



FREIE UNIVERSITÄT BOZEN

LIBERA UNIVERSITÀ DI BOLZANO

FREE UNIVERSITY OF BOZEN · BOLZANO

Updating Description Logic Knowledge Bases

A DISSERTATION SUBMITTED TO THE FACULTY OF COMPUTER SCIENCE,
FREE UNIVERSITY OF BOZEN-BOLZANO IN FULFILLMENT TO THE DEGREE OF

MASTER SCIENCE IN COMPUTER SCIENCE

as part of the

EUROPEAN MASTER PROGRAM IN
COMPUTATIONAL LOGIC

Submitted by: Dmitriy ZHELEZNYAKOV

Faculty Advisor: prof. Diego CALVANESE

October 2009

FACULTY OF COMPUTER SCIENCE

ABSTRACT

In the last decade there has been a constantly increasing interest in ontologies. In particular, ontologies are regarded as one of the prime concepts in the Semantic Web, where they can be used to describe the semantics of information at various sites, overcoming the problem of implicit and hidden knowledge, and thus enabling content exchange. Currently the standard ontology language for the Web is OWL2, that has Description Logics (DLs) as its core.

Description Logics provide a solid foundation for ontology representation and reasoning. Although results on DLs provide effective reasoning techniques for ontology management tools, they hardly can be used for supporting other tasks. One of the remarkable examples is ontology update, which is especially important in the Semantic Web context due to dynamicity of the Web.

In this thesis we address the problem of update for a rich family of Description logics called DL-Lite, that is a most tractable fragment of OWL2. DL-Lite ontologies are composed of two components, a TBox that expresses general knowledge about the concepts and their relationships, and an ABox that describes the properties of individuals that are instances of concepts. In this paper, we focus on updates for both the ABox and the TBox level of KBs.

Updating ABoxes corresponds to the situation when the state of affairs in an application domain changes. Description Logics are generally not closed with respect to instance-level update, in the sense that the set of models corresponding to the application of any of these operations to a KB in a Description Logic \mathcal{L} may not be expressible by an ABox in \mathcal{L} . In particular, we show that this is true for DL-Lite_{core}. To deal with this problem, we first define a minimal extension of standard DL-Lite languages that is closed under update, and introduce a polynomial algorithm for computing instance-level update in this logic. Then we provide a notion of approximation with respect to a fixed language \mathcal{L} of instance-level update, and propose algorithms that obtain an approximate update for DL-Lite _{\mathcal{R}} and DL-Lite _{\mathcal{A}} .

Regarding TBox updates, the situation is very different from the case of ABox update, since TBox changes may also affect the ABox level. We propose a polynomial algorithm which produces update for DL-Lite _{\mathcal{FR}} , taking into account all implicit relations implied by the TBox and their influence on ABox assertions. Since Description Logics are typically not closed with respect to intensional-level update either, here we also use the notion of approximate update. We characterize the semantics of update on the basis of the approaches proposed by Winslett and by Poggi.

The thesis extends previous results on the semantics and complexity of DL-Lite updates of both ABox and TBox for the logics DL-Lite \mathcal{FR} , DL-Lite $_{core}$, DL-Lite \mathcal{A} , and DL-Lite \mathcal{R} . The thesis also provides algorithms to compute updates.

CONTENTS

1. <i>Introduction</i>	1
2. <i>Preliminaries</i>	7
2.1 The language	7
2.2 DL-Lite _{(\mathcal{FR})\mathcal{S}}	10
2.3 Semantics of DL constructs	10
2.4 The notion of chase of a knowledge base	12
2.5 Closure of negative inclusions	12
3. <i>Update and Erasure at the Instance-Level</i>	15
3.1 Update and Erasure in DL-Lite _{\mathcal{RS}}	17
3.1.1 Update in DL-Lite _{\mathcal{RS}}	17
3.1.2 Erasure in DL-Lite _{\mathcal{RS}}	25
3.2 Update and Erasure in DL-Lite _{\mathcal{AS}}	27
3.2.1 Update in DL-Lite _{\mathcal{AS}}	27
3.2.2 Erasure in DL-Lite _{\mathcal{AS}}	36
3.3 Instance-level update and erasure in DL-Lite _{\mathcal{A}}	36
3.3.1 Update in DL-Lite _{\mathcal{A}}	36
3.3.2 Erasure in DL-Lite _{\mathcal{A}}	39
4. <i>Updating DL TBoxes</i>	41
4.1 Updating TBox: Problems and Their Solutions	45
4.1.1 Updating TBoxes with PIs	45
4.1.2 Updating TBoxes with NIs	46
4.1.3 Updating TBoxes with functional assertions	48
4.2 TBox Update in DL-Lite _{\mathcal{FR}}	48
4.2.1 TBox Update: from DL-Lite _{\mathcal{FR}} to DL-Lite _{\mathcal{FRS}}	48
4.2.2 TBox Approximate Update: from DL-Lite _{\mathcal{FRS}} to DL-Lite _{\mathcal{FR}}	61
5. <i>Conclusion</i>	63

1. INTRODUCTION

In the last decade there has been a constantly increasing interest in ontologies. In particular, ontologies are regarded as one of the prime concepts in the Semantic Web [BLHL01], where they can be used to describe the semantics of information at various sites, overcoming the problem of implicit and hidden knowledge, and thus enabling content exchange. Generally speaking, the term “ontology” has come to computer science from philosophy. In both fields, the essence of ontology is the representation of entities, ideas, and events, along with their properties and relations, according to a system of categories.

One can single out three levels of an ontology. The first one, the meta-level, specifies a set of modeling categories, such as entities, individuals, etc. The second level that is known as the intensional level specifies a set of conceptual elements, in rough words, instances of categories, and a set of rules to describe the conceptual structure of the domain. While the first level defines the language which will be used for describing, the syntax and the semantics of an ontology, the intensional level describes a specific area of interest, delineating classes, entities, their attributes and relations among them. Finally, the extensional, or instance, level specifies a set of instances of the conceptual elements described at the intensional level (Fig. 1.1).

Description Logics (DLs) provide a solid foundation for ontology representation and reasoning [BCM⁺03]. Though results on DLs provide effective reasoning techniques for ontology management tools [Hor98, Ho01], they hardly can be used for supporting other tasks. One of the remarkable examples is ontology update.

Suppose one wants to construct, use, and maintain a DL knowledge base (KB) describing what is known about one’s area of interest. For example, the concept *AvailablePlayer* describes the class of all players who can play the next match, and the concept *UnavailablePlayer* represents those players who are traumatized or suspended. This concept is formulated in DL-Lite_{core}, the basic DL. Concepts are the most important elements of description logics, and they are used to make a primary description of the domain of interest. The ABox assertion *AvailablePlayer(inzaghi)* says that Inzaghi is able to play the next match. In DL applications, an ABox is usually used to represent the current state of affairs in the application domain [BCM⁺03]. In such applications there exists a need to update the ABox if that state has changed. In other words, one would naturally wish to have a means of updating the information kept in the ontology, reflecting changes in the domain of interest. But how can new facts be added to a KB when the new information may contradict preexisting facts? E.g., if we know that Inzaghi has got serious injury

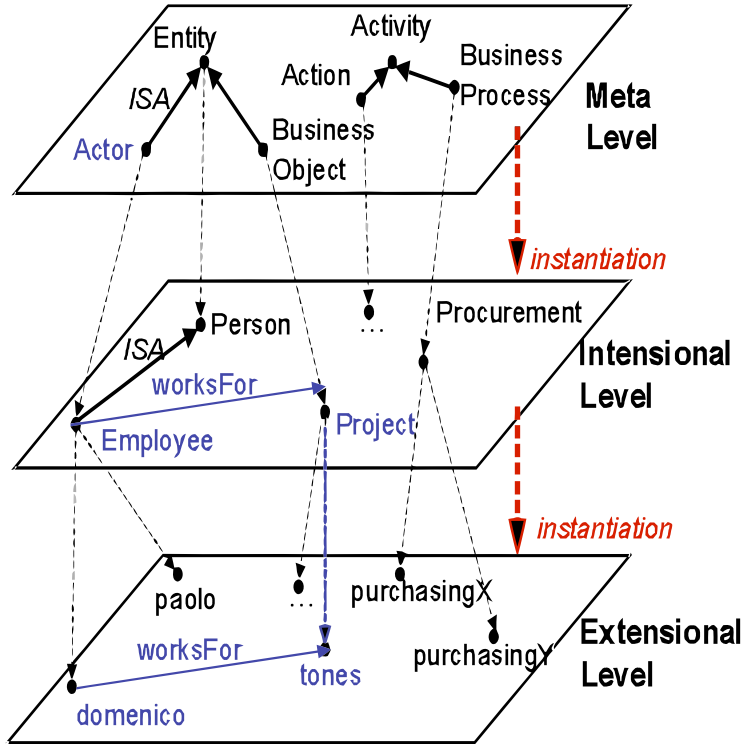


Fig. 1.1: The Three Levels of an Ontology

at the last training and cannot play the next match, we should update the above ABox in a corresponding way. The simplest choice is to make the users determine exactly what to add and remove from the knowledge base. But what is really needed is a way to specify the desired change intensionally, by stating a formula that the state of the world is now known to satisfy and by letting the knowledge base update algorithms automatically accomplish that change.

First research on the subject was started in the late 1970s, beginning from the examination of the problems posed by queries against databases containing incomplete information [Gra89]. The type of incompleteness that was considered was usually that of null values, which is a value that is known to be in the certain domain, but where concrete value is unknown. For example, $PlayInClub(ronaldo, ?)$ says that Ronaldo plays in some club, but what this club is exactly we do not know. The challenge here is, firstly, to define what the correct answer should be, and justify that; secondly, to develop a way to obtain the desired query answer efficiently. E.g., if we have a query asking whether Ronaldo plays in Milan, what should the answer be? An additional complication arises from the fact that a

null value is not value itself, but represents an existentially quantified variable (for example, $PlayInClub(ronaldo, ?)$ and $PlayInClub(ronaldinho, ?)$ do not imply that Ronaldo and Ronaldinho play in the same club).

Another type of incompleteness arising in database applications comes not from insufficient information but schema attributes that are not applicable for a particular tuple. For example, the “child’s name” attribute for a player who has no children might get a special type of null value: “inapplicable”. The latter value indicates a lack of correspondence between the database schema and the state of the real world represented by the schema. In theory, one could mend the schema in order to avoid the appearance of inapplicability. But in practice the resulting schema may spread out and be inconvenient to use, so that from an engineering point of view, it may be better to keep an imperfect schema and learn how to handle “inapplicable” null values.

Along with the problem of query answering in incomplete databases, the problem of updating such databases also drew attention. Problems arise when updating depends on the missing value. Though it is syntactically simple to allow null values in relational tables and their updates, any sensible semantics for these updates yields result relations that cannot be stored as relational tables. For these reasons, we should relief the restrictions of traditional relational databases and instead consider databases as simple, restricted theories in first-order logic with equality [Win90].

So, given a database encoded as a set of first-order sentences T , one would like to have ability to update T when new information arrives. Unfortunately, there is no obvious and application-independent choice of semantics for updating T , and a lot of candidate semantics have appeared in the literature [AG85, MLG87, Win88a]. Among them, we can mark out two basic classes of semantics: those based on the formulas presented in T (*formula-based*), and those based only on the models of T (*model-based*).

Under the model-based paradigm, the objects of change are not the formulas of a theory T , but its individual models. Note that the update of a model is not in general a model, but a set of them, since it is not uncommon that there exist more than one way to update. In all these models, the inserted formulas (new information) must be obviously true. At the same time, the “difference” between the original model and the obtained ones must be as small as possible in some meaning; otherwise, it is intuitively clear that an arbitrary set of models satisfying the insertion cannot generally reflect new possible states of the world. Therefore, the way of doing the update must ensure that the resulting models are satisfiable if the insertion is. From the model-based point of view, to update a theory means to update all its models and obtain a new theory that describes exactly all new models. The principles mentioned above are the basic ones, however, different semantics may have additional restrictions depending on applications.

Under the formula-based semantics, a theory T is highlighted. Roughly speaking, an update is performed by adding the inserted formulas to T . If the union is contradictory, one should remove from T some formulas, different from the inserted ones, in order to make the result consistent. One way to make an update

is to remove all the formulas of T . But one would like to keep as much of the old data as possible, so the deleted part should be as small as possible. It turns out that there may be several subsets of T that are candidates for the deletion. There were several approaches considered to find the best candidate for the deletion:

- to have a set of theories as an output, one theory for each minimal set;
- to make a cross-product of those theories;
- to remove all formulas of all minimal sets.

The main disadvantage of the first two approaches is that it is difficult to see how to construct an efficient implementation of the semantics. In addition, since the formula-based semantics are sensitive to the syntax, the behavior of the resulting theory is almost unpredictable. The latter approach, called WIDTIO (the acronym stands for “When In Doubt Throw It Out”), has the advantage that it is easier to implement than the two previous ones, despite it tends to lose information. Other restrictions on the formula based semantics are quite natural and similar to the case of model-based semantics:

- the inserted formulas must be implied by the resulting theory;
- satisfiable insertion must yield satisfiable result.

Winslett in [Win98b] studied the relationships between the two classes of semantics, model-based and formulas-based ones. In a number of cases the former class behaves computationally better. She considers a so-called Possible World Approach (PWA), which is an example of a formula-based WIDTIO semantics, and a so-called Possible Model Approach (PMA), which is an example of a model-based semantics. She compared the behavior of PWA and PMA on a number of examples, and presented several problems that were unavoidable under the PWA but not under the PMA. At the same time all the PMA limitations are also present under the PWA. E.g., the frame problem may appear under the PWA since the principle of minimality there is measured only by the effect of a change of the formulas in T rather than by considering the effect of a change of the world itself. It leads to the second-class status of those formulas that can be derived from T and makes the PWA too reluctant to retract the formulas presented in T . The situation with the ramification problem is even worse. If the PWA may just fail drawing certain wanted conclusions, it may lie dealing with incomplete information, as opposed to the PMA.

In spite of its advantages, the PMA as well as another minimal change methods is generally problematic for updating knowledge bases with disjunctive information. In [ZF96] two different approaches have been proposed to deal with this problem – one is called the *Minimal Change with Exceptions* (MCE), the other is called the *Minimal Change with maximal Disjunctive inclusions* (MCD). The first method is syntax based, while the second one is model theoretic. Under MCE, if the truth value of a literal is logically indefinite with respect to the update, then this literal

is treated as an exception to the minimal change principle. In this case the change of this literal’s truth value will not obey the rule of minimal change. The MCD as opposed to the MCE proposes to describe every possible interpretation of the disjunction without losing the minimal change criterion for other information.

Since more and more data is stored in DL ontologies, it is desirable to have a means to update exactly DL KBs with new information. In [Pog06], instance level update of DL-Lite \mathcal{F} knowledge bases has been introduced. The underlying method is the PMA adapted to the DL-Lite case, in which an efficient algorithm of obtaining update can be devised. Also the notion of *erasure* has been examined there. Despite it is quite close to the notion of update, it has a little but important difference. Erasure is not just update with negation of information (that is, having $P(a)$, we do update with $\neg P(a)$), which would forbid the state of the world in which the erased information is true (that is $P(a)$ is not satisfied by any model of an updated KB), but it is proposed to *erase* information by making it uncertain (that is $P(a)$ may either appear or not appear in a model).

In our work we study the notion of update of an ontology expressed as a DL knowledge base. We recall that DL-Lite knowledge bases consist of a TBox that expresses the intensional level of the ontology, i.e. general knowledge about concepts and their relations; and an ABox that represents the instance, level of the ontology, i.e. the state concerning the instances of concepts and relations.

In the Chapter 2, we recall the main notions about DLs, their basic constructs, their syntax and semantics, and some definitions that will be useful later.

In the Chapter, we consider the notion of instance level update for DLs. We recall some of the results in [Pog06]. Firstly, we provide the general framework for instance level update of DL ontologies, by specifying in particular, the formal semantics for update. Then, we show that the DL-Lite \mathcal{RS} language introduced in the Chapter 1 is closed with respect to instance-level update, in the sense that the result of an update is always expressible in the language of the original ontology. It contrasts with the results in [LYV⁺98], which imply that if we use more expressive logics, instance-level update generally is not expressible in the logic of the original database. Also, we give an algorithm for computing updates for DL-Lite \mathcal{RS} , i.e., an ABox that corresponds to the update, in polynomial time. Then, combining the result with the similar one from [Pog06] concerning DL-Lite \mathcal{FS} we obtain an algorithm working for DL-Lite \mathcal{AS} . Afterwards, we examine the notion of approximate instance level update for DL-Lite \mathcal{A} . Since an update of a DL-Lite \mathcal{A} KB is not always expressible in DL-Lite \mathcal{A} itself, we have to build an ABox, approximately describing it. Since the notion of instance level erasure for DLs is quite close to the notion of update, we obtain analogous results for it the update results on update.

In the Chapter 4, we study the notion of TBox update of DL-Lite \mathcal{FR} KBs. Despite that the TBox level of an ontology is more “stable” and unchangeable than the ABox level, it may also need to be modified. Though an insertion includes only TBox assertions, that is role inclusion assertions, concept inclusion assertions, and role functional assertions, changes at intensional level inevitably affect the instance level. Intensional level of an ontology described in a DL language may include an even greater number of implicit relations than those explic-

itly mentioned in a TBox. For example, if we know that $Humans \sqsubseteq Chordates$, $Chordates \sqsubseteq Animals$, and $Animals \sqsubseteq \neg Plants$, it immediately follows that $Humans \sqsubseteq Animals$, $Humans \sqsubseteq \neg Plants$, and $Chordates \sqsubseteq \neg Plants$. Thus, when making changes to a TBox, it is necessary to take into account such implicit relations. Moreover, we should also change the ABox in order to satisfy a new TBox, and not to lose information that could be kept. The latter requirement gives rise to some problems, i.e., inserting a negative inclusion assertion, we could face a problem that two membership assertions in an ABox must be replaced by a new one that provides a choice between them. But since one cannot express in DL-Lite ABox any disjunction, this problem does not have a trivial solution. A similar problem arises when we insert a new functional assertion into a TBox. We solve this problems by means of the prioritization over predicates.

The algorithm introduced in the Chapter 4 provides update of a TBox of a DL-Lite $_{\mathcal{FR}}$ knowledge base. Since that algorithm in general returns a DL-Lite $_{\mathcal{FRS}}$ ABox, we else show how to compute an approximation of a set of models of the updated KB.

All the algorithms introduced in the work have good computational properties, computing result in polynomial time in the size of the original TBox \mathcal{T} , of the original ABox \mathcal{A} , and of the insertion \mathcal{F} .

2. PRELIMINARIES

In this chapter we give definitions of the basic DL-Lite languages: DL-Lite \mathcal{F} and DL-Lite \mathcal{R} , their “hybrid” DL-Lite \mathcal{A} , and their extension, the DL-Lite \mathcal{S} subfamily. In addition, some useful notions regarding DL-Lite are defined.

2.1 The language

Description Logics (DLs) are knowledge representation formalisms, tailored for representing the domain of interest in terms of *concepts* and *roles*. In DLs [BCM⁺03] complex concept and role expressions (or simply, concepts and roles) are obtained starting from atomic concepts and roles (which are simply names) by applying suitable constructs. Concepts and roles are then used in a DL knowledge base (KB) to model the domain of interest. Specifically, a DL KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is formed by two distinct parts, a *TBox* \mathcal{T} and an *ABox* \mathcal{A} . The TBox \mathcal{T} represents the *intensional-level* of the KB, i.e., the general knowledge. The ABox provides information on the *instance-level* of the KB. In this paper we focus on a family of DLs called *DL-Lite*. We now introduce DL-Lite $_{core}$, DL-Lite \mathcal{F} , and DL-Lite \mathcal{R} .

DL-Lite $_{core}$. This logic includes those constructs that are used in every DL-Lite language.

$$B ::= A \mid \exists R$$

$$C ::= B \mid \neg B$$

$$Q ::= P \mid P^-,$$

where A denotes an *atomic concept*, B a *basic concept*, and C a *general concept*. P denotes an *atomic role*, and Q a *basic role*.

A DL-Lite $_{core}$ TBox is a set of *concept inclusion assertions* of the form

$$B \sqsubseteq C,$$

and an ABox is a set of *membership assertions* of the form

$$A(a), P(a, b).$$

DL-Lite _{\mathcal{F}} . This logic is an extension of the previous one that includes all the constructs of *DL-Lite_{core}*, and additionally allows one to have in the TBox so-called *role functional assertions* of the form

$$(\text{funct } Q).$$

An ABox is of the same form as a *DL-Lite_{core}* ABox.

DL-Lite _{\mathcal{R}} . This logic, is an extension of *DL-Lite_{core}*, but it has an additional construct

$$R ::= Q \mid \neg Q,$$

where, R denotes a *general role*, and it allows to have in its TBox *role inclusion assertions* (instead of functional assertions) of the form

$$Q \sqsubseteq R.$$

A *DL-Lite _{\mathcal{R}}* ABox is of the same form as a *DL-Lite_{core}* ABox.

DL-Lite _{\mathcal{A}} . Both *DL-Lite _{\mathcal{F}}* and *DL-Lite _{\mathcal{R}}* have nice computational properties, e.g., knowledge base satisfiability is PTime in the size of TBox and LOGSPACE in data complexity. But can we make a hybrid of them, keeping their computational properties? The answer is ‘yes’, but with some restrictions on the TBox described below.

DL-Lite _{\mathcal{A}} is a hybrid of the two previous logics. It distinguishes between objects and values, by allowing to use value-domains, a.k.a. concrete domains (like booleans, integers, strings, and so on); concept attributes (binary relations between objects and values); and role attributes (binary relations between pair of objects and values) [CDGL⁺06, CDGL⁺09]. So, *DL-Lite _{\mathcal{A}}* has the following constructs:

$$B ::= A \mid \exists Q \mid \delta(U_C)$$

$$C ::= \top_C \mid B \mid \neg B \mid \exists Q.C \mid \delta_F(U_C) \mid \exists \delta_F(U_R) \mid \exists \delta_F(U_R)^-,$$

where A , B and C denote atomic, basic and general concepts, respectively; \top_C denotes the *universal (top) concept*. Then, given an *atomic concept attribute* U_C , $\delta(U_C)$ denotes the domain of U_C , i.e., the set of objects that U_C relates to values. In a similar way, $\delta(U_R)$ denotes the domain of an *atomic role attribute* U_R , i.e., the set of pairs of objects that U_R relates to values. Moreover, $\delta_F(U_C)$ (resp., $\delta_F(U_R)$) is a designation of the set of objects (resp., of pairs of objects) that U_C (resp. U_R) relates to values in the value-domain F .

$$Q ::= P \mid P^- \mid \delta(U_R) \mid \delta(U_R)^-$$

$$R ::= Q \mid \neg Q \mid \delta_F(U_R) \mid \delta_F(U_R)^-$$

where P , Q and R denote atomic, basic and general roles respectively.

$$E ::= D \mid \rho(U_C) \mid \rho(U_R)$$

$$F ::= \top_D \mid E \mid \neg E \mid T_1 \mid \dots \mid T_n,$$

where D , E and F denote *atomic*, *basic* and *general value-domains* respectively; \top_D denotes the *universal value-domain*; each T_i is an RDF data type. And finally,

$$V_C ::= U_C \mid \neg U_C$$

$$V_R ::= U_R \mid \neg U_R,$$

where U_C denotes an *atomic concept attribute*, V_C a *general concept attribute*, U_R an *atomic role attribute*, V_R a *general role attribute*.

A DL-Lite_A TBox is of the following form:

$$\begin{aligned} B \sqsubseteq C \quad Q \sqsubseteq R \quad E \sqsubseteq F \\ U_C \sqsubseteq V_C \quad U_R \sqsubseteq V_R \\ (\text{funct } Q) \quad (\text{funct } U_C) \quad (\text{funct } U_R). \end{aligned}$$

In this work we will use the notation described below. Writing A , we will mean an atomic concept, C is for general concept, P is an atomic concepts and so on. Mark that, for example, C designates both B and $\neg B$ and we will not distinguish those cases when there is no need for this. Analogously, Q is for both P and P^- , R is for both Q and $\neg Q$, V is for both U and $\neg U$.

Then, the following syntactical sugar will be used: notation $\neg C$ designates $\neg B$, if $C = B$, and designates B , if $C = \neg B$; and the similarly for the other constructs (see the following table).

Designation	Meaning	Condition	Designation	Meaning	Condition
$\neg C$	$\neg B$	if $C = B$	$\neg R$	$\neg Q$	if $R = Q$
	B	if $C = \neg B$		Q	if $R = \neg Q$
Q^-	P^-	if $Q = P$	$\neg V$	$\neg U$	if $V = U$
	P	if $Q = P^-$		U	if $V = \neg U$

The TBox restrictions mentioned before are as follows:

- if $\exists Q.C$ appears in \mathcal{T} , then $(\text{funct } Q)$ is not in \mathcal{T} ;
- if $Q_1 \sqsubseteq Q_2$ appears in \mathcal{T} , then $(\text{funct } Q_2)$ is not in \mathcal{T} ;
- if $U_C \sqsubseteq U'_C$ appears in \mathcal{T} , then $(\text{funct } U'_C)$ is not in \mathcal{T} ;
- if $U_R \sqsubseteq U'_R$ appears in \mathcal{T} , then $(\text{funct } U'_R)$ is not in \mathcal{T} .

Since DL-Lite_A has more constructs, it allows more assertion types in its ABox:

Concept membership assertion	$A(c)$
Role membership assertion	$P(c_1, c_2)$
Domain membership assertion	$D(v)$
Concept attribute membership assertion	$U_C(c, v)$
Role attribute membership assertion	$U_R(c_1, c_2, v)$

where c , c_1 and c_2 denotes object constants, and v is a value constant.

2.2 DL-Lite_{(FRA)S}

We introduce new DL-Lite logics: DL-Lite_{F \mathcal{S}} , DL-Lite_{R \mathcal{S}} , and DL-Lite_{A \mathcal{S}} , which are extensions of DL-Lite_F, DL-Lite_R, and DL-Lite_A, respectively. The difference with the previous ones is in their ABoxes. Since that DL-Lite_S logics allow to express the existence of objects (or values) that are instances of concepts (or value-domains), without naming the actual objects (or values), by means of so-called soft-constants.

Definition 2.1. A soft-constant (or variable) is a constant that is not interpreted under the Unique Name Assumption.

Recall that in DL-Lite_F, DL-Lite_R and DL-Lite_A all constants appearing in an ABox are under Unique Name Assumption, i.e. if $a_1^{\mathcal{I}} = a_2^{\mathcal{I}}$, then $a_1 = a_2$.

Moreover, both general concepts and general roles are allowed in their ABoxes. Thus, a DL-Lite_S ABox is of the form:

$$C(a) \quad C(x) \quad R(a, b) \quad F(v) \quad F(y) \quad V_C(a, v) \quad V_R(a, b, v)$$

where the last four assertion types are present only in DL-Lite_{A \mathcal{S}} .

