http://webdam.inria.fr/

# Web Data Management

## Querying data through ontologies

Serge Abiteboul
INRIA Saclay & ENS Cachan

Ioana Manolescu
INRIA Saclay & Paris-Sud University

Philippe Rigaux
CNAM Paris & INRIA Saclay

Marie-Christine Rousset
Grenoble University

Pierre Senellart
Télécom ParisTech

*Copyright @2011 by Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart;*
*to be published by Cambridge University Press 2011. For personal use only, not for distribution.*

http://webdam.inria.fr/Jorge/

## Contents

.

## 1 Introduction

As we saw in the previous chapter, ontologies form the backbone of the Semantic Web by providing a conceptual view of data and services available worldwide via the Web. We discussed the RDF language for describing knowledge, and a family of languages, called description logics, that provide formal foundations for ontologies and in particular for the OWL ontology language recommended by the W3C.

In this chapter, the focus is on querying RDF data. Since massive volumes of RDF sources are more and more present on the Web, specifying queries for RDF data that can be evaluated efficiently is becoming every day more and more important.

We will see that the set of query answers strongly depends on the semantic context. We will study query answering when the ontology is specified in RDFS and then when the ontology is specified in DL-LITE. The ontology language DL-LITE belongs to the DL family. It has been designed as a trade-off between the ability to describe a wide range of domains of interest and query evaluation efficiency. More precisely, we will focus on two important fragments of DL-LITE, namely DL-LITE$_\mathcal{R}$ and DL-LITE$_\mathcal{F}$.

We will observe that the problem of answering queries through ontologies is quite different from that of answering database queries for the following two essential reasons:

**Implicit facts.** In a DBMS setting, all the facts are explicit. For instance, the constraint "Every PhD student is a student" enforces that, before inserting a value $v$ in the *PhDStudent* table, this value is also inserted in the *Student* table (if not already there). In an ontology context, someone may be a student not explicitly but because of the constraint used as an inference rule. In addition, the implicit facts may be incompletely known, coming from constraints such as "A professor teaches at least one master course". From a fact such as $Professor(dupond)$, one can infer the two facts $Teaches(dupond, x)$ and $MasterCourse(x)$ for some unknown value $x$. Such partially known implicit facts may however be useful for answering queries such as "Give me all the persons who teach a master course".

**Inconsistency.** In a DBMS setting, the constraint "each course must have a single responsible" is also viewed as a law that cannot be violated. An update of the corresponding table that would violate this law would simply be rejected. In an ontology context, such local verifications are not sufficient for checking data inconsistency. Because of data incompleteness, this may require intricate reasoning on different constraints and data distributed over different tables. For instance, if in addition of the previous key constraint, it is declared that "only professors can be responsible of courses in which they must teach", "a master course is taught by a single teacher", and "lecturers are not professors", the presence of the three following facts in different tables of the database makes it inconsistent: $Lecturer(jim)$, $TeachesIn(jim, ue431)$ and $MasterCourse(ue431)$. The reason is that because of the constraint "only professors can be responsible of courses in which they must teach", we can infer that the course *ue431* must have a responsible $x$ who is unknown but for whom we have a partial information: s/he is a professor and s/he teaches in the course *ue431*. Without knowing $x$, the fact that she is a professor is sufficient to infer that $x \neq jim$ (since *jim* is a lecturer and thus not a professor). Therefore, the course *ue431* is taught by two distinct teachers, which is forbidden for a master course.

From this, it should be clear that query answering through ontologies is more complicated that in classical databases. We have to reason to find which inferences may participate in answering a given query. We also have to reason to verify the consistency of our knowledge.

## 2 Querying RDF data: notation and semantics

In this section, we set the stage for querying RDF data. We also discuss the impact of ontologies (knowledge on the domain of interest) on the answers. To simplify, we ignore here blank nodes.

Figure 1 is an enhanced version of the University example that we will use throughout this chapter. The first column provides RDF data in the triple syntax, while the second column shows the corresponding facts in FOL.

RDF triples can be asserted in a very flexible way and almost without constraints. The association of some ontology is not a requirement. Users can update a collection of RDF statements freely by just adding/removing triples. The only reserved word in the RDF vocabulary is `rdf:type` that is used to relate constant names to types, i.e., classes in domain of interest or unary predicates in FOL world.

Let us now consider querying a set of RDF facts, for which the query language SPARQL has been proposed. We briefly consider it next. query language. We briefly consider it next.

SPARQL (pronounced "sparkle") is a recursive acronym standing for *S*PARQL *P*rotocol *A*nd *R*DF *Q*uery *L*anguage. It is a W3C recommendation as of 2008. Although it does borrow some features from XQuery (functions and operators), it is based on the graph model underlying RDF data.

For instance, the following query expresses in SPARQL the search of all the individuals who are enrolled in a department led by a Professor.

```
select x where x EnrolledIn y, z Leads y, z rdf:type Professor
```

| **Subject** | **Predicate** | **Object** | **FOL semantics** |
|---|---|---|---|
| dupond | Leads | infoDept | Leads(dupond,infoDept) |
| dupond | `rdf:type` | Professor | Professor(dupond) |
| durand | ResponsibleOf | ue111 | ResponsibleOf(durand,ue111) |
| durand | Leads | csDept | Leads(durand,csDept) |
| paul | TeachesTo | pierre | TeachesTo(paul,pierre) |
| paul | `rdf:type` | PhdStudent | PhDStudent(paul) |
| paul | EnrolledIn | infoDept | EnrolledIn(paul, infodept) |
| pierre | EnrolledIn | infoDept | EnrolledIn(pierre, infodept) |
| pierre | `rdf:type` | Undergrad | Undergrad(pierre) |
| pierre | RegisteredTo | ue111 | Registered(pierre, ue111) |
| ue111 | OfferedBy | infoDept | OfferedBy(ue111,infoDept) |
| ue111 | `rdf:type` | CSCourse | CSCourse(ue111) |
| jim | EnrolledIn | csDept | EnrolledIn(jim, csDept) |
| csDept | `rdf:type` | TeachingDept | TeachingDept(csDept) |

Figure 1: RDF triple syntax and its FOL semantics

We used here an SQL-like syntax. There exists competing syntaxes for expressing SPARQL queries. The corresponding query in FOL notation is:

$$q(x) \ :- \ \exists y \exists z [\mathit{EnrolledIn}(x,y) \wedge \mathit{Leads}(z,y) \wedge \mathit{Professor}(z)]$$

This is a *conjunctive query*, i.e., a FOL formula without negation or disjunction, of the form

$$q(x_1,...,x_m) \ :- \ \exists y_1,...,y_n [R_1(u_1) \wedge ... \wedge R_p(u_p)]$$

where each $u_i$ is a vector of variables in $\{x_1,...,x_m,y_1,...,y_n\}$ or constants, and each variable $x_i$ appears in the body of the query (i.e., for each $x \in \{x_1,...,x_m\}$, there exists $u_i$ such that $x \in u_i$).

In the remainder of this chapter, we use conjunctive queries as the query language for RDF. From the example, it should be clear that all we say is relevant to SPARQL. We use a (standard) simplified notation for conjunctive queries. We omit the existential quantifiers and denote the connector $\wedge$ by a comma. Observe that this does not introduce any ambiguity. In particular, all variables not occurring in the "head of the query" (i.e., in $q(x_1,...,x_m)$) are understood as existentially quantified. The variables in $x_1,...,x_m$ are said to be *distinguished*.

In this simplified form, the example SPARQL query becomes:

```
q(x) :- EnrolledIn(x,y), Leads(z,y), Professor(z)
```

Now consider a query in the general form:

$$q(x_1,...,x_m) \ :- \ R_1(u_1),...,R_p(u_p)$$

with existential variables $y_1,...,y_n$. Following the standard FOL semantics, the evaluation of the query consists in finding valuations $\nu$ of the variables for which the closed fact $R_i(\nu(u_i))$ "holds" for each $i$. The corresponding answer is then $q(\nu(x_1),...,\nu(x_m))$. Equally, we may say that $(\nu(x_1),...,\nu(x_m))$ is an answer for the query $q$. When the query is unary (i.e., it has a single distinguished variable), we either say "$q(a)$ is an answer" or "$a$ is an answer for $q$".

An essential issue is in the meaning of "holds" in the previous informal definition.

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

**Inference with data only.** In the simplest case, a fact $R_i(\nu(u_i))$ holds if it is a known fact (explicitly stated in the data store). For instance, consider the previous conjunctive query in the University example. The evaluation of the query $q(x)$ against the facts of Figure 1 returns {*paul*, *pierre*} as its answer set. To see why *paul* is an answer, we just check that, by mapping the distinguished variable to the constant *paul*, and the existential variables $y$ and $z$ respectively to the constants *infoDept* and *dupond*, all the conjuncts in the query definition are satisfied by facts in the database. The same holds if the distinguished variable $x$ is instantiated with the constant *pierre*.

**More inference using an ontology.** Now, let us assume that a fact $R_i(\nu(u_i))$ holds if it is a known fact or if it is a consequence of the known facts by taking the ontological statements into account. Suppose now that we also have the knowledge that someone responsible for a class has to be a professor, that is in DL syntax:

$$\exists \mathit{ResponsibleOf} \sqsubseteq \mathit{Professor}.$$

Additional answers can then be inferred. For instance, for the query $q(x)$, the additional answer *jim* is obtained. It would come from the presence in the data of the facts *EnrolledIn(jim, csDept)*, *Leads(durand, csDept)*, and from the fact *Professor(durand)*, which, without being explicitly stated in the data, is logically entailed from the fact *ResponsibleOf(durand, ue111)* and $\exists \mathit{ResponsibleOf} \sqsubseteq \mathit{Professor}$.

To see why, we just have to consider the FOL semantics of this DL statement:

$$\forall x \forall y [ResponsibleOf(x,y) \Rightarrow Professor(x)].$$

This logical implication indeed allows inferring the fact *Professor(durand)* from the fact *ResponsibleOf(durand, ue111)*.

More subtly, we can get answers from *partially instantiated* facts that can be logically entailed by the knowledge base. Suppose that we know that a professor teaches at least one course, that is in DL syntax:

$$\mathit{Professor} \sqsubseteq \exists \mathit{TeachesIn}.$$

and consider the query $q(x) :- TeachesIn(x,y)$

From the explicit ground fact *Professor(durand)* and the contraint $\mathit{Professor} \sqsubseteq \exists \mathit{TeachesIn}$, we know that $TeachesIn(durand, v)$ holds for some unknown value $v$. The valuation of $v$ may vary in the different "worlds" satisfying the constraint. This is however sufficient to infer that answer $q(durand)$ is true in all these possible worlds.

