent sets, and that both V_2 and V_6 overlap with $\mathsf{cl}_{\mathcal{T}}(F)$.

One can verify that the only assertion α in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ for which there exists a $\mathcal{T}$-inconsistent set V in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ such that (i) $\alpha \in V$, (ii) $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent, and (iii) $V \cap \mathsf{cl}_{\mathcal{T}}(F) \neq \emptyset$, is in fact $\{\mathit{connectedTo}(p_1, p_2)\}$.

Hence, by exploiting Proposition 27 there is, up to C-equivalence, only one ABox that accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, that is:

$$\mathcal{A}_{ins} \quad = \quad \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F) \setminus \{connectedTo(p_1, p_2)\}.$$

Thus, according to Definition 40, we have:

$$\langle \mathcal{T}, \mathcal{A} \rangle \oplus^{\mathcal{T}}_{CAR} F \quad = \quad \langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F) \setminus \{connectedTo(p_1, p_2)\} \rangle.$$

□

The example above shows how one can exploit Proposition 27 with the aim of updating a possibly inconsistent KB expressed in *DL-Lite*$_{A,id,den}$ with the insertion of a $\mathcal{T}$-consistent set of facts. Thus, on the base of such a result we present the algorithm $\mathsf{ComputeInsertion}_{CAR}(\mathcal{K}, F)$ which takes in input a possibly inconsistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K}$ and a finite set of ABox assertion F, and computes the *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F$.

Input: a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a finite set of ABox assertions F
Output: a *DL-Lite*$_{A,id,den}$-KB
begin
  if F is $\mathcal{T}$-inconsistent
  then return $\mathcal{K}$;
  $W \leftarrow \mathsf{InconsistentSets}(\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F) \rangle)$;
  $D \leftarrow \emptyset$;
  foreach $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$ **do**
    if $\exists w \in W$ s.t.
      (i) $\alpha \in w$ and
      (ii) $F \cup (w \setminus \{\alpha\})$ is $\mathcal{T}$-consistent, and
      (iii) $\mathsf{cl}_{\mathcal{T}}(F) \cap w \neq \emptyset$
    then $D \leftarrow D \cup \{\alpha\}$;
  return $\langle \mathcal{T}, F \cup \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus D \rangle$;
end

Algorithm 19: *ComputeInsertion*$_{CAR}(\mathcal{K}, F)$

The algorithm essentially computes the set D of ABox assertions in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$ which do not belong to at least one ABox accomplishing the insertion of F into $\mathcal{K}$ minimally. It proceeds as follows. For exploiting Proposition 27 the algorithm needs to compute the $\mathcal{T}$-inconsistent sets in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$. To this end, $\mathsf{ComputeInsertion}_{CAR}$ uses the algorithm $\mathsf{InconsistentSets}$ that computes the set W of all $\mathcal{T}$-inconsistent sets in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$. Afterwards it adds to the set D each assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$ that is contained in at

least one $w \in W$ that overlaps with $\mathsf{cl}_{\mathcal{T}}(F)$ and such that $F \cup w \setminus \{\alpha\}$ is $\mathcal{T}$-consistent. Indeed, from Proposition 27, every assertion in D cannot appear in $\bigcap_{\mathcal{A}'_i \in \mathcal{U}} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}'_i)$, where $\mathcal{U}$ is the set containing all the ABoxes accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Taking into account such observation, finally, the algorithm returns the KB $\langle \mathcal{T}, F \cup \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus D \rangle$.

The following lemma establishes the termination the algorithm.

Lemma 54. *Let $\mathcal{K}$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and let F be a finite set of ABox assertions. Then $\mathsf{ComputeInsertion}_{CAR}(\mathcal{K}, F)$ terminates.*

Proof. Termination of $\mathsf{ComputeInsertion}_{CAR}(\mathcal{K}, F)$ directly follows from the finiteness of the sets $\mathcal{T}$, $\mathcal{A}$, and F, from Lemma 15, which establishes the termination oft $\mathsf{InconsistentSets}(\langle \mathcal{T}, \mathcal{A} \cup F \rangle)$, and from the fact that there is an algorithm for checking KB satisfiability in *DL-Lite*$_{A,id,den}$ (cf. Section 5.1). ∎

The following lemma states that we can use $\mathsf{ComputeInsertion}_{CAR}(\mathcal{K}, F)$ for computing $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F$, where $\mathcal{K}$ is a possibly inconsistent KB expressed in *DL-Lite*$_{A,id,den}$ and F is a finite set of ABox assertions.

Lemma 55. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and let F be a finite set of ABox assertions. Then*

$$\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F = \mathsf{ComputeInsertion}_{CAR}(\mathcal{K}, F).$$

Proof. The first step of the algorithm is checking if F is $\mathcal{T}$-consistent. If F is not $\mathcal{T}$-consistent, then the algorithm returns the original KB $\mathcal{K}$ according to Definition 40. Let F be $\mathcal{T}$-consistent. Suppose that the *DL-Lite*$_{A,id,den}$-KB $\langle \mathcal{T}, \mathcal{A}_u \rangle$ results from updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion of F. From Definition 40 and Proposition 22, we have that, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) = \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$, where $\mathcal{U} = \{\mathcal{A}_i, \ldots, \mathcal{A}_n$ is the set containing all the ABox accomplishing the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$. Let $\langle \mathcal{T}, \mathcal{A}' \rangle = \mathsf{ComputeInsertion}_{CAR}(\mathcal{K}, F)$. We have to prove that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$.

By analyzing the algorithm $\mathsf{ComputeInsertion}_{CAR}$, we have that $\mathcal{A}' = F \cup \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus D$. One can see that the set D contains those assertions α in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ such that: (i) there exists a $\mathcal{T}$-inconsistent set w in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$, (ii) $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent, and (iii) $\mathsf{cl}_{\mathcal{T}}(F) \cap w \neq \emptyset$. Hence, from Proposition 27, it follows that for each assertion α in D, there is an ABox $\mathcal{A}_i \in \mathcal{U}$ such that $\alpha \notin \mathcal{U}$.

We first show that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Toward a contradiction, suppose that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. This means that there is an assertions α such that $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$, but $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Since $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) = \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$, then $\alpha \in \mathcal{A}_i$ for each $\mathcal{A}_i \in \mathcal{U}$. The following cases are conceivable:

1. $\alpha \in \mathsf{cl}_{\mathcal{T}}(F)$. Since from Lemma 16 it follows that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') = \mathsf{cl}_{\mathcal{T}}(F) \cup (\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus D)$, then $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$.

2. $\alpha \notin \mathsf{cl}_{\mathcal{T}}(F)$. Proposition 26 guarantees that for every ABox $\mathcal{A}_i \in \mathcal{U}$, we have that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i) \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$. Since $\alpha \notin \mathsf{cl}_{\mathcal{T}}(F)$, then $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. This means that $\alpha \in D$, but from Proposition 27, it follows that there is an ABox $\mathcal{A}'' \in \mathcal{U}$ such that $\alpha \notin \mathcal{A}''$, Hence, $\alpha \notin \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$. Thus $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$.

