FOWLA, A Federated Architecture for Ontologies

Tarcisio Farias, Ana Roxin, Christophe Nicolle

FOWLA, a Federated Architecture for Ontologies

Farias[1], M. T., Roxin[2], A. & Nicolle[2], C.

[1]Active3D, Dijon, France
t.mendesdefarias@active3D.net
[2]Checksem, Laboratory LE2I (UMR CNRS 6306), University of Burgundy, Dijon, France
{ana-maria.roxin,cnicolle}@u-bourgogne.fr

Abstract. The progress of information and communication technologies has greatly increased quantity of data to process. Thus managing data heterogeneity is a problem nowadays. In the 1980s, the concept of a Federated Database Architecture (FDBA) was introduced in the field of database management as a collection of components that, by means of loosely coupled federation, share and exchange information. Semantic web technologies mitigate the data heterogeneity problem, however due to the data structure heterogeneity the integration of several ontologies is still a complex task. For tackling this problem, we propose a loosely coupled federated ontology architecture (FOWLA). Our approach allows the coexistence of various ontologies sharing common data dynamically at query execution through Horn-like rules. To show the benefits of using FOWLA, we have implemented this architecture for the integration of 2 ontologies, having more than 2500 concepts and properties. We also identify the advantages FOWLA over other existing initiatives, notably its reduced data redundancy and modularized maintainability.

Keywords: SWRL, Horn-like rules, Federated Ontology Architecture, Semantic web, OWL.

1 Introduction

With advances of the information and communication technologies the amount of data to process and share has exponentially increased. Consequently, there is a growing demand for information interoperability. Indeed, with the advent of the personal computer in the 1980s, data interoperability first became an issue, then with the advent of the Internet has risen the need for more principled mechanisms for interoperability. When considering enterprise information systems, three layers of interoperability exist. Physical interoperability (first level) concerns the levels of the ISO/OSI network hierarchy and has been solved through network protocols such as Ethernet, IP, TCP and HTTP. The second level concerns syntactic interoperability, namely the form of the messages exchanged in the information system. This issue has been solved through syntactic standards such as XML, HTML, WSDL, SOAP, etc. Finally, the third level of interoperability addresses the meaning of the exchanged messages and is called semantic interoperability. When implemented, semantic interoperability allows automatic machine

processing of information (selection, composition, reasoning, etc.). These advances are really important in the context of enterprise information integration (EII) [XX], building information models [XX] and more in general in semantic web [XX]. Ontologies defined using standard ontology languages such as OWL represent the building bricks for achieving such semantic interoperability. Indeed, as interoperability at the data model level has been pointed out as a solution to information integration [XX], and the usage of ontologies allows having data exchanges respecting the same original schema meaning (i.e. semantics).

Nevertheless, semantic heterogeneity remains a problem when integrating data from various ontologies which model the same information in different ways. Indeed, even if an ontology is defined as an "explicit and shared specification of a conceptualization of a given knowledge domain" [REF], different ontologists (i.e. ontology designers) can produce different ontologies for a same knowledge domain. Thus, just adopting ontologies, like just using XML [XX], does not eliminate heterogeneity for good: it elevates heterogeneity problems at a higher level. As noted by Alon Y. Halevy in [XX], semantic heterogeneity exists whenever there is more than one way to structure a body of data (i.e. schema).

Therefore, in order to address the problem of semantic interoperability by means of ontologies, we propose a loosely coupled federated architecture for OWL ontologies. This architecture is based on ontology alignments, logical rules and inference mechanisms. The article at hand is structured as follows: Section **Erreur ! Source du renvoi introuvable.** gives the scientific background for our work; Section **Erreur ! Source du renvoi introuvable.** presents most important related work in the considered domain. Section **Erreur ! Source du renvoi introuvable.** details our approach, notably the components and underlying processes of the FOWLA architecture. Numerical results in terms of query time execution improvement are illustrated in Section **Erreur ! Source du renvoi introuvable.**. Finally, we conclude this article by identifying several additional works that could be undertaken.

2 Background

Semantic Web technologies constitute one of the most promising trends for the future Web, notably as they come with the promise of making existing data machine-understadable. The architecture of Semantic Web comprises several layers and components. The RDF (Ressource description Framework) data model is the reference component. On top of it, three components exist [REF]:

- Components for ontologies: several standard languages for specifying ontologies exist, from the most basic (RDF Schema) to the most expressive Web Ontology Language (OWL) and its inheritor OWL 2. Ontology languages allow specifying knowledge pertaining to a given domain, according to the Open World Assumption. This assumption states that the facts that are not defined in the knowledge base are not false, but unknown. Ontology languages rely on Description Logics (DL) formalisms; an ontology concept is defined through several necessary and sufficient conditions.An knowledge base comprises a terminological model (formal definitions

of the concepts) and an assertional model (instantiation of these concepts). The terminological model is called TBox, and the assertional model is called ABox. In this work, we use the term ontology or knowledge base for referring to the whole TBox and ABox.