**Formal definition of answer set.** Recall that $\varphi \models \psi$ (i.e., $\varphi$ *implies* $\psi$) if each interpretation making $\varphi$ true also makes $\psi$ true, or equivalently, every model of $\varphi$ is a model of $\psi$. We next provide a formal definition of the *answer set* of a query for a DL knowledge base, that captures the general setting where data (the Abox $\mathcal{A}$) is associated to an ontology (the Tbox $\mathcal{T}$) to form a DL *knowledge base* $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$. A *query* to the knowledge base is a conjunctive query using class or property predicates from the given knowledge base with the proper arity. (Class predicates have arity one and property predicates arity 2.)

A valuation $\nu$ of a set of variables $\{z_1, ..., z_p\}$ is a substitution (denoted $\{z_1/a_1, ..., z_p/a_p\}$) that assigns each variable $z_i$ to a constant $a_i$ (two distinct variables may be assigned to a same constant). Given two valuations $\nu$ and $\nu'$ of two disjoint sets of variables $\{z_1, ..., z_p\}$

and $\{v_1,...,v_k\}$ , $\nu \circ \nu'$ denotes the valuation assigning the variables $z_i$ to the corresponding constants in $\nu$, and the variables $v_j$ to the corresponding constants in $\nu'$.

We can now formally define the notion of answers.

**Definition 2.1** *Let* $q(x_1,...,x_m)\ :-\ R_1(u_1),...,R_p(u_p)$ *be a query to a knowledge base* $\mathcal{K}$. *An* answer *is a ground fact* $q(\nu(x_1),...,\nu(x_m))$ *for some valuation* $\nu$ *of the distinguished variables such that in every model of* $\mathcal{K}$ *there exists a valuation* $\nu'$ *of the existential variables for which* $R_i(\nu \circ \nu'(u_i))$ *is true for each* $i$. *The* answer set *of* $q$ *for* $\mathcal{K}$ *is the set of all such answers. It is denoted* $q(\mathcal{K})$.

Consider again the previous University query example. We have seen that its answer set varies depending on the knowledge base against which it is evaluated. In particular, if $\mathcal{A}$ is the set of facts of Figure 1 and $\mathcal{T}$ is $\{\exists ResponsibleOf \sqsubseteq Professor\}$, we have:

- $q(\mathcal{A}) = \{paul,pierre\}$.
- $q(\mathcal{A} \cup \mathcal{T}) = \{paul,pierre,jim\}$.

**Boolean queries.** To conclude this section, we consider a particular interesting case of queries, that of Boolean queries. The *arity* of a query is the number of its distinguished variables. A query of arity *0*, i.e., a query of the form $q()\ :\ ...$, is called a *Boolean query*. Note that there is a single possible answer, namely $q()$. In this case, we see that as a positive answer to the query, i.e., as *true*. If the answer set is empty, we see that as *false*.

To see an example, consider the query

```
q'() :- Student(x), TeachesTo(x,y)
```

This Boolean query asks whether there exists a student teaching to other students. Suppose $\mathcal{T}' = \{PhDStudent \sqsubseteq Student\}$. Then we can distinguish two cases:

- $q'(\mathcal{A}) = \emptyset$ and the answer is no.
- $q'(\mathcal{A},\mathcal{T}') = \{q'()\}$ and the answer is yes.

In the second case, the fact $Student(paul)$, although not in the Abox $\mathcal{A}$, can be inferred from the fact $PhDStudent(paul)$ in $\mathcal{A}$ and the inclusion statement $PhDStudent \sqsubseteq Student$ in $\mathcal{T}'$. Together with the fact $TeachesTo(paul,pierre)$ present in the Abox, it makes the body of the query $q'$ satisfied.

## 3 Querying through RDFS ontologies

In this section, we consider RDF data without blank nodes (that can be seen as an Abox), associated to an RDFS ontology (that can be seen as a very simple Tbox). RDF data and RDFS statements can be denoted and stored as triples. However, the important point is that RDFS statements have a logical semantics which can be operationalized as a set of inference rules (see Section **??** in Chapter **??**). We illustrate here how this can be used to answer queries.

Figure 2 is an example of an RDFS ontology that can be associated to the RDF data in Figure 1. The RDFS statements composing the ontology are given in three notations: the triple notation, the DL notation, and the corresponding FOL notation.

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

| **RDFS notation** | **DL notation** | **FOL notation** |
|---|---|---|
| ⟨*AcademicStaff* `rdfs:subClassOf` *Staff*⟩ | $AcademicStaff \sqsubseteq Staff$ | $AcademicStaff(X) \Rightarrow Staff(X)$ |
| ⟨*Professor* `rdfs:subClassOf` *AcademicStaff*⟩ | $Professor \sqsubseteq AcademicStaff$ | $Professor(X) \Rightarrow AcademicStaff(X)$ |
| ⟨*Lecturer* `rdfs:subClassOf` *AcademicStaff*⟩ | $Lecturer \sqsubseteq AcademicStaff$ | $Lecturer(X) \Rightarrow AcademicStaff(X)$ |
| ⟨*PhDStudent* `rdfs:subClassOf` *Lecturer*⟩ | $PhDStudent \sqsubseteq Lecturer$ | $PhDStudent(X) \Rightarrow Lecturer(X)$ |
| ⟨*PhDStudent* `rdfs:subClassOf` *Student*⟩ | $PhDStudent \sqsubseteq Student$ | $PhDStudent(X) \Rightarrow Student(X)$ |
| ⟨*TeachesIn* `rdfs:domain` *AcademicStaff*⟩ | $\exists TeachesIn \sqsubseteq AcademicStaff$ | $TeachesIn(X,Y) \Rightarrow AcademicStaff(X)$ |
| ⟨*TeachesIn* `rdfs:range` *Course*⟩ | $\exists TeachesIn^{-} \sqsubseteq Course$ | $TeachesIn(X,Y) \Rightarrow Course(Y)$ |
| ⟨*ResponsibleOf* `rdfs:domain` *Professor*⟩ | $\exists ResponsibleOf \sqsubseteq Professor$ | $ResponsibleOf(X,Y) \Rightarrow Professor(X)$ |
| ⟨*ResponsibleOf* `rdfs:range` *Course*⟩ | $\exists ResponsibleOf^{-} \sqsubseteq Course$ | $ResponsibleOf(X,Y) \Rightarrow Course(Y)$ |
| ⟨*TeachesTo* `rdfs:domain` *AcademicStaff*⟩ | $\exists TeachesTo \sqsubseteq AcademicStaff$ | $TeachesTo(X,Y) \Rightarrow AcademicStaff(X)$ |
| ⟨*TeachesTo* `rdfs:range` *Student*⟩ | $\exists TeachesTo^{-} \sqsubseteq Student$ | $TeachesTo(X,Y) \Rightarrow Student(Y)$ |
| ⟨*Leads* `rdfs:domain` *AdminStaff*⟩ | $\exists Leads \sqsubseteq AdminStaff$ | $Leads(X,Y) \Rightarrow AdminStaff(X)$ |
| ⟨*Leads* `rdfs:range` *Dept*⟩ | $\exists Leads^{-} \sqsubseteq Dept$ | $Leads(X,Y) \Rightarrow Dept(Y)$ |
| ⟨*RegisteredIn* `rdfs:domain` *Student*⟩ | $\exists RegisteredIn \sqsubseteq Student$ | $RegisteredIn(X,Y) \Rightarrow Student(X)$ |
| ⟨*RegisteredIn* `rdfs:range` *Course*⟩ | $\exists RegisteredIn^{-} \sqsubseteq Course$ | $RegisteredIn(X,Y) \Rightarrow Course(Y)$ |
| ⟨*ResponsibleOf* `rdfs:subPropertyOf` *TeachesIn*⟩ | $ResponsibleOf \sqsubseteq TeachesIn$ | $ResponsibleOf(X,Y) \Rightarrow TeachesIn(X,Y)$ |

Figure 2: An RDFS ontology expressed in different notations

As already said, these RDFS statements can be used to infer new triples (i.e., new facts) from the RDF database. For example, the RDF triple ⟨*durand ResponsibleOf ue111*⟩ in Figure 1 corresponds to the fact $ResponsibleOf(durand, ue111)$, and the RDFS statement ⟨*ResponsibleOf* `rdfs:domain` *Professor*⟩ corresponds to the logical rule: $ResponsibleOf(X,Y) \Rightarrow Professor(X)$. The condition of this rule can be mapped with the fact $ResponsibleOf(durand, ue111)$ by the substitution {X/durand, Y/ue111}, and thus the corresponding instantiation of the conclusion $Professor(durand)$ can be inferred. This new fact can in turn trigger a rule such as $Professor(X) \Rightarrow AcademicStaff(X)$, thereby allowing the inference of additional facts such as $AcademicStaff(durand)$.

More generally, RDFS statements correspond to rules that can be applied in a forward-chaining manner to the initial set of facts until saturation, i.e., until no more fact can be inferred. It is important to see that the variables in the head of rule all occur in the body. In other words, no variable is quantified existentially. So rules always infer new ground facts. Such rules are said to be *safe*. We will use unsafe rules when we consider DL-LITE, which will render query processing more complicated.

The simple forward-chaining Algorithm 1 starts with the set of initial facts and repeats inference steps until saturation.

Figure 3 shows the facts resulting from the application of Algorithm 1 to the facts of Figure 1 and the rules of Figure 2.

To answer queries from RDF facts associated to an RDFS ontology, one can proceed as follows. First one compute all the inferred facts (in a bottom-up manner) with the previous algorithm. Each step of the loop can be computed, for instance, using a standard relational query engine. This yields a new database consisting of the set of all the facts (asserted or inferred). Then one can evaluate the query directly on that database using a standard relational query engine.

For example, the standard evaluation against the set of (asserted + inferred) facts in Figure 3 of the query

$$q(x) \; :- \; Enrolled(x,y), Leads(z,y), Professor(z)$$

```
Algorithm 1: The Saturation algorithm
Saturation($\mathcal{A}$,$\mathcal{T}$)
Input: An Abox $\mathcal{A}$ and an RDFS Tbox $\mathcal{T}$
Output: The set of facts that are inferred: $\Delta_0$
(1)     $F \leftarrow \mathcal{A}$
(2)     $\Delta_0 \leftarrow \mathcal{A}$
(3)     repeat $\Delta_1 \leftarrow \emptyset$
(4)         foreach rule $condition \Rightarrow conclusion$ in $\mathcal{T}$,
(5)             if there exists a substitution $\sigma$ such that $\sigma.condition \in \Delta_0$
(6)             and $\sigma.conclusion \notin F$
(7)                 add $\sigma.conclusion$ to $\Delta_1$
(8)         $F \leftarrow F \cup \Delta_1$
(9)         $\Delta_0 \leftarrow \Delta_1$
(10)    until $\Delta_1 = \emptyset$
```

(searching for individuals enrolled in a department led by a Professor) returns $\{paul, pierre, jim\}$ as its answer set. If we evaluate the same query against the set of asserted facts only, we do not find the answer *jim*.

