SomeRDFS in the Semantic Web

P. Adjiman[1], F. Goasdoué[1], and M.-C. Rousset[2]

[1] LRI, bâtiment 490, Université Paris-Sud 11, 91405 Orsay Cedex, France
[2] LSR-IMAG, BP 72, 38402 St Martin d'Heres Cedex, France

Abstract. The Semantic Web envisions a world-wide distributed architecture where computational resources will easily inter-operate to coordinate complex tasks such as query answering. Semantic marking up of web resources using ontologies is expected to provide the necessary glue for making this vision work. Using ontology languages, (communities of) users will build their own ontologies in order to describe their own data. Adding semantic mappings between those ontologies, in order to semantically relate the data to share, gives rise to the Semantic Web: data on the web that are annotated by ontologies networked together by mappings. In this vision, the Semantic Web is a huge semantic peer data management system. In this paper, we describe the SomeRDFS peer data management systems that promote a "simple is beautiful" vision of the Semantic Web based on data annotated by RDFS ontologies.

1 Introduction

The Semantic Web [1] envisions a world-wide distributed architecture where computational resources will easily inter-operate to coordinate complex tasks such as query answering. Semantic marking up of web resources using *ontologies* is expected to provide the necessary glue for making this vision work.

Recent W3C efforts led to recommendations for annotating data with ontologies. The Resource Description Framework (RDF, http://www.w3.org/RDF) allows organizing data using simple taxonomies of classes and properties with RDF Schema (RDFS), the ontology language that comes with RDF. The Ontology Web Language (OWL, http://www.w3.org/2004/OWL) is defined on top of RDF and allows building more complex statements about data. It corresponds in its decidable versions (OWL-lite and OWL-DL) to expressive description logics.

Using ontology languages, (communities of) users will build their own ontologies in order to describe their own data. Adding semantic mappings between those ontologies, in order to semantically relate the data to share, gives rise to the Semantic Web: data on the web that are annotated by ontologies networked together by mappings. In this vision, the Semantic Web is a huge semantic peer data management system (PDMS).

Some PDMSs have been developped for the Semantic Web like Edutella [2], RDFPeers [3], GridVine [4] or SomeOWL [5]. Edutella, RDFPeers and GridVine use RDF while SomeOWL uses a fragment of OWL.
Edutella is made of a network of super-peers, the topology of which is a hypercube. Super-peers are mediators with the same schema: a reference ontology

(e.g., http://demoz.org). The data sources of a super-peer are its connected peers. Therefore, data are distributed over the peers while the ontologies are distributed over the super-peers. A peer must annotate its data in terms of the ontology of the super-peer to which it is connected. To answer queries, there is no need of mappings between the super-peer ontologies since they are identical: queries are efficiently routed in the network, using its topology of hypercube, in order to find super-peers that can provides answers with their peers.
In RDFPeers and GridVine PDMSs, the peers are organized according to a Distributed Hash Table using CHORD [6]. As in Edutella, such a fixed structure allows efficient routing of messages between peers. While RDFPeers only addresses the problem of query answering without taking into account the ontologies that annotate the data, GridVine takes into account the ontologies. On each GridVine peer, data are annotated with an RDFS ontology and mappings with ontologies of other peers are stated by equivalences between properties of peer ontologies.
In SomeOWL PDMSs[3] [5], peers are not organized according to a fixed topology: the topology is induced by the mappings between the peers ontologies. SomeOWL PDMSs are based on a simple data model: the ontologies and mappings are expressed in a fragment of OWL-DL that corresponds to the $\mathcal{CLU}$ description logic ($\neg$, $\sqcap$, and $\sqcup$). Query answering takes into account the ontologies and is achieved using a rewrite and evaluate strategy. The rewriting part is reduced to a consequence finding problem in distributed propositional theories. It is performed by SomeWhere, a peer-to-peer inference system that implements DeCA: DEcentralized Consequence finding Algorithm [5]. Query answering in a SomeOWL PDMS is sound, complete and terminates. Moreover, the detailed experiments reported in [7] show that it scales up to 1000 peers.

The contribution of this paper is to show how to deploy a PDMS using a data model based on RDF on top of the SomeWhere infrastructure: we will call such a PDMS a SomeRDFS PDMS. To express the ontologies and mappings, we consider the core fragment of RDFS allowing to state (sub)classes, (sub)properties, typing of domain and range of properties. A mapping is an inclusion statement between classes or properties of two distinct peers, or a typing statement of a property of a given peer with a class of another peer. Therefore, mappings are RDFS statements involving vocabularies of different peers which thus establish semantic correspondances between peers.

Like in a SomeOWL PDMS, the topology is induced by the mappings between the peers' ontologies. We show that query answering in SomeRDFS PDMSs can be achieved using a rewrite and evaluate strategy, and that the corresponding rewriting problem can be reduced to the same consequence finding problem in distributed propositional theories as in [5]. SomeWhere can then be used to compute the rewritings, with the same properties as mentionned above. We thus provide an operational solution for deploying a Semantic Web of data annotated with RDFS ontologies related by mappings. Moreover, the

[3] In this article, we denote by SomeOWL PDMSs the PDMSs based on OWL that have been designed in [5].

consequence finding problem resulting from the propositional encoding of the fragment of RDFS that we consider is tractable since the resulting propositional theories are reduced to clauses of length 2 for which the reasoning problem is in P. The experiments reported in [7] show that it takes in mean 0.07s to SomeWhere for a complete reasoning on randomly generated sets of clauses of length 2 distributed on 1000 peers.

The paper is organized as follows. Section 2 defines the fragment of RDFS that we consider as data model for SomeRDFS. Section 3 relates the problems of query answering and query rewriting, and shows how query rewriting can be reduced to a consequence finding problem in distributed propositional theories. Section 4 presents the query rewriting algorithm which is built on top of the DeCA algorithm of SomeWhere. We conclude with related work in Section 5 and a discussion in Section 6.

2 Data model of a SomeRDFS PDMS

We consider the core constructors of RDFS based on unary relations called *classes* and binary relations called *properties*. Those constructors are: class inclusion, property inclusion, and domain/range typing of a property. We denote this language core-RDFS.

While the logical semantics of the whole RDFS raises non trivial problems [8–11], core-RDFS has a first-order logical semantics which is clear and intuitive. This semantics can be defined in terms of interpretations or can be given by the first-order formulas expressing the logical meaning of each constructor. Based on the FOL semantics, it can be seen that core-RDFS is a fragment of *DL-Lite*$_R$, which is a description logic (DL) of the *DL-Lite* family [12, 13]. The *DL-Lite* family has been designed for allowing tractable query answering over data described w.r.t ontologies.

The following table provides the logical semantics of core-RDFS by giving the DL notation and the corresponding first-order logical (FOL) translation of the core-RDFS constructors.

Constructor	DL notation	FOL translation
Class inclusion	$C_1 \sqsubseteq C_2$	$\forall X(C_1(X) \Rightarrow C_2(X))$
Property inclusion	$P_1 \sqsubseteq P_2$	$\forall X \forall Y(P_1(X,Y) \Rightarrow P_2(X,Y))$
Domain typing of a property	$\exists P \sqsubseteq C$	$\forall X \forall Y(P(X,Y) \Rightarrow C(X))$
Range typing of a property	$\exists P^- \sqsubseteq C$	$\forall X \forall Y(P(X,Y) \Rightarrow C(Y))$

Ontologies, data descriptions and mappings of SomeRDFS peers are stated in core-RDFS. To make the semantics clear, we have chosen to use the FOL notation to denote ontologies, data and mappings as (possibly distributed) sets of FOL formulas. As seen in the previous table, the correspondence with the DL notation is obvious. It is important to note that core-RDFS belongs to the intersection of two logical languages that have been extensively studied: Horn rules without function and description logics. Therefore, core-RDFS is a fragment of DLP [14].

2.1 Peer ontologies

Peer ontologies are made of core-RDFS statements involving only relations of a *peer vocabulary.* A peer vocabulary is the union of a set of classe names and a set of property names that are disjoint. The class and property names are unique to each peer. We use the notation $\mathcal{P}{:}R$ for identifying the relation (class or property) R of the ontology of the peer $\mathcal{P}$.

2.2 Peer storage descriptions

The specification of the data stored in a peer is done through the declaration of assertional statements relating data of a peer to relations of its vocabulary. The DL notation and the FOL translation of assertional statements is the following (a and b are constants):

Constructor	DL notation and FOL translation
Class assertion	$C(a)$
Property assertion	$P(a,b)$

2.3 Peer mappings

Mappings are the key notion for establishing semantic connections between ontologies in order to share data. We define them as core-RDFS statements involving relations of two different peer vocabularies. The DL notation and the FOL translation of the mappings that are allowed in SomeRDFS are given in the following table.

Mappings between $\mathcal{P}_1$ and $\mathcal{P}_2$	DL notation	FOL translation
Class inclusion	$\mathcal{P}_1{:}C_1 \sqsubseteq \mathcal{P}_2{:}C_2$	$\forall X(\mathcal{P}_1{:}C_1(X) \Rightarrow \mathcal{P}_2{:}C_2(X))$
Property inclusion	$\mathcal{P}_1{:}P_1 \sqsubseteq \mathcal{P}_2{:}P_2$	$\forall X \forall Y(\mathcal{P}_1{:}P_1(X,Y) \Rightarrow \mathcal{P}_2{:}P_2(X,Y))$
Domain typing of a property	$\exists \mathcal{P}_1{:}P \sqsubseteq \mathcal{P}_2{:}C$	$\forall X \forall Y(\mathcal{P}_1{:}P(X,Y) \Rightarrow \mathcal{P}_2{:}C(X))$
Range typing of a property	$\exists \mathcal{P}_1{:}P^- \sqsubseteq \mathcal{P}_2{:}C$	$\forall X \forall Y(\mathcal{P}_1{:}P(X,Y) \Rightarrow \mathcal{P}_2{:}C(Y))$

The definition of *shared relations* follows from that of mappings.

Definition 1 (Shared relation). *A relation is* shared *between two peers if it belongs to the vocabulary of one peer and it appears in a mapping in the second peer.*

2.4 Schema & data of a SomeRDFS PDMS

In a PDMS, both the schema and data are distributed through respectively the union of the peer ontologies and mappings, and the union of the peer storage descriptions. The important point is that each peer has a partial knowledge of the PDMS. In a SomeRDFS PDMS, a peer just knows its ontology, its mappings with other peers and the relations shared with them, and its own data.
The schema of a SomeRDFS PDMS $\mathcal{S}$, denoted $schema(\mathcal{S})$, is the union of the ontologies and the sets of mappings of all the peers.
The data of a SomeRDFS PDMS $\mathcal{S}$, denoted $data(\mathcal{S})$, is the union of the peers data descriptions.
A SomeRDFS knowledge base is the union of its schema and data.