- Components for queries: the equivalent of SQL for databases is the SPARQL language. SPARQL allows querying RDF graphs and OWL ontologies, along with several possibilities for results processing (e.g. limit, order, offset). A SPARQL query comes in the form of a triple pattern <*Subject*, *Predicate*, *Object*>, in which the user defines the known term values and specifies the other as unknown. A triple matches the query pattern if its terms correspond to the ones specified in the query. Main drawbacks of SPARQL queries regard path expressions, expressions for transitive closure, rules or updates computing.
- Components for reasoning: With the Open World Assumption, queries over the data present in a knowledge base are often incompletly answered. Moreover, when applying reasoning over such data, conclusions cannot be drawn. The Close World Assumption states that knowledge that cannot be derived from existing data is considered to be false. With this assumption, and by means of logical rules (expressed using rule languages), one can perform rule inference on top of ontology-based knowledge specifications. Rules are expressed with terms defined in ontologies. Rule languages have been developped since 2000, with the RuleML initiative [9], which is based on the Logic Programming paradigm and implements a RDF syntax. The Semantic Web Rule Language (SWRL) [8, 10] is based on Logic Programming as well, but combines OWL and RuleML. SWRL allows defining conjunctive rules over the concepts and relationships present in an OWL ontology.

Besides the above considerations and for a better understanding of the work presented in this paper, we provide the following terms definition:

Definition 1. (*Ontology matching*) When determining if two ontologies have the same meaning (adressing the issue of semantic interoperability), an ontology matching process has to be implemented. Matching is the process of identifying correspondences between entities of different ontologies **Erreur ! Source du renvoi introuvable.**.

Definition 2. (*Ontology alignment*) An alignment is a set of correspondences between one or more ontologies. The alignment is the output of the process of ontology matching **Erreur ! Source du renvoi introuvable.**. In this wpaper, we consider that such alignment is expressed by means of Horn rules (rule axioms).

Definition 3. (*Rule or rule axiom*) A rule is composed of a rule head (also called consequent) and a body (also called antecedent). If the head of a rule is true, then its body is derived as a new assertion.

Definition 4. (*Horn clause*) A Horn clause is an implication from an antecedent (set of atomic formulae), to a consequent (single atomic formula) **Erreur ! Source du renvoi introuvable.**.

Definition 5. (*Target and source ontology*) When considering a positive Horn rule, its body generally comprises terms from ontologies different than the ones referencing terms in the rule's head. Indeed, when considering semantic interoperability by means

of ontologies, one usually shares data from several source ontologies to one target ontology. In this context, an ontology is considered as a source or a target ontology depending on the syntax of rules in the alignment and depending onto which the considered query is adressed.

3 Related Work

Many efforts were done since the 1980s to interoperate different database schemas, for instance, Sheth and Larson in [XX] classify the multi-database systems into two types: non-federated database systems and federated database systems. An example of a non-federated database is a centralized database which means a single integrated database schema. The expression Federated Database Architecture (FDBA) was first introduced by Heimbigner and McLeod in 1985 as a "collection of components to unite loosely coupled federation in order to share and exchange information" using "an organization model based on equal, autonomous databases, with sharing controlled by explicit interfaces." [XX]. Despite of our work being inspired on such definition of federated architecture; we define FOWLA as an architecture based on autonomous ontologies (including TBox and ABox) with sharing described as a rule-based format controlled by inference mechanisms (e.g. SWRL engine associated to OWL reasoner).

[1] presents a SPARQL query rewriting approach in the context of implementing interoperability over various ontologies stored in federated RDF datasets. Queries are addressed to different SPARQL endpoints and are rewritten based on the alignments defined among the ontologies. Alignments implement a specific alignment format, as specifyied by the authors. Still, [1] authors do not clearly justify the need of this alignment format. Their approach is further detailed in 3, notably by defining several functions for graph pattern rewriting.

In **Erreur ! Source du renvoi introuvable.**, Correndo et al. present a similar approach. They perform query rewriting for retrieving data from several different SPARQL endpoints. However, their algorithm takes into account only information specified as a graph pattern. It ignores constructs such as constraints expressed within the SPARQL reserved word FILTER.

When comparing both methods, 1 has the advantage of relying on Description Logic, and consequently supporting different query types (SELECT, CONSTRUCT, etc.), along with different SPARQL solution modifiers (LIMIT, ORDER BY, etc.). In this approach, the query rewritting process does not modify graph pattern operators. Still, both methods ignore the cases where several source and target ontologies can be involved. Correndo et al. **Erreur ! Source du renvoi introuvable.** provide a explanation for SPARQL query rewriting implementation by stating that ontology alignments defined on top of the logical layer implie reasoning over a considerable amount of data thus compromising query execution time. Approaches presented in 1**Erreur ! Source du renvoi introuvable.** represent successful optimizations of query execution times. Still, their main drawbacks concern adressing the possibility for writing queries using terms from different ontologies, along with offering extended inference capabilities (e.g. through reasoners and rule engines).

Despite the extensive studies, to the best of our knowledge, there is no work proposing a federated architecture in the context of semantic interoperability of OWL ontologies.

4 A Federated Architecture for Ontologies (FOWLA)

For addressing the issue of ontology interoperability, we have developed an approach based on a federated architecture for ontologies, FOWLA. This architecture contains two main components: the Federal Descriptor (FD) and the Federal Controller (FC). The FD component is responsible for describing ontology alignments. The FC module is executed at query time and allows exchanging data among ontologies according to FD generated alignment. It is also at query time that we check the data access policy for federated ontologies. The FOWLA architecture is illustrated in Fig. 1.