**Complexity analysis.** It is interesting to estimate both the maximum number of inferred triples and the worst-case time complexity for inferring them. Of course, this depends on the number of asserted triples (i.e., the size of the data) and also on the number of axioms in the ontology (i.e., the size of the ontology).

Let $M$ be the number of facts in the Abox and $N$ the number of axioms in the Tbox. From the presence of some initial fact $C(a)$, one can derive a number of new facts $C'(a)$ for some class $C'$. Note that the number of such $C'(a)$ is bounded by the number of axioms in the ontology, i.e., it is less than $N$. Now consider some initial fact $R(a,b)$. From it, one can derive some facts $R'(a,b)$ or $R'(b,a)$ as well as some facts $C'(a)$ or $C'(b)$ for some $R'$ and $C'$. Again, one can observe that for a particular $R(a,b)$, the number of new facts one can derive is bounded by the number of axioms in the ontology, i.e., it is less than $N$. Since the number of initial facts is $M$, the number of facts one can derive is bounded by $M \times N$. Observe in particular that it is linear in the number of database facts.

Now consider the worst-case time complexity for inferring them by the Algorithm 1. We have to perform at most $M \times N$ iterations. Each iteration can be performed in polynomial time. So the algorithm is in PTIME. One can show more precisely that it is in $0((M \times N)^2)$.

## 4 Answering queries through DL-LITE ontologies

In this section, we consider two important fragments of the DL-LITE ontology language of the DL family. As we will see, querying is feasible for these two languages even though they provide a quite rich framework for describing semantics. We study querying through ontologies expressed in these two fragments.

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

| **Asserted facts** | **Inferred facts** |
|---|---|
| Leads(dupond,infoDept) | AdminStaff(dupond) |
| Professor(dupond) | Dept(infoDept) |
| ResponsibleOf(durand,ue111) | AcademicStaff(dupond) |
| Leads(durand,csDept) | Professor(durand) |
| TeachesTo(paul,pierre) | Course(ue111) |
| PhDStudent(paul) | AcademicStaff(durand) |
| EnrolledIn(paul, infodept) | AdminStaff(durand) |
| EnrolledIn(pierre, infodept) | Dept(csDept) |
| Undergrad(pierre) | AcademicStaff(paul) |
| Registered(pierre, ue111) | Student(pierre) |
| OfferedBy(ue111,infoDept) | Student(paul) |
| CSCourse(ue111) | Student(pierre) |
| EnrolledIn(jim, csDept) | Lecturer(paul) |
| TeachingDept(csDept) | AcademicStaff(paul) |
| | Staff(paul) |
| | Staff(dupond) |
| | Staff(durand) |

Figure 3: Inferred facts from RDF facts and an associated RDFS ontology

## 4.1 DL-LITE

A DL-LITE ontology may contain axioms corresponding (up to the syntax) to those allowed in an RDFS ontology. Besides, it may contain other axioms, of three kinds: positive inclusions (PI for short), negative inclusions (NI) and key constraints (Key). Figure 4 shows examples of these three kinds of DL-LITE axioms with their corresponding FOL semantics. These constraints are not expressible in RDFS.

| | **DL notation** | **Corresponding logical rule** |
|---|---|---|
| **PI** | $Professor \sqsubseteq \exists TeachesIn$ | $Professor(X) \Rightarrow \exists Y TeachesIn(X,Y)$ |
| | $Course \sqsubseteq \exists RegisteredIn^-$ | $Course(X) \Rightarrow \exists Y RegisteredIn(Y,X)$ |
| **NI** | $Student \sqsubseteq \neg Staff$ | $Student(X) \Rightarrow \neg Staff(X)$ |
| **Key** | $(funct\ ResponsibleOf^-)$ | $ResponsibleOf(Y,X) \wedge ResponsibleOf(Z,X) \Rightarrow Y = Z$ |

Figure 4: Examples of DL-LITE axioms not expressible in RDFS

We next consider in turn these new kinds of axioms.

**Positive inclusion and incompleteness.** A *positive* inclusion axiom is an expression of one of the following forms:

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

| **DL notation** | **Corresponding logical rule** |
|---|---|
| $B \sqsubseteq \exists P$ | $B(X) \Rightarrow \exists Y P(X,Y)$ |
| $\exists Q \sqsubseteq \exists P$ | $Q(X,Y) \Rightarrow \exists Z P(X,Z)$ |
| $B \sqsubseteq \exists P^-$ | $B(X) \Rightarrow \exists Y P(Y,X)$ |
| $\exists Q \sqsubseteq \exists P^-$ | $Q(X,Y) \Rightarrow \exists Z P(Z,X)$ |
| $P \sqsubseteq Q^-$ or $P^- \sqsubseteq Q$ | $P(X,Y) \Rightarrow Q$(Y,X) |

where $P$ and $Q$ denote properties and $B$ denotes a class. Recall that $P^-$ denotes the *inverse* of $P$, i.e., $P^-(x,y)$ iff $P(y,x)$ for all $x,y$.

Observe that expressions of the form $\exists P \sqsubseteq B$ belong to DL-LITE since they already are in RDFS. Expressions of the form $P \sqsubseteq Q$ (so equivalently $P^- \sqsubseteq Q^-$) also belong to DL-LITE for the same reason.

It is important to note that the logical rules corresponding to PI axioms expressible in DL-LITE are *not* necessarily safe (as opposed to RDFS that uses only safe rules.) Consider the rule

$$\forall X(Professor(X) \Rightarrow \exists Y(TeachesIn(X,Y)))$$

The variable $Y$ is existentially quantified. As already mentioned, the main issue is that, as a consequence, such an axiom does not produce new facts (i.e., ground atoms) from initial facts, but only an *incomplete* information in the form of atoms that may be partially instantiated. For example, from the fact $Professor(durand)$, the previous axiom permits to infer that there exists some course(s) $y$ that *durand* teaches. In other words, we know that there exists some fact of the form $TeachesIn(durand,y)$ that is true but we do not know the value of $y$. This makes it difficult to apply the bottom-up approach described in Section 3. Such an approach is not appropriate for answering queries through DL-LITE ontologies.

**Negative inclusion and inconsistencies.** A *negative* inclusion axioms is an expression that takes one of the forms:

| **DL notation** |
|---|
| $B_1 \sqsubseteq \neg B_2$ |
| $R_1 \sqsubseteq \neg R_2$ |

where

- $B_1$ and $B_2$ are either classes or expressions of the form $\exists P$ or $\exists P^-$ for some property $P$
- where $R_1$ and $R_2$ are either properties or inverses of properties.

The corresponding logic rules are left as an exercise. An example of NI (expressing the constraint "Students do not teach courses") and the corresponding logical rule are as follows:

| **DL notation** | **Corresponding logical rule** |
|---|---|
| $Student \sqsubseteq \neg\exists TeachesIn$ | $Student(X) \Rightarrow \neg\exists Y TeachesIn(X,Y)$ or equivalently, $\exists Y TeachesIn(X,Y) \Rightarrow \neg Student(X)$ |

NIs express disjointness constraints between classes or between properties, and thus introduce negation in the language. Therefore, the knowledge base against which the queries have to be evaluated may be *inconsistent*, i.e., a model of the corresponding theory may not

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

exist. Note that this is not possible with RDFS ontologies: we showed an algorithm that computed a model (indeed the smallest model).

For example, adding the NI $\mathit{Student} \sqsubseteq \neg \mathit{Staff}$ to the ontology of Figure 2 leads to the inconsistency of the knowledge base made of the facts in Figure 1 and of the axioms in the ontology of Figure 2 enriched with that NI. The reason is that from the fact $\mathit{PhDStudent}(\mathit{paul})$, and the inclusion axiom $\mathit{PhdStudent} \sqsubseteq \mathit{Student}$, we can infer the fact $\mathit{Student}(\mathit{paul})$, and in turn the literal $\neg \mathit{Staff}(\mathit{paul})$ from the NI $\mathit{Student} \sqsubseteq \neg \mathit{Staff}$. On the other hand, the fact $\mathit{Staff}(\mathit{paul})$ can be inferred from the fact $\mathit{PhDStudent}(\mathit{paul})$ and the inclusion axioms $\mathit{PhdStudent} \sqsubseteq \mathit{Lecturer}$, $\mathit{Lecturer} \sqsubseteq \mathit{AcademicStaff}$ and $\mathit{AcademicStaff} \sqsubseteq \mathit{Staff}$.

**Key constraints and more inconsistencies.** Key constraints are expressed by functionality axioms of the form $(\mathit{funct}\, P)$ or $(\mathit{funct}\, P^-)$ where $P$ is a property and $P^-$ denotes the inverse property of $P$. Figure 5 shows their logical semantics in the form of logical rules.

| DL notation | corresponding logical rule |
|---|---|
| $(\mathit{funct}\, P)$ | $P(X,Y) \wedge P(X,Z) \Rightarrow Y = Z$ |
| $(\mathit{funct}\, P^-)$ | $P(Y,X) \wedge P(Z,X) \Rightarrow Y = Z$ |

Figure 5: Functionality axioms expressible in DL-LITE and not in RDFS

Observe that key constraints may also lead to inconsistencies. This is the case if we attempt to equate two distinct constants, e.g., durand and dupond. For instance, the axiom $(\mathit{funct}\ \mathit{ResponsibleOf}^-)$ expresses that a course must have a unique professor responsible for it. Therefore, a knowledge base containing this axiom and the two facts:

$$\mathit{ResponsibleOf}(\mathit{durand}, \mathit{ue111}) \text{ and } \mathit{ResponsibleOf}(\mathit{dupond}, \mathit{ue111}),$$

would be inconsistent. This is because we assume implicitly that an individual is denoted by a single constant. This natural (in practice) assumption is called in logic the *unique name assumption*.

We will consider the following two fragments of DL-LITE:

DL-LITE$_{\mathcal{R}}$ is obtained by extending the axioms of RDFS with the PI and NI axioms.