2.5 Queries

Many query languages have been recently developped for RDF [15] (e.g., RDQL, SPARQL,...). Most of them offer select-project-join queries which are known as the core relational query language in the database literature, namely the *conjunctive queries*.

Conjunctive queries can be expressed in first-order logic as open formulas with free variables $\bar{X}$ and made of conjunction of atoms. The free variables are called the *distinguished variables* of the query and they correspond to the variables of interest for the users. The other variables are existential variables that appear in the atoms of the query. The conjunction of atoms models the request of a user. For example, the following query expresses that the user is interested in knowing which artists have created paintings belonging to the cubism movement. The existential variable Y is just there to denote the existence of paintings created by the artist denoted by the variable X: the user wants to get as answers the instances of X satisfying the formula but he/she is not interested to know the instances of Y.

$$Q(X) \equiv \exists Y\ Artist(X) \wedge Creates(X,Y) \wedge Painting(Y) \wedge BelongsTo(Y, cubism)$$

The FOL translation for a conjunctive query is given in the following table.

Conjunctive query	FOL translation
$Q : \{(\bar{X}) \mid \bigwedge_{i=1}^{n} r_i(\bar{X}_i, \bar{Y}_i)\}$	$Q(\bar{X}) \equiv \exists \bar{Y} \bigwedge_{i=1}^{n} r_i(\bar{X}_i, \bar{Y}_i)$, where $\bar{X} = \bigcup_{i=1}^{n} \bar{X}_i$ are free variables and $\bar{Y} = \bigcup_{i=1}^{n} \bar{Y}_i$ are existential variables.

The most general queries that we will consider, the SomeRDFS *queries*, are conjunctive queries that may involve the vocabularies of several peers. In contrast, the users' queries involve the vocabulary of a single peer, since a user interrogates the PDMS through a peer of his choice.

Definition 2 (Query). *A* query *is a conjunctive query in terms of relations of peer vocabularies.*

Definition 3 (User query). *A* user query *is a query in terms of relations of a single peer vocabulary.*

2.6 Semantics

In a SomeRDFS PDMS, ontologies, storage descriptions, mappings, and queries have all a FOL correspondence. From a logical point of view, a SomeRDFS PDMS is a FOL knowledge base made of Horn rules (the schema) and ground atoms (the data). As mentioned before, it could also be equivalently seen as a DL knowledge base made of a Tbox (the schema) and an Abox (the data).

Two main semantics have been investigated for a PDMS based on FOL: the standard FOL semantics and the epistemic FOL semantics (see Section 5). Roughly

speaking, the standard FOL semantics does not distinguish mappings from ontology formulae, i.e., they are interpreted in the same way. In contrast, the epistemic FOL semantics restricts the expressivity of mapping formulae.
In a SomeRDFS PDMS, we adopt the standard FOL semantics. As for data, we stick to the usual information integration assumption, namely the *unique name assumption.*

Semantics of a query While a user query is given in terms of the relations of a single peer, its expected answers may be found all over the PDMS. An answer is a tuple made of constants stored in the PDMS for which it can be logically inferred (from the union of ontologies, storage descriptions and mappings) that it satisfies the expression defining the query. It corresponds to an extension of the notion of *certain answer* in information integration systems.

Definition 4 (Answer set of a query). *Let $\mathcal{S}$ be a* SomeRDFS *PDMS and Q a n-ary query. Let $\mathcal{C}$ be a set of constants appearing in data($\mathcal{S}$). The* answer set *of Q is: $Q(\mathcal{S}) = \{\bar{t} \in \mathcal{C}^n \mid \mathcal{S} \models Q(\bar{t})\}$.*

Subsumption The *subsumption* (a.k.a. containment) relation allows to compare two relations (classes, properties and queries) of the same arity.
A relation r_1 subsumes (respectively strictly subsumes) a relation r_2, iff for every interpretation I, $r_2^I \subseteq r_1^I$ (respectively $r_2^I \subset r_1^I$).
Given a SomeRDFS PDMS $\mathcal{S}$, a relation r_1 subsumes (respectively strictly subsumes) a relation r_2 w.r.t. $\mathcal{S}$, iff for every model I of *schema*($\mathcal{S}$), $r_2^I \subseteq r_1^I$ (respectively $r_2^I \subset r_1^I$).

2.7 Graphical conventions

In the following, we adopt some graphical conventions in order to represent a SomeRDFS schema.

A class is denoted by a node labeled with the class name. A property is a directed edge labeled with the property name. Such an edge is directed from the domain to the range of the property. A relation inclusion is denoted by a directed dotted edge labeled with the subsumption symbol $\sqsubseteq$. Such an edge is directed from the subsumee to the subsumer. A peer ontology is a subgraph totally contained in a box labeled with a peer name. A mapping is a directed edge from a peer ontology to another peer ontology. The owner of the mapping is the peer the box of which contains the corresponding edge label. Moreover, since there is no ambiguity with the owners of the ontology relations, we omit to prefix a relation name with its owner name in order to alleviate the notations.

For instance, let consider the mapping $\forall X(\mathcal{P}_1{:}Artifact(X) \Rightarrow \mathcal{P}_2{:}Works(X))$ between a peer $\mathcal{P}_1$ and a peer $\mathcal{P}_2$. This mapping is a class inclusion: the class $\mathcal{P}_1{:}Artifact$ of $\mathcal{P}_1$ is contained in the class $\mathcal{P}_2{:}Work$ of $\mathcal{P}_2$. If we suppose that it belongs to $\mathcal{P}_2$, its graphical notation is the one in Figure 1. In that case, $\mathcal{P}_1{:}Artifact$ is a shared relation between $\mathcal{P}_1$ and $\mathcal{P}_2$.

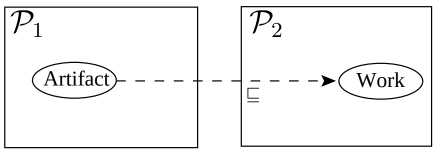


Fig. 1. Class inclusion mapping

Let consider another mapping $\forall X \forall Y(\mathcal{P}_1{:}paints(X,Y) \Rightarrow \mathcal{P}_2{:}painting(Y))$ between $\mathcal{P}_1$ and $\mathcal{P}_2$. This mapping is a range typing of the property $\mathcal{P}_1{:}paints$ of $\mathcal{P}_1$, the domain of which is typed with the class $\mathcal{P}_1{:}Painter$ of $\mathcal{P}_1$. The range typing is made with the class $\mathcal{P}_2{:}Painting$ of $\mathcal{P}_2$. If we suppose that this mapping belongs to $\mathcal{P}_1$, its graphical notation is the one in Figure 2. In that case, $\mathcal{P}_2{:}Painting$ is a shared relation between $\mathcal{P}_1$ and $\mathcal{P}_2$.

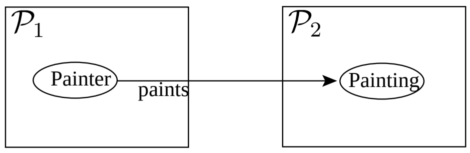


Fig. 2. Range typing mapping

2.8 Illustrative example

We will illustrate our contributions throughout the article on the following simple SomeRDFS $\mathcal{S}$ consisting of two peers $\mathcal{P}_1$ and $\mathcal{P}_2$.

$\mathcal{P}_1$ can store data about artists (some of them being sculptors and/or painters), artifacts artists have created, and the artistic movements the artifacts belong to. Some artist creations are distinguished according to whether their creators are sculptors or painters. $\mathcal{P}_2$ can store data about works (some of them being paintings, sculptures or musics) and the artistic period they refer to. The FOL notation of their ontologies is given in Figure 3.

$\mathcal{P}_1$ ontology	$\mathcal{P}_2$ ontology
$\forall X(\mathcal{P}_1{:}Sculptor(X) \Rightarrow \mathcal{P}_1{:}Artist(X))$	$\forall X(\mathcal{P}_2{:}Painting(X) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X(\mathcal{P}_1{:}Painter(X) \Rightarrow \mathcal{P}_1{:}Artist(X))$	$\forall X(\mathcal{P}_2{:}Sculpture(X) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X \forall Y(\mathcal{P}_1{:}creates(X,Y) \Rightarrow \mathcal{P}_1{:}Artist(X))$	$\forall X(\mathcal{P}_2{:}Music(X) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X \forall Y(\mathcal{P}_1{:}creates(X,Y) \Rightarrow \mathcal{P}_1{:}Artifact(Y))$	$\forall X \forall Y(\mathcal{P}_2{:}refersTo(X,Y) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X \forall Y(\mathcal{P}_1{:}paints(X,Y) \Rightarrow \mathcal{P}_1{:}creates(X,Y))$	$\forall X \forall Y(\mathcal{P}_2{:}refersTo(X,Y) \Rightarrow \mathcal{P}_2{:}Period(Y))$
$\forall X \forall Y(\mathcal{P}_1{:}sculpts(X,Y) \Rightarrow \mathcal{P}_1{:}creates(X,Y))$	
$\forall X \forall Y(\mathcal{P}_1{:}sculpts(X,Y) \Rightarrow \mathcal{P}_1{:}Sculptor(X))$	
$\forall X \forall Y(\mathcal{P}_1{:}paints(X,Y) \Rightarrow \mathcal{P}_1{:}Painter(X))$	
$\forall X \forall Y(\mathcal{P}_1{:}belongsTo(X,Y) \Rightarrow \mathcal{P}_1{:}Artifact(X))$	
$\forall X \forall Y(\mathcal{P}_1{:}belongsTo(X,Y) \Rightarrow \mathcal{P}_1{:}Movement(Y))$	

Fig. 3. Ontologies of $\mathcal{P}_1$ and $\mathcal{P}_2$

$\mathcal{P}_1$ actually stores that Picasso has painted “Les demoiselles d’Avignon” which belongs to the Picasso’s pink movement, and has sculpted “La femme au chapeau” which belongs to the Modern art movement. $\mathcal{P}_2$ stores that “Le déjeuner des canotiers" is a painting and that "Les demoiselles d'Avignon" refers to the Cubism artistic period. It also stores that "The statue of David" is a sculpture and that "Nutcracker" is a music. The storage description of $\mathcal{P}_1$ and $\mathcal{P}_2$ is given in Figure 4.