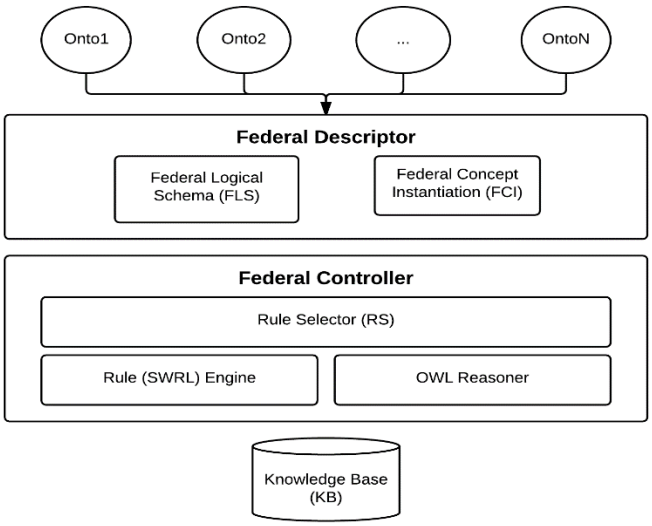

Fig. 1. FOWLA, Federated Architecture for Ontologies.

In order to describe ontology federation, we can rely on any alignment format present in the literature [2]. However, as the FC is a rule-based controller, it is preferable to use alignment formats based on rule syntax, as, for instance, SWRL rules. This avoids to convert alignment formats later on in the process.

As illustrated in Fig. 1, the FD module contains two sub-modules: Federal Logical Schema (FLS) and Federal Concept Instantiation (FCI). The first sub-module is an ensemble of logical rules describing the correspondences between ontologies. These mappings are expressed as logical rules, such as SWRL. Nevertheless, such logical rules are not capable of creating new concept instances. This is due to undecidability problems when integrating OWL and SWRL. Therefore, DL-safe rules are implemented for

regaining decidability [6]. To overcome the drawback of new instances' inference, we propose including the FCI sub-module in our architecture.

Indeed, data can be modelled in various ways [XX Alon Y. Halevy]. This implies schema heterogeneity, and consequently, increases the difficulty for establishing ontology interoperability. A first step in mitigating this issue is to use only ontologies specified with one formal language, such as OWL. Still, the same data can be encapsulated through several concepts from different and independent OWL ontologies. To illustrate this, let us suppose we want to achieve interoperability over two OWL ontologies (*Onto1* and *Onto2*), for which we define an alignment through SWRL rules. For the sake of simplicity, we consider that the URI's (Uniform Resource Identifier) namespace identifying a predicate specifies the ontology containing the predicate's definition. In other words, *onto1:q1(?x,?y)* means that predicate *q1* is defined in the ontology *Onto1*. We consider the alignment between *Onto1* and *Onto2* is defined using the SWRL rules listed in 4.1.

$$
\begin{aligned}
&swrl_1\text{: } onto1{:}D(?x) \rightarrow onto2{:}C(?x)\\
&swrl_2\text{: } onto2{:}C(?x) \rightarrow onto1{:}D(?x)\\
&swrl_3\text{: } onto1{:}D(?x) \wedge onto1{:}q1(?x, ?y) \wedge onto1{:}A(?y) \wedge onto1{:}q2(?y, ?z) \rightarrow onto2{:}p2(?x, ?z)\\
&swrl_4\text{: } onto2{:}C(?x) \wedge onto2{:}p2(?x,?z) \wedge onto1{:}A(?y) \wedge onto1{:}q1(?x, ?y) \rightarrow onto1{:}q2(?y,?z)
\end{aligned}
\tag{4.1}
$$

where *onto1:q1* is an OWL object property; *onto1:q2* and *onto2:p2* are OWL datatype properties. $swrl_1$ and $swrl_2$ state that *onto1:D* is equivalent to *onto2:C*. $swrl_3$ exemplifies a complex alignment that maps a graph pattern from *Onto1* to a datatype property from *Onto2*(i.e. *p2*). $swrl_4$ is another complex alignment mapping a graph pattern from *Onto1* and *Onto2* to a datatype property from *Onto1* (i.e. *q2*).

In our approach, these rules are part of the FLS sub-module. When considering $swrl_4$, sharing the data values of *onto2:p2* to *Onto1* implies creating the necessary instances for concept *onto1:A*. This is the case because these data values are represented (i.e. encapsulated) in a different way by *Onto1*. Nevertheless, defining an alignment rule such as "*onto2:C(?x) ∧ onto2:p2(?x,?z) → onto1:q1(?x, ?y) ∧ createInstances(?y, onto1:A) ∧ onto1:q2(?y,?z)"* is not possible due to undecidability issues.

To tackle this limitation, the FCI sub-module previously creates instances of necessary concepts from the target ontology to encapsulate the data shared by the source ontology. In other words, the FCI sub-module creates a graph pattern in the Knowledge Base (KB) by means of class instantiation and property assertion. Doing so, the data for *onto2:p2* is represented with vocabulary terms from *Onto1* based on previously defined alignment rules.