Since both the cases above lead to a contradiction, we can conclude that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$.

Now, we prove that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$. Suppose, by way of contradiction, that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$. This means that there is an assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ which does not belongs to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$. The following cases are conceivable:

1. $\alpha \in \mathsf{cl}_{\mathcal{T}}(F)$. This means that $\mathsf{cl}_{\mathcal{T}}(F) \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$, and then for each *CA*-repair $\mathcal{A}_{rep}$ of $\langle \mathcal{T}, \mathcal{A}_u \rangle$, we have that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \not\models F$. It follows that $\langle \mathcal{T}, \mathcal{A}_u \rangle \not\models_{CAR} F$, which contradicts Proposition 21.

2. $\alpha \notin \mathsf{cl}_{\mathcal{T}}(F)$. This means that $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus D$. From Proposition 27, it follows that for every ABox $\mathcal{A}_i \in \mathcal{U}$, $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$. Hence, $\alpha \in \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$, and therefore $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$. Which is a contradiction.

So, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$, which concludes the proof. ∎

Finally, based on the algorithm presented above, we analyze the computational complexity of updating a possibly inconsistent *DL-Lite*$_{A,id,den}$-KB according to the semantics given in Definition 40.

Theorem 45. *ComputeInsertion*$_{CAR}(\langle \mathcal{T}, \mathcal{A} \rangle, F)$ *computes* $\mathcal{K} \oplus^{\mathcal{T}}_{CAR} F$ *in polynomial time with respect to* $|\mathcal{A}|$ *and* $|F|$*, and in exponential time with respect to* $|\mathcal{T}|$*.*

Proof. The proof follows directly from Lemma 54, Lemma 55, and from the observations below:

- The first step of the algorithm is checking if $\langle \mathcal{T}, F \rangle$ is consistent. Theorem 7 guarantees that this step can be done in exponential time with respect to the size of $\mathcal{T}$ and in polynomial time with respect to the size of F.

- The second step of the algorithm is computing the set W by means of InconsistentSets. $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ and $\mathsf{clc}_{\mathcal{T}}(F)$ can be computed in polynomial time with respect to $|\mathcal{T}|$ and in polynomial time with respect to $|\mathcal{A}|$, and with respect to $|F|$ respectively. Since $\mathcal{Q}^{unsat}_{\mathcal{T}}$ can be computed in exponential time with respect to $|\mathcal{T}|$, then InconsistentSets$(\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F) \rangle)$ runs in exponential time with respect to the size of $\mathcal{T}$ and in polynomial time with respect to the size of $\mathcal{A} \cup F$,

- The size of W is polynomial with respect to $|\mathcal{T} \setminus \mathcal{T}_{id} \cup \mathcal{T}_{den}|$, $|\mathcal{A}|$ and $|F|$, and exponential with respect to $|\mathcal{T}_{id} \cup \mathcal{T}_{den}|$; moreover, the size of each set $w \in W$ is polynomial in the size of $\mathcal{T}$.
- For each $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$ and for each $w \in W$ deciding if $F \cup (w \setminus \{\alpha\})$ is $\mathcal{T}$-consistent can be done in polynomial time with respect to $|F \cup (w \setminus \{\alpha\})|$ and in exponential time with respect to $|\mathcal{T}|$;
- the size of D is polynomial with respect to the size of $\mathcal{A}$;
- for each $\alpha \in D$ the cost of eliminating α from $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ is polynomial in the size of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

■

11.2 Inconsistency-tolerant deletion in *DL-Lite*$_{A,id,den}$

In this section we study the problem of updating a possibly inconsistent KB expressed in *DL-Lite*$_{A,id,den}$ with the deletion of a set of ABox assertions according to the semantics given in Section 10.1. Thus, in what follows, we implicitly refer to a *DL-Lite*$_{A,id}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

Let F be a finite set of ABox assertions, Proposition 20 guarantees that if $\langle \mathcal{T}, \emptyset \rangle \not\models F$, then there exists an ABox that accomplishes the deletion of F from $\mathcal{K}$. As a consequence of Lemma 16, we have that, for every *DL-Lite*$_{A,id,den}$-KB $\mathcal{K}$, and for every set F, an ABox accomplishing the deletion of F from $\mathcal{K}$ always exists. Hence, we have the following lemma.

Lemma 56. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and let F be a finite set of ABox assertions. An ABox accomplishing the deletion of F from $\mathcal{K}$ always exists.*

Proof. Let $\mathcal{U}$ be the set of ABoxes accomplishing the deletion of F from $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. By way of contradiction, suppose that $\mathcal{U} = \emptyset$. From Proposition 20 it follows that the KB $\mathcal{K}' = \langle \mathcal{T}, \emptyset \rangle \not\models F$, but from Lemma 16, we have that for every assertion $\alpha \in F$, there exists in the ABox of $\mathcal{K}'$ an assertion α' such $\langle \mathcal{T}, \{\alpha'\} \rangle \models \alpha$. Hence the ABox of $\mathcal{K}'$ is not empty, which is a contradiction. ■

The lemma above actually guarantees that the set $\mathcal{U}$ containing the ABoxes that accomplish the deletion of F from $\mathcal{K}$ is non-empty.

By Definition 37 we have that an ABox $\mathcal{A}'$ accomplishes the deletion of F from $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if every *CA*-repair of $\langle \mathcal{T}, \mathcal{A}' \rangle$ does not imply F. As stated by Lemma 53, in the case where F is $\mathcal{T}$-inconsistent, the original ABox $\mathcal{A}$ accomplishes the deletion of F from $\mathcal{K}$ minimally. Moreover, Lemma 52 guarantees that if $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then $\mathcal{A}$ accomplishes the deletion of F from $\mathcal{K}$.

From the observations above directly follows that if F is $\mathcal{T}$-inconsistent, or $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then the deletion operator does not modify the original KB. Thus, in what follows we focus on the case where F is $\mathcal{T}$-consistent.

We start discussing the case where the set F is constituted by a single ABox assertion f. Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent *DL-Lite*$_{A,id,den}$-KB, and let $\mathcal{A}'$ be an ABox. From Theorem 44, if $\mathcal{A}'$ accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, then $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Therefore, by exploiting Lemma 16, we can compute $\mathcal{A}'$ by removing from $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ every assertion α such that $\langle \mathcal{T}, \alpha \rangle \models f$. Note that from the definition of *consistent logical consequences* given in Section 8.1, we have that $\{\alpha\}$ is $\mathcal{T}$-consistent for every ABox assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$.

Now, we turn our attention to the case of arbitrary set F, i.e., the case where $F = \{f_1, \ldots, f_m\}$, for $m \geq 0$. Let $\mathcal{U} = \{\mathcal{A}_1 \ldots \mathcal{A}_m\}$ be a set of ABoxes $\mathcal{A}_i$, such that, for every $1 \leq i \leq m$, $\mathcal{A}_i$ accomplishes the deletion of $\{f_i\} \subseteq F$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. The following lemmas are the key to our solution.