$\mathcal{P}_1$ Data	$\mathcal{P}_2$ Data
$\mathcal{P}_1$:*paints*(Picasso,Les-demoiselles-d-Avignon)	$\mathcal{P}_2$:*Painting*(Le-dejeuner-des-canotiers)
$\mathcal{P}_1$:*sculpts*(Picasso,La-femme-au-chapeau)	$\mathcal{P}_2$:*refersTo*(Les-demoiselles-d-Avignon,Cubism)
$\mathcal{P}_1$:*belongsTo*(Les-demoiselles-d-Avignon,Picasso-pink)	$\mathcal{P}_2$:*Sculpture*(The-statue-of-David)
$\mathcal{P}_1$:*belongsTo*(La-femme-au-chapeau,Modern-art)	$\mathcal{P}_2$:*Music*(Nutcracker)

Fig. 4. Data of $\mathcal{P}_1$ and $\mathcal{P}_2$

In order to share data with $\mathcal{P}_2$, $\mathcal{P}_1$ has established two mappings to distinguish sculptor and painter creations from artist creations. This is done by the range typing of $\mathcal{P}_1$:*sculpts* and $\mathcal{P}_1$:*paints* with respectively the classes of sculptures and paintings of $\mathcal{P}_2$. $\mathcal{P}_2$ has also established mappings with $\mathcal{P}_1$ in order to state that the class of artifacts of $\mathcal{P}_1$ is contained in its class of artistic works, and that the property $\mathcal{P}_1$:*belongsTo* is contained in its property $\mathcal{P}_2$:*refersTo*. Their mappings are given in Figure 5. Those mappings indicate that the shared relations of $\mathcal{P}_1$ are $\mathcal{P}_1$:*Artifact* and $\mathcal{P}_1$:*belongsTo*, while the shared relations of $\mathcal{P}_2$ are $\mathcal{P}_2$:*Painting* and $\mathcal{P}_2$:*Sculpture*.

$\mathcal{P}_1$ mappings	$\mathcal{P}_2$ mappings
$\forall X \forall Y(\mathcal{P}_1{:}paints(X,Y) \Rightarrow \mathcal{P}_2{:}Painting(Y))$	$\forall X(\mathcal{P}_1{:}Artifact(X) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X \forall Y(\mathcal{P}_1{:}sculpts(X,Y) \Rightarrow \mathcal{P}_2{:}Sculpture(Y))$	$\forall X \forall Y(\mathcal{P}_1{:}belongsTo(X,Y) \Rightarrow \mathcal{P}_2{:}refersTo(X,Y))$

Fig. 5. Mappings of $\mathcal{P}_1$ and $\mathcal{P}_2$

Following our graphical conventions, the above SOMERDFS schema is given in Figure 6.

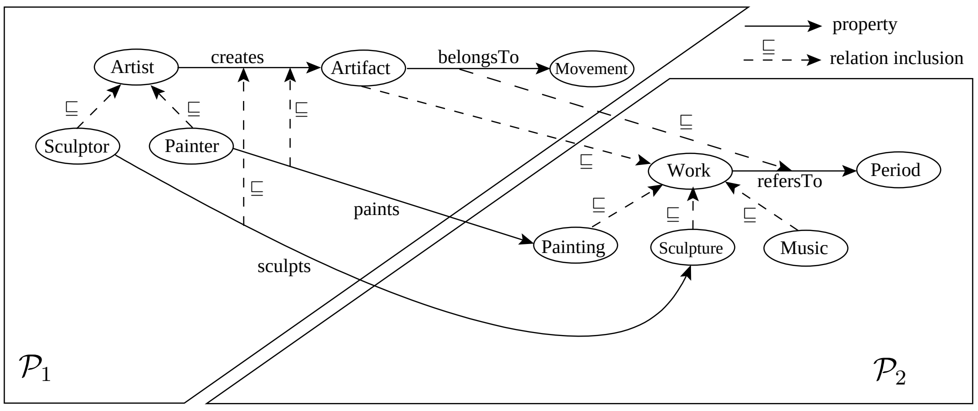


Fig. 6. Graphical representation of *schema*($\mathcal{S}$)

3 Query answering through query rewriting

Query answering is the main inference in a PDMS. Finding all the answers of a user query is, in general, a critical issue [16]. It has been shown in [17] that when a query has a finite number of *maximal conjunctive rewritings*, then its answer set can be obtained as the union of the answer sets of its rewritings.

Definition 5 (Conjunctive rewriting). *Given a* SomeRDFS *PDMS $\mathcal{S}$, a query R is a conjunctive rewriting of a query Q iff Q subsumes R w.r.t. $\mathcal{S}$. R is a maximal conjunctive rewriting of Q if there does not exist another conjunctive rewriting R' of Q strictly subsuming R.*

Theorem 1 shows that query answering in a SomeRDFS PDMS can be done through query rewriting.

Theorem 1. *Query answering of a user query can be achieved by a rewrite and evaluate strategy. The rewriting complexity is polynomial w.r.t. the size of the schema of the* SomeRDFS *PDMS and exponential w.r.t. the number of atoms in the query. The evaluation complexity is polynomial w.r.t. the size of the data of the* SomeRDFS *PDMS.*

Proof. The schema of a SomeRDFS PDMS forms a knowledge base $\mathcal{R}$ of function-free Horn rules with single conditions (see the FOL axiomatization of core-RDFS in Figure 2). A simple backward chaining algorithm [18] with cycle detection applied to each atom of a user query Q ensures to find all the maximal conjunctive rewritings of each atom of Q with atmost n chaining steps, if n is the number of rules in the schema. The reason is that each rule in a schema can only be used at most once (assuming cycle detections) because they have a single condition. Therefore, there are at most n maximal conjunctive rewritings (each one being reduced to one atom) for each of the k atoms of the user query.

It follows from [19] that when views, queries and rewritings are conjunctive queries, the set of all the (maximal) rewritings using views of a query can be obtained from the conjunctions of the rewritings of each atom of the query. In order to apply that result to our setting, we just have to reformulate the rewriting problem that we consider into the rewriting problem using views considered in [19]. For doing so, for each atom $p(\bar{X})$ we create a view, named $p(\bar{X})$, the body of which is the conjunction of the different atoms that can be derived (e.g., by standard forward-chaining) from $p(\bar{X})$ using the set $\mathcal{R}$ of rule:

$$p(\bar{X}) \equiv \bigwedge_{\{p(\bar{X})\} \cup \mathcal{R} \vdash a(\bar{Y})} a(\bar{Y}).$$

Those views have no existential variable (for every $a(\bar{Y})$ such that $\{p(\bar{X})\} \cup \mathcal{R} \vdash a(\bar{Y})$, $\bar{Y} \subseteq \bar{X}$) because the FOL axiomatization of core-RDFS (see Figure 2) is made of safe rules only. Therefore, conjuncting views that are relevant to each atom of the query provides rewritings of the query. As shown in [20], it is not true in the general case where the conjunctions of views relevant to each atom of the query are just candidate rewritings for which subsumption with the query must be checked.

By construction, there are at most n views relevant for each atom of the query. Therefore, there are at most n^k maximal conjunctive rewritings of the user query, obtained by conjuncting rewritings of each atom of the query. Therefore, rewriting complexity in SomeRDFS is in $O(n^k)$. Note that in practice the value of n might be quite large while the one of k should be small.
Finally, evaluating a conjunctive query is in P w.r.t. data complexity [21]. Since a user query has a finite number of maximal conjunctive rewritings to evaluate, answering such a query is in P w.r.t. data complexity. $\square$

The proof of Theorem 1 provides a solution in order to deploy a SomeRDFS PDMS: one needs a peer-to-peer reasoner that performs backward chaining in distributed knowledge bases of FOL function-free Horn rules. To the best of our knowledge such a FOL peer-to-peer reasoner does not exist. However, there exists a propositional peer-to-peer reasoner: SomeWhere [5]. We will show that it is possible to encode FOL reasoning in SomeRDFS into propositional reasoning in SomeWhere.

Before presenting the corresponding reduction, we illustrate the rewrite and evaluate strategy for query answering in a SomeRDFS PDMS on the example of Section 2.8.

3.1 Illustrative example (continued)

Let us consider the user query $Q_1(X) \equiv \mathcal{P}_2{:}Work(X)$ asked to the peer $\mathcal{P}_2$. It is easy to see that its maximal rewritings are (e.g., using backward chaining on the Horn rules of *schema*$(\mathcal{S})$):

1. $R^1_1(X) \equiv \mathcal{P}_2{:}Work(X)$
2. $R^1_2(X) \equiv \mathcal{P}_2{:}Painting(X)$
3. $R^1_3(X) \equiv \mathcal{P}_2{:}Sculpture(X)$
4. $R^1_4(X) \equiv \mathcal{P}_2{:}Music(X)$
5. $R^1_5(X) \equiv \exists Y\ \mathcal{P}_2{:}refersTo(X, Y)$
6. $R^1_6(X) \equiv \mathcal{P}_1{:}Artifact(X)$
7. $R^1_7(X) \equiv \exists Y\ \mathcal{P}_1{:}belongsTo(X, Y)$
8. $R^1_8(X) \equiv \exists Y\ \mathcal{P}_1{:}creates(Y, X)$
9. $R^1_9(X) \equiv \exists Y\ \mathcal{P}_1{:}paints(Y, X)$
10. $R^1_{10}(X) \equiv \exists Y\ \mathcal{P}_1{:}sculpts(Y, X)$

The answer set of Q_1 is obtained by evaluating those rewritings.

$$Q_1(\mathcal{S}) = \underbrace{\emptyset}_{R^1_1(\mathcal{S})} \cup \underbrace{\{\text{Le-dejeuner-des-canotiers}\}}_{R^1_2(\mathcal{S})} \cup \underbrace{\{\text{The-statue-of-David}\}}_{R^1_3(\mathcal{S})}$$
$$\cup \underbrace{\{\text{Nutcracker}\}}_{R^1_4(\mathcal{S})} \cup \underbrace{\{\text{Les-demoiselles-d-Avignon}\}}_{R^1_5(\mathcal{S})} \cup \underbrace{\emptyset}_{R^1_6(\mathcal{S})}$$
$$\cup \underbrace{\{\text{Les-demoiselles-d-Avignon,La-femme-au-chapeau}\}}_{R^1_7(\mathcal{S})} \cup \underbrace{\emptyset}_{R^1_8(\mathcal{S})}$$

$$\cup \underbrace{\{\text{Les-demoiselles-d-Avignon}\}}_{R_9^1(\mathcal{S})} \cup \underbrace{\{\text{La-femme-au-chapeau}\}}_{R_{10}^1(\mathcal{S})}.$$

Consider now the user query $Q_2(X,Y) \equiv \mathcal{P}_2{:}Painting(X) \wedge \mathcal{P}_2{:}refersTo(X,Y)$ asked to $\mathcal{P}_2$. Its maximal rewritings are:

1. $R_1^2(X,Y) \equiv \mathcal{P}_2{:}Painting(X) \wedge \mathcal{P}_2{:}refersTo(X,Y)$
2. $R_2^2(X,Y) \equiv \mathcal{P}_2{:}Painting(X) \wedge \mathcal{P}_1{:}belongsTo(X,Y)$
3. $R_3^2(X,Y) \equiv \exists Z\ \mathcal{P}_1{:}paints(Z,X) \wedge \mathcal{P}_2{:}refersTo(X,Y)$
4. $R_4^2(X,Y) \equiv \exists Z\ \mathcal{P}_1{:}paints(Z,X) \wedge \mathcal{P}_1{:}belongsTo(X,Y)$

The answer set of Q_2 is obtained by evaluating those rewritings.

$$Q_2(\mathcal{S}) = \underbrace{\emptyset}_{R_1^2(\mathcal{S})} \cup \underbrace{\emptyset}_{R_2^2(\mathcal{S})} \cup \underbrace{\{(\text{Les-demoiselles-d-Avignon, Cubism})\}}_{R_3^2(\mathcal{S})}$$
$$\cup \underbrace{\{(\text{Les-demoiselles-d-Avignon, Picasso-pink})\}}_{R_4^2(\mathcal{S})}.$$

Note that the above rewritings suggest the need of optimization in order to be efficiently evaluated: some atoms appear in several rewritings and are thus evaluated several times. Standard caching techniques can be used for that purpose.