For the rules listed in 4.1, the FCI sub-module only considers class instantiation and property assertion for predicates in the $swrl_4$'s body. Therefore, for each instance C_i of type *onto2:C* (which becomes also an instance of type *onto1:D* when applying $swrl_2$), one *onto1:q1* property is asserted to C_i having as value one newly created instance A_i of type *onto1:A.* Once this assertion performed, the SWRL rule engine is capable of inferring the value of *onto1:q2* for C_i by applying $swrl_4$. In addition to what has been

said, the data value of *onto2:p2* is not materialized for *onto1:q2* by the FCI sub-module. This, however, is inferred by the rule engine. Fig. 2 illustrates the process of class instantiation and property assertion (in bold), as implemented when sharing the data value of *onto2:p2* to ontology *Onto1*, based on rule $swrl_4$.

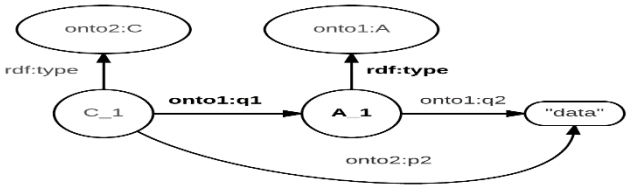

Fig. 2. Class instantiation and property assertion (in bold) for interoperability.

The FC module performs the bulk of necessary inferences to satisfy a data request from a system based on one or more federated ontologies. To do so, FC contains the following sub-modules: a Rule Selector (RS) and a Rule Engine associated to an OWL reasoner. These components are responsible to control the interoperation among the considered ontologies based on an ensemble of rules (contained in the FLS sub-module) and some description logic formalism (e.g.: OWL).

The Rule Selector (RS) sub-module is responsible for improving backward-chaining reasoning. Indeed, when considering the context of executing queries over complex and numerous alignments, the number of SWRL rules highly impacts query execution time. The RS module attempts to select the necessary and sufficient ensemble of rules to answer a given query. This avoids the reasoner to perform unnecessary inferences which would considerably slow down query processing. Further details of the functioning of the RS sub-module are presented in section 4.2.

We motivate our choice of a backward-chaining (or hybrid) reasoner for the FC module by the fact that interoperating several ontologies with forward-chaining reasoner requires storing a considerable amount of materialized data. Besides, any ontology modification can imply a re-computation of all inferred data.

Our implementation of the FOWLA architecture comprises two phases (see Fig. 3): a pre-processing phase and a query execution phase.

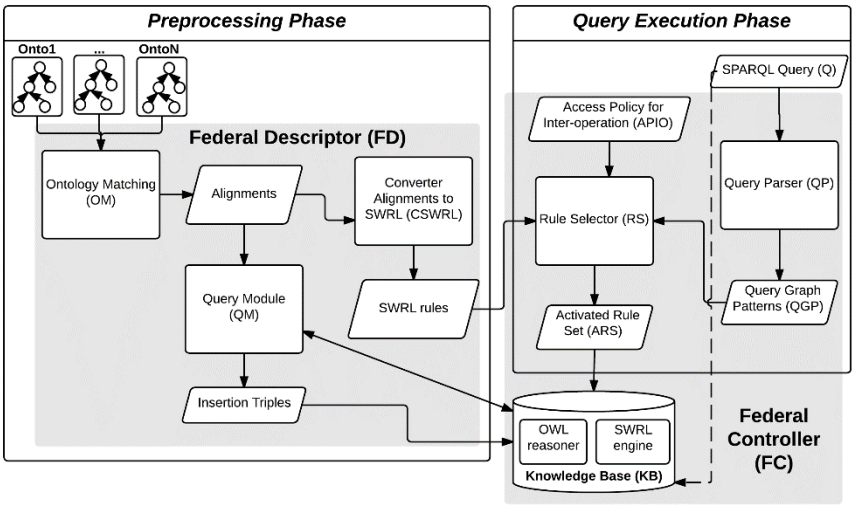

Fig. 3. FOWLA: pre-processing and query execution phases.

The pre-processing phase is responsible for creating the FD. The query execution phase relies on the FC module for retrieving data from the federated ontologies. These two phases are detailed in the sections 4.1 and 4.2, respectively.

4.1 Pre-processing Phase

For a full ontology interoperation, several complex alignments can be necessary. The ontology matching process is a fastidious and time consuming task. Because of this, we recommend the use of automatic ontology matching tools such as ASMOV [7] to support the alignments' conception (i.e. matching results). Nevertheless, these automatic matching solutions depend on the level of user involvement when verifying and validating the output alignments. Moreover, such solutions are not able to output complex alignments, such as the one listed in 4.2, where a sub-graph of *Onto2* is mapped to a sole *Onto1* property. Therefore, the user involvement in the ontology matching process is crucial as it was also noticed by Shvaiko and Euzenat in [7].