2.3 Semantics of DL constructs

The semantics of DL-Lite is based on the notion of interpretation. An interpretation $\mathcal{I} = \langle \cdot^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of an interpretation domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$. For DL-Lite_A (and DL-Lite_{A \mathcal{S}}), where we distinguish objects and values, the interpretation domain is split into two non-overlapping subdomains, notably, the object domain $\Delta_O^{\mathcal{I}}$ and the value domain $\Delta_V^{\mathcal{I}}$. The function *val* mapping from the set of soft-constants to $\Delta_V^{\mathcal{I}}$ is predefined. The interpretation of DL-Lite constructs is as follows:

Construct	Syntax	Semantics
Atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}}$
Existential restriction	$\exists Q$	$\{a \mid \exists a'. (a, a') \in Q^{\mathcal{I}}\}$
Concept attribute domain	$\delta(U_C)$	$\{a \mid \exists v. (a, v) \in U_C^{\mathcal{I}}\}$
Top concept	\top_C	$\top_C^{\mathcal{I}} = \Delta_O^{\mathcal{I}}$
Concept negation	$\neg B$	$\Delta_O^{\mathcal{I}} \setminus B^{\mathcal{I}}$
Qualified existential restriction	$\exists Q.C$	$\{a \mid \exists a'. (a, a') \in Q^{\mathcal{I}} \wedge a' \in C^{\mathcal{I}}\}$
Concept attribute domain restriction	$\delta_F(U_C)$	$\{a \mid \exists v. (a, v) \in U_C^{\mathcal{I}} \wedge v \in F^{\mathcal{I}}\}$
Atomic role	P	$P^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}$
Inverse role	P^-	$\{(a, a') \mid (a', a) \in P^{\mathcal{I}}\}$

Construct	Syntax	Semantics
Role attribute domain	$\delta(U_R)$	$\{(a, b) \mid \exists v.(a, b, v) \in U_R^{\mathcal{I}}\}$
Role negation	$\neg Q$	$(\Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}) \setminus Q^{\mathcal{I}}$
Role attribute domain restriction	$\delta_F(U_R)$	$\{(a, b) \mid \exists v.(a, b, v) \in U_R^{\mathcal{I}} \wedge v \in F^{\mathcal{I}}\}$
Concept attribute range	$\rho(U_C)$	$\{v \mid \exists a.(a, v) \in U_C^{\mathcal{I}}\}$
Role attribute range	$\rho(U_R)$	$\{v \mid \exists(a, b).(a, b, v) \in U_R^{\mathcal{I}}\}$
Top domain	\top_D	$\top_D^{\mathcal{I}} = \Delta_V^{\mathcal{I}}$
Domain negation	$\neg E$	$\Delta_V^{\mathcal{I}} \setminus E^{\mathcal{I}}$
Data type	T_i	$val(T_i) \subseteq \Delta_V^{\mathcal{I}}$
Atomic attribute	U	$U^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}$
Attribute negation	$\neg U$	$(\Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}) \setminus U^{\mathcal{I}}$
Object constant	c	$c^{\mathcal{I}} \in \Delta_O^{\mathcal{I}}$
Value constant	v	$val(v) \in \Delta_V^{\mathcal{I}}$

The semantics of TBox assertions is specified by saying when an assertion is *satisfied* in \mathcal{I} (or, is a model of \mathcal{I}).

Assertion	Syntax	Semantics
Concept inclusion assertion	$B \sqsubseteq C$	$B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
Role inclusion assertion	$Q \sqsubseteq R$	$Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
Value-domain inclusion assertion	$E \sqsubseteq F$	$E^{\mathcal{I}} \subseteq F^{\mathcal{I}}$
Attribute inclusion assertion	$U \sqsubseteq V$	$U^{\mathcal{I}} \subseteq V^{\mathcal{I}}$
Role functionality assertion	$(\text{funct } Q)$	$\forall a, a', a''. (a, a') \in Q^{\mathcal{I}} \wedge (a, a'') \in Q^{\mathcal{I}} \rightarrow a' = a''$
Attribute functionality assertion	$(\text{funct } U)$	$\forall a, v', v''. (a, v') \in U^{\mathcal{I}} \wedge (a, v'') \in U^{\mathcal{I}} \rightarrow v' = v''$

The semantics of ABox assertions is defined similarly:

Assertion	Syntax	Semantics
Concept membership assertion	$A(c)$	$c^{\mathcal{I}} \in A^{\mathcal{I}}$
Role membership assertion	$P(c_1, c_2)$	$(c_1^{\mathcal{I}}, c_2^{\mathcal{I}}) \in P^{\mathcal{I}}$
Concept attribute membership assertion	$U_C(c, v)$	$(c^{\mathcal{I}}, val(v)) \in U_C^{\mathcal{I}}$
Role attribute membership assertion	$U_R(c, d, v)$	$(c^{\mathcal{I}}, d^{\mathcal{I}}, val(v)) \in U_R^{\mathcal{I}}$

Model of DL KB. We say that an interpretation $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ is a model of a TBox \mathcal{T} if and only if it is a model of each assertion in \mathcal{T} . Similarly, \mathcal{I} is a model of an ABox \mathcal{A} if and only if it is a model of each membership assertion in \mathcal{A} . Finally, if $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a DL KB, then \mathcal{I} is a model of \mathcal{K} if and only if it is a model of both \mathcal{T} and \mathcal{A} .

$Mod(T)$ stands for the set of models of T , where T is a set of formulas, i.e., applied to the work, it is a set of TBox assertions or membership assertions. If $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a DL KB, then its set of models $Mod(\mathcal{T} \cup \mathcal{A})$ is designated as $Mod(\mathcal{K})$.

In the following, with some abuse to notation, we may use the same symbol, say, c to denote both a constant appearing in a KB and an object in the domain of interpretation such that the interpretation of c is equal to the object.

2.4 The notion of chase of a knowledge base

Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, one can build a structure which is called $chase(\mathcal{K})$ [CDGL⁺07].

Definition 2.2. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite _{\mathcal{F}} or DL-Lite _{\mathcal{R}} KB. Then, $chase(\mathcal{K})$ is a set of DL-Lite _{\mathcal{F}} or DL-Lite _{\mathcal{R}} membership assertions obtained by applying recursively the following rules:

- cr0 $\mathcal{A} \subseteq chase(\mathcal{K})$;
- cr1 if $A_1 \sqsubseteq A_2 \in \mathcal{T}_p$ and $A_1(a) \in chase(\mathcal{K})$, then $A_2(a) \in chase(\mathcal{K})$;
- cr2 if $A \sqsubseteq \exists R \in \mathcal{T}_p$ and $A(a) \in chase(\mathcal{K})$ and cr2 has not already been applied for $A \sqsubseteq \exists R$ and $A(a)$, then $R(a, b') \in chase(\mathcal{K})$, where b' is a new constant;
- cr3 if $\exists R \sqsubseteq A \in \mathcal{T}_p$ and $R(a, b) \in chase(\mathcal{K})$, then $A(a) \in chase(\mathcal{K})$;
- cr4 if $\exists R_1 \sqsubseteq \exists R_2 \in \mathcal{T}_p$ and $R_1(a, b) \in chase(\mathcal{K})$, then $R_2(a, b') \in chase(\mathcal{K})$, where b' is a new constant;
- cr5 if $R_1 \sqsubseteq R_2 \in \mathcal{T}_p$ and $R_1(a, b) \in chase(\mathcal{K})$ and cr4 has not already been applied for $R_1 \sqsubseteq R_2$ and $R_1(a, b)$, then $R_2(a, b) \in chase(\mathcal{K})$;

where \mathcal{T}_p is the set of positive inclusions in \mathcal{T} .

2.5 Closure of negative inclusions

Here we define the notion of closure of negative inclusions. Originally, it was introduced in [CDGL⁺07] and was adapted in [Pog06] to DL-Lite _{\mathcal{A}} for checking satisfiability of an ontology, but it is also useful for our goal. Firstly, let us give the following definition:

Definition 2.3. An inclusion assertion is a negative inclusion if it is of the form $L \sqsubseteq \neg R'$.

Now, we can turn to the closure of negative inclusions:

Definition 2.4. Let \mathcal{T} be a DL-Lite_{AS} TBox. The *NI-closure* of \mathcal{T} , denoted by $cln(\mathcal{T})$, is the TBox obtained inductively as follows:

1. all negative inclusions in \mathcal{T} are also in $cln(\mathcal{T})$;
2. 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 $cln(\mathcal{T})$;
3. if $E_1 \sqsubseteq E_2$ is in \mathcal{T} and $E_2 \sqsubseteq \neg E_3$ or $E_3 \sqsubseteq \neg E_2$ is in $cln(\mathcal{T})$, then also $E_1 \sqsubseteq \neg E_3$ is in $cln(\mathcal{T})$;
4. 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 $cln(\mathcal{T})$;
5. 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 $cln(\mathcal{T})$;
6. if one of the assertions $\exists Q \sqsubseteq \neg \exists Q$, $\exists Q^- \sqsubseteq \neg \exists Q^-$, or $Q \sqsubseteq \neg Q$ is in $cln(\mathcal{T})$, then all three such assertions are in $cln(\mathcal{T})$;
7. if $U_{C_1} \sqsubseteq U_{C_2}$ is in \mathcal{T} and $\delta(U_{C_2}) \sqsubseteq \neg B$ or $B \sqsubseteq \neg \delta(U_{C_2})$ is in $cln(\mathcal{T})$, then also $\delta(U_{C_1}) \sqsubseteq \neg B$ is in $cln(\mathcal{T})$;
8. if $U_{C_1} \sqsubseteq U_{C_2}$ is in \mathcal{T} and $\rho(U_{C_2}) \sqsubseteq \neg E$ or $E \sqsubseteq \neg \rho(U_{C_2})$ is in $cln(\mathcal{T})$, then also $\rho(U_{C_1}) \sqsubseteq \neg E$ is in $cln(\mathcal{T})$;
9. if $U_{C_1} \sqsubseteq U_{C_2}$ is in \mathcal{T} and $U_{C_2} \sqsubseteq \neg U_{C_3}$ or $U_{C_3} \sqsubseteq \neg U_{C_2}$ is in $cln(\mathcal{T})$, then also $U_{C_1} \sqsubseteq \neg U_{C_3}$ is in $cln(\mathcal{T})$;
10. if one of the assertions $\rho(U_C) \sqsubseteq \neg \rho(U_C)$, $\delta(U_C) \sqsubseteq \neg \delta(U_C)$, or $U_C \sqsubseteq \neg U_C$, then all three such assertions are in $cln(\mathcal{T})$;
11. if $U_{R_1} \sqsubseteq U_{R_2}$ is in \mathcal{T} and $\rho(U_{R_2}) \sqsubseteq \neg E$ or $E \sqsubseteq \neg \rho(U_{R_2})$ is in $cln(\mathcal{T})$, then also $\rho(U_{R_1}) \sqsubseteq \neg E$ is in $cln(\mathcal{T})$;
12. if $U_{R_1} \sqsubseteq U_{R_2}$ is in \mathcal{T} and $\delta(U_{R_2}) \sqsubseteq \neg P$ or $P \sqsubseteq \neg \delta(U_{R_2})$ is in $cln(\mathcal{T})$, then also $\delta(U_{R_1}) \sqsubseteq \neg P$ is in $cln(\mathcal{T})$;
13. if $U_{R_1} \sqsubseteq U_{R_2}$ is in \mathcal{T} and $U_{R_2} \sqsubseteq \neg U_{R_3}$ or $U_{R_3} \sqsubseteq \neg U_{R_2}$ is in $cln(\mathcal{T})$, then also $U_{R_1} \sqsubseteq \neg U_{R_3}$ is in $cln(\mathcal{T})$;
14. if one of the assertions $\rho(U_R) \sqsubseteq \neg \rho(U_R)$, $\delta(U_R) \sqsubseteq \neg \delta(U_R)$, or $U_R \sqsubseteq \neg U_R$ is in $cln(\mathcal{T})$, then all three such assertions are in $cln(\mathcal{T})$.

It is obvious that the same definition is applied to both DL-Lite_A and DL-Lite_{AS}, since the ABox is not involved in the definition. Moreover, because each DL-Lite language we mentioned in this section is a sublanguage of DL-Lite_{AS}, the definition is applied to all of them.

3. UPDATE AND ERASURE AT THE INSTANCE-LEVEL

In this chapter we study the notion of update and erasure of a DL knowledge base. We focus on the changes concerning only the instance-level, that is the ABox. Firstly, we give the necessary definitions [DGLPR06].

Definition 3.1 (Containment between interpretations). Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{I}' = \langle \Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'} \rangle$ be two interpretations (over the same alphabet). We say that \mathcal{I} is contained in \mathcal{I}' , written $\mathcal{I} \subseteq \mathcal{I}'$, if $\mathcal{I}, \mathcal{I}'$ are such that:

- i $A^{\mathcal{I}} \subseteq A^{\mathcal{I}'}$, for every atomic concept A .
- ii $P^{\mathcal{I}} \subseteq P^{\mathcal{I}'}$, for every atomic role P .
- iii $U_C^{\mathcal{I}} \subseteq U_C^{\mathcal{I}'}$, for every atomic concept attribute U_C .
- iv $U_R^{\mathcal{I}} \subseteq U_R^{\mathcal{I}'}$, for every atomic role attribute U_R .

We say that \mathcal{I} is properly contained \mathcal{I}' , written $\mathcal{I} \subset \mathcal{I}'$, if $\mathcal{I} \subseteq \mathcal{I}'$ and $\mathcal{I}' \not\subseteq \mathcal{I}$.

Definition 3.2 (Difference between interpretations). Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{I}' = \langle \Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'} \rangle$ be two interpretations (over the same alphabet). We define the difference between \mathcal{I} and \mathcal{I}' , written $\mathcal{I} \ominus \mathcal{I}'$, as the interpretation $\langle \Delta^{\mathcal{I} \ominus \mathcal{I}'}, \cdot^{\mathcal{I} \ominus \mathcal{I}'} \rangle$ such that:

- i $A^{\mathcal{I} \ominus \mathcal{I}'} = A^{\mathcal{I}} \ominus A^{\mathcal{I}'}$, for every atomic concept A .
- ii $P^{\mathcal{I} \ominus \mathcal{I}'} = P^{\mathcal{I}} \ominus P^{\mathcal{I}'}$, for every atomic role P .
- iii $U_C^{\mathcal{I} \ominus \mathcal{I}'} = U_C^{\mathcal{I}} \ominus U_C^{\mathcal{I}'}$, for every atomic concept attribute U_C .
- iv $U_R^{\mathcal{I} \ominus \mathcal{I}'} = U_R^{\mathcal{I}} \ominus U_R^{\mathcal{I}'}$, for every atomic role attribute U_R .

where $S \ominus S'$ denotes the symmetric difference between sets S and S' , i.e. $S \ominus S' = (S \cup S') \setminus (S \cap S')$.

Definition 3.3 (Model update). Let \mathcal{T} be a TBox in a DL \mathcal{L} , \mathcal{I} a model of \mathcal{T} , and \mathcal{F} a finite set of membership assertions in \mathcal{L} such that $Mod(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$. The update of \mathcal{I} with \mathcal{F} , denoted $U^{\mathcal{I}}(\mathcal{I}, \mathcal{F})$, is the set of interpretations defined as follows:

$$U^{\mathcal{I}}(\mathcal{I}, \mathcal{F}) = \{ \mathcal{I}' \mid \mathcal{I}' \in Mod(\mathcal{T} \cup \mathcal{F}) \text{ and} \\ \text{there exists no } \mathcal{I}'' \in Mod(\mathcal{T} \cup \mathcal{F}) \text{ s.t. } \mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}' \}.$$

Intuitively, updated models not only satisfy an insertion \mathcal{F} , but the made changes must be minimal.

Definition 3.4 (Update). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB expressed in a DL \mathcal{L} , and \mathcal{F} a finite set of membership assertions expressed in \mathcal{L} such that $Mod(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$. The *update of \mathcal{K} with \mathcal{F}* , denoted $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$, is defined as follows:

$$\mathcal{K} \circ_{\mathcal{T}} \mathcal{F} = \bigcup_{\mathcal{I} \in Mod(\mathcal{K})} U^{\mathcal{T}}(\mathcal{I}, \mathcal{F}).$$

Definition 3.5 (Erasure). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB expressed in a DL \mathcal{L} , and \mathcal{F} a finite set of membership assertions expressed in \mathcal{L} such that $Mod(\mathcal{T} \cup \neg\mathcal{F}) \neq \emptyset$, where $\neg\mathcal{F}$ denotes the set of membership assertions $\{\neg F_i \mid F_i \in \mathcal{F}\}$. The *erasure of \mathcal{K} with \mathcal{F}* , denoted $\mathcal{K} \bullet_{\mathcal{T}} \mathcal{F}$, is defined as follows:

$$\mathcal{K} \bullet_{\mathcal{T}} \mathcal{F} = Mod(\mathcal{K}) \cup \left(\bigcup_{\mathcal{I} \in Mod(\mathcal{K})} U^{\mathcal{T}}(\mathcal{I}, \neg\mathcal{F}) \right).$$

Note that the notion of erasure is quite close to the notion of update, it has a little but important difference. Erasure is not just update with negation of information (that is, having $P(a)$, we do update with $\neg P(a)$), which would forbid the state of the world in which the erased information is true (that is $P(a)$ is not satisfied by any model of an updated KB), but it is proposed to *erase* information by making it uncertain (that is $P(a)$ may either appear or not appear in a model).

Hereby, the object of change is not the theory itself, but the set of its models. Then, it seems reasonable to have result of an update (or erasure) expressed in the same language as the original knowledge base.

Definition 3.6 (Expressible update (erasure)). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base expressed in a DL \mathcal{L} and \mathcal{F} a set of membership assertions expressed in \mathcal{L} such that $Mod(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$. We say that the *update* (resp., *erasure*) of \mathcal{K} with \mathcal{F} is *expressible in \mathcal{L}* iff there exists an ABox \mathcal{A}' expressed in \mathcal{L} such that $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F} = Mod(\langle \mathcal{T}, \mathcal{A}' \rangle)$ (resp., $\mathcal{K} \bullet_{\mathcal{T}} \mathcal{F} = Mod(\langle \mathcal{T}, \mathcal{A}' \rangle)$).

But unfortunately, an update may not be an expressible one, as it is shown in the following example.

Example 3.1. Consider the following DL-Lite_{core} Knowledge Base \mathcal{K} that follows:

TBox \mathcal{T}	ABox \mathcal{A}
$A_1 \sqsubseteq \neg \exists P$	$\exists P(a)$
$\exists P \sqsubseteq \neg A_2$	
$\exists P^- \sqsubseteq A_3$	

We want to compute the (DL-Lite_{core}, \mathcal{T})-update of \mathcal{K} with $\mathcal{F} = \{A_1(a)\}$. We have that $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F} \models \neg A_2(a)$. Moreover, for each model \mathcal{I} of \mathcal{K} for each tuple

$(a, x) \in P^{\mathcal{I}}$, x should be still interpreted as belonging to $A_3^{\mathcal{I}}$. Thus, $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F} \models \exists x A_3(x)$.

There is no way to express neither the former nor the latter logical implication in DL-Lite_{core} (nor in DL-Lite_R, nor in DL-Lite_A).

△

It follows from this that neither DL-Lite_F, nor DL-Lite_R, nor DL-Lite_A updates are expressible in the same logic. Our goal is given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a set of membership assertions \mathcal{F} to find an ABox \mathcal{A}' such that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) = \mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$ (resp., $= \mathcal{K} \bullet_{\mathcal{T}} \mathcal{F}$). So, one of the possibilities to solve the problem of unexpressibility is:

- to find a DL language that in which update and erasure can be expressed;
- to approximate an update (resp., an erasure) of the found language to the original one.

As we will see, in both DL-Lite_{RS} and DL-Lite_{AS} update and erasure can be expressed. That is why we begin with updating the DL-Lite_S family.

3.1 Update and Erasure in DL-Lite_{RS}

In this section we consider the problem of update and erasure of DL-Lite_{RS} at the instance-level. The algorithm in Fig. 3.1 described in the first subsection offers a way to update DL-Lite_{RS}. The correctness of the algorithm is shown in the subsection as well.

In the second subsection we consider an algorithm (Fig. 3.3) of instance-level erasure of DL-Lite_{RS}, which is based on the the previous one.

3.1.1 Update in DL-Lite_{RS}

The algorithm in the Fig. 3.1 takes as input a DL-Lite_{RS} knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a set of DL-Lite_{RS} membership assertions \mathcal{F} , and returns as output a DL-Lite_{RS} ABox \mathcal{A}' such that $Mod(\mathcal{K}') = \mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$, where $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$. Or more precisely:

1. First, it checks whether \mathcal{T} and \mathcal{F} are satisfiable (line 1), and if they are, it adds to \mathcal{F} all membership assertions that are logically implied by \mathcal{F} . The membership assertions $\exists R(a)$ and $\exists R^-(b)$ come from $R(a, b)$ trivially (lines 3-4).
2. Second, the algorithm computes the set \mathcal{F}' of all membership assertions that contradict \mathcal{F} and are logically implied by \mathcal{K} (lines 5-19).

In the case, when $f \in \mathcal{F}$ and it is a concept assertion, the algorithm calls the function $Saturate(\neg f, \mathcal{T})$ that returns the set \mathcal{F}_C of membership assertions which, according to \mathcal{T} , contradict \mathcal{F} (line 8). If a formula $f_C \in \mathcal{F}_C$ is logically

ALGORITHM $ComputeUpdate_{RS}(\mathcal{T}, \mathcal{A}, \mathcal{F})$

INPUT: finite set of ground membership assertions \mathcal{F} , satisfiable with $DL-Lite_{RS}$

KB $\langle \mathcal{T}, \mathcal{A} \rangle$

OUTPUT: an ABox \mathcal{A}' , or ERROR

```

[1]  if  $\langle \mathcal{T}, \mathcal{F} \rangle$  is not satisfiable then return ERROR
[2]  else
[3]    {for each  $f \in \mathcal{F}$  do
[4]      if  $f = R(a, b)$  then  $\mathcal{F} := \mathcal{F} \cup \{\exists R(a), \exists R^-(b)\}$ 
[5]       $\mathcal{F}' := \emptyset$ 
[6]    for each  $f \in \mathcal{F}$  do
[7]      if  $f = C(a)$  then
[8]         $\{\mathcal{F}_C := Saturate(\neg C(a), \mathcal{T})$ 
[9]        for each  $f_C \in \mathcal{F}_C$  do
[10]          {if  $\langle \mathcal{T}, \mathcal{A} \rangle \models f_C$  then  $\mathcal{F}' := \mathcal{F}' \cup \{f_C\}$ 
[11]          if  $f_C = \exists R(a)$  then  $\mathcal{F}' := \mathcal{F}' \cup \{R(a, b) \mid R(a, b) \in \mathcal{A}\}$ 
[12]          }
[13]        }
[14]      else
[15]        if  $f = R(a, b)$  then
[16]           $\{\mathcal{F}_R = Saturate(\neg R(a, b), \mathcal{T})$ 
[17]          for each  $f_R \in \mathcal{F}_R$  do
[18]            { if  $\langle \mathcal{T}, \mathcal{A} \rangle \models f_R$  then  $\mathcal{F}' := \mathcal{F}' \cup \{f_R\}$  }
[19]          }
[20]       $\mathcal{A}' := \mathcal{A} \cup \mathcal{F}'$ ;
[21]    for each  $f' \in \mathcal{F}'$  do
[22]      if  $f' = C(a)$  then
[23]         $\{\mathcal{A}' := \mathcal{A}' \setminus \{C(a)\}$ 
[24]        for each  $C \sqsubseteq C_1$  in  $cl(\mathcal{T})$  do
[25]          if  $(C_1(a) \notin \mathcal{F}')$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_1(a)\}$ 
[26]          if  $f' = \exists R(a)$  then
[27]            for each  $\exists R^- \sqsubseteq C_2$  in  $cl(\mathcal{T})$  do
[28]               $\mathcal{A}' := \mathcal{A}' \cup \{C_2(z_{\exists R(a)})\}$ , with  $z_{\exists R(a)}$  new variable
[29]            }
[30]        }
[31]      else
[32]        if  $f' = R(a, b)$  then
[33]           $\{\mathcal{A}' := \mathcal{A}' \setminus \{R(a, b), \exists R(a), \exists R^-(b)\}$ 
[34]          for each  $R \sqsubseteq R_1$  in  $cl(\mathcal{T})$  do
[35]            if  $R_1(a, b) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{R_1(a, b)\}$ ;
[36]          for each  $\exists R \sqsubseteq C_3$  in  $cl(\mathcal{T})$  do
[37]            if  $C_3(a) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_3(a)\}$ 
[38]          for each  $\exists R^- \sqsubseteq C_4$  in  $cl(\mathcal{T})$  do
[39]            if  $C_4(b) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_4(b)\}$ 
[40]          }
[41]        }
[42]      }
[43]    }

```

Fig. 3.1: Algorithm $ComputeUpdate_{RS}$

ALGORITHM *Saturate*(f, \mathcal{T})
INPUT: a membership assertion f , TBox \mathcal{T}
OUTPUT: set \mathcal{P} of ground membership assertions
 $\mathcal{P} = \{f\}$
repeat
 $\mathcal{P}' := \mathcal{P}$
 for each $f \in \mathcal{P}'$
 case $f =$
 $C(a)$ **then**
 if $C' \sqsubseteq C \in cl(\mathcal{T})$ **then** $\mathcal{P} := \mathcal{P} \cup \{C'(a)\}$
 $R(a, b)$ **then**
 if $R' \sqsubseteq R \in cl(\mathcal{T})$ **then** $\mathcal{P} := \mathcal{P} \cup \{R'(a, b)\}$
until $\mathcal{P} = \mathcal{P}'$

Fig. 3.2: Algorithm *Saturate*

implied by a KB \mathcal{K} , it adds f_C to \mathcal{F}' (line 10). Moreover, if f_C is of the form $\exists R(a)$, it adds to \mathcal{F}' all role membership assertions $R(a, b)$ which are in \mathcal{A} (line 11).

Similarly, if f is a role membership assertions of the form $R(a, b)$, the algorithm calls the function *Saturate*($\neg f, \mathcal{T}$) (line 16). If an assertion $f_R \in \mathcal{F}_R$, where \mathcal{F}_R is the set returned by *Saturate* contradicts \mathcal{F} , the algorithm adds it to \mathcal{F}' .

3. Third, it computes the result set \mathcal{A}' . For each membership assertion f' from \mathcal{A} , the algorithm deletes it if it belongs to \mathcal{F}' , but adds those assertions which are logically implied by \mathcal{F}' and do not contradict \mathcal{F} (lines 19-35).

We denote by $cl(\mathcal{T})$ the deductive closure of \mathcal{T} , which can be defined as the generalization of $cln(\mathcal{T})$, notably:

Definition 3.7. Let \mathcal{T} be a DL-Lite_{AS} TBox. The *closure* of \mathcal{T} , denoted by $cl(\mathcal{T})$, is the TBox obtained inductively as follows:

1. all inclusion assertions in \mathcal{T} are also in $cl(\mathcal{T})$;
2. the negative closure of \mathcal{T} $cln(\mathcal{T})$ is a subset of $cl(\mathcal{T})$;
3. the relation “ \sqsubseteq ” is transitive in $cl(\mathcal{T})$, i.e. if $L \sqsubseteq M$ and $M \sqsubseteq R$ are in $cl(\mathcal{T})$, then also $L \sqsubseteq R$ is in $cl(\mathcal{T})$;
4. if $Q \sqsubseteq R$ is in $cl(\mathcal{T})$, then also $\exists Q \sqsubseteq \exists R$ and $\exists Q^- \sqsubseteq \exists R^-$ are in $cl(\mathcal{T})$;
5. if $U \sqsubseteq V$ is in $cl(\mathcal{T})$, then also $\delta(U) \sqsubseteq \delta(V)$ and $\rho(U) \sqsubseteq \rho(V)$ are in $cl(\mathcal{T})$, where U is either U_C or U_R , and V is either V_C or V_R ;
6. if $B \sqsubseteq \exists Q.C$, then also $B \sqsubseteq \exists Q$;

Let us prove the correctness of the algorithm step by step. We start from satisfiability.

Lemma 3.1 (Satisfiability). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable DL-Lite $_{\mathcal{RS}}$ knowledge base, \mathcal{F} a finite set of ground DL-Lite $_{\mathcal{RS}}$ membership assertions such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and \mathcal{K}' the DL-Lite $_{\mathcal{RS}}$ knowledge base such that $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{A}' = \text{ComputeUpdate}_{\mathcal{RS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. We have that \mathcal{K}' is satisfiable.*

Proof. According to Algorithm 3.1, \mathcal{K}' is obtained by

1. inserting into \mathcal{A} :
 - a finite set of membership assertions \mathcal{F} such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$.
 - a finite set of membership assertions \mathcal{F}'' that do not contradict \mathcal{F} and are logically implied by \mathcal{K} ; it is done in the lines 23-27, 32-35.

Thus, $\text{Mod}(\mathcal{T} \cup \mathcal{F} \cup \mathcal{F}'') \neq \emptyset$;

2. deleting from \mathcal{A} the maximum set of membership assertions \mathcal{F}' that contradict \mathcal{F} .