DL-LITE$_{\mathcal{F}}$ is obtained by extending the axioms of RDFS with key constraints, the PI and NI axioms, but excluding inclusion between properties. Note that, since DL-LITE$_{\mathcal{F}}$ does not permit to express inclusion between properties, RDFS is not included in DL-LITE$_{\mathcal{F}}$.

One may wonder why one would choose such a convoluted language. Why not simply extend RDFS with the 3 kinds of axioms? This is because functional constraints interact with inclusion constraints in intricate ways. Query evaluation when they are all present is much more complex. This will be illustrated by an example in Section 4.4.

From the previous discussion, there are two fundamental differences between query answering in the context of RDFS and of DL-LITE knowledge bases:

**Inconsistency.** RDFS does not permit expressing any form of negation, so an RDFS knowledge base is always consistent. On the other hand, a DL-LITE knowledge base may be inconsistent. Thus, answering queries through DL-LITE ontologies requires to make sure that the data is consistent with respect to the constraints expressed in the ontology.

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

**Incompleteness.** The rules corresponding to RDFS axioms are safe thereby allowing the simple bottom-up algorithm we described. On the other hand, axioms in DL-LITE may correspond to unsafe rules. Thus a bottom-up approach may infer atoms that are not ground, i.e., some incomplete facts. Therefore, we will have to use a top-down approach for evaluating the queries that is more appropriate than the bottom-up approach.

In Section 4.2, we show an algorithm for checking consistency of a DL-LITE knowledge base, and in Section 4.3 an algorithm for answering conjunctive queries posed to a DL-LITE knowledge base. The particularity of these two algorithms is that they work in two-steps:

1. in a first step, we reason with the Tbox alone (i.e., the ontology without the data) and some conjunctive queries;
2. in the second step, we evaluate these conjunctive queries against the data in the Abox.

Separating ontology reasoning from data processing is typically a desired feature (when possible). In particular, such an approach has the practical interest that it makes it possible to use an SQL engine for the second step, thus taking advantage of well-established query optimization strategies supported by standard relational data management systems. In the first step, we deal with the Tbox only, typically of much smaller size.

In Section 4.4, we show by an example that DL-LITE$_{\mathcal{R}}$ and DL-LITE$_{\mathcal{F}}$ are two maximal fragments of the DL-Lite family for which reformulating queries into SQL is possible: combining constraints expressible in DL-LITE$_{\mathcal{R}}$ and DL-LITE$_{\mathcal{F}}$ may result in an *infinite* number of non redundant SQL reformulations for some queries.

## 4.2 Consistency checking

Towards consistency checking, the first step uses the Tbox alone. It consists in computing the *deductive closure* of the Tbox, i.e., all the inclusion axioms that are logically entailed by the axioms declared in the Tbox. More precisely, the deductive closure (closure for short) of a DL-LITE Tbox is defined as follows.

**Definition 4.1 (Closure of a Tbox)** *Let $\mathcal{T}$ be a* DL-LITE$_{\mathcal{F}}$ *or a* DL-LITE$_{\mathcal{R}}$ *Tbox. The* closure *of $\mathcal{T}$, denoted by $cl(\mathcal{T})$, is inductively defined as follows:*

1. *All the statements in $\mathcal{T}$ are also in $cl(\mathcal{T})$.*
2. *If $B_1 \sqsubseteq B_2$ and $B_2 \sqsubseteq B_3$ are in $cl(\mathcal{T})$, then $B_1 \sqsubseteq B_3$ is in $cl(\mathcal{T})$.*
3. *If $R_1 \sqsubseteq R_2$ and $\exists R_2 \sqsubseteq B$ are in $cl(\mathcal{T})$, then $\exists R_1 \sqsubseteq B$ is in $cl(\mathcal{T})$.*
4. *If $R_1 \sqsubseteq R_2$ and $\exists R_2^- \sqsubseteq B$ are in $cl(\mathcal{T})$, then $\exists R_1^- \sqsubseteq B$ is in $cl(\mathcal{T})$.*
5. *If $R_1 \sqsubseteq R_2$ and $R_2 \sqsubseteq R_3$ are in $cl(\mathcal{T})$, then $R_1 \sqsubseteq R_3$ is in $cl(\mathcal{T})$.*
6. *If $R_1 \sqsubseteq R_2$ is in $cl(\mathcal{T})$, then $R_1^- \sqsubseteq R_2^-$ is in $cl(\mathcal{T})$.*
7. *If $B_1 \sqsubseteq B_2$ and $B_2 \sqsubseteq \neg B_3$ (or $B_3 \sqsubseteq \neg B_2$) are in $cl(\mathcal{T})$, then $B_1 \sqsubseteq \neg B_3$ is in $cl(\mathcal{T})$.*
8. *If $R_1 \sqsubseteq R_2$ and $\exists R_2 \sqsubseteq \neg B$ (or $B \sqsubseteq \neg \exists R_2$) are in $cl(\mathcal{T})$, then $\exists R_1 \sqsubseteq \neg B$ is in $cl(\mathcal{T})$.*
9. *If $R_1 \sqsubseteq R_2$ and $\exists R_2^- \sqsubseteq \neg B$ (or $B \sqsubseteq \neg \exists R_2^-$) are in $cl(\mathcal{T})$, then $\exists R_1^- \sqsubseteq \neg B$ is in $cl(\mathcal{T})$.*

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

10. *If $R_1 \sqsubseteq R_2$ and $R_2 \sqsubseteq \neg R_3$ (or $R_3 \sqsubseteq \neg R_2$) are in $cl(\mathcal{T})$, then $R_1 \sqsubseteq \neg R_3$ is in $cl(\mathcal{T})$.*

11. *If $R_1 \sqsubseteq \neg R_2$ or $R_2 \sqsubseteq \neg R_1$ is in $cl(\mathcal{T})$, then $R_1^- \sqsubseteq \neg R_2^-$ is in $cl(\mathcal{T})$.*

12. (a) *In the case in which $\mathcal{T}$ is a* DL-LITE$_\mathcal{F}$ *Tbox, if one of the statements $\exists R \sqsubseteq \neg\exists R$ or $\exists R^- \sqsubseteq \neg\exists R^-$ is in $cl(\mathcal{T})$, then both such statements are in $cl(\mathcal{T})$.*
    (b) *In the case in which $\mathcal{T}$ is a* DL-LITE$_\mathcal{R}$ *Tbox, if one of the statements $\exists R \sqsubseteq \neg\exists R$, $\exists R^- \sqsubseteq \neg\exists R^-$, or $R \sqsubseteq \neg R$ is in $cl(\mathcal{T})$, then all three such statements are in $cl(\mathcal{T})$.*

Observe that although all axioms should be considered to construct this closure, only negative inclusions and key constraints can raise an inconsistency. The set of all the negative inclusions and key constraints in $cl(\mathcal{T})$ is called the *NI-closure*. For example, consider the Tbox of Figure 6 made of the RDFS ontology shown in Figure 2 enriched with the PIs and NI shown in Figure 4. The NI-closure of that Tbox is shown in Figure 7.

| **DL notation** | **FOL notation** |
|---|---|
| $AcademicStaff \sqsubseteq Staff$ | $AcademicStaff(X) \Rightarrow Staff(X)$ |
| $Professor \sqsubseteq AcademicStaff$ | $Professor(X) \Rightarrow AcademicStaff(X)$ |
| $Lecturer \sqsubseteq AcademicStaff$ | $Lecturer(X) \Rightarrow AcademicStaff(X)$ |
| $PhDStudent \sqsubseteq Lecturer$ | $PhDStudent(X) \Rightarrow Lecturer(X)$ |
| $PhDStudent \sqsubseteq Student$ | $PhDStudent(X) \Rightarrow Student(X)$ |
| $\exists TeachesIn \sqsubseteq AcademicStaff$ | $TeachesIn(X,Y) \Rightarrow AcademicStaff(X)$ |
| $\exists TeachesIn^- \sqsubseteq Course$ | $TeachesIn(X,Y) \Rightarrow Course(Y)$ |
| $\exists ResponsibleOf \sqsubseteq Professor$ | $ResponsibleOf(X,Y) \Rightarrow Professor(X)$ |
| $\exists ResponsibleOf^- \sqsubseteq Course$ | $ResponsibleOf(X,Y) \Rightarrow Course(Y)$ |
| $\exists TeachesTo \sqsubseteq AcademicStaff$ | $TeachesTo(X,Y) \Rightarrow AcademicStaff(X)$ |
| $\exists TeachesTo^- \sqsubseteq Student$ | $TeachesTo(X,Y) \Rightarrow Student(Y)$ |
| $\exists Leads \sqsubseteq AdminStaff$ | $Leads(X,Y) \Rightarrow AdminStaff(X)$ |
| $\exists Leads^- \sqsubseteq Dept$ | $Leads(X,Y) \Rightarrow Dept(Y)$ |
| $\exists RegisteredIn \sqsubseteq Student$ | $RegisteredIn(X,Y) \Rightarrow Student(X)$ |
| $\exists RegisteredIn^- \sqsubseteq Course$ | $RegisteredIn(X,Y) \Rightarrow Course(Y)$ |
| $ResponsibleOf \sqsubseteq TeachesIn$ | $ResponsibleOf(X,Y) \Rightarrow TeachesIn(X,Y)$ |
| $Professor \sqsubseteq \exists TeachesIn$ | $Professor(X) \Rightarrow \exists Y TeachesIn(X,Y)$ |
| $Course \sqsubseteq \exists RegisteredIn^-$ | $Course(X) \Rightarrow \exists Y RegisteredIn(Y,X)$ |
| $Student \sqsubseteq \neg Staff$ | $Student(X) \Rightarrow \neg Staff(X)$ |

Figure 6: A DL-LITE Tbox

This example shows that it is possible to infer an important number of new NIs. In fact, we have to compute *all* the consequences. But as we will see there is at most a polynomial number of consequences.

We use three propositions for analyzing the consistency problem: one for evaluating the complexity of evaluating the closure and the last two for showing the logical soundness and completeness of this closure. Finally, a fourth proposition will show how to use these results for data consistency checking.