$$onto2{:}C_{21}(?x1) \wedge onto2{:}C_{22}(?x6) \wedge onto2{:}C_{23}(?x3) \wedge onto2{:}C_{23}(?x7) \wedge onto2{:}C_{24}(?x5) \wedge onto2{:}C_{25}(?x4) \wedge onto2{:}C_{26}(?x2) \wedge onto2{:}p_{21}(?x4, ?x5) \wedge onto2{:}p_{22}(?x5, ?x6) \wedge onto2{:}p_{23}(?x2, ?x4) \wedge onto2{:}p_{24}(?x2, ?x1) \wedge onto2{:}p_{25}(?x2, ?x3) \wedge onto2{:}p_{28}(?x7, ?x8) \wedge onto2{:}p_{26}(?x5, ?x7) \wedge onto2{:}p_{27}(?x6; ``Category'') \wedge onto2{:}p_{28}(?x3; ``ProductResource'') \rightarrow onto1{:}p_{11}(?x1; ?x8) \quad (4.2)$$

Once we defined the rules forming ontology alignments, if the alignment format is not a rule-based format such as SWRL, a conversion process is executed (as illustrated in Fig. 3). The resulting alignments in SWRL rules format are included in the FLS sub-module. Afterwards, the Query Module (QM) identifies each alignment presenting

schema heterogeneity, and therefore needing class instantiations and property assertions for modelling data from other ontologies. The QM retrieves instances which do not have property assertions for mapping the data from a source ontology to one target ontology. For doing so, it relies on SPARQL queries addressed over the knowledge base (KB). To exemplify this process, let us consider rule $swrl_4$ (see 4.1) as an input to QM. If a triple *onto2:C_1 onto2:p2 "data"^^xsd:string* is inserted by an external system into the knowledge base, QM materializes the triples *onto1:A_1 rdf:type onto1:q1* (i.e. class instantiation) and *onto2:C_1 onto1:q1 onto1:A_1* (i.e. property assertion). Besides, if *Onto2* is already populated, QM executes the query *Q* (see 4.3) over KB based on $swrl_4$ to retrieve the instances of *onto2:C* (also *onto1:D* by applying $swrl_2$) with no property assertions.

SPARQL Query : executed	*Q : SELECT ?x WHERE { ?x rdf:type onto2:C. FILTER NOT EXISTS {?x onto1:q1 ?y} }*	(4.3)

These properties block the data mapping between *Onto2* and *Onto1* created when applying $swrl_4$. Finally, the absent properties are materialized along with new instances for each one (i.e. object property value).

The pre-processing phase materializes some property assertions if and only if necessary due to schema heterogeneity. This materialized data is deleted when the contents of the FLS sub-module changes. Besides, if ontology alignments are modified, the QM is re-executed. The pre-processing phase outputs an ensemble of SWRL rules for the query execution phase that is described in the following section.

4.2 Query Execution Phase

Once federal description is accomplished, we select the specific rules necessary to answer a given query addressed over the federated ontologies. For addressing this task, we have developed a SPARQL Query Parser (QP). As shown in Fig. 3, the SPARQL query is passed to the QP module which parses it and isolates the concepts and properties it contains. Based on elements such as domain/range restrictions for properties involved in the query, the RS sub-module selects SWRL rules that have to be taken into account for answering the query. The first action performed by the RS module is to filter rules in FLS sub-module selecting only those rules that can infer data for the properties and/or concepts in the query (i.e. query graph patterns QGP as illustrated in Fig. 3). Secondly, for further rule filtering, the RS sub-module identifies the rules which have the same property in their head and selects only those respecting the domain/range restriction defined in the query. To exemplify this, let us suppose the query *Q'* (see 4.4) and the same FLS described in (4.1).

Federal Logical Schema	$swrl_1$: *onto1:D(?x) → onto2:C(?x)* $swrl_2$: *onto2:C(?x) → onto1:D(?x)* $swrl_3$: *onto1:D(?x) ∧ onto1:q1(?x, ?y) ∧ onto1:A(?y) ∧ onto1:q2(?y, ?z) → onto2:p2(?x, ?z)* $swrl_4$: *onto2:C(?x) ∧ onto2:p2(?x,?z) ∧ onto1:A(?y)*	(4.4)

	∧ onto1:q1(?x, ?y) → onto1:q2(?y,?z)
SPARQL Query executed	*Q' : SELECT ?x ?y WHERE{ ?x rdf:type onto2:C. ?x onto2:p2 ?y }*

Considering query *Q'*, the RS sub-module selects only the rules $swrl_1$ and $swrl_3$ because they are the only ones capable of inferring data for *onto2:C* and for *onto2:p2*, respectively. Besides, $swrl_3$ is chosen because it satisfies the domain restriction defined in the query *Q'* (i.e. *onto2:C*). These rules represent the necessary and sufficient subset of FLS for answering *Q'*. Moreover, access policy for interoperation (APIO) is also used as an input for the RS sub-module (see Fig. 3). This input identifies which rules are allowed to be considered by the FC module, with respect to data access rights. For example, if we consider the case where one system is based uniquely on *Onto1*, such system could choose not to share the data from *onto1:q2*. In this case, and if query *Q'* is addressed by another system uniquely based on *Onto2*, RS does not select $swrl_3$. This is justified by the fact that the system addressing query *Q'* is not allowed to have access to *onto1:q2* data values. Finally, our system outputs the eligible set of rules for interoperate the federated ontologies. This is called the Activated Rule Set (ARS). We therefore execute the initial query over the data contained in the KB and considering only the rules present in the ARS set.