Thereby, we have that $\mathcal{A}' = (\mathcal{A} \cup \mathcal{F} \cup \mathcal{F}'') \setminus \mathcal{F}'$. Then, by the data \mathcal{K} is satisfiable (i.e. $\text{Mod}(\mathcal{T} \cup \mathcal{A}) \neq \emptyset$), this leads to satisfiability of \mathcal{K}' : $\text{Mod}(\mathcal{T} \cup \mathcal{A}') = \text{Mod}(\mathcal{T} \cup [(\mathcal{A} \cup \mathcal{F} \cup \mathcal{F}'') \setminus \mathcal{F}']) = \text{Mod}((\mathcal{T} \cup \mathcal{A} \cup \mathcal{F} \cup \mathcal{F}'') \setminus \mathcal{F}') \neq \emptyset$. \square

Lemma 3.2 (Termination). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{RS}}$ knowledge base, \mathcal{F} a finite set of ground DL-Lite $_{\mathcal{RS}}$ membership assertions. Then the algorithm $\text{ComputeUpdate}_{\mathcal{RS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$ terminates, returning *ERROR* if $\text{Mod}(\mathcal{T} \cup \mathcal{F}) = \emptyset$, and an ABox \mathcal{A}' such that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is a DL-Lite $_{\mathcal{RS}}$ knowledge base, otherwise.*

Proof. The proof of the theorem follows from the observations below:

1. The algorithm calls *Saturate* a finite number of times. This algorithm stops, since $cl(\mathcal{T})$ is finite for a finite \mathcal{T} , returning a finite output [CDGL⁺07].
2. The algorithm of checking whether $\mathcal{K} \models F$ for some F terminates; the number of membership assertions to be checked is finite since they belong to outputs of *Saturate*.
3. Deleting membership assertions of \mathcal{F}' and introducing new assertions (lines 20-35) terminate since both \mathcal{F}' and $cl(\mathcal{T})$ is finite.

\square

Now we consider the completeness and soundness of the algorithm.

Lemma 3.3 (Soundness). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{RS} knowledge base, \mathcal{F} a finite set of ground DL-Lite_{RS} membership assertions such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and \mathcal{K}' the DL-Lite_{RS} knowledge base such that $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{A}' = \text{ComputeUpdate}_{\text{RS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then, for every model $\mathcal{I}' \in \text{Mod}(\mathcal{K}')$, we have that:*

$$\exists \mathcal{I} \in \text{Mod}(\mathcal{K}) \text{ such that } \mathcal{I}' \in U^{\mathcal{T}}(\mathcal{I}, \mathcal{F}).$$

Proof. The structure of the proof is as follows: first, we show how to build a model \mathcal{I} of \mathcal{K} given $\mathcal{I}' \in \mathcal{K}'$, then we prove, that for the obtained model \mathcal{I} , $\mathcal{I}' \in U^{\mathcal{T}}(\mathcal{I}, \mathcal{F})$.

Let \mathcal{I}' be a model of \mathcal{K}' and not a model of \mathcal{K} (otherwise the theorem is proved trivially). By construction, \mathcal{I}' is a model of \mathcal{T} , that means that there exists a subset \mathcal{F}' of \mathcal{A} such that for each $F \in \mathcal{F}'$, $\mathcal{I}' \notin \text{Mod}(\{F\})$.

We build a model \mathcal{I} step by step as it is described below.

STEP 1 We set $\mathcal{I}_0 := \mathcal{I}'$.

STEP 2 We modify \mathcal{I}_0 as follows. For each $F \in \mathcal{F}'$ do:

1. if $F = C(a)$, then $a^{\mathcal{I}_0} \in C^{\mathcal{I}_0}$.
2. if $F = Q(a, b)$, then $(a^{\mathcal{I}_0}, b^{\mathcal{I}_0}) \in Q^{\mathcal{I}_0}$, $a^{\mathcal{I}_0} \in \exists Q^{\mathcal{I}_0}$, and $b^{\mathcal{I}_0} \in \exists Q^{-\mathcal{I}_0}$.

STEP 3 We set $\mathcal{I}_i := \mathcal{I}_{i-1}$ and $i := 1$.

Repeat the following rules:

1. If $a \in B^{\mathcal{I}_{i-1}}$, $B \sqsubseteq C \in \mathcal{T}$ and $a \notin C^{\mathcal{I}_{i-1}}$, then set $a \in C^{\mathcal{I}_i}$.
2. If $a \in \exists Q^{\mathcal{I}_{i-1}}$ and there is no $b \in \Delta^{\mathcal{I}_{i-1}}$ such that $(a, b) \in Q^{\mathcal{I}_{i-1}}$, then add $(a, b') \in Q^{\mathcal{I}_i}$ and $b' \in \exists Q^{-\mathcal{I}_i}$, where b' is a new element.
3. If $(a, b) \in Q^{\mathcal{I}_{i-1}}$, $Q \sqsubseteq R \in \mathcal{T}$ and $(a, b) \notin R^{\mathcal{I}_{i-1}}$, then set $(a, b) \in R^{\mathcal{I}_i}$.
4. If $a \in \neg \exists Q^{\mathcal{I}_{i-1}}$, then for each $(a, b) \in Q^{\mathcal{I}_{i-1}}$ we set $(a, b) \notin Q^{\mathcal{I}_i}$, and if there exists no $a' \neq a$ such that $(a', b) \in Q^{\mathcal{I}_{i-1}}$, then we set $b \notin \exists Q^{-\mathcal{I}_i}$.
5. $i := i + 1$.

Until $\mathcal{I}_i = \mathcal{I}_{i-1}$

We apply STEP 3 until $\mathcal{I}_{n+1} = \mathcal{I}_n$, and \mathcal{I}_n will be the required interpretation \mathcal{I} . It is obvious that \mathcal{I} is a model of \mathcal{T} (\mathcal{I}_0 is a model of \mathcal{T} , and in STEP 3 we make those settings that are logically implied by \mathcal{T}). Also, \mathcal{I} satisfies \mathcal{F}' by construction (STEP 2). Furthermore, \mathcal{I} satisfies all membership assertions in \mathcal{A} (it satisfies both \mathcal{A}' and \mathcal{F}').

Now, let us show that $\mathcal{I}' \in U^{\mathcal{T}}(\mathcal{I}, \mathcal{F})$. As we mentioned before, \mathcal{I}' is a model of \mathcal{T} ; moreover, it is also a model of \mathcal{F} , since $\mathcal{F} \subseteq \mathcal{A}'$ and \mathcal{I}' is a model of \mathcal{A}' . By definition, $\mathcal{I}' \in U^{\mathcal{T}}(\mathcal{I}, \mathcal{F})$ if and only if there is no interpretation \mathcal{I}'' such that $\mathcal{I}'' \in \text{Mod}(\mathcal{T} \cup \mathcal{F})$ and $\mathcal{I} \oplus \mathcal{I}'' \subset \mathcal{I} \oplus \mathcal{I}'$.

Let us suppose that there exists such \mathcal{I}'' . The inclusion $\mathcal{I} \oplus \mathcal{I}'' \subset \mathcal{I} \oplus \mathcal{I}'$ means that there is some difference in interpretations \mathcal{I} , \mathcal{I}'' and \mathcal{I}' , i.e. one of the following cases occurs:

1. There is a such that $a \in A^{\mathcal{I}}$ and $a \in A^{\mathcal{I}''}$, but $a \notin A^{\mathcal{I}'}$.
2. There is a such that $a \notin A^{\mathcal{I}}$ and $a \notin A^{\mathcal{I}''}$, but $a \in A^{\mathcal{I}'}$.
3. There is (a, b) such that $(a, b) \in Q^{\mathcal{I}}$ and $(a, b) \in Q^{\mathcal{I}''}$, but $(a, b) \notin Q^{\mathcal{I}'}$.
4. There is (a, b) such that $(a, b) \notin Q^{\mathcal{I}}$ and $(a, b) \notin Q^{\mathcal{I}''}$, but $(a, b) \in Q^{\mathcal{I}'}$.

Let us consider all these cases:

1. In this case, since \mathcal{I} is built from \mathcal{I}' in the way described above, there are two possibilities:
 - $A(a) \in \mathcal{F}'$ and $A(a) \in \mathcal{A}$, that is $A(a)$ was removed by the algorithm *ComputeUpdate_{RS}*. In this case, $A(a) \in \text{Saturate}(\neg C(a), \mathcal{T})$ for some $C(a) \in \mathcal{F}$ (iff $A \sqsubseteq \neg C \in \text{cl}(\mathcal{T})$), that is $A(a)$ contradicts \mathcal{F} . Since \mathcal{I}'' is a model of \mathcal{T} , $a \in A^{\mathcal{I}''}$ implies $a \notin C^{\mathcal{I}''}$. The latter expression contradicts the fact that \mathcal{I}'' is a model of \mathcal{F} .
 - a was set in $A^{\mathcal{I}}$ in STEP 3 in the first item of building \mathcal{I} . This means that $a \in B^{\mathcal{I}}$ and $B \sqsubseteq A \in \mathcal{T}$, but $a \notin A^{\mathcal{I}'}$. That is firstly $a \in B^{\mathcal{I}}$ ¹ was introduced to satisfy \mathcal{F}' , and $a \in A^{\mathcal{I}}$ was introduced to satisfy \mathcal{T} . So, if $a \in B^{\mathcal{I}''}$, it is not a model of \mathcal{F} , and if $a \notin B^{\mathcal{I}''}$ it is not model of \mathcal{T} . Thus, we have a contradiction.
2. We can reduce this case to the previous one by considering that $a \in \neg A^{\mathcal{I}}$, $a \in \neg A^{\mathcal{I}''}$, and $a \notin \neg A^{\mathcal{I}'}$.
3. As it is in the first case, an assertion $Q(a, b)$ belongs to \mathcal{F}' , where $\mathcal{F}' \subseteq \mathcal{A}$ and it was removed by *ComputeUpdate_{RS}* because of a contradiction with \mathcal{F} . So, we have several possibilities:
 - $Q(a, b) \in \text{Saturate}(\neg Q_1(a, b), \mathcal{T})$, $Q \sqsubseteq Q_1 \in \text{cl}(\mathcal{T})$, and $Q_1(a, b) \in \mathcal{F}$. Thus, $(a, b) \in Q^{\mathcal{I}''}$ yields $(a, b) \notin Q_1^{\mathcal{I}''}$, that contradicts that \mathcal{I}'' is a model of \mathcal{F} .
 - For some $C(a) \in \mathcal{F}$, $Q(a, b)$ logically implies $\neg C(a)$, that comes from *Saturate*. Here we have a contradiction as well as in the previous case.

Another possibility is that (a, b) was inserted in $Q^{\mathcal{I}}$ in STEP 3 of building \mathcal{I} . We have two possibilities here:

- (a, b) was set in $Q^{\mathcal{I}}$ in STEP 3 in the third item of building \mathcal{I} . This means that $(a, b) \in Q^{\mathcal{I}}$ and $Q' \sqsubseteq Q \in \mathcal{T}$, but $(a, b) \notin Q^{\mathcal{I}'}$. That is firstly $(a, b) \in Q^{\mathcal{I}2}$ was set to satisfy \mathcal{F}' , and $(a, b) \in Q^{\mathcal{I}}$ was set to satisfy \mathcal{T} . So, if $(a, b) \in Q^{\mathcal{I}''}$, it is not a model of \mathcal{F} , and if $(a, b) \notin Q^{\mathcal{I}''}$ it is not model of \mathcal{T} . Thereby, we have a contradiction.

¹ We can suppose this without loss of generality.

² We can suppose this without loss of generality.

- (a, b) was inserted in $Q^{\mathcal{I}}$ in STEP 3 in the second item of building \mathcal{I} . It means that $a \in \exists Q^{\mathcal{I}}$ and there is no b' such that $(a, b') \in Q^{\mathcal{I}'}$. It means that $\exists Q(a)$ is logically implied by some assertion in F' that contradicts F . Thus, if $(a, b) \in Q^{\mathcal{I}''}$, we have a contradiction, since $a \in \exists Q^{\mathcal{I}''}$ implies that \mathcal{I}'' is not model of \mathcal{F} .

4. Let us consider the last case. Here we have several possibilities:

- Firstly (a, b) belonged to $Q^{\mathcal{I}^i}$ for some $i < j$, but it was removed on the j -th iteration by the rule four of the second step, so $a \in \neg \exists Q^{\mathcal{I}}$. Thus, $(a, b) \in Q^{\mathcal{I}}$ occurs at the first step to satisfy \mathcal{F}' , but then it was removed to satisfy \mathcal{T} ($a \in \neg \exists Q^{\mathcal{I}}$ appeared at the second step at some iteration). So, if $(a, b) \in Q^{\mathcal{I}''}$, then \mathcal{I}'' is not a model of \mathcal{F} ; if $(a, b) \notin Q^{\mathcal{I}''}$, then \mathcal{I}'' is not a model of \mathcal{T} . Thereby, we have contradiction.
- The case when $(a, b) \in Q^{\mathcal{I}}$ and $b \in \exists Q^{-\mathcal{I}}$ is absolutely analogous to the previous one.

Therefore we have that there is no interpretation \mathcal{I}'' such that $\mathcal{I}'' \in \text{Mod}(\mathcal{T} \cup \mathcal{F})$ and $\mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}'$ that yields $\mathcal{I}' \in U^{\mathcal{T}}(\mathcal{I}, \mathcal{F})$. □

Lemma 3.4 (Completeness). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{RS} knowledge base, \mathcal{F} a finite set of ground DL-Lite_{RS} membership assertions such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and \mathcal{K}' the DL-Lite_{RS} knowledge base such that $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{A}' = \text{ComputeUpdate}_{\text{RS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then, for every model $\mathcal{I} \in \text{Mod}(\mathcal{K})$, we have that:*

$$U^{\mathcal{T}}(\mathcal{I}, \mathcal{F}) \subseteq \text{Mod}(\mathcal{K}').$$

Proof. Let us suppose that there exists an interpretation $\mathcal{I}'' \in U^{\mathcal{T}}(\mathcal{I}, \mathcal{F}) \setminus \text{Mod}(\mathcal{K}')$. Then, \mathcal{I}'' does not satisfy a membership assertion $F' \in \mathcal{A}' \setminus \mathcal{F}$. By construction of \mathcal{A}' , it contains all the assertions of \mathcal{A} , that do not contradict \mathcal{F} , and those assertions that were introduced into \mathcal{A}' at lines 24, 27, 33, 35, 37 of the algorithm *ComputeUpdate_{RS}* that are logically implied by \mathcal{K} and do not contradict \mathcal{F} . Then, let us modify \mathcal{I}'' in order to make it satisfy F' . For this, we will use the procedure described below. Note that we can suppose that F' is the only membership assertion belonging to $\mathcal{A}' \setminus \mathcal{F}$ which is not satisfied by \mathcal{I}'' ³.

So, the procedure is described below:

STEP 1 We set $\check{\mathcal{I}} = \mathcal{I}''$.

STEP 2 We modify $\check{\mathcal{I}}$ as follows:

1. If $F' = C(a)$, then we set $a \in C^{\check{\mathcal{I}}}$.

³ Otherwise, we can apply the procedure the necessary number of times to obtain a model that does not satisfy exactly one membership assertion.

2. If $F' = C(z)$, where z is a variable, then we find a constant $b \in C^{\mathcal{I}}$, and we set $b \in C^{\check{\mathcal{I}}}$. Note that such a constant always exists. F' is inserted into \mathcal{A}' in the line 27, that is there exists a constant a that belongs to $\exists R^{\check{\mathcal{I}}}$ for some role R ; this leads to that there exists b such that $(a, b) \in R^{\check{\mathcal{I}}}$.
3. If $F' = R(a, b)$, then we set $(a, b) \in R^{\check{\mathcal{I}}}$, $a \in \exists R^{\check{\mathcal{I}}}$, $b \in \exists R^{-\check{\mathcal{I}}}$.

STEP 3 We apply recursively the following rules in order to make $\check{\mathcal{I}}$ satisfy \mathcal{T} .

1. If $a \in B^{\check{\mathcal{I}}}$, $B \sqsubseteq C \in \mathcal{T}$, and $a \notin C^{\mathcal{I}''}$, then we set $a \in C^{\check{\mathcal{I}}}$.
2. If $a \in \exists Q^{\check{\mathcal{I}}}$ and there exists no individual $b \in \Delta$ such that $(a, b) \in Q^{\mathcal{I}''}$, then for each $(a, b') \in Q^{\check{\mathcal{I}}}$, set $(a, b') \in Q^{\check{\mathcal{I}}}$.
3. If $a \in \exists Q^{-\check{\mathcal{I}}}$ and there exists no individual $b \in \Delta$ such that $(b, a) \in Q^{\mathcal{I}''}$, then for each $(b', a) \in Q^{\check{\mathcal{I}}}$, set $(b', a) \in Q^{\check{\mathcal{I}}}$.
4. If $(a, b) \in Q^{\check{\mathcal{I}}}$, $Q \sqsubseteq R \in \mathcal{T}$, and $(a, b) \notin R^{\check{\mathcal{I}}}$, then we set $(a, b) \in R^{\check{\mathcal{I}}}$.

It is obvious that the interpretation $\check{\mathcal{I}}$ is a model of \mathcal{T} (it is provided by STEP 3). Also $\check{\mathcal{I}}$ satisfies F' by construction (STEP 2). Moreover, $\check{\mathcal{I}}$ satisfies all other membership assertions in \mathcal{A}' (since nothing was deleted). The latter affirmation yields that $\check{\mathcal{I}}$ is a model of \mathcal{K}' .

Finally, let us show that $\mathcal{I} \ominus \check{\mathcal{I}} \subset \mathcal{I} \ominus \mathcal{I}''$. We know that membership assertion F' logically implied by \mathcal{K} , so \mathcal{I} and $\check{\mathcal{I}}$ agree on interpreting this part, but not \mathcal{I}'' . The same situation with setting in STEP 3. The rest is interpreted by both \mathcal{I}'' and $\check{\mathcal{I}}$ in the same way. Thus, we have that $\mathcal{I} \ominus \check{\mathcal{I}} \subset \mathcal{I} \ominus \mathcal{I}''$. Contradiction with minimality of \mathcal{I}'' (see Definition 3.3).

Thereby, $\mathcal{I}'' \in \text{Mod}(\mathcal{K}')$. □

The ABox \mathcal{A}' obtained by the algorithm $\text{ComputeUpdate}_{\mathcal{RS}}$ is a DL-Lite $_{\mathcal{RS}}$ ABox, which is proved in the following lemma.

Lemma 3.5 (Expressibility). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{RS}}$ knowledge base, \mathcal{F} a finite set of ground DL-Lite $_{\mathcal{RS}}$ membership assertions such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and $\mathcal{A}' = \text{ComputeUpdate}_{\mathcal{RS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then \mathcal{A}' is a DL-Lite $_{\mathcal{RS}}$ ABox.*

Proof. Note that those membership assertions that belong to $\mathcal{A}' \setminus (\mathcal{A} \cup \mathcal{F})$ are inserted in the lines 24, 27, 33 or 35, and they are DL-Lite $_{\mathcal{RS}}$ membership assertions by definition. □

From lemmas 3.1, 3.2, 3.3, 3.4, and 3.5 the following theorem follows directly.

Theorem 3.6. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{RS}}$ knowledge base, \mathcal{F} a finite set of ground DL-Lite $_{\mathcal{RS}}$ membership assertions. Then, the algorithm $\text{ComputeUpdate}_{\mathcal{RS}}$ returns *ERROR* if $\text{Mod}(\mathcal{T}) \cap \text{Mod}(\mathcal{F}) = \emptyset$, or returns \mathcal{A}' , otherwise, such that $\mathcal{K} \circ_{\mathcal{I}} \mathcal{F} \equiv \text{Mod}(\mathcal{K}')$, where $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$.*

ALGORITHM *ComputeErasure*_{RS}($\mathcal{T}, \mathcal{A}, \mathcal{F}$)

INPUT: finite set of ground membership assertions \mathcal{F} such that $\neg\mathcal{F}$ satisfiable with DL-Lite_{RS} KB $\langle \mathcal{T}, \mathcal{A} \rangle$

OUTPUT: an ABox \mathcal{A}' , or ERROR

```

[1]  if  $\langle \mathcal{T}, \neg\mathcal{F} \rangle$  is not satisfiable then return ERROR
[2]  else {
[3]     $\mathcal{A}'' = \text{ComputeUpdate}_{RS}(\mathcal{T}, \mathcal{A}, \neg\mathcal{F})$ 
[4]    for each  $f \in \neg\mathcal{F}$  do
[5]      if  $f = Q(a, b)$  then  $\neg\mathcal{F} := \neg\mathcal{F} \cup \{\exists Q(a), \exists Q^-(b)\}$ 
[6]     $\mathcal{A}' := \mathcal{A}'' \setminus \neg\mathcal{F}$ 
[7]  }
```

Fig. 3.3: *ComputeErasure*_{RS}

Now we turn to computational complexity.

Theorem 3.7. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{RS} knowledge base, \mathcal{F} a finite set of ground DL-Lite_{RS} membership assertions such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and $\mathcal{A}' = \text{ComputeUpdate}_{RS}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then:*

- the size of \mathcal{A}' is polynomially bounded by the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$;
- computing \mathcal{A}' can be done in polynomial time in the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$.

Proof. The theorem is proved by the following observations:

1. The function *Saturate* runs in polynomial time in the size of \mathcal{T} .
2. *Saturate* is called a polynomial number of times by *ComputeUpdate*_{RS}.
3. For each $f \in \text{Saturate}$, checking whether $\mathcal{K} \models f$ is LOGSPACE in \mathcal{A} .
4. For each $f' \in \mathcal{F}'$ the cost of eliminating f' from \mathcal{A}' is polynomial in the size of \mathcal{A} .

□

3.1.2 Erasure in DL-Lite_{RS}

In Fig. 3.3 we provide the algorithm for obtaining erasure for DL-Lite_{RS} knowledge bases. The algorithm takes a TBox \mathcal{T} , an ABox \mathcal{A} , and the set of membership assertions \mathcal{F} as input, and returns an ABox \mathcal{A}' such that $\text{Mod}(\langle \mathcal{T}, \mathcal{A}' \rangle) = \mathcal{K} \bullet_{\mathcal{T}} \mathcal{F}$ as output. It works as described below:

- First, an ABox $\mathcal{A}'' = \text{ComputeUpdate}_{RS}(\mathcal{T}, \mathcal{A}, \neg\mathcal{F})$ is obtained;
- Second, we get the required ABox \mathcal{A}' by removing $\neg\mathcal{F}$ from \mathcal{A}'' .

The soundness and completeness of the algorithm is considered in the following lemma.

Lemma 3.8 (Soundness and completeness). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{RS} knowledge base, \mathcal{F} a finite set of ground DL-Lite_{RS} membership assertions such that $\text{Mod}(\mathcal{T} \cup \neg\mathcal{F}) \neq \emptyset$, and \mathcal{K}' the DL-Lite_{RS} knowledge base such that $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{A}' = \text{ComputeErasure}_{\text{RS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then:*

$$\text{Mod}(\mathcal{K}') = \mathcal{K} \bullet_{\mathcal{T}} \mathcal{F}.$$

Proof. Let $\bar{\mathcal{A}}$ be the set of those membership assertions that were inserted into \mathcal{A}'' in the lines 24, 27, 33, 35, 37; those are the assertions that are logically implied by \mathcal{K} . For an assertion f , the fact that $\langle \mathcal{T}, \mathcal{A} \rangle \models f$ implies that $\text{Mod}(\langle \mathcal{T}, \mathcal{A} \rangle) \subseteq \text{Mod}(\{f\})$. The latter inclusion yields that

$$\begin{aligned} \text{Mod}(\langle \mathcal{T}, \mathcal{A} \rangle) &= \text{Mod}(\langle \mathcal{T}, \mathcal{A} \rangle) \cap \text{Mod}(\{f\}) \\ &= \text{Mod}(\langle \mathcal{T}, \mathcal{A} \rangle \cup \{f\}) = \text{Mod}(\mathcal{T} \cup \mathcal{A} \cup \{f\}) \\ &= \text{Mod}(\langle \mathcal{T}, \mathcal{A} \cup \{f\} \rangle). \end{aligned} \tag{3.1}$$

Since (3.1) holds for each $f \in \bar{\mathcal{A}}$, so $\text{Mod}(\langle \mathcal{T}, \mathcal{A} \rangle) = \text{Mod}(\langle \mathcal{T}, \mathcal{A} \cup \bar{\mathcal{A}} \rangle)$. Let us designate $\mathcal{A} \cup \bar{\mathcal{A}}$ as \mathcal{A}_0 .

According to Definition 3.5, $\mathcal{K} \bullet_{\mathcal{T}} \mathcal{F} = \text{Mod}(\mathcal{K}) \cup \mathcal{K} \circ_{\mathcal{T}} \neg\mathcal{F} = \text{Mod}(\mathcal{K}) \cup \text{Mod}(\mathcal{K}'')$. Since $\text{Mod}(\mathcal{K}) = \text{Mod}(\mathcal{K}_0)$ ⁴, we can write the following sequence of equations:

$$\begin{aligned} \mathcal{K} \bullet_{\mathcal{T}} \mathcal{F} &= \text{Mod}(\mathcal{K}_0) \cup \text{Mod}(\mathcal{K}'') \\ &= \text{Mod}(\mathcal{T} \cup \mathcal{A}_0) \cup \text{Mod}(\mathcal{T} \cup \mathcal{A}'') \\ &= \text{Mod}(\mathcal{T} \cup \mathcal{A}_0) \cup \left[\text{Mod}(\mathcal{T}) \cap \text{Mod}(\mathcal{A}'') \right] \\ &= \left[\text{Mod}(\mathcal{T} \cup \mathcal{A}_0) \cup \text{Mod}(\mathcal{T}) \right] \cap \left[\text{Mod}(\mathcal{T} \cup \mathcal{A}_0) \cup \text{Mod}(\mathcal{A}'') \right] \\ &= \text{Mod}\left((\mathcal{T} \cup \mathcal{A}_0) \cap \mathcal{T}\right) \cap \text{Mod}\left((\mathcal{T} \cup \mathcal{A}_0) \cap \mathcal{A}''\right) \\ &= \text{Mod}(\mathcal{T}) \cap \text{Mod}\left((\mathcal{T} \cap \mathcal{A}'') \cup (\mathcal{A}_0 \cap \mathcal{A}'')\right) \\ &= \text{Mod}(\mathcal{T}) \cap \text{Mod}\left((\emptyset \cup (\mathcal{A}_0 \cap \mathcal{A}''))\right). \end{aligned}$$

Note that the intersection $\mathcal{A}_0 \cap \mathcal{A}''$ includes (by construction of both \mathcal{A}_0 and \mathcal{A}'') (i) $\bar{\mathcal{A}}$; (ii) all membership assertions of \mathcal{A} that do not contradict $\neg\mathcal{F}$. In fact, the intersection is exactly \mathcal{A}' . Thus, we have that

$$\mathcal{K} \bullet_{\mathcal{T}} \mathcal{F} = \text{Mod}(\mathcal{T}) \cap \text{Mod}(\mathcal{A}') = \text{Mod}(\mathcal{K}').$$

□

The remaining properties of the algorithm (such as termination, satisfiability, etc.) come from the analogous properties of *ComputeUpdate*_{RS}.

⁴ $\mathcal{K}_0 = \langle \mathcal{T}, \mathcal{A}_0 \rangle$.

Theorem 3.9. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{RS} knowledge base, \mathcal{F} a finite set of ground DL-Lite_{RS} membership assertions. Then, the algorithm *ComputeErasure_{RS}* returns *ERROR* if $\text{Mod}(\mathcal{T}) \cap \text{Mod}(\neg\mathcal{F}) = \emptyset$, or returns a DL-Lite_{RS} ABox \mathcal{A}' , otherwise, such that $\mathcal{K} \bullet_{\mathcal{T}} \mathcal{F} \equiv \text{Mod}(\mathcal{K}')$, where $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$.*

3.2 Update and Erasure in DL-Lite_{AS}

This section is dedicated to the algorithm of updating DL-Lite_{AS} at the instance-level. It represents a kind of hybrid of DL-Lite_{RS} update algorithm described in the section 3.1, and the algorithm of update of DL-Lite_{FS} introduced in [Pog06]. Some number of DL-Lite_{AS} constructs can be viewed as syntactic sugar and can be expressed through the “standard” ones, but since we should keep a TBox unchangeable, we must take into account all relations and interactions between different constructs. Moreover, keeping distinction between objects and values that takes place in DL-Lite_{AS} drives us to consider the DL-Lite_{AS} more detailed.