**Proposition 4.2 (Size of the closure of a Tbox and complexity of its computation)** *Let $\mathcal{T}$ be a* DL-LITE$_\mathcal{F}$ *or a* DL-LITE$_\mathcal{R}$ *Tbox.*

1. *The number of statements in $cl(\mathcal{T})$ is at most polynomial in the size of $\mathcal{T}$.*

| **DL notation** | **FOL notation** |
|---|---|
| $Student \sqsubseteq \neg Staff$ | $Student(X) \Rightarrow \neg Staff(X)$ |
| $PhDStudent \sqsubseteq \neg Staff$ | $PhDStudent(X) \Rightarrow \neg Staff(X)$ |
| $\exists TeachesTo^- \sqsubseteq \neg Staff$ | $TeachesTo(Y,X) \Rightarrow \neg Staff(X)$ |
| $\exists RegisteredIn \sqsubseteq \neg Staff$ | $RegisteredIn(X,Y) \Rightarrow \neg Staff(X)$ |
| $Lecturer \sqsubseteq \neg Student$ | $Lecturer(X) \Rightarrow \neg Student(X)$ |
| $Lecturer \sqsubseteq \neg PhDStudent$ | $Lecturer(X) \Rightarrow \neg PhDStudent(X)$ |
| $Lecturer \sqsubseteq \neg \exists TeachesTo^-$ | $Lecturer(X) \Rightarrow \neg \exists Y[TeachesTo(Y,X)]$ |
| $Lecturer \sqsubseteq \neg \exists RegisteredIn$ | $Lecturer(X) \Rightarrow \neg \exists Y[RegisteredIn(X,Y)]$ |
| $Professor \sqsubseteq \neg Student$ | $Professor(X) \Rightarrow \neg Student(X)$ |
| $Professor \sqsubseteq \neg PhDStudent$ | $Professor(X) \Rightarrow \neg PhDStudent(X)$ |
| $Professor \sqsubseteq \neg \exists TeachesTo^-$ | $Professor(X) \Rightarrow \neg \exists Y[TeachesTo(Y,X)]$ |
| $Professor \sqsubseteq \neg \exists RegisteredIn$ | $Professor(X) \Rightarrow \neg \exists Y[RegisteredIn(X,Y)]$ |
| $AcademicStaff \sqsubseteq \neg Student$ | $AcademicStaff(X) \Rightarrow \neg Student(X)$ |
| $AcademicStaff \sqsubseteq \neg PhDStudent$ | $AcademicStaff(X) \Rightarrow \neg PhDStudent(X)$ |
| $AcademicStaff \sqsubseteq \neg \exists TeachesTo^-$ | $AcademicStaff(X) \Rightarrow \neg \exists Y[TeachesTo(Y,X)]$ |
| $AcademicStaff \sqsubseteq \neg \exists RegisteredIn$ | $AcademicStaff(X) \Rightarrow \neg \exists Y[RegisteredIn(X,Y)]$ |
| $Staff \sqsubseteq \neg Student$ | $Staff(X) \Rightarrow \neg Student(X)$ |
| $Staff \sqsubseteq \neg PhDStudent$ | $Staff(X) \Rightarrow \neg PhDStudent(X)$ |
| $Staff \sqsubseteq \neg \exists TeachesTo^-$ | $Staff(X) \Rightarrow \neg \exists Y[TeachesTo(Y,X)]$ |
| $Staff \sqsubseteq \neg \exists RegisteredIn$ | $Staff(X) \Rightarrow \neg \exists Y[RegisteredIn(X,Y)]$ |
| $\exists TeachesTo \sqsubseteq \neg Student$ | $TeachesTo(X,Y) \Rightarrow \neg Student(X)$ |
| $\exists TeachesTo \sqsubseteq \neg PhDStudent$ | $TeachesTo(X,Y) \Rightarrow \neg PhDStudent(X)$ |
| $\exists TeachesTo \sqsubseteq \neg \exists TeachesTo^-$ | $TeachesTo(X,Y) \Rightarrow \neg \exists Z[TeachesTo(Z,X)]$ |
| $\exists TeachesTo \sqsubseteq \neg \exists RegisteredIn$ | $TeachesTo(X,Y) \Rightarrow \neg \exists Z[RegisteredIn(X,Z)]$ |
| $\exists TeachesIn \sqsubseteq \neg Student$ | $TeachesIn(X,Y) \Rightarrow \neg Student(X)$ |
| $\exists TeachesIn \sqsubseteq \neg PhDStudent$ | $TeachesIn(X,Y) \Rightarrow \neg PhDStudent(X)$ |
| $\exists TeachesIn \sqsubseteq \neg \exists TeachesTo^-$ | $TeachesIn(X,Y) \Rightarrow \neg \exists Z[TeachesTo(Z,X)]$ |
| $\exists TeachesIn \sqsubseteq \neg \exists RegisteredIn$ | $TeachesIn(X,Y) \Rightarrow \neg \exists Z[RegisteredIn(X,Z)]$ |

Figure 7: The NI-closure of the Tbox in Figure 6

2. *$cl(\mathcal{T})$ can be computed in polynomial time in the size of $\mathcal{T}$.*

**Proof** (sketch). (1.) Follows from the form of the statements that are allowed in a DL-LITE$_\mathcal{F}$ or a DL-LITE$_\mathcal{R}$ Tbox.

For (2.), consider the items (2.) to (12.) in Definition 4.1. These are *closure rules* that are exhaustively applied to the Tbox until saturation. Let $\mathcal{T}_0 = \mathcal{T}$. For each $i$, let $\Delta_i$ be the set of statements that can be derived from $\mathcal{T}_i$ directly using the closure rules (2.) to (12.). Let $\mathcal{T}_{i+1} = \mathcal{T}_i \cup \Delta_i$. Clearly, for each $i$, $\mathcal{T}_i \subseteq cl(\mathcal{T})$, so its size is polynomial in the size of $\mathcal{T}$.

Now since the size of $\mathcal{T}_i$ is polynomial in the size of $\mathcal{T}$, each step of the computation can clearly be performed in PTIME. Since the number of steps is less than the number of statements in $cl(\mathcal{T})$, the entire computation can be performed in PTIME. □

The next proposition states the soundness of the closure.

**Proposition 4.3 (Soundness of the closure of a Tbox)** *For each $\mathcal{T}$, $\mathcal{T} \equiv cl(\mathcal{T})$. In other words, for each Abox $\mathcal{A}$ satisfying a Tbox $\mathcal{T}$, $\mathcal{A}$ also satisfies $cl(\mathcal{T})$.*

**Proof** (sketch). $cl(\mathcal{T}) \models \mathcal{T}$, since $\mathcal{T}$ is included in $cl(\mathcal{T})$. Clearly, the application of each closure rule is sound. So for each $i$, $\mathcal{T}_i \models \mathcal{T}_{i+1}$. By induction, $\mathcal{T} = \mathcal{T}_0 \models \mathcal{T}_i$ for each $i$. Thus

$\mathcal{T} \models cl(\mathcal{T})$, so $\mathcal{T} \equiv cl(\mathcal{T})$. □

The next proposition establishes the completeness of the closure of a Tbox $\mathcal{T}$: $cl(\mathcal{T})$ contains all the PIs, NIs and key constraints that are logically entailed by that Tbox (up to equivalence).

**Proposition 4.4 (Completeness of the closure of a Tbox)** *Let $\mathcal{T}$ be a* DL-LITE$_{\mathcal{F}}$ *or a* DL-LITE$_{\mathcal{R}}$ *Tbox. Then*

1. *Let $X \sqsubseteq Y$ be a NI or a PI. If $\mathcal{T} \models X \sqsubseteq \neg X$*
   *then $X \sqsubseteq \neg X \in cl(\mathcal{T})$,*
   *otherwise $\mathcal{T} \models X \sqsubseteq Y$ iff $X \sqsubseteq Y \in cl(\mathcal{T})$ or $\neg Y \sqsubseteq \neg X \in cl(\mathcal{T})$.*

2. *$\mathcal{T} \models (funct\ R)$ iff $(funct\ R) \in cl(\mathcal{T})$ or $\exists R \sqsubseteq \neg \exists R \in cl(\mathcal{T})$.*

**Proof** (sketch). We build a canonical interpretation $I$ of the classes and properties appearing in $\mathcal{T}$ as follows: for each $X$ such that $X \sqsubseteq \neg X \in cl(\mathcal{T})$, $I(X) = \emptyset$; for the other classes or properties, we associate a constant (respectively a pair of constants) with each class (respectively each property) and we initialize their interpretations with those (pairs of) constants. Then, we complete these interpretations by applying the positive inclusions in $cl(\mathcal{T})$ as logical rules in a bottom up manner until saturation. For instance, if $A \sqsubseteq \exists P$ is in $cl(\mathcal{T})$, from the initial state where $a$ is $I(A)$, and $(p_1, p_2)$ is in $I(P)$, we add a new constant $p_3$ in the domain of interpretation and the pair $(a, p_3)$ in $I(P)$. Now, if $\exists P \sqsubseteq \exists Q$ is also in $cl(\mathcal{T})$, we add two new constants $p_4$ and $p_5$ in the domain and the pairs $(a, p_4)$ and $(p_1, p_5)$ to $I(Q)$.

Clearly, by construction, $I$ is a model of each PI in $\mathcal{T}$. It is also a model of each NI $X \sqsubseteq \neg Y$ in $\mathcal{T}$. Suppose that this not the case: there would exist a constant $x$ which is in $I(X)$ and in $I(Y)$. By construction of $I$, it would exist a chain of positive inclusions in $cl(\mathcal{T})$ between $X$ and $Y$ and thus $X \sqsubseteq Y$ would be in $cl(\mathcal{T})$, and therefore $X \sqsubseteq \neg X$ would be in $cl(\mathcal{T})$ too, and in this case $I(X)$ would be empty, which contradicts that $I$ is not a model of $X \sqsubseteq \neg Y$.

To prove (1), if $\mathcal{T} \models X \sqsubseteq \neg X$, in every model of $\mathcal{T}$, $X$ must be empty, in particular in $I$. By construction of $I$, this means that $X \sqsubseteq \neg X \in cl(\mathcal{T})$. Otherwise, consider a PI $X \sqsubseteq Y$ such that $\mathcal{T} \models X \sqsubseteq Y$. Since $I$ is a model of $\mathcal{T}$, $I(X) \subseteq I(Y)$. By construction of $I$, this means that there exists a chain of positive inclusions in $cl(\mathcal{T})$ between $X$ and $Y$ and thus $X \sqsubseteq Y$ is in $cl(\mathcal{T})$.

Consider now a NI $X \sqsubseteq \neg Y$ such that neither $X \sqsubseteq \neg Y$ nor $Y \sqsubseteq \neg X$ belong to $cl(\mathcal{T})$. Let us define the interpretation $J$ such that

- $J(Z) = \emptyset$ for each class or property $Z$ appearing in the right-hand side of a NI in $cl(\mathcal{T})$ of the form $X \sqsubseteq \neg U$ or $U \sqsubseteq \neg X$,

- $J(A) = D$ (where $D$ is the whole domain of interpretation) for the other classes, and $J(P) = D \times D$ for the other properties.