Lemma 57. *Let $\mathcal{A}_i$ and $\mathcal{A}_j$ be two ABoxes in $\mathcal{U}$ such that $\mathcal{A}_i$ and $\mathcal{A}_j$ accomplishes respectively the deletion of $\{f_i\} \subseteq F$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, and the deletion of $\{f_i\} \subseteq F$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$ has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j)$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ if and only if $\langle \mathcal{T}, \{f_i\} \rangle \models f_j$ and $\langle \mathcal{T}, \{f_j\} \rangle \not\models f_i$.*

Proof.
($\Rightarrow$) We start proving that if $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$ has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j)$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then $\langle \mathcal{T}, \{f_i\} \rangle \models f_j$ and $\langle \mathcal{T}, \{f_j\} \rangle \not\models f_i$.

Suppose, by way of contradiction, that $\langle \mathcal{T}, \{f_i\} \rangle \not\models f_j$. Since $\mathcal{A}_i$ accomplishes the deletion of $\{f_i\}$ from $\mathcal{K}$, then $f_i \notin \mathcal{A}_i$. Analogously, $f_i \notin \mathcal{A}_j$. Theorem 44 guarantees that both $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$ and $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j)$ are subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Hence, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$ has fewer changes than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j)$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ since $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$ has fewer deletions than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j)$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. This means that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i) \subset \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j)$. It follows that both f_i and f_j does not belongs to $\mathcal{A}_j$. Consider the ABox $\mathcal{A}_j \cup \{f_i\}$. Since $\mathcal{A}_j$ accomplishes the deletion of $\{f_j\}$ from $\mathcal{K}$ minimally, then $\mathcal{A}_j \cup \{f_i\}$ cannot accomplish the deletion of $\{f_j\}$ from $\mathcal{K}$. Hence, there is a *CAR*-repair $\mathcal{A}_{rep}$ of $\langle \mathcal{T}, \mathcal{A}_j \cup \{f_i\} \rangle$ such that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \models f_j$. But this means that $f_j \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j) \cup \mathsf{cl}_{\mathcal{T}}(f_i)$. Since $f_j \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_j)$, then, from Lemma 16, it follows that $\langle \mathcal{T}, \{f_i\} \rangle \models f_j$. Hence, we have a contradiction. Now, suppose, by contradiction, that $\langle \mathcal{T}, \{f_j\} \rangle \models f_i$. There are two possible cases. (*i*) $\{f_j\} \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$. From Lemma 52, it follows that $\mathcal{A}_i$ does not accomplish the deletion of $\{f_j\}$ from $\mathcal{K}$, and then there is a *CAR*-repair $\mathcal{A}_{rep}$ of $\mathcal{A}_i$ such that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \models f_j$, but this means that $\langle \mathcal{T}, \mathcal{A}_{rep} \rangle \models f_i$. It follows that $\mathcal{A}_i$ does not accomplish the deletion of $\{f_i\}$ from $\mathcal{K}$, which is a contradiction. (*ii*) $\{f_j\} \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$. It follows from Lemma 52, that $\mathcal{A}_i$ accomplishes the deletion of $\{f_j\}$ from $\mathcal{K}$, but since $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$ has fewer

changes than $\mathsf{clc}_\mathcal{T}(\mathcal{A}_j)$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$, $\mathcal{A}_j$ does not accomplish the deletion of F from $\mathcal{K}$ minimally, which is a contradiction. Hence. one can conclude that $\langle \mathcal{T}, \{f_i\}\rangle \models f_j$ and $\langle \mathcal{T}, \{f_j\}\rangle \not\models f_i$.

($\Leftarrow$) We have to prove that if $\langle \mathcal{T}, \{f_i\}\rangle \models f_j$ and $\langle \mathcal{T}, \{f_j\}\rangle \not\models f_i$, then $\mathsf{clc}_\mathcal{T}(\mathcal{A}_i)$ has fewer changes than $\mathsf{clc}_\mathcal{T}(\mathcal{A}_j)$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. Toward a contradiction, suppose that $\mathsf{clc}_\mathcal{T}(\mathcal{A}_i)$ has not fewer changes than $\mathsf{clc}_\mathcal{T}(\mathcal{A}_j)$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. The following two cases are conceivable:

- $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{clc}_\mathcal{T}(\mathcal{A}_i) \not\subseteq \mathsf{clc}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{clc}_\mathcal{T}(\mathcal{A}_j)$. This means that there is an assertions $\alpha \in \mathsf{clc}_\mathcal{T}(\mathcal{A})$, such that $\alpha \in \mathsf{clc}_\mathcal{T}(\mathcal{A}_j)$ and $\alpha \notin \mathsf{clc}_\mathcal{T}(\mathcal{A}_i)$. Since $\mathcal{A}_i$ accomplishes the deletion of $\{f_i\}$ from $\mathcal{K}$ minimally, it follows from Theorem 44 that $\langle \mathcal{T}, \{\alpha\}\rangle \models f_i$, but since $\langle \mathcal{T}, \{f_i\}\rangle \models f_j$ then $\langle \mathcal{T}, \{\alpha\}\rangle \models f_j$. Hence, $f_j \in \mathsf{clc}_\mathcal{T}(\mathcal{A}_j)$, which is a contradiction.

- $\mathsf{clc}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{clc}_\mathcal{T}(\mathcal{A}_i) = \mathsf{clc}_\mathcal{T}(\mathcal{A}) \setminus \mathsf{clc}_\mathcal{T}(\mathcal{A}_j)$. Since from Lemma 52 it follows that $f_j \notin \mathsf{clc}_\mathcal{T}(\mathcal{A}_j)$, then $f_j \notin \mathsf{clc}_\mathcal{T}(\mathcal{A}_i)$. But since $\langle \mathcal{T}, \{f_j\}\rangle \not\models f_i$, from Lemma 16 it follows that the ABox $f_i \notin \mathsf{clc}_\mathcal{T}(\mathcal{A}_i \cup \{f_j\})$. Hence, from Lemma 52, $\mathcal{A}_i \cup \{f_j\}$ accomplishes the deletion of $\{f_i\}$ from $\mathcal{K}$, moreover, $\mathsf{clc}_\mathcal{T}(\mathcal{A}_i) \cup \mathsf{clc}_\mathcal{T}(\{f_j\})$ has fewer deletions than $\mathsf{clc}_\mathcal{T}(\mathcal{A}_i)$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. Hence, $\mathcal{A}_i$ does not accomplish the deletion of $\{f_i\}$ from $\mathcal{K}$ minimally, which is a contradiction.

From the observation above, we can conclude that $\mathsf{clc}_\mathcal{T}(\mathcal{A}_i)$ has fewer changes than $\mathsf{clc}_\mathcal{T}(\mathcal{A}_j)$ with respect to $\mathsf{clc}_\mathcal{T}(\mathcal{A})$. ∎

As a consequence of Lemma 57 we have that if an ABox $\mathcal{A}_i$ accomplishes the deletion of $\{f_i\} \subseteq F$ from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally, and there does not exist any other assertion $f_j \neq f_i$ in F such that $\langle \mathcal{T}, \{f_j\}\rangle \models f_i$ and $\langle \mathcal{T}, \{f_i\}\rangle \not\models f_j$, then $\mathcal{A}_i$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally.