Q_1 and Q_2 highlight three kinds of rewritings.

- *Local rewritings* involve relations of the queried peer's vocabulary. For example, the rewriting R_4^1 shows that the data Nutcraker of $\mathcal{P}_2$, which is known as music, is an artistic work.
- *Distant rewritings* involve relations of a single distant peer's vocabulary. For example, the rewriting R_4^2 shows that Les demoiselles d'Avignon is a painting that refers to Cubism. It is worth noticing that Les demoiselles d'Avignon is already known from $\mathcal{P}_2$, but not as a painting.
- *Integration rewritings* involve relations of several peer's vocabularies. For example, the rewriting R_3^2 shows that Les demoiselles d'Avignon which is already known from $\mathcal{P}_2$ to refer to Cubism, refers also to the Pink period of Picasso.

3.2 Propositional reduction of query rewriting in a SomeRDFS PDMS

In this section, we describe how to equivalently reduce query rewriting in a SomeRDFS PDMS to consequence finding over logical propositional theories in SomeWhere. To do this, we have to convert the distributed FOL knowledge base that corresponds to a SomeRDFS PDMS into a distributed propositional theory T that corresponds to a SomeWhere peer-to-peer inference system, and to show that we obtain the maximal conjunctive rewritings of a query $Q(\bar{X})$ from the proper prime implicates of $\neg Q$ w.r.t. T using the DeCA algorithm of SomeWhere.

SomeWhere [5] is a peer-to-peer inference system (P2PIS), in which each peer theory is a set of propositional clauses built from a set of propositional variables. Any variable common to two connected peers can be stated as *shared.* In that case, both peers know that they share that variable. Any variable of a peer's vocabulary can also be stated as *target.* When a peer is solicited for computing consequences, the consequences it can send back are only those that contain target variables. From a logical point of view, the global theory of a SomeWhere P2PIS is the union of the propositional theories of its peers. From a reasoning point of view, each SomeWhere peer runs DeCA [5] (DEcentralized Consequence finding Algorithm), which is a message-passing algorithm that computes the proper prime implicates of literals w.r.t. the global theory of the P2PIS. The point is that it does it in a fully decentralized manner, without knowing the whole global theory. DeCA is sound, i.e., it computes only proper implicates of the input literal w.r.t. the global theory. DeCA always terminates and notifies the user of its termination. We have exhibited in [5] a sufficient condition for DeCA to be complete, i.e., to return *all* the *proper prime implicates* of the input literal (w.r.t. the global theory): for any two peers having a variable in common, there is a path of connected peers sharing that variable.

The following definition recalls the notion of proper prime implicate of a clause w.r.t. a propositional clausal theory.

Definition 6 (Proper prime implicate w.r.t. a theory). *Let T be a clausal theory and q be a clause. A clause m is said to be:*

- *a* prime implicate *of q w.r.t. T iff $T \cup \{q\} \models m$ and for any other clause m', if $T \cup \{q\} \models m'$ and $m' \models m$ then $m' \equiv m$.*
- *a* proper prime implicate *of q w.r.t. T iff it is a prime implicate of q w.r.t. T and $T \not\models m$.*

The propositional encoding of a SomeRDFS that we consider is given in Definition 7. It must translate appropriately the semantic connection between classes and properties. In particular, the typing of the properties must distinguish the typing of the domain from the typing of the range of a given property P. As we will see, this distinction is important for rebuilding the FOL rewritings from the propositional rewritings. In the FOL notation, the distinction relies on the place of the typing variable as argument of the property: in $\forall X \forall Y(P(X, Y) \Rightarrow C(X))$ the fact that the typing variable (i.e., X) appears as the first argument of P indicates that the *domain* of the property is typed by the class C, while in $\forall X \forall Y(P(X, Y) \Rightarrow C(Y))$ the fact that the typing variable (i.e., Y) appears as the second argument of P indicates that the *range* of the property is typed by the class C.

Therefore, for a given class C, we distinguish its two typing roles for properties by encoding it by two propositional variables C^{dom} and C^{range}. Thus, we encode the domain typing of a property P by a class C with the clausal form $\neg P \vee C^{dom}$ of the implication $P \Rightarrow C^{dom}$. Similarly, we encode the range typing of a property P by a class C with the clausal form $\neg P \vee C^{range}$ of the implication $P \Rightarrow C^{range}$.

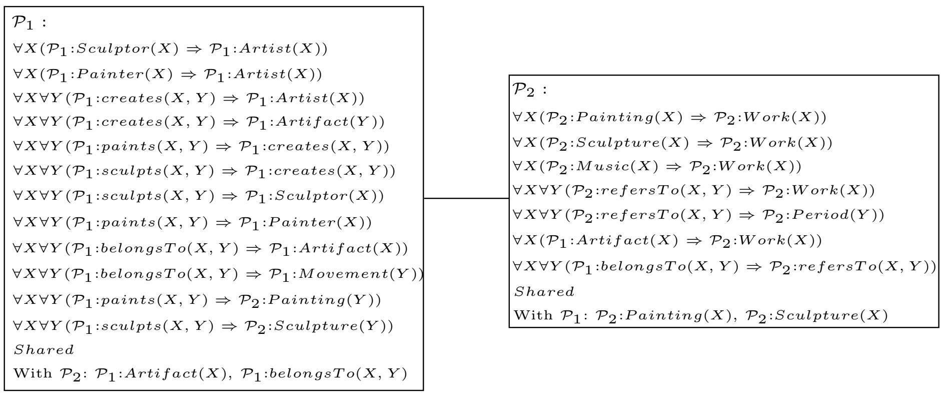
Definition 7 (Propositional encoding of SOMERDFS**).** *We encode a* SOMERDFS $\mathcal{S}$ *into a* SOMEWHERE *Prop*($\mathcal{S}$) *by encoding each* SOMERDFS *peer* $\mathcal{P}$ *in* $\mathcal{S}$ *into a* SOMEWHERE *peer Prop*($\mathcal{P}$) *in Prop*($\mathcal{S}$)*:*

- *if* $\forall X(C_1(X) \Rightarrow C_2(X))$ *is in* $\mathcal{P}$, $\neg C_1^{dom} \vee C_2^{dom}$ *and* $\neg C_1^{range} \vee C_2^{range}$ *are in Prop*($\mathcal{P}$)*.*
- *if* $\forall X, Y(P(X,Y) \Rightarrow C(X))$ *is in* $\mathcal{P}$, $\neg P^{prop} \vee C^{dom}$ *is in Prop*($\mathcal{P}$)*.*
- *if* $\forall X, Y(P(X,Y) \Rightarrow C(Y))$ *is in* $\mathcal{P}$, $\neg P^{prop} \vee C^{range}$ *is in Prop*($\mathcal{P}$)*.*
- *if* $\forall X, Y(P_1(X,Y) \Rightarrow P_2(X,Y))$ *is in* $\mathcal{P}$, $\neg P_1^{prop} \vee P_2^{prop}$ *is in Prop*($\mathcal{P}$)*.*

It is important to notice that a dual encoding would have been possible, consisting in distinguishing the domain and range typing by encoding each property P by two propositional variables P^{dom} and P^{range}: a clause $\neg P^{dom} \vee C$ would encode the domain typing of a property P by a class C, while $\neg P^{range} \vee C$ would encode the range typing of a property P by a class C.

All the propositional variables in *Prop*($\mathcal{P}$) are stated as *target*. A variable in *Prop*($\mathcal{P}$) that corresponds to a relation shared with another peer $\mathcal{P}'$, is stated as *shared* with *Prop*($\mathcal{P}'$).

As an illustration, Figure 7 represents the two peers of the SOMERDFS PDMS introduced in Section 2.8.



$\mathcal{P}_1$:
$\forall X(\mathcal{P}_1{:}Sculptor(X) \Rightarrow \mathcal{P}_1{:}Artist(X))$
$\forall X(\mathcal{P}_1{:}Painter(X) \Rightarrow \mathcal{P}_1{:}Artist(X))$
$\forall X \forall Y(\mathcal{P}_1{:}creates(X,Y) \Rightarrow \mathcal{P}_1{:}Artist(X))$
$\forall X \forall Y(\mathcal{P}_1{:}creates(X,Y) \Rightarrow \mathcal{P}_1{:}Artifact(Y))$
$\forall X \forall Y(\mathcal{P}_1{:}paints(X,Y) \Rightarrow \mathcal{P}_1{:}creates(X,Y))$
$\forall X \forall Y(\mathcal{P}_1{:}sculpts(X,Y) \Rightarrow \mathcal{P}_1{:}creates(X,Y))$
$\forall X \forall Y(\mathcal{P}_1{:}sculpts(X,Y) \Rightarrow \mathcal{P}_1{:}Sculptor(X))$
$\forall X \forall Y(\mathcal{P}_1{:}paints(X,Y) \Rightarrow \mathcal{P}_1{:}Painter(X))$
$\forall X \forall Y(\mathcal{P}_1{:}belongsTo(X,Y) \Rightarrow \mathcal{P}_1{:}Artifact(X))$
$\forall X \forall Y(\mathcal{P}_1{:}belongsTo(X,Y) \Rightarrow \mathcal{P}_1{:}Movement(Y))$
$\forall X \forall Y(\mathcal{P}_1{:}paints(X,Y) \Rightarrow \mathcal{P}_2{:}Painting(Y))$
$\forall X \forall Y(\mathcal{P}_1{:}sculpts(X,Y) \Rightarrow \mathcal{P}_2{:}Sculpture(Y))$
Shared
With $\mathcal{P}_2$: $\mathcal{P}_1{:}Artifact(X)$, $\mathcal{P}_1{:}belongsTo(X,Y)$