In particular $J(X) = D$ (since $X \sqsubseteq \neg X$ is not in $cl(\mathcal{T})$), and $J(Y) = D$ (since neither $X \sqsubseteq \neg Y$ nor $Y \sqsubseteq \neg X$ belong to $cl(\mathcal{T})$). Clearly, $J$ is a model of $\mathcal{T}$, but it is not a model of $X \sqsubseteq \neg Y$. Therefore, $\mathcal{T} \not\models X \sqsubseteq \neg Y$. This ends the proof of (1).

For proving (the contraposite of) (2), we adapt the above canonical interpretation $I$ by initializing with $\{(p,q),(p,r)\}$ the interpretation of all the properties $R$ such that neither $(functR)$ nor $\exists R \sqsubseteq \neg \exists R$ belong to $cl(\mathcal{T})$. And we show that the resulting interpretation $I'$ is a model of $\mathcal{T}$ in which the constraints of functionality of such $R$ is not satisfied. □

Finally, the last proposition establishes that checking consistency can be reduced to check whether the data in $\mathcal{A}$ satisfy every NI in the closure.

**Proposition 4.5 (Consistency checking using NI-closure)** *Let $\mathcal{T}$ be a* DL-LITE$_{\mathcal{F}}$ *or a* DL-LITE$_{\mathcal{R}}$ *Tbox. Let $\mathcal{A}$ be an Abox associated to $\mathcal{T}$. $\langle \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable iff there exists a NI or a key constraint in the closure of $\mathcal{T}$ which is violated by some facts of $\mathcal{A}$.*

**Proof** (sketch). For every constant $a$ appearing in the Abox $\mathcal{A}$, we define $\mathcal{A}(a)$ as the set of facts extracted from $\mathcal{A}$ as follows:

- if $A(a) \in \mathcal{A}$, then $A(a)$ is added in $\mathcal{A}(a)$
- if $P(a,b) \in \mathcal{A}$, then $(\exists P)(a)$ is added in $\mathcal{A}(a)$ and $(\exists P^{-})(b)$ is added in $\mathcal{A}(b)$

We first show that if $\langle \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable, there exists a constant $a$ such that $\langle \mathcal{T}, \mathcal{A}(a) \rangle$ is unsatisfiable. In fact we show the contrapositive: suppose that for every constant $a$, $\langle \mathcal{T}, \mathcal{A}(a) \rangle$ is satisfiable: for each $a$, there exists an interpretation $I_a$ satisfying the inclusions in $\mathcal{T}$ and all the facts in $\mathcal{A}(a)$. It is easy to show that the interpretation $I$ defined on the union of domains of interpretations as follows is a model of $\langle \mathcal{T}, \mathcal{A} \rangle$ (which is then satisfiable):

- for every class or property $X$: $I(X) = \bigcup_a I_a(X)$
- for every constant $a$: $I(a) = I_a(a)$

Then, since each $\mathcal{A}(a)$ is a conjunction of facts of the form $X(a)$, if $\langle \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable, there exists a constant $a$ such that $\mathcal{T}, X_1(a) \wedge ... \wedge X_n(a)$ is unsatisfiable. Therefore, $\mathcal{T}, \exists x(X_1(x) \wedge ... \wedge X_n(x))$ is unsatisfiable. This entails: $\mathcal{T} \models \forall x(\neg X_1(x) \vee ... \vee \neg X_n(x))$, which in DL notation corresponds to: $\mathcal{T} \models X_1 \sqsubseteq \neg X_2 \sqcup ... \sqcup \neg X_n$. Because of the form of the inclusion allowed in DL-LITE, there must exist $i$ such that $\mathcal{T} \models X_1 \sqsubseteq \neg X_i$. According to Proposition 4.4, this entails that the corresponding NI $X_1 \sqsubseteq \neg X_i$ is in the closure of $\mathcal{T}$ and that $\mathcal{A}$ violates it (since it includes $X_1(a)$ and $X_i(a)$).

Conversely, it is easy to show that if a NI in the closure of $\mathcal{T}$ is violated by some facts in the Abox, then $\langle \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable. If it were not the case, since according Proposition 4.3 $\mathcal{T}$ and $cl(\mathcal{T})$ have the same models, there would be a model in which all the NIs in the closure of $\mathcal{T}$ would be satisfied by the facts in $\mathcal{A}$. □

The second step of consistency checking, after the NI-closure is computed, does not require any further computation on the Tbox $\mathcal{T}$. This second step simply consists in evaluating against the Abox $\mathcal{A}$ (seen as a relational database) a Boolean query corresponding to each *negated* NI in the NI-closure of the Tbox. If one of those Boolean queries is evaluated to *true* against $\mathcal{A}$, it means that some data in the Abox $\mathcal{A}$ violates the corresponding NI, and therefore the knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent.

For example, consider the NI: $\exists \mathit{TeachesTo} \sqsubseteq \neg \mathit{PhDStudent}$. Its corresponding FOL formula $\varphi$ and its negation are:

$$\begin{array}{ll} \forall x,y' [\mathit{TeachesTo}(x,y') \Rightarrow \neg \mathit{PhDStudent}(x)] & \varphi \\ \exists x,y' [\mathit{TeachesTo}(x,y') \wedge \mathit{PhDStudent}(x)] & \neg\varphi \end{array}$$

and the corresponding Boolean query is:

$$q_{unsat}() \; :- \; \mathit{TeachesTo}(x,y'), \mathit{PhDStudent}(x)$$

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

i.e., the direct translation of the negation of the NI.

Consider the evaluation of the $q_{unsat}$ query against the Abox $\mathcal{A}$ of Figure 1. It evaluates to *true*: consider valuation $\nu$ with $\nu(x) = pierre$, $\nu(y') = paul$ and the facts

$$PhDStudent(paul) \text{ and } TeachesTo(paul,pierre).$$

Thus, the knowledge base $\mathcal{K}$ made of the Tbox of Figure 6 and the Abox of Figure 1 is inconsistent.

The transformation of NIs into Boolean queries that correspond to their negation is described in Definition 4.6.

**Definition 4.6 (Transformation of NIs into Boolean queries)** *The transformation $\delta$ of NIs into Boolean queries corresponding to their negation is defined as follows:*

$$\begin{array}{lcl}
\delta(B_1 \sqsubseteq \neg B_2) & = & q_{unsat} :- \gamma_1(x), \gamma_2(x) \text{ such that} \\
 & & \gamma_i(x) = A_i(x) \text{ if } B_i = A_i \\
 & & \gamma_i(x) = P_i(x,y_i) \text{ if } B_i = \exists P_i \\
 & & \gamma_i(x) = P_i(y_i,x) \text{ if } B_i = \exists P_i^- \\
\delta(R_1 \sqsubseteq \neg R_2) & = & q_{unsat} :- \rho_1(x,y) \text{ , } \rho_2(x,y) \text{ such that} \\
 & & \rho_i(x,y) = P_i(x,y) \text{ if } R_i = P_i \\
 & & \rho_i(x,y) = P_i(y,x) \text{ if } R_i = P_i^- \\
\delta((funct\ P)) & = & q_{unsat} :- P(x,y) \text{ , } P(x,z) \text{ , } y \neq z \\
\delta((funct\ P^-)) & = & q_{unsat} :- P(x,y) \text{ , } P(z,y) \text{ , } x \neq z
\end{array}$$

This second step of consistency checking is summarized in the *Consistent* Algorithm (Algorithm 2). In the algorithm, for each NI clause $\alpha$, the query $q_{unsat,\alpha}$ is an SQL query computing the Boolean conjunctive queries $\delta(\alpha)$. Also, $db(\mathcal{A})$ denotes the $\mathcal{A}$ set in a relational database.

```
Algorithm 2: The Consistent algorithm
Consistent($\mathcal{T}$,$\mathcal{A}$)
Input: a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: true if $\mathcal{K}$ is satisfiable, false otherwise
(1)    $q_{unsat} = \emptyset$ (i.e., $q_{unsat}$ is $false$)
(2)    foreach $\alpha \in cln(\mathcal{T})$  let  $q_{unsat} = q_{unsat} \cup q_{unsat,\alpha}(db(\mathcal{A}))$
(3)
(4)        if $q_{unsat} = \emptyset$ return true
(5)        else return false
```

It is important to emphasize that this two-step approach for consistency checking does not require any inference on the data. The only inferences concern the Tbox and consist in computing the deductive closure of its axioms, from which the NI-closure (denoted $cln(\mathcal{T})$ in the Algorithm) is extracted.

Consider the Abox $\mathcal{A}'$ obtained from the inconsistent Abox $\mathcal{A}$ in Figure 1 by deleting the fact $PhDStudent(paul)$. The knowledge base made of the Abox $\mathcal{A}'$ in Figure 8 and the Tbox $\mathcal{T}$ in Figure 6 is consistent. (See Exercise 6.4.)

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

| **Subject** | **Predicate** | **Object** | **FOL semantics** |
|---|---|---|---|
| dupond | Leads | infoDept | Leads(dupond,infoDept) |
| dupond | `rdf:type` | Professor | Professor(dupond) |
| durand | ResponsibleOf | ue111 | ResponsibleOf(durand,ue111) |
| durand | Leads | csDept | Leads(durand,csDept) |
| paul | TeachesTo | pierre | TeachesTo(paul,pierre) |
| pierre | EnrolledIn | infoDept | EnrolledIn(pierre, infodept) |
| pierre | `rdf:type` | Undergrad | Undergrad(pierre) |
| pierre | RegisteredTo | ue111 | Registered(pierre, ue111) |
| ue111 | OfferedBy | infoDept | OfferedBy(ue111,infoDept) |
| ue111 | `rdf:type` | CSCourse | CSCourse(ue111) |
| jim | EnrolledIn | csDept | EnrolledIn(jim, csDept) |
| csDept | `rdf:type` | TeachingDept | TeachingDept(csDept) |

Figure 8: $\mathcal{A}'$: an Abox consistent w.r.t the Tbox of Figure 6

## 4.3 Answer set evaluation

In the previous section, the negative constraints played the main role. Once we know the knowledge base is consistent and move to query answering, the positive constraints take over.

Answering queries to a DL-LITE knowledge base is done in two steps. The first step is the query reformulation, which consists in translating the original query $q$ into a set $Q$ of queries. The second step consists in evaluating the queries in $Q$ over the Abox (again seen as a relational database). The beauty of the approach is that this will provide the answer set. Of course, simply evaluating $q$ over the Abox would possibly yield an incomplete answer. Completeness is achieved by the "reasoning" in the reformulation step. During this step, we access only the Tbox and not the data.