Therefore, the data present in the KB, for the considered ontologies, is automatically restructured according to the rules in the ARS. The SWRL engine associated with the OWL reasoner processes these rules (see Fig. 1). This mechanism allows us to handle different schemas, thus addressing schema interoperability issue.

5 Results

The implementation of the FOWLA architecture comes with several advantages for interoperating several ontologies: (1) it allows inferring new ontology alignments; (2) it allows avoiding data redundancy; (3) it allows modularizing the maintainability, thought preserving the autonomy among ontology-based systems, (4) it allows querying with vocabulary terms issued from different ontologies and (5) it allows improving query execution time.

For demonstrating advantage (1), let us suppose four ontologies (*A*, *B*, *C* and *D*) and the alignments described as *FD(A, B)*, *FD(B, C)*, *FD(C, D)*, *FD(A, D)* and illustrated in Fig. 4. *FD(X, Y)* represents the contents of the Federal Descriptor module between ontologies *X* and *Y*. For each of the considered ontologies, we define an Interoperable Schema (IS) as the sub-graph of this ontology that contains all schemas necessary for exchanging data with another ontology. The IS sub-graph is composed of correspondent classes and properties between the two considered ontologies. We note *IS(X,Y)* the interoperable schema of ontology *X* for ontology *Y*. Allowing the inference of data from one ontology to others (i.e. rule-based interoperability) reduces the number of alignments (i.e. rules) that we need to conceive and, in some cases, the conception of the whole *FD(X,Y)* component between two ontologies.

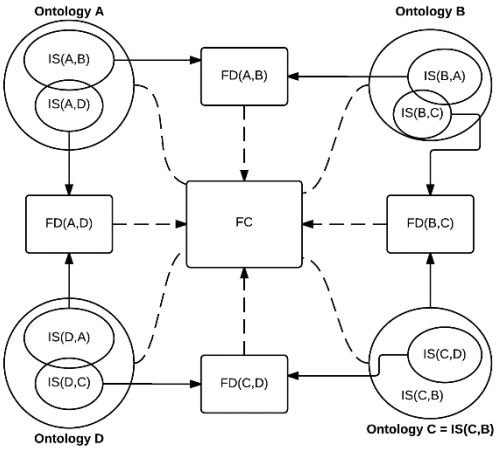

Fig. 4. Case study of four ontologies implementing FOWLA

For example, let us consider that ontology *C* is already populated and *IS(C,B)* is equivalent to the whole ontology *C*. Therefore, with the complete *FD(B,C)* component defined, all data of *C* is accessible by querying ontology B. Moreover, the definition of *FD(A,B)* allows retrieving data from the so populated ontology *C* by querying *A* (more precisely the data modelled using *IS(B,A)* ∩ *IS(B,C)*). In this case, we do not need to define *FD(C,A)* because *A* and *C* are indirectly aligned and totally integrated by the FC using *FD(A,B)* and *FD(B,C)*. For further explanation, let us suppose that *FD(A,B)* and *FD(B,C)* respectively contain $swrl_5$ and $swrl_6$.

SWRL rules	$swrl_5$: *ontoA:Aa(?x)* → *ontoB:Bb(?x)* $swrl_6$: *ontoB:Bb(?x)* → *ontoC:Cc(?x)*	(5.1)
SPARQL Query executed	*Q'' : SELECT ?x WHERE { ?x rdf:type ontoC:Cc. }*	(5.2)
Inferred fact(s)	*ontoA:Aa(?x)* → *ontoC:Cc(?x)*	(5.3)

Considering a rule engine for interpreting those rules and the SPARQL query *Q''* (5.2), the rule engine infers the transitive relation: *ontoA:Aa(?x)* → *ontoC:Cc(?x)* (5.3). Then, the query *Q''* retrieves all instances which belong to *ontoA:Aa*, *ontoB:Bb* and *ontoC:Cc* classes. Therefore, we do not need to define the alignment (5.3) in *FD(A,C)*, because it is inferred by the FC at query execution time. With this example, we demonstrate that, for a given query, query rewriting approaches such as 1 or **Erreur ! Source du renvoi introuvable.** (as presented in Section 3) do not retrieve all pertaining results. Compared to 1 or **Erreur ! Source du renvoi introuvable.**, FOWLA allows simplifying the ontology matching process when applied to several ontologies adressing similar knowledge domains (i.e. schemas which aim at modelling similar kinds of information).

The use of backward-chaining techniques by the FC module allows the federated ontologies to not replicate interoperability data among them. This is because rule inference is performed at query execution and doesn't need materializing the same shareable data. Consequently, the data modelled with one ontology is available (inferred) for others by applying rule-based alignments. Then, once data changes in the source ontology, FC infers the new modified data for the target ontologies. The advantage (2) is illustrated in one previous example in the section 4 and illustrated in Fig. 2.