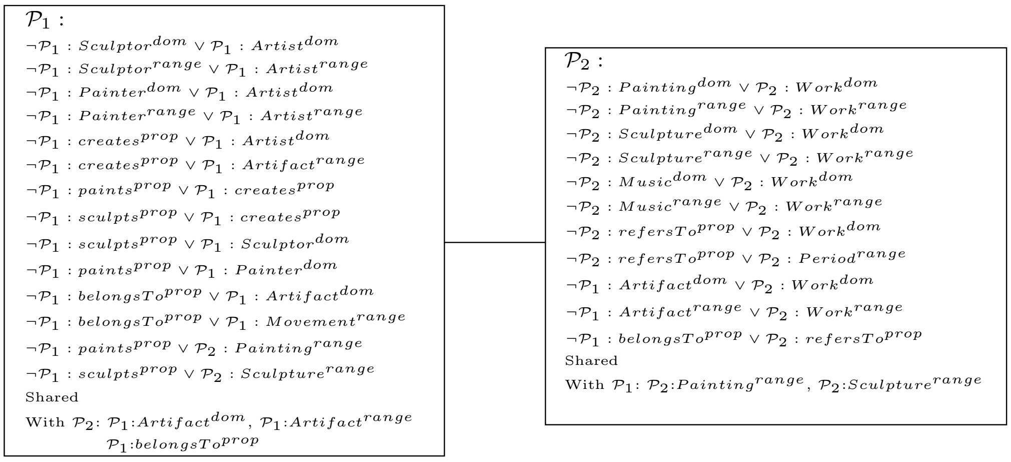
$\mathcal{P}_2$:
$\forall X(\mathcal{P}_2{:}Painting(X) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X(\mathcal{P}_2{:}Sculpture(X) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X(\mathcal{P}_2{:}Music(X) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X \forall Y(\mathcal{P}_2{:}refersTo(X,Y) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X \forall Y(\mathcal{P}_2{:}refersTo(X,Y) \Rightarrow \mathcal{P}_2{:}Period(Y))$
$\forall X(\mathcal{P}_1{:}Artifact(X) \Rightarrow \mathcal{P}_2{:}Work(X))$
$\forall X \forall Y(\mathcal{P}_1{:}belongsTo(X,Y) \Rightarrow \mathcal{P}_2{:}refersTo(X,Y))$
Shared
With $\mathcal{P}_1$: $\mathcal{P}_2{:}Painting(X)$, $\mathcal{P}_2{:}Sculpture(X)$

Fig. 7. The SOMERDFS PDMS of Section 2.8

Figure 8 corresponds to the encoding of this SOMERDFS PDMS into a SOMEWHERE P2PIS. The *Shared* section in the Figure 7 (resp. 8) makes explicit which local relations (resp. propositional variables) are known to be shared with others peers.

Proposition 1 states that the propositional encoding of a SOMERDFS PDMS leads to a SOMEWHERE P2PIS for which the DECA algorithm is complete.

Proposition 1 (Completeness of DECA **for the propositional encoding of a** SOMERDFS **PDMS).** *Let* $\mathcal{S}$ *be a* SOMERDFS *PDMS. Let Prop*($\mathcal{S}$) *be the* SOMEWHERE *P2PIS resulting from the propositional encoding of* $\mathcal{S}$. DECA *is complete for Prop*($\mathcal{S}$)*.*



$\mathcal{P}_1$:

$\neg\mathcal{P}_1 : Sculptor^{dom} \vee \mathcal{P}_1 : Artist^{dom}$
$\neg\mathcal{P}_1 : Sculptor^{range} \vee \mathcal{P}_1 : Artist^{range}$
$\neg\mathcal{P}_1 : Painter^{dom} \vee \mathcal{P}_1 : Artist^{dom}$
$\neg\mathcal{P}_1 : Painter^{range} \vee \mathcal{P}_1 : Artist^{range}$
$\neg\mathcal{P}_1 : creates^{prop} \vee \mathcal{P}_1 : Artist^{dom}$
$\neg\mathcal{P}_1 : creates^{prop} \vee \mathcal{P}_1 : Artifact^{range}$
$\neg\mathcal{P}_1 : paints^{prop} \vee \mathcal{P}_1 : creates^{prop}$
$\neg\mathcal{P}_1 : sculpts^{prop} \vee \mathcal{P}_1 : creates^{prop}$
$\neg\mathcal{P}_1 : sculpts^{prop} \vee \mathcal{P}_1 : Sculptor^{dom}$
$\neg\mathcal{P}_1 : paints^{prop} \vee \mathcal{P}_1 : Painter^{dom}$
$\neg\mathcal{P}_1 : belongsTo^{prop} \vee \mathcal{P}_1 : Artifact^{dom}$
$\neg\mathcal{P}_1 : belongsTo^{prop} \vee \mathcal{P}_1 : Movement^{range}$
$\neg\mathcal{P}_1 : paints^{prop} \vee \mathcal{P}_2 : Painting^{range}$
$\neg\mathcal{P}_1 : sculpts^{prop} \vee \mathcal{P}_2 : Sculpture^{range}$
Shared
With $\mathcal{P}_2$: $\mathcal{P}_1{:}Artifact^{dom}$, $\mathcal{P}_1{:}Artifact^{range}$
$\mathcal{P}_1{:}belongsTo^{prop}$

$\mathcal{P}_2$:

$\neg\mathcal{P}_2 : Painting^{dom} \vee \mathcal{P}_2 : Work^{dom}$
$\neg\mathcal{P}_2 : Painting^{range} \vee \mathcal{P}_2 : Work^{range}$
$\neg\mathcal{P}_2 : Sculpture^{dom} \vee \mathcal{P}_2 : Work^{dom}$
$\neg\mathcal{P}_2 : Sculpture^{range} \vee \mathcal{P}_2 : Work^{range}$
$\neg\mathcal{P}_2 : Music^{dom} \vee \mathcal{P}_2 : Work^{dom}$
$\neg\mathcal{P}_2 : Music^{range} \vee \mathcal{P}_2 : Work^{range}$
$\neg\mathcal{P}_2 : refersTo^{prop} \vee \mathcal{P}_2 : Work^{dom}$
$\neg\mathcal{P}_2 : refersTo^{prop} \vee \mathcal{P}_2 : Period^{range}$
$\neg\mathcal{P}_1 : Artifact^{dom} \vee \mathcal{P}_2 : Work^{dom}$
$\neg\mathcal{P}_1 : Artifact^{range} \vee \mathcal{P}_2 : Work^{range}$
$\neg\mathcal{P}_1 : belongsTo^{prop} \vee \mathcal{P}_2 : refersTo^{prop}$
Shared
With $\mathcal{P}_1$: $\mathcal{P}_2{:}Painting^{range}$, $\mathcal{P}_2{:}Sculpture^{range}$

Fig. 8. Propositional encoding of the SomeRDFS PDMS of Section 2.8

Proof. By definition, in a SomeRDFS PDMS, a relation which appears in two peers comes from a mapping between those two peers and is shared between those two peers. In the propositional encoding, the only variables that can be common to two peer theories result from the encoding of a mapping. Therefore, in a SomeWhere P2PIS resulting from the encoding of a SomeRDFS PDMS, all the variables that are common to two peer theories are necessarily shared by those two peers, and the sufficient condition for the completeness of DeCA is obviously satisfied. □

Proposition 2 establishes the connection between maximal conjunctive rewritings of queries made of a single atom in a SomeRDFS PDMS and proper prime implicates of a literal in a SomeWhere P2PIS. Note that Proposition 2 also suggests an optimization of the propositional encoding when query rewriting is used for query answering: the target variables should be only the ones resulting from relations for which facts are stored. Doing this, each conjunctive rewriting will be useful for query answering: it will provide at least one answer.

Proposition 2 (Propositional transfer). *Let $\mathcal{S}$ be a* SomeRDFS *PDMS and let Prop($\mathcal{S}$) be its propositional encoding into a* SomeWhere *P2PIS.*

(i) $R(X) \equiv C'(X)$ *is a maximal conjunctive rewriting of a query* $Q(X) \equiv C(X)$ *w.r.t.* $\mathcal{S}$ *iff* $\neg C'^{dom}$ *is a proper prime implicate of* $\neg C^{dom}$ *w.r.t. Prop($\mathcal{S}$)*
(ii) $R(X) \equiv \exists Y P(X, Y)$ *is a maximal conjunctive rewriting of a query* $Q(X) \equiv C(X)$ *w.r.t.* $\mathcal{S}$ *iff* $\neg P^{prop}$ *is a proper prime implicate of* $\neg C^{dom}$ *w.r.t. Prop($\mathcal{S}$).*
(iii) $R(X) \equiv \exists Y P(Y, X)$ *is a maximal conjunctive rewriting of a query* $Q(X) \equiv C(X)$ *w.r.t.* $\mathcal{S}$ *iff* $\neg P^{prop}$ *is a proper prime implicate of* $\neg C^{range}$ *w.r.t. Prop($\mathcal{S}$).*
(iv) $R(X, Y) \equiv P'(X, Y)$ *is a maximal conjunctive rewriting of a query* $Q(X, Y) \equiv P(X, Y)$ *w.r.t.* $\mathcal{S}$ *iff* $\neg P'^{prop}$ *is a proper prime implicate of* $\neg P^{prop}$ *w.r.t. Prop($\mathcal{S}$).*

Proof. We first exhibit some properties that will be used in the proof of the proposition. Let $\mathcal{S}$ be a SomeRDFS PDMS and let $Prop(\mathcal{S})$ be its propositional encoding into a SomeWhere P2PIS. Let C and P be respectively a class and a property of $\mathcal{S}$, and C^{dom}, C^{range}, and P^{prop} be their corresponding variables in $Prop(\mathcal{S})$.

Let $I = (\Delta^I, .^I)$ be an interpretation of $\mathcal{S}$ and $(o, o') \in \Delta^I \times \Delta^I$. We build an interpretation $p_{o,o'}(I)$ of $Prop(\mathcal{S})$ as follows:

α_1. $(C^{dom})^{p_{o,o'}(I)} = true$ iff $o \in C^I$ and $(C^{range})^{p_{o,o'}(I)} = true$ iff $o' \in C^I$.
α_2. $(P^{prop})^{p_{o,o'}(I)} = true$ iff $(o, o') \in P^I$.

Let J be an interpretation of $Prop(\mathcal{S})$. We build $i(J) = (\Delta^I = \{dom, range\}, .^{i(J)})$ an interpretation of $\mathcal{S}$ as follows:

β_1. $dom \in C^{i(J)}$ iff $(C^{dom})^J = true$ and $range \in C^{i(J)}$ iff $(C^{range})^J = true$.
β_2. if $(P^{prop})^J = true$ then $R^{i(J)} = \{(dom, range)\}$ else $R^{i(J)} = \emptyset$.