The query reformulation step is performed by the *PerfectRef* (Algorithm 3). It consists in reformulating the initial query by using the PIs in $\mathcal{T}$ as rewriting rules. The intuition is that PIs are seen as logical rules that are applied in backward-chaining to query atoms in order to expand them (in a resolution style). In databases, this is called a *chase*.

The queries we consider, i.e., the conjunctive queries, consist of several atoms. In general, because of the existential variables, new variables are introduced in queries. So we could be lead to generate more and more queries with new variables. It turns out that we will be able to control this process and generate only a finite number of distinct queries. This is due to the limitations of the constraints allowed in the Tbox. As outlined in Section 4.4, as soon as we allow the combination of key constraints with inclusions of properties, we may generate an infinite number of non redundant queries.

Consider a PI rule $\alpha \Rightarrow \beta$. Applicability of the rule to an atom of a query is defined by:

- It is *applicable to an atom* $A(x)$ of a query if $A$ occurs in $\beta$.
- It is *applicable to an atom* $P(x_1, x_2)$ of a query if
  - $\alpha \Rightarrow \beta$ is a role inclusion assertion and $P$ or $P^-$ occurs in $\beta$;
  - $x_2 = \_$ and $\beta$ is $\exists P$;

– $x_1 = \_$ and $\beta$ is $\exists P^-$.

As usual, _ denotes here an unbounded existential variable of a query.

The following definition defines the result $gr(g, I)$ of the *goal reduction* of the atom $g$ using the PI $I$, which is at the core of *PerfectRef*.

**Definition 4.7 (Backward application of a PI to an atom)** *Let $I$ be an inclusion assertion that is applicable to the atom $g$. Then, $gr(g, I)$ is the atom defined as follows:*

| | | | | | |
|---|---|---|---|---|---|
| *if* | $g = A(x)$ | *and* | $I = A_1 \sqsubseteq A$, | *then* | $gr(g,I) = A_1(x)$ |
| *if* | $g = A(x)$ | *and* | $I = \exists P \sqsubseteq A$, | *then* | $gr(g,I) = P(x,\_)$ |
| *if* | $g = A(x)$ | *and* | $I = \exists P^- \sqsubseteq A$, | *then* | $gr(g,I) = P(\_,x)$ |
| *if* | $g = P(x,\_)$ | *and* | $I = A \sqsubseteq \exists P$, | *then* | $gr(g,I) = A(x)$ |
| *if* | $g = P(x,\_)$ | *and* | $I = \exists P_1 \sqsubseteq \exists P$, | *then* | $gr(g,I) = P_1(x,\_)$ |
| *if* | $g = P(x,\_)$ | *and* | $I = \exists P_1^- \sqsubseteq \exists P$, | *then* | $gr(g,I) = P_1(\_,x)$ |
| *if* | $g = P(\_,x)$ | *and* | $I = A \sqsubseteq \exists P^-$, | *then* | $gr(g,I) = A(x)$ |
| *if* | $g = P(\_,x)$ | *and* | $I = \exists P_1 \sqsubseteq \exists P^-$, | *then* | $gr(g,I) = P_1(x,\_)$ |
| *if* | $g = P(\_,x)$ | *and* | $I = \exists P_1^- \sqsubseteq \exists P^-$, | *then* | $gr(g,I) = P_1(\_,x)$ |
| *if* | $g = P(x_1,x_2)$ | *and* | *either* $I = P_1 \sqsubseteq P$ *or* $I = P_1^- \sqsubseteq P^-$ | *then* | $gr(g,I) = P_1(x_1,x_2)$ |
| *if* | $g = P(x_1,x_2)$ | *and* | *either* $I = P_1 \sqsubseteq P^-$ *or* $I = P_1^- \sqsubseteq P$ | *then* | $gr(g,I) = P_1(x_2,x_1)$ |

The subtle point of *PerfectRef* is the need of simplifying the produced reformulations, so that some PIs that were not applicable to a reformulation become applicable to its simplifications. A simplification amounts to unify two atoms of a reformulation using their *most general unifier* and then to switch the possibly new unbounded existential variables to the anonymous variable denoted _.

Let us illustrate the reformulation step of the following query using the PIs in the Tbox $\mathcal{T}$ of Figure 6:

```
q(x):- TeachesIn(x,y), RegisteredIn(z,y), Student(z).
```

Figure 9 shows the result returned by *PerfectRef*$(q(x),\mathcal{T})$.

We detail here the inference chain leading to some reformulations that are particularly interesting for getting answers for $q$ from the data in the Abox $\mathcal{A}'$ of Figure 8. This also illustrates the need of the simplification step. The reformulation:

```
q4(x):- TeachesIn(x,y), RegisteredIn(_,y).
```

is obtained by:

- the backward application to the atom $Student(z)$ of $q(x)$ of the PI: $\exists RegisteredIn \sqsubseteq Student$, which leads to the reformulation

$$q'(x)\ :-\ TeachesIn(x,y), RegisteredIn(z,y), Registered(z,\_)$$

  in which the anonymous variable _ appearing in the atom $Registered(z,\_)$ denotes the unbounded existential variable produced by the backward application of the PI,

| **Initial query** q(x):- TeachesIn(x,y), RegisteredIn(z,y), Student(z) | | |
|---|---|---|
| **Reformulations** | **applied PI** | **Reformulated query** |
| $q_1$(x):- ResponsibleOf(x,y), RegisteredIn(z,y), Student(z) | *ResponsibleOf* $\sqsubseteq$ *TeachesIn* | q(x) |
| $q_2$(x):- TeachesIn(x,y), RegisteredIn(z,y), PhDStudent(z) | *PhDStudent* $\sqsubseteq$ *Student* | q(x) |
| $q_3$(x):- TeachesIn(x,y), RegisteredIn(z,y), TeachesTo(_,z) | $\exists$*TeachesTo*$^-$ $\sqsubseteq$ *Student* | q(x) |
| $q_4$(x):- TeachesIn(x,y), RegisteredIn(_,y) | $\exists$*RegisteredIn* $\sqsubseteq$ *Student* | q(x) |
| $q_5$(x):- ResponsibleOf(x,y), RegisteredIn(z,y), PhDStudent(z) | *PhDStudent* $\sqsubseteq$ *Student* | $q_1$(x) |
| $q_6$(x):- ResponsibleOf(x,y), RegisteredIn(z,y), TeachesTo(_,z) | $\exists$*TeachesTo*$^-$ $\sqsubseteq$ *Student* | $q_1$(x) |
| $q_7$(x):- ResponsibleOf(x,y), RegisteredIn(_,y), | $\exists$*RegisteredIn* $\sqsubseteq$ *Student* | $q_1$(x) |
| $q_8$(x):- ResponsibleOf(x,y), RegisteredIn(z,y), PhDStudent(z) | *ResponsibleOf* $\sqsubseteq$ *TeachesIn* | $q_2$(x) |
| $q_9$(x):- ResponsibleOf(x,y), RegisteredIn(z,y), TeachesTo(_,z) | *ResponsibleOf* $\sqsubseteq$ *TeachesIn* | $q_3$(x) |
| $q_{10}$(x):- ResponsibleOf(x,y), RegisteredIn(_,y) | *ResponsibleOf* $\sqsubseteq$ *TeachesIn* | $q_4$(x) |
| $q_{11}$(x):- TeachesIn(x,y), Course(y) | *Course* $\sqsubseteq$ $\exists$*RegisteredIn*$^-$ | $q_4$(x) |
| $q_{12}$(x):- TeachesIn(x,_) | $\exists$*TeachesIn*$^-$ $\sqsubseteq$ *Course* | $q_{11}$(x) |
| $q_{13}$(x):- ResponsibleOf(x,_) | *ResponsibleOf* $\sqsubseteq$ *TeachesIn* | $q_{12}$(x) |
| $q_{14}$(x):- Professor(x) | *Professor* $\sqsubseteq$ $\exists$*TeachesIn* | $q_{12}$(x) |

Figure 9: A query and its reformulations obtained by *PerfectRef* applied to the Tbox of Figure 6

- followed by a simplification step, consisting in unifying the two redundant atoms in the body of $q'$: the atom *Registered*$(z,y)$ is kept instead of the atom *Registered*$(z,\_)$ because $y$ is an existential variable which is bounded within the body of $q'$. But now, the existential variable $z$ is unbounded within the body of $q'$: it is replaced by the anonymous variable _.

In turn, $q_4(x)$ can be reformulated by the backward application of the PI *Course* $\sqsubseteq$ $\exists$*RegisteredIn*$^-$ to the atom *RegisteredIn*$(\_,y)$, which results in the reformulation $q_{11}(x)$:

```
q11(x):- TeachesIn(x,y), Course(y).
```

Then, the reformulation $q_{12}(x)$ is produced by the backward application of the PI $\exists$ *TeachesIn*$^-$ $\sqsubseteq$ *Course*, and the simplification by unification of the two atoms followed by the replacement of the existential variable $y$, now unbounded, with the anonymous variable _.

```
q12(x):- TeachesIn(x,_).
```

Finally, the reformulations $q_{13}(x)$ and $q_{14}(x)$ are obtained from the backward application of the PIs *ResponsibleOf* $\sqsubseteq$ *TeachesIn* and *Professor* $\sqsubseteq$ $\exists$*TeachesIn* respectively.

```
q13(x):- ResponsibleOf(x,_).
```

```
q14(x):- Professor(x).
```

It is important to notice that the answers *durand* and *dupond* are obtained for the initial query $q(x)$ thanks to those reformulations $q_{13}(x)$ and $q_{14}(x)$: they would not be returned by the standard evaluation of the query $q(x)$ against the Abox $\mathcal{A}'$ of Figure 8.