The erasure algorithm presented in the second subsection is based on the update algorithm as well as for DL-Lite_{RS}.

3.2.1 Update in DL-Lite_{AS}

On the fig. 3.4-3.6 one can see the algorithm of the instance-level update in DL-Lite_A KB. The main idea of it is the same as in both DL-Lite_F and DL-Lite_R: given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a set of membership assertions \mathcal{F} , to obtain an ABox \mathcal{A}' such that $\text{Mod}(\langle \mathcal{T}, \mathcal{A}' \rangle) = \mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$. For this, we delete from \mathcal{A} all membership assertions that contradict \mathcal{F} and add all that assertions that are logically implied from the deleted ones and does not contradict \mathcal{F} . Note that if an assertion of the form $\exists Q.C(a)$ is in \mathcal{F} , there exists nothing in an original KB that contradict it, since

- neither direct assertion $\neg\exists Q.C(a)$ can appear in \mathcal{A} ,
- nor contradiction can be derived from \mathcal{T} and \mathcal{A} : $\exists Q.C$ cannot appear at the left side of a negative inclusion assertion; also in cannot occur at the right side with negation.

A contradiction can arise if we have $\neg\exists Q(a)$ in the original \mathcal{A} , but for this case we add to \mathcal{F} assertion $\exists Q(a)$ in the line 7, and then we search a possible contradiction with this new assertion (lines 13-17).

Saying above is also true for the following constructs: $\exists\delta_F(U_C)$, $\exists\delta_F(U_R)$, $\exists\delta_F(U_R)^-$, \top_C , \top_D , rdfDataTypes , $\delta_F(U_R)$, and $\delta_F(U_R)^-$.

Now, we consider the correctness of the algorithm. Firstly, we examine satisfiability of a KB obtained as the result of *ComputeUpdate_{AS}*.

Lemma 3.10. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable DL-Lite_{AS} KB, \mathcal{F} a finite set of ground DL-Lite_{AS} membership assertions such that $\text{Mod}(\mathcal{K}) \cap \text{Mod}(\mathcal{F}) \neq \emptyset$, and \mathcal{K}' the DL-Lite_{AS} KB such that $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{A}' = \text{ComputeUpdate}^{\text{AS}}$. Then, \mathcal{K}' is always satisfiable.*

Proof. The proof is similar to the one of the theorem 3.1, taking into account the note given above. \square

The next property, termination, is also proved like the analogous property of $\text{ComputeUpdate}_{\mathcal{RS}}$ algorithm.

Lemma 3.11 (Termination). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{RS}}$ knowledge base, \mathcal{F} a finite set of ground DL-Lite $_{\mathcal{RS}}$ membership assertions. Then the algorithm $\text{ComputeUpdate}_{\mathcal{RS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$ terminates, returning *ERROR* if $\text{Mod}(\mathcal{T} \cup \mathcal{F}) = \emptyset$, and an ABox \mathcal{A}' such that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is a DL-Lite $_{\mathcal{RS}}$ knowledge base, otherwise.*

Proof. The proof is similar to the one of the theorem 3.2. Just notice that we extended function *Saturate* to make it works also with domains, concept and role attributes. \square

Now, it is a turn of completeness and soundness. Take notice that new (in comparison with DL-Lite $_{\mathcal{R}}$) constructs, domains and attributes, are in fact not very new, but they can be expressed in DL-Lite $_{\mathcal{FR}}$ language. But since the expression requires changes in a TBox, so we have to consider each case separately.

Lemma 3.12 (Soundness). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{AS}}$ knowledge base, \mathcal{F} a finite set of ground DL-Lite $_{\mathcal{AS}}$ membership assertions such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and \mathcal{K}' the DL-Lite $_{\mathcal{AS}}$ knowledge base such that $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{A}' = \text{ComputeUpdate}_{\mathcal{AS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then, for every model $\mathcal{I}' \in \text{Mod}(\mathcal{K}')$, we have that:*

$$\exists \mathcal{I} \in \text{Mod}(\mathcal{K}) \text{ such that } \mathcal{I}' \in U^{\mathcal{I}}(\mathcal{I}, \mathcal{F}).$$

Proof. The structure of this proof is the same as one of theorem 3.3.

Let \mathcal{I}' be a model of \mathcal{K}' and not be a model of \mathcal{K} (otherwise the theorem is proved trivially). By construction, \mathcal{I}' is a model of \mathcal{T} , that means that there exists a subset \mathcal{F}' of \mathcal{A} such that for each $F \in \mathcal{F}'$, $\text{Mod}(\{F\}) \cap \text{Mod}(\mathcal{I}') = \emptyset$.

We build a model \mathcal{I} step by step as it is described above.

STEP 1 We set $\mathcal{I}_0 := \mathcal{I}'$.

STEP 2 We modify \mathcal{I}_0 as follows. For each $f \in \mathcal{F}'$ do:

1. if $f = C(a)$ then $a^{\mathcal{I}_i} \in C^{\mathcal{I}_i}$;
2. if $f = R(a, b)$ then $(a^{\mathcal{I}_i}, b^{\mathcal{I}_i}) \in R^{\mathcal{I}_i}$, $a^{\mathcal{I}_i} \in \exists R^{\mathcal{I}_i}$, and $b^{\mathcal{I}_i} \in \exists R^{\mathcal{I}_i}$. Moreover, if $(\text{funct } R) \in \mathcal{T}$, then for each $(a^{\mathcal{I}_{i-1}}, b^{\mathcal{I}_{i-1}}) \in R^{\mathcal{I}_{i-1}}$ such that $b' \neq b$ we set $(a^{\mathcal{I}_i}, b^{\mathcal{I}_i}) \notin R^{\mathcal{I}_i}$, and if there is no $a'' \neq a$ such that $(a''^{\mathcal{I}_{i-1}}, b^{\mathcal{I}_{i-1}}) \in R^{\mathcal{I}_{i-1}}$, then we set $b^{\mathcal{I}_i} \in \exists R^{-\mathcal{I}_i}$ ⁵;
3. if $f = F(d)$ then $d^{\mathcal{I}_i} \in F^{\mathcal{I}_i}$;

⁵ The same settings can be done for $f = U_C(a, v)$.

ALGORITHM *ComputeUpdate_{AS}($\mathcal{T}, \mathcal{A}, \mathcal{F}$)*

INPUT: finite set of ground membership assertions \mathcal{F} , satisfiable with *DL-Lite_{FS}*

KB $\langle \mathcal{T}, \mathcal{A} \rangle$

OUTPUT: an ABox \mathcal{A}' , or ERROR

```

[1]  if  $\langle \mathcal{T}, \mathcal{F} \rangle$  is not satisfiable then return ERROR
[2]  else
[3]    {for each  $f \in \mathcal{F}$  do
[4]      if  $f = R(a, b)$  then  $\mathcal{F} := \mathcal{F} \cup \{\exists R(a), \exists R^-(b)\}$ ;
[5]      if  $f = V_C(a, v)$  then  $\mathcal{F} := \mathcal{F} \cup \{\delta(V_C)(a), \rho(V_C)(v)\}$ ;
[6]      if  $f = V_R(a, b, v)$  then  $\mathcal{F} := \mathcal{F} \cup \{\delta(V_R)(a, b), \rho(V_R)(v),$ 
[6]                                      $\exists \delta(V_R)(a), \exists \delta(V_R)^-(b)\}$ ;
[7]      if  $f = \exists Q.C(a)$  then  $\mathcal{F} := \mathcal{F} \cup \{\exists Q(a), \}$ ;
[8]      if  $f = \delta_F(U_C)(a)$  then  $\mathcal{F} := \mathcal{F} \cup \{\delta(U_C)(a)\}$ ;
[9]      if  $f = \delta_F(U_R)(a, b)$  then  $\mathcal{F} := \mathcal{F} \cup \{\delta(U_R)(a, b)\}$ ;
[10]   }
[11]    $\mathcal{F}' := \emptyset$ 
[12]   for each  $f \in \mathcal{F}$  do
[13]     if  $f = C(a)$  and either  $C = B$  or  $C = \neg B$  then
[14]        $\{\mathcal{F}_C := \text{Saturate}(\neg C(a), \mathcal{T})$ 
[15]       for each  $f_C \in \mathcal{F}_C$  do
[16]         if  $\langle \mathcal{T}, \mathcal{A} \rangle \models f_C$  then  $\mathcal{F}' := \mathcal{F}' \cup \{f_C\}$ ;
[17]         if  $f_C = \exists R(a)$  then  $\mathcal{F}' := \mathcal{F}' \cup \{R(a, b) \mid R(a, b) \in \mathcal{A}\}$ ;
[18]         if  $f_C = \delta(U)(a)$  then  $\mathcal{F}' := \mathcal{F}' \cup \{U(a, v) \mid U(a, v) \in \mathcal{A}\}$ ;
[19]       }
[20]     }
[21]   or else
[22]     {if  $f = Q(a, b)$  and (funct  $Q$ )  $\in \mathcal{T}$  then
[23]       for each  $b' \neq b$  s.th.  $R(a, b') \in \mathcal{A}$  do  $\mathcal{F}' := \mathcal{F}' \cup \{R(a, b')\}$ ;
[24]       if  $f = R(a, b)$  and (funct  $R$ )  $\notin \mathcal{T}$  and either  $R = Q$  or  $R = \neg Q$  then
[25]          $\mathcal{F}_R = \text{Saturate}(\neg R(a, b), \mathcal{T})$ 
[26]         for each  $f_R \in \mathcal{F}_R$  do
[27]           if  $\langle \mathcal{T}, \mathcal{A} \rangle \models f_R$  then  $\mathcal{F}' := \mathcal{F}' \cup \{f_R\}$ 
[28]         }
[29]     }

```

Fig. 3.4: Algorithm *ComputeUpdate_{AS}*, part 1

```

[30]     if  $f = F(d)$  and either  $F = E$  or  $F = \neg E$  then
[31]        $\{\mathcal{F}_D := \text{Saturate}(\neg F(d), \mathcal{T})$ 
[32]       for each  $f_D \in \mathcal{F}_D$  do
[33]         if  $\langle \mathcal{T}, \mathcal{A} \rangle \models f_D$  then  $\mathcal{F}' := \mathcal{F}' \cup \{f_D\}$ 
[34]         if  $f_D = \rho(U)(d)$  then  $\mathcal{F}' := \mathcal{F}' \cup \{U(d, v) \mid U(d, v) \in \mathcal{A}\}$ 
[35]       }
[36]     }
[37]   or else
[38]     if  $f = U(a, v)$  and  $(\text{funct } U) \in \mathcal{T}$  then
[39]       for each  $v' \neq v$  s.th.  $U(a, v') \in \mathcal{A}$  do  $\mathcal{F}' := \mathcal{F}' \cup \{U(a, v')\}$ ;
[40]     if  $f = V(a, v)$  and  $(\text{funct } V) \notin \mathcal{T}$  then
[41]        $\mathcal{F}_V = \text{Saturate}(\neg V(a, v), \mathcal{T})$ 
[42]       for each  $f_V \in \mathcal{F}_V$  do
[43]         if  $\langle \mathcal{T}, \mathcal{A} \rangle \models f_V$  then  $\mathcal{F}' := \mathcal{F}' \cup \{f_V\}$ 
[44]       }
[45]    $\mathcal{A}' := \mathcal{A} \cup \mathcal{F}$ ;
[46]   for each  $f' \in \mathcal{F}'$  do
[47]     if  $f' = C(a)$  then
[48]        $\{\mathcal{A}' := \mathcal{A}' \setminus \{C(a)\}$ 
[49]       for each  $C \sqsubseteq C_1$  in  $cl(\mathcal{T})$  do
[50]         if  $(C_1(a) \notin \mathcal{F}')$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_1(a)\}$ 
[51]       if  $f' = \exists R(a)$  then
[52]         for each  $\exists R^- \sqsubseteq C_2$  in  $cl(\mathcal{T})$  do
[53]            $\mathcal{A}' := \mathcal{A}' \cup \{C_2(z_{\exists R(a)})\}$ , with  $z_{\exists R(a)}$  new variable
[54]       if  $f' = \delta(U)(a)$  then
[55]         for each  $\rho(U) \sqsubseteq F_1$  in  $cl(\mathcal{T})$  do
[56]            $\mathcal{A}' := \mathcal{A}' \cup \{C_2(z_{\delta(U)(a)})\}$ , with  $z_{\delta(U)(a)}$  new variable
[57]       }
[58]     or else
[59]     if  $f' = R(a, b)$  then
[60]        $\{\mathcal{A}' := \mathcal{A}' \setminus \{R(a, b), \exists R(a), \exists R^-(b)\}$ 
[61]       for each  $R \sqsubseteq R_1$  in  $cl(\mathcal{T})$  do
[62]         if  $R_1(a, b) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{R_1(a, b)\}$ 
[63]       for each  $\exists R \sqsubseteq C_3$  in  $cl(\mathcal{T})$  do
[64]         if  $C_3(a) \notin \mathcal{F}$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_3(a)\}$ 
[65]       for each  $\exists R^- \sqsubseteq C_4$  in  $cl(\mathcal{T})$  do
[66]         if  $C_4(b) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_4(b)\}$ 
[67]       }
[68]     or else
[69]     if  $f' = F(d)$  then

```

Fig. 3.5: Algorithm *ComputeUpdate_{AS}*, part 2

```

[70]    $\{\mathcal{A}' := \mathcal{A} \setminus \{F(d)\}$ 
[71]   for each  $F \sqsubseteq F_1$  in  $cl(\mathcal{T})$  do
[72]     if  $(F_1(d) \notin \mathcal{F}')$  then  $\mathcal{A}' := \mathcal{A}' \cup \{F_1(d)\}$ 
[73]   if  $f' = \rho(U_C)(d)$  then
[74]     for each  $\delta(U_C) \sqsubseteq C_5$  in  $cl(\mathcal{T})$  do
[75]        $\mathcal{A}' := \mathcal{A}' \cup \{C_5(z_{\rho(U_C)(a)})\}$ , with  $z_{\rho(U_C)(a)}$  new variable
[76]   if  $f' = \rho(U_R)(d)$  then
[77]     for each  $\delta(U_R) \sqsubseteq R_2$  in  $cl(\mathcal{T})$  do
[78]        $\mathcal{A}' := \mathcal{A}' \cup \{R_2(z_{\rho(U)(a)_1}, z_{\rho(U)(a)_2})\}$ , with  $z_{\rho(U_R)(a)_i}$  new variables
[79]   or else
[80]   if  $f' = V_C(a, v)$  then
[81]      $\{\mathcal{A}' := \mathcal{A}' \setminus \{V_C(a, v), \delta(V_C)(a), \rho(V_C)(v)\}$ 
[82]     for each  $V_C \sqsubseteq V_{C1}$  in  $cl(\mathcal{T})$  do
[83]       if  $V_{C1}(a, v) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{V_{C1}(a, v)\}$ 
[84]     for each  $\delta(V_C) \sqsubseteq C_6$  in  $cl(\mathcal{T})$  do
[85]       if  $C_6(a) \notin \mathcal{F}$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_6(a)\}$ 
[86]     for each  $\rho(V_C) \sqsubseteq F_2$  in  $cl(\mathcal{T})$  do
[87]       if  $F_2(v) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{F_2(v)\}$ 
[88]   or else
[89]   if  $f' = V_R(a, b, v)$  then
[90]      $\{\mathcal{A}' := \mathcal{A}' \setminus \{V_R(a, b, v), \delta(V_R)(a, b), \rho(V_R)(v), \exists\delta(V_R)(a), \exists\delta(V_R)^-(b)\}$ 
[91]     for each  $V_R \sqsubseteq V_{R1}$  in  $cl(\mathcal{T})$  do
[92]       if  $V_{R1}(a, b, v) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{V_{R1}(a, b, v)\}$ 
[93]     for each  $\delta(V_R) \sqsubseteq R_3$  in  $cl(\mathcal{T})$  do
[94]       if  $R_3(a, b) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{R_3(a, b)\}$ 
[95]     for each  $\rho(V_R) \sqsubseteq F_3$  in  $cl(\mathcal{T})$  do
[96]       if  $F_3(v) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{F_3(v)\}$ 
[97]     for each  $\exists\delta(V_R) \sqsubseteq C_7$  in  $cl(\mathcal{T})$  do
[98]       if  $C_7(a) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_7(a)\}$ 
[99]     for each  $\exists\delta(V_R)^- \sqsubseteq C_8$  in  $cl(\mathcal{T})$  do
[100]      if  $C_8(b) \notin \mathcal{F}'$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_8(b)\}$ 
[101]   }

```

Fig. 3.6: Algorithm *ComputeUpdate_{AS}*, part 3

ALGORITHM *Saturate*(f, \mathcal{T})
INPUT: a membership assertion f , TBox \mathcal{T}
OUTPUT: set \mathcal{P} of ground concept membership assertions
 $\mathcal{P} = \{f\}$
repeat
 $\mathcal{P}' := \mathcal{P}$
for each $f \in \mathcal{P}'$
case $f =$
 $C(a)$ **then**
if $C' \sqsubseteq C \in cl(\mathcal{T})$ **then** $\mathcal{P} := \mathcal{P} \cup \{C'(a)\}$
 $R(a, b)$ **then**
if $R' \sqsubseteq R \in cl(\mathcal{T})$ **then** $\mathcal{P} := \mathcal{P} \cup \{R'(a, b)\}$
 $F(d)$ **then**
if $F' \sqsubseteq F \in cl(\mathcal{T})$ **then** $\mathcal{P} := \mathcal{P} \cup \{F'(d)\}$
 $U_C(a, v)$ **then**
if $U'_C \sqsubseteq U_C \in cl(\mathcal{T})$ **then** $\mathcal{P} := \mathcal{P} \cup \{U'_C(a, v)\}$
 $U_R(a, b, v)$ **then**
if $U'_R \sqsubseteq U_R \in cl(\mathcal{T})$ **then** $\mathcal{P} := \mathcal{P} \cup \{U'_R(a, b, v)\}$
until $\mathcal{P} = \mathcal{P}'$

Fig. 3.7: Algorithm *Saturate*

4. if $f = U_R(a, b, v)$ then $(a^{\mathcal{I}_i}, b^{\mathcal{I}_i}, val(v)) \in U_R^{\mathcal{I}_i}$, $(a^{\mathcal{I}_i}, b^{\mathcal{I}_i}) \in \delta(U_R)^{\mathcal{I}_i}$, $val(v) \in \rho(U_R)^{\mathcal{I}_i}$, $a^{\mathcal{I}_i} \in \exists\delta(U_R)^{\mathcal{I}_i}$, and $b^{\mathcal{I}_i} \in \exists\delta(U_R)^{-\mathcal{I}_i}$. Moreover, if $(funct U_R) \in \mathcal{T}$, then for each $(a^{\mathcal{I}_{i-1}}, b^{\mathcal{I}_{i-1}}, v^{\mathcal{I}_{i-1}}) \in U_R^{\mathcal{I}_{i-1}}$ such that $v' \neq v$ we set $(a^{\mathcal{I}_i}, b^{\mathcal{I}_i}, v^{\mathcal{I}_i}) \notin U_R^{\mathcal{I}_i}$, and if there is no $(a'', b'') \neq a, b$ such that $(a''^{\mathcal{I}_{i-1}}, b^{\mathcal{I}_{i-1}}, v^{\mathcal{I}_{i-1}}) \in U_R^{\mathcal{I}_{i-1}}$, then we set $v^{\mathcal{I}_i} \in \exists U_R^{-\mathcal{I}_i}$.

STEP 3 We set $\mathcal{I}_i := \mathcal{I}_{i-1}$ and $i := 1$.

Repeat the following rules:

1. If $a \in B^{\mathcal{I}_{i-1}}$ (resp. $v \in E^{\mathcal{I}_i}$), $B \sqsubseteq C \in \mathcal{T}$ (resp. $E \sqsubseteq F \in \mathcal{T}$) and $a \notin C^{\mathcal{I}_{i-1}}$ (resp. $v \notin F^{\mathcal{I}_{i-1}}$), then set $a \in C^{\mathcal{I}_i}$ (resp. $v \in F^{\mathcal{I}_i}$).
2. If $a \in \exists Q^{\mathcal{I}_{i-1}}$ and there is no $b \in \Delta_O^{\mathcal{I}_{i-1}}$ such that $(a, b) \in Q^{\mathcal{I}_{i-1}}$, then add $(a, b') \in Q^{\mathcal{I}_i}$ and $b' \in \exists Q^{-\mathcal{I}_i}$, where b' is a new element of $\Delta_O^{\mathcal{I}_i}$ such that if $(funct Q) \in \mathcal{T}$ then there exists no a'' such that $(a'', b') \in Q^{\mathcal{I}_{i-1}}$ ⁶.
3. If $a \in \exists Q.C^{\mathcal{I}_{i-1}}$ and there is no $b \in \Delta_O^{\mathcal{I}_{i-1}}$ such that $(a, b) \in Q^{\mathcal{I}_{i-1}}$ and $b \in C^{\mathcal{I}_{i-1}}$, then add $(a, b') \in Q^{\mathcal{I}_i}$, $b' \in \exists Q^{-\mathcal{I}_i}$ and $b' \in C^{\mathcal{I}_i}$, where b' is a new element⁷.

⁶ The analogous constructions can be done also for $\delta(U_C)$, $\rho(U_C)$ and $\rho(U_R)$. We do not adduce them in order not to overload the text of the proof.

⁷ The analogous constructions can be done also for $\delta_F(U_C)$, $\exists\delta_F(U_R)$ and $\exists\delta_F(U_R)^-$. We do not adduce them in order not to overload the text of the proof.

4. If $(a, b) \in Q^{\mathcal{I}_{i-1}}$, $Q \sqsubseteq R \in \mathcal{T}$ and $(a, b) \notin R^{\mathcal{I}_{i-1}}$, then set $(a, b) \in R^{\mathcal{I}_i}$.
5. If $(a, b) \in \delta(U_R)^{\mathcal{I}_{i-1}}$ and there is no $v \in \Delta_V^{\mathcal{I}_{i-1}}$ such that $(a, b, v) \in U_R^{\mathcal{I}_{i-1}}$, then add $(a, b, v') \in U_R^{\mathcal{I}_i}$ and $v' \in \rho(U_R)^{-\mathcal{I}_i}$, where v' is a new element of $\Delta_V^{\mathcal{I}_i}$ such that if $(\text{funct } U_R) \in \mathcal{T}$ then there exists no (a'', b'') such that $(a'', b'', v') \in U_R^{\mathcal{I}_{i-1}}$ ⁸.
6. If $(a, b) \in \delta_F(U_R)^{\mathcal{I}_{i-1}}$ and there is no $v \in \Delta_V^{\mathcal{I}_{i-1}}$ such that $(a, b, v) \in U_R^{\mathcal{I}_{i-1}}$ and $v \in F^{\mathcal{I}_{i-1}}$, then add $(a, b, v') \in U_R^{\mathcal{I}_i}$, $v' \in \rho(U_R)^{\mathcal{I}_i}$ and $v' \in F^{\mathcal{I}_i}$, where v' is a new element⁹.
7. If $a \in \neg\exists Q^{\mathcal{I}_{i-1}}$, then for each $(a, b) \in Q^{\mathcal{I}_{i-1}}$ we set $(a, b) \notin Q^{\mathcal{I}_i}$, and if there exists no $a' \neq a$ such that $(a', b) \in Q^{\mathcal{I}_{i-1}}$, then we set $b \notin \exists Q^{-\mathcal{I}_i}$ ¹⁰.
8. If $(a, b) \in \neg\delta(U_R)^{\mathcal{I}_{i-1}}$, then for each $(a, b, v) \in U_R^{\mathcal{I}_{i-1}}$ we set $(a, b, v) \notin U_R^{\mathcal{I}_i}$, and if there exists no $(a', b') \neq (a, b)$ such that $(a', b', v) \in Q^{\mathcal{I}_{i-1}}$, then we set $v \notin \rho(U_R)^{-\mathcal{I}_i}$ ¹¹.
9. $i := i + 1$.

Until $\mathcal{I}_i = \mathcal{I}_{i-1}$

We apply STEP 3 until $\mathcal{I}_{n+1} = \mathcal{I}_n$, and \mathcal{I}_n will be the required interpretation \mathcal{I} . As well as in the lemma 3.3, \mathcal{I} is a model of \mathcal{T} (\mathcal{I}_0 is a model of \mathcal{T} and \mathcal{I} satisfies all membership assertions in \mathcal{A}).

Now, let us show that $\mathcal{I}' \in U^{\mathcal{I}}(\mathcal{I}, \mathcal{F})$. It is true, by the definition, if and only if there is no interpretation \mathcal{I}'' such that $\mathcal{I}'' \in \text{Mod}(\mathcal{T} \cup \mathcal{F})$ and $\mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}'$.

Let us suppose that there exists such \mathcal{I}'' and $\mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}'$. It means that one of the following cases occurs:

1. There is a such that $a \in L^{\mathcal{I}}$ and $a \in L^{\mathcal{I}''}$, but $a \notin L^{\mathcal{I}'}$, where L is either A or D .
2. There is a such that $a \notin L^{\mathcal{I}}$ and $a \notin L^{\mathcal{I}''}$, but $a \in L^{\mathcal{I}'}$, where L is either A or D .
3. There is (a, b) such that $(a, b) \in M^{\mathcal{I}}$ and $(a, b) \in M^{\mathcal{I}''}$, but $(a, b) \notin M^{\mathcal{I}'}$, where M is either Q or U_C .
4. There is (a, b) such that $(a, b) \notin M^{\mathcal{I}}$ and $(a, b) \notin M^{\mathcal{I}''}$, but $(a, b) \in M^{\mathcal{I}'}$, where M is either Q or U_C .
5. There is (a, b, v) such that $(a, b, v) \in U_R^{\mathcal{I}}$ and $(a, b, v) \in U_R^{\mathcal{I}''}$, but $(a, b, v) \notin U_R^{\mathcal{I}'}$.

⁸ The same construct is for $\delta(U_R)^-$.

⁹ The same construct is for $\delta_F(U_R)^-$.

¹⁰ The analogous constructions can be done also for $\neg\delta(U_C)$, $\neg\rho(U_C)$ and $\neg\rho(U_R)$. We do not adduce them in order not to overload the text of the proof.

¹¹ The same construct is for $\neg\delta(U_R)^-$.

6. There is (a, b, v) such that $(a, b, v) \notin U_R^{\mathcal{I}}$ and $(a, b, v) \notin U_R^{\mathcal{I}''}$, but $(a, b, v) \in U_R^{\mathcal{I}'}$.

Cases 1-2 are considered in the proof of the lemma 3.3. In cases 3-4, in addition to the described contradictions, the following ones arise.

- 3 Let us consider the case, when \mathcal{I}' does not satisfy an assertion $f' = Q(a, b) \in \mathcal{F}'$. The new possibility is that f' contradicts \mathcal{F} because of a functionality assertion, e.g. $(\text{funct } Q) \in \mathcal{T}$, for some $Q(a, b') \in \mathcal{F}$, $b \neq b'$. Then $(a, b) \in Q^{\mathcal{I}''}$ would imply that $(a, b') \notin Q^{\mathcal{I}''}$ which would contradict that \mathcal{I}'' is a model of \mathcal{F} .
- 4 The new possible contradiction here is that $f' = Q(a, b') \in \mathcal{F}'$ contradicts $f \in \mathcal{F}$ for some $b' \neq b$ where $f = Q(a, b)$ and $(\text{funct } Q) \in \mathcal{T}$. So, if $(a, b') \in Q^{\mathcal{I}'}$, we must set that $(a, b) \notin Q^{\mathcal{I}}$. But if $(a, b) \notin Q^{\mathcal{I}}$, then we have a contradiction, since \mathcal{I}'' is not a model if \mathcal{F} .

Cases 5-6 are similar to the cases 3-4.

Therefore we have that there is no interpretation \mathcal{I}'' such that $\mathcal{I}'' \in \text{Mod}(\mathcal{T} \cup \mathcal{F})$ and $\mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}'$ that yields $\mathcal{I}' \in U^{\mathcal{I}}(\mathcal{I}, \mathcal{F})$. □

Now, we consider completeness of the algorithm.