For exemplifying the advantage (3), let us suppose a modification in the ontology schema *A*, more precisely in the sub-graph *IS(A,D) – { IS(A,B) ∩ IS(A,D) }* from our previous case study represented in Fig. 4. In this case, the sole components which have to evolve are *FD(A,D)* and *IS(D,A)*. Doing so, we preserve the full interoperability among *A*, *D* and the other ontologies. The other FDs and ontologies remain unchanged. Note that only the system based on ontology *A* has to evolve which is not the case for systems based on ontologies *B*, *C* and *D*. This is explained by the fact that the underlying schemas for ontologies *B*, *C* and *D* have not been modified. Therefore, besides implementing ontology interoperability, we also preserve each systems' autonomy.

For evaluating FOWLA and justifying the advantage (4) and (5), we consider two OWL-Lite ontologies (*Onto1* and *Onto2*), for which we define the *FD(Onto1,Onto2)*. Table 1 lists some characteristics of these ontologies. The FLS (i.e. alignments) between these two ontologies comprises 474 SWRL rules which were manually created (most of them are complex alignments, involving numerous predicates).

Table 1. Characteristics of *Onto1* and *Onto2*

OWL entities	**Onto1**	**Onto2**
Classes	30	802
Object properties	32	1292
Data properties	125	247
Inverse properties	7	115
Triples in the TBox	2212	9978
DL expressivity	$ALCHIF^{(D)}$	$ALUIF^{(D)}$

For our experiments, we have used a 2.2.1 Stardog triple store [16] which played the role of the server and was encapsulated in a virtual machine with the following configuration: one microprocessor Intel Xeon CPU E5-2430 at 2.2GHz with 2 cores out of 6, 8GB of DDR3 RAM memory and the "Java Heap" size for the Java Virtual Machine set to 6GB. We chose Stardog because it provides an OWL reasoner associated to a SWRL engine and it is based on backward-chaining reasoning [XX]. Indeed, our RS sub-module only aims at hybrid or backward-chaining reasoning approaches [XX] (as in forward-chaining reasoning [XX] are materialized all facts entailed). So, Stardog's reasoner and the RS sub-module constitute the Federal Controller (FC) module. The considered triple store contains 4 repositories. Each repository stores the *Onto1* and *Onto2* knowledge base (*Onto1's* TBox and ABox and *Onto2's* TBox and ABox). We name those repositories KB1, KB2, KB3 and KB4. For the considered example, each respository's ABox contains more than one million triples. For testing purposes and for

each repository, we have implemented sets of interoperability rules with different cardinalities. Table 2 lists the considered set of rules along with their characteristics.

Table 2. Rules implemented for each knowledge base (KB)

	Number of rules	**Characteristics**
KB1	474	All the rules contained in the FLS (all the rules forming the alignment between *Onto1* and *Onto2*)
KB2	266	All subsumption rules along with all the rules that have elements from *Onto1* in their head
KB3	178	All rules from KB2 minus some of the rules that have elements from *Onto1* in their head (we aimed at reducing the data inferred)
KB4	variable	All the rules contained in the Activated Rule Set (ARS) conceived by the RS.

For these tests, we have used a client machine with the following configuration: one microprocessor Intel Core CPU I5-3470 at 3.2GHz with 4 cores, 4GB of DDR3 RAM memory at 800MHz and a "Java Heap" size set to 1GB. The client machine executes the RS sub-module presented in this paper.

Table 3 shows the queries used in our experiments. For the sake of simplicity, we note *Cij* the class *Cj* in ontology *Ontoi*, respectively p_{kl} the p_l property in ontology *Ontok*, where $i, j, k, l \in \mathbb{N}^*$. Each one was executed 30 times over the knowledge bases KB1, KB2, KB3 and KB4. Table 4 shows the results we obtained. The capability of retrieving results for query Q2 and Q3 demonstrate that our approach allows querying federated ontologies, treating them as one unique ontology. Because, we can write queries using at same time terms from *Onto1* and *Onto2*. So, this justifies the advantage (4).

Table 3. List of queries addressed over the considered knowledge bases

Query name	**SPARQL Query**
Q1	SELECT ?x ?y WHERE { ?x onto1:p_{11} ?y . }
Q2	SELECT ?x ?y WHERE { ?x a onto2:C_{21} . ?x onto1:p_{11} ?y . }
Q3	SELECT ?x ?u WHERE { ?x a onto1:C_{11} . ?y a onto2:C_{22} . ?x onto1:p_{12} ?y . ?y onto1:p_{11} ?x . }

In Table 4, the "**#RuleSet**" column displays the number of rules as implemented over the considered KB, at query execution time. The "**#Results**" column shows the number of tuples that were retrieved as a result for the considered query (e.g. {?x,?y} for Q1). In Table 4, "-" means that no results were retrieved for the considered query after more than 1 minute waiting time. The reason relies in the fact that the memory heap size (6GB) for the Java Virtual Machine is exceeded.