The following lemma guarantees that no ABox, other than those contained in $\mathcal{U}$, accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally.

Lemma 58. *Let $\langle \mathcal{T}, \mathcal{A}\rangle$ be a DL-Lite$_{A,id}$-KB, and let F be a $\mathcal{T}$-consistent set of ABox assertions such that $F \subseteq \mathsf{clc}_\mathcal{T}(\mathcal{A})$. If an ABox $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally, then there exists an assertion $f' \in F$ such that $\mathcal{A}'$ accomplishes the deletion of $\{f'\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally.*

Proof. Since $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A}\rangle$, and since F is $\mathcal{T}$-consistent, then, from Lemma 52, it follows that $F \not\subseteq \mathsf{clc}_\mathcal{T}(\mathcal{A}')$. This means that there is at least one assertion $f' \in F$ such that $\{f'\} \not\subseteq \mathsf{clc}_\mathcal{T}(\mathcal{A}')$, so, again from Lemma 52, it follows that $\mathcal{A}'$ accomplishes the deletion of $\{f'\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$. It remains to prove that $\mathcal{A}'$ does it minimally. Suppose, by way of contradiction, that $\mathcal{A}'$ does not accomplish the deletion of $\{f'\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally. From Theorem 44 it follows that $\mathsf{clc}_\mathcal{T}(\mathcal{A}') \subseteq \mathsf{clc}_\mathcal{T}(\mathcal{A})$. Then, there

exists an assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ different to f' that does not belong to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, and such that $\mathcal{A}' \cup \{\alpha\}$ accomplishes the deletion of $\{f'\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$. Hence, $\mathcal{A}' \cup \{\alpha\}$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}' \cup \alpha)$ has fewer deletions than $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ with respect to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Therefore, it is not true that $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, which is a contradiction. ∎

Lemma 57, and Lemma 58 suggest a direct strategy for computing the result of updating a possibly inconsistent *DL-Lite*$_{A,id}$-KB with the deletion of a set of ABox assertions. Such a strategy is illustrated in the algorithm $\mathsf{ComputeDeletion}_{CAR}$.

Input: a *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a finite set of ABox assertions F
Output: a *DL-Lite*$_{A,id}$-KB
begin
  if $Mod(\langle \mathcal{T}, F \rangle) = \emptyset$ or $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$
  then return $\langle \mathcal{T}, \mathcal{A} \rangle$;
  $F' \leftarrow F$;
  foreach $f_i \in F'$ and $f_j \in F$ such that $f_i \neq f_j$ **do**
    if $\langle \mathcal{T}, \{f_j\} \rangle \models f_i$ and $\langle \mathcal{T}, \{f_i\} \rangle \not\models f_j$
    then $F' \leftarrow F' \setminus \{f_i\}$;
  $F^- \leftarrow \emptyset$;
  foreach $f \in F'$ **do**
    foreach $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ **do**
      if $\langle \mathcal{T}, \{\alpha\} \rangle \models f$
      then $F^- \leftarrow F^- \cup \{\alpha\}$;
  return $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus F^- \rangle$;
end

Algorithm 20: $\mathsf{ComputeDeletion}_{CAR}(\mathcal{K}, F)$

The algorithm takes in input a possibly inconsistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a finite set of ABox assertions F, and returns the *DL-Lite*$_{A,id,den}$-KB $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F$. Essentially, it proceed as follows. Firstly, it checks if F is $\mathcal{T}$-inconsistent, or if $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. If one of such checks succeed, then the algorithm returns the original KB $\mathcal{K}$. On the other hand, if F is $\mathcal{T}$-consistent and $F \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then the algorithm computes the set F' in the following manner: for each assertion α in F, if there does not exist in $F \setminus \{\alpha\}$ an assertion β such that $\langle \mathcal{T}, \{\beta\} \rangle \models \alpha$ and $\langle \mathcal{T}, \{\alpha\} \rangle \not\models \beta$, then it adds to F' the assertion α. Then, the algorithm computes the set F^- by adding to F' every assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \{\alpha\} \rangle \models \beta$, where β is an assertion in F'. Finally, the algorithm returns the KB $\langle \mathcal{T}, \mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus F^- \rangle$.

The following lemma guarantees that the algorithm $\mathsf{ComputeDeletion}_{CAR}$ terminates, when applied to a *DL-Lite*$_{A,id,den}$-KB an a finite set of ABox assertions.

Lemma 59. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and let F be a finite set of ABox assertions. Then, $\mathsf{ComputeDeletion}_{CAR}(\mathcal{K}, F)$ terminates, and runs in polynomial time with respect to the size of $\mathcal{A}$ and F, and in exponential time with respect to the size of $\mathcal{T}$.*

Proof. Since F is finite, then the proof follows from the following facts:

(1) Theorem 7 guarantees that deciding if F is $\mathcal{T}$-consistent can be done in exponential time with respect to $|\mathcal{T}|$ and in polynomial time with respect to $|F|$.

(2) $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ can be computed in polynomial time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$.

(3) Deciding if $F \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ can be done in polynomial time with respect to $|F|$ and $|\mathsf{clc}_{\mathcal{T}}(\mathcal{A})|$.

(4) By assuming that F is $\mathcal{T}$-consistent, for each pair of assertions f_i and f_j, deciding if $\langle \mathcal{T}, \{f_j\} \rangle \models f_i$ and $\langle \mathcal{T}, \{f_i\} \rangle \not\models f_j$, can be done in polynomial time with respect to $|\mathcal{T}|$ and $|F|$.

(5) By assuming that F is $\mathcal{T}$-consistent, for each assertion f in F, and for each assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, deciding if $\langle \mathcal{T}, \{\alpha\} \rangle \models f$ can be done in polynomial time with respect to $|\mathcal{T}|$ and $|F|$. Hence, the set F^+ can be computed in polynomial time with respect to $|\mathsf{clc}_{\mathcal{T}}(\mathcal{A})|$, $|F|$, and $|\mathcal{T}|$.

(6) $|F^+| \leq |\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)|$.

(7) $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) \setminus F^-$ can be computed in polynomial time with respect to $|\mathsf{clc}_{\mathcal{T}}(\mathcal{A})|$ and $|F^+|$.

The points above imply that each step of the algorithm terminates and that, in the worst case, it can be carried out in exponential time with respect to the size of $\mathcal{T}$, and in polynomial time with respect to the size of $\mathcal{A}$ and F. ∎

The following lemma states that, given a possibly inconsistent *DL-Lite*$_{A,id,den}$-KB $\mathcal{K}$ and a finite set of ABox assertions F, $\mathsf{ComputeDeletion}_{CAR}$ can be used for computing $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F$.