Lemma 3.13 (Completeness). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{AS} knowledge base, \mathcal{F} a finite set of ground DL-Lite_{AS} membership assertions such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and \mathcal{K}' the DL-Lite_{AS} knowledge base such that $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{A}' = \text{ComputeUpdate}_{AS}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then, for every model $\mathcal{I} \in \text{Mod}(\mathcal{K})$, we have that:*

$$U^{\mathcal{I}}(\mathcal{I}, \mathcal{F}) \subseteq \text{Mod}(\mathcal{K}').$$

Proof. The proof is absolutely analogous to the one of the lemma 3.13; the changes we need will be in the procedure of building $\check{\mathcal{I}}$:

STEP 1 We set $\check{\mathcal{I}} = \mathcal{I}''$.

STEP 2 We modify $\check{\mathcal{I}}$ as follows:

1. If $f' = R(a)$, then we set $a \in R^{\check{\mathcal{I}}}$, where R is either C or F .
2. If $f' = R(z)$, where z is a variable, then we find a constant $b \in R^{\mathcal{I}}$, and we set $b \in R^{\check{\mathcal{I}}}$, where R is either C or F .
3. If $f' = S(a, b)$, then we set $(a, b) \in S^{\check{\mathcal{I}}}$, $a \in \exists S^{\check{\mathcal{I}}}$, $b \in \exists S^{-\check{\mathcal{I}}}$, where S is either R or V_C . In the latter case, $\exists S$ designates $\delta(U_C)$, $\exists S^-$ means $\rho(U_C)$.
4. If $f' = V_R(a, b, v)$, then we set $(a, b, v) \in V_R^{\check{\mathcal{I}}}$, $(a, b) \in \delta(V_R)^{\check{\mathcal{I}}}$, $v \in \rho(V_R)^{\check{\mathcal{I}}}$, $a \in \exists \delta(V_R)^{\check{\mathcal{I}}}$, $b \in \exists \delta(V_R)^{-\check{\mathcal{I}}}$.

STEP 3 We apply recursively the following rules in order to make $\check{\mathcal{I}}$ satisfy \mathcal{T} .

1. If $a \in B^{\check{\mathcal{I}}}$, $B \sqsubseteq C \in \mathcal{T}$, and $a \notin C^{\mathcal{I}''}$, then we set $a \in C^{\check{\mathcal{I}}}$.
2. If $a \in \exists Q^{\check{\mathcal{I}}}$ and there exists no individual $b \in \Delta$ such that $(a, b) \in Q^{\mathcal{I}''}$, then for each $(a, b') \in Q^{\check{\mathcal{I}}}$, set $(a, b') \in Q^{\mathcal{I}''}$.
3. If $a \in \exists Q^{-\check{\mathcal{I}}}$ and there exists no individual $b \in \Delta$ such that $(b, a) \in Q^{\mathcal{I}''}$, then for each $(b', a) \in Q^{\check{\mathcal{I}}}$, set $(b', a) \in Q^{\mathcal{I}''}$.
4. If $(a, b) \in Q^{\check{\mathcal{I}}}$, $Q \sqsubseteq R \in \mathcal{T}$, and $(a, b) \notin R^{\mathcal{I}''}$, then we set $(a, b) \in R^{\check{\mathcal{I}}}$.

It is obvious that the interpretation $\check{\mathcal{I}}$ is a model of \mathcal{T} (it is provided by STEP 3). Also $\check{\mathcal{I}}$ satisfies F' by construction (STEP 2). Moreover, $\check{\mathcal{I}}$ satisfies all other membership assertions in \mathcal{A}' (since nothing was deleted). The latter affirmation yields that $\check{\mathcal{I}}$ is a model of \mathcal{K}' .

Finally, let us show that $\mathcal{I} \ominus \check{\mathcal{I}} \subset \mathcal{I} \ominus \mathcal{I}''$. We know that membership assertion F' logically implied by \mathcal{K} , so \mathcal{I} and $\check{\mathcal{I}}$ agree on interpreting this part, but not \mathcal{I}'' . The same situation with setting in STEP 3. The rest is interpreted by both \mathcal{I}'' and $\check{\mathcal{I}}$ in the same way. Thus, we have that $\mathcal{I} \ominus \check{\mathcal{I}} \subset \mathcal{I} \ominus \mathcal{I}''$. Contradiction with minimality of \mathcal{I}'' (see Definition 3.3).

Thereby, $\mathcal{I}'' \in \text{Mod}(\mathcal{K}')$. □

Since the property of expressibility of the algorithm $\text{ComputeUpdate}_{AS}$ can be proved in the same way as one of $\text{ComputeUpdate}_{RS}$, we can formulate the following correctness theorem, namely:

Theorem 3.14. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{AS} knowledge base, \mathcal{F} a finite set of ground DL-Lite_{AS} membership assertions. Then, the algorithm $\text{ComputeUpdate}_{AS}$ returns *ERROR* if $\text{Mod}(\mathcal{T}) \cap \text{Mod}(\mathcal{F}) = \emptyset$, or returns \mathcal{A}' , otherwise, such that $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F} \equiv \text{Mod}(\mathcal{K}')$, where $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$.*

Proof. The theorem is implied by lemmas 3.10, 3.11, 3.12, and 3.13. □

Finally, let us turn to the computational complexity of the algorithm.

Theorem 3.15. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{AS} knowledge base, \mathcal{F} a finite set of ground DL-Lite_{AS} membership assertions such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and $\mathcal{A}' = \text{ComputeUpdate}_{AS}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then:*

- *the size of \mathcal{A}' is polynomially bounded by the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$;*
- *computing \mathcal{A}' can be done in polynomial time in the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$.*

Proof. The proof is similar to the proof of analogous theorem for $\text{ComputeUpdate}_{RS}$. □

ALGORITHM $ComputeErasure_{AS}(\mathcal{T}, \mathcal{A}, \mathcal{F})$

INPUT: finite set of ground membership assertions \mathcal{F} such that $\neg\mathcal{F}$ satisfiable with $DL\text{-}Lite_{AS}$ KB $\langle\mathcal{T}, \mathcal{A}\rangle$

OUTPUT: an ABox \mathcal{A}' , or ERROR

```

[1]  if  $\langle\mathcal{T}, \neg\mathcal{F}\rangle$  is not satisfiable then return ERROR
[2]  else {
[3]     $\mathcal{A}'' = ComputeUpdate_{AS}(\mathcal{T}, \mathcal{A}, \mathcal{F})$ 
[4]     $\mathcal{A}' := \mathcal{A} \cap \mathcal{A}''$ 
[5]  }
```

Fig. 3.8: $ComputeErasure_{AS}$

3.2.2 Erasure in $DL\text{-}Lite_{AS}$

The algorithm $ComputeErasure_{AS}$ takes as an input a satisfiable $DL\text{-}Lite_{AS}$ $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ and returns a $DL\text{-}Lite_{AS}$ ABox \mathcal{A}' , such that $Mod(\langle\mathcal{T}, \mathcal{A}'\rangle) = \mathcal{K} \bullet_{\mathcal{T}} \mathcal{F}$. The idea of the algorithm is the same as in the case of erasure of $DL\text{-}Lite_{RS}$ KB.

Both completeness and soundness of the algorithm can be proved as it is done in the theorem 3.9.

Lemma 3.16 (Soundness and completeness). *Let $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ be a $DL\text{-}Lite_{RS}$ knowledge base, \mathcal{F} a finite set of ground $DL\text{-}Lite_{RS}$ membership assertions such that $Mod(\mathcal{T} \cup \neg\mathcal{F}) \neq \emptyset$, and \mathcal{K}' the $DL\text{-}Lite_{RS}$ knowledge base such that $\mathcal{K}' = \langle\mathcal{T}, \mathcal{A}'\rangle$, where $\mathcal{A}' = ComputeErasure_{RS}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then:*

$$Mod(\mathcal{K}') = \mathcal{K} \bullet_{\mathcal{T}} \mathcal{F}.$$

The rest of algorithm properties (such as termination, satisfiability, etc.) comes from the analogous properties of $ComputeUpdate_{AS}$.

Theorem 3.17. *Let $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ be a $DL\text{-}Lite_{AS}$ knowledge base, \mathcal{F} a finite set of ground $DL\text{-}Lite_{AS}$ membership assertions. Then, the algorithm $ComputeErasure_{AS}$ returns ERROR if $Mod(\mathcal{T}) \cap Mod(\neg\mathcal{F}) = \emptyset$, or returns a $DL\text{-}Lite_{AS}$ ABox \mathcal{A}' , otherwise, such that $\mathcal{K} \bullet_{\mathcal{T}} \mathcal{F} \equiv Mod(\mathcal{K}')$, where $\mathcal{K}' = \langle\mathcal{T}, \mathcal{A}'\rangle$.*

3.3 Instance-level update and erasure in $DL\text{-}Lite_A$

Unfortunately, as it known from the results in the literature [LLMW06], the result of an update is not generally expressible in the same language as the original KB. What we can do in such a case is to use approximate update. Thus, in this section we consider the notion of approximate update and erasure for $DL\text{-}Lite_A$.

3.3.1 Update in $DL\text{-}Lite_A$

From the example 3.1 we see that $DL\text{-}Lite_A$ is inexpressible. That is why we introduce the notion of approximation in Description Logics [DGLPR07, DGLPR09].

ALGORITHM $ComputeUpdate_{\mathcal{A}}^{app}(\mathcal{T}, \mathcal{A}, \mathcal{F})$

INPUT: finite set of ground membership assertions \mathcal{F} , satisfiable with DL-Lite _{$\mathcal{F}\mathcal{S}$}

KB $\langle \mathcal{T}, \mathcal{A} \rangle$

OUTPUT: an ABox \mathcal{A}^{app} , or ERROR

- [1] **if** $\langle \mathcal{T}, \mathcal{F} \rangle$ is not satisfiable **then return** ERROR
- [2] **else**
- [3] $\mathcal{A}^{app} := ComputeUpdate_{\mathcal{AS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$;
- [4] **for each** $f \in \mathcal{A}^{app}$ **do**
- [5] **if** f is not DL-Lite _{\mathcal{A}} membership assertion **then** $\mathcal{A}^{app} := \mathcal{A}^{app} \setminus \{f\}$;
- [6] **return** \mathcal{A}^{app} ;

Fig. 3.9: $ComputeUpdate_{\mathcal{A}}^{app}$

The notion of approximation is based on fixing a priori both the language \mathcal{L} and the TBox \mathcal{T} .

Definition 3.8 (Sound $(\mathcal{L}, \mathcal{T})$ -approximation). Let \mathcal{T} be a TBox in a DL \mathcal{L} , and \mathcal{M} a set of models such that $\mathcal{M} \subseteq Mod(\mathcal{T})$. We say that a DL KB \mathcal{K} is a *sound $(\mathcal{L}, \mathcal{T})$ -approximation* of \mathcal{M} in \mathcal{L} if

1. \mathcal{K} is in \mathcal{L} ;
2. \mathcal{K} is of the form $\langle \mathcal{T}, \mathcal{A} \rangle$;
3. $\mathcal{M} \subseteq Mod(\mathcal{K})$.

In other words, \mathcal{M} is a subset of models of \mathcal{T} expressed in \mathcal{L} and its sound $(\mathcal{L}, \mathcal{T})$ -approximation is KB \mathcal{K} with the same TBox \mathcal{T} , \mathcal{K} still expressed in \mathcal{L} and the set of its models includes \mathcal{M} . It is obvious that there are a lot of sound $(\mathcal{L}, \mathcal{T})$ -approximations of \mathcal{M} . Intuitively, some of them are better in the sense of closeness to \mathcal{M} , that leads us to the following definition.

Definition 3.9 (Maximal $(\mathcal{L}, \mathcal{T})$ -approximation). Let \mathcal{T} be a TBox in a DL \mathcal{L} , and \mathcal{M} a set of models such that $\mathcal{M} \subseteq Mod(\mathcal{T})$. We say that a DL KB \mathcal{K} is a *maximal $(\mathcal{L}, \mathcal{T})$ -approximation* of \mathcal{M} in \mathcal{L} if

1. \mathcal{K} is a sound $(\mathcal{L}, \mathcal{T})$ -approximation of \mathcal{M} ;
2. there exists no \mathcal{K}' such that \mathcal{K}' is a sound $(\mathcal{L}, \mathcal{T})$ -approximation of \mathcal{M} and $Mod(\mathcal{K}') \subset Mod(\mathcal{K})$.

In other words, maximal $(\mathcal{L}, \mathcal{T})$ -approximation of \mathcal{M} is a sound $(\mathcal{L}, \mathcal{T})$ -approximation \mathcal{K} of \mathcal{M} such that its semantics is so close to \mathcal{M} as possible.

In [DGLPR07] it was shown that when a maximal $(\mathcal{L}, \mathcal{T})$ -approximation exists, it is unique up to logical equivalence. Unfortunately, there are DLs \mathcal{L} , KBs $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and set of membership assertions \mathcal{F} such that maximal $(\mathcal{L}, \mathcal{T})$ -approximations of $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$ do not exist [DGLPR09].

So, given a notion of maximal $(\mathcal{L}, \mathcal{T})$ -approximation, we can turn to an approximate update in DL-Lite $_{\mathcal{A}}$. Our goal is to find an $(\mathcal{L}, \mathcal{T})$ -update of a DL-Lite $_{\mathcal{A}}$ KB.

Definition 3.10 ($(\mathcal{L}, \mathcal{T})$ -update). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\mathcal{K}^{app} = \langle \mathcal{T}, \mathcal{A}^{app} \rangle$ be two KBs in DL \mathcal{L} , and \mathcal{F} a finite set of membership assertions expressed in \mathcal{L} such that $Mod(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$. We say that \mathcal{K}^{app} is a $(\mathcal{L}, \mathcal{T})$ -update of \mathcal{K} with \mathcal{F} if \mathcal{K}^{app} is a maximal $(\mathcal{L}, \mathcal{T})$ -approximation of $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$.

As it was shown in [DGLPR09], if $(\mathcal{L}, \mathcal{T})$ -update of \mathcal{K} with \mathcal{T} exists, it is unique up to logical equivalence. Moreover, it was also proved that $(\mathcal{L}, \mathcal{T})$ -update captures exactly the logical implications of the membership assertions of the “exact” update (i.e. $\mathcal{K}^{app} \models \alpha$ iff $\mathcal{K} \models \alpha$, where α is an ABox assertion in \mathcal{L}).

Despite that there are cases when $(\mathcal{L}, \mathcal{T})$ -update does not exist, it can be built for DL-Lite $_{\mathcal{A}}$, that will be demonstrated below. Let us take a look at the algorithm $ComputeUpdate_{\mathcal{A}}^{app}$.

The algorithm on fig. 3.9 takes a satisfiable DL-Lite $_{\mathcal{A}}$ KB $\langle \mathcal{T}, \mathcal{A} \rangle$ and returns a DL-Lite $_{\mathcal{A}}$ ABox \mathcal{A}^{app} : since each DL-Lite $_{\mathcal{A}}$ KB is also a DL-Lite $_{\mathcal{AS}}$ KB, we can run $ComputeUpdate_{\mathcal{AS}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$ that returns a DL-Lite $_{\mathcal{AS}}$ (in general) ABox. Then it deletes all the membership assertions from it that are not DL-Lite $_{\mathcal{A}}$ membership assertions, obtaining an output ABox \mathcal{A}^{app} . Though this algorithm is not intuitively obvious, it is correct and its correctness is based on the following lemma.

Lemma 3.18. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable DL-Lite $_{\mathcal{AS}}$ KB, and f a DL-Lite $_{\mathcal{A}}$ membership assertion. If $\mathcal{K} \models f$, then there exists a DL-Lite $_{\mathcal{A}}$ membership assertion $f' \in \mathcal{A}$ such that $\langle \mathcal{T}, \{f'\} \rangle \models f$.*

Proof. To prove this theorem, we need to recall the notion of chase of a KB. Despite that it was defined for DL-Lite $_{\mathcal{F}}$ and DL-Lite $_{\mathcal{R}}$ KBs, it can be easily extended for the case of DL-Lite $_{\mathcal{A}}$ KBs. Just note that domains and concept attributes are essentially concepts and roles correspondingly, $\delta(U_C)$ and $\rho(U_C)$ act as $\exists Q$. A role attribute U_R is like a role adjusted for that $\delta(U_R)$ is a role, not a concept.

In [CDGL⁺07] it has been shown that $chase(\mathcal{K})$ identifies a canonical model of \mathcal{K} and that for every membership assertion f , $\mathcal{K} \models f$ iff $f \in chase(\mathcal{K})$.

So, suppose that $\mathcal{K} \models f$. By the property mentioned above, $f \in chase(\mathcal{K})$. Here we have to possibilities:

1. $f \in \mathcal{A}$ and the theorem is proved;
2. $f \notin \mathcal{A}$; by the inductive definition of $chase(\mathcal{K})$, it follows that there is a sequence of membership assertions f_1, f_2, \dots, f_n such that $f_1 \in \mathcal{A}$, $f_n = f$, and for each $i = 1, \dots, n-1$ f_{i+1} is obtained from f_i by applying one of the chase rules. Thus, taking $f' = f_1$, the theorem is proved.

□

Now, we are ready to prove correctness of $ComputeUpdate_{\mathcal{A}}^{app}$.

Theorem 3.19. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable DL-Lite_A knowledge base, \mathcal{F} a finite set of ground DL-Lite_A membership assertions. Then, the algorithm *ComputeUpdate_A^{app}* returns *ERROR* if $\text{Mod}(\mathcal{T}) \cap \text{Mod}(\mathcal{F}) = \emptyset$, or returns DL-Lite_A ABox \mathcal{A}^{app} , otherwise, such that \mathcal{K}^{app} is a (DL-Lite_A, \mathcal{T})-update of \mathcal{K} with \mathcal{F} , where $\mathcal{K}^{\text{app}} = \langle \mathcal{T}, \mathcal{A}^{\text{app}} \rangle$.*

Proof. Satisfiability of \mathcal{K}^{app} and termination of *ComputeUpdate_A^{app}* come directly from analogous properties of *ComputeUpdate_{AS}*.

We split the proof into two parts. In the first one we prove that \mathcal{K}^{app} is a sound (DL-Lite_A, \mathcal{T})-approximation of $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$. In the second part we show maximality of approximation.

- Let $\mathcal{A}' = \text{ComputeUpdate}_{AS}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. By construction, $\mathcal{A}^{\text{app}} \subseteq \mathcal{A}'$ that yields $\text{Mod}(\mathcal{A}') \subseteq \text{Mod}(\mathcal{A}^{\text{app}})$, and, therefore, $\text{Mod}(\langle \mathcal{T}, \mathcal{A}' \rangle) \subseteq \text{Mod}(\langle \mathcal{T}, \mathcal{A}^{\text{app}} \rangle)$. By theorem 3.14, we have that $\text{Mod}(\langle \mathcal{T}, \mathcal{A}' \rangle) = \mathcal{K} \circ_{\mathcal{T}} \mathcal{F} \subseteq \text{Mod}(\langle \mathcal{T}, \mathcal{A}^{\text{app}} \rangle)$. Thus, by definition, \mathcal{K}^{app} is a sound (DL-Lite_A, \mathcal{T})-approximation of $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$.
- Let us assume that there is a KB $\mathcal{K}'' = \langle \mathcal{T}, \mathcal{A}'' \rangle$ that is a sound (DL-Lite_A, \mathcal{T})-approximation of $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$, and $\text{Mod}(\mathcal{K}'') \subset \text{Mod}(\mathcal{K}^{\text{app}})$. Then, $\text{Mod}(\mathcal{K}'' \cup \mathcal{K}^{\text{app}}) = \text{Mod}(\mathcal{K}'') \cap \text{Mod}(\mathcal{K}^{\text{app}}) = \text{Mod}(\mathcal{K}'')$, that implies that $\mathcal{K}^{\text{app}} \subset \mathcal{K}''$, and there exists a DL-Lite_A membership assertion $f \in \mathcal{A}''$ such that $\mathcal{K} \circ_{\mathcal{T}} \mathcal{F} \models f$ and $\mathcal{K}^{\text{app}} \not\models f$. By theorem 3.14, we have that $\langle \mathcal{T}, \mathcal{A}' \rangle \models f$, and by lemma 3.18 there must exist a DL-Lite_A membership assertion $f' \in \mathcal{A}'$ such that $\langle \mathcal{T}, f' \rangle \models f$. By construction, $f' \in \mathcal{A}^{\text{app}}$, and we get a contradiction since $\mathcal{K}^{\text{app}} \models f$. Thus, \mathcal{K}^{app} is a (DL-Lite_A, \mathcal{T})-update of \mathcal{K} with \mathcal{F} .

□

Since *ComputeUpdate_{AS}* runs in polynomial time, it follows immediately that *ComputeUpdate_A^{app}* runs in polynomial time as well.

Theorem 3.20. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_A knowledge base, \mathcal{F} a finite set of ground DL-Lite_{AS} membership assertions such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and $\mathcal{A}^{\text{app}} = \text{ComputeUpdate}_{A}^{\text{app}}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then:*

- *the size of \mathcal{A}^{app} is polynomially bounded by the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$;*
- *computing \mathcal{A}^{app} can be done in polynomial time in the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$.*

3.3.2 Erasure in DL-Lite_A

In the same way as we have done it in the case of update, we now introduce the notion of maximal approximation of instance-level erasure in a DL \mathcal{L} .

Definition 3.11 ((\mathcal{L}, \mathcal{T})-erasure). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\mathcal{K}^{\text{app}} = \langle \mathcal{T}, \mathcal{A}^{\text{app}} \rangle$ be two KBs in DL \mathcal{L} , and \mathcal{F} a finite set of membership assertions expressed in \mathcal{L} such that $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$. We say that \mathcal{K}^{app} is a (\mathcal{L}, \mathcal{T})-erasure of \mathcal{K} with \mathcal{F} if \mathcal{K}^{app} is a maximal (\mathcal{L}, \mathcal{T})-approximation of $\mathcal{K} \bullet_{\mathcal{T}} \mathcal{F}$.

ALGORITHM $ComputeErasure_{\mathcal{A}}^{app}(\mathcal{T}, \mathcal{A}, \mathcal{F})$
INPUT: finite set of ground membership assertions \mathcal{F} , satisfiable with $DL-Lite_{\mathcal{F}\mathcal{S}}$
KB $\langle \mathcal{T}, \mathcal{A} \rangle$
OUTPUT: an ABox \mathcal{A}^{app} , or ERROR
[1] **if** $\langle \mathcal{T}, \neg \mathcal{F} \rangle$ is not satisfiable **then return** ERROR
[2] **else**
[3] $\mathcal{A}^{app} := ComputeErasure_{\mathcal{AS}}(\mathcal{T}, \mathcal{A}, \mathcal{F});$
[4] **for each** $f \in \mathcal{A}^{app}$ **do**
[5] **if** f is not $DL-Lite_{\mathcal{A}}$ membership assertion **then** $\mathcal{A}^{app} := \mathcal{A}^{app} \setminus \{f\};$
[6] **return** $\mathcal{A}^{app};$

Fig. 3.10: $ComputeErasure_{\mathcal{A}}^{app}$

This algorithm works as $ComputeUpdate_{\mathcal{A}}^{app}$ does: it takes original ABox \mathcal{A} , computes $ComputeErasure_{\mathcal{AS}}$, and deletes from it all the membership assertions which are not $DL-Lite_{\mathcal{A}}$ ones.

The correctness of the algorithm can be proved in a similar way with $ComputeUpdate_{\mathcal{A}}^{app}$.

Theorem 3.21. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable $DL-Lite_{\mathcal{A}}$ knowledge base, \mathcal{F} a finite set of ground $DL-Lite_{\mathcal{A}}$ membership assertions. Then, the algorithm $ComputeErasure_{\mathcal{A}}^{app}$ returns ERROR if $Mod(\mathcal{T}) \cap Mod(\mathcal{F}) = \emptyset$, or returns $DL-Lite_{\mathcal{A}}$ ABox \mathcal{A}^{app} , otherwise, such that \mathcal{K}^{app} is a $(DL-Lite_{\mathcal{A}}, \mathcal{T})$ -erasure of \mathcal{K} with \mathcal{F} , where $\mathcal{K}^{app} = \langle \mathcal{T}, \mathcal{A}^{app} \rangle$.*

Proof. Satisfiability of \mathcal{K}^{app} and termination of $ComputeErasure_{\mathcal{A}}^{app}$ come directly from analogous properties of $ComputeErasure_{\mathcal{AS}}$.

The fact that \mathcal{K}^{app} is a maximal $(DL-Lite_{\mathcal{A}}, \mathcal{T})$ -erasure of \mathcal{K} with \mathcal{F} can be proved in the same way as it has done for $ComputeUpdate_{\mathcal{A}}^{app}$ in theorem 3.19. \square

Finally, computational complexity result has no difference with one of $ComputeUpdate_{\mathcal{A}}^{app}$.

Theorem 3.22. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $DL-Lite_{\mathcal{A}}$ knowledge base, \mathcal{F} a finite set of ground $DL-Lite_{\mathcal{AS}}$ membership assertions such that $Mod(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$, and $\mathcal{A}^{app} = ComputeErasure_{\mathcal{A}}^{app}(\mathcal{T}, \mathcal{A}, \mathcal{F})$. Then:*

- the size of \mathcal{A}^{app} is polynomially bounded by the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$;
- computing \mathcal{A}^{app} can be done in polynomial time in the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$.

4. UPDATING DL TBOXES

Typically, intension-level does not change so often as instance-level, but it can be necessary if the structure of domain of interest has changed. For example, if in an enterprise has been restructured such that some department has got another superdepartment and cannot participate in projects of the old one, it must be reflected in the corresponding ontology by changing its intensional level. So, now we want to consider what happens if the set of protected formulas [Win88b] is not equal to a TBox, but included in it. Let us observe different updates and possible problems that can arise. But firstly, note that inserting assertion must be consistent with a set of protected formulas.

First of all, let DL-Lite $_{\mathcal{FR}}$ be a combination of DL-Lite $_{\mathcal{F}}$ and DL-Lite $_{\mathcal{R}}$ such that if $Q \sqsubseteq R$ appears in a TBox \mathcal{T} , then (*funct* R) is not in \mathcal{T} .

So, Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{FR}}$ KB, $\mathcal{T}_{pr} \subset \mathcal{T}$ a set of protected formulas. Then, we can extend the notion of update to the considered case.

Definition 4.1 (Model update). Let \mathcal{T} be a TBox in a DL \mathcal{L} , \mathcal{T}_{pr} a set of protected assertions, \mathcal{I} a model of \mathcal{T} , and \mathcal{F} a finite set of assertions in \mathcal{L} such that $Mod(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$. The *update of \mathcal{I} with \mathcal{F}* , denoted $U^{\mathcal{T}_{pr}}(\mathcal{I}, \mathcal{F})$, is defined as follows:

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

Definition 4.2 (Update). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB expressed in DL \mathcal{L} , and \mathcal{F} a finite set of assertions expressed in \mathcal{L} such that $Mod(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$. The *update of \mathcal{K} with \mathcal{F}* , denoted $\mathcal{K} \circ_{\mathcal{T}_{pr}} \mathcal{F}$, is defined as follows:

$$\mathcal{K} \circ_{\mathcal{T}_{pr}} \mathcal{F} = \bigcup_{\mathcal{I} \in Mod(\mathcal{K})} U^{\mathcal{T}_{pr}}(\mathcal{I}, \mathcal{F}).$$

Then, let us examine different cases of an insertion f .

1. $f = B_1 \sqsubseteq B_2$. Such an assertion cannot contradict ABox assertions, so if we have that for every interpretation \mathcal{I} of \mathcal{K} , $B_1^{\mathcal{I}} \not\subseteq B_2^{\mathcal{I}}$, that means that the assertion $B_1 \sqsubseteq \neg B_2$ is in $cln(\mathcal{T})$. The latter statement is implied by a theorem in [CDGL⁺07] which says that if a TBox includes only positive inclusion assertions (PIs), that a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is always satisfiable. Thus, if $\mathcal{T}' = \mathcal{T} \cup \{f\}$ is inconsistent, then we

know that a KB $\langle \mathcal{T}'_P, \mathcal{A} \rangle$, where \mathcal{T}'_P is a set of all PIs ($f \in \mathcal{T}'_P$), is satisfiable and inconsistency is caused by f and some negative inclusion assertion (NI).