Properties 1. and 2. follow from the definition of the above interpretations:
For every interpretation I of $\mathcal{S}$ and $(o, o') \in \Delta^I \times \Delta^I$, for every interpretation J of $Prop(\mathcal{S})$:

1. I is a model of $\mathcal{S}$ iff $p_{o,o'}(I)$ is a model of $Prop(\mathcal{S})$.
2. $i(J)$ is a model of $\mathcal{S}$ iff J is a model of $Prop(\mathcal{S})$.

We now give the proof of the item (i) of the proposition. We do not provide the proofs of the items (ii), (iii) and (iv) because they are very similar to that of (i).

(i) $(\Leftarrow)$ We have to prove that if $\neg C'^{dom}$ is a proper prime implicate of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$ then $R(X) \equiv C'(X)$ is a maximal conjunctive rewriting of a query $Q(X) \equiv C(X)$ w.r.t. $\mathcal{S}$.

Suppose that $\neg C'^{dom}$ is a proper prime implicate of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$. Let us first show that $R(X) \equiv C'(X)$ is a conjunctive rewriting of $Q(X) \equiv C(X)$ w.r.t. $\mathcal{S}$. If it is false, then there exists a model I of $\mathcal{S}$ and a constant b such that $b \in C'^I$ and $b \notin C^I$. Note that there always exists such a b since the core-RDFS data model does not allow building unsatisfiable logical sentences (w.r.t. $\mathcal{S}$). According to property 1., $p_{b,b}(I)$ is a model of $Prop(\mathcal{S})$. According to definition α_1 we have $(C'^{dom})^{p_{b,b}(I)} = true$ and $(C^{dom})^{p_{b,b}(I)} = false$, i.e., $(\neg C^{dom})^{p_{b,b}(I)} = true$ and $(\neg C'^{dom})^{p_{b,b}(I)} = false$. This contradicts the fact that $\neg C'^{dom}$ is an implicate of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$.

Let us show now that $R(X) \equiv C'(X)$ is a *maximal* conjunctive rewriting. If it is false, there exists a *maximal* conjunctive rewriting R' of Q w.r.t. $\mathcal{S}$ strictly subsuming R, i.e., there exists a model I of $\mathcal{S}$ and an element $o \in \Delta^I$ such that $o \in (R')^I$, $o \in Q^I$, and $o \notin R^I$. Theorem 1 states that all the maximal conjunctive rewritings of a user query can be obtained using a backward chaining algorithm with cycle detection. Because of the form of the core-RDFS rules in $schema(\mathcal{S})$ (Section 2), any maximal conjunctive rewriting of a query made of a single atom $Q(X) \equiv C(X)$ is either of the form $R'(X) \equiv A(X)$, or $R'(X) \equiv \exists Y B(X, Y)$, or $R'(X) \equiv \exists Y B(Y, X)$:

– $R'(X) \equiv A(X)$: According to property 1, $p_{o,o}(I)$ is a model of $Prop(\mathcal{S})$ and according to definition α_1 we have: $(A^{dom})^{p_{o,o}(I)} = true$, $(C^{dom})^{p_{o,o}(I)} = true$, and $(C'^{dom})^{p_{o,o}(I)} = false$, i.e., $(\neg A^{dom})^{p_{o,o}(I)} = false$, $(\neg C^{dom})^{p_{o,o}(I)} = false$, and $(\neg C'^{dom})^{p_{o,o}(I)} = true$. This contradicts the fact that $\neg C'^{dom}$ is a *prime* implicate of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$.
– $R'(X) \equiv \exists Y B(X,Y)$: Since $o \in R'$ then there exists $o' \in \Delta^I$ such that $(o,o') \in B^I$. According to property 1, $p_{o,o'}(I)$ is a model of $Prop(\mathcal{S})$ and according to definition α_1 and α_2 we have: $(B^{prop})^{p_{o,o'}(I)} = true$, $(C^{dom})^{p_{o,o'}(I)} = true$, and $(C'^{dom})^{p_{o,o'}(I)} = false$, i.e., $(\neg B^{prop})^{p_{o,o'}(I)} = false$, $(\neg C^{dom})^{p_{o,o'}(I)} = false$, $(\neg C'^{dom})^{p_{o,o'}(I)} = true$. This contradict the fact that $\neg C'^{dom}$ is a *prime* implicate of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$.
– $R'(X) \equiv \exists Y B(Y,X)$: This case is similar to the previous one.

(i) $(\Rightarrow)$ We have to prove that if $R(X) \equiv C'(X)$ is a maximal conjunctive rewriting of a query $Q(X) \equiv C(X)$ w.r.t. $\mathcal{S}$ then $\neg C'^{dom}$ is a proper prime implicate of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$.

Suppose that $R(X) \equiv C'(X)$ is a maximal conjunctive rewriting of a query $Q(X) \equiv C(X)$ w.r.t. $\mathcal{S}$.
Let us first show that $\neg C'^{dom}$ is an *implicate* of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$. Since $C'(X)$ is a conjunctive rewriting of $C(X)$, for every model I of $\mathcal{S}$: $(C')^I \subseteq (C)^I$. If $\neg C'^{dom}$ is not an implicate of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$, then $\{\neg C^{dom}\} \cup Prop(\mathcal{S}) \not\models \neg C'^{dom}$, i.e., $\{C'^{dom}\} \cup Prop(\mathcal{S}) \not\models C^{dom}$, i.e., there exists a model J of $\{C'^{dom}\}$ $\cup Prop(\mathcal{S})$ such that $(C^{dom})^J = false$ and $(C'^{dom})^J = true$. According to property 2., $i(J)$ is a model of $\mathcal{S}$. According to definition β_1 we have $dom \notin (C)^{i(J)}$ and $dom \in (C')^{i(J)}$ thus $(C)^{i(J)} \not\subseteq (C')^{i(J)}$. This contradicts the fact that $R(X) \equiv C'(X)$ is a conjunctive rewriting of $Q(X) \equiv C(X)$ w.r.t. $\mathcal{S}$.

Let us show now that $\neg C'^{dom}$ is a *proper* implicate of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$. Let I be a model of $\mathcal{S}$ such that $R^I \neq \emptyset$. Note that such a model always exists since the core-RDFS data model does not allow building unsatisfiable logical sentences (w.r.t. $\mathcal{S}$). Let o be in C'^I. According to property 1, $p_{o,o}(I)$ is a model of $Prop(\mathcal{S})$ and according to definition α_1 we have $(C'^{dom})^{p_{o,o}(I)} = true$, i.e., $(\neg C'^{dom})^{p_{o,o}(I)} = false$. Therefore, there exists a model of $Prop(\mathcal{S})$ which is not a model of $\neg C'^{dom}$. That means that $\neg C'^{dom}$ is not an implicate of $Prop(\mathcal{S})$ alone.

Finally, let us show that $\neg C'^{dom}$ is a *prime* implicate of $\neg C^{dom}$ w.r.t. $Prop(\mathcal{S})$. Suppose that there exists a clause cl such that $Prop(\mathcal{S}) \cup \{\neg C^{dom}\} \models cl$ and $cl \models \neg C'^{dom}$. Either cl is $\neg C'^{dom}$ since $\neg C'^{dom}$ is a literal and thus $\neg C'^{dom}$ is *prime*, or cl is the empty clause and thus $Prop(\mathcal{S}) \cup \{\neg C^{dom}\}$ is unsatisfiable. Let us show that the latter case is not possible. Let I be a model of $\mathcal{S}$ such that $o \notin (C^{dom})^I$. It is always possible to build such a model: let $K = (\Delta^K, .^K)$ be a model of $\mathcal{S}$ such that $o \notin \Delta^K$, then $I = (\Delta^K \cup \{o\}, .^K)$ is a model of $\mathcal{S}$ such that $o \notin (C^{dom})^I$. According to the property 1, $p_{o,o}(I)$ is model of $Prop(\mathcal{S})$, and according to the definition α_1 we have: $(C^{dom})^{p_{o,o}(I)} = false$. Therefore, $p_{o,o}(I)$ is a model of $Prop(\mathcal{S}) \cup \{\neg C^{dom}\}$ and thus $Prop(\mathcal{S}) \cup \{\neg C^{dom}\}$ is always satisfiable. $\square$

In the next section, we provide an algorithm built on top of DeCA which computes *all* the maximal conjunctive rewritings of any user query.

4 Query rewriting algorithm of a SomeRDFS PDMS

The query rewriting algorithm of SomeRDFS, namely DeCA$^{\text{RDFS}}$, is designed on top of DeCA. On each SomeRDFS peer $\mathcal{P}$, DeCA$^{\text{RDFS}}$ acts as an interface between the user and DeCA which works on $Prop(\mathcal{P})$.

The strategy of DeCA$^{\text{RDFS}}$ is to rewrite the user query's atoms independently with DeCA, based on the result of Proposition 2, and then to combine their rewritings in order to generate some conjunctive rewritings of the user query w.r.t. a SomeRDFS PDMS. DeCA$^{\text{RDFS}}$ guarantees that all the maximal conjunctive rewritings of the user query w.r.t. a SomeRDFS PDMS are generated.

DeCA$^{\text{RDFS}}$ is presented in Algorithm 1. It uses the conjunctive distribution operator $\oslash$ on sets of FOL formulas: $S_1 \oslash \cdots \oslash S_n = \oslash_{i=1}^{n} S_i = \{F_1 \wedge \cdots \wedge F_n \mid F_1 \in S_1, \ldots, F_n \in S_n\}$. Note that if $S_i = \emptyset$, $i \in [1..n]$, then $\oslash_{i=1}^{n} S_i = \emptyset$.