Table 4. Query Performance Evaluation

Query	Knowledge base	Mean execution time (in seconds)	Standard Deviation (σ)	#RuleSet	#Results
Q1	KB1	-	-	474	0
	KB2	-	-	266	0
	KB3	9.25	12.21	178	1683
	KB4	**2.23**	**1.78**	**16**	**38318**
Q2	KB1	-	-	474	0
	KB2	-	-	266	0
	KB3	32.99	0.75	178	74
	KB4	**0.16**	**0.04**	**2**	**74**
Q3	KB1	-	-	474	0
	KB2	-	-	266	0
	KB3	71.62	0.95	178	0
	KB4	**0.88**	**0.43**	**5**	**9**

When analyzing results, we can see that, for answering query Q1, our methodology has selected 16 rules from the initial set of 474 rules (i.e. the FLS). The results also indicate that without our approach no result is retrieved as long as the entire FLS is considered, due to memory overload and after about 3 minutes of query execution over KB1. When executed over KB2, Q1 evidences that reducing the cardinality of the initial rule set to 266 does not prevent memory overload. When executing Q1 over KB3 (which implements less than 40% of FLS rules), Q1 returns less than 5% of all expected results. This is explained by the fact that several of the relevant rules for Q1 were removed when conceiving our test knowledge bases. Moreover, when compared to Q1 over KB4, Q1 over KB3 has a duration 4 times greater and retrieves 22 times less results. Indeed, KB4 implements the only rules contained in the FLS, so the results of Q1 executed over KB4 represent the gain (in terms of query execution time and results retrieved) achieved by implementing our approach. When applied to Q2, the RS sub-module takes into account the domain restriction defined within Q2 (e.g. ?x a onto2:C_{21}). It then creates an FLS set containing only 2 rules instead of 16, as it was previously the case for Q1 (which did not had any domain information for the property *onto1:*p_{11}). For the above considered tests, the mean query execution times have been considerably reduced. The standard deviation for the query response time is much lower using our RS sub-module, meaning the query response time is more centralized onto the mean.

6 Conclusion and Future Work

In this work, we focus on schema-level heterogeneity however implementing FOWLA can reduce also data-level heterogeneity if the Federated Logic Schema is defined using the SWRL built-ins (e.g. *swrlb* vocabulary [XX]).

Acknowledgement

This work has been financed by the French company ACTIVe3D[1] and supported by the Burgundy Regional Council[2].

7 References

1. Konstantinos Makris, Nektarios Gioldasis, Nikos Bikakis, and Stavros Christodoulakis. Ontology mapping and sparql rewriting for querying federated rdf data sources. In Proceedings of the 2010 International Conference on On the Move to Meaningful Internet Systems: Part II, OTM'10, pages 1108–1117, Berlin, Heidelberg, 2010. Springer-Verlag.
2. J. Euzenat, P. Shvaiko. Ontology Matching, Second Edition, Springer-Verlag Berlin Heidelberg, Germany, DOI: 10.1007/978-3-642-38721-0, 2013
3. Konstantinos Makris, Nektarios Gioldasis, Nikos Bikakis, and Stavros Christodoulakis. Sparql rewriting for query mediation over mapped ontologies. http://www.music.tuc.gr/reports/SPARQLREWRITING.PDF, 2010. [Online; accessed 2015-02-26].
4. Gianluca Correndo, Manuel Salvadores, Ian Millard, Hugh Glaser, and Nigel Shadbolt. Sparql query rewriting for implementing data integration over linked data. In Proceedings of the 2010 EDBT/ICDT Workshops, EDBT '10, pages 4:1–4:11, New York, NY, USA, 2010. ACM
5. Dizza Beimel and Mor Peleg. Using owl and swrl to represent and reason with situation-based access control policies. Data & Knowledge Engineering, 70(6):596–615, 2011.
6. Boris Motik, Ulrike Sattler, Rudi Studer, Query Answering for OWL-DL with Rules, in Journal of Web Semantics (Elsevier) 3 (1): 41–60. http://www.cs.ox.ac.uk/boris.motik/pubs/mss05query-journal.pdf [Online; accessed 2015-02-26].
7. Shvaiko Pavel and Jerome Euzenat. Ontology matching: State of the art and future challenges. IEEE Trans. on Knowl. and Data Eng., 25(1):158–176, January 2013.
8. Sheth A. P. & Larson J. A.: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys, Volume 22, N° 3 (1990)
9. The World Wide Web Consortium (W3C), `http://www.w3.org/`
10. RuleML, `http://www.ruleml.org`
11. Horrocks, I., Patel-Schneider, P. F., Bechhofer, S. & Tsarkov, D. (2005). OWL rules: A proposal and prototype implementation. Journal of Web Semantics, 3(1):23-40
12. Gehre, A., Katranuschkov, P., Wix, J., & Beetz, J.: Interoperability of Virtual Organizations on a Complex Semantic Grid. Dresden: InteliGrid (2006)
13. Allemang, D. & Hendler, J.: Semantic Web for the Working Ontologist. (M. Kaufmann, Ed.) San Francisco: In Dean Allemang and James Hendler (2008)

[1] http://www.active3d.net/fr/

[2] http://www.region-bourgogne.fr/

14. Dibley, M. J.: An Intelligent System for Facility Management. Wales, UK: Cardiff School of Engineering, Cardiff University (2011)
15. Tim Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y. & Hendler, J. N3Logic: A logical framework for the World Wide Web. Theory and Practice of Logic Programming, 8, pp 249-269 (2008).
16. Stardog version 2.1.3, http://stardog.com/