Lemma 60. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB, and let F be a finite set of ABox assertions. Then*

$$\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F = \mathsf{ComputeDeletion}_{CAR}(\mathcal{K}, F).$$

Proof. Let $\mathcal{U} = \{\mathcal{A}_i, \ldots, \mathcal{A}_n$ be the set containing all the ABox accomplishing the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. From Lemma 56, it directly follows that $\mathcal{U} \neq \emptyset$. Suppose that the *DL-Lite*$_{A,id,den}$-KB $\langle \mathcal{T}, \mathcal{A}_u \rangle$ is the result of updating $\langle \mathcal{T}, \mathcal{A} \rangle$ with the deletion of F. From Definition 41, and Proposition 22 it follows that, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) = \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$. Hence, if $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the KB computed by $\mathsf{ComputeDeletion}_{CAR}(\mathcal{K}, F)$, then we have to prove that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. We start by proving that if $Mod(\langle \mathcal{T}, F \rangle) = \emptyset$ or $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$ then $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. If $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, then Lemma 53 guarantees that $\mathcal{A}$ accomplishes the deletion of F minimally. Since Theorem 44 states that for every ABox $\mathcal{A}' \in \mathcal{U}$ is a subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then, for every ABox $\mathcal{A}' \in \mathcal{U}$, $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') = \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. It follows that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) = \bigcap_{\mathcal{A}_i \in \mathcal{U}} \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$. If $F \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, then Lemma 52 guarantees that $\mathcal{A}$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$. Clearly, it does this minimally. From this, as shown above, it follows that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}) = \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$. Let $Mod(\langle \mathcal{T}, F \rangle) \neq \emptyset$ and $F \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. We first prove that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Suppose, by way of contradiction, that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u) \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. This means that there is in $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$ an assertions α that does not belong to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$. Hence, $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$ for each ABox $\mathcal{A}_i \in \mathcal{U}$. Theorem 44 states that for every ABox $\mathcal{A}_i \in \mathcal{U}$ we have that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}_i)$ is a maximal subset of $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, therefore $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Since $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, then, by observing the algorithm, there is in F an assertion f such that: (i) there does not exist in $F \setminus \{f\}$ an assertion f' such that $\langle \mathcal{T}, \{f'\} \rangle \models f$ and $\langle \mathcal{T}, \{f\} \rangle \not\models f'$; (ii) $\langle \mathcal{T}, \{\alpha\} \rangle \models f$. From Lemma 57 and Lemma 58, it follows that there is an ABox $\mathcal{A}''$ that accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, that accomplishes also the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Hence, $\mathcal{A}'' \in \mathcal{U}$, and then $f \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$. Since $\langle \mathcal{T}, \{\alpha\} \rangle \models f$, then $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$, which is a contradiction.

We now show that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$. Suppose, by way of contradiction, that $\mathsf{clc}_{\mathcal{T}}(\mathcal{A}') \not\subseteq \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$. Therefore, there is an assertion $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$ such that $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$. Since, the algorithm does not add any assertion to $\mathsf{clc}_{\mathcal{T}}(\mathcal{A})$, it follows that $\alpha \in \mathsf{clc}_{\mathcal{T}}(\mathcal{A})$. Since $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}_u)$, then there is an ABox $\mathcal{A}'' \in \mathcal{U}$ such that $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}'')$. From Lemma 58 it follows that there is in F and assertion f such that $\langle \mathcal{T}, \{\alpha\} \rangle \models f$ and $\mathcal{A}''$ accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. Lemma 57, together with Lemma 58, guarantees that there is not an assertion f' in F such that $\langle \mathcal{T}, \{f'\} \rangle \models f$ and $\langle \mathcal{T}, \{f\} \rangle \not\models f'$. Hence, the algorithm adds f to the set F'. Consequently, since $\langle \mathcal{T}, \{\alpha\} \rangle \models f$, the algorithm adds α to the set F^-. Hence $\alpha \notin \mathsf{clc}_{\mathcal{T}}(\mathcal{A}')$, which is a contradiction. ∎

Lemma 60 and Lemma 59 allow us to establish the complexity of computing the KB resulting from the update of a possibly inconsistent KB expressed in *DL-Lite*$_{A,id,den}$ with the deletion of a finite set of ABox assertions.

Theorem 46. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a possibly inconsistent DL-Lite$_{A,id,den}$-KB,*

and let F be a finite set of ABox assertions. $\mathsf{ComputeDeletion}_{CAR}(\langle\mathcal{T},\mathcal{A}\rangle, F)$ *computes* $\mathcal{K} \ominus^{\mathcal{T}}_{CAR} F$ *in polynomial time with respect to the size of* $\mathcal{A}$ *and* F*, and in exponential time with respect to the size of* $\mathcal{T}$*.*

Proof. The proof directly follows from Lemma 60 and Lemma 59. ∎

Conclusions

In this dissertation, we have studied several issues related to the problem of equipping Ontology-based Information System with ASK and TELL operations, in both the cases where the KB underlying the systems is consistent and inconsistent. In particular, we have focussed on Stand-alone Ontology-based Information Systems (SOISs).

We have proposed a new DL, member of the *DL-Lite* family, particularly tailored for representing complex domains of interest. We have studied the problem of detecting whether a knowledge base expressed in this language is inconsistent, and we have addressed the problem of answering queries posed over KBs expressed in this language.

We have proposed a new semantics for updating consistent KBs at the instance level with both the insertion and deletion of a set of facts, and we have studied the problem of applying this update semantics to DL KBs. Furthermore, we have proposed and analyzed four inconsistency-tolerant semantics, showing that two of these semantics are particularly suitable for performing consistent query answering over KB expressed in tractable languages. Finally, we have proposed a semantics for updating inconsistent KBs, and we have provided algorithms for updating possibly inconsistent *DL-Lite*-KBs according to such semantics.

To be more precise, we have presented the DL $DL\text{-}Lite_{A,id,den}$ of the *DL-Lite* family, which extends the DL $DL\text{-}Lite_{A,id}$ with denial assertions, a new kind of assertions particularly useful to specify general forms of disjointness. We have shown that both KB satisfiability and query answering are FOL-rewritable in this language. Thus, we have shown that both these problems can be managed efficiently with respect to the size of the ABox, and that they can be reduced to the evaluation of a first-order query over a database.

Afterwards, we have proposed formula-based update semantics for updating a consistent DL KB through both the insertion and the deletion of ABox assertions. We have first shown that these update semantics possess several notable properties, and then we have provided a discussion on how our update approach relates to the classical update postulates. Moreover, we have provided polynomial algorithms for updating KBs expressed in $DL\text{-}Lite_{A,id,den}$.

Then, we focused on the problem of dealing with inconsistencies in DL KBs. In particular, we have adopted the strategy of leaving the KBs unchanged, and trying to obtain consistent information during the query answering. Thus, by embracing this idea, we present four inconsistency-tolerant semantics: the *ABox Repair* semantics (*AR*-semantics), the *Closed ABox Repair* (*CAR*-semantics), the *Intersection ABox Repair* semantics (*IAR*-semantics), and the *Intersection Closed ABox Repair* semantics (*ICAR*-semantics).In this regard, we have provided the general conditions under which consistent query answering, by adopting one of these semantics, is decidable. Moreover, we provide FOL-rewriting procedures for answering boolean conjunctive queries posed over possibly inconsistent *DL-Lite*$_{A,id,den}$-KBs under both *IAR* and *ICAR*-semantics.