Algorithm 1 DeCA$^{\text{RDFS}}$

```
Require: A user query $Q(\bar{X}) \equiv \exists \bar{Y} \bigwedge_{i=1}^{n} r_i(\bar{X}_i, \bar{Y}_i)$ s.t. $\bar{X} = \bigcup_{i=1}^{n} X_i$ and $\bar{Y} = \bigcup_{i=1}^{n} Y_i$
Ensure: Output contains only conjunctive rewritings of $Q$ w.r.t $\mathcal{S}$, including all the
    maximal conjunctive rewritings of $Q$ w.r.t $\mathcal{S}$
 1: for $i \in [1..n]$ do
 2:   AtomRewritings$_i$ = $\emptyset$
 3:   if $r_i(\bar{X}_i, \bar{Y}_i)$ is of the form $C(U)$ then
 4:     for $imp \in DeCA(\neg C^{dom})$ do
 5:       if $imp$ has the form $\neg C'^{dom}$ then
 6:       AtomRewritings$_i$ = AtomRewritings$_i$ $\cup \{C'(U)\}$
 7:       else if $imp$ has the form $\neg P'^{prop}$ then
 8:       AtomRewritings$_i$ = AtomRewritings$_i$ $\cup \{\exists Z P'(U, Z)\}$ endif
 9:     end for
10:     for $imp \in DeCA(\neg C^{range})$ do
11:       if $imp$ has the form $\neg P'^{prop}$ then
12:       AtomRewritings$_i$ = AtomRewritings$_i$ $\cup \{\exists Z P'(Z, U)\}$ endif
13:     end for
14:   else if $r_i(\bar{X}_i, \bar{Y}_i)$ is of the form $P(U_1, U_2)$ then
15:     for $imp \in DeCA(\neg P^{prop})$ do
16:       if $imp$ has the form $\neg P'^{prop}$ then
17:       AtomRewritings$_i$ = AtomRewritings$_i$ $\cup \{P'(U_1, U_2)\}$
18:     end for
19:   end if
20: end for
21: return $\oslash_{i=1}^{n}$AtomRewritings$_i$
```

4.1 Illustrative example (continued)

Let us consider the user query $Q_1(X) \equiv \mathcal{P}_2{:}Work(X)$ asked to $\mathcal{P}_2$ in the example of Section 2.8.
At Line 4 of DeCA$^{\text{RDFS}}$, $\neg\mathcal{P}_2{:}Work^{dom}$ is asked to DeCA:
DeCA$(\neg\mathcal{P}_2{:}Work^{dom}) = \{\neg\mathcal{P}_2{:}Work^{dom}, \neg\mathcal{P}_2{:}Painting^{dom}, \neg\mathcal{P}_2{:}Sculpture^{dom}, \neg\mathcal{P}_2{:}Music^{dom}, \neg\mathcal{P}_2{:}refersTo^{prop}, \neg\mathcal{P}_1{:}Artifact^{dom}, \neg\mathcal{P}_1{:}belongsTo^{prop}\}$.
At Line 10 of DeCA$^{\text{RDFS}}$, $\neg\mathcal{P}_2{:}Work^{range}$ is asked to DeCA:
DeCA$(\neg\mathcal{P}_2{:}Work^{range}) = \{\neg\mathcal{P}_2{:}Work^{range}, \neg\mathcal{P}_2{:}Painting^{range}, \neg\mathcal{P}_2{:}Sculpture^{range}, \neg\mathcal{P}_2{:}Music^{range}, \neg\mathcal{P}_1{:}Artifact^{range}, \neg\mathcal{P}_1{:}creates^{prop}, \neg\mathcal{P}_1{:}paints^{prop}, \neg\mathcal{P}_1{:}sculpts^{prop}\}$.
It follows that, at Line 21, $\circledwedge_{i=1}^{1}$AtomRewritings$_i$ $= \{\mathcal{P}_2{:}Work(X), \mathcal{P}_2{:}Painting(X), \mathcal{P}_2{:}Sculpture(X), \mathcal{P}_2{:}Music(X), \exists Z \mathcal{P}_2{:}refersTo(X,Z), \mathcal{P}_1{:}Artifact(X), \exists T \mathcal{P}_1{:}belongsTo(X,T), \exists U \mathcal{P}_1{:}creates(U,X), \exists V \mathcal{P}_1{:}paints(V,X), \exists W \mathcal{P}_1{:}sculpts(W,X)\}$.
Therefore, DeCA$^{\text{RDFS}}$ returns the maximal conjunctive rewritings of Q_1 w.r.t. $\mathcal{S}$ exhibited in the example of Section 3.1.

Let us consider now the user query $Q_2(X,Y) \equiv \mathcal{P}_2{:}Painting(X) \wedge \mathcal{P}_2{:}refersTo(X,Y)$ asked to $\mathcal{P}_2$ in the example of Section 2.8.
In the first iteration of DeCA$^{\text{RDFS}}$, $\neg\mathcal{P}_2{:}Painting^{dom}$ is asked to DeCA at Line 4 and $\neg\mathcal{P}_2{:}Painting^{range}$ is asked to DeCA at Line 10 with the following results:
DeCA$(\neg\mathcal{P}_2{:}Painting^{dom}) = \{\neg\mathcal{P}_2{:}Painting^{dom}\}$,
DeCA$(\neg\mathcal{P}_2{:}Painting^{range}) = \{\neg\mathcal{P}_2{:}Painting^{range}, \neg\mathcal{P}_1{:}paints^{prop}\}$.
In the second iteration of DeCA$^{\text{RDFS}}$, $\neg\mathcal{P}_2{:}refersTo^{prop}$ is asked to DeCA at Line 15 with the following results:
DeCA$(\neg\mathcal{P}_2{:}refersTo^{prop}) = \{\neg\mathcal{P}_2{:}refersTo^{prop}, \neg\mathcal{P}_1{:}belongsTo^{prop}\}$.
It follows that, at Line 21, $\circledwedge_{i=1}^{2}$AtomRewritings$_i$ $= \{$
$\mathcal{P}_2{:}Painting(X) \wedge \mathcal{P}_2{:}refersTo(X,Y)$, $\mathcal{P}_2{:}Painting(X) \wedge \mathcal{P}_1{:}belongsTo(X,Y)$,
$\exists Z\ \mathcal{P}_1{:}paints(Z,X) \wedge \mathcal{P}_2{:}refersTo(X,Y)$,
$\exists Z\ \mathcal{P}_1{:}paints(Z,X) \wedge \mathcal{P}_1{:}belongsTo(X,Y)\}$.
Therefore, DeCA$^{\text{RDFS}}$ returns the maximal conjunctive rewritings of Q_2 w.r.t. $\mathcal{S}$ exhibited in the example of Section 3.1.

4.2 Properties of DeCA$^{\text{RDFS}}$

The main properties of DeCA$^{\text{RDFS}}$ are stated by the two following theorems.

Theorem 2 (Soundness of DeCA$^{\text{RDFS}}$). *Let $\mathcal{S}$ be a* SomeRDFS *PDMS and let $\mathcal{P}$ be one of its peers. Any user query Q asked to $\mathcal{P}$ will produce a set* DeCA$^{\text{RDFS}}(Q)$ *of queries containing only conjunctive rewritings of Q w.r.t. $\mathcal{S}$.*

Proof. Theorem 1 in [5] states the soundness of DeCA. Therefore, according to Proposition 2 in Section 3, AtomRewritings$_i$ ($i \in [1..n]$) at Line 21 contains only conjunctive rewritings of the i^{th} atom of the user query Q.

The soundness of the output of DECA$^{\text{RDFS}}$ at Line 21 results from the fact that, as we have shown in the proof of Theorem 1, we are in a setting where it has been proved [19] that conjuncting conjunctive rewritings of each atom of the query provides conjunctive rewritings of the query. □

Theorem 3 (Completeness of DECA$^{\text{RDFS}}$). *Let $\mathcal{S}$ be a* SOMERDFS *PDMS and let $\mathcal{P}$ be one of its peers. Any user query Q asked to $\mathcal{P}$ will produce a set* DECA$^{\text{RDFS}}(Q)$ *of queries containing all the maximal conjunctive rewritings of Q w.r.t. $\mathcal{S}$.*

Proof. Theorem 2 in [5] states a sufficient condition for the completeness of DECA. Proposition 1 in Section 3 ensures that this condition is satisfied in *Prop*($\mathcal{S}$). Therefore, the use of DECA at Line 4, Line 10, and Line 15 produces all the proper prime implicates of the given literals w.r.t. *Prop*($\mathcal{S}$).
According to Proposition 2 in Section 3, ATOMREWRITINGS$_i$ ($i \in [1..n]$) at Line 21 contains *all* the maximal conjunctive rewritings of the i^{th} atom of the user query Q.
The completeness of the output of DECA$^{\text{RDFS}}$ at Line 21 results from the fact that, as we have shown in the proof of Theorem 1, we are in a setting where it has been proved [19] that conjuncting all the maximal conjunctive rewritings of each atom of the query provides all the maximal conjunctive rewritings of the query. □

Theorem 4 (Termination of DECA$^{\text{RDFS}}$). *Let $\mathcal{S}$ be a* SOMERDFS *PDMS and let $\mathcal{P}$ be one of its peers. Any user query Q asked to $\mathcal{P}$ will produce a computation that always terminates.*

Proof. Theorem 1 in [5] states that DECA always terminates after having produced a *finite* set of proper prime implicates of a given literal w.r.t. *Prop*($\mathcal{S}$). Therefore, it is obvious that DECA$^{\text{RDFS}}$ always terminates. □

Other interesting properties are inherited from DECA's properties: *anytime computation* and *termination notification*. Note that the latter property is crucial for an anytime algorithm.

Theorem 5 (Anytime computation of DECA$^{\text{RDFS}}$). *Let $\mathcal{S}$ be a* SOMERDFS *PDMS and let $\mathcal{P}$ be one of its peers. Any user query Q asked to $\mathcal{P}$ will return a set* DECA$^{\text{RDFS}}(Q)$ *of query rewritings as a stream.*

Proof. It is obvious that the n iterations at Line 1 are independent. Therefore, they can be parallelized.
Within an iteration, if there are several calls to DECA, those calls and the computations that follows are independent: they only *add* results in the same variable ATOMREWRITING$_i$. Therefore, those calls can be parallelized. Moreover, since DECA performs an anytime computation, the feeding of the ATOMREWRITING$_i$ can be made anytime: each time a result is produced by DECA (at Line 4, Line 10, and Line 15), that result is processed (within the for loops at Line 4, Line 10, and Line 15).

It follows that the computation at Line 21 can also be made anytime: each time an AtomRewriting$_k$ ($k \in [1..n]$) is fed with a formula F, $\circledwedge_{i=1}^{k-1}$AtomRewritings$_i$ $\circledwedge \{F\} \circledwedge \circledwedge_{i=k+1}^{n}$AtomRewritings$_i$ is returned in the output stream. □

Theorem 6 (Termination notification of DeCA$^{\text{RDFS}}$). *Let $\mathcal{S}$ be a SomeRDFS PDMS and let $\mathcal{P}$ be one of its peers. Any user query asked to $\mathcal{P}$ will produce a computation, the end of which will be notified to the user.*

Proof. Theorem 3 in [5] states that DeCA, which is anytime, notifies of its termination. Therefore, it is obvious that as soon as all the *finite number* of calls to DeCA have notified of there termination and the AtomRewritings$_i$ ($i \in [1..n]$) have been properly fed according to the results of these calls, DeCA$^{\text{RDFS}}$ can notifies the user of its termination after having returned $\circledwedge_{i=1}^{n}$AtomRewritings$_i$ at Line 21. □

4.3 Scalability of DeCA$^{\text{RDFS}}$

The scalability of DeCA$^{\text{RDFS}}$ is directly related to the scalability of DeCA. We can infer that DeCA$^{\text{RDFS}}$ has good scalability properties from the DeCA scalability experiments that are reported in [7]. Those experiments have been performed on networks of 1000 peers deployed on a cluster of 75 heterogeneous computers. The networks that have been considered have a topology of "small world" [22] like in social networks: there are clusters of very connected peers and a few connections between peers of different clusters. Each peer theory is randomly generated as 70 clauses of length 2 from 70 variables, 40 of which are ramdomly chosen as target variables. A peer is connected to 10 other peers with which it shares 2 variables (randomly chosen). These connections take into account the "small world" topology. From a peer point of view, these connections are done by adding in its theory 20 new clauses modeling the mappings with its neighbours. Among the experiments performed on SomeWhere P2PISs [7], the ones that are the most representative of the propositional encodings of SomeRDFS PDMSs correspond to the *very easy* case for DeCA in which all the mappings correspond to clauses of length 2. It is due to the simplicity of the RDFS model (no class constructor and no negation). In that case, it has been experimentally shown that all the proper prime implicates of a given literal are computed in 0.07 second in mean (over more than 300 different input literals). This lets envision a good scalability of DeCA$^{\text{RDFS}}$, since for any user query, atoms are independently rewritten in parallel. Thus, the expected time to add to the above 0.07 second in mean is the time needed to combine the rewritings of the user query atoms (Line 21 in Algorithm 1).