An intuitive solution is make $B_1 \sqsubseteq \neg B_2$ not belonging to $cl(\mathcal{T})$ any longer.

Example 4.1. Let us consider the following DL-Lite KB:

$$\begin{array}{ll} \text{TBox} & \text{ABox} \\ A_1 \sqsubseteq A_2 & A_1(a) \text{ ,} \\ A_2 \sqsubseteq \neg A_3 & \end{array}$$

and $\mathcal{T}_{pr} = \emptyset^1$. The canonical interpretation \mathcal{I} of the KB is:

$$\begin{array}{l} A_1^{\mathcal{I}} = \{a\} \\ A_2^{\mathcal{I}} = \{a\} \\ A_3^{\mathcal{I}} = \emptyset \end{array}$$

Let an inserted assertion f be equal $A_1 \sqsubseteq A_3$. An update $U^{\mathcal{T}_{pr}}(\mathcal{I}, \{f\})$ includes only one model:

$$\begin{array}{l} A_1^{\mathcal{I}} = \{a\} \\ A_2^{\mathcal{I}} = \{a\} \\ A_3^{\mathcal{I}} = \{a\} \end{array}$$

It is obvious that f does not contradict with $A_1 \sqsubseteq A_2$ and $A_2 \sqsubseteq \neg A_3$ separately, but three of them make inconsistency. What is a TBox assertion that must be deleted? It is obvious that it is the latter one. So, the resulting KB is:

$$\begin{array}{ll} \text{TBox} & \text{ABox} \\ A_1 \sqsubseteq A_2 & A_1(a) \\ A_1 \sqsubseteq A_3 & \end{array}$$

△

2. $f = A_1 \sqsubseteq \neg A_2$. In this case we may have a contradiction with both TBox and ABox (or more precisely a contradiction involving ABox). Let us consider the following example.

Example 4.2. Having the resulting KB from the example 4.1 Its canonical interpretation is

$$\begin{array}{l} A_1^{\mathcal{I}} = \{a\} \\ A_2^{\mathcal{I}} = \{a\} \\ A_3^{\mathcal{I}} = \{a\} \end{array}$$

Let insertion $f = A_1 \sqsubseteq \neg A_3$. We have two interpretations in the update:

I	II
$A_1^{\mathcal{I}} = \emptyset$	$A_1^{\mathcal{I}} = \{a\}$
$A_2^{\mathcal{I}} = \{a\}$	$A_2^{\mathcal{I}} = \{a\}$
$A_3^{\mathcal{I}} = \{a\}$	$A_3^{\mathcal{I}} = \emptyset$

It obvious, that there exists no DL-Lite $_{\mathcal{FR}}$ KB \mathcal{K}' that describes this pair of models, since it is required that $\mathcal{K}' \models (A_1(a) \vee A_3(a)) \wedge \neg(A_1(a) \wedge A_3(a))$.

△

¹ Further we will mean that $\mathcal{T}_{pr} = \emptyset$ if it is not specified.

Now let us glance at the instance-level conflict.

Example 4.3. So, let us consider the following DL-Lite KB:

TBox	ABox	Interpretation
	$A_1(a)$	$A_1^{\mathcal{I}} = \{a\}$
	$A_2(a)$	$A_2^{\mathcal{I}} = \{a\}$

The insertion of an assertion $f = A_1 \sqsubseteq \neg A_2$ gives us two resulting models:

I	II
$A_1^{\mathcal{I}} = \{a\}$	$A_1^{\mathcal{I}} = \emptyset$
$A_2^{\mathcal{I}} = \emptyset$	$A_2^{\mathcal{I}} = \{a\}$

From the fact that $\mathcal{K}_1 \models (A_1(a) \vee A_2(a)) \wedge \neg(A_1(a) \wedge A_2(a))$, where \mathcal{K}_1 is a resulting KB, we can gather that \mathcal{K}_1 cannot be expressed in DL-Lite $_{\mathcal{FR}}$.

△

3. $f = Q_1 \sqsubseteq Q_2$. This case has not many differences with the first one.

Example 4.4. Given a DL-Lite KB and its interpretation

TBox	ABox	Interpretation
$Q_1 \sqsubseteq \neg Q_2$	$Q_1(a, b)$	$Q_1^{\mathcal{I}} = \{(a, b)\}$

Updating the model with $f = Q_1 \sqsubseteq Q_2$, we obtain

TBox	ABox	Interpretation
$Q_1 \sqsubseteq Q_2$	$Q_1(a, b)$	$Q_1^{\mathcal{I}} = \{(a, b)\}$
		$Q_2^{\mathcal{I}} = \{(a, b)\}$

△

But in contrast to the case when $f = B_1 \sqsubseteq B_2$, a new problem may arise. Recall that we have the restriction that if a role Q appears at the right side of some role inclusion assertion $Q' \sqsubseteq Q$, a functional assertion (*funct* Q) is not allowed.

Example 4.5. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{FR}}$ knowledge base:

\mathcal{T}	\mathcal{A}	Canonical interpretation \mathcal{I}
<i>(funct</i> Q)	$Q_1(a, b)$	$Q_1^{\mathcal{I}} = \{(a, b)\}$
	$Q_2(a, c)$	$Q_2^{\mathcal{I}} = \{(a, c)\}$

Let f be $Q_1 \sqsubseteq Q_2$. Then the update consist of the only model

$$\begin{aligned} Q_1^{\mathcal{I}} &= \{(a, b)\} \\ Q_2^{\mathcal{I}} &= \{(a, b), (a, c)\} \end{aligned}$$

What is the KB \mathcal{K}' that describes the set of updating models? At first sight, it must be

\mathcal{T}'	\mathcal{A}'
$Q_1 \sqsubseteq Q_2$	$Q_1(a, b)$
	$Q_2(a, c)$

Well, it is obvious that $Mod(\mathcal{K}')$ includes the obtained model. But the problem is that the model \mathcal{I}'

$$\begin{aligned} Q_1^{\mathcal{I}'} &= \{(a, b)\} \\ Q_2^{\mathcal{I}'} &= \{(a, b), (a, c), (a, d)\} \end{aligned}$$

also belongs to $Mod(\mathcal{K}')$ and there is no model $\check{\mathcal{I}}$ in $Mod(\mathcal{K})$ such that $\mathcal{I}' \in U^{\mathcal{I}'}(\check{\mathcal{I}}, \{f\})$, since $(a, d) \notin Q_2^{\check{\mathcal{I}}}$ because of functionality of Q_2 and its appearance in \mathcal{I}' contradicts the minimality change principle. It is cannot be either that $(d, b) \in Q_1^{\check{\mathcal{I}}}$ and insertion of f gives us $(a, d) \in Q_2^{\mathcal{I}'}$ because (a, d) must be in $Q_1^{\mathcal{I}'}$ by the same minimality principle.

The only legal KB \mathcal{K}' here is

$$\frac{\mathcal{T}' \quad \mathcal{A}'}{Q_1 \sqsubseteq Q_2 \quad Q_1(a, b) \quad (funct\ Q_2) \quad Q_2(a, c)},$$

but using both role functionality and role inclusion assertions gives the following complexity [ACKZ09]:

Checking satisfiability of the ontology:

ExpTime-complete in the size of the ontology (combined complexity)

PTime-complete in the size of the ABox (data complexity)

TBox reasoning:

ExpTime-complete in the size of the TBox

Query answering:

NP-complete in the size of the query and the ontology (comb. com.)

ExpTime-complete in the size of the ontology

PTime-complete in the size of the ABox (data complexity)

△

4. $f = Q_1 \sqsubseteq \neg Q_2$. This case is similar to the second one.

5. $f = (funct\ Q)$. Here we have the similar conflict as we did in the case 3: syntactical restriction on DL-Lite $_{\mathcal{FR}}$ TBox.

Example 4.6. Given a DL-Lite $_{\mathcal{FR}}$ KB

$$\frac{\begin{array}{cc} \text{TBox} & \text{ABox} \\ Q_1 \sqsubseteq Q_2 & Q_1(a, b) \\ & Q_2(a, c) \end{array}}{\text{Canonical interpretation } \mathcal{I}} \quad \begin{array}{l} Q_1^{\mathcal{I}} = \{(a, b)\} \\ Q_2^{\mathcal{I}} = \{(a, b), (a, c)\} \end{array}$$

and an insertion $f = (funct\ Q)$. Then $U^{\mathcal{I}'}(\mathcal{I}, \{f\})$ consist of two models

$$\begin{array}{cc} \text{I} & \text{II} \\ \hline Q_1^{\mathcal{I}'} = \{(a, b)\} & Q_1^{\mathcal{I}'} = \{(a, b)\} \\ Q_2^{\mathcal{I}'} = \{(a, b)\} & Q_2^{\mathcal{I}'} = \{(a, c)\} \end{array}.$$

And again, the fact that $\mathcal{K}' \models (Q_2(a, b) \vee Q_2(a, c)) \wedge \neg(Q_2(a, b) \wedge Q_2(a, c))$ is not expressible in DL-Lite $_{\mathcal{FR}}$.

△

Having $Q(a, b)$ and $Q(a, c)$ in an ABox, an insertion $f = (funct\ Q)$ brings about the same problem as in the example above.

4.1 Updating TBox: Problems and Their Solutions

In this section we consider the mentioned problems in more details and offer ways to solve them.

4.1.1 Updating TBoxes with PIs

Suppose we have a DL-Lite $_{\mathcal{FR}}$ knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a positive inclusion assertion $f = B_1 \sqsubseteq B_2$ such that $Mod(\{f\}) \cap Mod(\mathcal{T}_{pr}) \neq \emptyset$. Our goal is to build such a KB $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ that $Mod(\mathcal{K}') = \mathcal{K} \circ_{\mathcal{T}_{pr}} \mathcal{F}$.

Example 4.7. Consider the following KB \mathcal{K} and its model:

$$\frac{\text{TBox} \quad \text{ABox}}{\text{Chordates} \sqsubseteq \text{Animals} \quad \text{Mammals}(\text{wolf}), \text{Chordates}(\text{crocodile})},$$

and its model:

$$\frac{\mathcal{I}}{\text{Mammals}^{\mathcal{I}} = \{\text{wolf}\}, \text{Chordates}^{\mathcal{I}} = \{\text{crocodile}\}, \text{Animals}^{\mathcal{I}} = \{\text{crocodile}\}}$$

Let us assume that we want to update \mathcal{K} with $f = \text{Mammals} \sqsubseteq \text{Chordates}$.

An update $U^{\mathcal{T}_{pr}}(\mathcal{I}, \{f\})$ of \mathcal{I} includes \mathcal{I}' :

$$\begin{aligned} \text{Mammals}^{\mathcal{I}'} &= \{\text{wolf}\} \\ \text{Chordates}^{\mathcal{I}'} &= \{\text{crocodile}, \text{wolf}\}, \\ \text{Animals}^{\mathcal{I}'} &= \{\text{crocodile}\} \end{aligned},$$

that intuitively wrong, since wolf must be also an animal.

△

Of course, $\text{Animals}^{\mathcal{I}'}$ will include *wolf* if we put $\text{Chordates} \sqsubseteq \text{Animals}$ in \mathcal{T}_{pr} . But if it is not done or there are some reasons not to do it, we anyway would like TBox assertions to hold in an update if they do not contradict f .

Thus, the definition of update given that worked at the instance-level is no longer adequate. Let us try to obtain a new definition. But firstly

Definition 4.3 (Model update). Let \mathcal{T} be a TBox in a DL \mathcal{L} , \mathcal{T}_{pr} a set of protected assertions, \mathcal{I} a model of \mathcal{T} , and \mathcal{F} a finite set of assertions in \mathcal{L} such that $Mod(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$. The *update of \mathcal{I} with \mathcal{F}* , denoted $U^{\mathcal{T}_{pr}}(\mathcal{I}, \mathcal{F})$, is defined as follows:

$$U^{\mathcal{T}_{pr}}(\mathcal{I}, \mathcal{F}) = \{\mathcal{I}' \mid \mathcal{I}' \in Mod(\mathcal{T}_m \cup \mathcal{F})\}$$

and \mathcal{I}' satisfies the conditions that follow:

1. for every membership assertion f implied by \mathcal{K} , if f does not contradict \mathcal{F} , then $\mathcal{I}' \in Mod(\{f\} \cup \mathcal{T}_m \cup \mathcal{F})$;
2. there exists no $\mathcal{I}'' \in Mod(\mathcal{T}_m \cup \mathcal{F})$ such that \mathcal{I}'' satisfies clause 1 and $\mathcal{I} \oplus \mathcal{I}'' \subset \mathcal{I} \oplus \mathcal{I}'$;

where \mathcal{T}_m is a maximal subset of $cl(\mathcal{T})$ such that $Mod(\mathcal{T}_m \cup \mathcal{F}) \neq \emptyset$.

Introducing \mathcal{T}_m in the last definition allow us to delete that assertions that contradict f and to keep the rest.

As it has been mentioned in the example 4.1, $f = B_1 \sqsubseteq B_2$ is inconsistent with a TBox \mathcal{T} iff there is an assertion $f' \in cln(\mathcal{T})$ such that f' is equal to either $B_1 \sqsubseteq \neg B_2$ or $B_2 \sqsubseteq \neg B_1$. By construction of a DL-Lite TBox it is obvious that B_1 is disjoint with B_2 iff either disjointness is already in the TBox, or there is a concept B_3 such that B_3 and B_2 are disjoint and $B_1 \sqsubseteq B_3 \in clt(\mathcal{T})$ (or B_1 and B_3 are disjoint and $B_2 \sqsubseteq B_3 \in clt(\mathcal{T})$), where $clt(\mathcal{T})$ is transitive closure of ' \sqsubseteq ' over \mathcal{T} defined in natural way. Thereby, when we delete this negative inclusion assertion, we delete inconsistency.

The example below demonstrated why we should delete a negative assertion and not a positive one.

Let us assume a DL-Lite KB

$$\begin{array}{c|c} \text{TBox} & \text{ABox} \\ \hline A_1 \sqsubseteq A_2 & A_1(a) \\ A_2 \sqsubseteq \neg A_3 & \end{array} \quad \frac{\mathcal{I}}{A_1^{\mathcal{I}} = \{a\} \\ A_2^{\mathcal{I}} = \{a\} \\ A_3^{\mathcal{I}} = \emptyset} ,$$

and let $f = A_1 \sqsubseteq A_3$. It is easy to see that $\mathcal{T} \cup \{f\}$ is inconsistent and we have two possibilities for $\mathcal{T}_m \cup \{f\}$: either (i) $A_1 \sqsubseteq A_2$, $A_1 \sqsubseteq A_3$, or (ii) $A_2 \sqsubseteq \neg A_3$ and $A_1 \sqsubseteq A_3$.

An update $U^{\mathcal{T}_m(i)}(\mathcal{I}, \{f\})$ includes \mathcal{I}' , and an update $U^{\mathcal{T}_m(ii)}(\mathcal{I}, \{f\})$ includes \mathcal{I}'' :

$$\frac{\mathcal{I}'}{A_1^{\mathcal{I}'} = \{a\} \\ A_2^{\mathcal{I}'} = \{a\} \\ A_3^{\mathcal{I}'} = \{a\}} \quad \frac{\mathcal{I}''}{A_1^{\mathcal{I}''} = \{a\} \\ A_2^{\mathcal{I}''} = \emptyset \\ A_3^{\mathcal{I}''} = \{a\}} .$$

It is plain that $\mathcal{I} \ominus \mathcal{I}' \subset \mathcal{I} \ominus \mathcal{I}''$. That is why $\mathcal{T}_m(ii)$ is more preferable.

△

4.1.2 Updating TBoxes with NIs

Updating KBs with NI, we can have conflicts on both intensional and instance levels. The first type of conflicts is essentially the same as in the case of PIs.

Example 4.8. Suppose we have the following KB

$$\begin{array}{c|c} \text{TBox} & \text{ABox} \\ \hline A_1 \sqsubseteq A_2 & A_1(a) \\ A_2 \sqsubseteq A_3 & \end{array} \quad \frac{\mathcal{I}}{A_1^{\mathcal{I}} = \{a\} \\ A_2^{\mathcal{I}} = \{a\} \\ A_3^{\mathcal{I}} = \{a\}} ,$$

and $f = A_1 \sqsubseteq \neg A_3$. As in the example 4.1.1 we have two possibilities for $\mathcal{T}_m \cup \{f\}$: (i) $A_1 \sqsubseteq A_2$ and $A_1 \sqsubseteq \neg A_3$, or (ii) $A_2 \sqsubseteq A_3$ and $A_1 \sqsubseteq \neg A_3$. And here we should choose the first one by the same reason as in the example 4.1.1.

△

Thus, as we can see both that cases are essentially the same. Inserting $B \sqsubseteq B'$, we can build a chain

$$B \sqsubseteq B_1, B_1 \sqsubseteq B_2, \dots, B_{n-1} \sqsubseteq B_n, B_n \sqsubseteq \neg B'.$$

Inserting $B \sqsubseteq \neg B'$, a chain looks like

$$B \sqsubseteq B_1, B_1 \sqsubseteq B_2, \dots, B_{n-1} \sqsubseteq B_n, B_n \sqsubseteq B'.$$

Note that the last assertions in both chains are in TBoxes. We should delete a ‘top’ of a chain, i.e. $B_n \sqsubseteq \neg B'$ or $B_n \sqsubseteq B'$ correspondingly. Why ‘top’? Suppose, that we have $a \in B^{\mathcal{I}}$, where \mathcal{I} is a canonical model of the original KB (that is $a \in B^{\check{\mathcal{I}}}$ for every model $\check{\mathcal{I}}$ of the KB). That yields $a \in B_i^{\mathcal{I}}$ for $i = 1, \dots, n$. Deleting $B_k \sqsubseteq B_{k+1}$ instead of the ‘top’, we get thereby that $a \notin B_i^{\mathcal{I}'}$ for $i = k + 1 \dots, n$, where \mathcal{I}' is a canonical model for the obtained KB. Thus, we have $\mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}'$, where \mathcal{I}'' is a canonical model of the KB obtained with deleting the ‘top’.

Now, let us consider the conflict describing in the example 4.3. Recall that we have $A_1(a), A_2(a)$ in ABox and an insertion $f = A_1 \sqsubseteq \neg A_2$. Obtained KB \mathcal{K}' must imply $(A_1(a) \vee A_2(a)) \wedge \neg(A_1(a) \wedge A_2(a))$. Obviously, it cannot be expressed in DL-Lite $_{\mathcal{FR}}$. We consider several possibilities to deal with this.

Prioritization. In [Win90] it was introduced the notion of prioritization of predicates, that is easily encoded heuristic that helps us cope with the problem of proliferation of models. If in standard definition one of the points that $\mathcal{I}' \in U^T(I, F)$ is unexisting of \mathcal{I}'' such that $\mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}'$, under prioritization \mathcal{I}'' must agree with \mathcal{I}' on predicates with high priority².

We cannot apply this method directly to our problem, but we can use the hierarchical approach. Let us suppose that we have an order on the set of concept and role names. Then, having conflict like in in the example 4.3, we keep unchanged that predicate that is higher in the hierarchy.

Example 4.9. Consider again the example 4.3. Let the order be $A_1 > A_2$. Then, the resulting model is

$$\begin{aligned} A_1^{\mathcal{I}} &= \{a\} \\ A_2^{\mathcal{I}} &= \emptyset. \end{aligned}$$

△

² \mathcal{I}' and \mathcal{I}'' agree on a predicate A if $A^{\mathcal{I}'} = A^{\mathcal{I}''}$.

Boolean ABox. Another possibility is extend our language to one that is closed under updating. For example, the set of models like in the example 4.3, can be described by DL with boolean ABox: $(A_1 \vee A_2) \wedge \neg(A_1 \wedge A_2)(a)$. But the problem here is that a boolean KB is equivalent to a standard KB with the union \sqcup [ABHM03], that e.g. gives us complexity NP-complete of concept satisfiability [DLNN97]. Updating boolean ABoxes is considered in [LLMW06, DLB⁺09].

4.1.3 Updating TBoxes with functional assertions

Recall the example 4.5. According to it, we cannot delete functional assertions from a TBox if it violates syntax restrictions we have. From the other hand, keeping such an assertion in a TBox, we can obtain a KB with the complexity properties that are mentioned in the example. That is why we should avoid updates in such a cases if we are going to face this kind of violation. So:

An update of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with an inclusion assertion $Q \sqsubseteq R$ is forbidden if $(\text{funct } R) \in \mathcal{T}$. (4.1)

and

An update of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with an functional assertion $(\text{funct } R)$ is forbidden if $Q \sqsubseteq R \in \mathcal{T}$ for some Q . (4.2)

The other problem described in the example 4.6 can be solved with the same methods from the section 4.1.2: either prioritization or using boolean ABox.

4.2 TBox Update in DL-Lite_{FR}

In this section we consider the process of TBox update for DL-Lite_{FR}. It is obvious that in general an update of DL-Lite_{FR} KB is not expressible in DL-Lite_{FR}. So, this section is split into two parts: in the first one the algorithm is described, which takes as input a DL-Lite_{FR} KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and returns as output a DL-Lite_{FRS} KB $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$, such that $\mathcal{K} \circ_{(\mathcal{T}_{pr}, \mathcal{P})} \mathcal{F} = \text{Mod}(\mathcal{K}')$. The algorithm in the second part takes \mathcal{K}' as input and returns \mathcal{K}^{app} , where \mathcal{K}^{app} is a maximal (DL-Lite_{FR}, \mathcal{T}')-approximation of $\mathcal{K} \circ_{(\mathcal{T}_{pr}, \mathcal{P})} \mathcal{F}$.

4.2.1 TBox Update: from DL-Lite_{FR} to DL-Lite_{FRS}

In this section we suggest the algorithm to update a DL-Lite_{FRS} knowledge base with a set of TBox assertions. First, we take the assertions 4.1 and 4.2 made in the section 4.1.3 about syntax conflict. Concerning ABox conflict, we will use prioritization.

First of all, let us define the notion of prioritization.

ALGORITHM *ComputeTBoxUpdate_{FRS}(T, A, F, P)*

INPUT: finite set of inclusion assertions \mathcal{F} , satisfiable with *DL-Lite_{FR}* KB $\langle T, \mathcal{A} \rangle$

OUTPUT: a TBox T' and an ABox \mathcal{A}' made wrt prioritization \mathcal{P} , or ERROR

```

[1]  if  $\langle T_{pr}, \mathcal{F} \rangle$  is not satisfiable then return ERROR
[2]  else {
[3]     $\mathcal{F} := cl(\mathcal{F})$ 
[4]     $T^- := \emptyset$ 
[5]     $\mathcal{A}^- := \emptyset$ 
[6]    for each  $F \in \mathcal{F}$  do
[7]      if  $F$  is  $B_1 \sqsubseteq B_2$  then {
[8]        if  $T, \mathcal{A} \models B_1 \sqsubseteq \neg B_2$  then  $T^- := T^- \cup \{B_1 \sqsubseteq \neg B_2\}$ 
[9]        if  $T, \mathcal{A} \models B_2 \sqsubseteq \neg B_1$  then  $T^- := T^- \cup \{B_2 \sqsubseteq \neg B_1\}$ 
[10]       for each  $B' \sqsubseteq \neg B_1 \in cl(T)$  and for each  $B_1 \sqsubseteq \neg B'' \in cl(T)$  do {
[11]         if  $T, \mathcal{A} \models B_2 \sqsubseteq B'$  then  $T^- := T^- \cup \{B' \sqsubseteq \neg B_1\}$ 
[12]         if  $T, \mathcal{A} \models B_2 \sqsubseteq B''$  then  $T^- := T^- \cup \{B_1 \sqsubseteq \neg B''\}$ 
[13]       }
[14]       for each  $B' \sqsubseteq \neg B_2 \in cl(T)$  and for each  $B_2 \sqsubseteq \neg B'' \in cl(T)$  do {
[15]         if  $T, \mathcal{A} \models B_1 \sqsubseteq B'$  then  $T^- := T^- \cup \{B' \sqsubseteq \neg B_2\}$ 
[16]         if  $T, \mathcal{A} \models B_1 \sqsubseteq B''$  then  $T^- := T^- \cup \{B_2 \sqsubseteq \neg B''\}$ 
[17]       }
[18]     }
[19]     if  $F$  is  $B_1 \sqsubseteq \neg B_2$  then {
[20]       if  $T, \mathcal{A} \models B_1 \sqsubseteq B_2$  then  $T^- := T^- \cup \{B_1 \sqsubseteq B_2\}$ 
[21]       if  $T, \mathcal{A} \models B_2 \sqsubseteq B_1$  then  $T^- := T^- \cup \{B_2 \sqsubseteq B_1\}$ 
[22]       for each  $B' \sqsubseteq B_1 \in cl(T)$  and for each  $B_1 \sqsubseteq B'' \in cl(T)$  do {
[23]         if  $T, \mathcal{A} \models B_2 \sqsubseteq B'$  then  $T^- := T^- \cup \{B' \sqsubseteq B_1\}$ 
[24]         if  $T, \mathcal{A} \models B_2 \sqsubseteq B''$  then  $T^- := T^- \cup \{B_1 \sqsubseteq B''\}$ 
[25]       }
[26]       for each  $B' \sqsubseteq B_2 \in cl(T)$  and for each  $B_2 \sqsubseteq B'' \in cl(T)$  do {
[27]         if  $T, \mathcal{A} \models B_1 \sqsubseteq B'$  then  $T^- := T^- \cup \{B' \sqsubseteq B_2\}$ 
[28]         if  $T, \mathcal{A} \models B_1 \sqsubseteq B''$  then  $T^- := T^- \cup \{B_2 \sqsubseteq B''\}$ 
[29]       }
[30]       for each  $B' \sqsubseteq \max(B_1, B_2) \in cl(T)$  do {
[31]         for each  $B'(a) \in \mathcal{A}$  and for each  $\max(B_1, B_2)(a) \in \mathcal{A}$  do
[32]           for each  $f_B \in Saturate(\min(B_1, B_2)(a), T)$ 
[33]             if  $T, \mathcal{A} \models f_B$  then  $\mathcal{A}^- := \mathcal{A}^- \cup \{f_B\}$ 
[34]       }