Finally, we have studied the problem of updating possibly inconsistent KBs. So, we have proposed inconsistency-tolerant update semantics for both insertion and deletion. After having discussed the main properties of such semantics, we have studied their application in the context of KBs expressed in *DL-Lite*$_{A,id,den}$, and we have provided algorithms for updating a possibly inconsistent *DL-Lite*$_{A,id,den}$-KB with both the insertion and the deletion of a set of ABox assertions. We have proven that both the algorithms compute the result of the update in polynomial time with respect the size of the ABox.

There are several interesting directions for continuing our research.

For what concerns the update, in both the cases of consistent and inconsistent KBs, it would be interesting to study the problem in the context of Ontology-based Data Integration Systems (ODISs), i.e., study the problem of updating systems where a domain description given by means of a DL TBox (ontology) is used to mediate the access to data sources. As we have shown, in dealing with update in case of classic KB, the major challenges are related to inconsistency management and to the expressibility problem. When the update problem is faced while dealing with ODISs, the major challenge is how modify the data source in order to reflect the changes. Moreover, we plan to study the problem of updating both KB and ODISs where the intensional level is expressed in more expressive languages. In particular, we are very interested in the tractable fragments of OWL. Concerning the problem of updating possibly inconsistent KBs, it would be interesting to study new semantics based on inconsistency-tolerant semantics that differ from *CAR*-semantics.

For what concerns inconsistency-tolerant query answering, we aim at finding a more efficient technique for rewriting conjunctive queries under both *IAR* and *ICAR*-semantics. Moreover, it is our intention to study the integration of such types of query rewriting with the query rewriting technique adopted in Ontology Based Data Access. In fact, we experienced several issues regarding the process of rewriting the queries posed over the KB into queries over the data sources, and we are interested in studying how these two different

rewriting steps work together.

Finally, we plan to integrate the results of our investigation on update and consistent query answering, in Mastro [26], which is a Java tool for ontology-based data access developed by our research group. Mastro manages OBDA systems in which the ontology is specified in $DL\text{-}Lite_{A,id,den}$ and is connected to external JDBC enabled data management systems through semantic mappings that associate SQL queries over the external data to the elements of the ontology.

Bibliography

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995. 12, 20, 23, 26, 30, 57, 61, 185, 202

[2] Andrea Acciarri, Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Mattia Palmieri, and Riccardo Rosati. QuOnto: QUerying ONTOlogies. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, pages 1670–1671, 2005. 152

[3] 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. 72, 90, 99

[4] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proceedings of the Eighteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'99)*, pages 68–79, 1999. 119, 153, 154

[5] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Specifying and querying database repairs using logic programs with exceptions. In *Proceedings of the Fourth International Conference on Flexible Query Answering Systems (FQAS 2000)*, pages 27–41. Springer, 2000. 154

[6] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The *DL-Lite* family and relations. *Journal of Artificial Intelligence Research*, 36:1–69, 2009. 25

[7] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003. 12

[8] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook:*

Theory, Implementation and Applications. Cambridge University Press, 2nd edition, 2007. 5, 145

[9] Kenneth Baclawski, Mieczyslaw M. Kokar, Richard Waldinger, and Paul A. Kogut. Consistency checking of semantic web ontologies. In *Proceedings of the First International Semantic Web Conference (ISWC 2002)*, 2002. 151

[10] Daniela Berardi, Andrea Calì, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams. Technical Report 11-03, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 2003. 25

[11] Philip A. Bernstein and Laura Haas. Information integration in the enterprise. *Communications of the ACM*, 51(9):72–79, 2008. 4

[12] Leopoldo E. Bertossi, Anthony Hunter, and Torsten Schaub, editors. *Inconsistency Tolerance*, volume 3300 of *Lecture Notes in Computer Science*. Springer, 2005. 119

[13] Meghyn Bienvenu. First-order expressibility results for queries over inconsistent *DL-Lite* knowledge bases. In *Proceedings of the Twentyfourth International Workshop on Description Logic (DL 2011)*, volume 745 of *CEUR Electronic Workshop Proceedings,* http://ceur-ws.org/, 2011. 154

[14] Meghyn Bienvenu. Inconsistency-tolerant conjunctive query answering for simple ontologies. In *Proceedings of the Twentyfifth International Workshop on Description Logic (DL 2012)*, volume 846 of *CEUR Electronic Workshop Proceedings,* http://ceur-ws.org/, 2012. 155, 156

[15] Alexander Borgida. Language features for flexible handling of exceptions in information systems. *ACM Transactions on Database Systems*, 10(4):565–603, 1985. 96

[16] François Bry. Query answering in information systems with integrity constraints. In *IFIP WG 11.5 Working Conf. on Integrity and Control in Information System*. Chapman & Hall, 1997. 154

[17] Marco Cadoli, Francesco M. Donini, Paolo Liberatore, and Marco Schaerf. The size of a revised knowledge base. *Artif. Intell.*, 115(1):25–64, 1999. 83

[18] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Accessing data integration systems through conceptual

schemas. In *Proceedings of the Twentieth International Conference on Conceptual Modeling (ER 2001)*, volume 2224 of *Lecture Notes in Computer Science*, pages 270–284. Springer, 2001. 4

[19] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. In *Proceedings of the Fourteenth International Conference on Advanced Information Systems Engineering (CAiSE 2002)*, volume 2348 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2002. 4

[20] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. In *Proceedings of the Twentyeighth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2009)*, pages 77–86, 2009. 41

[21] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics*, 14:57–83, 2012. 156

[22] Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proceedings of the Twentysecond ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2003)*, pages 260–271, 2003. 129, 154

[23] Andrea Calì, Domenico Lembo, and Riccardo Rosati. Query rewriting and answering under constraints in data integration systems. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 16–21, 2003. 154

[24] D. Calvanese and G. De Giacomo. Data integration: A logic-based perspective. *AI magazine*, 26(1):59, 2005. 4

[25] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, and R. Rosati. Ontologies and databases: The *DL-Lite* approach. *Reasoning Web. Semantic Technologies for Information Systems*, pages 255–356, 2009. 25, 26, 30, 33, 34, 35, 36, 60

[26] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The Mastro system for ontology-based data access. *Semantic Web Journal*, 2(1):43–53, 2011. 120, 151, 159, 207, 245

[27] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. Ontology-based database access. In *Proceedings of the Fifteenth Italian Conference on Database Systems (SEBD 2007)*, pages 324–331, 2007. 21

[28] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proceedings of the Eighteenth International Workshop on Description Logic (DL 2005)*, volume 147 of *CEUR Electronic Workshop Proceedings*, http://ceur-ws.org/, 2005. 25, 26

[29] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. *DL-Lite*: Tractable description logics for ontologies. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005. 25