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

In the *PerfectRef* algorithm (Algorithm 3):

- The notation $q[g/gr(g, I)]$ (Line 7) denotes the replacement of the atom $g$ in the body of the query $q$ with the result $gr(g, I)$ of the backward application of the PI $I$ to the atom $g$,
- The operator $reduce(q, g, g')$ (Line 10) denotes the simplification of the body of $q$ obtained by replacing the conjunction of its two atoms $g$ and $g'$ with their *most general unifier* (if $g$ and $g'$ can be unified),
- The operator $\tau$ (Line 10) replaces in the body of a query all the possibly new *unbounded* existential variables with the anonymous variable denoted _.

```
Algorithm 3: The PerfectRef algorithm
PerfectRef($q$, $\mathcal{T}$)
Input: a conjunctive query $q$ and a Tbox $\mathcal{T}$
Output: a union of conjunctive queries: $PR$
(1)    $PR := \{q\}$
(2)    repeat
(3)        $PR' := PR$
(4)        foreach $q \in PR'$
(5)            (a) foreach $g \in q$
(6)                if a PI $I \in \mathcal{T}$ is applicable to $g$
(7)                    $PR := PR \cup \{q[g/gr(g, I)]\}$
(8)            (b) foreach $g_1, g_2 \in q$
(9)                if $g_1$ et $g_2$ sont unifiables
(10)                   $PR := PR \cup \{\tau(reduce(q, g_1, g_2))\}$
(11)       until $PR' = PR$
```

Figure 9 shows the result returned by *PerfectRef*$(q(x), \mathcal{T})$, where $q(x)$ is the query of the previous example, and $\mathcal{T}$ is the Tbox of Figure 6. The second column makes explicit the PI used for obtaining the corresponding reformulation. Note that equivalent reformulations can be produced by different inferences, such as for example the reformulations $q_4(x)$ and $q_7(x)$.

Although we will not prove it here, the following properties hold:

**soundness.** All the facts computed using *PerfectRef* are correct query answers.

**completeness.** All query answers are obtained.

**complexity.** Since we touch the data only for the evaluation of FOL queries, the worst-case complexity is PTIME in the size of the Abox. The number of reformulations is PTIME in the size of the Tbox. Therefore, the complexity of evaluating a query against a DL-LITE$_{\mathcal{R}}$ or DL-LITE$_{\mathcal{F}}$ knowledge base is PTIME in the size of the knowledge base.

## 4.4 Impact of combining DL-LITE$_{\mathcal{R}}$ and DL-LITE$_{\mathcal{F}}$ on query answering

In this section, we exhibit an example showing that the interaction of key constraints (the specificity of DL-LITE$_{\mathcal{F}}$) with inclusion constraints between properties (the specificity of DL-LITE$_{\mathcal{R}}$) may lead to a reformulation of a query into an *infinite* number of conjunctive rewritings, each one likely to bring additional answers. This makes an algorithmic approach such as the one we used for DL-LITE$_\mathcal{R}$ and DL-LITE$_\mathcal{F}$ in isolation incomplete for query answering when DL-LITE$_\mathcal{R}$ and DL-LITE$_\mathcal{F}$ are combined together.

Consider a Tbox made of the following inclusion axioms, in which `R` and `P` are two properties and `S` is a class:

$$R \sqsubseteq P$$
$$(functP)$$
$$S \sqsubseteq \exists R$$
$$\exists R^- \sqsubseteq \exists R$$

Let us consider the following query:

$$q(x) \text{ :- } R(z,x)$$

The following query expression is a valid reformulation for the query $q$:

$$r_1(x) \text{ :- } S(x_1), P(x_1,x)$$

To see this, we observe that from the fact $S(x_1)$ and the PI $S \sqsubseteq \exists R$, it can be inferred that there exists $y$ such that $R(x_1,y)$ holds, and thus $P(x_1,y)$ holds too (since `R` $\sqsubseteq$ `P`). From the functionality constraint on $P$ and the conjunct $P(x_1,x)$ in the body of $r_1$, we can now infer that $y = x$, and thus that $R(x_1,x)$ holds. Therefore, $\exists z R(z,x)$ is logically entailed by $\exists x_1 S(x_1) \wedge P(x_1,x)$, i.e., $r_1(x)$ is contained in the query $q(x)$, and thus is a valid reformulation of the query $q(x)$.

It turns out that the situation is even more subtle. Surprisingly, this reformulation $r_1(x)$ is not the only one. In fact there exists an *infinite* number of different reformulations for $q(x)$. Let $k \geq 2$. The following query is a *valid reformulation* of $q(x)$:

$$r_k(x) \text{ :- } S(x_k), P(x_k, x_{k-1}), \ldots, P(x_1, x)$$

To show that $r_k(x)$ is logically contained in $q(x)$, we exploit again the axiom of functionality of $P$ and the inclusion axiom between $R$ and $P$: from the fact $S(x_k)$ and the PI $S \sqsubseteq \exists R$, it can be inferred that there exists $y_k$ such that $R(x_k,y_k)$ holds, and thus $P(x_k,y_k)$ holds too (since `R` $\sqsubseteq$ `P`). Since $P$ is functional, we get: $y_k = x_{k-1}$, and thus $R(x_k,x_{k-1})$ holds. Now, based on the PI $\exists$`R`$^-$ $\sqsubseteq$ $\exists$`R`, there exists $y_{k-1}$ such that $R(x_{k-1},y_{k-1})$ holds, and with the same reasoning as before, we get $y_{k-1} = x_{k-2}$, and thus $R(x_{k-1},x_{k-2})$ holds. By induction, we obtain that $R(x_1,x)$ holds, i.e., $r_k(x)$ is logically contained in the query $q(x)$.

One can also show that for each $k$, there exists an Abox such that the reformulation $r_k$ returns answers that are not returned by the reformulation $r_{k'}$ for $k' < k$. Thus, there exists an infinite number of non redundant conjunctive reformulations.

It can be shown that if we combine key constraints and inclusions of properties in a restricted way, this problem can be avoided. For instance, if key constraints are forbidden on properties involved in right-hand side of an inclusion axiom, there is a finite number of non redundant conjunctive reformulations and they can be found by the $PerfectRef$ algorithm.

## 5 Further reading

The spreading of RDF data on the Web and the importance of queries for such data is illustrated by the Billion Triple Track of the Semantic Web Challenge [1]. The idea is to "do something" efficiently with one billion of RDFS triples.

**RDFS.** Reasoners for RDFS are available on line and can be downloaded, like for instance the Jena2 ontology API [Jen], which implements the forward-chaining algorithm we described. A SPARQL [W3C08] engine included in the same programmatic Jena environment enables storing and querying data sets made of (asserted + inferred) triples. In fact, since RDFS statements are also stored as RDF triplets, SPARQL can also be used to query the schema, and not only the RDF data. For guaranteeing the completeness of the answers to a schema query, all the inference rules that we have given Figure **??** ( section describing RDFS) in Chapter **??** must be taken into account, and not only the subset that we considered in the forward-chaining algorithm we described.

We mentioned the practical advantage of separating the computation over a Tbox from that over the Abox. This is useful also from a theoretical point of view. This gives a bound on the data complexity (the complexity in terms of the Abox only) of consistency checking and of query answering. We showed that they can be performed using FOL queries and it is known [Var82, ARV95] that evaluating FOL queries over a relational database is in LOGSPACE in the size of the database.

**DL-lite.** DL-LITE$_{\mathcal{R}}$ has been recently incorporated into the version OWL2 [W3C09] of OWL as the profile called OWL2 QL. The proof that the *PerfectRef* Algorithm computes the *whole* answer is shown in [CGL$^+$07]. It follows that the complexity of query answering by reformulation in these fragments of DL-LITE is polynomial in the size of the Tbox, and in LOGSPACE in the size of the Abox.

A major result in [CGL$^+$07] is that DL-LITE$_{\mathcal{R}}$ and DL-LITE$_{\mathcal{F}}$ are two maximal fragments of the DL-Lite family supporting tractable query answering over large amounts of data. It has been shown in [CGL$^+$07] that consistency checking and instance recognition (which a particular case of query answering), while being LOGSPACE both for DL-LITE$_{\mathcal{R}}$ and DL-LITE$_{\mathcal{F}}$ Tboxes, are PTIME-COMPLETE for the languages that combine the axioms of both (denoted DL-LITE$_{\mathcal{RF}}$). This complexity result shows it is unlikely that an approach based on query reformulation would provide a complete query answering algorithm for DL-LITE$_{\mathcal{RF}}$.

QuOnto ([ACG$^+$05]) is a JAVA tool implementing the DL-Lite family of ontology representation languages. It permits the declaration of an ontology as a DL-LITE Tbox, the construction of an associated Abox that can be stored and as a MySQL database. The consistency checking of the resulting DL-LITE knowledge base, and query answering by reformulation are the core functionalities of QuOnto, based on the implementation in Java of the algorithms presented in this chapter.

**Datalog$^{+-}$.** Recent research [AGL09b, AGL09a] has extended the Datalog database query language towards query answering over ontologies. This has resulted in a unifying framework based on a family of expressive extensions of Datalog, called Datalog$^{+-}$, that captures DL-LITE$_{\mathcal{R}}$ and DL-LITE$_{\mathcal{F}}$.

[1]http://challenge.semanticweb.org/

@ Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, 2011; to be published by Cambridge University Press 2011.

## 6 Exercises

**Exercise 6.1** *With the University example, find a new query that has a different answer:*

1. *on the RDF data vs. the RDF data together with the RDFS ontology.*
2. *on the RDF data vs. the RDF data together with the* DL-LITE *ontology.*

**Exercise 6.2** *Prove that the* Saturation *algorithm runs in* $0((M \times N)^2)$.

**Exercise 6.3** *Prove that the rules used for computing the TBox closure are sound.*

**Exercise 6.4** *Consider Abox* $\mathcal{A}$ *in Figure 1 and* $\mathcal{A}'$ *obtained by deleting the fact* $PhDStudent(paul)$, *and the Tbox* $\mathcal{T}$ *in Figure 6. Show that:*

1. *The knowledge base* $\mathcal{A} \cup \mathcal{T}$ *is inconsistent.*
2. *The knowledge base* $\mathcal{A}' \cup \mathcal{T}$ *is consistent.*

**Exercise 6.5** *Give the FOL rule corresponding to the different cases of negative inclusion axioms.*

[ACG+05] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. Quonto: Querying ontologies. In *Proc. Intl. Conference on Artificial Intelligence (AAAI)*, 2005.

[AGL09a] A.Cali, G.Gottlob, and T Lukasiewicz. Datalog+-: a unified approach to ontologies and integrity constraints. In *Proc. Intl. Conf. on Database Theory (ICDT)*, 2009.

[AGL09b] A.Cali, G.Gottlob, and T Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *Proc. ACM Symp. on Principles of Database Systems (PODS)*, 2009.

[ARV95] S. Abiteboul, R.Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[CGL+07] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-LITE Family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[Jen] Jena - a semantic web framework for java. http://jena.sourceforge.net/.

[Var82] Moshe Vardi. The Complexity of Relational Query Languages. In *Proc. ACM SIGACT Symp. on the Theory of Computing (STOC)*, pages 137–146, 1982.

[W3C08] W3C. SPARQL query language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`, January 2008.

[W3C09] W3C. Owl 2 web ontology language profiles. `http://www.w3.org/2004/OWL/`, 2009.