```

Fig. 4.1: Algorithm *ComputeTBoxUpdate_{FRS}*, part 1

```

[35]   if  $F$  is  $Q_1 \sqsubseteq Q_2$  then {
[36]     if  $\mathcal{T}, \mathcal{A} \models Q_1 \sqsubseteq \neg Q_2$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q_1 \sqsubseteq \neg Q_2\}$ 
[37]     if  $\mathcal{T}, \mathcal{A} \models Q_2 \sqsubseteq \neg Q_1$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q_2 \sqsubseteq \neg Q_1\}$ 
[38]     for each  $Q' \sqsubseteq \neg Q_1 \in cl(\mathcal{T})$  and for each  $Q_1 \sqsubseteq \neg Q'' \in cl(\mathcal{T})$  do {
[39]       if  $\mathcal{T}, \mathcal{A} \models Q_2 \sqsubseteq Q'$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q' \sqsubseteq \neg Q_1\}$ 
[40]       if  $\mathcal{T}, \mathcal{A} \models Q_2 \sqsubseteq Q''$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q_1 \sqsubseteq \neg Q''\}$ 
[41]     }
[42]     for each  $Q' \sqsubseteq \neg Q_2 \in cl(\mathcal{T})$  and for each  $Q_2 \sqsubseteq \neg Q'' \in cl(\mathcal{T})$  do {
[43]       if  $\mathcal{T}, \mathcal{A} \models Q_1 \sqsubseteq Q'$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q' \sqsubseteq \neg Q_2\}$ 
[44]       if  $\mathcal{T}, \mathcal{A} \models Q_1 \sqsubseteq Q''$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q_2 \sqsubseteq \neg Q''\}$ 
[45]     }
[46]   }
[47]   if  $F$  is  $Q_1 \sqsubseteq \neg Q_2$  then {
[48]     if  $\mathcal{T}, \mathcal{A} \models Q_1 \sqsubseteq Q_2$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q_1 \sqsubseteq Q_2\}$ 
[49]     if  $\mathcal{T}, \mathcal{A} \models Q_2 \sqsubseteq Q_1$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q_2 \sqsubseteq Q_1\}$ 
[50]     for each  $Q' \sqsubseteq Q_1 \in cl(\mathcal{T})$  and for each  $Q_1 \sqsubseteq Q'' \in cl(\mathcal{T})$  do {
[51]       if  $\mathcal{T}, \mathcal{A} \models Q_2 \sqsubseteq Q'$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q' \sqsubseteq Q_1\}$ 
[52]       if  $\mathcal{T}, \mathcal{A} \models Q_2 \sqsubseteq Q''$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q_1 \sqsubseteq Q''\}$ 
[53]     }
[54]     for each  $Q' \sqsubseteq Q_2 \in cl(\mathcal{T})$  and for each  $Q_2 \sqsubseteq Q'' \in cl(\mathcal{T})$  do {
[55]       if  $\mathcal{T}, \mathcal{A} \models Q_1 \sqsubseteq Q'$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q' \sqsubseteq Q_2\}$ 
[56]       if  $\mathcal{T}, \mathcal{A} \models Q_1 \sqsubseteq Q''$  then  $\mathcal{T}^- := \mathcal{T}^- \cup \{Q_2 \sqsubseteq Q''\}$ 
[57]     }
[58]     for each  $Q' \sqsubseteq \max(Q_1, Q_2) \in cl(\mathcal{T})$  do
[59]       for each  $Q'(a, b) \in \mathcal{A}$  and for each  $\max(Q_1, Q_2)(a, b) \in \mathcal{A}$  do
[60]         for each  $f_Q \in \text{Saturate}(\min(Q_1, Q_2)(a, b), \mathcal{T})$ 
[61]           if  $\mathcal{T}, \mathcal{A} \models f_Q$  then  $\mathcal{A}^- := \mathcal{A}^- \cup \{f_Q\}$ 
[62]       }
[63]   if  $F$  is (funct  $Q$ ) then {
[64]     for each  $Q(a, b) \in \mathcal{A}$  do
[65]       for each  $Q_S(a, c) \in \text{Saturate}(Q(a, c), \mathcal{T})$  do
[66]         if  $\mathcal{T}, \mathcal{A} \models Q_S(a, c)$  and  $\min(b, c) = c$  then  $\mathcal{A}^- := \mathcal{A}^- \cup \{Q_S(a, c)\}$ 
[67]       for each  $Q'(a, b) \in \mathcal{A}$  such that  $Q' \sqsubseteq Q \in cl(\mathcal{T})$  do
[68]         for each  $Q_S(a, c) \in \text{Saturate}(Q(a, c), \mathcal{T})$  do
[69]           if  $\mathcal{T}, \mathcal{A} \models Q_S(a, c)$  and  $\min(b, c) = c$  then  $\mathcal{A}^- := \mathcal{A}^- \cup \{Q_S(a, c)\}$ 
[70]         }
[71]   }
[71]  $\mathcal{T}' := \mathcal{T} \cup \mathcal{F}$ 
[72] for each  $F' \in \mathcal{T}^-$  do {
[73]   if  $F'$  is  $B_1 \sqsubseteq \neg B_2$  then {
[74]      $\mathcal{T}' := \mathcal{T}' \setminus \{F'\}$ 

```

Fig. 4.2: Algorithm *ComputeTBoxUpdate_{FRS}*, part 2

```

[75]     for each  $B' \sqsubseteq B_1 \in cl(\mathcal{T})$  do
[76]       if  $B' \sqsubseteq B_1 \notin \mathcal{T}^-$  then  $\mathcal{T}' := \mathcal{T}' \cup \{B' \sqsubseteq \neg B_2\}$ 
[77]     for each  $B'' \sqsubseteq B_2 \in cl(\mathcal{T})$  do
[78]       if  $B'' \sqsubseteq B_2 \notin \mathcal{T}^-$  then  $\mathcal{T}' := \mathcal{T}' \cup \{B'' \sqsubseteq \neg B_1\}$ 
[79]     }
[80]   if  $F'$  is  $B_1 \sqsubseteq B_2$  then
[81]      $\mathcal{T}' := \mathcal{T}' \setminus \{F'\}$ 
[82]   if  $F'$  is  $Q_1 \sqsubseteq \neg Q_2$  then {
[83]      $\mathcal{T}' := \mathcal{T}' \setminus \{F'\}$ 
[84]     for each  $Q' \sqsubseteq Q_1 \in cl(\mathcal{T})$  do
[85]       if  $Q' \sqsubseteq Q_1 \notin \mathcal{T}^-$  then  $\mathcal{T}' := \mathcal{T}' \cup \{Q' \sqsubseteq \neg Q_2\}$ 
[86]     for each  $Q' \sqsubseteq Q_2 \in cl(\mathcal{T})$  do
[87]       if  $Q' \sqsubseteq Q_2 \notin \mathcal{T}^-$  then  $\mathcal{T}' := \mathcal{T}' \cup \{Q' \sqsubseteq \neg Q_1\}$ 
[88]     }
[89]   if  $F'$  is  $Q_1 \sqsubseteq Q_2$  then
[90]      $\mathcal{T}' := \mathcal{T}' \setminus \{F'\}$ 
[91]   }
[92]  $\mathcal{A}' := \mathcal{A}$ 
[93] for each  $f' \in \mathcal{A}^-$  do {
[94]   if  $f' = B(a)$  then {
[95]      $\mathcal{A}' := \mathcal{A}' \setminus B(a)$ 
[96]     for each  $B \sqsubseteq B_1 \in cl(\mathcal{T})$  do
[97]       if  $B_1(a) \notin \mathcal{A}^-$  then  $\mathcal{A}' := \mathcal{A}' \cup \{B_1(a)\}$ 
[98]     if  $f' = \exists Q(a)$  then
[99]       for each  $\exists Q^- \sqsubseteq C_1 \in cl(\mathcal{T})$  do
[100]         $\mathcal{A}' := \mathcal{A}' \cup \{C_1(z_{\exists R(a)})\}$ , with  $z_{\exists R(a)}$  new variable
[101]      }
[102]   if  $f' = Q(a, b)$  then {
[103]      $\mathcal{A}' := \mathcal{A}' \setminus \{Q(a, b), \exists Q(a), \exists Q^-(b)\}$ 
[104]     for each  $Q \sqsubseteq Q_1$  in  $cl(\mathcal{T})$  do
[105]       if  $Q_1(a, b) \notin \mathcal{A}^-$  then  $\mathcal{A}' := \mathcal{A}' \cup \{Q_1(a, b)\}$ ;
[106]     for each  $\exists Q \sqsubseteq C_2$  in  $cl(\mathcal{T})$  do
[107]       if  $C_2(a) \notin \mathcal{A}^-$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_2(a)\}$ 
[108]     for each  $\exists Q^- \sqsubseteq C_3$  in  $cl(\mathcal{T})$  do
[109]       if  $C_3(b) \notin \mathcal{A}^-$  then  $\mathcal{A}' := \mathcal{A}' \cup \{C_3(b)\}$ 
[110]     }
[111]   }
[112] }

```

Fig. 4.3: Algorithm *ComputeTBoxUpdate_{FRS}*, part 3

Definition 4.4. Let us have a DL KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{F} \rangle$, and let us assume that there exist:

- a linear order on the set of all basic concepts occurring in \mathcal{K} : $B_1 <_C B_2 <_C \dots <_C C_n$ ³;
- a linear order on the set of all basic roles occurring in \mathcal{K} : $Q_1 <_R Q_2 <_R \dots <_R Q_m$;
- a linear order on the set of all constants occurring in \mathcal{K} : $a_1 <_I a_2 <_I \dots <_I a_k$.

Then the prioritization \mathcal{P} is a pair $\langle \max, \min \rangle$ of function such that

$$\max(X_1, X_2) = \begin{cases} X_1 & \text{if } X_2 <_Y X_1 \\ X_2 & \text{if } X_1 <_Y X_2 \\ \text{ERROR} & \text{otherwise} \end{cases};$$

$$\min(X_1, X_2) = \begin{cases} X_2 & \text{if } X_2 <_Y X_1 \\ X_1 & \text{if } X_1 <_Y X_2 \\ \text{ERROR} & \text{otherwise} \end{cases},$$

where $Y \in \{C, R, I\}$.

Now, let us introduce the notion of model update under prioritization.

Definition 4.5 (Model update under prioritization). Let \mathcal{T} be a TBox in a DL \mathcal{L} , \mathcal{T}_{pr} a set of protected assertions, \mathcal{P} a prioritization over \mathcal{K} , \mathcal{I} a model of \mathcal{T} , and \mathcal{F} a finite set of assertions in \mathcal{L} such that $Mod(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$. The *update of \mathcal{I} with \mathcal{F}* , denoted $U^{\mathcal{T}_{pr}, \mathcal{P}}(\mathcal{I}, \mathcal{F})$, is defined as follows:

$$U^{\mathcal{T}_{pr}, \mathcal{P}}(\mathcal{I}, \mathcal{F}) = \{\mathcal{I}' \mid \mathcal{I}' \in Mod(\mathcal{T}_m \cup \mathcal{F})\},$$

and \mathcal{I}' satisfies the conditions that follow:

1. for every NI $L \sqsubseteq R \in cl(\mathcal{F})$, $\min(L, R)^{\mathcal{I}'} \cap \max(L, R)^{\mathcal{I}} = \emptyset$;
2. for every (*funct* Q) $\in \mathcal{F}$, such that if for every $a \in \exists Q^{\mathcal{I}}$, $\mathcal{I} \models Q(a, b_1), \dots, Q(a, b_n)$, then \mathcal{I}' satisfies $Q(a, b)$, where $b = \max(b_1, \dots, b_n)$ ⁴ and does not satisfy $Q(a, b')$ for every $b' \neq b$;
3. for every membership assertion f implied by \mathcal{K} , if f does not contradict \mathcal{F} with respect to clauses 1 and 2, then $\mathcal{I}' \in Mod(\{f\} \cup \mathcal{T}_m \cup \mathcal{F})$;
4. there exists no $\mathcal{I}'' \in Mod(\mathcal{T}_m \cup \mathcal{F})$ such that \mathcal{I}'' satisfies clauses 1-3 and $\mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}'$;

where \mathcal{T}_m is a maximal subset of $cl(\mathcal{T})$ such that $Mod(\mathcal{T}_m \cup \mathcal{F}) \neq \emptyset$.

³ This set of all concepts includes also $\exists Q$ and $\exists Q^-$ for any Q appearing in \mathcal{K} .

⁴ The function \max of arity two can be extended to n -ary case trivially.

The condition $\min(B_1, B_2)^{\mathcal{I}'} \cap \max(B_1, B_2)^{\mathcal{I}'} = \emptyset$ provides that in the case of ABox conflicts (see example 4.2) it will be solved according to the prioritization.

Definition 4.6 (Update). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB expressed in DL \mathcal{L} , subset \mathcal{T}_{pr} of \mathcal{T} is a set of protected assertions, \mathcal{P} is a prioritization over \mathcal{K} , and \mathcal{F} a finite set of TBox assertions expressed in \mathcal{L} such that $Mod(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$. The *update of \mathcal{K} with \mathcal{F}* with respect to \mathcal{P} , denoted $\mathcal{K} \circ_{(\mathcal{T}_{pr}, \mathcal{P})} \mathcal{F}$, is defined as follows:

$$\mathcal{K} \circ_{\mathcal{T}} \mathcal{F} = \bigcup_{\mathcal{I} \in Mod(\mathcal{K})} U^{\mathcal{T}_{pr}, \mathcal{P}}(\mathcal{I}, \mathcal{F}).$$

The algorithm on the Fig. 4.1-4.3 takes as input a DL-Lite_{FR} knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a set of finite DL-Lite_{FS} TBox assertions \mathcal{F} , and returns as output an DL-Lite_{FRS} TBox \mathcal{T}' and ABox \mathcal{A}' such that $Mod(\mathcal{K}') = \mathcal{K} \circ_{\mathcal{T}} \mathcal{F}$, where $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$. Or more precisely:

1. First, it checks whether \mathcal{T}_{pr} and \mathcal{F} are satisfiable (line 1), and if they are, it adds to \mathcal{F} all those assertions that logically implied by \mathcal{F} , i.e. it computes $cl(\mathcal{F})$.
2. Second, it makes sets \mathcal{T}^- and \mathcal{A}^- of TBox and membership assertions correspondingly which contradict \mathcal{F} and are logically implied by \mathcal{K} (lines 3-70).

In the case when $F \in \mathcal{F}$ is a positive inclusion assertion (line 7) $B_1 \sqsubseteq B_2$, NIs $B_1 \sqsubseteq \neg B_2$ and $B_2 \sqsubseteq \neg B_1$ contradict F directly, therefore, if they are implied by \mathcal{K} , we add them to \mathcal{T}^- (lines 8-9). Moreover, if some superconcept B' (or B'') of B_2 is disjoint with B_1 , it yields disjointness of B_2 and B_1 ; that is why we add $B' \sqsubseteq \neg B_1$ (or $B_1 \sqsubseteq \neg B''$) to \mathcal{T}^- . Then, we should do the same checking for superconcepts of B_1 (lines 10-16).

Analogously, if F is a negative inclusion assertion $B_1 \sqsubseteq \neg B_2$ (line 19), we check if $B_1 \sqsubseteq B_2$ and $B_1 \sqsubseteq B_2$ are implied by \mathcal{K} (lines 20-21), and we make checking about superconcepts of B_1 and B_2 in the same way as for inserting a PI (lines 22-28).

The difference between these two cases above (PIs and NIs) does not actually matter, since $\neg C = \neg B$ if $C = B$, and $\neg C = B$ if $C = \neg B$. But we consider them separately because in the case of NI we may have additional conflict on the ABox level. Thus, we choose the “bigger” predicate $\max(B_1, B_2)$ (line 30), and for each constant a such that $\max(B_1, B_2)(a)$ is implied by \mathcal{K} (line 31), we check whether \mathcal{K} implies $\min(B_1, B_2)(a)$ as well. If it does, we add the candidate $\min(B_1, B_2)(a)$ to be deleted to \mathcal{A}^- (lines 32-33).

Further, we do the similar things for role inclusion assertion, positive $Q_1 \sqsubseteq Q_2$ (lines 35-44) and negative $Q_1 \sqsubseteq \neg Q_2$ (lines 47-61). Note that interactions between roles and concepts are reflected in $cl(\mathcal{T})$. For example, having $Q' \sqsubseteq Q''$ and $\exists Q'' \sqsubseteq A'$ in a TBox, the assertion $\exists Q' \sqsubseteq A'$ is implied; by definition of $cl(\mathcal{T})$, the assertion belongs to it.

The last type of TBox assertions is a functional one. Here we do not change \mathcal{T}^- , but only \mathcal{A}^- , since we may have only problems at the ABox level. Thereby, if $(\text{funct } Q) \in \mathcal{F}$ (line 63), we check for each $Q(a, b)$ in the ABox (line 64) whether functionality is violated. We examine every role Q_S such that $Q_S(a, c) \in \mathcal{A}$ for some c and $\mathcal{T}, Q_S(a, c) \models Q(a, c)$ (line 65), and if $c <_I b$, $Q_S(a, c)$ is added to \mathcal{A}^- (line 66). Then we do the similar thing for those $Q(a', b')$ that are not in \mathcal{A} but are implied by \mathcal{K} (lines 67-69). Note that if in line 66 $b <_I c$, then we should add $Q(a, b)$ to \mathcal{A}^- , that is done at the lines 67-69.

3. Third, we make changes in the TBox as consistent with \mathcal{F} (line 71-90). We start with \mathcal{T}' equal to $\mathcal{T} \cup \mathcal{F}$ (line 71), and then delete all the assertions that contradict \mathcal{F} and add those assertions that are implied by \mathcal{T} and do not contradict \mathcal{F} . Particularly, if $B_1 \sqsubseteq B_2 \in \mathcal{T}^-$ then we exclude it from \mathcal{T}' and add $B' \sqsubseteq \neg B_2$ if $B' \sqsubseteq B_1$ and it is not in \mathcal{T}^- (lines 73-76). Similarly, we do the same for all subconcepts of B_2 (lines 77-78). In analogous way, we do it for role PIs (lines 80-81), role NIs (line 82-87), and role PIs (lines 89-90).

Note that for the case of excluding concept PI (and role PI as well) we need not do the subconcept checking, since $B_1 \sqsubseteq B_2$ (resp. $Q_1 \sqsubseteq Q_2$) can appear in \mathcal{T}^- if and only if they must be disjoint according to \mathcal{F} . Thus, if $B' \sqsubseteq B_1$, the original KB \mathcal{K} implies that $B' \sqsubseteq B_2$, but since B_1 and B_2 must be disjoint with respect to \mathcal{F} , so all the subconcepts of B_1 are also disjoint with B_2 . Subconcept checking is required in the case of PIs in \mathcal{T}^- , because they arise if we have $B_1 \sqsubseteq B_2$ in \mathcal{F} and B_1 can be disjoint with a subconcept of B_2 and be another subconcept of B_2 .

4. Finally, we change the ABox. Starting with \mathcal{A}' equal \mathcal{A} , we delete all membership assertions that contradict \mathcal{F} (NIs and functional assertions) and put those assertions that are implied by deleted ones and do not contradict \mathcal{F} , i.e. do not belong to \mathcal{A}^- , in the similar way as we did for DL-Lite $_{\mathcal{R}\mathcal{S}}$ instance level update (fig. 3.1) (lines 92-109).

Consider correctness of the algorithm. Start from satisfiability of \mathcal{K}' .

Lemma 4.1 (Satisfiability). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{F}\mathcal{R}}$ knowledge base, \mathcal{P} a prioritization over \mathcal{K} , \mathcal{F} a finite set of DL-Lite $_{\mathcal{F}\mathcal{R}}$ TBox assertions such that $\text{Mod}(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$, \mathcal{F} satisfies the assertions 4.1 and 4.2, and \mathcal{K}' the DL-Lite $_{\mathcal{F}\mathcal{R}\mathcal{S}}$ knowledge base such that $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$, where $(\mathcal{T}', \mathcal{A}') = \text{ComputeTBoxUpdate}_{\mathcal{F}\mathcal{R}\mathcal{S}}(\mathcal{T}, \mathcal{A}, \mathcal{F}, \mathcal{P})$. Then, we have that \mathcal{K}' is always satisfiable.*

Proof. According to algorithm $\text{ComputeTBoxUpdate}_{\mathcal{F}\mathcal{R}\mathcal{S}}$, \mathcal{K}' is obtained by

1. Inserting into TBox:
 - a finite set of TBox assertions \mathcal{F} such that $\text{Mod}(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$ (line 71).

- a finite set of TBox assertions \mathcal{T}'' that do not contradict \mathcal{F} and are logically implied by $cl(\mathcal{T}_m \cup \mathcal{F})$; it is done in the lines 76, 78, 85, and 87.
2. Deleting from TBox: all those PIs and NIs were deleted that contradict \mathcal{F} (lines 8-16, 20-28, 36-44, and 48-56).
 3. Inserting into ABox: a finite set \mathcal{A}'' of ABox assertions that are implied by \mathcal{K} and do not contradict \mathcal{F} (lines 97, 105, 107, and 109).
 4. Deleting from ABox: all those membership assertions that contradict \mathcal{F} with respect to \mathcal{P} (lines 33, 61, 66, and 69).

Thus, we have that $\mathcal{A}' = (\mathcal{A} \cup \mathcal{A}'') \setminus \mathcal{A}^-$, $\mathcal{T}' = (\mathcal{T} \cup \mathcal{F} \cup \mathcal{T}'') \setminus \mathcal{T}^-$. Then, by the data \mathcal{K} is satisfiable (i.e. $Mod(\mathcal{T} \cup \mathcal{A}) \neq \emptyset$) that leads to satisfiability of \mathcal{K}' : $Mod(\mathcal{T}' \cup \mathcal{A}') = Mod(((\mathcal{T} \cup \mathcal{F} \cup \mathcal{T}'') \setminus \mathcal{T}^-) \cup ((\mathcal{A} \cup \mathcal{A}'') \setminus \mathcal{A}^-)) = Mod((\mathcal{T} \cup \mathcal{F} \cup \mathcal{T}'' \cup \mathcal{A} \cup \mathcal{A}'') \setminus (\mathcal{T}^- \cup \mathcal{A}^-))$. \square

Consider the problem of algorithm termination.

Lemma 4.2 (Termination). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{FR} knowledge base, \mathcal{P} a prioritization over \mathcal{K} , \mathcal{F} a finite set of DL-Lite_{FR} TBox assertions such that $Mod(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$, \mathcal{F} satisfies the assertions 4.1 and 4.2. Then the algorithm $ComputeTBoxUpdate_{FRS}(\mathcal{T}, \mathcal{A}, \mathcal{F}, \mathcal{P})$ terminates, returning *ERROR* if $Mod(\mathcal{T}_{pr} \cup \mathcal{F}) = \emptyset$, and a TBox \mathcal{T}' and an ABox \mathcal{A}' such that $\langle \mathcal{T}', \mathcal{A}' \rangle$ is a DL-Lite_{FRS} knowledge base, otherwise.*

Proof. The proof of the theorem follows from the observations below:

1. The algorithm calls *Saturate* a finite number of times. This algorithm stops, since $cl(\mathcal{T})$ is finite for a finite \mathcal{T} , returning a finite output.
2. The algorithm of checking whether $\mathcal{K} \models F$ for some F terminates; the number of membership assertions to be checked is finite since they belong to outputs of *Saturate*.
3. Deleting TBox assertions of \mathcal{T}^- and introducing new assertions (lines 72-90) terminates since both $cl(\mathcal{T})$ and \mathcal{T}^- are finite. The latter set is finite again because of finiteness of $cl(\mathcal{T})$.
4. Deleting membership assertions of \mathcal{A}^- and introducing new assertions (lines 93-109) terminates since both \mathcal{A}^- and $cl(\mathcal{T})$ are finite.

$\langle \mathcal{T}', \mathcal{A}' \rangle$ is obviously a DL-Lite_{FRS} KB, since all inserted TBox assertions are DL-Lite_{FR} assertions, every inserted ABox assertion is a DL-Lite_{FRS} assertion by definition. \square

Now, let us consider soundness and completeness of the algorithm.

Lemma 4.3 (Soundness). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{FR}}$ knowledge base, \mathcal{P} a prioritization over \mathcal{K} , \mathcal{F} a finite set of DL-Lite $_{\mathcal{FR}}$ TBox assertions such that $\text{Mod}(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$, \mathcal{F} satisfies the assertions 4.1 and 4.2, and \mathcal{K}' the DL-Lite $_{\mathcal{FRS}}$ knowledge base such that $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$. Then, for every model $\mathcal{I}' \in \text{Mod}(\mathcal{K}')$, we have that:*

$$\exists \mathcal{I} \in \text{Mod}(\mathcal{K}) \text{ such that } \mathcal{I}' \in U^{\mathcal{T}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F}),$$

where $(\mathcal{T}', \mathcal{A}') = \text{ComputeTBoxUpdate}_{\mathcal{FRS}}(\mathcal{T}, \mathcal{A}, \mathcal{F}, \mathcal{P})$.

Proof. The structure of proof is following: firstly, we show how to build for each model \mathcal{I}' of \mathcal{K}' a model \mathcal{I} of \mathcal{K} . Then, we show that $\mathcal{I} \in U^{\mathcal{T}_{pr}}(\mathcal{I}, \mathcal{F})$.

Let \mathcal{I}' be a model of \mathcal{K}' and not be a model of \mathcal{K} . This means that there exist a subset \mathcal{A}^- of \mathcal{A} and a subset \mathcal{T}^- of $\mathcal{T} \setminus \mathcal{T}_{pr}$ such that for each $f \in \mathcal{A}^-$, $\text{Mod}(\{f\}) \cap \text{Mod}(\mathcal{I}') = \emptyset$, and for each $F \in \mathcal{T}^-$, $\text{Mod}(\{F\}) \cap \text{Mod}(\mathcal{I}') = \emptyset$. Let \mathcal{F}^- designates $\mathcal{T}^- \cup \mathcal{A}^-$.

We build a model \mathcal{I} step by step as it described below.

STEP 1. Set $\mathcal{I}_0 := \mathcal{I}'$.

STEP 2. Modify \mathcal{I}_0 as follows. For each $f \in \mathcal{A}^-$ do:

1. if f is $B(a)$, then set $a^{\mathcal{I}_0} \in B^{\mathcal{I}_0}$;
2. if f is $Q(a, b)$, then set $(a, b)^{\mathcal{I}_0} \in Q^{\mathcal{I}_0}$, $a^{\mathcal{I}_0} \in \exists Q^{\mathcal{I}_0}$, and $b^{\mathcal{I}_0} \in \exists Q^{-\mathcal{I}_0}$.

STEP 3. For each $F \in \mathcal{T}^-$ do:

1. if F is $B_1 \sqsubseteq B_2$, then for each $c \in B_1^{\mathcal{I}_0}$ set $c \in B_2^{\mathcal{I}_0}$;
2. if F is $B_1 \sqsubseteq \neg B_2$, then for each $c \in B_1^{\mathcal{I}_0}$ set $c \notin B_2^{\mathcal{I}_0}$;
3. if F is $Q_1 \sqsubseteq Q_2$, then for each $(c, d) \in Q_1^{\mathcal{I}_0}$ set $(c, d) \in Q_2^{\mathcal{I}_0}$, $c \in \exists Q_2^{\mathcal{I}_0}$, and $d \in \exists Q_2^{-\mathcal{I}_0}$;
4. if F is $Q_1 \sqsubseteq \neg Q_2$, then for each $(c, d) \in Q_1^{\mathcal{I}_0}$ set $(c, d) \notin Q_2^{\mathcal{I}_0}$.

STEP 4. Set $\mathcal{I}_i := \mathcal{I}_{i-1}$ and $i := 1$.

Repeat the following rules:

1. if $c \in B^{\mathcal{I}_i}$, $B \sqsubseteq C \in \mathcal{T}$, and $c \notin C^{\mathcal{I}_{i-1}}$, then set $a \in C^{\mathcal{I}_i}$;
2. if $c \in \exists Q^{\mathcal{I}_i}$ and there is no $d \in \Delta^{\mathcal{I}_{i-1}}$ such that $(c, d) \in Q^{\mathcal{I}_{i-1}}$, then add $(c, d') \in Q^{\mathcal{I}_i}$ and $d' \in \exists Q^{-\mathcal{I}_i}$, where $d' \in \Delta^{\mathcal{I}_i}$ is a new element such that if $(\text{funct } Q) \in \mathcal{T}$, then there exists no c'' such that $(c'', d') \in Q^{\mathcal{I}_{i-1}}$;
3. if $(c, d) \in Q^{\mathcal{I}_i}$, $Q \sqsubseteq R \in \mathcal{T}$, and $(c, d) \notin R^{\mathcal{I}_{i-1}}$, then set $(c, d) \in R^{\mathcal{I}_i}$.
4. if $(c, d) \in Q^{\mathcal{I}_i}$ and $c \notin \exists Q^{\mathcal{I}_{i-1}}$ (resp., $d \notin Q^{-\mathcal{I}_{i-1}}$), then set $c \in \exists Q^{\mathcal{I}_i}$ (resp., $d \in \exists Q^{-\mathcal{I}_i}$);
5. if $c \in \neg \exists Q^{\mathcal{I}_i}$, then for each $(c, d) \in Q^{\mathcal{I}_{i-1}}$ set $(c, d) \notin Q^{\mathcal{I}_i}$, and if there exists no $c' \neq c$ such that $(c', d) \in Q^{\mathcal{I}_{i-1}}$, then set $d \notin \exists Q^{-\mathcal{I}_i}$.
6. $i := i + 1$.

Until $\mathcal{I}_i = \mathcal{I}_{i-1}$.