[30] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proceedings of the Tenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006. 5, 41

[31] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated Reasoning*, 39(3):385–429, 2007. 5, 7, 8, 25, 26, 29, 30, 31, 33, 34, 35, 47, 48, 49, 50, 54, 55, 56, 57, 101, 130, 159, 160, 162, 164, 173, 196, 197

[32] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Path-based identification constraints in description logics. In *Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 231–241, 2008. 26, 48, 49, 56

[33] Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Dmitriy Zheleznyakov. Evolution of *DL-Lite* knowledge bases. In *Proceedings of the Ninth International Semantic Web Conference (ISWC 2010)*, volume 6496 of *Lecture Notes in Computer Science*, pages 112–128. Springer, 2010. 6, 69, 71, 83, 96, 97, 100, 102

[34] Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Dmitriy Zheleznyakov. Updating ABoxes in *DL-Lite*. In *Proceedings of the Fourth*

Alberto Mendelzon International Workshop on Foundations of Data Management (AMW 2010), volume 619 of *CEUR Electronic Workshop Proceedings*, http://ceur-ws.org/, pages 3.1–3.12, 2010. 83

[35] B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins. What are ontologies, and why do we need them? *Intelligent Systems and Their Applications, IEEE*, 14(1):20–26, 1999. 5

[36] Jan Chomicki. Consistent query answering: Five easy pieces. In *Proceedings of the Eleventh International Conference on Database Theory (ICDT 2007)*, volume 4353 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2007. 8, 119, 153, 154, 207

[37] Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. Technical Report cs.DB/0212004 v1, arXiv.org e-Print archive, December 2002. Available at http://arxiv.org/abs/cs/0212004. 153

[38] Jan Chomicki and Jerzy Marcinkowski. On the computational complexity of minimal-change integrity maintenance in relational databases. In Leopoldo Bertossi, Anthony Hunter, and Torsten Schaub, editors, *Inconsistency Tolerance*, volume 3300 of *Lecture Notes in Computer Science*, pages 119–150. Springer, 2005. 152, 153

[39] Jan Chomicki, Jerzy Marcinkowski, and Slawomir Staworko. Computing consistent query answers using conflict hypergraphs. In *Proceedings of the Thirteenth International Conference on Information and Knowledge Management (CIKM 2004)*, pages 417–426, 2004. 154

[40] Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. On instance-level update and erasure in description logic ontologies. *Journal of Logic and Computation, Special Issue on Ontology Dynamics*, 19(5):745–770, 2009. 6, 36, 37, 66, 69, 70, 71, 96, 97

[41] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Li Ma, Edith Schonberg, Kavitha Srinivas, and Xingzhi Sun. Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In *Proceedings of the Seventh International Semantic Web Conference (ISWC 2008)*, volume 5318 of *Lecture Notes in Computer Science*, pages 403–418. Springer, 2008. 5, 7, 159

[42] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992. 96, 131, 152, 208

[43] S.M. Embury, S.M. Brandt, J.S. Robinson, I. Sutherland, F.A. Bisby, W.A. Gray, A.C. Jones, and R.J. White. Adapting integrity enforcement techniques for data reconciliation. *Information Systems*, 26(8):657–689, 2001. 153

[44] Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the semantics of updates in databases. In *Proceedings of the Second ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS'83)*, pages 352–365, 1983. 71, 73, 76, 78, 82, 83, 99, 121, 152, 154, 209

[45] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou. Ontology change: Classification and survey. *Knowledge Engineering Review*, 23(2):117–152, 2008. 6, 69

[46] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. On applying the AGM theory to DLs and OWL. In *Proceedings of the Fourth International Semantic Web Conference (ISWC 2005)*, 2005. 72, 90

[47] Kenneth D. Forbus. Introducing actions into qualitative simulation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1273–1278, 1989. 97

[48] Ariel Fuxman and Renée J. Miller. First-order query rewriting for inconsistent databases. In *Proceedings of the Tenth International Conference on Database Theory (ICDT 2005)*, volume 3363 of *LNCS*, pages 337–351. Springer, 2005. 154

[49] P. Gärdenfors and H. Rott. Belief revision. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 35–132. Oxford University Press, 1995. 207

[50] M. R. Garey and D. S. Johnson. *Computers and Intractability — A guide to NP-completeness*. W. H. Freeman and Company, San Francisco (CA, USA), 1979. 11

[51] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. Technical Report KSL-86-65, Knowledge Systems, AI Laboratory, 1987. 83, 218

[52] M.L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30(1):35–79, 1986. 82, 83, 99, 218

[53] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1993. 4

[54] Laura M. Haas. Beauty and the beast: The theory and practice of information integration. In *Proceedings of the Eleventh International Conference on Database Theory (ICDT 2007)*, volume 4353 of *Lecture Notes in Computer Science*, pages 28–43. Springer, 2007. 3

[55] Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure. A framework for handling inconsistency in changing ontologies. In *Proceedings of the Fourth International Semantic Web Conference (ISWC 2005)*, volume 3729 of *Lecture Notes in Computer Science*, pages 353–367. Springer, 2005. 120, 151

[56] Jeff Heflin and James Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001. 25

[57] Jean Henrard, Didier Roland, Anthony Cleve, and Jean-Luc Hainaut. Large-scale data reengineering: Return from experience. In *Proc. of the 15th Working Conference on Reverse Engineering (WCRE 2008)*, pages 305–308. IEEE Computer Society Press, 2008. 4

[58] M. Horridge, B. Parsia, and U. Sattler. Explaining inconsistencies in owl ontologies. *Scalable Uncertainty Management*, pages 124–137, 2009. 151

[59] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 454–459, 2003. 120, 151

[60] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 466–471, 2005. 5

[61] David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *Journal of Computer and System Sciences*, 28(1):167–189, 1984. 30, 31

[62] Hirofumi Katsuno and Alberto Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 387–394, 1991. 69, 72, 85, 90, 93, 94, 96

[63] E. Kharlamov and D. Zheleznyakov. Capturing instance level ontology evolution for *DL-Lite*. *Proceedings of the Tenth International Semantic Web Conference (ISWC 2011)*, pages 321–337, 2011. 97

[64] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 247–257, 2010. 7, 159

[65] Dexter Kozen. *Theory of Computation*. Springer, New York, 2006. 11

[66] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D.F. Savo. Inconsistency-tolerant first-order rewritability of dl-lite with identification and denial assertions. In *Proceedings of the Twentyfifth International Workshop on Description Logic (DL 2012)*, volume 846 of *CEUR Electronic Workshop Proceedings,* http://ceur-ws.org/, 2012. 9

[67] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Source inconsistency and incompleteness in data integration. In *Proceedings of the Ninth International Workshop on Knowledge Representation meets Databases (KRDB 2002)*, volume 54 of *CEUR Electronic Workshop Proceedings,* http://ceur-ws.org/, 2002. 153

[68] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant semantics for description logics. In *Proceedings of the Fourth International Conference on Web Reasoning and Rule Systems (RR 2010)*, 2010. 9