5 Related work

We have already presented in the introduction some PDMSs that have been developped for the Semantic Web: Edutella [2], RDFPeers [3], GridVine [4] or

SomeOWL [5]. Like a SomeRDFS PDMS, a GridVine PDMS is based on RDFS and considers mappings between peer ontologies. However, the mappings considered in a GridVine PDMS are restricted to equivalence of properties, while we allow in a SomeRDFS PDMS more expressive mappings that can be inclusion of classes, inclusion of properties, and domain and range typing of properties. In contrast with a GridVine PDMS, the topology of a SomeRDFS PDMS is not fixed and results from the existence of mappings between peers.

Several peer-to-peer data management systems for other data models than those of the Semantic Web have been proposed recently.
Piazza [16, 23], in contrast with Edutella, does not consider that the data distributed over the different peers must be described relatively to some existing reference schemas. Each peer has its own data and schema and can mediate with some other peers by declaring *mappings* between its schema and the schemas of those peers. The topology of the network is not fixed (as in Edutella) but accounts for the existence of mappings between peers (as in SomeOWL and SomeRDFS PDMSs): two peers are logically connected if there exists a mapping between their two schemas. The underlying data model of the first version of Piazza [16] is relational and the mappings between relational peer schemas are inclusion or equivalence statements between conjunctive queries. Such a mapping formalism encompasses the *Local-as-View* and the *Global-as-View* [24] formalisms used in information integration systems based on single mediators. The price to pay is that query answering is undecidable except if some restrictions are imposed on the mappings or on the topology of the network [16]. The currently implemented version of Piazza [23] relies on a tree-based data model: the data is in XML and the mappings are equivalence and inclusion statements between XML queries. Query answering implementation is based on practical (but not complete) algorithms for XML query containment and rewriting. The scalability of Piazza so far does not go up to more than about 80 peers in the published experiments and relies on a wide range of optimizations (mappings composition [25], paths pruning [26]), made possible by the centralized storage of all the schemas and mappings in a global server.
The peer data management system considered in [27] is similar to that of [16] but proposes an alternative semantics based on epistemic logic. With that semantics it is shown that query answering is always decidable (even with cyclic mappings). Answers obtained according to this semantics correspond to a subset of those that would be obtained according to the standard FOL semantics. However, to the best of our knowledge, these results are not implemented.
The Kadop system [28] is an infastructure based on distributed hash tables for constructing and querying peer-to-peer warehouses of XML resources semantically enriched by taxonomies and mappings. The mappings that are considered are simple inclusion statement between atomic classes.

We will end this section by relating the DeCA$^{\mathrm{RDFS}}$ rewriting algorithm that we have described in Section 4, with the rewriting algorithm $PerfectRef$ used in [12] for reformulating a query w.r.t. *DL-Lite* Tboxes. The subtle step of $PerfectRef$ consists in rewriting each atom of the query by applying *positive* *inclusions*. The result of the application of the two positive inclusion statements expressing domain and range role typing corresponds exactly to respectively Line 8 and Line 12 of the DeCA$^{\text{RDFS}}$ algorithm (applied in the centralized case for making the comparison meaningful). The difference is that while inclusion statements are applied on first-order atoms in $PerfectRef$, we proceed in two steps: first the variables of the query are removed and we compute propositional rewritings, second we obtain the FOL rewritings by simply adding variables (possibly fresh existential variables) appropriately, depending on whether the propositional rewritings come from propositional atoms of the form C^{dom} or C^{range}. The equivalent of the application of the positive inclusions is done in the first step, which applies to propositional atoms. This is an advantage for scalability issues in the decentralized case.

6 Conclusion anf future work

We have presented the SomeRDFS model of PDMSs based on RDFS. It is the first work on distributed RDFS handling semantic heterogeneity between peers through more complex mappings than equivalence statements. The mappings that we have considered are RDFS statements involving classes or properties of different peers. Our approach for query answering in this setting relies on two steps: query rewriting results in a union of conjunctive queries over possibly different peers ; the answers are then obtained by evaluating each of those conjunctive queries on the appropriate peers. This paper has focused on the first step which is crucial for scalablity issues since it is within this step that the different peers possibly relevant to the query are discovered. For the answering step, we know which peers have to be interrogated and with which queries. The optimization issues that are relevant to this evaluation step are out of the scope of this paper. Query answering by rewriting is a standard approach in information integration systems based on centralized mediators. It raises new problems in a decentralized setting, in particular scalabiliy and even decidability [29]. The fact that in our approach query rewritings can be obtained through a propositional encoding step guarantees decidability and scalability.

In fact, we have shown how to deploy SomeRDFS PDMSs on top of the SomeWhere infrastructure for which experiments [7] have shown good scalability properties. As a comparison, a simple GridVine PDMS of 60 peers have been deployed and experimented in [4]: peer ontologies are in core-RDFS, 15 ontologies are used (each of which is used by 4 peers), each peer has 2 mappings and stores only 1 fact. On such a PDMS, a user query, which is similar to our user query made of a single atom, is answered in more that 10 seconds. In contrast, experiments presented in [7] show that on a more complex network with bigger ontologies (1000 peers, 1000 ontologies, 10 mappings per peer), the rewritings produced by DeCA for any user query made of a single atom are obtained in 0.07 second in mean. This lets envision that the whole query answering (rewriting and evaluation) could be made in less than 1 or 2 seconds. Such a hint must be confirmed by a large-scale experimental study that we plan to conduct in the near future.

We also plan to extend the current SomeRDFS model for handling more complex ontologies and more complex mappings. In particular, it seems doable to consider *DL-Lite* both for expressing local ontologies over RDF facts and for expressing mappings between ontologies. Since the negation is supported at the propositional level in SomeWhere the DeCA algorithm can be used to check the satisfiability of the global schema. Then it should be straightforward to extend the two-step DeCA$^{\text{RDFS}}$ rewriting algorithm to handle the additional constructors of *DL-Lite*$_R$. As a consequence, by a slight extension of the approach presented in this paper we could obtain a fast deployment of PDMS based on distributed *DL-Lite* (in which the mappings are interpreted in first-order semantics).

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5) (2001)
2. Nedjl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., al.: EDUTELLA: a P2P networking infrastructure based on RDF. In: WWW. (2002)
3. Cai, M., Frank, M.: RDFPeers: a scalable distributed RDF repository based on a structured P2P network. In: WWW. (2004)
4. Aberer, K., Cudré-Mauroux, P., Hauswirth, M., Pelt, T.V.: GridVine: Building internet-scale semantic overlay networks. In: ISWC. (2004)
5. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.C., Simon, L.: Distributed reasoning in a P2P setting: Application to the semantic web. Journal of Artificial Intelligence Research (JAIR) (2006)
6. Stoica, I., Morris, R., Karger, D., Kaasshoek, M., Balakrishnan, H.: CHORD a scalable P2P lookup service for internet applications. In: ACM SIGCOMM. (2001)
7. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.C., Simon, L.: Scalability study of P2P consequence finding. In: IJCAI. (2005)
8. ter Horst, H.J.: Extending the RDFS entailment lemma. In: ISWC. (2004)
9. de Bruijn, J., Franconi, E., Tessaris, S.: Logical reconstruction of normative RDF. In: OWLED. (2005)
10. de Bruijn, J., Franconi, E., Tessaris, S.: Logical reconstruction of RDF and ontology languages. In: PPSWR. (2005)
11. Farrugia, J.: Model-theoretic semantics for the web. In: WWW. (2003)
12. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: AAAI. (2005)
13. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: KR. (2006)
14. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: combining logic programs with description logic. In: WWW. (2003)
15. Haase, P., Broekstra, J., Eberhart, A., Volz, R.: A comparison of RDF query languages. In: ISWC. (2004)
16. Halevy, A., Ives, Z., Suciu, D., Tatarinov, I.: Schema mediation in peer data management systems. In: ICDE. (2003)

17. Goasdoué, F., Rousset, M.C.: Answering queries using views: a KRDB perspective for the semantic web. ACM Journal - Transactions on Internet Technology (TOIT) **4**(3) (2004)
18. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. 2nd edition edn. Prentice-Hall, Englewood Cliffs, NJ (2003)
19. Levy, A.Y., Mendelzon, A.O., Sagiv, Y., Srivastava, D.: Answering queries using views. In: PODS. (1995)
20. Pottinger, R., Halevy, A.Y.: MiniCon: A scalable algorithm for answering queries using views. In: VLDB Journal 10(2-3). (2001)
21. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
22. Watts, D.J., Strogatz, S.H.: Models of the small world. Nature **393** (1998)
23. Halevy, A., Ives, Z., Tatarinov, I., Mork, P.: Piazza: data management infrastructure for semantic web applications. In: WWW. (2003)
24. Halevy, A.Y. In: Logic-based techniques in data integration. Kluwer Academic Publishers (2000)
25. Madhavan, J., Halevy, A.: Composing mappings among data sources. In: VLDB. (2003)
26. Tatarinov, I., Halevy, A.: Efficient query reformulation in peer data management systems. In: SIGMOD. (2004)
27. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Logical fondation of P2P data integration. In: PODS. (2004)
28. Abiteboul, S., Manolescu, I., Preda, N.: Constructing and querying P2P warehouses of XML resources. In: SWDB. (2004)
29. Tatarinov, I., Ives, Z., Madhavan, J., Halevy, A., Suciu, D., Dalvi, N., Dong, X., Kadiyska, Y., Miklau, G., Mork, P.: The Piazza peer data management project. In: SIGMOD Record. Volume 32. (2003)