We apply STEP 4 until $\mathcal{I}_{n+1} = \mathcal{I}_n$, and then set $\mathcal{I} = \mathcal{I}_n$. It is obvious that \mathcal{I} is a model of \mathcal{K} : (i) initially, \mathcal{I}_0 is a model of $\mathcal{T} \setminus \mathcal{T}^-$ (by STEP 1, since \mathcal{I}' is) and it comes to be a model of \mathcal{T}^- after STEP 3; (ii) in the STEP 3 we make those settings that are logically implied by \mathcal{T} , so \mathcal{I}_i is a model of \mathcal{T} for each i ; (iii) also, \mathcal{I} is a model of \mathcal{A} , since \mathcal{I}_0 is a model of $\mathcal{A} \setminus \mathcal{A}^-$ and turns to be a model of \mathcal{A}^- after STEP 2. Note that \mathcal{I}' is a model of \mathcal{F} since $\mathcal{F} \subseteq \mathcal{T}'$.

Now, let us suppose that $\mathcal{I}' \notin U^{\mathcal{T}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F})$. That means that there exists a model $\mathcal{I}'' \in Mod(\mathcal{T}_m \cup \mathcal{F})$ such that $\mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}'$. The latter inclusion means that one of the following cases occurs:

1. There is c such that $c \in A^{\mathcal{I}}$ and $c \in A^{\mathcal{I}''}$, but $c \notin A^{\mathcal{I}'}$.
2. There is c such that $c \notin A^{\mathcal{I}}$ and $c \notin A^{\mathcal{I}''}$, but $c \in A^{\mathcal{I}'}$.
3. There is (c, d) such that $(c, d) \in Q^{\mathcal{I}}$ and $(c, d) \in Q^{\mathcal{I}''}$, but $(c, d) \notin Q^{\mathcal{I}'}$.
4. There is (c, d) such that $(c, d) \notin Q^{\mathcal{I}}$ and $(c, d) \notin Q^{\mathcal{I}''}$, but $(c, d) \in Q^{\mathcal{I}'}$.

Let us consider these cases one by one.

1. In this case, we have following possibilities. First of them is: $A(a) \in \mathcal{A}^-$ and was removed by the algorithm *ComputeTBoxUpdate_{FRS}*.
 - It was added to \mathcal{A}^- at the line 33 (because of inserting some NI). In this case, $A(c) \in Saturate(B'(c), \mathcal{T})$ and it contradicts an NI $B' \sqsubseteq \neg B''$ or $B'' \sqsubseteq \neg B'$, i.e., $A(c)$ contradicts \mathcal{F} (with respect to the prioritization \mathcal{P}). Since \mathcal{I}'' is a model of \mathcal{F} , so $c \in A^{\mathcal{I}''}$ yields contradiction.

Second possibility is that c was set into \mathcal{A} by the procedure of building \mathcal{I} .

- Suppose it was done in STEP 2, item 1. That is $c \in A^{\mathcal{I}}$ was set to satisfy \mathcal{A}^- . That is if $c \in A^{\mathcal{I}''}$, then \mathcal{I}'' is not a model of \mathcal{F} . Thus, we have a contradiction.
- Suppose it was done in STEP 3, item 1. That is there is an inclusion assertion $B' \sqsubseteq A$ contradicting \mathcal{F} . The latter fact leads to that B' and A must be disjoint in any model of update of \mathcal{I} . Furthermore, since $c \in B'^{\mathcal{I}}$, $\mathcal{K} \models B'(c)$ ⁵ and $\mathcal{K} \models A(c)$, but $\mathcal{K}' \not\models A(c)$, it means that $\max(B', A) = B'$. Then, $c \in \min(B', A)^{\mathcal{I}} \cap \max(B', A)^{\mathcal{I}''}$ contradicts the notion of update according with the definition 4.5 and $\mathcal{I}'' \notin U^{\mathcal{T}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F})$.
- Suppose it was done in STEP 4, item 1. This means that $c \in B^{\mathcal{I}}$ and $B \sqsubseteq A \in \mathcal{T}$. That is $c \in B^{\mathcal{I}}$ was set to satisfy \mathcal{A}^- and $c \in A^{\mathcal{I}}$ was set to satisfy \mathcal{T} . That is $\mathcal{K} \models B(c)$ and $\mathcal{K} \models A(c)$ because of $B \sqsubseteq A$.

⁵ The case when $\mathcal{K} \not\models B'(c)$ but $\mathcal{K}' \models B'(c)$ because of PI $B'' \sqsubseteq B' \in \mathcal{F}$ can be solved in similar way taking into account that B'' and A must be disjoint in any model of update of \mathcal{I} as well.

But $B(c)$ was removed from \mathcal{A}' at the line 95, and the fact that $A(c)$ was not added to \mathcal{A}' at the line 97 means that $A(c)$ contradicts \mathcal{F} with respect to \mathcal{P} . Thus, $c \in A^{\mathcal{I}''}$ contradicts that \mathcal{I}'' belongs to $U^{\mathcal{I}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F})$.

2. This case can be reduced to the previous one by considering that $c \in \neg A^{\mathcal{I}}$, $a \in \neg A^{\mathcal{I}''}$, and $a \notin \neg A^{\mathcal{I}'}$.
3. This case also yields two main possibilities. The first one is that $Q(c, d)$ was removed by the algorithm.
 - Similar to the concept case 1, $Q(c, d)$ was added to \mathcal{A}^- at the line 61. In this case, $Q(c, d) \in \text{Saturate}(Q'(c, d), \mathcal{T})$ and it contradicts an NI $Q' \sqsubseteq \neg Q''$ or $Q'' \sqsubseteq \neg Q'$, i.e., $Q(c, d)$ contradicts \mathcal{F} (with respect to the prioritization \mathcal{P}). Since \mathcal{I}'' is a model of \mathcal{F} , so $c \in A^{\mathcal{I}''}$ gives contradiction.
 - $Q(c, d)$ logically implies some $B(c)$ that comes from $\text{Saturate}(B(c), \mathcal{T})$, and $B(c)$ contradicts \mathcal{F} with respect to \mathcal{P} (lines 60-61)⁶. Here, $(c, d) \in Q^{\mathcal{I}''}$ also contradicts that \mathcal{I}'' is a model of \mathcal{F} .
 - $Q(c, d)$ was put to \mathcal{A}^- either at the line 66 or at the line 69. Anyway, it violets a functional assertion in \mathcal{F} , and $(c, d) \in A^{\mathcal{I}''}$ contradicts that \mathcal{I}'' is a model of \mathcal{F} .

Another possibility is that (c, d) was set into \mathcal{Q} by the procedure of building \mathcal{I} .

- Suppose it was done in STEP 2, item 2. That is $(c, d) \in Q^{\mathcal{I}}$ was set to satisfy \mathcal{A}^- . That is if $(c, d) \in Q^{\mathcal{I}''}$, then \mathcal{I}'' is not a model of \mathcal{F} . Thus, we have a contradiction.
- Suppose it was done in STEP 3, item 3. This case is analogous to the similar case in the section 1 about concepts. Like there, we have that there is an inclusion assertion $Q' \sqsubseteq Q$ that contradicts \mathcal{F} . This leads that Q' and Q must be disjoint in any model of update of \mathcal{I} . Then, $\mathcal{K} \models Q'(c, d)$ and $\mathcal{K} \models Q(c, d)$, but $\mathcal{K}' \not\models Q(c, d)$, it means that $\max(Q', Q) = Q'$. Thus, $(c, d) \in \min(Q', Q)^{\mathcal{I}} \cap \max(Q', Q)^{\mathcal{I}''}$ contradicts the notion of update according with the definition 4.5 and $\mathcal{I}'' \notin U^{\mathcal{I}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F})$.
- Suppose it was done in STEP 4, item 3. This means that $(c, d) \in Q^{\mathcal{I}}$ and $Q' \sqsubseteq Q \in \mathcal{T}$. That is $(c, d) \in Q^{\mathcal{I}}$ was set to satisfy \mathcal{A}^- and $(c, d) \in Q^{\mathcal{I}}$ was set to satisfy \mathcal{T} . That is $\mathcal{K} \models Q'(c, d)$ and $\mathcal{K} \models Q(c, d)$ because of $Q' \sqsubseteq Q$. But $Q'(c)$ was removed from \mathcal{A}' at the line 103, and the fact that $Q(c, d)$ was not added to \mathcal{A}' at the line 105 means that $Q(c, d)$ contradicts \mathcal{F} with respect to \mathcal{P} . Thus, $(c, d) \in Q^{\mathcal{I}''}$ contradicts that \mathcal{I}'' belongs to $U^{\mathcal{I}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F})$.
- Suppose it was done in STEP 4, item 2. That is $c \in \exists Q^{\mathcal{I}}$ and there exists no d' such that $(c, d') \in Q^{\mathcal{I}'}$. That means that $\exists Q(c)$ is logically

⁶ Recall that Q designates either P or P^- .

implied by some assertion from \mathcal{F}^- . Thus, if $(c, d) \in Q^{\mathcal{I}''}$, then it contradicts that \mathcal{I}'' is a model of $U^{\mathcal{I}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F})$.

4. Let us consider the latter case. Since the building \mathcal{I} was begun from \mathcal{I}' , so $(c, d) \notin Q^{\mathcal{I}}$ was set in either:
- STEP 3, item 4. This means that $(c, d) \notin Q^{\mathcal{I}}$ was set to satisfy some NI $Q' \sqsubseteq \neg Q \in \mathcal{T}^-$ that contradicts \mathcal{F} . Thus, if $(c, d) \in Q^{\mathcal{I}''}$, it cannot be a model of $U^{\mathcal{I}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F})$. There is a contradiction.
 - Or STEP 4, item 5. This means that firstly (c, d) belonged to $Q^{\mathcal{I}^i}$, but it was removed at the iteration j , where $i < j$, to satisfy \mathcal{T} , so $c \in \neg \exists Q^{\mathcal{I}}$. That is $c \notin \exists Q^{\mathcal{I}^0}$ appeared at the STEP 3 item 4 to satisfy a NI contradicting \mathcal{F} . Since $\mathcal{K} \models \neg \exists Q(c)$, so $c \in \exists Q^{\mathcal{I}'}$, or $(c, d) \in Q^{\mathcal{I}'}$ appeared because of algorithm. They cannot be set at the line 97, neither at the line 103, since they add only those membership assertions that implied by \mathcal{K} . Thus, they appeared because of some PI in \mathcal{F} : (i) $Q' \sqsubseteq Q \in \mathcal{F}$, and $Q'(c, d)$ is implied by $(\mathcal{T} \setminus \mathcal{T}^-, \mathcal{A} \setminus \mathcal{A}^-, \mathcal{F})$; so, if $(c, d) \notin Q^{\mathcal{I}''}$, \mathcal{I}'' is not a model of \mathcal{F} . (ii) $B \sqsubseteq \exists Q \in \mathcal{F}$ and $B(c)$ is implied by $(\mathcal{T} \setminus \mathcal{T}^-, \mathcal{A} \setminus \mathcal{A}^-, \mathcal{F})$. Since both \mathcal{I}' and \mathcal{I}'' are models of \mathcal{F} , we have that $c \in \exists Q^{\mathcal{I}'}$ and $c \in \exists Q^{\mathcal{I}''}$, but according to assumption, $d \notin \exists Q^{\mathcal{I}''}$. Thus, for some d' , $(c, d') \in Q^{\mathcal{I}''}$ and $(c, d') \notin Q^{\mathcal{I}'}$. The latter expression yields $\mathcal{I} \ominus \mathcal{I}'' \not\subseteq \mathcal{I} \ominus \mathcal{I}'$.

Therefore, we have that there is no interpretation \mathcal{I}'' such that $\mathcal{I}'' \in \text{Mod}(\mathcal{T}_m \cup \mathcal{F})$ and $\mathcal{I} \ominus \mathcal{I}'' \subset \mathcal{I} \ominus \mathcal{I}'$, that yields $\mathcal{I}' \in U^{\mathcal{I}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F})$. \square

Lemma 4.4 (Completeness). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite_{FR} knowledge base, \mathcal{P} a prioritization over \mathcal{K} , \mathcal{F} a finite set of DL-Lite_{FR} TBox assertions such that $\text{Mod}(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$, \mathcal{F} satisfies the assertions 4.1 and 4.2, and \mathcal{K}' the DL-Lite_{FRS} knowledge base such that $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$. Then, for every model $\mathcal{I} \in \text{Mod}(\mathcal{K})$:*

$$U^{\mathcal{I}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F}) \subseteq \text{Mod}(\mathcal{K}'),$$

where $(\mathcal{T}', \mathcal{A}') = \text{ComputeTBoxUpdate}_{\text{FRS}}(\mathcal{T}, \mathcal{A}, \mathcal{F}, \mathcal{P})$.

Proof. Suppose that there exists an interpretation $\mathcal{I}'' \in U^{\mathcal{I}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F}) \setminus \text{Mod}(\mathcal{K}')$. Therefore, $\mathcal{I}'' \notin \text{Mod}(\mathcal{T}') \cap \text{Mod}(\mathcal{A}')$. Then \mathcal{I}'' does not satisfy some assertion F occurring in \mathcal{K}' .

$\text{Mod}(\mathcal{T}')$. Let us suppose, that $F \in \mathcal{T}' \setminus \mathcal{F}$. Note that \mathcal{T}' includes the following membership assertions: (i) assertions that are implied by \mathcal{K} and do not contradict \mathcal{F} (contradicting ones were added to \mathcal{T}^- and removed); (ii) assertions belonging to $cl(\mathcal{F})$; (iii) assertions that are implied by \mathcal{F} and the assertions of type (i), in other words, belonging to $cl(\mathcal{T}_m \cup \mathcal{F})$ (see definition 4.5). The assertions of the latter type were added to \mathcal{T}' in the lines 76, 78, 85, 87.

Thus, if \mathcal{I}'' does not satisfy assertion of type (i), it is not a model of \mathcal{T}_m . If \mathcal{I}'' does not satisfy assertion of type (ii), it is not a model of \mathcal{F} . Finally, if \mathcal{I}'' does

not satisfy assertion of type (iii), it is not a model of $\mathcal{T}_m \cup \mathcal{F}$. All three cases lead to $\mathcal{I}'' \notin U^{\mathcal{T}_{pr} \cdot \mathcal{P}}(\mathcal{I}, \mathcal{F})$, yielding contradiction.

$Mod(\mathcal{A}')$. As it has been shown, $\mathcal{I}'' \in Mod(\mathcal{T}')$. Now, let us suppose that $\mathcal{I} \notin Mod(\mathcal{A}')$, i.e. $F \in \mathcal{A}'$. Note that every assertion in \mathcal{A}' either is in \mathcal{A} , or is not in \mathcal{A} but implied by \mathcal{K} (added to \mathcal{A}' at the line 97, or 100, or 105, or 107, or 109). Moreover, by construction of \mathcal{A}' , all its assertions do not contradict \mathcal{F} with respect to prioritization \mathcal{P} . Hence, according to definition 4.5 clause 3, \mathcal{I}'' satisfies F . Thus, we have a contradiction, and $\mathcal{I}'' \in Mod(\mathcal{K}')$. \square

From lemmas 4.1, 4.2, 4.3, and 4.4 the following theorem is got directly.

Theorem 4.5. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{FR}}$ knowledge base, \mathcal{P} a prioritization over \mathcal{K} , \mathcal{F} a finite set of DL-Lite $_{\mathcal{FR}}$ TBox assertions such that $Mod(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$, \mathcal{F} satisfies the assertions 4.1 and 4.2. Then, the algorithm *ComputeTBoxUpdate $_{\mathcal{FRS}}$* ($\mathcal{T}, \mathcal{A}, \mathcal{F}, \mathcal{P}$) returns *ERROR* if $Mod(\mathcal{T}_{pr}) \cap Mod(\mathcal{F}) = \emptyset$, or returns $(\mathcal{T}', \mathcal{A}')$, otherwise, such that $\mathcal{K} \circ_{\mathcal{T}_{pr} \cdot \mathcal{P}} \mathcal{F} \equiv Mod(\mathcal{K}')$, where $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$.*

Now we turn to computational complexity.

Theorem 4.6. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite $_{\mathcal{FR}}$ knowledge base, \mathcal{P} a prioritization over \mathcal{K} , \mathcal{F} a finite set of DL-Lite $_{\mathcal{FR}}$ TBox assertions such that $Mod(\mathcal{T}_{pr} \cup \mathcal{F}) \neq \emptyset$, \mathcal{F} satisfies the assertions 4.1 and 4.2, and \mathcal{K}' the DL-Lite $_{\mathcal{FRS}}$ knowledge base such that $\mathcal{K}' = \langle \mathcal{T}', \mathcal{A}' \rangle$. Then:*

- the size of \mathcal{T}' is polynomially bounded by the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$;
- computing \mathcal{T}' can be done in polynomial time in the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$.
- the size of \mathcal{A}' is polynomially bounded by the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$;
- computing \mathcal{A}' can be done in polynomial time in the size of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$;

where $(\mathcal{T}', \mathcal{A}') = \text{ComputeTBoxUpdate}_{\mathcal{FRS}}(\mathcal{T}, \mathcal{A}, \mathcal{F}, \mathcal{P})$

Proof. The theorem is proved by the following observations:

1. The function *Saturate* runs in polynomial time in the size of \mathcal{T} .
2. *Saturate* is called a finite number of times by *ComputeTBoxUpdate $_{\mathcal{FRS}}$* : the number of calling at each of lines 32, 60, 65, and 68 no more then cardinality of \mathcal{A} .
3. For each $f \in \text{Saturate}$, the checking whether $\mathcal{K} \models f$ is **LogSpace** in \mathcal{A} .
4. For each $F \in cl(\mathcal{T})$, the checking whether $\mathcal{K} \models F$ is **LogSpace** in \mathcal{A} .
5. For each $f' \in \mathcal{A}^-$ the cost of eliminating f' from \mathcal{A}' is polynomial in the size of \mathcal{A} .

ALGORITHM $ComputeTBoxUpdate_{FR}^{app}(\mathcal{T}, \mathcal{A}, \mathcal{F}, \mathcal{P})$

INPUT: finite set of TBox assertions \mathcal{F} , DL-Lite_{FS} KB $\langle \mathcal{T}, \mathcal{A} \rangle$, a prioritization \mathcal{P} over \mathcal{K}

OUTPUT: a TBox \mathcal{T}^{app} and an ABox \mathcal{A}^{app} , or ERROR

- [1] **if** $\langle \mathcal{T}_{pr}, \mathcal{F} \rangle$ is not satisfiable **then return** ERROR
- [2] **else**
- [3] $(\mathcal{T}^{app}, \mathcal{A}^{app}) := ComputeTBoxUpdate_{FRS}(\mathcal{T}, \mathcal{A}, \mathcal{F}, \mathcal{P});$
- [4] **for each** $f \in \mathcal{A}^{app}$ **do**
- [5] **if** f is not DL-Lite_{FR} membership assertion **then** $\mathcal{A}^{app} := \mathcal{A}^{app} \setminus \{f\};$
- [6] **return** $(\mathcal{T}^{app}, \mathcal{A}^{app});$

Fig. 4.4: $ComputeTBoxUpdate_{FR}^{app}$

6. For each $F' \in \mathcal{T}^-$ the cost of eliminating f' from \mathcal{T}' is polynomial in the size of \mathcal{T} .

□

4.2.2 TBox Approximate Update: from DL-Lite_{FRS} to DL-Lite_{FR}

The final section is dedicated to the approximate update of DL-Lite_{FR} KB. First of all, note that the lemma 3.18 is still valid, since each DL-Lite_{FR} KB is a DL-Lite_A KB.

So, we can prove correctness of $ComputeTBoxUpdate_{FR}^{app}(\mathcal{T}, \mathcal{A}, \mathcal{F}, \mathcal{P})$ algorithm.

Theorem 4.7. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable DL-Lite_{FR} knowledge base, \mathcal{P} a prioritization over \mathcal{K} , \mathcal{F} a finite set of DL-Lite_A TBox assertions. Then, the algorithm $ComputeTBoxUpdate_{FR}^{app}$ returns ERROR if $Mod(\mathcal{T}_{pr}) \cap Mod(\mathcal{F}) = \emptyset$, or returns DL-Lite_{FR} TBox \mathcal{T}^{app} and ABox \mathcal{A}^{app} , otherwise, such that \mathcal{K}^{app} is a maximal (DL-Lite_{FR}, \mathcal{T}')-approximation of $\mathcal{K} \circ_{(\mathcal{T}_{pr}, \mathcal{P})} \mathcal{F}$, where $\mathcal{K}^{app} = \langle \mathcal{T}^{app}, \mathcal{A}^{app} \rangle$, \mathcal{T}' is updated TBox.*

Proof. The proof of this theorem is absolutely analogous to the proof of the theorem 3.19. Note that if in the case of instance update a TBox does not change and it is the same in an original DL-Lite_A KB, in its DL-Lite_{AS} update, in its approximate update, and as a set of protected formulas. But in this case we should take into account that we compute approximation for DL-Lite_{FRS} update of an initial KB. Thus, if \mathcal{T} concerns the result of $ComputeTBoxUpdate_{FRS}$ algorithm, it should be \mathcal{T}' . The theorem which is referred in the proof is theorem 4.5.

□

5. CONCLUSION

In this work, we have studied the problem of updating knowledge bases described in DL-Lite languages. We have considered updates for both the instance-level and the intensional-level.

More specifically, in the first part, we have focused on updating ABoxes with new membership assertions. Our update approach is based on the Possible Model Approach proposed in [Win90], and then adapted to the DL case in [Pog06]. In the thesis we extended the results of [Pog06] for DL-Lite \mathcal{F} by considering DL-Lite \mathcal{R} and the more expressive logic DL-Lite \mathcal{A} . Our main contribution of the first part is showing that both languages have good closure characteristic concerning updates. That is the fact that though the result of an update is in general not expressible in the language of the original KB (DL-Lite \mathcal{R} or DL-Lite \mathcal{A}), it is always expressible within DL-Lite itself. We have defined a subfamily of the DL-Lite languages, DL-Lite \mathcal{S} , which is a minimal extension of the “classic” DL-Lite languages closed under update. This subfamily allows for looser membership assertions in the ABox, notably, $C(x)$ and $R(a, b)$, where C is a general concept, R is a general role, and x is a so-called soft constant, which is in essence an existentially quantified variable.

Then, we have focused on updates for the logics DL-Lite \mathcal{AS} and DL-Lite \mathcal{A} . Our main contribution in this part is that we introduced an algorithm that takes a DL-Lite \mathcal{AS} knowledge base \mathcal{K} as input and returns a DL-Lite \mathcal{AS} ABox that corresponds to the update. Since every DL-Lite \mathcal{A} KB is also a DL-Lite \mathcal{AS} one, we can apply the algorithm to such a KB and get its update which in general is in DL-Lite \mathcal{AS} . Since it is natural to have updates expressed in the original language, we have considered the notion of approximate update. That is, having a DL-Lite \mathcal{A} KB, we can get another one which is also described in DL-Lite \mathcal{A} and is as close to the update as possible, using a suitable notion of distance between KBs. A corresponding approximation algorithm has been presented and this is our second contribution of this part of the thesis. Our next contribution is an algorithm to obtain erasure in DL-Lite \mathcal{AS} . An important characteristic of erasure is that deleting information from a KB we do not prohibit it, but we make it uncertain, as opposed to updating with the negation of the fact that just forbids it. Our algorithm can handle this notion of erasure. We notice that the two latter algorithms are based on the one for updating DL-Lite \mathcal{A} , hence, they inherit its tractable computational properties: they can be run in time polynomial in the size of the KB.

In the last part, we have examined intensional level update. Since a DL-Lite TBox usually implies more assertions than it contains explicitly, the definition that has been adapted for the instance level case is no more applicable. TBox update under the instance level definition contradicts intuition. Another problem that has arisen is that changes in the TBox have an effect on the extensional level, and they may cause appearing disjunctive information in the KB. Since we would like to stay within DL-Lite with its tractable computational properties, and disjunctions increase complexity, we have proposed to solve this problem by putting prioritization over KB predicates. The main contribution of this part of the thesis is an algorithm that takes a DL-Lite \mathcal{FR} knowledge base and returns its TBox update, but expressed in DL-Lite \mathcal{FRS} . To get back to DL-Lite \mathcal{FR} , we proposed an approximation of the update. The latter algorithm can run in polynomial time in the size of the KB.

We see several interesting directions for continuing our research. The first one is to implement the algorithms presented in this thesis. Related to this point, the algorithms must be studied in the sense of optimization techniques to deal with ontologies that include very large ABoxes. Particularly, it would be interesting to avoid chasing of the ABox and TBox data with TBox ontologies at run time of our update algorithms. Instead it is desirable to “push” updates and erasures into ABoxes stored under relational DBMS by using techniques analogous to query rewriting.

Second, a classical model-based approach has been adopted to update and erasure. Other approaches to update and erasure might be interesting, among them, approaches based on belief revision.

Finally, prioritization proposed in Chapter 4 could be improved. In the current algorithm, the order over predicates is up to the user. It would, for example, be of interest to consider building an order on the fly according to the number of TBox assertions a predicate takes part in.

BIBLIOGRAPHY

- [ABHM03] Carlos Areces, Patrick Blackburn, Bernadette Martinez Hernandez, and Maarten Marx. Handling boolean aboxes. In *In Proc. of the 2003 Description Logic Workshop (DL 2003) (2003)*, CEUR (<http://ceur-ws.org>), 2003.
- [ACKZ09] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyashev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36, 2009.
- [AG85] Serge Abiteboul and Gösta Grahne. Update semantics for incomplete databases. In *VLDB '1985: Proceedings of the 11th international conference on Very Large Data Bases*, pages 1–12. VLDB Endowment, 1985.
- [BCM⁺03] 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.
- [BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.
- [CDGL⁺06] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. Linking data to ontologies: The description logic *DL-Lite_A*. In *Proc. of the 2nd Int. Workshop on OWL: Experiences and Directions (OWLED 2006)*, volume 216 of *CEUR Electronic Workshop Proceedings*, <http://ceur-ws.org/>, 2006.
- [CDGL⁺07] 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. *J. of Automated Reasoning*, 39(3):385–429, 2007.
- [CDGL⁺09] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. Ontologies and databases: The dl-lite approach. In Sergio Tessaris and Enrico Franconi, editors, *Semantic Technologies for Informations Systems - 5th Int. Reasoning Web Summer School (RW 2009)*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer, 2009.

- [DGLPR06] Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. On the update of description logic ontologies at the instance level. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*, pages 1271–1276, 2006.
- [DGLPR07] Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. On the approximation of instance level update and erasure in description logics. In *Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007)*, pages 403–408, 2007.
- [DGLPR09] Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. On instance-level update and erasure in description logic ontologies. *J. of Logic and Computation, Special Issue on Ontology Dynamics*, 2009.
- [DLB⁺09] Conrad Drescher, Hongkai Liu, Franz Baader, Steffen Guhle-
mann, Uwe Petersohn, Peter Steinke, and Michael Thielscher. Putting abox updates into action. LTCS-Report 09-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2009. See <http://lat.inf.tu-dresden.de/research/reports.html>.
- [DLNN97] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.
- [Gra89] G. Grahne. Horn tables—an efficient tool for handling incomplete information in databases. In *PODS '89: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 75–82, New York, NY, USA, 1989. ACM.
- [Ho01] Volker Haarslev and Ralf Möller. Racer system description. pages 701–705. Springer-Verlag, 2001.
- [Hor98] Ian Horrocks. The fact system, 1998.
- [LLMW06] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic ABoxes. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 46–56, 2006.
- [LYV⁺98] Chen Li, Ramana Yerneni, Vasilis Vassalos, Hector Garcia-Molina, Yannis Papakonstantinou, Jeffrey D. Ullman, and Murty Valiveti. Capability based mediation in TSIMMIS. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 564–566, 1998.
- [MLG87] David E. Smith Matthew L. Ginsberg. Reasoning about action i: A possible worlds approach. Technical Report KSL-86-65, Knowledge Systems, AI Laboratory, 1987. 25 pages.
- [Pog06] Antonella Poggi. *Structured and Semi-Structured Data Integration*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”, 2006.

-
- [Win88a] Marianne Winslett. A model-based approach to updating databases with incomplete information. *ACM Trans. Database Syst.*, 13(2):167–196, 1988.
- [Win88b] Marianne Winslett. Reasoning about action using a possible models approach. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, 1988.
- [Win90] Marianne Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.
- [ZF96] Yan Zhang and Norman Y. Foo. Updating knowledge bases with disjunctive information. In *Proceedings of Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 562–568. AAAI/MIT Press, 1996.