[69] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Query rewriting for inconsistent *DL-Lite* ontologies. In *Proceedings of the Fifth International Conference on Web Reasoning and Rule Systems (RR 2011)*, 2011. 9

[70] Domenico Lembo and Marco Ruzzi. Consistent query answering over description logic ontologies. In *Proceedings of the First International Conference on Web Reasoning and Rule Systems (RR 2007)*, 2007. 120, 129, 154, 160, 161, 162

[71] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twentyfirst ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002. 4, 25, 152, 153

[72] Maurizio Lenzerini. Ontology-based data management: present and future. In *Proceedings of the Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2012)*, 2012. Invited talk. 25

[73] Maurizio Lenzerini and Domenico Fabio Savo. On the evolution of the instance level of *DL-Lite* knowledge bases. In *Proceedings of the Twenty-fourth International Workshop on Description Logic (DL 2011)*, volume 745 of *CEUR Electronic Workshop Proceedings,* http://ceur-ws.org/, 2011. 9

[74] Maurizio Lenzerini and Domenico Fabio Savo. Updating inconsistent Description Logic knowledge bases. In *Proceedings of the Twentieth European Conference on Artificial Intelligence (ECAI 2012)*, 2012. 9

[75] Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984. 5

[76] Paolo Liberatore and Marco Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Trans. Knowl. Data Eng.*, 10(1):76–90, 1998. 83

[77] Jinxin Lin and Alberto O. Mendelzon. Merging databases under constraints. *International Journal of Cooperative Information Systems*, 7(1):55–76, 1998. 153

[78] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic ABoxes. In *Proceedings of the Tenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 46–56, 2006. 6, 69, 70, 71, 97

[79] Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari. Inconsistency-tolerant query rewriting for linear datalog+/-. In *Datalog 2.0*, volume 7494 of *Lecture Notes in Computer Science*, pages 123–134. Springer, 2012. 156

[80] Carsten Lutz. Description logics with concrete domains: A survey. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logics*, volume 4. King's College Publications, 2003. 27

[81] Carsten Lutz, David Toman, and Frank Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proceedings of the Twentyfirst International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 2070–2075, 2009. 5

[82] Y. Ma, P. Hitzler, and Z. Lin. Algorithms for paraconsistent reasoning with owl. *The Semantic Web: Research and Applications*, pages 399–413, 2007. 119, 151

[83] Yue Ma and Pascal Hitzler. Paraconsistent reasoning for OWL 2. In *Proceedings of the Third International Conference on Web Reasoning and Rule Systems (RR 2009)*, pages 197–211, 2009. 119, 120, 151

[84] Giulia Masotti, Riccardo Rosati, and Marco Ruzzi. Practical abox cleaning in dl-lite (progress report). In *Proceedings of the Twentyfourth International Workshop on Description Logic (DL 2011)*, 2011. 119, 130

[85] Thomas Meyer, Kevin Lee, and Richard Booth. Knowledge integration for description logics. In *In Proceedings of the Seventh International Symposium on Logical Formalizations of Commonsense Reasoning*, volume 20, pages 645–650. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005. 152

[86] Bernhard Nebel. A knowledge level analysis of belief revision. In *Proceedings of the First International Conference on the Principles of Knowledge Representation and Reasoning (KR'89)*, pages 301–311, 1989. 99

[87] Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning*, 41(1):61–98, 2008. 5

[88] Maria Magdalena Ortiz de la Fuente, Diego Calvanese, Thomas Eiter, and Enrico Franconi. Data complexity of answering conjunctive queries over $\mathcal{SHIQ}$ knowledge bases. Technical report, Faculty of Computer Science, Free University of Bozen-Bolzano, July 2005. Also available as CORR technical report at http://arxiv.org/abs/cs.LO/0507059/. 5

[89] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley Publ. Co., 1994. 11

[90] O. Papini. Knowledge-base revision. *The Knowledge Engineering Review*, 15(04):339–370, 2000. 96

[91] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL ontologies. In *Proceedings of the Fourteenth International World Wide Web Conference (WWW 2005)*, pages 633–640, 2005. 120, 151

[92] Peter F. Patel-Schneider. Adding number restrictions to a four-valued terminological logic. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI'88)*, pages 485–490, 1988. 119, 151

[93] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. A comparison of query rewriting techniques for *DL-lite*. In *Proceedings of the Twenty-second International Workshop on Description Logic (DL 2009)*, volume

477 of *CEUR Electronic Workshop Proceedings*, http://ceur-ws.org/, 2009. 5

[94] Antonella Poggi. *Structured and Semi-Structured Data Integration*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 2006. 35, 55

[95] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *Journal on Data Semantics*, X:133–173, 2008. 5, 7, 21, 25, 26, 29, 30, 55, 120, 159, 183, 207

[96] G. Qi, W. Liu, and D. Bell. A revision-based approach to handling inconsistency in description logics. *Artificial Intelligence Review*, 26(1):115–128, 2006. 119, 152

[97] G. Qi and F. Yang. A survey of revision approaches in description logics. In *Proceedings of the Second International Conference on Web Reasoning and Rule Systems (RR 2008)*, pages 74–88, 2008. 72

[98] Guilin Qi and Jianfeng Du. Model-based revision operators for terminologies in description logics. In *Proceedings of the Twentyfirst International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 891–897, 2009. 120

[99] Guilin Qi, Weiru Liu, and David A. Bell. Knowledge base revision in description logics. In *Proceedings of the Tenth European Workshop on Logics in Artificial Intelligence (JELIA 2006)*, volume 4160 of *Lecture Notes in Computer Science*, pages 386–398. Springer, 2006. 72, 90

[100] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4), 2008. 12

[101] Mariano Rodríguez-Muro, Lina Lubyte, and Diego Calvanese. Realizing ontology based data access: A plug-in for Protégé. In *Proc. of the ICDE Workshop on Information Integration Methods, Architectures, and Systems (IIMAS 2008)*, pages 286–289. IEEE Computer Society Press, 2008. 5

[102] Riccardo Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *Proceedings of the Twentysecond International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 1057–1062, 2011. 156

[103] Riccardo Rosati and Alessandro Almatelli. Improving query answering over *DL-Lite* ontologies. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 290–300, 2010. 173

[104] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of descritpion logic terminologies. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 355–360, 2003. 151

[105] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic. User-driven ontology evolution management. In *In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*, pages 133–140, 2002. 83

[106] U. Straccia. A sequent calculus for reasoning in four-valued description logics. *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 343–357, 1997. 119, 151

[107] Moshe Y. Vardi. The complexity of relational query languages. In *Proceedings of the Fourteenth ACM SIGACT Symposium on Theory of Computing (STOC'82)*, pages 137–146, 1982. 21, 57

[108] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer, 1999. 12

[109] Zhe Wang, Kewen Wang, and Rodney W. Topor. A new approach to knowledge base revision in *DL-Lite*. In *Proceedings of the Twentyfourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010. 6, 69, 96, 120

[110] Marianne Winslett. *Updating Logical Databases*. Cambridge University Press, 1990. 71, 83, 97, 131, 208